Kernel Methods and Measures for Classification with Transparency, Interpretability and Accuracy in Health Care

by

André M. Carrington

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering,
University of Waterloo

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the committee is by majority vote.

External Examiner:	Dr. Douglas Manuel Professor Clinician Scientist
Supervisor(s):	Dr. Paul Fieguth Professor Department Chair
	Dr. Helen Chen Assistant Professor
Internal Member(s):	Dr. David Clausi Professor
	Dr. John Zelek Associate Professor
Internal-External Member:	Dr. Jose Arocha Associate Professor

Author's declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 6 of this thesis is an adaptation of an accepted paper that was co-authored by myself and my supervisors, Dr. Paul Fieguth and Dr. Helen Chen. I developed and documented the concepts, measures, methodology, experiments, conclusions and related work. My co-authors reviewed the paper and provided feedback which removed one type of measure, recommended one additional experiment not included due to time constraints, and edited content in general.

Section 4.7.1 of Chapter 4 contains content adapted from portions of a published paper that was co-authored by myself and my supervisors, Dr. Paul Fieguth and Dr. Helen Chen. I developed and documented the kernel, proof, methodology, experiments, conclusions and related work. My co-authors reviewed the paper and edited content in general.

Abstract

Support vector machines (SVM) are a popular method in machine learning. They learn from data about a subject, for example, lung tumors in a set of patients, to classify new data, such as, a new patient's tumor. The new tumor is classified as either cancerous or benign, depending on how similar it is to the tumors of other patients in those two classes—where similarity is judged by a kernel.

The adoption and use of support vector machines in health care, however, is inhibited by a perceived and actual lack of rationale, understanding and transparency for how they work and how to interpret information and results from them. For example, a user must select the kernel, or similarity function, to be used, and there are many kernels to choose from but little to no useful guidance on choosing one.

The primary goal of this thesis is to create accurate, transparent and interpretable kernels with rationale to select them for classification in health care using SVM—and to do so within a theoretical framework that advances rationale, understanding and transparency for kernel/model selection with atomic data types. The kernels and framework necessarily co-exist.

The secondary goal of this thesis is to quantitatively measure model interpretability for kernel/model selection and identify the types of interpretable information which are available from different models for interpretation.

Testing my framework and transparent kernels with empirical data I achieve classification accuracy that is better than or equivalent to the Gaussian RBF kernels. I also validate some of the model interpretability measures I propose.

Acknowledgements

I wish to thank my supervisors Professor Paul Fieguth and Professor Helen Chen for their astute, timely and constant advice and support. I wish to thank all of the funding agencies that supported my research, including the Ontario Centres for Excellence, Natural Sciences and Engineering Research Council (NSERC) of Canada and the University of Waterloo Faculty of Engineering. Finally, I wish to thank all the significant people in my life, past and present, among my family and friends, committee members, previous research supervisors and colleagues at the University of Waterloo.

Table of Contents

List of Figures

List of Tables

List of abbreviations

Table 1: Abbreviations for kernels

Common kernels	
RBF	Gaussian RBF kernel
Lin	Linear kernel
Poly	Polynomial kernel
Sig	Sigmoid kernel

Uncommon kernels	
Del	Delta kernel
Exp	Exponential kernel
HE	Hamming Euclidean hybrid kernel
Ham	Hamming kernel
IMQ	Inverse multiquadric kernel
Log	Logarithmic kernel
Pwr	Power distance kernel
tRBF	Truncated Gaussian RBF kernel

Proposed kernels	
BBN	Bayesian binormal kernel
BD	Bayesian density kernel
DC	Delta composite kernel
JPBN	Joint probability binormal kernel
JPD	Joint probability density kernel
GD	Gaussian derivative kernel
ISVgc	Insensitive sigmoid variant kernel (Gaussian) constrained
ISVhc	Insensitive sigmoid variant kernel (hyperbolic tangent) constrained
MSig	Mercer sigmoid kernel
SigN	Normalized sigmoid kernel
OMBcs	Orthant and Mercer sigmoid Bayesian binormal composite sum kernel
OLin	Orthant linear kernel
OISVgc	Orthant insensitive sigmoid variant kernel (Gaussian) constrained
OISVgc	Orthant insensitive sigmoid variant kernel (hyperbolic tangent) constrained
OSig	Orthant sigmoid kernel
Pos	Positive match kernel

List of symbols

The following tables list symbols for (1) vectors, matrices and algebra, (2) statistics and information theory, (3) classification and kernel methods and (4) accuracy and interpretability.

Table 2: Symbols for vectors, matrices and algebra

Syntax	Definition
i, j	General indices
x, z	Random variables, e.g., features
$\underline{x}$, $\underline{z}$	Column vectors, e.g., instances
x_i	The ith element of vector $\underline{x}$
$\underline{x}_i$	The ith vector in a sequence
$\mathbb{R}^n$	An n-dimensional space of reals
$\mathbb{Z}^n$	An n-dimensional space of integers
$\mathbb{B}^n$	An n-dimensional space of binary values
$\mathbb{M}_k^n$	An n-dimensional space of nominals with k levels
$\mathbb{D}^n$	An n-dimensional space of dates
$\underline{1}$	A vector with the value 1 in each element
$\underline{\mathrm{v}}$	An eigenvector
$\overline{b}$	The logical compliment of a binary value b
X, Y	Matrices, e.g., data matrix X, target matrix $Y = \underline{y}\,\underline{y}^{\mathrm{T}}$
X^T	A matrix transposed
X^{-1}	A matrix inverse
$\mathbb{R}^{n \times k}$	The space of matrices of reals in n rows and k columns
I	The identity matrix
$\|\underline{x}\|$, $\|\underline{x}\|_1$,	Vector norms: 2-norm (default), 1-norm
$\|\underline{x}\|_\infty$, $\|\underline{x}\|_p$	Vector norms: supremum and p-norm
$\|X\|_F$, $\|X\|_1$	Matrix norms: Frobenius norm (default), 1-norm
$\|X\|_2$, $\|X\|_\infty$	Matrix norms: 2-norm and supremum
$\|X\|, \det(X)$	The determinant of X
$\mathrm{tr}(X)$	The trace of X
$\kappa(X)$	The condition of X
$\mathrm{diag}(X)$	A vector from the diagonal of X
$\mathrm{Diag}(\underline{x})$	A diagonal matrix from elements of vector $\underline{x}$
$\mathrm{rank}(X)$	The rank of X
x_{ji}	The element at row j and column i of matrix X
$f(\cdot)$, $g(\cdot)$	Functions with a scalar output
$\underline{f}(\cdot)$, $\underline{g}(\cdot)$	Functions with a vector output

Table 3: Symbols in statistics and information theory

Syntax	Definition
$\Pr(\cdot)$	Probability
$p(x)$	Probability density function (pdf) of x
$p_{\mathrm{m}}(b)$	Probability mass function of discrete random variable b
$p(x\|y)$	Conditional probability density function of x given y
$\sim$	Is distributed as ...
$\underline{x} \sim \mathcal{N}(\underline{\mu}, P)$	$\underline{x}$ is a vector of normal distributions, or Gaussians, with means $\underline{\mu}$ and covariance P.
$\underline{x} \sim (\underline{\mu}, P)$	$\underline{x}$ has unspecified distributions with means $\underline{\mu}$ and covariance P
$\underline{x} \sim P$	$\underline{x}$ has unspecified distributions and unspecified means with covariance P
$\mathrm{B}(l, p)$	Binomial distribution with probability p of each event, for l samples with replacement, i.e., for binary features
$\mathrm{Mult}_k(l, \underline{p})$	Multinomial distribution with k classes and vector of class probabilities $\underline{p}$, for l samples with replacement, i.e., for nominal features
$\underline{x} \in \mathrm{B}^n$	$\underline{x}$ is a vector with binary values from unspecified Binomial distributions
$\underline{x} \in \mathrm{Mult}^n$	$\underline{x}$ is a vector of nominals from unspecified Multinomial distributions
$H(b)$	Information entropy of discrete random variable b
$H(b\|y)$	Conditional entropy of b given y.
$I(a; b)$	Mutual information of a and b

Table 4: Symbols in classification and kernel methods

Symbol	Definition
$\mathcal{X}$	Input space, e.g., $\mathcal{X} \in \mathbb{R}^n$
$\underline{x}$, $\underline{z} \in \mathcal{X}$	An instance (or sample or patient) in the input space
$\underline{x}_i$, $\underline{z}_i \in \mathcal{X}$	The ith instance (or sample or patient) in the input space.
x_j	The jth feature of an instance in the input space.
$x_{i,j}$	The jth feature of the ith instance in the input space.
n	Number of features in each instance
N	Number of instances in the input space
X	The data matrix (or feature matrix) with instances as rows, $X \equiv \begin{bmatrix} \underline{x}_1 & \underline{x}_2 & \cdots & \underline{x}_N \end{bmatrix}^{\mathrm{T}}$ and feature vectors as columns, $X \equiv \begin{bmatrix} \underline{q}_1 & \underline{q}_2 & \cdots & \underline{q}_n \end{bmatrix}$, e.g., $X \in \mathbb{R}^{N \times n}$
$\underline{q}_i$	The ith feature vector as a column in the data matrix
$\underline{\phi}$	Feature map or basis function $\underline{\phi} : \mathcal{X} \to \mathcal{F}$, i.e., $\underline{x}^* = \underline{\phi}(\underline{x})$
$\mathcal{F}$	The feature space, e.g., $\mathcal{F} \in \mathbb{R}^{n_\mathcal{F}}$
$\underline{x}_i^* \in \mathcal{F}$	Transformed instance in the feature space
$n_\mathcal{F}$ or n^*	Number of transformed features in the feature space
$\underline{q}_i^*$	The ith transformed feature vector.
k, $k(\underline{x}, \underline{z})$	Kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k(\underline{x}, \underline{z}) \equiv \left\langle \underline{\phi}(\underline{x}), \underline{\phi}(\underline{z}) \right\rangle$, $\underline{\phi}$ may or may not be explicitly known, but must exist for a Mercer/p.d. kernel
$d(\underline{x}, \underline{z})$	A distance function (between scalars or vectors)
$s(\underline{x}, \underline{z})$	A similarity function (betweeen scalars or vectors)
$\underline{\theta}$	A set of hyperparameters for a kernel or function, e.g., $\underline{\theta}_k$
$\alpha, \beta, \gamma, \theta, \sigma, \tau, \psi$	A hyperparameter or model weight (specified or learned).
G	The Gram matrix: a square matrix computed on training data. $G = \left\{ k\left(\underline{x}_i, \underline{x}_j\right) \right\} \forall\, i, j$
K	The kernel matrix, computed pairwise between training and test data.
$h : \mathcal{X} \to \mathcal{Y}$	A classifier
$\mathcal{Y}$	The target space, e.g., $\mathcal{Y} \in \{-1, +1\}$ (binary) or $\mathcal{Y} \in \{\text{none, in situ, invasive, metastasis}\}$ (multiclass) or $\mathcal{Y} \in \mathbb{R}$ (continuous)
y, $y_i \in \mathcal{Y}$	Known/given target or outcome
$\hat{y}$, $\hat{y}_i \in \mathcal{Y}$	Estimated (or decided or predicted) target or outcome
$\hat{u}$, $\hat{u}_i \in \mathbb{R}$	Estimated underlying target, e.g., classification score or probability
$\underline{y}$	Target vector $\underline{y} \equiv \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}^{\mathrm{T}}$
Y	Target matrix $Y \equiv \underline{y}\,\underline{y}^{\mathrm{T}}$

Table 5: Symbols for accuracy and interpretability

Symbol	Definition
a	**Posterior** accuracy
$\check{T}_{\partial}$	**Prior** model transparency
U_{sv}	**Posterior** inherent model interpretability from **support vectors**
$U_{H\mathrm{st}}$	**Posterior** inherent model interpretability from simplicity of sensitivity with **Sturges** binning
U^*_{rdeT}	**Initial** inherent model interpretability from the **relevant dimensionality estimate two-component** model
U^*_{rdeL}	**Initial** inherent model interpretability from the relevant **relevant dimensionality estimate leave-one-out** model
$U^*_{H\mathrm{st}}$	**Initial** inherent model interpretability from simplicity of sensitivity with **Sturges** binning
$U_{H\mathrm{fd}}$	**Posterior** inherent model interpretability from simplicity of sensitivity with **Freedman-Diaconis** binning
$U_{H\mathrm{sc}}$	**Posterior** inherent model interpretability from simplicity of sensitivity with **Scott** binning
$U^*_{H\mathrm{fd}}$	**Initial** inherent model interpretability from simplicity of sensitivity with **Freedman-Diaconis** binning
$U^*_{H\mathrm{sc}}$	**Initial** inherent model interpretability from simplicity of sensitivity with **Scott** binning

Chapter 1

Introduction

In classification studies/problems the task is to estimate or decide a binary outcome, such as, if a patient has heart disease or not based on a comparison of their data to those of past patients with known outcomes.

In health care, logistic regression and decision trees are commonly used methods for this purpose with a growing demand for more advanced methods such as machine learning [250]. However, machine learning (ML) methods are perceived as black boxes [169] which limits understanding and explanation as required in health care.

This work focuses on support vector machines (SVM), as a popular and important ML method [293] and a mainstay among state-of-the-art methods [47, 198, 301]. To understand SVM [292] one must also understand one of its key components: a kernel.

A kernel (depicted in Figure 1.1 on page 2) is a function or model that takes two instances of data as inputs (defined in Figure 1.2 on page 3), e.g., two patient health records, or two (genomic) microarrays, and outputs a real-valued number that represents their **similarity**. There are different kernels according to different concepts of similarity—e.g., patients observations that are close to each other (**proximity**) versus patients observations that vary together in the same direction (**covariance** similar to correlation[1]). Proximity and covariance correspond to two main types of kernels—stationary kernels and dot product kernels—which are the subject of later discussion.

[1]Correlation describes the association between multiple observations of a single feature in one patient to multiple observations of that same feature in another patient. In this case, I am interested in how a single observation in each feature of a patient, varies together (covaries) with a single observation in each feature of another patient. Formally it is a sum of one-sample covariances in each feature. In math it is called a dot product and in geometry it is called a scalar projection.

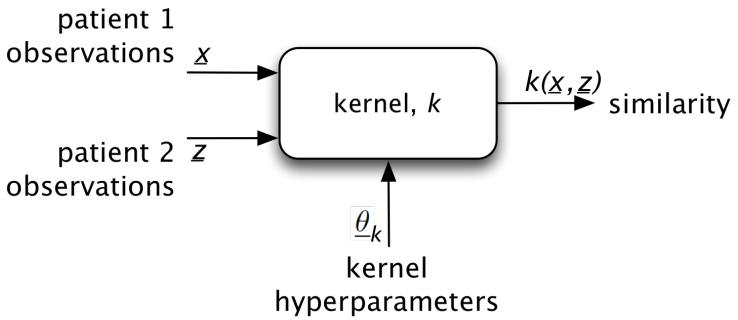

Figure 1.1: Kernel inputs, outputs and hyperparameters. Note: kernel is a popular term with at least 10 other definitions (see Chapter 7.1 Glossary for disambiguation).

The choice of kernel can adversely affect classification accuracy [18, 24, 36] as well as model interpretability. Hence model selection is an important step in the ML process (Figure 1.4 on page 5) (Section 2.1).

The word kernel refers to at least eleven different concepts, most of which are not related. I refer to two related concepts of kernels. I, **primarily**, and **by default**, refer to kernels as those used in SVM and other **kernel methods** (Section 2.7)—and these are a **subset** of the second concept: kernels used in **mathematics and statistics** for **kernel density estimation** (KDE) to estimate a probability density function or perform **smoothing** of time-series data. I refer to the latter concept by the term: mathematical kernels.

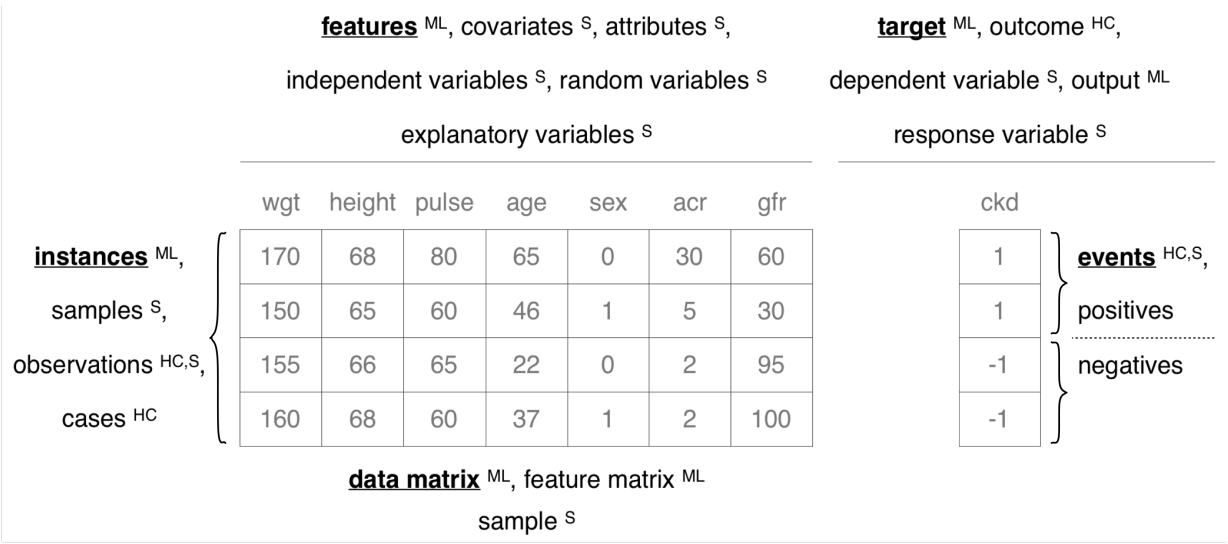

Figure 1.2: The machine learning (ML) terms I use in bold underlined font differ from key terms used in other domains (normal font) such as statistics (S) or health care (HC). Text in light gray font are examples as visual context for the terms. We additionally note that features may be further identified as confounders or influencers in the statistical or health care domains.

1.1 The problem and elements of my thesis

In this section, I introduce the study/problem in relation to elements of my thesis—deferring the complete study/problem definition to Chapter 3.

SVM is said to be an black box that is difficult to understand [15, 58]. Hence SVM does not readily facilitate the need for clinicians to be able to trust, use, explain, justify or advocate their results and methods to colleagues and patients [21, 106]. However, not all kernels are opaque black boxes (Section 4.4) and my work focuses on defining and selecting models (or kernels) which are **transparent** and **inherently interpretable** for doctors (Figure 1.3 on page 4).

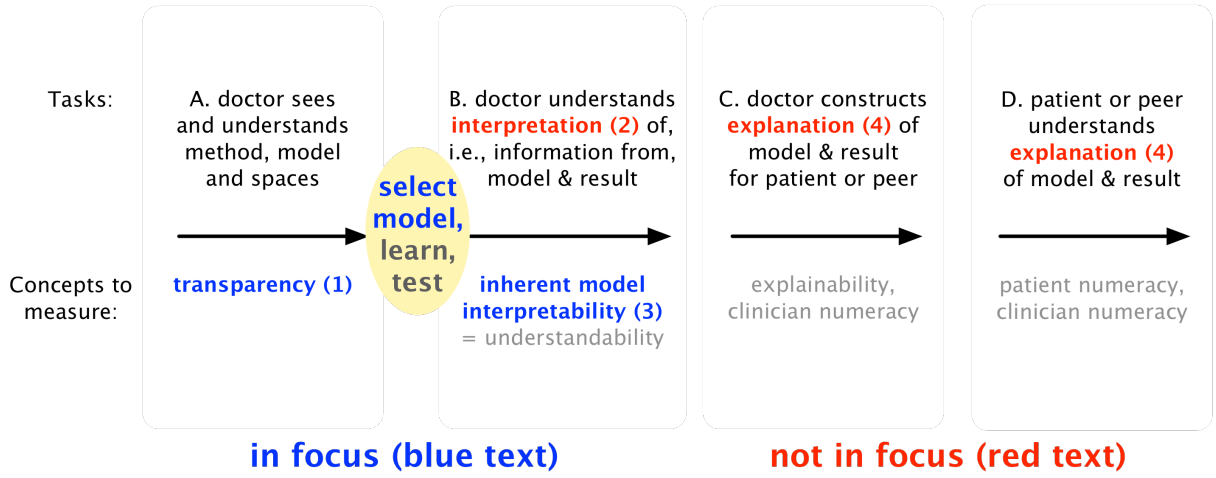

Figure 1.3: Model selection and measures of **transparency** and **inherent model interpretability** are foci for my thesis, rather than interpretations, which are addressed to a lesser extent, and explanations which are not addressed.

Montavon *et al.* [192] also differentiate **interpretations** from **explanations**, where an interpretation is information in a form that can be perceived and elementally understood by a person (doctor in my case) versus an explanation, which according to Miller, involves selecting information, reasoning and communicating with a recipient [188].

I define **inherent model interpretability**, as independent of a person's abilities and background, per my definition in [46] (Section 6.2). It is measured for several different models—prior, initial and posterior models—at different points in the machine learning process (Figure 1.4).

A **lack of guidance** in the literature about SVM is partially responsible for its perception as a black box. This is confirmed by my review (Section 3.1) of 22 papers applying support vector machines to classification in health care: the vast majority of papers provide little to **no rationale** for the kernels (hence models) they use, because they lack guidance and they fail to provide guidance themselves.

Guidance from **automated and manual kernel selection** methods [5, 194, 276, 277] seek to apply a fixed/learned rule to new data sets, however I assert that such rules are sensitive to how the study/problem is setup, i.e., the fixed set of kernels considered, the data sets considered, the way the data are pre-processed, etc. Including or excluding a single data set can change or bias the rule. For example, in several of those methods, decision tree rules are learned to select kernels—however decision tree rules are **unstable** [205]. Also, the rules only apply to the kernels included in those papers which are kept few so that the rules are fairly simple and general.

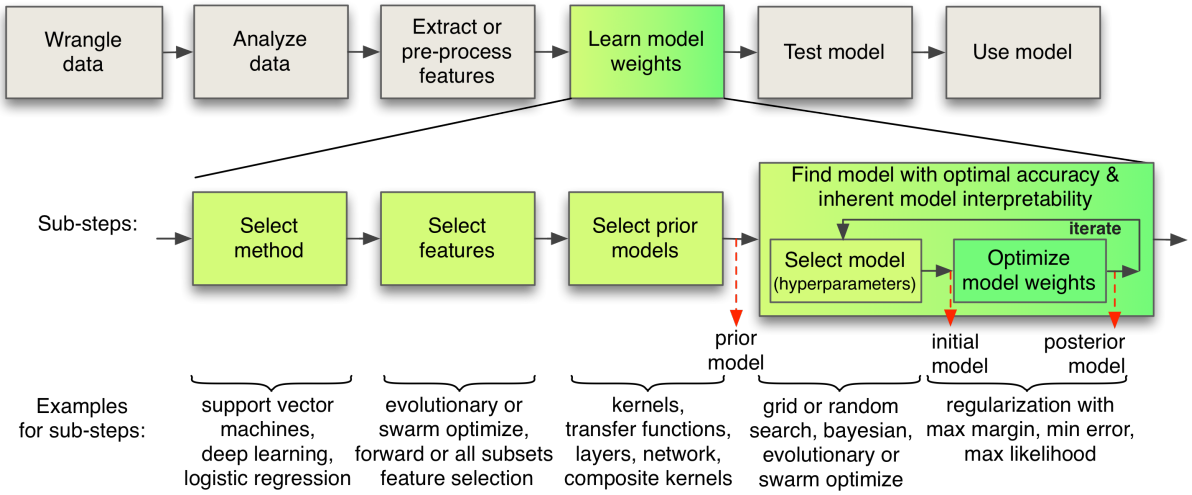

Figure 1.4: Model interpretability can be measured at several different points (dashed red arrows) in the process of machine/statistical learning (partially derived from [138]). Note: some steps may not apply to some methods and models.

Of greater importance is **understanding** how kernels relate to data and/or data types—but the only understanding or lesson I find [276] pertains to the **local versus global** effect of kernels (3.4.2) which I already know from other/earlier sources [161, 288, 227]. Without understanding, the rules like those in the kernel selection papers are black boxes too—which does not help matters. This work provides a **framework** to understand and trace kernels to data types, distributions and requirements for similarity, classification and transparency, via **kernel data modeling**.

A framework, however, does not provide kernels themselves, hence I also derive and design **kernels** (Section 4.5) from the class of explicit Mercer kernels (Section 4.4). As a kernel without SVM, my proposed kernels are additive, while in the context of SVM, they are generalized additive models (GAM) [172] — and they meet requirements for transparency from literature [168, 172][2]. My transparent kernels are competitive in accuracy [45, 295] (Section A.9) with the Gaussian RBF kernel and can provide **better views of results for transparency** (Section 5.7). They also differ from existing methods such as feature shaping (4.4.4).

Returning to the point about SVM's opacity—there are some concerns when SVM is used in its most popular configuration, i.e., with a Gaussian RBF kernel, because one **cannot** easily and accurately determine the **contribution of one feature** independent of others or in interaction with others (Section 5.6). Also, given an instance classified as negative, one **cannot easily tell what changes to the instance would tip the classification**

[2]I acknowledge the original authors of generalized additive models as well [116, 115] but note that I do not adopt their full technique.

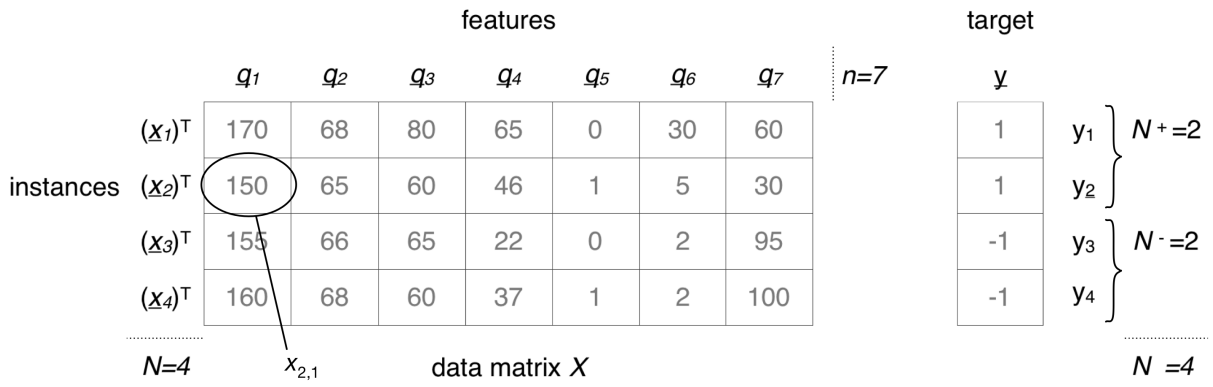

Figure 1.5: A visual glossary of notation, i.e., symbols in black font. Text in light gray font are examples as visual context for the terms. $x_{2,1}$ refers to the value of the 2^{nd} instance's 1^{st} feature.

to positive—i.e., what are the set of closest points on the class boundary? Or what changes would lead to a stable change in classification that reduces uncertainty? Computing and visualizing the answers to these questions is possible in the **feature space**—however for a **Gaussian RBF kernel** the feature space is infinite and therefore not computable nor interpretable, whereas it is computable with the kernels that I propose. The feature space explains how SVM works and allows us to **verify it is working as expected for a specific kernel with specific data.** It can potentially allow us to interpret and justify the class boundary, but it is only available with the class of kernels I propose.

Alternatively authors [15, 209] in the literature provide views or measures of the input space which provide some insight, but when these views are used with kernels like the Gaussian RBF kernel they are limited relative to kernels I propose. My kernels allow better interpretation of the input space (Section 5.7).

This opaque and infinite nature of the Gaussian RBF kernel is simultaneously its strength: it implicitly and automatically uses an infinite number of transformed features in the feature space—i.e., an infinite number of powers of each feature, e.g., $x_{11},\ x_{11}^2,\ x_{11}^3, \ldots$ (see Figure 1.5 as a quick reference on notation) as well as an infinite number of interaction terms in higher orders, e.g., $x_{11}x_{12},\ x_{11}^2x_{12},\ x_{11}^3x_{12}^2$ [55]. This provides a lot of flexibility to fit data.

If I use **explicit Mercer kernels** instead, to view the feature space in which SVM and the kernel act, then opacity disappears and the only challenge is in interpreting and visualizing a multidimensional space. This class of kernel includes the truncated version of the Gaussian RBF kernel [55, 278].

There is also a question about the **rationale** for a Gaussian RBF kernel. That is, the

kernel begins with a Euclidean distance $\|\underline{x} - \underline{z}\|_2$ between the readings of two patients—this is the concept of distance that I apply in real life, which makes sense. The kernel then applies a simplified[3] Gaussian or normal curve, $f(\underline{x}) = \exp(-\gamma \cdot \underline{x}^2)$, $\gamma = \frac{1}{2\sigma^2}$, to that distance to arrive at the kernel, $k_{\text{RBF}}(\underline{x}, \underline{z}) = \exp\left(-\gamma \cdot \|\underline{x} - \underline{z}\|_2^2\right)$. Why? Let us examine the underlying assumptions.

I know that many phenomena of a single variable exhibit Gaussian or normal behaviour, e.g., errors in a measurement, the mean of any statistic, etc. A multivariate Gaussian, however, is an ideal assumed by researchers more often than it is encountered in real life. In classification, if both classes are multivariate Gaussian, then the difference between two instances, from the same class or different classes, is multivariate Gaussian—and since that difference is the input to a Gaussian RBF kernel (as a stationary kernel), this seems to be the reason for applying a Gaussian RBF kernel.

However, it is unlikely that either of the classes are multivariate Gaussian. By my measurements most of the data sets I use are not Gaussian in each class for most features, in the marginal, separately, much less multivariate Gaussian which is a tighter requirement of the joint density. Even if both classes are multivariate Gaussian, or it is a reasonable approximation, there is another problem. The Gaussian RBF kernel is a multivariate Gaussian with a diagonal covariance matrix and a uniform width—which assumes that the features are not correlated and have the same variance. It is standard practice to center data (subtract the mean) and standardize the data (divide by the standard deviation), so the same variance is a correct assumption, however uncorrelated features are not a valid assumption, unless the features have been decorrelated as well, e.g., whitened by applying the Mahalanobis distance, which is not standard practice—nor should it be.

Decorrelating (or whitening) features is a complicated matrix transformation similar to principal components analysis (PCA), factor analysis (FA), etc (see Section A.3), which may be useful for the purpose of visualization or similarity measurement, but the resultant dimensions or features lack clear validated clinical meaning and description, although they may sometimes be intuitive.

While this rationale for a Gaussian RBF kernel has notable concerns, it does not preclude rationale from a different perspective—but that is the problem, there is a lack of rationale and understanding in general for this and other kernels. An alternative rationale for the Gaussian RBF kernel may exist in its **local** behaviour, which creates sparsity (few non-zero values) in the kernel matrix, which allows big data to be computable, similar to what

[3]In ML, the Gaussian RBF kernel is defined without a few constants, which have no effect on optimization—but do affect my interpretation of parameter values!

Genton [94] identifies as a benefit of **compact** kernels. Such sparsity can also help with interpretability—these are not the reasons why authors use the Gaussian RBF kernel.

Returning to the idea of a kernel that begins with a Euclidean distance $\|\underline{x} - \underline{z}\|_2$ between the observations for two patients. Rather than make the assumptions of the Gaussian RBF kernel above, a much more intuitive idea of similarity is proximity, as the inverse of distance, without any assumptions of distributions.

In this case, I am interested in the negative of distance (i.e., additive inverse [272]) known as the **power distance** kernel (2.9.1) for $\beta = 1$: $k_{\text{Pwr}}(\underline{x}, \underline{z}) = -\|\underline{x} - \underline{z}\|_2^{\beta}$ [30, 226]. As distance grows, proximity decreases in a linear/proportional manner. This is the most intuitive kernel.

Notably, the power distance kernel **often performs better** than the Gaussian RBF kernel in studies/results [30] (Section 5.3) in addition to its **clear rationale**. Hence, it begs the question, why do I use the Gaussian RBF kernel? Why or when should one consider using a kernel that is more more **local** rather than **global** [161, 227, 288] or effectively **compact** [94]?

The Gaussian RBF kernel is more popular because Vapnik, the creator of SVM, introduced it, because it is built into most platforms and because many SVM users are unaware that positive definite (p.d.) kernels and conditionally positive definite (c.p.d.) kernels—are both **equally valid** (and **equally opaque**) for SVM [30, 167, 226, 242] but the power distance kernel has clearer **rationale**.

Is there **guidance** to match the effectiveness of a kernel's geometry to data types and other characteristics of data? I attempt to answer that question by deriving, designing and matching new and existing kernels to data types with rationale for transparency, interpretability and trust. The status quo of choosing less transparent and less accurate kernels based on convenience with little to no rationale, no longer suffices in the face of regulatory, business and political pressures.

1.2 The importance of this research

The importance of research on this study/problem pertains to the need for: (1) model interpretability in medicine; (2) guidance on model/kernel selection and use; (3) transparency to meet regulations; (4) basic scientific research on features and similarity; and (5) better accuracy with ML. I covered the first two points in the previous section, so I only discuss points the remaining three points in this section.

Transparency to meet regulations

There is a demand for transparency and understanding of machine learning (ML) models, data and results [126] to satisfy a citizen's "right to explanation" in the European Union [100] and the need for general assurance [274] for decisions informed by, or made by ML and/or big data.

Basic scientific research on features and similarity

My research on kernels as similarity functions for SVM has implications beyond similarity functions and beyond SVM. Scientific inquiry into the nature of features and similarity give us the opportunity to better understand the nature of kernels and the SVM algorithm, with lessons which may apply to other classifiers equivalent or similar to SVM.

Improving the accuracy, transparency and interpretability of kernels can potentially improve kernels in ensembles, multiple kernel learning, composite kernel learning, Auto-ML or metalearning, Gaussian processes, and deep learning with Gaussian processes.

Better accuracy with ML

Common ML classifiers[4] offer better accuracy [21, 71, 198] with statistical significance[5] [47, 301] on a substantive proportion of health care studies/problems and data, than **common** alternatives among statistical classifiers[6] and instance or rule-based classifiers[7]. When ML is not more accurate, it is usually as accurate.

Exactly how much better ML performs and how often, varies between studies based on which methods a study includes, how expertly the authors use each method, how the authors pre-process and impute data, what kinds of data are tested and which data sets are selected.

A few contrary results exist in studies which omit methods [165], use a non-standard or inexpert approach [292], or focus exclusively on text or categorical data which yield similar

[4]Common machine learning classifiers based on literature are: support vector machines [233], deep learning and convolutional neural networks [157], artificial neural networks [71], random forests [38], gradient boosting [85], Adaboost [83] and bagging [37].

[5]Note that none of these comparisons are statistically rigorous in that paired t-tests (and lesser methods) are deemed insufficient [64, 67, 25]. The literature does not offer data from more appropriate tests, e.g., the Wilcoxon signed ranks test, etc.

[6]Common alternative statistical classifiers for health care, based on literature are: logistic regression [71], naive Bayes [24], linear discriminant analysis or Fisher's linear discriminant [24], quadratic discriminant [24] analysis and Bayesian (or belief) networks [24].

[7]Common instance and rule-based classifiers for health care, based on literature are: k nearest neighbours, decision trees (including C4.5, J48, CART) and fuzzy logic.

performance across many different methods. However, on the whole, health care (and other fields), can benefit from the better accuracy[8] of ML versus alternatives.

1.3 Thesis outline

My thesis is organized as follows. Lists of abbreviations, symbols, figures and tables precede the introduction while back matter consists of an appendix, glossary and references.

In chapter 2, background is provided on machine learning, support vector machines and kernels along with the following intersecting fields of study: model interpretation, model selection, data types, and similarity and distance. Chapter 3 formulates the study/problem as a lack of model interpretability coincident with accuracy, a lack of rationale for use of kernels in SVM and a lack of transparency and interpretability in their use. Limits of interpretation in views of the study/problem and data are demonstrated.

The next three chapters are the main body of the thesis, as they address key aspects of the study/problem.

Chapter 4 lays out an approach to kernel selection, called kernel data modeling, as a hybrid between the **algorithmic approach** in machine learning and the **data modeling** approach in traditional statistics. It defines a framework of requirements for kernels to match data types, distributions, and requirements for similarity, classification and transparency. The chapter also proposes the foundational kernel class, explicit Mercer kernels, needed to create transparent kernels.

Chapter 5 performs tests of the proposed kernels and selected aspects of the framework in the preceding chapter. It confirms that I am able to create kernels with accuracy and transparency.

Chapter 6 defines new measures of model interpretability, validates these measures and applies them to perform model selection for accuracy with model interpretability.

Each of the core chapters provides its own summary and then I conclude this work with conclusions and future work (Chapter 7)..

[8] In health care other measures are sometimes more important than accuracy, sensitivity or specificity—such as the positive predictive value and negative predictive value of a test [7], or decision curve analysis [284].

Chapter 2

Background

This work covers a broad set of topics, relevant to each of the chapters with core content. I suggest reading it as follows.

The first two sections on **machine learning** and **SVM** provide a **general** background and delineate aspects which are out of scope for my thesis—readers strong in these topics may skip these, on a first reading.

Section 2.3 **Model interpretability** is groundwork for Chapter 6 **Measuring model interpretability**, while Section 2.4 **Model viewing and selection** is groundwork for Section 6.7 **Model selection with accuracy and inherent model interpretability**.

Section 2.5 **Data types** and Section 2.6 **Similarity and distance** are background for Chapter 4 **Kernel data modeling**.

Section 2.7 **Kernels** is very mathematical in some spots—hence I note that the content of greatest relevance is that section's **introduction** and the terms used in the subsection on **kernel classes**—as relevant to chapters 4 on kernel data modeling and transparent kernels.

Section 2.8 on **common kernels** is recommended as **general** background as well as subsection 2.9.1 **notable kernels for experiments** in the section on uncommon kernels.

2.1 Machine learning methods

Machine learning (ML) and pattern recognition[1] refer to the task of computers, i.e., machines, learning patterns in data for prediction, modeling or analysis. ML and pattern recognition arose from separate evolutionary paths in computer science and engineering respectively [24], but they are two names for the same subject–so I will only refer to ML.

ML is used to solve complex problems, as an alternative to classical/traditional statistical methods (Section A.1), when a direct solution or explicit model is not known for a set of data, or when computation is infeasible or too expensive [57]—as with high-dimensional data or data with a large sample size.

A study/problem consists of **data matrix** and a **target** to predict (Figure 1.2 on page 3). For example, I could have data on more than one table (**instances**) and I may wish to classify if the table can be used for dinner with a party of four (the **target**). The tables have **features** such as: table top shape, length of widest table top section, length of shortest table top section, table height, and number of legs. My data may have a number of tables to learn from: coffee tables, desks, dining room tables, ping pong tables, etc.

My thesis focuses on **machine learning** (ML) with **support vector machines** (SVM). I refer to **machine learning methods** as differentiated from **classical/traditional statistical methods** (Section A.1) and **classical/traditional instance and rule-based methods** (Section A.2). However if I omit the qualifier "classical/traditional" then there is overlap between the four categories of methods.

Since SVM learns and predicts with instances, but came after classical/traditional instance-based methods, it may be considered a *modern* instance-based method; and since the kernels used in SVM originated in math and statistics, but SVM came after classical/traditional statistical methods, SVM may be considered a *modern* statistical method. However it is most commonly referred to as a machine learning method and differentiated from members of the other groups.

My thesis focuses on **classification** as described in the introduction, however ML and kernels may be used for other tasks, such as, regression, clustering, reinforcement learning, dimensionality reduction and feature extraction (2.1.1).

My thesis focuses on classifying **atomic data types** (4.2) **not complex data types** such as text, images, video and audio signals.

[1] also called data mining, statistical pattern recognition or data analytics; and related to statistical image processing, business intelligence and artificial intelligence

My thesis **does not focus** on nor attempt to include **censored or time-series data**. That said, I have performed classification experiments with censored Canadian and American transplantation data with results similar to the results in this thesis—i.e., the relative results between kernels.

My thesis **does not focus** on **big data**, but includes two data sets (e.g., colon and prostate cancer data with $n = 2000$ and $n = 12,600$ features respectively) which are considered high dimensional by some [238] (features $n > 100$ or instances $N > 10,000$) but not others[2] (features $n > 10,000$ or instances $N > 10,000$). Numerous academic papers [51, 90, 286] do not provide quantitative definitions of big data since they define it in terms of the challenges it poses to computation and storage, an ever-changing target.

Notably, the **transparent kernels** that I propose, are **explicit Mercer** kernels **or additive separable** kernels, which are related to simpler **separable** kernels that Genton asserts offer **benefits** which are important for **big data** [94], because of reduced storage and computational complexity in implementation[3]. That said, implicit kernels reduce computational complexity in a different more well-known manner.

My thesis focuses on classification based on **supervised learning** [72, 233, 290] as one of four types of learning (Figure 2.1). Supervised learning is learning from examples [57], i.e., from instances or patients in a data matrix and target (Figure 1.2).

My thesis focuses on **SVM** as one ML classification method among various ML methods and as one kernel method among various kernel methods.

Common ML classifiers for health care based on literature are: support vector machines [233], deep learning and convolutional neural networks [157], artificial neural networks [71], random forests [38], gradient boosting [85], Adaboost [83] and bagging [37]. **Alternative ML methods** include: Gaussian processes [213], (simple) boosting [225] and radial basis function networks [24].

Methods in machine learning, statistics or instance and rule-based approaches, may be categorized as **non-parametric**, **semi-parametric** or **parametric**.

Examples of a **parametric** or **data-modeling** approach are found in classical/traditional statistics. The statistical approach examines the features in data and identifies one or

[2] A joint ICML and IJCAI 2018 workshop on computational biology (https://sites.google.com/view/wcb2018) defines high-dimensional data as "tens of thousands of features as well as thousands to millions of observations".

[3] Custom code may be required to realize this benefit however, since libsvm precomputes custom kernels and Matlab uses in-memory objects for SVM models.

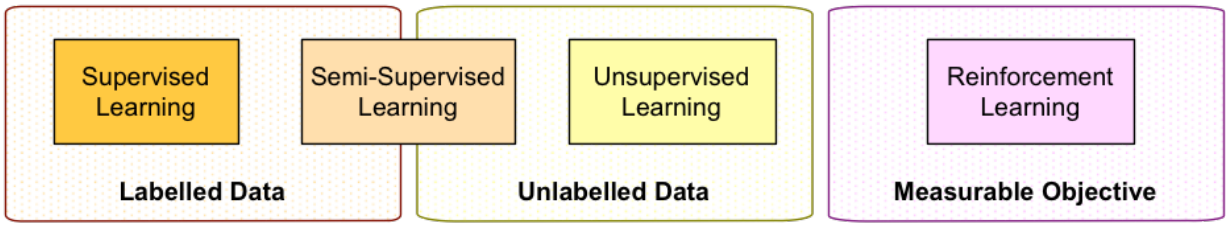

Figure 2.1: My thesis focused on supervised learning [72, 233, 290]—not semi-supervised learning [233, 290], unsupervised learning [72, 233, 290] or reinforcement learning [72, 290].

more models to describe those features, from a set of well-known models[4], e.g., Gaussian, Poisson, Weibull or Beta, where each model has parameters. I fit the model to the data to solve for the parameter values. This approach assumes or requires the data to have a **prior distribution or shape** with a **fixed** number of parameters with fixed meanings.

Non-parametric or **data-driven** techniques, typified by machine learning methods, do **not** have a fixed number of parameters with fixed meanings. For example, SVM with a Gaussian RBF kernel has one fixed parameter for the kernel width, but the model also learns model weights which are based on the number of instances in the data, and only uses a **variable** subset of those instances as support vectors for prediction. I note that a Gaussian RBF kernel that has a large kernel width generally creates a smoother and less curved class boundary in classification—i.e., there is a prior assumption about the **distribution** of the data, whereas for a small kernel width there is less or no assumption and the class boundary is purely data-driven.

Machine learning is a **process** (Figure 1.4) with steps that include: data wrangling, data analysis, feature and instance pre-processing, learning, testing and use (at the highest level). I note that some steps can overlap with each other. In particular, feature extraction or pre-processing can done prior to the learning step, or within it as part of the model.

There are other perspectives that may be applied to categorize and understand machine learning methods and models, e.g., convex versus non-convex optimization [34], filter versus wrapper versus embedded methods [109], ensembles [307] versus individual algorithms and weak versus strong learners [225].

There are also other types of learning: metalearning or auto-ML [36], multi-view learning [294], online learning [146], passive versus active learning [232], metric learning[151, 285], transductive versus inductive learning [155, 308] and generative learning [208].

[4]whose properties have been analyzed in the literature

2.1.1 Applications of classification in health care

Classification applies to a variety of tasks in health care, with respect to clinical practice, epidemiology and public health surveillance. Since my thesis focuses on binary classification and kernels as similarity functions, I review some examples of classification applied in these contexts.

- Binary classification (the class/decision and its probability of error) and kernels as similarity functions, may be used in:
  - Clinical practice, for
    - Testing or forensics to detect a condition
    - Diagnosis of a single/specific disease or condition—not differential diagnosis
    - Case-based (similar case) retrieval for a single/specific disease or condition, to assist diagnosis and treatment optimization
    - Prognosis at a fixed endpoint for a single/specific disease or condition, e.g., chance of survival at five years
  - Epidemiology and public health surveillance, to
    - Model organ functions by classifying types of cellular behaviour/signaling among known types
    - Investigate outbreaks by identifying/retrieving similar cases

Multi-class classification is a simple extension of binary classification but is beyond the scope of my thesis. Anomaly or one-class classification has other unique applications which can help identify outbreaks or fraud.

2.2 Support vector machines

There are a number of resources [24, 57, 233, 240, 281] that introduce **support vector machines (SVM)**, however few are adequate introductions for the layperson because they almost immediately express things mathematically—which is fine for a reader comfortable in that realm, but not otherwise. In the first subsection below I provide **simpler guidance** to expand and encourage the use and acceptance of SVM. In then discuss SVM theory and then I provide an overview of SVM from the perspective of an engineer, statistician or mathematician.

Before I explain what SVM is, I provide a little bit of context. My thesis focuses on SVM, as a popular and important ML method [293] and a mainstay among state-of-the-art methods [47, 198, 301] with competitive accuracy on the data and tasks in my scope[5]. SVM has a strong theoretical foundation [187, 281] with applications to clinical research including diagnosis and prognosis [58, 144, 262].

SVM is a classification technique, while **support vector regression (SVR) [241] is a regression** technique. SVM is fairly intuitive once understood, but requires a few steps to explain, each one building upon the last.

The first SVM concept is **linear separability**. Suppose I have data for two classes of patients, e.g., with and without heart disease, and the data consist of three predictive features, e.g., age, weight and diastolic blood pressure, which I may view as a three dimensional space. To classify a new patient as either having or not having heart disease, I learn from data on multiple existing patients which appear as points in that space.

If the data for the two classes can be separated by a plane, then I say that the data are linearly separable and I call the plane a **class boundary**. In the general case of n-dimensions the class boundary is called a **hyperplane** and it is a flat that is one dimension less than the space.

There may be many different hyperplanes as class boundaries that can separate data with different angles and distances to one class or the other. A good class boundary **maximizes the distance (or margin)** between itself and the two classes, so that it will generalize well to new data, where that new data **may** have points in one class closer to the other class than in my training data. That is, new data may close the gap between the two classes (Figure 2.2). This criterion is called the **maximum margin** and SVM is referred to as a **maximum margin classifier**.

That was the **simplest case—linearly separable data** and a **linear class boundary**. Many different algorithms will classify linearly separable data very well, i.e., perfectly on training data, and very well on test/validation data.

A more complex scenario is when data from the **two classes are overlapping** so that data are **not linearly separable** (without errors). The original conception of SVM (hard-margin SVM) which simply maximizes the margin is extended to allow for errors (called soft-margin SVM) and takes on **two objectives**: **maximize the margin** and **minimize**

[5]Deep learning has garnered recent attention for markedly higher accuracy than other methods with tasks pertaining to image/video/object recognition and tracking. That said, my work on kernels may also apply to deep kernel learning with Gaussian processes [288].

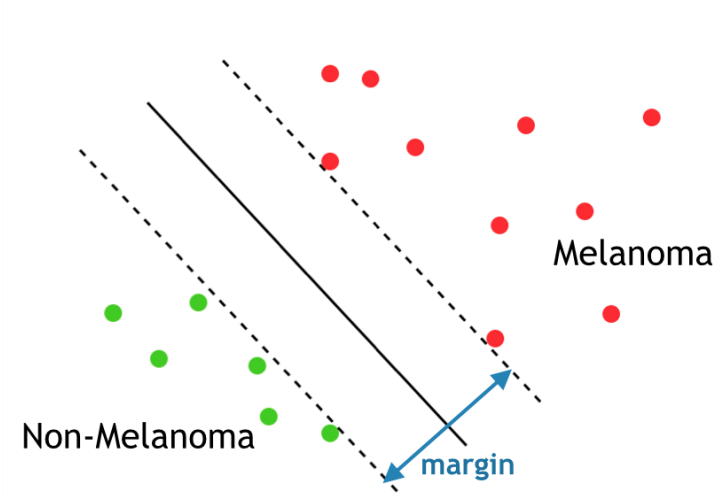

Figure 2.2: SVM class boundary (solid line) and margin (dotted lines)

errors. Errors are points that fall on the wrong side of the class boundary and points which fall within the margin. The margin is soft in that points can fall within the margin.

A parameter C called the **cost of error** (or box constraint, as a geometric interpretation) [240] specifies how much weight to place on one criterion versus the other. Higher C places greater emphasis on minimizing error. Such a parameter is called a **regularizer**, and balancing the weight between two criteria is called **regularization**. In this case, regularization is used to choose between paying greater attention to the data/errors (**i.e., data/error-driven**) versus focusing on the **preconceived model or concept of a maximum margin (model-driven)**.

There is one more key concept in SVM. Thus far I have described linear class boundaries, i.e., linear SVM, however a **curved** (or **curvilinear** or non-linear) class boundary may be more appropriate in some cases. This is achieved with a trick in mathematics known as the **kernel trick**.

In the introduction I explained that a key component of SVM is a kernel, which is a similarity function used to compare two instances of data $\underline{x}$ and $\underline{z}$, such as, two patients each represented by a vector of numeric readings. Linear SVM uses a dot product, or scalar projection, of the two vectors, as its similarity function—and that is called a linear kernel: $k_{\text{Lin}}(\underline{x}, \underline{z}) = \langle \underline{x}, \underline{z} \rangle = \underline{x}^{\text{T}} \underline{z}$. This is essentially a sum of one-sample covariances (or correlations when data are standardized as is common practice in machine learning). SVM computes a matrix of similarities between every possible pair of instances in the training data X as a set of points in the multidimensional space $\mathcal{X}$. SVM learns from that matrix

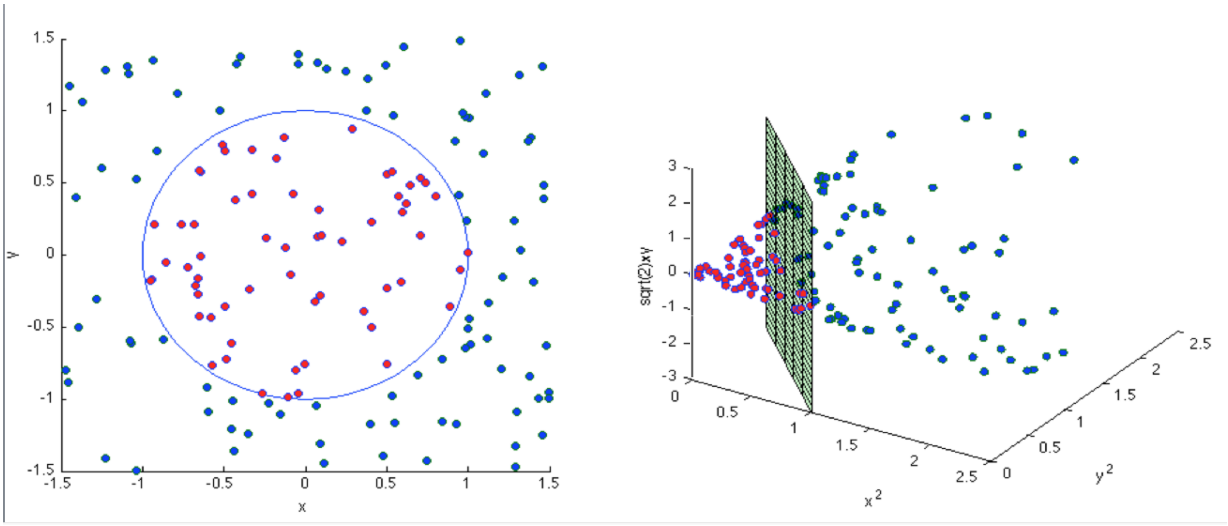

Figure 2.3: An example of how a feature map (or basis function) ϕ_{new} transforms the original data in an input space $\mathcal{X}$ at left, which requires a non-linear class boundary, to a feature space $\mathcal{F}$, at right, where the class boundary becomes a hyperplane.

and the targets and during the learning process it constructs a kernel or Gram matrix $G = XX^{\mathrm{T}} = \left\{k_{\text{Lin}}\left(\underline{x}_i, \underline{x}_j\right)\right\} \forall i, j$.

The **kernel trick** [24, 233, 240] is a mathematically valid **substitution**, that replaces the **dot product of the original data** $k_{\text{Lin}}\left(\underline{x}_i, \underline{z}_j\right) = \left\langle \underline{x}_i, \underline{z}_j \right\rangle = \underline{x}_i^{\mathrm{T}} \underline{z}_j$, with a **dot product of transformed data** $k_{\text{new}}\left(\underline{x}_i, \underline{z}_j\right) = \left\langle \underline{\phi}_{\text{new}}\left(\underline{x}_i\right), \underline{\phi}_{\text{new}}\left(\underline{x}_j\right) \right\rangle$, where ϕ_{new} is a function that **maps or transforms** data from the space $\mathcal{X}$ (Figure 2.3 at left) to new **feature space** $\mathcal{F}$ (Figure 2.3 at right) where the class boundary is a hyperplane in the latter space, but in the former space it is distinctly curvilinear or non-linear.

The transformation should assist the objective of separation with a hyperplane and/or it should assist the objective of a wider margin and/or lesser error. Clearly with the right shape for the class boundary in the input space, lesser error is achieved. I call ϕ_{new} a **feature map** or **basis function** and express it in mathematical notation as follows: $\phi_{\text{new}} : \mathcal{X} \rightarrow \mathcal{F}$.

This example (Figure 2.3) is derived from Shawe-Taylor and Cristianini [233] and Burges [40], and thereafter referenced by Wu *et al.* [292] and others. The kernel trick allows us to use curvilinear or non-linear class boundaries, however there is a constraint on the equation of the kernel—it must be a dot product in a feature space $\mathcal{F}$ with a basis function ϕ_{new}, regardless of whether or not I actually know that feature space and function (assurance of their existence is sufficient).

There are a few final points at a high level about SVM.

First, as with any learning method, it works in two steps: (a) learn the model weights (in this case, support vectors/coefficients) and (b) make a decision, estimation or prediction; where the step (a) requires optimization and may be computationally intensive and step (b) is very fast in comparison.

Second, SVM is an instance-based method. That is, to make a prediction, SVM essentially computes the similarity between a new instance and a weighted sum of selected instances from the training data—where the selected instances are called support vectors. The SVM prediction function is linear in instances but not features (for all SVM kernels).

Finally, while I have explained how SVM works, there is additional theory (in the next subsection) that provides rationale as to why it is useful. Such rationale helps justify the method's firm theoretical foundation, although a doctor (or other user of SVM) does not need to know the finer mathematical details of that theory.

2.2.1 SVM theory

Support vector classification is a machine learning technique with a theoretical foundation in computational learning theory, regularization and convex optimization, as discussed in this section, and kernels (discussed separately in section 2.7).

Computational learning theory includes concepts such as empirical risk, capacity control, Vapnik Chervonenkis (VC) dimension and VC bound [281].

Empirical risk [281] refers to the probability of error for a learning machine with respect to a probability distribution function (p.d.f.) on a ***sample*** with a ***finite*** amount of data drawn from the population.

Regularization (explained in the previous section) is implemented in SVM with the hyperparameter C, the **cost of error** (also explained in the previous section) which governs the **trade-off** between two objectives: **minimizing error** and **maximizing margin**. In SVM, this regularization is also called **capacity control**.

Capacity control [281] refers to precise control over how specific versus general (i.e., how complex) the classification boundary is in SVM. A class boundary that is **very specific** (has a lot of **capacity** to memorize the data) may not generalize well to classifying test data because it has paid too much attention to specifics/detail in training data, i.e., it may **overfit** the training data. Conversely, a class boundary that is **very general** (has too little **capacity** to learn/remember data) may not be specific enough to classify test

data because it has paid too little attention to specifics/detail in training data, i.e., it may **underfit** the training data.

The **Vapnik Chervonenkis (VC) dimension** [281] for ***any*** binary classifier, relates to the concept of capacity control (or ability to learn/remember). Consider a set of functions with deterministic outputs, parameterized by α, which may in fact be the cost of error, C, as in SVM. For a binary classification study/problem, given a set of N points, there are 2^N possible ways to classify those points. If the set of functions can achieve each one of those output combinations then I say it **shatters** those N points.

The VC dimension, h, for a set of functions is defined as the **maximum number of training points that can be shattered** by it (*in at least one configuration, but not all configurations*). *Not every* set of h points can or must be shattered by the set of functions. There is an example of a function with infinite VC dimension which cannot shatter 4 equally spaced points in a certain labelling configuration. So to ascertain the VC dimension h, the question becomes: is there ***any*** set of h_{try} points, in ***any*** configuration (triangle, square, straight line) that can be shattered by the set of functions? If so, then $h \geq h_{\text{try}}$. I note that h is a non-negative integer.

The **VC bound** (2.1) [281] holds with probability $1 - \eta$, where I choose small η, for a classifier with VC dimension h and a data set with N points. The bound is independent of the distribution $P(\mathbf{x}, y)$ and the rightmost term of the bound, on the right-hand side of the equation is called the VC confidence.

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h\left(\log\left(2N/h\right) + 1\right) - \log\left(\eta/4\right)}{N}} \tag{2.1}$$

It is said that in general with more features or more dimensions it is usually easier to achieve linear separability [117]. With some kernels there are an infinite number of dimensions in the feature space (as I discuss in the introduction) however, SVM uses these dimensions in a finite way [117]—and the dimensionality of the model in terms of number of instances, is reduced compared to original data, since support vectors are a subset of instances and support vectors are ideally sparse.

In the two-step process of learning model weights (support vectors in this case) and then making a prediction. SVM is usually solved using the following optimization formula in

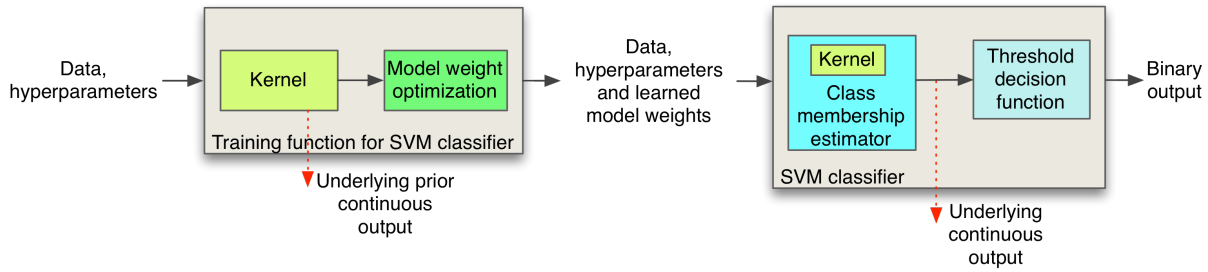

Figure 2.4: SVM occurs in two steps: first, learn model weights with optimization (left), then use the weights in the SVM classifier (right).

its dual form (2.2) along with its associated constraints (2.4),

$$W(\alpha) = -\sum_{i=1}^{\ell}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j k\left(\underline{x}_i, \underline{x}_j\right) \tag{2.2}$$

$$\underset{\alpha}{\text{argmax}} \quad W(\alpha) \tag{2.3}$$

$$\text{subject to} \qquad 0 \le \alpha_i \le C, \quad i = 1, ..., \ell \tag{2.4}$$

$$\sum_{i=1}^{\ell}\alpha_i y_i = 0$$

where $\underline{x}_i$ and $\underline{x}_j$ are vectors of explanatory variables, y_i and y_j are the corresponding outcomes, α_i and α_j are weight parameters in constrained optimization derived from Lagrange's method (which identify the support vectors), $k\left(\underline{x}_i, \underline{x}_j\right)$ is a kernel, and C is the cost of error.

2.2.2 A mathematical view of SVM

As a learning method, SVM occurs in two steps: first, learn the model weights with optimization, then use the weights in the SVM classifier (Figure 2.4).

In SVM, i.e., 1-norm soft margin SVM, as implemented in Matlab, the model weights, or support vectors α_i are learned by optimizing the following equation [24, 233]:

$$\underset{\alpha_i}{\text{argmax}} \; -\sum_{i=1}^{\ell}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j z_i z_j k\left(\underline{x}_i, \underline{x}_j\right) \tag{2.5}$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i z_i = 0 \quad \text{and} \quad 0 \le \alpha_i \le C \tag{2.6}$$

The above equation, known as the dual form is used in implementation but in the equivalent

primal form (2.7) [233] it is easier to recognize the effect of C as a trade off variable (or regularization parameter):

$$\underset{\underline{\beta}, \xi_i, b}{\operatorname{argmin}} \quad \frac{1}{2}\underline{\beta}^{\mathrm{T}}\underline{\beta} + C\sum_{i=1}^{\ell}\xi_i \tag{2.7}$$

$$\text{s.t.} \quad z_i \cdot \left(\beta^{\mathrm{T}}\underline{x}_i + b\right) \geq 1 - \xi_i \quad \forall i, \quad \xi_i \geq 0 \tag{2.8}$$

In (2.7) C is called the cost of error because it specifies the weight of training errors ξ_i in the second term. C also regulates the trade off between the goals of minimizing training error and maximizing the margin $\gamma = \frac{1}{\left\|\underline{\beta}\right\|}$ in the first term, where the latter offers better generalization, i.e., lower validation/test error [233]. We can maximize margin γ by minimizing its inverse $\left\|\underline{\beta}\right\|$ or the square of its inverse $\frac{1}{2}\left\|\underline{\beta}\right\|^2 = \frac{1}{2}\underline{\beta}^{\mathrm{T}}\underline{\beta}$ as in the first term of (2.7).

From the above interpretation of C, one can retrospectively see its similar influence in the dual form. In (2.5) and (2.6), changes in C affect the first and second terms by ΔC and $(\Delta C)^2$, respectively, increasing the effect of training instances (or errors) and decreasing the effect of the margin. The second term in (2.5) can be re-written as $-\frac{1}{2}\underline{\beta}^{\mathrm{T}}\underline{\beta}$ which pertains to the margin (as previously described) and where $\underline{\beta}$ is a vector along the class boundary $f(x) = \underline{\beta}^{\mathrm{T}}x + b$ in the feature space, $\underline{\beta}^{\mathrm{T}}$ is a vector perpendicular to the class boundary and $\underline{\beta} = \sum_i \alpha_i z_i \phi(\underline{x}_i)$ which explains how $-\frac{1}{2}\underline{\beta}^{\mathrm{T}}\underline{\beta}$ equals the second term.

In the second part of SVM, we use the learned model weights α_i in the classifier h to predict (or estimate) the binary target as $\hat{z}$ for a test instance $\underline{v}$ (which is disjoint from $\underline{x}_i$), as follows:

$$\hat{z}(\underline{v}) = h(\underline{v}) = \operatorname{sign}(\hat{y}(\underline{v})) \tag{2.9}$$

$$\hat{y}(\underline{v}) = \sum_{i=1}^{l} \alpha_i z_i k(\underline{x}_i, \underline{v}) + b \tag{2.10}$$

$$b = {}^{-1}\!/_{2}\left(\sum_{i=1}^{l} \alpha_i z_i k(\underline{x}_i, \underline{x}_{a^-}) + \sum_{i=1}^{l} \alpha_i z_i k(\underline{x}_i, \underline{x}_{a^+})\right) \tag{2.11}$$

$$\text{where } a^- \text{ chosen s.t. } \alpha_{a^-} y_{a^-} < 0 \tag{2.12}$$

$$a^+ \text{ chosen s.t. } \alpha_{a^+} y_{a^+} > 0 \tag{2.13}$$

Note that we express the SVM equations with an underlying continuous output $\hat{y}$ (2.10) which provides more information toward a quantitative measure of model interpretability than the binary output $\hat{z}$ (2.9). Logistic regression [71], neural networks [24] and gaussian mixture models [24] are examples of other classifiers which also have such an underlying

output.

2.3 Model interpretability

As discussed in the introduction, I differentiate between interpretation and explanation following Montavon et al [192]—where interpretation refers to providing information in a form that can be perceived and understood by a person, whereas explanation requires additional work, such as selecting information [188], where the latter is best done with human judgement.

I assert (Chapter 6) that model interpretability is then a function of the individual (e.g., learning and forgetting curves [210]) and **inherent model interpretability** as my proposed term and concept (independent of the individual and inherent to the model and data).

Lipton [168] provides a good taxonomy for **model interpretability** with concepts falling into two broad categories: transparency (the opposite of a black box) and post-hoc interpretability.

Post-hoc interpretability involves an explanatory model separate from the predictive model, or visuals that transform data where the transformation is also a separate explanatory model. Liang [164] cautions against explaining a black box predictive model with another black box explanatory model.

Riberio et al [216] create an external **local linear model** to approximate the prediction model in a post-hoc approach called LIME. They jointly optimize accuracy and model complexity but they do not elucidate much about model complexity as in my work. LIME perturbs features in a separate binary representation of features, which sometimes map to **non-local** features in the original space of data. In their examples they use the binary model output, only referring in passing to the possibility of using a continuous output for classifiers, as I do.

Transparency, on the other hand, focuses on the predictive model itself, and has three aspects: decomposability, simulatability and algorithmic transparency [168].

Decomposability refers to being able to see and understand the parts of the model of the model, e.g., kernels and parameters and the parts of the data, i.e., features and instances—and how they contribute to a result from the predictive model. Some authors refer to the output from decomposition as an **interpretation**, e.g., initial understanding,

separate from an **explanation** [126, 192] that may require **analysis**, **selection** or perhaps **synthesis**. Miller adds that explanations are **selected** and **social** [189].

Since the social and synthesis tasks are more suitable to a person than a computer—it is reasonable for my work to focus on inherent measures of interpretability, rather than explanations.

[172] express that some types of models are more **intelligible** (i.e., decomposable) than others. I include categories for generalized linear and generalized additive models in my measures as a result of their work.

Simulatability, as another aspect of transparency, refers to a model that a person can mentally simulate or manually compute in reasonable time [168] and is correlated, for example, with the number of features in a linear model, or the depth of the tree in a decision tree. **Model complexity** is implied Lipton's examples but the term is not invoked although other authors refer to it [183, 205, 39].

Occam's razor, also called the principle of **parsimony** [244], is a well known principle related to model complexity. Regarding models, it says that among sufficient explanations (e.g., equally accurate[6] models), the simplest[7] should be preferred. A quick note on sufficiency: for multiple equally accurate models, none are necessary, because any one of them is sufficient. Model accuracy is sought first, then simplicity. Using my proposed measure one can search for the model with highest interpretability among models which are equally sufficient in accuracy.

Backhaus *et al.* [12] propose a quantitative measure of model interpretability—but that is for a different meaning or definition—the ability for a model to interpret data, with relevance in relevance vector machines as the context.

Related to my work, **sensitivity analysis of model outputs** (SAMO) [9, 113] describes how sensitive a model output is to a change in feature values, one at a time—which is the approach of my proposed general measure.

In variance-based sensitivity analysis, Sobol [245] finds the variance in the output explained by an input feature. Liu et al [170] performs entropy-based sensitivity analysis, called global response probabilistic sensitivity analysis (GRPSA), to find the influence of input features—where entropy is used to compute the effect as information loss. Lemaire *et al.* [160] apply sensitivity analysis but their perturbations are non-local and could easily create

[6] where accuracy cannot be distinguished with statistical significance

[7] [Sober] refers to [Akaike]'s definition of the simplest model as the model with the least degrees of freedom, i.e., least number of (independent) coefficients.

points outside of any known clusters of instances and true states of nature. Poulin *et al.* [209] provides effective visualization and analysis tools but for SVM they only apply their method to linear SVM and its binary output.

Automatic model selection methods have been proposed for accuracy [4, 194]—these are based on rules computed from many data sets. The rule-based approach is brittle in comparison to my measures, since it only works with a fixed set of candidate kernels.

2.4 Model viewing and selection

Barbella *et al.* [15] try to find the closest point on the class boundary in the feature space, but since they can only do computations in the input space, they can only find a local not global minimum in distance—and they use the wrong formula for distance[8]. Furthermore, finding a single point doesn't characterize the stability and uncertainty of changes in the space.

Other authors provide methods which have various limitations and shortcomings: e.g., Poulin *et al.* [209] provide an interactive method for a linear kernel; Lemaire and Clérot use a sensitivity-based approach with sampling that is flawed because it is non-local and sampled from either class; Lemaire *et al.* [160] has the same shortcomings and uses a ranking with unclear benefits and squares the sensitivity without rationale.

Montavon *et al.* [192] use a Taylor decomposition method which on the one hand is more complete than sensitivity analysis, but is less interpretable and requires differentiation of formulas. The advantage of sensitivity analysis is that it is readily understood, i.e., transparent and interpretable and easy to compute.

Guidance from kernel selection methods [5, 194, 276, 277] in the literature is insufficient. For example, in several cases, decision tree rules are learned to select kernels—however decision tree rules are notoriously unstable [205] because they are sensitive to how the study/problem is setup, i.e., the kernels considered, the data sets considered, the way the data are pre-processed, etc. If you want to use any kernel outside of those considered the rules do not apply.

[8]The distance should always be computed using the kernel not the Euclidean distance inside the kernel

2.5 Data types

One of my objectives is to examine learning methods and models for specific data types, e.g., real numbers versus binary values. I therefore review data type definitions and standards (Table 2.1).

A data type is "a set of distinct values, characterized by properties of those values, and by operations on those values" [132]. A less abstract view of data types are Stevens' four scales of measurement [249, 290] as the progenitor for data types: nominals, ordinals, intervals (e.g., dates) and ratios (e.g., reals and integers). However, these are still abstract compared to the data types that I use with machine learning platforms, so I review other sources such as Weka and Matlab as machine learning platforms, and unified modeling language (UML) [81], ISO 11404 [132] and ISO 21090 [134], as standards (Table 2.1).

My thesis is concerned with simple (or primitive) data types in my review (Table 2.1) versus the types I categorize as complex data types or documents. I leave complex data types out of scope, i.e., data involving text, images, video and audio—since learning with each of those types have domain-specific needs, tasks, measures of performance and specialized kernels. Some standards, like UML [81], distinguish between simple and complex data types, while others may not—in any case, it is an accepted way to categorize data types [123].

For my purposes in kernel data modeling, I find that the Weka data types are not rich enough compared to others, while the Matlab, ISO 11404 [132] and ISO 21090 [134] data types are overly rich. The UML [81] simple data types (primitives) are most appropriate but do not include ordinals (except as a complex data type).

Table 2.1: Data types from different sources versus Stevens' scales of measurement [249]. Further details are found in the appendix (Section A.4 and Section A.5).

Source	**Simple data types** Note: $\boldsymbol{xx} \in \{8, 16, 32, 64\}$				**Complex data types or documents**	
Stevens	**Nominal**	**Ordinal**	**Interval**	**Ratio**		
Weka	nominal		date	numeric	string	relational
Matlab	logical, nominal	ordinal	datetime	int$\boldsymbol{xx}$*, uint$\boldsymbol{xx}$*, float, single, double	string	vector, matrix, etc.
UML	boolean, enumeration		date	integer, real	string	set, bag, sequence, orderedset, tuple
ISO 11404	boolean, state	enum-erated, ordinal	date-and-time	integer, rational, scaled, real, complex	chr*, chrstr*	set, bag, sequence, class, octet
ISO 21090**	bl; cs, sc, cd, ii, pqr	co	ts	real, int, mo, pq, rto	st	coll, dset, list, glist, slist, ed, ed.image, ed.text, ed.signature, bag, rto en, ad, tel, xp, adxp, enxp, qset, strucdoc, etc.

2.6 Similarity and distance

In this section I explain concepts regarding **similarity**, **distance**, **measures** and **metrics**, which I use in chapters 6 on measuring model interpretability and chapter 4 on kernel data modeling. I discuss kernels (as similarity functions) separately in sections that follow.

In this section, **this introduction** and the **first subsection** on similarity and distance (simdist) functions are the most important, while the remaining subsections can be skipped since they provide detail on simdist functions for specific data types. They provide grounded examples as context, and specific parts will be referenced in later chapters where relevant—e.g., the asymmetric weighting of positive versus negative matches in the azzoo similarity function is analogous to the asymmetric class balancing feature in some of my proposed transparent kernels.

In mathematics the terms **measure** and **metric** are distinct from each other and have formal definitions. When a function does not fulfill either of those formal definitions, I simply refer to it as a function. Kernels are usually **functions**.

A **measure**, operates on a set, e.g., a set of numbers, and is said to be a "systematic way to assign a number to each suitable subset of that set, intuitively interpreted as its size". A

measure has three properties: non-negativity, countable additivity and an output of zero for a null input.

A **metric** or **distance function**, d, is a function of two variables on a set, X, that outputs a real-valued scalar and satisifies four properties for $x, y, z \in X$:

1. $d(x, y) \geq 0$, i.e., is non-negative
2. $d(x, y) = 0$ iff $x = y$, i.e., is only zero for equal inputs
3. $d(x, y) = d(y, x)$, i.e., is symmetric
4. $d(x, z) \leq d(x, y) + d(y, z)$, i.e., satisfies the triangular inequality

In general, **kernels** and **similarity functions** are **not** metrics, because their outputs are not necessarily positive, and they do not satisfy the triangular inequality (as I next discuss).

The inverse of distance is **proximity** (a type of similarity). I can therefore convert a distance function into a similarity function with the additive inverse $s(\underline{x}, \underline{z}) = -d(\underline{x}, \underline{z})$ [272] or the multiplicative inverse $s(\underline{x}, \underline{z}) = \frac{1}{1+d(\underline{x}, \underline{z})}$ [166], or other means such as a Gaussian RBF kernel applied to a distance function.

Similarity may be explained in terms of nine key concepts from well-cited authors on the subject in the literature [19, 162, 166, 273, 299]. A **similarity function**:

1. Must be symmetric in its inputs [299]
2. Should increase with the commonality of inputs [166]
3. Should decrease with differences in inputs [166]
4. Should be maximal when two inputs are the same [166]
5. Should be minimal for no commonality in the inputs [166]
6. Should be a weighted sum of perspectives, where the weights are the amounts of information in the descriptions [166]
7. May obey the triangular inequality, although this may lead to counter-intuitive scenarios [166]
8. May have a finite maximum [166]

9. May have a finite minimum [166]

Some authors also claim that similarity functions are "universal" if they are based on information theory [166] or better yet, information distance per Kolmogorov complexity [19, 162]. However, it is hard to see how these apply to my context of kernels which compare one instance to another with only one observation per feature.

It is important to note that I do **not** include **feature extraction, dimension reduction and visualization** techniques (Section A.3)—such as principal component analysis (PCA), maximum variance unfolding (MVU), non-negative matrix factorization (NMF), probabilistic latent component analysis (PLCA), t-distributed stochastic neighbourhood embedding (t-SNE), etc—in my discussion of similarity and distance. I also do not include the **Mahalanobis distance** which includes implicit whitening per PCA (see Section 2.9 for further discussion).

Why? The simdist function literature [27, 49, 102, 103, 166] does not include these topics and techniques; and these techniques result in transformed features which do not have prima-facie or immediate **clinical meaning or interpretation**, in opposition to my **objectives** and in contrast to the original features which are clinically meaningful (in benchmark health care data sets).

2.6.1 Similarity and distance functions

Similarity and distance (simdist) functions are used to compare two vectors of data—where the vectors may be samples with multiple observations as a common context, but in my case the context is vectors that represent instances with multiple features. Simdist functions broadly include kernels, however because kernels have additional requirements I discuss them separately. Simdist functions are created for specific data types, so I review them according to each type in the sections that follow.

Simdist functions may be used to provide information in their own right, as in correlation, or they may be used in statistical, instance and rule-based methods for classification and clustering, such as k-nearest neighbour classification, large margin nearest neighbour classification and k-means clustering.

2.6.2 Binary simdist functions

For machine learning purposes I encode binary values symmetrically to the set $\{-1, +1\}$, whereas in the simdist function literature the common convention for binary values is

$\{0, 1\}$. I denote similarity and distance functions by s and d respectively.

When two binary values are positive, i.e., both $+1$, then there is a positive match and I can count the number of positive matches between two vectors $\underline{x}$ and $\underline{z}$ with the expression $\underline{x}^{\mathrm{T}}\underline{z}$. Similarly negative matches, i.e., both 0, are counted by $\overline{\underline{x}}^{\mathrm{T}}\overline{\underline{z}}$. The number of mismatches between both vectors is counted by the expression $\underline{x}^{\mathrm{T}}\overline{\underline{z}} + \overline{\underline{x}}^{\mathrm{T}}\underline{z}$.

Cha *et al.* [49] uses four groups to categorize binary simdist functions: Hamming based, inner-product based, correlation based and weighted measures.

The first group pertains to distance in terms of mismatches or differences. It contains similarity measures related to the Hamming distance (2.14) which is a L_1 distance metric also known as the edit distance, city block distance or Manhattan distance [49]. This group of functions have been used in health care for breast cancer detection [26], similarity of events in a health care event log [28] and quality and safety evaluation in health care institutions [74]. Measures in the same group by Sokal and Michener or Rogers and Tanimoto involve normalization by different amounts [49].

$$d_{\text{Hamming}}\left(\underline{x}, \underline{z}\right) = \underline{x}^{\mathrm{T}}\overline{\underline{z}} + \overline{\underline{x}}^{\mathrm{T}}\underline{z} \qquad \underline{x}, \underline{z} \in \mathbb{R}^n \tag{2.14}$$

$$s_{\text{Hamming}}\left(\underline{x}, \underline{z}\right) = \underline{x}^{\mathrm{T}}\underline{z} + \overline{\underline{x}}^{\mathrm{T}}\overline{\underline{z}} = n - d_{\text{Hamming}} \tag{2.15}$$

The second group of binary functions pertains to the inner-product similarity function (2.16) [49] which only measures positive matches. The Jaccard similarity coefficient, $s_{\text{JaccardNeedham}}$ is a function in this group which is used in health care applications [76]. Cha *et al.* [49] indicate that the inclusion or exclusion of negative matches is one of the most contentious issues in binary simdist functions. In ecology, the presence of a species may be observed as a certain value, whereas the absence of a species in an area is uncertain in observation—hence such data are called presence-only data, where the inner product is appropriate.

$$s_{\text{innerProduct}}\left(\underline{x}, \underline{z}\right) = \underline{x}^{\mathrm{T}}\underline{z} \tag{2.16}$$

$$s_{\text{JaccardNeedham}}\left(\underline{x}, \underline{z}\right) = \frac{\underline{x}^{\mathrm{T}}\underline{z}}{\underline{x}^{\mathrm{T}}\underline{z} + \underline{x}^{\mathrm{T}}\overline{\underline{z}} + \overline{\underline{x}}^{\mathrm{T}}\underline{z}} \tag{2.17}$$

The third group of binary measures discussed by Cha *et al.* [49] are correlation based

measures. These measure positive and negative matches equally.

$$s_{\text{correlation}}\left(\underline{x},\underline{z}\right)=\frac{\underline{x}^{\mathrm{T}}\underline{z}\cdot\overline{\underline{x}}^{\mathrm{T}}\overline{\underline{z}}-\underline{x}^{\mathrm{T}}\overline{\underline{z}}\cdot\overline{\underline{x}}^{\mathrm{T}}\underline{z}}{\sqrt{\left(\underline{x}^{\mathrm{T}}\overline{\underline{z}}+\underline{x}^{\mathrm{T}}\underline{z}\right)\left(\overline{\underline{x}}^{\mathrm{T}}\underline{z}+\overline{\underline{x}}^{\mathrm{T}}\overline{\underline{z}}\right)\left(\underline{x}^{\mathrm{T}}\underline{z}+\overline{\underline{x}}^{\mathrm{T}}\underline{z}\right)\left(\overline{\underline{x}}^{\mathrm{T}}\overline{\underline{z}}+\underline{x}^{\mathrm{T}}\overline{\underline{z}}\right)}} \tag{2.18}$$

$$s_{\text{Yule}}\left(\underline{x},\underline{z}\right)=\frac{\underline{x}^{\mathrm{T}}\underline{z}\times\overline{\underline{x}}^{\mathrm{T}}\underline{z}-\underline{x}^{\mathrm{T}}\overline{\underline{z}}\times\overline{\underline{x}}^{\mathrm{T}}\underline{z}}{\underline{x}^{\mathrm{T}}\underline{z}\times\overline{\underline{x}}^{\mathrm{T}}\underline{z}+\underline{x}^{\mathrm{T}}\overline{\underline{z}}\times\overline{\underline{x}}^{\mathrm{T}}\underline{z}} \tag{2.19}$$

Cha *et al.* [49] propose a measure s_{azzoo}[9] with variable weight that performs optimally in their experiments as a measure that is in between an inner product and a correlation (or at either extreme).

$$s_{azzoo}=\underline{x}^{\mathrm{T}}\underline{z}+\sigma\overline{\underline{x}}^{\mathrm{T}}\overline{\underline{z}}\qquad\qquad\sigma\in[0,1] \tag{2.20}$$

Finally, Cha *et al.* [49] describe weighted measures, where different weights are applied to each feature dimension.

$$s_{\text{lightedHamming}}\left(\underline{x},\underline{z}\right)=\sum_{i=1}^{n}w_i\left(x_iz_i+\overline{x}_i\overline{z}_i\right) \tag{2.21}$$

Gower and Legendre [102] identified which binary similarity and distance measures are positive semi-definite in their symmetric similarity matrix.

2.6.3 Nominal simdist functions

Boriah *et al.* [27] put nominal similarity functions into three classes based on how they populate a similarity matrix for training data:

1. matrices with diagonal entries, and a minimum value of 0 uniformly in the off-diagonal entries (representing mismatches);
2. matrices with off-diagonal entries, and a maximum 1 uniformly in the diagonal entries (representing matches);
3. matrices with on and off-diagonal entries that are not uniform.

Boriah *et al.* [27] suggest that nominal similarity functions may also be classified based on: if they apply different weights based on the frequency of feature values, or the rationale for proposing the measure: probabilistic, information-theoretic, etc.

[9] azzoo [sic] stands for: alter zero zero one one

Cha *et al.* [49] defines eight families of distance measures for nominal type histograms, but without rationale or validation. These families are: Minkowski, L_1, intersection, inner product, fidelity (or squared-chord), $(L_2)^2$ or χ^2 (chi-squared), Shannon entropy and combination measures. They also cluster functions do not relate the clusters to the eight families.

It is confusing in Cha et al's [49] taxonomy that the Intersection measure is a positive-match measure in the Intersection family while the Jaccard distance, another positive-match measure, is in the Inner Product family, and the Jaccard coefficient is in the Fidelty or Squared-chord family.

2.6.4 Real simdist functions

Similarity functions between vectors of real-valued features are based on either proximity (the inverse of distance) or covariance (such as a dot product, i.e., inner product or projection). For real valued data proximity is derived from distance as an additive inverse [272] or a multiplicative inverse [166].

Kotsiantis *et al.* [148] review the following five distance measures for vectors of real valued features: the Manhattan distance based on the L_1-norm, the Euclidean distance based on the L_2-norm, the Minkowski distance based on the L_p-norm, the Chebyshev distance based on the L_∞-norm, the Canberra distance based on the ratio of the difference to the sum. Wilson and Martinez [289] review all of the same distance measures, as well as four additional measures: the Quadratic, Mahalanobis, Correlation and Chi-square distance measures. Whereas Euclidean distance assumes that features are equally important and/or have been normalized, the Mahalanobis distance generalizes Euclidean distance with a matrix of weights [268].

Cosine similarity is used in text information retrieval [223] while the cosine distance is found in audio [77] and image [302] retrieval; and the tangent distance [234] is used in image processing to achieve local invariance.

2.6.5 Ordinal simdist functions

There are six ordinal similarity measures in the literature: Spearman's rank correlation [300], Kendall's coefficient of concordance W, Kendall's rank correlation Tau, Goodman Kruskal Gamma, Lin's information theoretic ordinal similarity measure; and Podani's coefficient of similarity for ordinals.

There are also a variety of ordinal distance measures, where the first two distances correspond to the first two similarities: Spearman distance, Kendall's distance.

2.7 Kernels

This section on **kernels** is very mathematical in some spots—hence I note that the content of greatest relevance is this **introduction** and the terms used in the subsection on **kernel classes**. These two parts lay a foundation for Chapter 4 on kernel data modeling.

Kernels are similarity functions [233] used in SVM and other kernel methods (Glossary) in general. In this section I define kernels in general and types or classes of kernels.

In general, unless otherwise qualified, kernels usually refer to Mercer, valid or positive definite (p.d.) kernels (these terms are interchangeable)—where such kernels may be used in any kernel method. There are a wider set of kernels which apply to SVM, called admissible kernels which include p.d. and conditionally positive definite (c.p.d.) kernels. I define these terms further in the sections that follow.

A kernel (in the context of my thesis) refers to a similarity function that takes two inputs of any type (e.g., atomic data type or complex data type) and outputs a real-valued number that represents the similarity between the inputs, where the function must meet other specific requirements. A kernel must be symmetric (i.e., switching the inputs yields the same output) and must be either admissible or valid for its purpose.

In the literature [233], a kernel k is defined as a function of two inputs $\underline{x}$ and $\underline{z}$ (which may be non-numeric, i.e., not necessarily vectors) in an arbitrary domain, or input space, $\mathcal{X}$ (i.e., the inputs can be of any type) with a real output resulting from an inner product of a basis function $\phi : \mathcal{X} \to \mathcal{F}$ applied to each input, as in:

$$k_{\mathrm{M}}\left(\underline{x},\, \underline{z}\right) = \left\langle \phi\left(\underline{x}\right),\, \phi\left(\underline{z}\right) \right\rangle \qquad \underline{x}, \underline{z} \in X,\, \phi \in \mathcal{F},\, k \in \mathbb{R} \tag{2.22}$$

The term $\mathcal{F}$ refers to the feature space, and I may optionally put a subscript after the inner product to denote that the inner product occurs in the feature space, i.e., $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. Also, the function ϕ may be denoted and considered vectorial, as in $\underline{\phi}$, if the feature space $\mathcal{F}$ is a vector space like $\mathbb{R}^n$.

2.7.1 Mercer kernels

As previously stated, in general, unless otherwise qualified, kernels usually refer to Mercer, valid or positive definite (p.d.) kernels[10]—these terms are equivalent when applied to a kernel. Mercer kernels may be used in any kernel method.

Before I define or describe a Mercer kernel, I first provide the motivation or purpose for one.

In machine learning it is ideal when I can find the **global optimum** (maximum or minimum) as the solution to a **specific study/problem**. If I use a **Mercer, positive definite or valid kernel** then the global optimum solution is **guaranteed** to exist, and I will find that solution if I use **convex optimization** in kernel methods like SVM.

However, problem solving is a little bit more complicated than that. While the solution is **globally optimal** *for the specific study/problem and model posed*, the model may not be the optimal choice and the study/problem may not be optimally expressed. For example, the optimum found may be for a specific method (e.g., a support vector machine) and its hyperparameters (C) and a specific kernel (e.g., a Gaussian RBF kernel) and its hyperparameters (σ)—but it is not, in general, the best possible solution over all possible methods, kernels and hyperparameters. So the solution is globally optimal in a specific or local context.

In contrast, neural networks are **non-convex optimization** problems, so they find locally optimal solutions. There is no way to say that one approach is better than the other. One difference is that convex optimization is **repeatable and deterministic**, whereas non-convex optimization will arrive at a different results for different model weights, different optimization step sizes, different stopping conditions, etc. Also some consider it an advantage that convex optimization is more well studied.

Having motivated the need for a Mercer kernel, I now define the properties of a Mercer kernel that allow us to identify or create one. I identify a Mercer kernel as follows:

[10]Note: a positive definite (p.d.) kernel is distinguished from a positive semi-definite (p.s.d.) Gram matrix (or kernel matrix).

Mercer specified that a kernel is valid if and only if (iff) it is positive definite (p.d.) and iff it meets the following condition (Mercer's condition) [187, 233]:

$$\iint_{\mathcal{X}\times\mathcal{X}} k\left(\underline{x},\,\underline{z}\right) f\left(\underline{x}\right) f\left(\underline{z}\right) \mathrm{d}\underline{x}\mathrm{d}\underline{z} > 0$$
$$\underline{x},\underline{z} \in \mathcal{X},\; f \in L_2\left(\mathcal{X}\right),\; k \in \mathbb{R}, k \,\text{symmetric}$$

This definition (or condition) is not very intuitive compared to others that follow.

A kernel is Mercer if and only if its Gram matrix G is positive semi-definite for all possible finite data sets and all possible support vector coefficients α_i and labels y_i (where $c_i = \alpha_i y_i$) [233]:

"A necessary and sufficient condition for a function $k\left(\underline{x},\underline{x}'\right)$ to be a valid kernel [233] is that the Gram matrix [or kernel matrix]...whose elements are given by $k\left(\underline{x}_n,\underline{x}_m\right)$ should be positive semi-definite for all possible choices of the set $\{\underline{x}_n\}$." [24]
Alternatively stated: $\sum_{n=1}^{\ell}\sum_{m=1}^{\ell} c_n c_m k\left(\mathbf{x}_n,\mathbf{x}_m\right) \geq 0$ for any set of examples $\{\mathbf{x}_n\}$ and any reals $\{c_n\}$ [94]

The simplest route to create a Mercer kernel is as follows:

Any kernel defined with an explicit basis function $\underline{\phi}$ in

$$k\left(\underline{x},\,\underline{z}\right) = \left\langle \underline{\phi}\left(\underline{x}\right),\,\underline{\phi}\left(\underline{z}\right)\right\rangle \qquad \underline{x},\underline{z} \in X,\; \underline{\phi} \in \mathcal{F},\; k \in \mathbb{R} \tag{2.23}$$

is by definition [233] Mercer, valid and p.d.

Some kernels are defined without an explicit basis function $\underline{\phi}$ however—and for those kernels to be Mercer, the basis function must be known to exist but I do not have to know the basis function itself. It may remain implicit rather than explicit.

A stationary kernel, i.e., one defined in terms of a difference in inputs, $k\left(\underline{x}-\underline{z}\right)$ or $k\left(\|\underline{x}-\underline{z}\|\right)$ is valid and Mercer iff the values in the amplitude spectrum of the Fourier transform are non-negative.

A dot product kernel, i.e., one defined in terms of an inner product of the inputs, $k\left(\underline{x}^{\mathrm{T}}\underline{z}\right)$ is valid and Mercer iff the values in its Taylor/MacLauren series at zero are non-negative [243].

A recent addition to methods of proof for a Mercer kernel pertains to similarity matrices but it is problematic because of its dependency on data [280]—this is an area for potential future work.

A kernel is Mercer if it is constructed using certain rules (Tables 2.3, 2.2, 2.4) [24, 233].

Table 2.2: Rules to construct Mercer kernels from functions

Product of Functions [24, 94, 233] $k(\mathbf{x},\mathbf{z}) = f(\mathbf{x})\, f(\mathbf{z})$ for $f : X \to \mathbb{R}$
Function Arguments [24, 52, 233] let $k(\mathbf{x},\mathbf{z}) = k_3(\phi_1(\mathbf{x}), \phi_1(\mathbf{z}))$ where $\phi_1 : X \to \mathbb{R}^p$ and $k_3 : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$
Covariance [94] $k(\mathbf{x},\mathbf{z}) = \frac{1}{4}\left[f(\mathbf{x}+\mathbf{z})\, f(\mathbf{x}-\mathbf{z})\right]$ where f is a variance function i.e., $f : X \to \mathbb{R}$, $f \geq 0$, $\min(f) = 0$
Kernel Substitution (based on[233]) given $k_1(\mathbf{x},\mathbf{z}) = f\left(\langle\mathbf{x},\mathbf{x}\rangle,\ \langle\mathbf{x},\mathbf{z}\rangle,\ \langle\mathbf{z},\mathbf{z}\rangle\right)$ let $k(\mathbf{x},\mathbf{z}) = f\left(k_1(\mathbf{x},\mathbf{x}),\ k_1(\mathbf{x},\mathbf{z}),\ k_1(\mathbf{z},\mathbf{z})\right)$

Table 2.3: Rules to construct Mercer kernels from Mercer kernels

Scaling [24, 233] $k(\mathbf{x}, \mathbf{z}) = c \cdot k_2(\mathbf{x}, \mathbf{z})$ for $c \epsilon \mathbb{R}^+,\ n \epsilon \mathbb{N}$
Sum [24, 94, 233] $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$ or $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}_a, \mathbf{z}_a) + k_2(\mathbf{x}_b, \mathbf{z}_b)$ where $\mathbf{x}_a, \mathbf{x}_b \subseteq \mathbf{x}$ not necessarily disjoint and $\mathbf{z}_a, \mathbf{z}_b \subseteq \mathbf{z}$ not necessarily disjoint
Product [24, 94, 233] $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$ or $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}_a, \mathbf{z}_a) k_2(\mathbf{x}_b, \mathbf{z}_b)$ where $\mathbf{x}_a, \mathbf{x}_b \subseteq \mathbf{x}$ not necessarily disjoint and $\mathbf{z}_a, \mathbf{z}_b \subseteq \mathbf{z}$ not necessarily disjoint
Polynomial [24, 94, 233] $k(\mathbf{x}, \mathbf{z}) = \sum_n c_n k_1(\mathbf{x}, \mathbf{z})^n$ for $c_n \epsilon \mathbb{R}^+,\ n \epsilon \mathbb{N}$
Exponential [24, 94, 233] $k(\mathbf{x}, \mathbf{z}) = \exp(k_1(\mathbf{x}, \mathbf{z}))$
Cosine (feature space) normalization [18] $k(\mathbf{x}, \mathbf{z}) = \frac{k_1(\mathbf{x}, \mathbf{z})}{\sqrt{k_1(\mathbf{x}, \mathbf{x}) k_1(\mathbf{z}, \mathbf{z})}}$

Table 2.4: Rules to construct Mercer kernels from matrices

Kernel Affine Transformation [24, 94, 233] $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T B \mathbf{z}$, where B is symmetric positive semi-definite.

2.7.2 Gram matrices

To learn from data, kernel methods apply the kernel to every pair of instances in training data with replacement, to obtain a Gram matrix G (also called a kernel matrix) which is symmetric (conjugate symmetric to be specific).

A kernel is Mercer iff its Gram matrix G is positive semi-definite (p.s.d.) for all possible finite data sets. The practical implication of the p.s.d property is that the quadratic formulas used for optimization in machine learning methods have a minimum when the Gram matrices are either p.d. or p.s.d. (Figure 2.5). Quadratics allow convex optimization (semi-definite programming), whereas if I try to minimize a function that isn't convex, my result or solution may be a local minimum rather than the global minimum.

A Gram matrix is positive semi-definite if and only if the eigenvalues of the Gram matrix are all non-negative.

A Gram matrix is positive semi-definite if $K = X^T X$ for $X \in \mathbb{R}$, i.e., I can find the matrix square root.

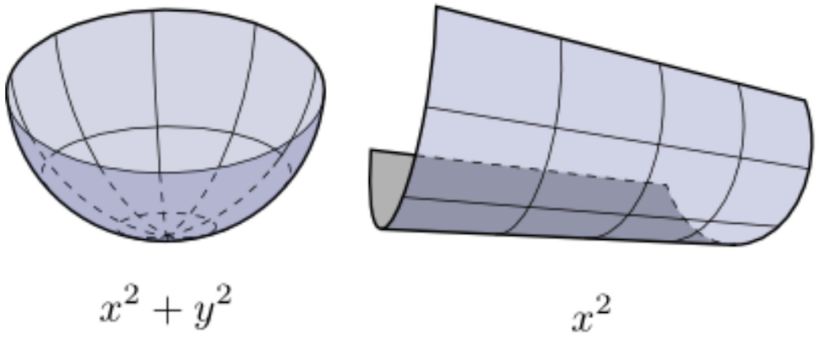

Figure 2.5: Positive definite (left) and positive semi-definite (right) forms are convex with global and local minima

2.7.3 Conditionally positive definite kernels

A conditionally positive definite (c.p.d.) [30, 226] kernel is admissible [229, 241, 304], i.e., only valid for specific methods such as SVM which shares the c.p.d. condition or constraint $\sum_i c_i = 0$ because SVM imposes Karush-Kuhn-Tucker (KKT) [233] conditions. It is that condition which differentiates a c.p.d. kernel from a Mercer kernel. Boughorbel *et al.* [30] prove a kernel is c.p.d. by imposing the condition on the kernel's formula and then algebraically showing that under that condition the formula meets the (second) definition of a Mercer kernel.

As a finer point, I observe that a Mercer or p.d. kernel is still Mercer or p.d. even if that constraint is imposed (it does not contradict the definition) hence all p.d. kernels are c.p.d. as well, but the literature and henceforth my thesis, only refer to c.p.d. kernels as those which are not p.d., for the sake of clarity. When creating kernels for SVM, it may be useful to know that negating a negative definite (n.d.) kernel, results in a c.p.d. kernel.

2.7.4 Kernel classes

For kernel methods, the first question regarding a kernel is whether or not I can use it in SVM—i.e., is it **admissible** [229, 241, 304] (Table 2.6)? **Admissible kernels** [229, 241, 304] include **Mercer or positive definite (p.d.)** kernels (2.7.1) as well as **conditionally positive definite (c.p.d.)** kernels (the previous section). This provides a functional overview of kernel classes for SVM.

A definitional or theoretical perspective on kernel classes for SVM, describes three disjoint kernel classes at the top-level (Table 2.6).

I now return to a functional perspective on kernels—this time with more detail (Tables 2.7 and 2.8). Kernels are not only p.d. or c.p.d., from the previous definitional or theoretical

Table 2.5: A functional overview of kernel classes for SVM is provided by admissible kernels. Terms in bold are used often in my thesis. Kernel classes at the same indentation level are disjoint. *Note: the term "valid" in the literature is a misnomer given that c.p.d. kernels are valid for use in SVM.

Admissible[242]
Mercer or **positive definite (p.d.)** [187], a.k.a valid*
Conditionally positive definite (c.p.d.) [242]
Not admissible

Table 2.6: A definitional or theoretical overview of kernel classes is provided by the three classes Mercer [187] defines. Terms in bold are used often in my thesis. Kernel classes at the same indentation level are disjoint. *Note: in the context of SVM, the term "valid" is a misnomer since c.p.d. kernels are also admissible [229, 241, 304] for SVM.

Positive definite (p.d.) [187] or **Mercer**, a.k.a. valid*	
Negative definite [187] (n.d.)	
Indefinite [187]	
	Conditionally positive definite (c.p.d.) [242]
	Conditionally negative definite (c.n.d.) [191]
	Others

perspective, they often also belong to other kernel classes such as **dot product kernels**, **stationary kernels**, or **separable kernels**.

The linear, polynomial and sigmoid kernels are dot product kernels of the form $k\left(\underline{x}^{\mathrm{T}}\underline{z}\right)$ (Section 3.4), while the Gaussian RBF and inverse multiquadric kernels are stationary (or proximity) kernels of the form $k\left(\underline{x}-\underline{z}\right)$[11] (Section 3.4).

I note that dot product kernels and stationary kernels are not necessarily Mercer kernels. For example, the sigmoid kernel is a dot product kernel which is neither Mercer nor admissible (except sometimes). The power distance and log kernels are stationary kernels which are admissible but not Mercer, while the multiquadric kernel is a stationary kernel that is neither Mercer nor admissible.

In this section, I describe classes of kernels which have a precise mathematical definition. I do not include descriptions or descriptors such as *transparent kernels* (4.3). The list is not exhaustive since new kernel classes and kernels emerge in the literature now and then.

Commonly used kernels operate on numeric data, where some of that data may have been converted to numeric data types from other data types (Section 2.5 and Glossary A.8 on page 187). I review common kernels and a selection of uncommon kernels in the sections that follow.

[11] I am being precise while using the most popular term: stationary. Isotropic stationary kernels of the form $k\left(\|\underline{x}-\underline{z}\|\right)$ are sometimes inaccurately (or incompletely) called stationary, when they are in fact isotropic stationary. Isotropic stationary kernels only care about the magnitude of difference (not its direction or phase), while stationary kernels in the most general sense ***may*** also consider the direction or phase.

Table 2.7: A hierarchy of kernel classes, where classes at the same indentation level are disjoint. Classes in **bold** are referred to throughout my thesis.

Dot product [243]
Locally stationary [94]
Stationary [94]
Homogeneous or isotropic stationary [94]
Compactly supported [94]
Anistropic stationary [94]
Exponentially convex [94]
Locally stationary reducible [94]
Stationary reducible [94]
Separable or separable non-stationary [94]
Generalized [94]

Table 2.8: Other noteworthy kernel classes (not *disjoint*) including my proposed class: ***Explicit Mercer***. Classes in **bold** or ***bold italic*** are used throughout my thesis.

Separable [121, 279]
Separable non-stationary [94]
Additive [179]
Explicit Mercer (4.4)
Composite [44]
Convolution [118]
Probabilistic [248]
Gamma homogeneous [94]

2.8 Common kernels

There are four kernels which are commonly used in the literature: the linear, polynomial, Gaussian radial basis function (RBF) and sigmoid kernels. They are used with a variety of data types converted to numeric data types. In this thesis my interests pertain to atomic data types (4.2)—not complex data types.

I also describe minor variations of these kernels, some of which I use in selected experiments—e.g., in specific cases it makes sense to include and analyze the effect of kernel width (or horizontal scale), which requires small modifications to the linear and polynomial kernels. Also the normalized sigmoid kernel is usually more accurate than the sigmoid kernel.

2.8.1 Linear kernel

The most general definition of a linear kernel [18, 24] is:

$$k_{\mathrm{Lin},c'}\left(\underline{x},\underline{z}\right)=\underline{x}^{\mathrm{T}}\underline{z}+c' \qquad c'\geq 0,\ c'\in\mathbb{R}$$

where the hyperparameter c is a vertical shift (or bias) in the kernel output. However, implementations such as Matlab use:

$$k_{\mathrm{Lin}}\left(\underline{x},\underline{z}\right)=\underline{x}^{\mathrm{T}}\underline{z}$$

Furthermore, in some experiments it is important to vary the kernel width, so the following variation is also a linear kernel:

$$k_{\mathrm{LinS}}\left(\underline{x},\underline{z}\right)=k_{\mathrm{Lin}}\left(\frac{\underline{x}}{\sigma},\frac{\underline{z}}{\sigma}\right)$$

Hence not all linear kernels are the same.

2.8.2 Polynomial kernel

The most general definition of a polynomial kernel (e.g., libsvm) [18, 24, 148, 152] is:

$$k_{\mathrm{Polyc}}\left(\underline{x},\underline{z}\right)=\left(\underline{x}^{\mathrm{T}}\underline{z}+c\right)^{d'} \qquad c\geq 0,\ d'\geq 2,\ c\in\mathbb{R},\ d'\in\mathbb{N}$$

However, the most popular implementation (e.g., Matlab, Weka using lower order terms) [18] is:

$$k_{\text{Poly}}\left(\underline{x},\underline{z}\right)=\left(\underline{x}^{\text{T}}\underline{z}+1\right)^{d'}\qquad d'\geq 2,\ d'\in\mathbb{N}$$

While the homogeneous polynomial kernel is simply (e.g., Weka default):

$$k_{\text{hPoly}}\left(\underline{x},\underline{z}\right)=\left(\underline{x}^{\text{T}}\underline{z}\right)^{d'}\qquad d'\geq 2,\ d'\in\mathbb{N}$$

The hyperparameter c is a balance parameter that determines the relative weight of the interaction terms, $x^p z^q$ for $p,\ q\in\mathbb{N},\ p,\ q\leq d$. A higher value of c emphasizes lower order interaction terms, and vice-versa, while $c=0$ omits interaction terms.

2.8.3 Gaussian RBF kernel

The definition of a Gaussian radial basis function (RBF) kernel [18, 24, 148, 152, 233], also called the RBF kernel or squared exponential kernel [247] is:

$$k_{\text{RBF}}\left(\underline{x},\underline{z}\right)=\exp\left(\frac{-\left\|\underline{x}-\underline{z}\right\|_2^2}{2\sigma^2}\right)\qquad \sigma>0;\ \sigma\in\mathbb{R} \tag{2.24}$$

The hyperparameter σ specifies the width of the Gaussian kernel. Sometimes it is equivalently defined in terms of a parameter γ instead:

$$k_{\text{RBF}}\left(\underline{x},\underline{z}\right)=\exp\left(-\gamma\left\|\underline{x}-\underline{z}\right\|_2^2\right)\quad \gamma=\frac{1}{2\sigma^2} \tag{2.25}$$

Note that related kernels—the exponential and Laplacian kernels—are defined without a factor of two. For interpretability of the kernel width relative to the data I must pay attention to the inclusion or exclusion of any factor. Some platforms (e.g., Matlab version 2017 and up) use the parameter, kernel scale s, which is related to the interpretable kernel

width by $s = \sqrt{2}\sigma$:

$$k_{\text{RBF}}(\underline{x}, \underline{z}) = \exp\left(-\left\|\frac{\underline{x}}{s} - \frac{\underline{z}}{s}\right\|_2^2\right) \quad s = \sqrt{2}\sigma;\ \sigma > 0;\ \sigma \in \mathbb{R}$$
$$= \exp\left(\frac{-\|\underline{x} - \underline{z}\|_2^2}{s^2}\right)$$

2.8.4 Sigmoid kernel

The sigmoid kernel [24, 40, 167], also called a multilayer perceptron (MLP) kernel or hyperbolic tangent kernel, is **sometimes** conditionally positive definite, but not always (it depends on the hyperparameter values relative to the data), and is defined by:

$$k_{\text{Sig}}(\mathbf{x}, \mathbf{z}) = \tanh\left(a' \cdot \underline{x}^{\text{T}}\underline{z} + r\right) \qquad a' > 0,\ r < 0;\ a', r \in \mathbb{R} \tag{2.26}$$
$$= \tanh\left(a' \cdot \sum_{i=1}^{p}\{x_i z_i\} + r\right) \tag{2.27}$$

The hyperparameter a' is a horizontal scaling parameter, and the hyperparameter r determines a vertical shift in the centre of the kernel's geometry. I define a normalized sigmoid kernel [45] as:

$$k_{\text{SigN}}(\underline{x}, \underline{z}) = k_{\text{Sig}}\left(\frac{\underline{x}}{\sqrt{n}}, \frac{\underline{z}}{\sqrt{n}}\right), \quad \underline{x} \in \mathbb{R}^n$$

and subsequently found a source in the literature that stated a good value for a' is $\frac{1}{n}$ (which is equivalent), as corroboration, but I have not been able to find or identify the source. The rationale for the normalization is simple: since a sigmoid saturates values (similar to top coding, but smoothly), the saturation should not depend on the number of dimensions in the data, but on the values in the data instead—otherwise with a large number of dimensions the kernel will always saturate and fail to operate as intended.

2.9 Uncommon kernels

The selection of kernels which follow are not as commonly used in the literature but they are either noteworthy for various reasons and/or the next most likely set of kernels which a user will encounter. Since the scope and objective of my thesis pertains only to atomic data types (4.2), I do not include kernels made for complex data types.

2.9.1 Notable kernels for experiments

I use the following uncommon kernels in experiments: (1) the **power distance** kernel as the most intuitive stationary kernel with high accuracy; (2) the **logarithmic** kernel as a related stationary kernel with high accuracy; and (3) the **inverse multiquadric** kernel which has a theoretical basis in three-dimensional interpolation.

Of the remaining ten kernels, eight are for real data types and two (the Delta and Hamming kernels) are for other data types.

I exclude the Circular and Spherical kernels because they are only defined for two and three dimensions respectively, hence not applicable to most studies/problems. I exclude the Mahalanobis kernel, because it only applies to data which are not already centered and normalized, in contradiction to the standard practice in machine learning to center and normalize data.

Power distance kernel

The power distance kernel [30, 226] is conditionally positive definite and is defined by:

$$k_{\text{Pwr}}\left(\underline{x}, \underline{z}\right) = -\left\|\underline{x} - \underline{z}\right\|_2^{\beta} \qquad 0 < \beta \leq 2,\ \beta \in \mathbb{R}$$

where β is an exponent. This kernel is notable, when $\beta = 1$, i.e., $k_{\text{Pwr1}}\left(\underline{x}, \underline{z}\right) = -\left\|\underline{x} - \underline{z}\right\|_2$ as the most intuitive distance kernel (as a pure Euclidean distance)—more intuitive than the Gaussian RBF kernel. It performs as well or better than the Gaussian RBF kernel in accuracy with most data sets. This kernel is also notable, when $\beta = 2$, i.e., $k_{\text{Pwr2}}\left(\underline{x}, \underline{z}\right) = -\sum_{i=1}^{n}\left(x_i - z_i\right)^2$, as perhaps the only stationary kernel in the literature that is additive in the differences within each feature—i.e., the feature dimensions are not confounded.

Logarithmic kernel

The logarithmic kernel[30, 226] for object classification (in images) is conditonally positive definite and is defined by:

$$k_{\text{Log}}\left(\underline{x}, \underline{z}\right) = -\log\left(1 + \left\|\underline{x} - \underline{z}\right\|_2^{\beta}\right) \qquad 2 \geq \beta > 0,\ \beta \in \mathbb{R}$$

where β is an exponent within the logarithm. It performs as well or better than the Gaussian RBF kernel in accuracy with most data sets.

Inverse multiquadric kernel

The Inverse multiquadric kernel [142] is a positive definite kernel:

$$k_{\text{IMQ}}\left(\underline{x},\underline{z}\right)=\frac{1}{\sqrt{\left\|\underline{x}-\underline{z}\right\|^{2}+\theta}}\qquad\theta>0;\,\theta=\sigma^{2};\,\theta\in\mathbb{R}\tag{2.28}$$

This kernel has a theoretical foundation in the interpolation of three-dimensional topologies, e.g., for map-making and it performs well in accuracy. How does this relate to classification? Classification is, in most cases a threshold applied to a regression study/problem, which in turn is simply interpolation, extrapolation and/or smoothing of a function (if these terms are familiar to the reader)—hence the relevance.

2.9.2 Uncommon kernels likely encountered

The following set of uncommon kernels are the **next most likely** one will encounter and they are all positive definite: the **exponential** (2.29) [94], **Laplacian** (2.30) [87, 98, 212], **rational quadratic** (2.31) [94] and **Gaussian kernel with a generalized distance** (2.32) [24, 196]. Note that for interpretability purposes I have added the clarification that $\theta=\sigma^{2}$.

$$k_{\text{Exp}}\left(\underline{x},\underline{z}\right)=\exp\left(\frac{-\left\|\underline{x}-\underline{z}\right\|_{2}}{\theta}\right)\quad\theta>0;\,\theta=\sigma^{2};\,\theta\in\mathbb{R}\tag{2.29}$$

$$k_{\text{Lap}}\left(\underline{x},\underline{z}\right)=\exp\left(\frac{-\left\|\underline{x}-\underline{z}\right\|_{1}}{\theta}\right)\quad\theta>0;\,\theta=\sigma^{2};\,\theta\in\mathbb{R}\tag{2.30}$$

$$k_{\text{RQ}}\left(\underline{x},\underline{z}\right)=1-\frac{\left\|\underline{x}-\underline{z}\right\|^{2}}{\left\|\underline{x}-\underline{z}\right\|^{2}+\theta}\qquad\theta>0;\,\theta=\sigma^{2};\,\theta\in\mathbb{R}\tag{2.31}$$

The hyperparameter θ is a kernel width parameter and β is an inverse square of kernel width parameter.

$$k_{\text{RBFgen}}\left(\underline{x},\underline{z}\right)=\exp\left(\frac{-d\left(\underline{x},\underline{z}\right)^{2}}{2\sigma^{2}}\right)\quad\sigma>0;\;\sigma\in\mathbb{R}\tag{2.32}$$

$$=\exp\left(-\gamma\right),\;\;d_{2}\left(\underline{x},\underline{z}\right)=\left\|\mathbf{x}-\mathbf{z}\right\|_{2}\quad\gamma>0;\;\gamma\in\mathbb{R}\tag{2.33}$$

$$\gamma=\frac{1}{2\sigma^{2}}\tag{2.34}$$

Following these uncommon kernels there are many more kernels for numeric data (not

complex data types) which one **may** encounter, which may or may not be admissible: the exponential dot-product [88, p.7], etc.

2.9.3 Transparent kernels

These uncommon kernels are not as likely to be encountered because to find them requires a keen interest in searching diligently for kernels which are separable and/or interpretable.

Separable kernels are explicitly of the form $k(\underline{x}, \underline{z}) = f(\underline{x})\, g(\underline{z})$ [279]. They have lower space complexity, since only the vectors $\underline{x}$ and $\underline{z}$ need to be stored instead of the kernel matrix [94]. Genton shows how to reduce some kernels to separable kernels and asserts this benefit as crucial for big data [94]. However Genton is referring to kernels of rank 1, which are likely to be insufficient for a number of classification studies/problems.

Separable kernels include: the linear, Hellinger [63, 120, 282], Hellinger exponential [266], wavelet [304], generalized histogram intersection [180], chi-square [282], probability product [68, 137] Bhattacharyya [68, 136], expected likelihood [68, 136], homogeneous polynomial [229] and **truncated Gaussian RBF** kernels [278]. The last is defined as follows:

$$k_{\text{tRBF}}(\underline{x}, \underline{z}) = \frac{2}{n(n-1)} \sum_{p=1}^{n} \sum_{q>p}^{n} k_{\text{RBF}} \left(\begin{bmatrix} x_p \\ x_q \end{bmatrix}, \begin{bmatrix} z_p \\ z_q \end{bmatrix} \right)$$

2.9.4 Kernels with other notable features

The **circular** (2.35) and **spherical** (2.36) kernels [94] are notable for their compact support, but I exclude them from experiments because they are only defined for two and three features in a data set respectively.

$$k_{\text{Circ}}(\mathbf{x}, \mathbf{z}) = \begin{cases} \frac{2}{\pi} \arccos\left(\frac{\|\mathbf{x}-\mathbf{z}\|}{\theta}\right) - \frac{2}{\pi}\left(\frac{\|\mathbf{x}-\mathbf{z}\|}{\theta}\right) \sqrt{1 - \left(\frac{\|\mathbf{x}-\mathbf{z}\|}{\theta}\right)^2} & \text{if } \|\mathbf{x}-\mathbf{z}\| < \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.35}$$

$$k_{\text{Sph}}(\mathbf{x}, \mathbf{z}) = \begin{cases} 1 - \frac{3}{2}\left(\frac{\|\mathbf{x}-\mathbf{z}\|}{\theta}\right) + \frac{1}{2}\left(\frac{\|\mathbf{x}-\mathbf{z}\|}{\theta}\right)^3 & \text{if } \|\mathbf{x}-\mathbf{z}\| < \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.36}$$

The **non-diagonal Mahalanobis** kernel k_{ndM} and **diagonal Mahalanobis** kernel k_{dM} [1] are notable because the Mahalanobis distance is notable—however, like metric learning, I

exclude it from experiments due to lack of interpretability. For a two-class study/problem only one class k is used to define each kernel using the following approximations [1]:

$$
\begin{aligned}
k_{\mathrm{ndM}}(\mathbf{x}, \mathbf{z}) &\doteq \exp\left(-\frac{\delta}{m}(\mathbf{x}-\mathbf{z})^{\mathrm{T}} Q^{-1}(\mathbf{x}-\mathbf{z})\right) \\
k_{\mathrm{dM}}(\mathbf{x}, \mathbf{z}) &\doteq \exp\left(-\frac{\delta}{m}(\mathbf{x}-\mathbf{z})^{\mathrm{T}} Q_{\mathrm{diag}}^{-1}(\mathbf{x}-\mathbf{z})\right) \\
Q_{\mathrm{diag}} &= \operatorname{diag}(Q) \\
Q &= \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_i-\mathbf{c})(\mathbf{x}_i-\mathbf{c})^{\mathrm{T}} \qquad \mathbf{x}_i \mid y_i = k \\
\mathbf{c} &= \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i \qquad \mathbf{x}_i \mid y_i = k
\end{aligned}
$$

The non-diagonal Mahalanobis kernel applies a metric that centers, normalizes and whitens (decorrelates) the data. Centering and normalizing are redundant of the standard pre-processing done in machine learning. The third aspect however, which whitens or decorrelates data, is problematic for transparency. Whitening involves a matrix transformation, that, like principal components analysis (PCA), and many other matrix transforms (discussed in Appendix A.3) yields features which are not inherently interpretable nor clinically meaningful.

Matrix transformed features inform us about combinations of features that matter, such as: 0.7 * systolic blood pressure + weight - 0.2 * age, but they do not tell us what the combination means or represents—and any interpretation that I apply is post-hoc unless separately validated, e.g., by an interventional study.

Decisions made in a feature space of transformed features are semi-transparent. I can either fully understand how the method and kernel behave in the feature space, but not fully understand the transformed features there, or I can view things in the input space where I fully understand the features, but cannot fully understand the behaviour of the method and kernel.

Since this approach is semi-transparent, I observe that it is an alternative approach to my work, however I do not pursue it further. Another alternative is to perform these matrix transformations as pre-processing to the methods that I propose. The same reasoning applies to metric learning. That is: the central goal of my work is to maximize transparency and observe if I can achieve the same, similar, or ideally better accuracy as less transparent methods and models. I seek metrics which are simple, sparse, derived, or justified/traced

to requirements for transparency and interpretability, **as** my **starting point**, rather than seeking maximum accuracy first.

With the diagonal Mahalanobis kernel, Abe [1] achieved the same accuracy as a Gaussian RBF kernel, but consistently with a smaller cost of error C. Since C regulates the trade-off between the two objectives—maximizing the margin and minimizing error—the diagonal Mahalanobis kernel allows for a potentially larger margin, potentially resulting in greater separability of data, greater probability of class membership and lower probability of error for an individual prediction. The Gaussian RBF kernel is a special case of the diagonal Mahalanobis kernel with $Q_{\text{diag}}^{-1} = \gamma \cdot I = \frac{1}{2\sigma} \cdot I$.

2.9.5 Non-numeric or heterogeneous kernels

I describe three Mercer kernels for non-numeric data or heterogeneous (or mixed) data with atomic data types (4.2)—however two of them are flawed in how they weight nominal data and the last uses the linear kernel for real data types, which is suboptimal.

The **delta kernel** [94] is a Mercer kernel that applies to discrete data types such as binary, nominals, ordinals or integers:

$$\begin{aligned} k\left(\underline{x}, \underline{z}\right) &= \delta\left(\underline{x}, \underline{z}\right) \\ &= \prod_{i=1}^{p} \delta\left(x_i, z_i\right) \\ &= \begin{cases} 1 & \text{if } \underline{x} = \underline{z} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

It only measures similarity as two levels: equal or not equal.

The **Hamming** kernel [8] is a Mercer kernel for n-dimensional vectors of binary $\mathbb{B}$ and/or nominal (multinomial) $\mathbb{M}$, features:

$$\begin{aligned} k_{\text{Ham}}\left(\underline{x}, \underline{z}\right) &= \sum_{i=1}^{n} \delta\left(x_i,\ z_i\right) \\ \delta\left(x_i,\ z_i\right) &= \begin{cases} 1 & \text{if } x_i = z_i \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

For multiple nominal features with different levels, or a mixture of binary and nominal features, it does **not** properly weight the features, because it gives the same weight to each level within a nominal feature as it does a binary feature.

Aradhye and Dorai [8] use the Hamming kernel in Kernel Principal Components Analysis (KPCA) [228] and feed that output to SVM for two classification tasks. They achieve comparable accuracy in one case and less accuracy in the other—identifying as a benefit the ability to train with less data points.

Aradhye and Dorai [8] also create a Mercer kernel for binary $\mathbb{B}$, or nominal (multinomial) $\mathbb{M}$ and real $\mathbb{R}$, features which they call a **Hamming-Euclidean hybrid** kernel:

$$k_{\text{HE}}\left(\underline{x},\underline{z}\right)=\alpha\cdot k_{\text{Ham}}\left(\underline{x}_{\{\mathbb{B},\mathbb{M}\}},\underline{z}_{\{\mathbb{B},\mathbb{M}\}}\right)+\beta\cdot k_{\text{Lin}}\left(\underline{x}_{\mathbb{R}},\underline{z}_{\mathbb{R}}\right)\qquad\underline{x}=\begin{bmatrix}\underline{x}_{\{\mathbb{B},\mathbb{M}\}}\\ \underline{x}_{\mathbb{R}}\end{bmatrix},\underline{z}=\begin{bmatrix}\underline{z}_{\{\mathbb{B},\mathbb{M}\}}\\ \underline{z}_{\mathbb{R}}\end{bmatrix}$$

Since the Hamming kernel is a component of this kernel, and since the Hamming kernel does **not** properly weight features, this kernel is similarly flawed. I note that Hamming refers to the type of similarity (not the space) for binary and nominal data types, whereas Euclidean refers to the space of the real data types.

There are other more seriously flawed kernels found in the literature for nominals in SVM [59, 60] which lack proof of admissibility (p.d. or c.p.d.) and therefore **importantly** should not be used, and in particular, cannot be trusted for use with clinical data. Admissibility is an essential requirement for kernels. Perhaps in the future the obscure branch of research on non-positive kernels for SVM [199] will be fruitful, however it is not trusted by the SVM community at large, at this time.

2.10 Other kernel topics

For a general sense of completeness I briefly describe other topics with kernels: kernel learning, multiple kernel learning and two-stage kernel learning.

Kernel learning is: selecting a kernel from a pre-defined set, by optimizing the error (empirical risk) on the training set [256], where the pre-defined set may consist of one kernel with different hyperparameters, or multiple kernels.

General types of kernel learning, i.e., approaches, include: cross-validation (most common), filters [256], wrappers [256], embedded methods [256] and hyperkernels [200]. Specific methods of kernel learning are as follows with the approach denoted in brackets:

- The cross-validation approach (the common/standard approach with SVM) [24]
- Multiple kernel learning [99, 155, 296] and two-stage learning [54] (wrapper)

- Composite kernel learning (embedded) [256]
- Data dependent kernels (filter approach) [256]
- Non-parameteric kernel learning (embedded) [256]
- Hyperkernels [200]

Filters adjust a kernel to the data before SVM/optimization. Wrappers perform nested optimization, with SVM optimization on the inside. Embedded methods optimize with respect to the SVM and kernel hyperparameters at the same time.

I use the **cross-validation approach** [24] (Section 5.2) to kernel learning primarily since it is widely-available on machine learning platforms. The wrapper approach is better than cross-validation, but not used because my objective is to facilitate widespread adoption of machine learning techniques.

In machine learning literature there are a good number of custom objective functions which are not implemented or validated—this is what the embedded approach requires. Custom objective functions involve programming and optimization logic with greater complexity and risk of error than kernels.

Multiple kernel learning (MKL), like kernel learning in general, is usually applied to numeric data types. Lanckriet *et al.* [155] introduced MKL, although the term MKL was introduced by subsequent authors. With MKL, one specifies multiple kernels in matrix form and the method finds an optimal linear combination of them. The title of the original paper implies that the kernel matrix is learned — but it is only partially-learned based on pre-specified kernels.

MKL formulates the study/problem and dataset into a quadratic problem (e.g., minimize the squared loss function) with specific quadratic constraints and an optimal solution is found in that context. Note that optimality is always context-specific — i.e., optimal for a specific objective function with specific constraints. Lanckriet *et al.* identified learning with data from multiple sources (heterogeneous data) as one of the key applications of this technique, although that seemed to be more of an afterthought than an initial motivation.

Ye *et al.* [296] applied MKL to integrate data from multiple sources (heterogeneous data) in the context of Alzheimer's Disease and demonstrated a marked improvement in the sensitivity and specificity of classification as compared with the use of one data type or mode. They also were able to identify new features: regions of the brain that showed up as

significant biomarkers in multimodal image data; and combine feature selection algorithms with MKL.

Composite kernels are defined in [44] and are used to instantiate kernels which classify two different types of real-numbered data (in images). **Composite kernel learning** (CKL) [256] generalizes that concept further including an approach to classify or regress data with heterogeneous data types, structures or channels. Szafranski *et al.* [256] demonstrate the concept with a multichannel (EKG) classification problem. Rakotomamonjy also develops CKL. CKL also uses existing kernels and therefore also suffer from the same shortcomings of common kernels, but also provides additional information like MKL. I use and develop composite kernels in my work, where the components are meaningfully related to data types.

Two-stage kernel learning includes work by Cortes *et al.* [54]. In standard SVM the user selects the kernel and specifies the range of hyperparameters, whereas in this approach a family of kernels is specified instead—as in, hyperkernels, Gaussian kernel families and non-linear kernel families.

Until Cortes *et al.*, two-stage learning did not achieve better accuracy than a uniform set of multiple kernels (i.e., the simplest form of MKL) and/or MKL in general. Cortes *et al.* performed better than MKL on all of the data sets they tested.

In the two-stage approach, the first stage **learns** a kernel that is a convex combination of p kernels. The second stage then performs traditional kernel-based learning with that kernel. Alternatively the steps can be swapped. Cortes determined a margin bound with a logarithmic dependency on the number of kernels p. Kernels are learned (optimized) according to a measure of kernel alignment between the kernel matrix and the target matrix. They provide a modified version of kernel alignment called centered alignment which behaves correctly in a particular case where traditional kernel alignment is faulty.

There are **data-dependent** kernels such as the Fisher and practical Fisher kernels [215, 233] which are based on Fisher information (statistics) calculated on the data set, however these methods require advanced statistical work and understanding to setup and use with confidence and validation—a challenge that is aggravated by the lack of reference implementation for the kernel. Another data-dependent kernel, the quotient-based kernel suffers from the same challenge—the lack of a reference implementation despite requests to the authors. We propose four data-dependent kernels—two based on kernel density estimates and two based on the binormality assumption with reference implementations available upon request.

Chapter 3

Study/problem formulation

For classification in health care, epidemiologists and clinicians want methods and models which provide:

1. the best discrimination or accuracy [106, 250], i.e., least error
2. good calibration (or fit) [250, 106], i.e., non-systematic error
3. the probability of error for individual predictions [250]
4. an understanding of how they work (i.e., transparency) [21, 106] for interpretation, explanation and/or justification; and
5. an understanding of how the data (as features or instances) are used to make predictions [21, 106] for interpretation and explanation.
6. parsimony [244], i.e., among sufficient models (e.g., equally accurate models) the simplest is preferred.

My thesis seeks models or kernels with **understanding** and **interpretation** coincident with high (or best achievable) accuracy. If that cannot be achieved then it seeks to understand when and how any **trade-off between interpretability** and **accuracy** occurs.

Understanding and interpreting support vector classification (SVM) and kernels, is important for clinicians to be able to trust, use, explain, justify or advocate their results and methods to colleagues and patients [21, 106].

3.1 Insufficient rationale for kernel selection

I review 22 academic papers (Appendix A.7) which conduct experiments using SVM in health care to observe the rationale given for using common kernels with atomic data types (4.2). The review includes eight papers on SVM for health care in general and 14 papers on SVM for melanoma detection.

Of the 22 papers reviewed, sixteen papers or **73% do not provide any rationale** for the kernels they use, nor a citation from which a rational could be inferred. Of the other six papers, two do not state any rationale but provide citations as a weak proxy, while the last four provide brief rationale. Hence, there is a **lack of rationale for kernel selection**.

Furthermore, fourteen of the 22 papers or **64% use a single kernel** which is problematic if the kernel is not suited to the data and study/problem—e.g., if other kernels achieve higher accuracy, or achieve the same or similar accuracy with better interpretability. There is a **lack of rationale for kernel selection**.

In some cases, **kernel selection** is based on **popularity** across domains, which is **not a sound strategy**, since a kernel may perform well/best for domains/data, but have inferior performance for others. The **No Free Lunch theorem** [72] is often cited as a justification for this observation, although that theorem only applies when **all possible data sets** for a set of data types are considered, which includes a great many nonsensical and implausible data sets that do not look like the distribution of data in the population being studied.

3.2 The status quo

The kernels most commonly used—i.e., the linear, Gaussian RBF, polynomial and sigmoid kernels—are those which are built-in to most SVM tools, with linear or RBF as the default. They seem to be commonly used because they are readily available, and four of the papers reviewed do not even indicate which kernel they use (hence probably the default).

Given that better performance can be achieved with other kernels aside from the Gaussian RBF, such as my **Mercer sigmoid** kernel, on some data sets [45, 295] with statistical significance [45], there is a **lack of rationale for kernel selection**. Also, as I indicated in the introduction, the **power distance** kernel also often performs better than the Gaussian RBF.

According to my review of SVM in health care and my experience, the Gaussian RBF kernel is the most widely used kernel. Is there rationale for using it, in general, even if that rationale is not specified? There is some rationale aside from their availability.

Is the most popular kernel, the Gaussian RBF kernel, sufficient for maximum accuracy in all studies/problems? Are kernels which are said to be universal approximators sufficient for maximum accuracy? What if I also want maximum or high interpretability of the kernel? Is it worth examining how kernels perform and/or relate to data given the **No Free Lunch theorem**?

The polynomial, Gaussian RBF and sigmoid kernels, as 3 out of 4 common kernels, are among the "first kernels investigated for the pattern recognition problem" [40]. Vapnik, the creator of SVM, states, but does not show, that the polynomial, Gaussian RBF and sigmoid kernels are all **universal approximators** [281]—i.e., they are able to **approximate any (continuous) function**, in the limit, given infinite data.

This is a reasonable assertion for the sigmoid kernel (also called neural network kernel), since universal approximation is proven for neural networks with one hidden layer and infinite sigmoid neurons in that layer, and since a support vector classifier is special type of neural network (where the output node is the classifier function). That said, the sigmoid kernel is not an admissible kernel [45, 167] (2.8.4 on page 44).

Given that other kernels achieve higher accuracy on various data sets, clearly **universal approximation [281] is not the sole determinant of a good classifier in the finite case**. In fact, little can be said about finite cases that do not approach the limit. A support vector classifier when viewed as a neural network has a neuron for each training data point and is therefore finite for finite training data.

So how does one choose a kernel? There is little guidance in the literature for general studies/problems and data—particularly in one place or as part of a **coherent framework**.

The **main rationale put forth** for using any kernel in general pertains to its ability to **separate (or shatter) points in the feature space**—which corresponds to some extent, to the ability to achieve a complex class boundary in the original data space. The VC dimension (introduced in section 2.2.1) is referenced in this case—however the VC dimension only applies to some configurations of data (which are ideal) not all configurations of data and it does not guarantee accuracy.

Burges [40] explains that if I use a classifier h_o based on oriented (linear) hyperplanes in the feature space to separate data (like a perceptron), it has a Vapnik-Chervonenkis dimension $VC \geq q$ when there is ***at least one*** configuration of q non-overlapping points

in the $q-1$ dimensional feature space (e.g., a triangle of three points in two dimensions, or a tetrahedron of four points in three dimensions, and so on, called simplexes) with any permutation of class labels, which can always be separated by a line or hyperplane.

If all subsets of q points in the data form a simplex in $q-1$ dimensions, then all of the points can be shattered. However if any subset of q points in the data form a straight line or linear hyperplane instead of a simplex, then that data cannot be shattered by a q-dimensional hyperplane. Such a hyperplane is a specific case or subspace within the space of possible data, whereas the simplex case covers the bulk of the space of possible data.

For h_{o} above, I can use a hard-margin support vector classifier or a soft-margin support vector classifier with a sufficiently large cost of error C instead [40], to achieve the same result. So it is argued that a support vector classifier h_k with a kernel k where $VC > n$ can perfectly classify (shatter) any simplex training data with sufficiently large VC and C[1], and it is on this basis, as well as anecdotal examples, that one may select SVM with a Gaussian RBF kernel, which has infinite VC dimension, or SVM with a polynomial kernel, which has a large VC dimension for a moderate number of dimensions in the input data.

According to Burges [40] however, **no theory "guarantees** that a given family of SVMs will have **high accuracy** on a given problem" — presumably in the context of C in general, data in general and test accuracy instead of training accuracy. The hyperparameter C controls the regularization in SVM, which causes SVM to choose training data points as support vectors with some degree of sparsity, and the complexity of the class boundary and margin is based on these support vectors of finite number and dimensionality. Since regularization limits complexity (for proper generalization to test data), the ability of SVM to discriminate is not just determined by the VC dimension of a kernel, it is also determined by how the kernel separates data.

The Mercer sigmoid kernel [45] with a finite dimensional feature space achieves better results than the Gaussian RBF kernel with an infinite dimensional feature space, for three clinical studies/problems and data sets. Therefore it is not sufficient to select a kernel based on its VC dimension alone.

Numerous authors state that the choice of SVM kernel significantly affects performance [18, 24, 36, 40, 128] — yet there is insufficient guidance about how to select a kernel for a given data set.

[1]Similarly, a sigmoid kernel can fully separate training data, if C is large enough[167], but with the additional condition that the optimization problem has at least one stationary point.

3.3 Need for new approach to kernel/model selection

Given the lack of rationale (Section 3.1) for kernel selection, a typical approach to SVM model/kernel selection (Figure 3.1) is to pick one at the outset, or perform cross-validated classification with several candidate kernels, and select one kernel which performs best in validation/testing.

This approach minimizes validation/test error (empirical risk) however it is an empirical approach which lacks theoretical justification and some of the common kernels used lack full interpretation/interpretability.

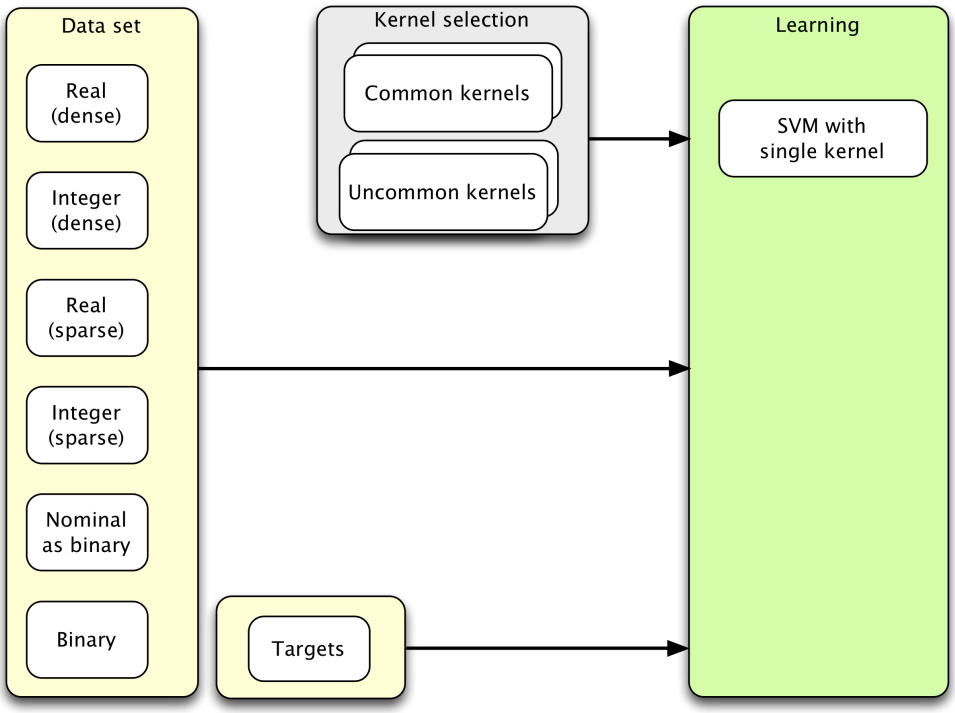

Figure 3.1: The status quo and de facto standard approach to model/kernel selection in SVM selects a single base kernel.

Alternative approaches to SVM model/kernel selection (Figure 3.2) in the literature do not provide further justification or interpretation. That is, multiple kernel learning, composite kernel learning and ensemble methods (Section 2.10) do not provide theoretical justification—and kernel alignment, kernel evolution and data-dependent kernels (Section 2.10), while providing justification, may still lack interpretability if they do not use transparent kernels.

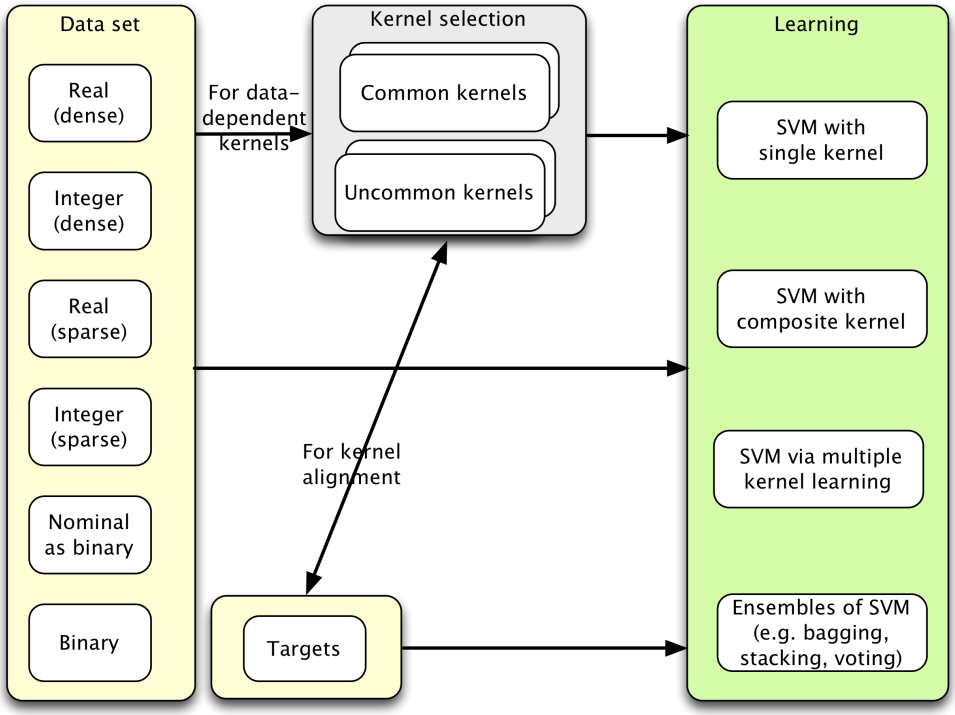

Figure 3.2: In contrast to the standard approach in the previous figure, alternative approaches to model/kernel selection in SVM use multiple kernels, eithjer in composite kernel learning, multiple kernel learning or ensembles. Data-dependent kernels and/or kernel alignment may also be used optionally.

Multiple kernel learning (Section 2.10) and **composite kernel learning** (Section 2.10) use multiple kernels in a linear combination to achieve better accuracy and to automate kernel selection, however there is no justification that the (base) kernels in the linear combination are sufficient. The original problem of an individual kernel not being justified persists, as well as the interpretability of individual kernels.

Kernel alignment (Section 2.10) quantitatively judges the match or alignment between a kernel and data. While it provides rationale for a kernel, it does not yield interpretability or transparency. Some **data-dependent** kernels (Section 2.10) may also lack interpretability if they are not based on a model or method of dependency which is interpretable.

3.4 Kernel requirements and gaps

I examine requirements for kernels from four sources:

1. similarity concepts in the literature (Section 2.6)

2. similarity and distance functions and kernels in the literature (2.6.1),

3. classification scenarios in literature, and

4. transparency literature.

3.4.1 Requirements from similarity concepts

The nine concepts of similarity in the background (Section 2.6) specify similarity requirements for kernels with three different priorities: must, should and may. I evaluate kernels against these requirements (Table 3.1 on page 59) and others that follow.

Table 3.1: Dot product kernels (D) do not have the ideal meanings for the maximum and minimum outputs, in comparison to stationary (S), equality (E) and Hamming (H) kernels.

Requirement	Priority	Numeric kernels							Non-numeric		
		RBF Exp	Lin	Poly	Sig	Pow Log	IMQ	tRBF	Del	Ham	HE
		S	D	D	D	S	S	S	E	H	H, D
Symmetric inputs	must	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Increase w commonality	should	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decrease w difference	should	✓	✓	✓	✓	✓	✓	✓	✓	×	×H
Max if same	should	✓	×	×	×	✓	✓	✓	✓	✓	×D
Min for no commonality	should	✓	×	×	×	✓	✓	✓	✓	✓	×D
Info. weighted sum of persp	should	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Finite min	may	✓			✓		✓	✓	✓	✓	
Finite max	may	✓			✓		✓	✓	✓	✓	
Triangle ineq	may										

In Lin's review of similarity requirements and functions [166], no function met all of the requirements. In my case, with kernels, no kernels fulfill the optional triangular inequality for similarity (Table 3.1)—but as Lin pointed out, that is not an intuitive requirement (and it is optional).

More importantly, I observe (Table 3.1) that dot product kernels do not have the ideal meanings for the maximum and minimum—whereby a maximum value should indicate

two inputs are the same and a minimum value should indicate that two inputs have no commonality. This is a concern if there is a need for dot product kernels—and there is such a need. Transparent kernels are almost always dot product kernels—e.g., the linear kernel, the homogeneous polynomial kernel, my proposed kernels, but not the truncated RBF kernel. Can I fulfill the requirements for a meaningful maximum and minimum with a dot product kernel? Are the maximum/minimum requirements compatible with other features that I require or desire in a dot product kernel?

I also observe that various dot-product and stationary kernels do not have a finite minimum and maximum—in other words, they are not bounded in their output. Unbounded kernels allow outlier values within a feature to have greater effect than may be desired or optimal. One suboptimal solution is to remove instances which have outlier values in any feature, prior to learning—however this results in a loss of information, a biased data set and the possible need to remove too many instances. Another problem with an unbounded kernel is that measures like the relevant dimensionality estimate [35] which may be useful (Section 6.4) are not well defined and reliable for unbounded kernels. Hence, I design and propose a number of bounded kernels as a better alternative (Chapter 4).

3.4.2 Requirements from similarity and distance functions and kernels

I also evaluate kernels against requirements from the literature [8, 16, 49, 77, 148] on similarity and distance functions and kernels. These requirements apply to some but not all studies/problems and data (Table 3.2). Some are also mutually exclusive. So I treat them all as requirements which one **may** optionally fulfill.

Table 3.2: Numeric kernels do not fulfill information entropy, asymmetry requirements or compactness requirements; non-numeric kernels do not support inner product or asymmetry requirements. Note: σ_ϵ denotes $\sigma \to 0, \epsilon$, while σ_∞ denotes $\sigma \to \infty$.

Requirement	Numeric kernels								Non-numeric		
	RBF	Exp	Lin	Poly	Sig	Pow Log	IMQ	tRBF	Del	Ham	HE
Equality (E)	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	✓		
Proximity, i.e., stationary (S)	✓	✓				✓	✓	✓		✓	✓
Covariance, i.e., dot product (D)			✓	✓	✓			~			✓
Info. entropy											
Other similarity										✓	
Probability	✓							✓	✓		
Asymmetric match weight										✓	
Heterogeneous data design											✓
Saturated	✓	✓			✓		✓		n/a	n/a	n/a
Compact									n/a	n/a	n/a
Local+ Global			✓	✓	✓	✓			n/a	n/a	n/a
Local only	σ_ϵ	σ_ϵ					✓	σ_ϵ	n/a	n/a	n/a
Global only	σ_∞	σ_∞						σ_∞	n/a	n/a	n/a

The first five requirements pertain to the different kinds of similarity which a kernel must implement—i.e., either **equality**, **proximity**, **covariance**, **information entropy** or some **other similarity**.

The simplest type of similarity function checks **equality**. This requirement comes from the delta kernel used in kernel supervised dimension reduction [16] however it may be used more broadly for similarity with discrete data types.

Two main types of similarity in kernels are **proximity** and **covariance**, corresponding to **stationary** and **dot product** kernels, respectively.

Proximity refers to how close two vector inputs are to each other, as the inverse of distance. For continuous data types, distance functions include the Euclidean, Manhattan and Minkowsky distance functions [148] and there are corresponding similarity functions for each one too. Proximity is found in the stationary class of kernels, such as the Power

kernel (2.9.1) which is a power of the Euclidean distance. It is also found in the Gaussian RBF kernel (2.8.3) which is a Gaussian function of the square of the Euclidean distance.

Proximity for binary and/or nominal data types, is computed with the **Hamming** kernel [8]. It computes the number of matches as similarity (or mismatches as distance). The kernel is similar to Hamming based simdist functions [49] for binary data types.

Covariance as a **dot product** or scalar projection also expresses similarity. For two continuous vector inputs covariance expresses how much they are going in the same direction (how much they co-vary) and it is weighted by their magnitudes, such that two similar vectors with large magnitudes have a greater inner product than two similar vectors with small magnitudes. The inner product is cosine similarity [77] without normalization, and is expressed by the the linear kernel $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^{\mathrm{T}}\mathbf{z}$.

Covariance and the **dot product** are also related to correlation [289]. Data are often centered in machine learning by the practioner prior to learning, i.e., $x = x' - E[x']$ and $z = z' - E[z']$. With centered data, I interpret the linear kernel as a sum of one-sample covariances $\mathrm{Cov}(x', z') = E[(x' - E[x'])(z' - E[z'])]$ in each dimension. Covariance (or correlation when standardized) is a key statistical measure that describes how two features either vary together in the same direction from the mean (positive correlation), vary in opposite directions from the mean (negative correlation), or don't vary together at all (zero or no correlation).

Covariance or **dot product** functions, for two binary vector inputs, can count the number of positive matches. The overlap [27, 49] and the heterogeneous Euclidean overlap [289] functions are examples which prompt this requirement.

Probability is sometimes included in the definition of a kernel when it represents a probability density function—as with the Gaussian RBF kernel, which represents a normal distribution. Probabilities may also be found in composite product kernels [118] or the Fisher kernel for graph models with hidden variables [61, 135] and in the similarity measure from Smirnov [27] and similarity coefficient from Burnaby [41].

Information entropy makes sense applied to multiple values of a single feature, but it is either not meaningful, or its meaning is not clear, if applied to multiple values not of the same feature, but z-scores across different features. Nevertheless I describe it as a possible, although low priority consideration. It is found in Lin's information theoretic similarity function [166], Li et al's similarity metric and its converse [162], Bennett et al's information distance function [19, 162]. It is also found in the maximum entropy kernel [269].

Other similarity concepts may also be employed by a kernel such as Hamman similarity [53] (Matching new and existing kernels to binary data).

Asymmetric match weighting is required for **nominals converted to binary indicators** and **presence-only binary data**. Consider for example two 10-level nominals converted to 10 binary indicators each—what matters in each is *which level is indicated positively*. The fact that other negatively indicated levels match is irrelevant, and if they are given weight in a kernel then the number of levels will have an impact as noise, drowning out the signal of the two levels that match or mismatch. Clearly the requirement for converted nominals is to **only count positives**—and I appear to be the first author to have identified the requirement for this case.

Presence-only binary data occurs in medicine. For example, in my research with Ehrsam's skin lesion data [45] there are 86 different **keywords** which a doctor may observe and attach to the image of a patient's skin lesion. A dermatologist attaches between 3 to 10 keywords, **positively indicating notable features**. Other keywords not attached/indicated are not necessarily absent—and not all of the keywords are mutually exclusive. Hence it is the **presence** of a keyword (a positive) that matters, **not the absence** (or negative).

Presence-only binary data also arise in ecology [203]: when a zoologist is observing an environment and notes the presence or absence of a species during the time of observation. The **presence** of a species has **certainty** while the **absence** has a good amount of **uncertainty**—the species may be present but unobserved. It is possible in this scenario to place a small amount of weight on absence (reflecting their uncertainty)—and the azzoo similarity function [49] does just that for binary data. Other examples are Eskin's distance measure [27] for nominal data and Gower's similarity coefficient [102] for binary, nominal or continuous data.

For two vectors with **heterogeneous** atomic data types (4.2), a simdist function or kernel can quantify similarity as a **weighted sum** [166] of similarities within each atomic data type. This requirement is found in the heterogeneous Euclidean overlap measure [289], the Hamming Euclidean hybrid kernel [8] for binary and real data and Szafranski's composite kernel for real data from different EKG channels [256]. Note that, I am **not** interested in the heterogeneity between reals, integers and ordinals where I assume a numeric kernel for reals performs sufficiently well. Instead, I **am** interested in the heterogeneity between reals, binary, nominals, and presence-only binary data, where there is a gap, or room for improved fit which may yield improved accuracy.

Saturation refers to a function that asymptotically and monotonically approaches a finite value—i.e., an upper or lower bound (or in math terms, a global or local infimum or

supremum). It is found in two specific applications of similarity and distance functions[221, 275], in the sigmoid transfer function for neural networks [201, 72], the sigmoid kernel and the Gaussian RBF kernel (Section 2.8). Saturation can help to ensure that a function is bounded. Also, it may, similar to compact support (discussed next), be used to keep the effect of a function local to a region, rather than global.

Compact support refers to a function that has a finite (or compact) region of input values which cause a non-zero output, while the remainder of that input region causes a zero output. This causes or ensures a local rather than global effect. This characteristic and/or requirement is found in wavelets [181] and Wendland's ψ-functions [80] as well as the circular and spherical kernels [94]. In medicine, there could be a **functional requirement** that is **study/problem-specific** to know with 100% surety that patient cases which are a certain distance away have 0% effect, e.g., for known physical constraints, or for patient similarity to apply only within a certain threshold in standard deviations. As a **logistical** or **computational** requirement, Genton [94] indicates that compact kernels could be advantageous, even crucial, for big data, since they can yield sparse matrices, i.e., only instances within the region of compact support will have non-zero entries in the matrix.

The **local** versus **global** effect of kernels is found in kernels for SVM and Gaussian processes [161, 288, 227]. Kernels or functions have **global** effect if one instance affects all of the predictions, whereas they have **local** effect if only instances nearby the instance being tested/predicted matter or have influence. Li *et al.* [161] explain that a local kernel provides good interpolation while a global kernel provides good extrapolation. While theoretically my classification studies/problems may only seem to involve interpolation, given the sparsity of data in multidimensional problems extrapolation or global behaviour is also useful.

The magnitude of **local** and **global** effects can be greater or lesser, or in the case of compact support, none. For example, a **Gaussian RBF kernel** with a small kernel width has a large local effect and a small effect globally, such that it is primarily considered as a **local** kernel. A **polynomial** kernel on the other hand has **global** effect because for two instances being compared which are distant from each other, it has a non-zero output.

3.4.3 Requirements from classification scenarios

Requirements for kernels also arise in five classification scenarios (Table 3.3 on page 65) for atomic data types (Table 3.3 on page 65), as follows.

Table 3.3: Numeric kernels in the literature do not support classification requirements for class balancing.

Require-ment	Numeric kernels							Non-numeric kernels		
	RBF Exp	Lin	Poly	Sig	Pow Log	IMQ	tRBF	Del	Ham	HamE
Admissible	✓	✓	✓	×	✓	✓	✓	✓	✓	✓
Mercer	✓	✓	✓	×	×	✓	✓	✓	✓	✓
Class balancing										
Translation invariance	✓				✓	✓	✓	n/a	n/a	n/a
Class denoising		✓D	✓D	✓D				n/a	n/a	n/a

Admissible kernels (2.7.4) are necessary for use with SVM in health care. Kernels which are not admissible, e.g., the sigmoid kernel, should not be used since theory and guarantees in SVM regarding optimization and stability do not apply unless a kernel is admissible [45, 167][2]—and the sigmoid kernel is sometimes admissible for a range of parameters that is difficult to determine [45, 167].

Mercer kernels can be used with any kernel method (which is advantageous for applications outside of SVM and select kernel methods that share the KKT constraints).

Translation invariance provides assurance that an algorithm will perform the same even if the data are biased[3]. This can be handy in case a population undergoes a change in bias over time (e.g., due to changes in policy; or due to the effect of inflation on the ability to pay for drugs or treatment in the United States).

Class balancing characteristics allow the SVM method and kernel to perform well with imbalanced data and positive match data (i.e., where matches of positive values aligned with the positive class, matter more than matches of negative values aligned with the negative class).

Class denoising characteristics allow the SVM method and kernel to perform well in the face of class overlap (Bayes error) and/or mislabelled instances (class noise).

Lastly I note that there are other requirements from classification scenarios (not shown in Table 3.3 on page 65) which pertain to complex data types which are beyond my scope

[2]Alternative theory [199] indicates that some guarantees of stability are possible

[3]This is of particular importance for image processing (which is outside of my scope of atomic data types), where a bias may refer to a shift in image position or intensity

with atomic data types (4.2). These additional requirements include, for example, **rotation invariance** and **scale invariance** for images or other spatial phenomena—where for example, dot product kernels are rotation invariant and isometric scale invariant while stationary kernels are not.

3.4.4 Requirements from model transparency

In this section I develop requirements for model transparency by extending statements and requirements from the literature [30, 168, 172, 187] as follows.

in three steps:

1. I identify (model agnostic) **criteria** from the literature, which are sometimes subjective, sometime objective.
2. I propose (model agnostic) **revised criteria** which are objective, and
3. I propose **SVM criteria** which are objective for use in a (**quantitative**) **measure** of model transparency.

I then identify which kernels meet the requirements (Table 3.4) and any gaps.

Table 3.4: Numeric kernels in the literature do not support transparency requirements

Requirement	Numeric kernels							
	RBF	Exp	Lin	Poly	Sig	Pow Log	IMQ	tRBF
(a) ∂_{fin}	×	×	✓	×	×	$\frac{1}{2}$	×	✓
(b) ∂_{essep}	×	×	×	×	×	×	×	×
(c) ∂_{eM}	×	×	✓	✓	×	×	×	×
(d) ∂_{+}	✓?	×	✓	✓	×	$\frac{1}{2}$	×	✓
(e) ∂_{glm}	×	×	✓	✓	✓	×	×	✓
(f) ∂_{lin}	×	×	✓	×	×	×	×	×
(g) $\partial_{\times}$	×	×	×	×	×	×	×	×
(h) ∂_{uni}	×	×	✓	×	×	×	×	✓
(i) ∂_{adm}	✓	✓	✓	✓	×	✓	✓	✓
score = sum/9	1/9	1/9	8/9	4/9	1/9	2/9	1/9	5/9

The first criterion applies to the **transparency of data** as opposed to methods and models.

Criterion from literature:

• Features are neither anonymous nor highly-engineered [168]. Note: if the input features are not transparent, then it does not matter if the methods and models are transparent.

Revised criterion:

• Features must be transparent—see my definition in 4.3.

Assuming the data criterion is met, **transparent models**, are those which are **interpretable** [172] or **decomposable** [168] in terms of the following four criteria:

Criteria from literature:

• Each calculation has an intuitive explanation [168]
• Additive models are interpretable [172]; or
  – The contributions of individual features are understandable [172], e.g., generalized linear models (GLM) are interpretable [172]

Revised criteria, respectively:

• The feature space has a finite number of dimensions; and
  – The feature space is explicit/known.
• A generalized additive model (GAM) with known/explicit functions; or
  – A generalized linear model [172]; or
    * A linear model (for even greater transparency); or
    * A multiplicative model with known/explicit functions; or
    * A model with uniform basis functions; or

SVM criteria, respectively:

• **(a)** The kernel has a finite number of feature space dimensions in the space $\mathcal{F}_k$, $\partial_{\text{fin}} : \dim(\mathcal{F}_k) < \infty$
  – **(b)** The kernel is explicit, symmetric and separable, $\partial_{\text{essep}} : k(\underline{x}, \underline{z}) = \phi(\underline{x})\,\phi(\underline{z}), \;\; \phi : \mathbb{R}^n \to \mathbb{R}, \; \phi$ known, or
  (c) see next item below.

• **(c)** The kernel is explicit Mercer, i.e., explicit, additive and separable in each feature $\partial_{\text{eM}} : k(\underline{x}, \underline{z}) = \left\langle \underline{\phi}(\underline{x}), \underline{\phi}(\underline{z}) \right\rangle = \underline{\phi}(\underline{x})^{\mathrm{T}} \underline{\phi}(\underline{z}) = \sum_q \phi_q(x_q)\,\phi_q(z_q), \; \phi : \mathbb{R} \to \mathbb{R}, \; \phi$ known
  (d) The kernel is additive $\partial_+ : k(\underline{x}, \underline{z}) = \sum_p f_p(\underline{x}, \underline{z})$, or

- **(e)** The kernel is a generalized linear model, i.e., a dot product kernel $\partial_{\text{glm}} : k(\underline{x}, \underline{z}) = g\left(\underline{x}^{\mathrm{T}}\underline{z}\right)$, where g^{-1} is a link function, or
  - **(f)** The kernel is linear $\partial_{\text{lin}} : k(\underline{x}, \underline{z}) = \underline{x}^{\mathrm{T}}\underline{z} = \sum_{i=1}^{n} x_i z_i$, or
  - **(g)** The kernel is multiplicative, explicit, symmetric and separable $\partial_{\times} : k(\underline{x}, \underline{z}) = \prod_{q=1}^{n} \phi_q(x_q)\,\phi_q(z_q)$, or
  - **(h)** The kernel is uniform $\partial_{\text{uni}} : k(\underline{x}, \underline{z}) = \sum_{q} \phi(x_q)\,\phi(z_q)$

Assuming the data criterion is met, **transparent methods** (or algorithms) [168] meet the following criterion:

Criterion from literature: Converges to a unique solution in training.

Revised criterion: Uses convex optimization. Note: for optimization to be convex in SVM, kernels must be admissible [242].

SVM criterion: **(i)** $\partial_{\text{adm}} : k$ is admissible, i.e., positive definite (p.d.) [187] or conditionally p.d. (c.p.d.) [30].

There does not seem to be any criterion in the literature that aptly describes the interpretability of a stationary kernel in the input space, despite the fact that it does have some level of intuitive interpretability based on distance, even if the features are confounded, albeit less.

3.5 Summary and thesis

From the requirements articulated in the preceding sections, seven gaps were identified of high or medium priority:

1. High: No admissible dot product kernels have a meaningful and finite minimum and maximum.
2. High: Only two kernels are transparent with a finite feature space: the linear and polynomial kernels and these two kernels are not as accurate as the Gaussian RBF kernel on atomic data types.
3. High: Only two kernels are interpretable as explicit Mercer kernels: the linear and polynomial kernels.

4. High: Only one kernel is designed for heterogeneous data and it is flawed (biased). The Hamming Euclidean hybrid kernel counts matching values (positive matches or negative matches) for every level of a nominal feature, such that a nominal with ten levels has twice the weight of a nominal with five levels and ten times the weight of a binary feature. It also uses a linear kernel used for real data types which is suboptimal.

5. Medium: The sigmoid kernel is not admissible (sometimes) and yet often performs better than the Gaussian RBF kernel. An admissible replacement is desired to fill this gap and gap #1, since the sigmoid kernel which only falls short as a solution to gap #1 in admissibility.

6. Medium: Only the linear kernel has uniform functions for interpretability.

7. Medium: Only the Hamming kernel has asymmetric match weighting for converted nominals and presence-only binary data. More options are desired and there are no kernels for continuously-imputed binary data with asymmetric match weighting.

Other findings were not deemed a priority, e.g., no compact kernels identified and no similarity with the triangular inequality.

To address the requirements above, the remaining body of my thesis provides the following three chapters:

- In Chapter 4, I propose a new approach to model/kernel selection, in which I select the family of models/kernels based on requirements and rationale from a new method I call **kernel data modeling**, which derives, designs or matches kernels to specific data types, distributions and requirements for interpretability and theoretical justification. To support the derivation and design of new kernels in this method, I propose a new kernel class which provides greater transparency and interpretability than common kernels (Section 2.8) and uncommon kernels (Section 2.9).

- In Chapter 5, I describe the experimental data, method and results which confirm that the kernels I propose are at least as accurate as the aforementioned common and uncommon kernels. I also show how transparency affects the innate and non-innate views of results in support vector machine classification, and I contribute improvements to such views in the process.

- In Chapter 6, I develop new quantitative measures (and concepts) of model transparency and inherent model interpretability and show how these can be used for

prior, initial and posterior model selection (Chapter 6). I demonstrate that high accuracy can be achieved with high inherent model interpretability.

These three chapters are followed by conclusions, future work and other back matter.

Chapter 4

Kernel data modeling

In this chapter I define a new kernel class which is underpins all of the new kernels which I propose within a new framework or approach to model/kernel selection (Figure 4.1). I also define ways to describe features, data types and kernels which are helpful for the purpose of transparency and other goals in the framework. I derive kernels for some data types, design kernels for some data types and match kernels to data types following my framework, which is described next.

4.1 Kernel data modeling framework

I propose a new approach to model/kernel selection (Figure 4.1), wherein I select the family of prior models based on kernel data modeling as well as *a priori* measures of inherent model interpretability, prior to optimization and cross-validation (or bootstrapping). Then after optimization, I can select the best kernel according to test/validation accuracy as well as *posterior* measures of model interpretability.

Kernel data modeling refers to deriving kernels, matching kernels and designing kernels for specific data types and distributions. I also satisfy other requirements not specific to a data modeling process.

Kernel data modeling is an approach that provides theoretical justification for a kernel and understanding about how the kernel works as a similarity measure—which in turn provides assurance about how the kernel behaves and generalizes to new data, independent of empirical evidence. The sections that follow explain the approach, beginning with new descriptions and definitions needed as a foundation.

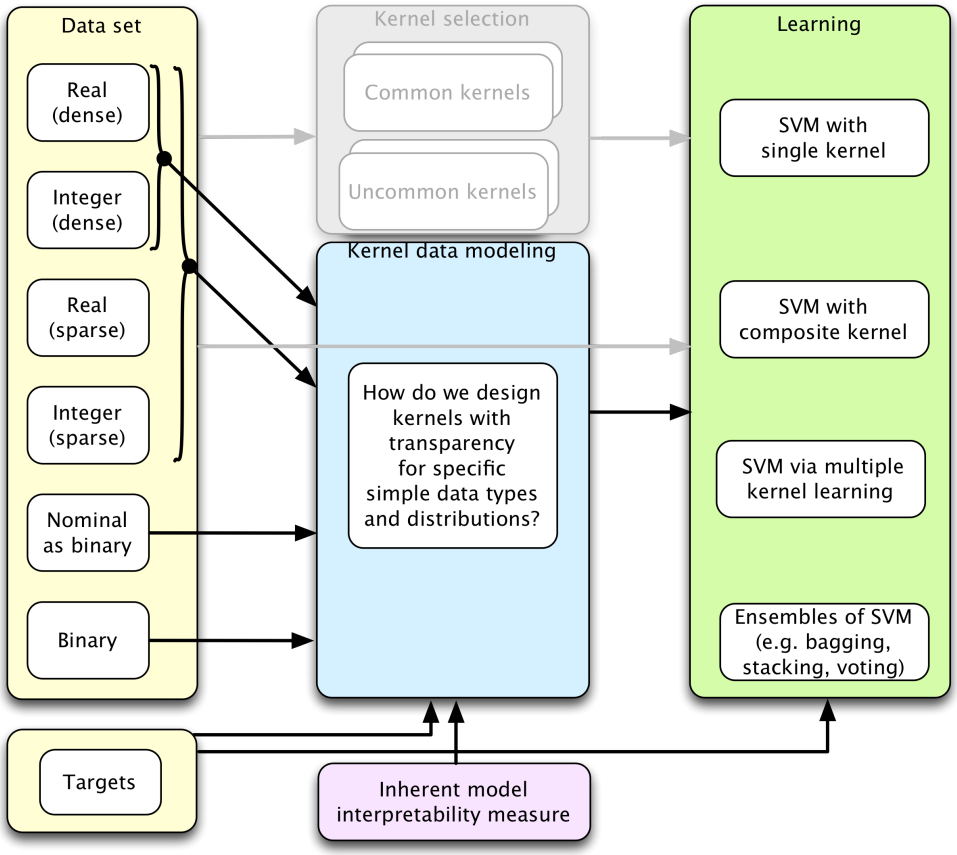

Figure 4.1: I propose a new approach (blue and pink) to model/kernel selection in SVM—however it poses the research question, how do I design kernels for this approach? Subsequent sections answer that question.

4.2 New data descriptions: atomic data types and transparent features

I propose two new descriptions or descriptors: atomic data types and transparent atomic features. In the context of the first I also briefly discuss related descriptors from the literature which are relevant to my thesis.

Definition 4.2.1. Atomic data types (Table 4.1) are: binary, nominals (or categoricals), ordinals, dates (and datetimes), integers (including counts) and reals. Binary values may be considered nominals, however there are methods and implications specific to binary data separate from nominals in general. Additional types, such as counts (natural numbers) may be considered integers with a bound.

Table 4.1: Examples of atomic data type

Atomic data types and subtypes	Illustrative examples
Real ($\mathbb{R}$)	sodium level, pulse, weight, voxel intensity, temperature
Integer ($\mathbb{Z}$)	no. of children compared to median, no. of children
Date, datetime	date, date and time
Ordinal	cancer stage; survey agreement scales; Charlson comorbidity index
Nominal	blood type, first language, dialysis modality, amino acid
Binary	male/female, presence/absence

Compared to other sets of data types reviewed in the background chapter (Section 2.5), atomic data types more closely matches the terms used in machine learning, the types found in practice and my purpose to derive, design and match kernels to data types via atomic kernels (Section 4.3).

Complex data types or documents, such as strings of text, images or audio, are not within the scope of this thesis—however one may extract features from such documents, e.g., Zernicke moments from images, as atomic data types.

Other characteristics of data and targets, if applicable, are conveyed in meta data (section A.6). Examples of meta data for each feature, include the following binary indicators: outliers-trimmed[1], top-coded, bottom-coded, non-negative (e.g., $\mathbb{R}^+$ or counts $\mathbb{N}$), normalized, centered (including binary), presence-only, positive-match, complete (vs. missing), and imputed. Examples of meta data for a data set include: complete-case-analysis, imputed, complete (vs. missing), normalized, and centered.

[1] prior to normalization

Definition 4.2.2. Transparent features $\mathcal{F}_{\mathrm{T}}$ are meaningful for the clinical or business purpose.

They are the original features q_i and/or simple transformations thereof, without introducing much collinearity. The original features are scaled (by b), shifted (by d) and/or reflected (by r) as v_i (4.1) and then simply transformed by a log, hyperbolic tangent, multiplicative inverse, square, top-coding or bottom coding (with a minimum or maximum function respectively) as w_j (4.2). First order interactions of w_i are also considered transparent (4.3). Some of the transformations in the set of transparent features are similar to fractional polynomials [219].

$$v_i = (-1)^r \frac{(q_i - d)}{b}, \qquad b, d \in \mathbb{R}; r \in \{0, 1\} \tag{4.1}$$

$$w_j \in \left\{ v_i,\ \frac{1}{v_i},\ \log(v_i),\ v_i^2,\ \tanh(v_i),\ \mathrm{abs}(v_i) \right. \\ \left. \min(c_{\mathrm{top}}, v_i),\ \max(c_{\mathrm{bottom}}, v_i) \right\} \quad \forall i \tag{4.2}$$

$$\mathcal{F}_{\mathrm{T}} \subseteq \{w_j,\ w_j w_{k \neq j}\} \qquad \forall j, k \tag{4.3}$$

To avoid collinearity only choose q_i or its reflection $-q_i$ but not both, or a top-coded feature or bottom-coded feature, but not both.

That is, features from $\mathcal{X}_{\mathrm{T}}$ do not include complex functions, e.g., PCA, whitening, etc (Section A.3), random projections or unidentifiable features.

4.3 New kernel descriptions: uniform, atomic, transparent and implicit

We describe kernels as uniform, atomic, transparent and/or implicit and define those descriptors or descriptions as follows.

Definition 4.3.1. A **uniform kernel** has the same functions and coefficients in each of its parts. An explicit Mercer kernel (defined in subsequent 4.4) is uniform if it has the same basis function ϕ in each feature dimension, e.g.,

$$k_{\mathrm{uni}}(\underline{x}, \underline{z}) = \sum_j \phi(x_j)\, \phi(z_j) \tag{4.4}$$

An additive kernel is uniform if it has the same kernel function k in each part. The linear kernel $k(\underline{x}, \underline{z}) = \underline{x}^{\mathrm{T}} \underline{z}$ and homogeneous polynomial kernel $k(\underline{x}, \underline{z}) = \left(\underline{x}^{\mathrm{T}} \underline{z}\right)^d$ are examples

of uniform kernels. A uniform kernel is easier to interpret than a non-uniform kernel, with all else being equal.

Definition 4.3.2. An **atomic kernel** is (described as) a kernel derived for, matched to, or designed for one (or two) specific atomic data type(s) without being specific to particular feature distributions.

The Bayesian density (BD) kernel which I propose in (Section 4.5.4) is an atomic kernel since it is specific to real data types with any feature distribution. Other atomic kernels I propose in subsequent sections include: the Mercer sigmoid kernel (MSig), the orthant sigmoid (OSig) kernel, the orthant linear (OLin) kernel, the insensitive sigmoid variant kernel with Gaussian constraint (ISVgc), the insensitive sigmoid variant kernel with hyperbolic tangent constraint (ISVhc), the orthant insensitive sigmoid variant kernel with gaussian constraint (OISVgc), the orthant insensitive sigmoid variant kernel with hyperbolic tangent constraint (OISVhc) and the Gaussian derivative (GD) kernel.

Atomic kernels support the kernel data modeling approach which I propose.

Definition 4.3.3. A **transparent kernel** scores at least 50% on the U_∂ transparency measure (as in Table 6.6 on page 154) where the measure is defined in Section 6.3. In the context of data, the features must also be transparent (see the next definition). Explicit Mercer kernels are transparent and uniform explicit Mercer kernels are even more transparent.

Definition 4.3.4. An **implicit kernel** is a kernel for which the basis functions ϕ or ϕ_j are implicit, i.e., not known, or they are known but implicitly (rather than explicitly) computed. The sigmoid, multiquadric, power distance and log kernels are examples of the former, while the Gaussian RBF kernel and polynomial kernels are examples of the latter.

4.4 A new kernel class: explicit Mercer kernels

Unlike Mercer kernels [187] I define a specific kernel class called explicit Mercer kernels. Mercer kernels include **all** separable kernels [94, 279] which are uniform, **some** dot product kernels [243], **some** stationary kernels [94] and **all** positive definite kernels [187]—whereas explicit Mercer kernels include **some** separable kernels which are uniform, **no** dot product kernels, **no** stationary kernels and **some** positive definite kernels. I explain this in detail below.

From an overview, Mercer kernels and explicit Mercer kernels are clearly different, yet the latter is a natural way to define a Mercer kernel based on the third definition of a Mercer

kernel repeated here (from Section 2.7.1 equation 2.23),

$$k\left(\underline{x},\,\underline{z}\right)=\left\langle \underline{\phi}\left(\underline{x}\right),\,\underline{\phi}\left(\underline{z}\right)\right\rangle \qquad \underline{x},\underline{z}\in X,\,\underline{\phi}\in\mathcal{F},\,k\in\mathbb{R} \tag{4.5}$$

The name, explicit Mercer, may seem to imply the subset of explicit kernels which are Mercer, but that is not the reason for the name—explicit Mercer kernels are thus named because they are the most natural way to define an explicit kernel which is Mercer and because they have specific characteristics as a class of kernels unique from any presently defined kernel class in the literature (e.g., [24, 94, 178, 233, 240]). The next two subsections (4.4.1 and 4.4.2) explain why this class of kernels has been overlooked.

I define explicit Mercer kernels as a new kernel class in four steps. First, the mathematical literature [279] defines the class of ***separable*** kernels by a ***mathematical*** kernel of the form $k_s\left(\underline{x},\underline{z}\right)=f\left(\underline{x}\right)g\left(\underline{z}\right)$ where f and g have scalar outputs, and the kernel has a Gram matrix of rank one. These are not symmetric however and for kernel methods I must have symmetric and admissible kernels. Also, a rank one matrix lacks the complexity or degrees of freedom required to perform well in classification with data that has more than one relevant dimension (e.g., more than one dimension in compression) required for optimal accuracy in classification. Instead I define the following.

1. A **separable (sep)** kernel is defined in mathematics [279] by the following expression where f and g exist but are not necessarily known,

$$k\left(\underline{x},\,\underline{z}\right)=f\left(\underline{x}\right)g\left(\underline{z}\right),\;\;f,g:\mathbb{R}^{n}\rightarrow\mathbb{R}$$

   This is a kernel class [94] as opposed to a specific kernel—and Genton alternatively refers to this class as: separable nonstationary kernels [94]. This class is not symmetric however, as required for kernel methods, so we therefore consider (and define) the next class.

2. I define a **symmetric separable (ssep)** kernel, with a common (or symmetric) basis function ϕ that exists but is not necessarily known, as follows

$$k_{\text{ssep}}\left(\underline{x},\,\underline{z}\right)=\phi\left(\underline{x}\right)\phi\left(\underline{z}\right),\;\;\phi:\mathbb{R}^{n}\rightarrow\mathbb{R}$$

   Notably, this kernel class, a subset of the previous class, when applied to any data, has a Gram matrix of rank one. If the Gram matrix has a lesser rank (or complexity or degrees of freedom) than the relevant dimensions in the data, then it probably will not be accurate in classification. Genton indicates that the computational reduction

of this class may be of crucial importance with very large training sets (i.e., big data) however, without accuracy, that is of little use.

3. I define an **explicit symmetric separable (essep)** kernel, as a subset of the previous class, where the basis function ϕ is explicit and known, for transparency regarding how the kernel works, as in

$$k_{\text{essep}}\left(\underline{x},\, \underline{z}\right) = \phi\left(\underline{x}\right)\phi\left(\underline{z}\right), \quad \phi : \mathbb{R}^n \to \mathbb{R}, \quad \phi \text{ known}$$

4. Finally, I define an **explicit Mercer (eM)** kernel by applying the above kernel class to each feature, one at a time, and then taking the sum of those kernels, as follows, where the basis functions ϕ_q are explicit and known, i.e.,

$$k_{\text{eM}}\left(\underline{x},\, \underline{z}\right) = \underline{\phi}\left(\underline{x}\right)^{\text{T}} \underline{\phi}\left(\underline{z}\right) = \sum_q \phi_q\left(x_q\right)\phi_q\left(z_q\right), \quad \phi : \mathbb{R} \to \mathbb{R}, \quad \phi \text{ known}$$

The complexity of an SVM model is bounded by the number of support vectors [240, 117]. In other words, for a data set with N instances, SVM has no more than N_{train} support vectors, i.e., $sv \leq N_{\text{train}}$ regardless of the kernel k. SVM is an instance-based method, whose kernel matrix has a rank $R\left(k\right)$ no greater than the support vectors: $R\left(k\right) \leq sv \leq N_{\text{train}}$.

Now, assume for the remaining discussion in this section that we choose the input features (or input space) to be the n original features in the data set—then for an explicit Mercer kernel the rank of the kernel matrix $R\left(k_{\text{eM}}\right)$ is the lesser of n or N_{train}, where typically $n \leq N_{\text{train}}$ except for multi-omic data (e.g., genomic, proteomic). The kernel class is **additive, explicit, symmetric and "separable in each feature"**. For brevity I refer to them as **explicit Mercer** kernels since, as I previously mentioned, they are the most natural and intuitive way to define kernels explicitly which are guaranteed to be Mercer.

Notably, kernels which are explicit and Mercer, are not necessarily **explicit Mercer**—they must be additive and "separable in each feature". Additive kernels [178] are not necessarily "separable in each feature" nor are they necessarily explicit—the explicit and **exact** expression for a Gaussian RBF kernel is additive [55].

An explicit Mercer kernel is additive by itself, but when it is used inside an SVM it becomes a generalized additive model (GAM), $k\left(\underline{x},\, \underline{z}\right) = g\left(\sum_q \phi_q\left(x_q\right)\phi_q\left(z_q\right)\right)$, with g^{-1} as a link function, and where g is the SVM prediction formula:

Proof of Mercer compliance: This kernel class is Mercer because it is explicitly defined as an inner product of a feature map at $\underline{x}$ and $\underline{z}$ respectively per (2.23) in

Section 2.7.1.

It is not a stationary kernel class and is therefore **not** translation-invariant—hence such kernels may benefit from a shift hyperparameter.

Whereas a symmetric separable kernel (ssep or essep) has reduced storage complexity [94] and computational complexity [206], as a tensor (outer) product of vector $\mathbf{a} = \begin{bmatrix} f(\underline{x}_1) & \cdots & f(\underline{x}_n) \end{bmatrix}^{\mathrm{T}}$ with itself, with storage complexity $\mathbf{O}(N)$.

An explicit Mercer kernel also has reduced storage complexity, as a tensor (outer) product of the rectangular matrix $\Phi^{N\times n} = \begin{bmatrix} \underline{\phi}(\underline{x}_1) & \cdots & \underline{\phi}(\underline{x}_q) \end{bmatrix}^{\mathrm{T}}$ and its transpose $\left(\Phi^{\mathrm{T}}\right)^{n\times N}$ with storage complexity $\mathbf{O}(nN)$, when $n \leq sv$, compared to common kernels such as the Gaussian RBF kernel, sigmoid kernel and power distance kernel which have storage complexity$\mathbf{O}(N^2)$.

This class of kernel has an explicit basis function or explicit feature map [282, 147], which provides the opportunity to define and analyze kernels with precision toward my goal of understanding kernels that I apply in SVM.

4.4.1 More on the complexity of explicit Mercer kernels

It can be argued that any kernel, such as an explicit Mercer kernel, with an explicit basis function and a finite number of terms (or parts) is equivalent to a linear kernel with non-linear pre-processing in each feature. Clearly an explicit Mercer kernel is more complex than a linear kernel because it has non-linearity instead of linearity, however, from the perspective of the rank of the kernel matrix, it has the same rank or complexity. Hence, complexity depends on the perspective or measure.

Another important perspective on model complexity comes from Lou *et al.* [172] who describe models as belonging to one of five categories with increasing complexity: linear models, generalized linear models (GLM), additive models, generalized additive models (GAM) and full complexity models. An explicit Mercer kernel is, on its own, an additive model, and within SVM it becomes a generalized additive model—the second most complex model that Lou *et al.* argues has more interpretability than the most complex models. Hastie and Tibshirani [115, 116, 84] write extensively about GAMs and techniques for using them—with particular applicability to medical research.

4.4.2 Explicit versus implicit kernels and the kernel trick

The complexity of kernels discussed in the previous section, also arises in the context of the feature space and the kernel trick [24, 240, 233] (both introduced in Section 2.2). Are kernels only intended to be implicit? Does the class of explicit Mercer kernels make sense in the context of the kernel trick? I answer these questions while revisiting the concept of the kernel trick below.

The kernel trick [24, 240, 233] is responsible for the emergence of kernel methods, however, recent literature [156] and lectures or seminars in less formal settings, focus on the lesser of two parts or purposes of the kernel trick, such that the term has diverged from its original meaning.

The first, most important and remarkable part of the kernel trick is that a linear algorithm becomes non-linear, when the dot product in the input space or original data is replaced with a different dot product [240], which is now non-linear in the input space but linear in a different new feature space—i.e., kernelization. This **essential** part of the kernel trick is responsible for the birth of kernel methods used in SVM and other kernel methods such as kernel principal components analysis, kernel logistic regression, kernel k nearest neighbor and kernel ridge regression[2].

The second remarkable part of the kernel trick is that, when replacing the linear kernel with another, I **may** (i.e., optionally) use a kernel which has an **implicit** basis function (or feature map), with an infinite or high number of features/dimensions (similar to a Taylor series expansion) in the feature space where the class boundary is drawn, but with **calculations** done in the input space, which saves computational cost. Some sources [156] refer to this elegant computational aspect as the "kernel trick" while other sources [24, 240, 233] refer to kernelizing an algorithm as the kernel trick .

A recent mistake with the kernel trick is to take the first and most important part, the emergence of kernel methods, for granted and describe the kernel trick as the latter—implicit kernels with an infinite or high dimensional feature space— thinking that the latter is part of SVM's essential nature and purpose, accounting for its accuracy. The following facts and evidence argue against this perspective:

1. SVM practitioners commonly use the linear kernel, an explicit kernel—particularly for text data, in which its use is dominant because it performs better than the Gaussian RBF kernel.

[2]Other kernel methods include, for example: kernel fisher discriminant, kernel ***supervised*** principal components analysis and many variations of SVM including relevance vector machines, import vector machines, calibrated SVM, sparse grid SVM, proximal SVM and minimal SVM.

2. The results with my proposed kernels show otherwise on data sets with atomic data (corroborated by others in Section A.9 and corroborated by my results with a third party's code base).

3. SVM has finite limitations (discussed in Section 4.4) which dispel the notion that the Gaussian RBF kernel's infinite dimensionality is critical—and this is corroborated by results with the truncated Gaussian RBF kernel [209].

4. If not infinite dimensionality then, high dimensionality—e.g., as achieved with a polynomial kernel—is also cited as an objective of kernels. Consider text classification, where the linear kernel is said to perform well because of high dimensionality (many features). However, if that is the case, then the linear kernel should also perform better/well in other high dimensional problems, such as in genomic data with thousands of features, and conversely it should not perform better/well in other (low dimensional) problems—but that is not observed in results (Section 5.3). With benchmark genomic data (Colon and Prostate cancer) it does not perform better or well and with other data sets its accuracy is not any worse (its accuracy may in fact be better, relatively speaking).

Returning to the two questions I posed at the beginning of this section: no, kernels are not only intended to be implicit—they are intended to add the flexibility of non-linearity into a linear algorithm; yes, the class of explicit Mercer kernels, which is not necessarily high or infinite dimensional, makes sense in the context of the kernel trick. There is much to be learned about the utility of each dimension in SVM—and work on transparency and interpretability will assist that endeavour.

In conclusion, the literature confounds several different purposes of the kernel trick without being clear on the necessity and priority of each. I have discussed two of these purposes above—non-linearity, and infinite or high feature space dimensionality—while reserving the next section to discuss a third purpose: computational efficiency. One might consider computational efficiency to be a subordinate requirement in the service of infinite or high dimensionality, but it is presented as a primary requirement by some authors [233], hence I discuss it separately in the next section, although it overlaps with the topic of dimensionality.

4.4.3 Explicit Mercer kernels and computational efficiency

Shawe-Taylor *et al.* [233] ***assume*** that applications require computationally efficient kernels which are implicit ***not*** explicit, without regard for other requirements which are

more important in medicine: accuracy, transparency and interpretability (which are singly and/or jointly improved by explicit Mercer kernels in Chapter 4 on page 71 and Chapter 6 on page 139). Explicit Mercer kernels can have computational challenges in one uncommon, possibly rare, case in health care: when there are over 2500 features ***without*** feature selection or dimension reduction **for** parsimony **and** potentially improved accuracy (explained next).

Health care requires parsimony which is measured by rules of thumb such as events per variable (EPV) and samples per feature (SPF)—and as a result, feature selection or dimension reduction are often necessary, especially in genomics. Health care also requires accuracy which is improved by feature selection in the two genomic data sets that I use. SVM classification of colon cancer data with 2000 genes has a benchmark accuracy of 90.32% [89] without feature selection and an accuracy of 100% with 2 genes [3], 6 genes [86] or 16 genes [110]. SVM classification of prostate cancer data with 1000 out of 12,600 genes has a benchmark accuracy of 84.8% (my result) which improves to 92.2% [263] or 93.09% [153] with 50 genes, 94.12% with 10 genes [202], 98.65% with 3 genes [3] and 98.66% with 2 genes [3]. The trend is not always monotonically decreasing, as evidenced by some results within a single paper [202]—nevertheless the general point stands as the potential for improved accuracy, but not as a rule or guarantee.

Explicit Mercer kernels can be designed to explicitly and traceably meet functional similarity requirements, which is the standard approach to design in general [217] and in the software industry [133], whereas implicit kernels may be more efficient and elegant even though efficiency is not a primary concern or requirement in health care.

Clear examples of kernels with **implicit** basis functions are the sigmoid kernel, power distance kernel, logarithmic kernel and inverse multiquadric kernel. The Gaussian RBF kernel has an exact **explicit** basis function [55, 209] but it is implicitly computed. The polynomial kernel also has an **explicit** basis function which is implicitly computed.

4.4.4 Feature shaping differs from my use of explicit Mercer kernels

It can be argued that any kernel, such as an explicit Mercer kernel, with an explicit basis function and a finite number of terms (or parts) is equivalent to a linear kernel with non-linear pre-processing in each feature. The latter approach is called feature shaping and is an alternative approach that can produce the same (kernel) output. The following question therefore arises: how does my use of explicit Mercer kernels differ from feature shaping?

My framework for designing kernels based on the class of explicit Mercer kernels differs from feature shaping in several important ways, so the two should not be confused. It often arises in mathematics that we can perform a computation in multiple ways (e.g., addition and multiplication are commutative, or unordered). In some cases, equivalent computations have substantively different implications for processing and interpretation, such as the equivalence between the primal and dual optimization formulas for SVM. In the case of my use of explicit Mercer kernels versus feature shaping, there are differences in the process, design, intent, interpretability or understanding, usability and risk of error.

Feature shaping refers to applying a basis function [57] or feature map [240] as a pre-processing step and then using a linear kernel for classification [78]. The pre-processing approach in feature shaping is problematic for explicit kernels which are parameterized (or explicit basis functions which are parameterized), because the user has to pre-process the data differently for each set of parameters and for each kernel. That is a lot of extra work and complexity in the storage, management and comparison of data that is error-prone and limits interpretability since the input data from one kernel to the next is different, making it difficult to validate, compare and notice errors.

In contrast, using a kernel directly as the standard approach, and as it is designed and intended, allows the original data to be used with multiple kernels and multiple sets of parameters which is standard machine learning practice.

Pre-processing also misses the point that thinking about kernels is more than just thinking about its parts, i.e., the basis functions as shaping features. For example, my Bayesian binormal kernel is designed based on three covariance requirements for the kernel as a whole, not just the basis functions and this leads to a different Bayesian design from other kernels in the literature. As another example, my orthant sigmoid kernel which provides asymmetric match weighting, requires thinking about the kernel output—not the basis functions independently.

Hence my proposed kernels are different from feature shaping [78] in the way that my kernels are derived, designed, matched and used.

4.5 Deriving new kernels for reals and integers

I derive two kernels for reals and integer data types with binormal distributions:

1. A **Bayesian binormal** (BBN) kernel

2. A **joint probability binormal** (JPBN) kernel

Then I posit two kernels for reals and integers with any distribution based on kernel density estimation (KDE) [163, 211]:

- A **Bayesian density** (BD) kernel
- A **joint probability density** (JPD) kernel

4.5.1 Bayesian binormal kernel (proposed)

Green and Swets [105] introduce the concept that an observation or test result X can be mapped to a line or axis whereby a decision maker applies a cutpoint c on that axis to classify the subject of the observation as either having or not having a condition for $X > c$ and $X < c$ respectively; and where it is assumed that the observation X is drawn from two separate (but usually overlapping) Gaussian distributions for subjects with and without the condition [306] (Figure 4.2). This is known as the binormality assumption [10, 306] and it is used in the literature to produce a parametric fit to Receiver-Operator Characteristic (ROC) data in diagnostic medicine and other fields [10, 112, 306].

The binormal model (Figure 4.2) is popular for parametric model fitting in ROC analysis [309] and several programs used today are based on it [306]. It is typically discussed in the context of ratings data [111], i.e., ordinals, since that is when a parametric fit to an ROC curve is most helpful.

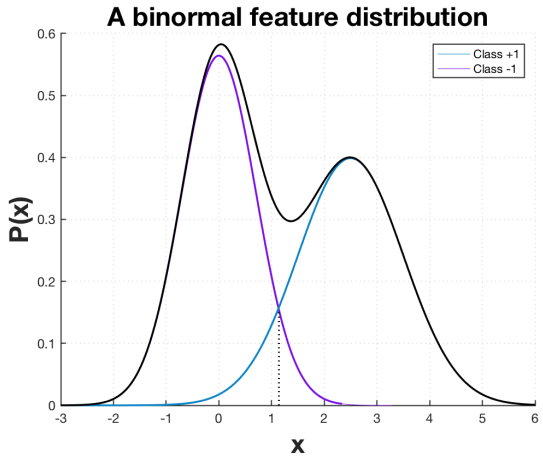

Figure 4.2: A binormal distribution with a vertical line as a cutpoint or decision threshold.

Hanley [111] reviews six justifications in the literature for the assumption. One justification is that random variables which describe natural phenomena are considered the sum of a large and constant number of other random variables such that the Central Limit Theorem justifies a normal distribution [105] for each class or health condition. Hanley expresses concern that the justifications are "either pragmatic or else too technical to evaluate" but observes that its continued use is likely. Austin and Steyerberg [10] state that an overall normal distribution is a potentially more plausible for an explanatory variable but even then the underlying distributions are still approximately normal in many instances—i.e., the explanatory variable is approximately binormal.

Hence there are some authors who question the binormal assumption or disagree with it as Zweig and Campbell [309] noted. In any case, I can test for normality and apply a kernel to features for which it is appropriate.

So if I assume that a continuous feature x_i has one normal distribution for patients with a condition (or outcome) $y_+ = +1$,

$$P\left(x_i \mid y_+\right) \sim \mathcal{N}\left(\mu_{x_i+}, \sigma_{x_i+}\right) \tag{4.6}$$

and a second normal distribution for patients without the condition (or outcome), labelled by $y_- = -1$,

$$P\left(x_i \mid y_-\right) \sim \mathcal{N}\left(\mu_{x_i-}, \sigma_{x_i-}\right) \tag{4.7}$$

then I can design a kernel to match this assumption. I begin with an explicit Mercer kernel where in each dimension i, I have k_i as a product of basis functions $f_i\left(\cdot\right)$:

$$k_{\text{ds}}\left(\mathbf{x}, \mathbf{z}\right) = \sum_{i=1}^{d} k_i = \sum_{i=1}^{d} f_i\left(x_i\right) f_i\left(z_i\right) \qquad x_i, z_i \in X,\, \mathbf{x}, \mathbf{z} \in X^d,\, k_i, f_i \in \mathbb{R} \tag{4.8}$$

This form aligns with the similarity requirement for covariance as a dot product (3.4.2), alternatively viewed as a sum of one-sample covariances, k_i in each dimension. For covariance in each dimension, three prima facie requirements arise:

Requirement 4.5.1. If x_i and z_i are probably from the same class, the covariance k_i (4.8) should be positive.

Requirement 4.5.2. If x_i and z_i are probably from different classes then the covariance k_i (4.8) should be negative.

Requirement 4.5.3. As the relative probability of x_i belonging to one class increases (or decreases) over its probability of belonging to the other class, so too should the magnitude of the covariance k_i (4.8) increase (or decrease). The same applies to z_i.

If I denote the most probable class for x_i and z_i, in dimension i as $\hat{y}_{x_i}$ and $\hat{y}_{z_i}$ respectively, then we may define the latter with positive and negative class labels $\{-1, +1\}$ as follows,

$$\hat{y}_{x_i} = \underset{y}{\operatorname{argmax}}\, P\left(y \mid x_i\right) \quad \hat{y}_{x_i} \in \{-1, +1\}$$
$$\hat{y}_{z_i} = \underset{y}{\operatorname{argmax}}\, P\left(y \mid z_i\right) \quad \hat{y}_{z_i} \in \{-1, +1\}$$

where the product,

$$k_i\left(x_i,\, z_i\right) = \hat{y}_{x_i} \cdot \hat{y}_{z_i} \tag{4.9}$$

from inspection, is positive for the same class and negative for different classes, thereby fulfilling the first two requirements (4.5.1 and 4.5.2). However, the product does not meet the third requirement because it only considers the probability of the maximum or most likely class for x_i, instead of the probability ***relative*** to the less likely class. The same gap exists with z_i.

However I can express the third requirement (4.5.3) as: the covariance k_i should be proportional to the difference of probabilities regarding x_i,

$$k_i \propto P\left(y_+ \mid x_i\right) - P\left(y_- \mid x_i\right) \tag{4.10}$$

and the covariance k_i should be proportional to the difference of probabilities regarding z_i

$$k_i \propto P\left(y_+ \mid z_i\right) - P\left(y_- \mid z_i\right) \tag{4.11}$$

Let

$$f_i\left(\cdot\right) = P\left(y_+ \mid \cdot\right) - P\left(y_- \mid \cdot\right) \tag{4.12}$$

then if I specify the covariance k_i as the product of f_i at x_i and z_i,

$$k_i\left(x_i,\, z_i\right) = f_i\left(x_i\right) f_i\left(z_i\right) \tag{4.13}$$
$$= \left[P\left(y_+ \mid x_i\right) - P\left(y_- \mid x_i\right)\right] \cdot \left[P\left(y_+ \mid z_i\right) - P\left(y_- \mid z_i\right)\right] \tag{4.14}$$

it clearly meets the third covariance requirement (4.5.3). It also meets the first and second covariance requirements: if $f_i\left(x_i\right)$ and $f_i\left(z_i\right)$ are both positive or both negative that yields a positive covariance indicating the same class, whereas if $f_i\left(x_i\right)$ and $f_i\left(z_i\right)$ are different signs that yields a negative covariance indicating different classes.

Using Bayes theorem I can re-write (4.12) as

$$f_i(x_i) = \frac{P(x_i \mid y_+) P(y_+)}{P(x_i)} - \frac{P(x_i \mid y_-) P(y_-)}{P(x_i)} \tag{4.15}$$

$$= \frac{P(x_i \mid y_+) P(y_+) - P(x_i \mid y_-) P(y_-)}{P(x_i)} \tag{4.16}$$

And from conditional probability I re-write the denominator as,

$$f_i(x_i) = \frac{P(x_i \mid y_+) P(y_+) - P(x_i \mid y_-) P(y_-)}{P(x_i \mid y_+) P(y_+) + P(x_i \mid y_-) P(y_-)} \tag{4.17}$$

Now I substitute the binormal distribution from (4.6) and (4.7) into (4.15),

$$f_i(x_i) = \frac{\mathcal{N}(x_i;\, \mu_{x_i+}, \sigma_{x_i+}) P(y_+) - \mathcal{N}(x_i;\, \mu_{x_i-}, \sigma_{x_i-}) P(y_-)}{\mathcal{N}(x_i;\, \mu_{x_i+}, \sigma_{x_i+}) P(y_+) + \mathcal{N}(x_i;\, \mu_{x_i-}, \sigma_{x_i-}) P(y_-)}$$

and I redefine my function f_i in terms of sample statistics, such as m_{x_i+} and s_{x_i+}; $\left(\frac{N_+}{N}\right)$ as the sample estimate of $P(y_+)$ where N_+ is the number of points in the positive class and N is the number of points overall; and $\left(1 - \frac{N_+}{N}\right)$ as the sample estimate of $P(y_-)$:

$$f_i(x_i) = \frac{\mathcal{N}(x_i;\, m_{x_i+},\, s_{x_i+}) \cdot \left(\frac{N_+}{N}\right) - \mathcal{N}(x_i;\, m_{x_i-},\, s_{x_i-}) \cdot \left(1 - \frac{N_+}{N}\right)}{\mathcal{N}(x_i;\, m_{x_i+},\, s_{x_i+}) \cdot \left(\frac{N_+}{N}\right) + \mathcal{N}(x_i;\, m_{x_i-},\, s_{x_i-}) \cdot \left(1 - \frac{N_+}{N}\right)} \tag{4.18}$$

I then use that function $f_i(x_i)$ in each dimension of an explicit Mercer kernel as the definition of a Bayesian binormal kernel (Figure):

$$k_{\text{bbn}}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{d} f_i(x_i) f_i(z_i) \tag{4.19}$$

Finally, I note that in (4.12) and (4.15) I use a difference of probability instead of an odds ratio or log odds ratio because a difference meets the second requirement for negativity whereas a ratio, being non-negative, does not. Furthermore, a ratio (or log ratio) can become infinite as the less probable class in the denominator approaches zero, which is undesirable in a kernel.

This kernel is Mercer according to the same proof for all explicit Mercer kernels (Section 4.4).

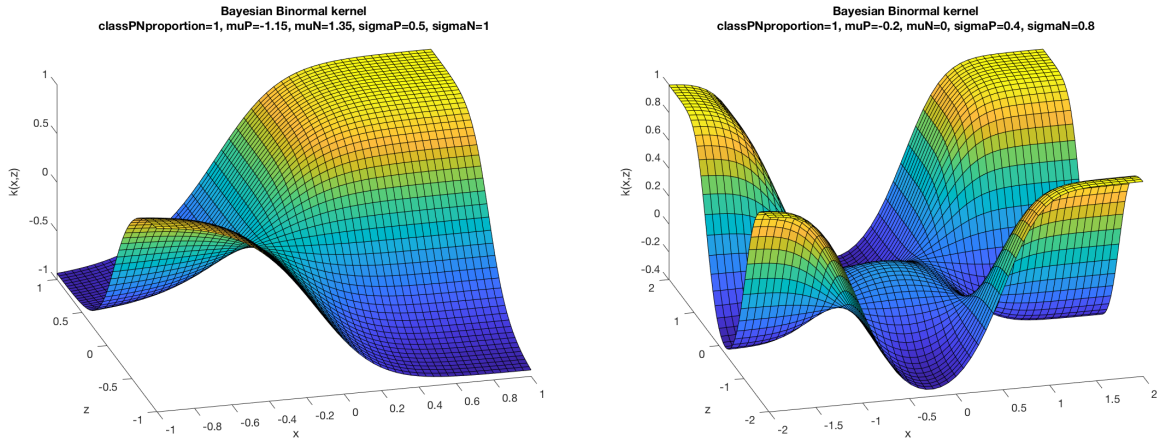

Figure 4.3: The Bayesian binormal kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ saturates to either zero or one as input feature values approach infinity.

4.5.2 Related work on Bayesian kernels

There are three kernels with major differences but minor similarities to my proposed binormal kernel: the likelihood ratio (LR) based discriminant kernel for speaker verification [50], the Fisher kernel for classification of protein homologies [135] and the TOP kernel. They operate on complex structured data while my kernel is independently derived for real-valued data. I discuss these kernels first and then discuss kernels which I identified in a subsequent review of the literature.

The LR based discriminant kernel is based on log likelihood ratios, whereas mine is based on a product (or covariance) of differences of an approximation to the posterior probability (where the likelihood and prior are used as a common approximation). Their kernel is derived from speech characteristics modelled as a Gaussian mixture model of **any number** of unimodal **multivariate** Gaussian densities parameterized by a mean vector and **covariance** matrix. My kernel is based on a **univariate binormal** model in each dimension **independently**.

Jaakola and Haussler's classification method [135] uses the logarithm of the **posterior odds ratio** as the discriminant along with a Fisher kernel based on a **hidden Markov model** and **partial derivatives of the log likelihood** (also known as the Fisher score). In contrast, my prototype binormal kernel uses the **difference** in posterior probabilities for the kernel as a probabilistic measure of similarity to one class versus the other. Also a Markov model is not applicable for general data without known or suspected structure or relationships, as with pixels in images, for example.

The TOP kernel is similar to Jaakola and Haussler's Fisher kernel method, since the TOP kernel is also based on the **posterior odds ratio** [270] and differs from the difference used in my kernel.

The binormal kernel I propose is a generative model and a data-dependent kernel, where the latter means that it uses statistics computed on the data (often separately or apriori) as kernel hyperparameters, like the Fisher kernel. Other data-dependent kernels include the quotient basis kernel [215] which is used in classification with two ontology-based health indices, and the context-dependent kernel [222] which uses scale-invariant feature transform (SIFT) information.

In a subsequent review of the literature I identify two kernels [297, 79] which are more closely related to my Bayesian binormal kernel, since they include the concept of binormality. My kernel has advantages over these two kernels because of its independent derivation which takes a unique approach—focusing on kernels instead of feature shaping (4.4.4).

4.5.3 Joint probability binormal kernel (proposed)

I derive a joint probability binormal kernel from the same three covariance requirements as the Bayesian binormal kernel, resulting in the following equations,

$$k_i(x_i, z_i) = f_i(x_i) f_i(z_i) \tag{4.20}$$

$$= [P(y_+ \mid x_i) - P(y_- \mid x_i)] \cdot [P(y_+ \mid z_i) - P(y_- \mid z_i)] \tag{4.21}$$

as a repetition of equations (4.13) and (4.14).

However, whereas the Bayesian binormal kernel then applies Bayes theorem, this kernel applies a common approximation to Bayes theorem—the joint probability,

$$P(y \mid x) \approx P(x \mid y) P(y)$$

which omits the denominator $P(x)$ as a common approximation. We therefore re-write $f_i(x_i)$ in (4.20) as

$$f_i(x_i) = P(x_i \mid y_+) P(y_+) - P(x_i \mid y_-) P(y_-) \tag{4.22}$$

This revised equation (4.22) also meets all three covariance requirements, if I describe the third requirement in a more general form instead of a simplistic (linear) proportion. That is, when I remove the common denominator $P(x_i)$, which is not a constant, it no longer

obeys a (linear) proportion, but the magnitude of the covariance (or each of the factors in the covariance) increases or decreases together with the difference in probabilities, as a more general requirement, i.e., they share the same sign in the first derivative:

$$\operatorname{sign}\left(\frac{d}{dx_i} f_i\left(x_i\right)\right) = \operatorname{sign}\left\{\frac{d}{dx_i}\left(P\left(y_+ \mid x_i\right) - P\left(y_- \mid x_i\right)\right)\right\} \tag{4.23}$$

Now I substitute the binormal distribution from (4.6) and (4.7) into (4.22),

$$f_i\left(x_i\right) = \mathcal{N}\left(\mu_{x_i+}, \sigma_{x_i+}\right) P\left(y_+\right) - \mathcal{N}\left(\mu_{x_i-}, \sigma_{x_i-}\right) P\left(y_-\right)$$

and I redefine the function f_i in terms of sample statistics $\bar{x}_+$ and s_{x_+}; $\left(\frac{N_+}{N}\right)$ as the sample estimate of $P\left(y_+\right)$; and $\left(1 - \frac{N_+}{N}\right)$ as the sample estimate of $P\left(y_-\right)$:

$$f_i\left(x_i\right) = \mathcal{N}\left(\bar{x}_{i+},\, s_{x_{i+}}\right) \cdot \left(\frac{N_+}{N}\right) - \mathcal{N}\left(\bar{x}_{i-},\, s_{x_{i-}}\right) \cdot \left(1 - \frac{N_+}{N}\right) \tag{4.24}$$

where N_+ is the number of points in the positive class and N is the number of points overall. I therefore end up with the following joint probability binormal kernel (Figure 4.4) derived from covariance requirements for real numbers:

$$k_{\text{pbn}}\left(\mathbf{x}, \mathbf{z}\right) = \sum_{i=1}^{d} f_i\left(x_i\right) f_i\left(z_i\right) \tag{4.25}$$

$$= \sum_{i=1}^{d} \left[\mathcal{N}\left(\bar{x}_{i+},\, s_{x_{i+}}\right) \cdot \left(\frac{N_+}{N}\right) - \mathcal{N}\left(\bar{x}_{i-},\, s_{x_{i-}}\right) \cdot \left(1 - \frac{N_+}{N}\right)\right] \cdot \tag{4.26}$$

$$\left[\mathcal{N}\left(\bar{z}_{i+}\,,\, s_{z_{i+}}\right) \cdot \left(\frac{N_+}{N}\right) - \mathcal{N}\left(\bar{z}_{i-}\,,\, s_{z_{i-}}\right) \cdot \left(1 - \frac{N_+}{N}\right)\right] \tag{4.27}$$

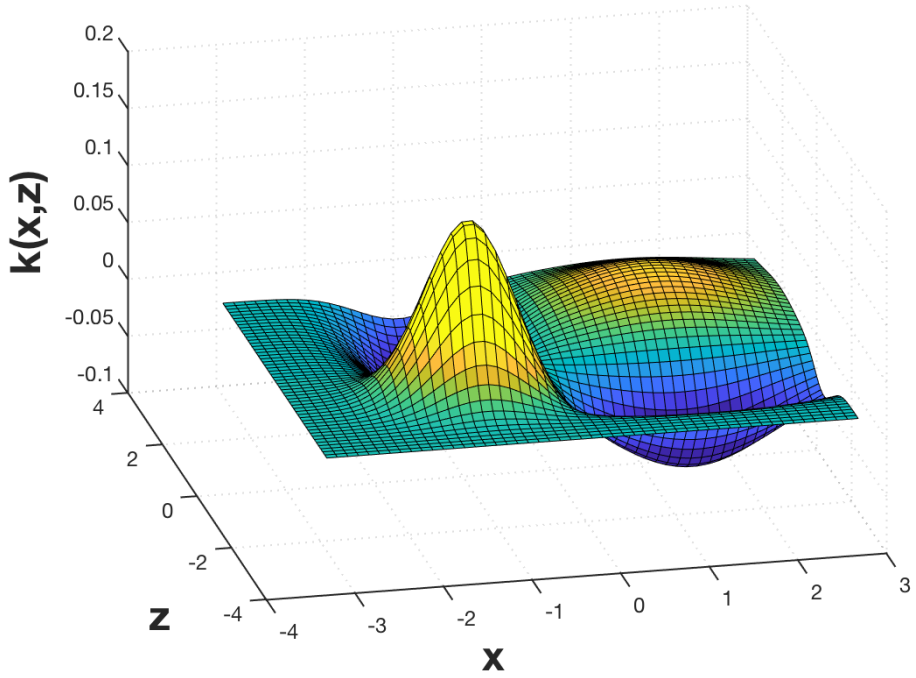

Figure 4.4: The joint probability binormal kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ saturates to zero as any one or more input feature values approach infinity.

4.5.4 Bayesian density kernel (proposed)

Whereas the Bayesian binormal kernel assumes a Gaussian distribution for the positive and negative class in (4.18), the Bayesian density kernel uses kernel density estimates $p_{\text{kde}}(x_i \mid y_+)$ and $p_{\text{kde}}(x_i \mid y_-)$ respectively:

$$f_i(x_i) = \frac{p_{\text{kde}}(x_i \mid y_+)\left(\frac{N_+}{N}\right) - p_{\text{kde}}(x_i \mid y_-)\left(1 - \frac{N_+}{N}\right)}{p_{\text{kde}}(x_i \mid y_+)\left(\frac{N_+}{N}\right) + p_{\text{kde}}(x_i \mid y_-)\left(1 - \frac{N_+}{N}\right)} \tag{4.28}$$

$$k_{\text{bd}}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} f_i(x_i)\, f_i(z_i) \tag{4.29}$$

4.5.5 Joint probability density kernel (proposed)

Whereas the joint probability binormal kernel assumes a Gaussian distribution for the positive and negative class in (4.18), the join probability density kernel uses kernel density

estimates $p_{\text{kde}}(x_i \mid y_+)$ and $p_{\text{kde}}(x_i \mid y_-)$ respectively:

$$f_i(x_i) = p_{\text{kde}}(x_i \mid y_+)\left(\frac{N_+}{N}\right) - p_{\text{kde}}(x_i \mid y_-)\left(1 - \frac{N_+}{N}\right) \tag{4.30}$$

$$k_{\text{jpd}}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} f_i(x_i) f_i(z_i) \tag{4.31}$$

4.6 Designing new kernels for nominals and presence-only binary data

In this section I design four kernels for **nominals** (as-is, or converted to binary) and **presence-only binary** data (which occur in medical data sets, e.g., the skin cancer data set we use, and which reflect nominals when converted to binary). These kernels fill gaps (Section 3.5) in kernel requirements in various combinations (Table 4.3 on page 108) corresponding to the requirements, characteristics and/or priorities for learning with a particular data set.

In machine learning it is standard practice to convert nominals with d categories or levels into d binary indicators [29, 128][3]. Recall from **asymmetric match weighting** (3.4.2) that there is a key characteristic of nominals converted to binary indicators and presence-only binary indicators in similarity testing—I should only count matching positive values, not negative values or mismatches.

4.6.1 Delta composite kernel (proposed)

A straightforward approach is to apply a delta kernel [16] to the set of binary indicators for each nominal, separately, and not to other data types. If I define a function based on the prior knowledge of which features represent which nominals,

$$\text{nom}(x_i) = \begin{cases} q & x_i \text{ is a binary indicator for the } q^{th} \text{ nominal} \\ 0 & \text{otherwise} \end{cases}$$

[3] sometimes called the one-attribute-per-value approach

then for the q^{th} nominal I can use the delta kernel,

$$k_q(\underline{x}, \underline{z}) = \prod_{\{i|\text{nom}(x_i)=q\}} \delta(x_i, z_i)$$

which is a product of ones if all indicators match, or zero overall if any indicator mismatches. I then take the sum across multiple nominals, i.e., **a composite kernel**,

$$k(\underline{x}, \underline{z}) = \sum_q \prod_{\{i|\text{nom}(x_i)=q\}} \delta(x_i, z_i)$$

This kernel is for **converted nominals** and **presence-only binary** data, where the binary indicators are encoded either as $\{0, 1\}$ or $\{-1, +1\}$.

4.6.2 Positive match kernel (proposed)

It would be preferable to have a kernel for nominals without meta data or with less meta data, so I define one for binary indicators coded as either $\{0, 1\}$ or $\{-1, +1\}$. Let us refer to $x_i = z_i = +1$ as a positive match, $x_i \neq z_i$ as a mismatch and $x_i = z_i = 0$ or $x_i = z_i = -1$ as a negative match. Rather than checking if every level matches as in the previous kernel, if I count only the positive matches,

$$p(x_i, z_i) = \begin{cases} 1 & \text{if } x_i = z_i = +1 \\ 0 & \text{otherwise} \end{cases}$$

I can get the same result without tracking as much information:

$$k(\underline{x}, \underline{z}) = \sum_{\{i|\text{is\_nom}(x_i)\}} p(x_i, z_i)$$

This kernel is for **converted nominals** and **presence-only binary** data, and handles binary indicators coded either as $\{0, 1\}$ or $\{-1, +1\}$.

4.6.3 Orthant sigmoid kernel (proposed)

For **converted nominals** and **presence-only binary** data I design the **orthant sigmoid kernel** to meet a requirement for **asymmetric match weighting** (Section 3.4.2)—i.e., to focus only on positive matches when $c = 1$, or only on negative matches when $c = -1$, or any weighted mixture of both. It meets the requirement with approximate values (Figure 4.5).

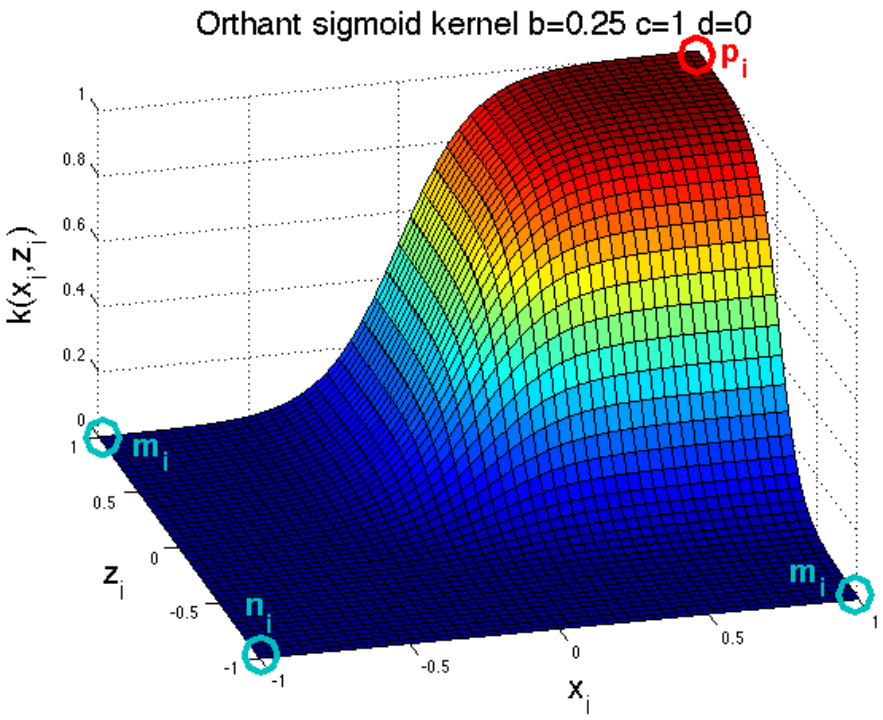

Figure 4.5: The orthant sigmoid kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ when $c = 1$ counts positive matches p_i (red circle) with an output value ≈ 1 in the all positive orthant and an output value ≈ 0 in other orthants (light blue circles) for $\{-1, +1\}$ encoding.

The orthant sigmoid kernel $k_{\text{OSig}}(\mathbf{x},\ \mathbf{z})$ is derived from my proposed Mercer sigmoid kernel (Section 4.7.1) [45], omitting $\frac{1}{p}$, and adding the hyperparameter c to the basis function:

$$k(\mathbf{x},\ \mathbf{z}) = \sum_{i=1}^{n} \left(\left[\tanh\left(\frac{x_i - d}{b} \right) + c \right] \cdot \left[\tanh\left(\frac{z_i - d}{b} \right) + c \right] \right)$$

The hyperparameter $c \in [-1, +1]$ controls how the kernel is balanced between sensing positive matches in the all positive orthant ($x_i, z_i > 0$) versus negative matches in the all negative orthant ($x_i, z_i < 0$). For $c = -1$ it is a negative match kernel and for $c = 0$ it degenerates to the Mercer sigmoid kernel (without normalization). For values in between the kernel places more weight on one type of match versus the other.

Normalization is applied within each dimension (see the Appendix for the derivation), to arrive at the following definition for the kernel:

$$k_{\text{OSig}}(\mathbf{x},\ \mathbf{z}) \triangleq \frac{1}{2(|c|+1)} \left\{ \sum_{i=1}^{n} \left(\left[\tanh\left(\frac{x_i - d}{b} \right) + c \right] \cdot \left[\tanh\left(\frac{z_i - d}{b} \right) + c \right] \right) + n\left(1 - c^2\right) \right\} \tag{4.32}$$

In the design of this kernel I did **not** add or include normalization of the output to the range $(0, 1)$ across all n feature space dimensions, on purpose, since it may be used more effectively in a composite kernel (e.g., 4.8.4) that way. To normalize it for other purposes, divide the kernel by the dimensionality n.

The kernel hyperparameters are as follows:

- b scales the kernel horizontally in each dimension, such that smaller values of b result in a steeper slope and larger values of b result in a slope which is less steep, i.e., a lower grade.
- d shifts the kernel in the horizontal plane along the $x_i = z_i$ axis in each dimension, toward the positive orthant for $d > 0$. If the features are properly centered then $d = 0$ should be optimally aligned with the kernel's geometry and behaviour, i.e., this hyperparameter should be unnecessary.
- c specifies the asymmetric weighting of the kernel in the domain $[-1, +1]$. For $c = 1$ as a maximum value, the kernel is weighted for matches or covariance only in the positive orthant only (i.e., x_i, $z > 0$); while for $c = -1$ as a minimum value, the kernel is weighted for matches or covariance only in the negative orthant only (i.e., x_i, $z < 0$). $c = 0$ achieves symmetry and reduces to the Mercer sigmoid kernel. At other values, both positive matches and negative matches are weighted with imbalance.

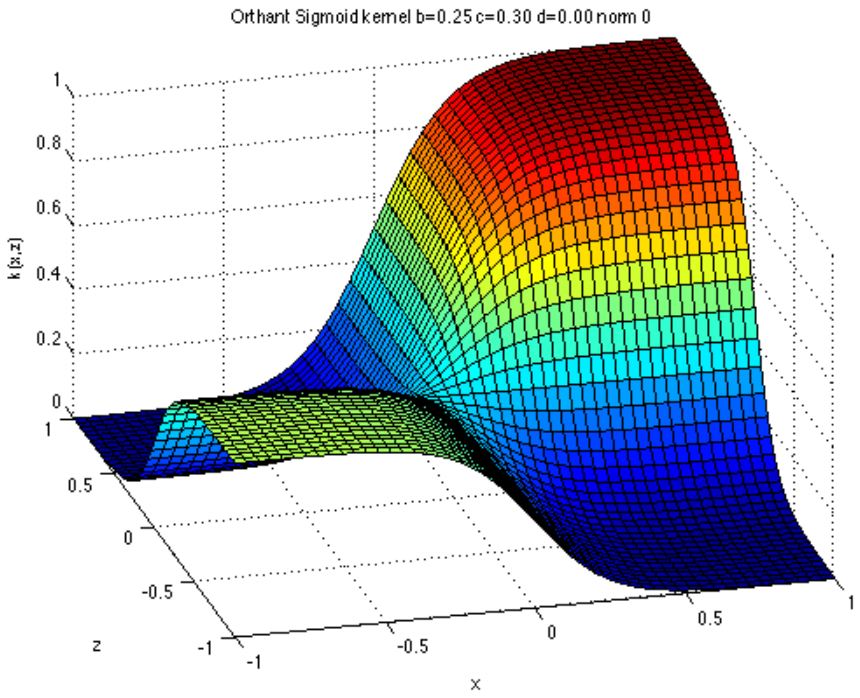

Figure 4.6: Orthant sigmoid kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with horizontal (xz plane) scale $b = 0.25$ and $c = 0.3$ for positive matches weighted more than negative matches.

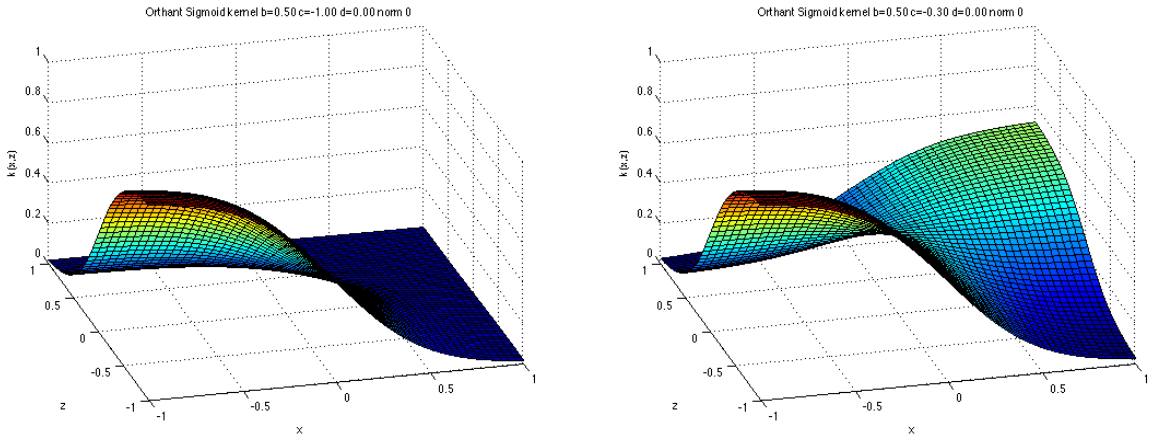

Figure 4.7: Orthant sigmoid kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with horizontal (xz plane) scale $b = 0.5$ causing smoother curves. $c = -1$ for negative-match weighting and $c = -0.3$ for asymmetric negative and positive match weighting, respectively.

4.6.4 Orthant linear kernel (proposed)

An exact result is achieved when I add **asymmetric match weighting** (3.4.2) to the linear kernel, creating the orthant linear kernel (Figure 4.8).

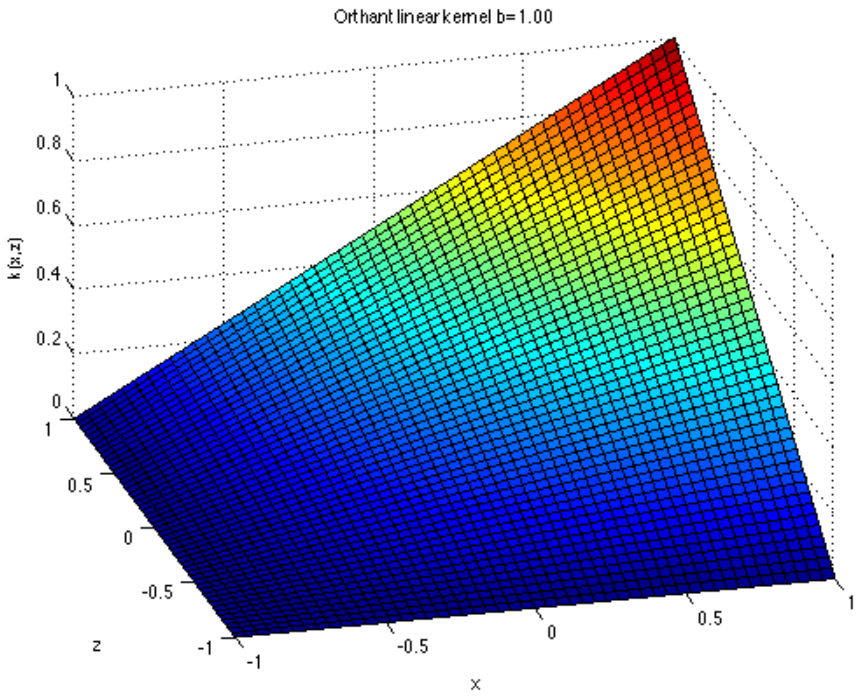

Figure 4.8: Orthant linear kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with $b = 1$ also counts positive matches with an output value $= 1$ in the all positive orthant and an output value $= 0$ in other orthants for $\{-1, +1\}$ encoding.

The orthant linear kernel $k_{\text{OLin}}(\mathbf{x},\ \mathbf{z})$ is defined by:

$$\begin{aligned} k(\mathbf{x},\ \mathbf{z}) &= k_{\text{lin},c}\left((\mathbf{x}+\mathbf{c}),\ (\mathbf{z}+\mathbf{c})\right) + c' \quad \text{where } \mathbf{c} = \begin{bmatrix} c & c & \dots & c \end{bmatrix}^{\text{T}} \\ &= \langle (\mathbf{x}+\mathbf{c}),\ (\mathbf{z}+\mathbf{c}) \rangle + c' \\ &= \sum_{i=1}^{n} \left([x_i + c] \cdot [z_i + c]\right) + c' \end{aligned}$$

however, within each feature dimension it is normalized and defined as follows:

$$\begin{aligned} k_{\text{OLin}}(\mathbf{x},\ \mathbf{z}) &\triangleq \frac{1}{2\left(|c|+1\right)} \cdot \left\{ \langle (\mathbf{x}+\mathbf{c}),\ (\mathbf{z}+\mathbf{c}) \rangle + n\left(1-c^2\right) \right\} + c' \quad \text{where } \mathbf{c} = \begin{bmatrix} c & c & \dots & c \end{bmatrix}^{\text{T}} \\ &= \frac{1}{2\left(|c|+1\right)} \cdot \left\{ \sum_{i=1}^{n} \left([x_i + c] \cdot [z_i + c]\right) + n\left(1-c^2\right) \right\} + c' \end{aligned}$$

In the design of this kernel I did **not** add or include normalization of the output to the range $(0, 1)$ across all n feature space dimensions, on purpose, since it may be used more effectively that way, in a composite kernel (like the orthant sigmoid kernel is used in 4.8.4). To normalize it for other purposes, divide the kernel by the dimensionality n.

The kernel hyperparameters are as follows:

- c specifies the asymmetric weighting of the kernel in the domain $[-1, +1]$. For $c = 1$ as a maximum value, the kernel is weighted toward matches or covariance in the positive orthant; while for $c = -1$ as a minimum value, the kernel is weighted toward matches or covariance in the negative orthant. $c = 0$ achieves symmetry and reduces to the linear kernel.
- c' specifies a vertical bias

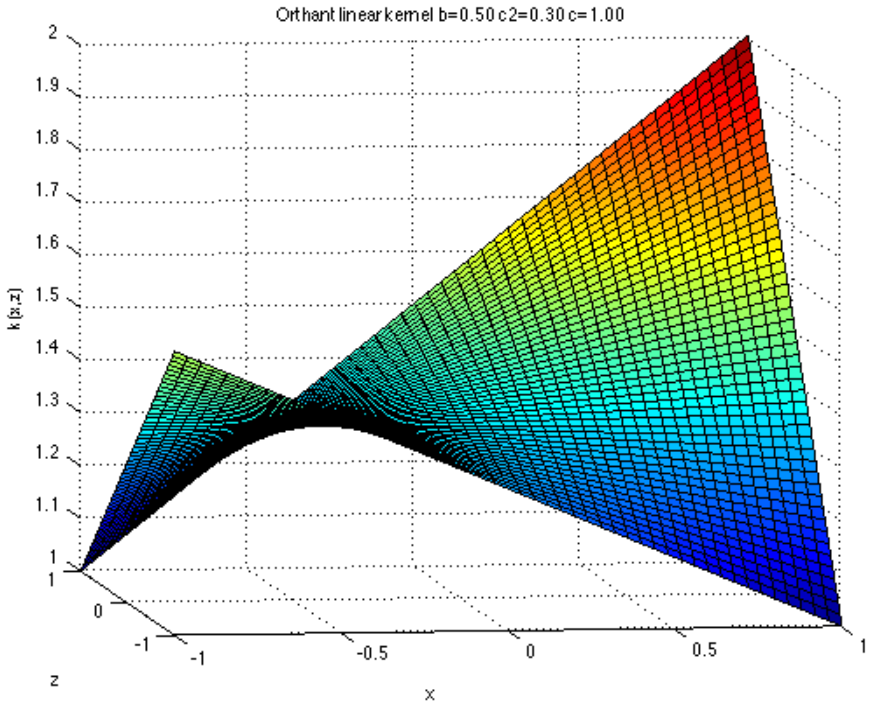

Figure 4.9: Orthant linear kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with horizontal scale $b = 0.5$.

4.7 Matching new and existing kernels to binary data

Aside from deriving and designing kernels I can also see whether existing numeric kernels (e.g., for reals) match the requirements of data types. I match or analyze the match between kernels and binary data.

If I select a machine learning method and model that handles continuous values, then I can treat features of any atomic data type as continuous, by following three steps described in Section A.8.

For binary data, Hamann similarity [53] as a simdist function adds the number of positive matches and negative matches, then subtracts the number of mismatches, and normalizes the result by the number of features.

My new proposed Mercer sigmoid kernel (the next subsection below) approximately meets the same requirements fulfilled by Hamann similarity, including normalization, with hyperparameters $b < \frac{1}{3}$, $d = 0$. The linear kernel, $k_{\text{Lin}}(\underline{x}, \underline{z})$, exactly matches the requirement with similar values output in the same highlighted corners/orthants as the Mercer sigmoid.

4.7.1 Mercer sigmoid kernel (proposed)

The purpose of the Mercer sigmoid kernel is to provide an admissible kernel as a substitute for the sigmoid kernel (Section 2.8.4) which is not admissible for a range of hyperparameters—a range which is difficult to determine [167, 45].

A normalized version of the sigmoid kernel which is still inadmissible, often achieves better accuracy than the sigmoid kernel and the Gaussian RBF kernel, hence its use is desired, but it is also not necessarily trustworthy for health care applications [45].

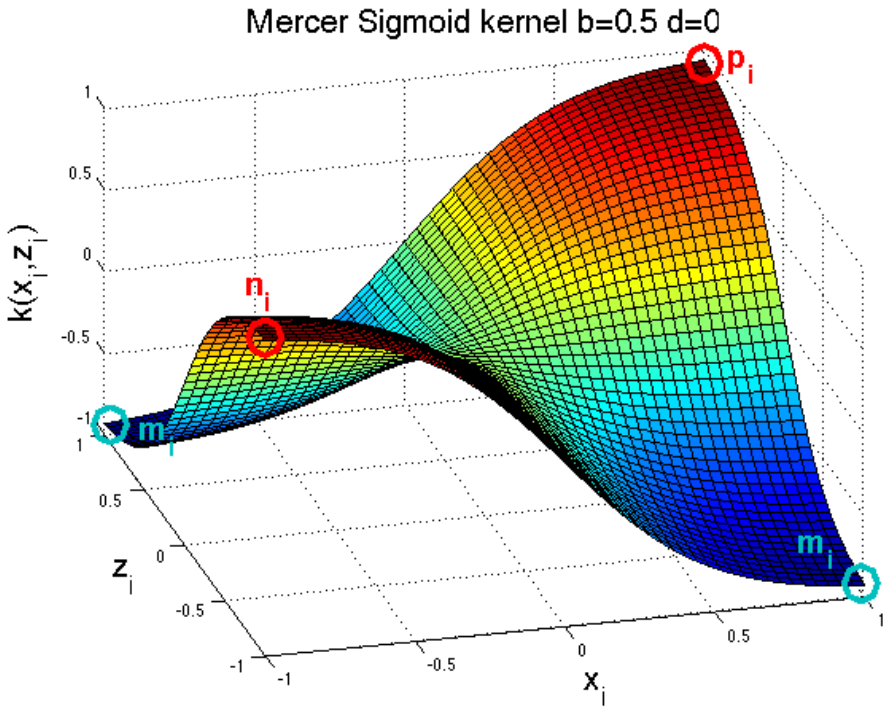

Figure 4.10: The Mercer sigmoid kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with $b = 0.5$, $d = 0$, adds positive and negative matches for inputs $(-1, -1)$ and $(+1, +1)$ and subtracts mismatches $(+1, -1)$ and $(-1, +1)$ in the other two corners/orthants.

To understand the Mercer sigmoid kernel, one must first understand the sigmoid kernel, and preceding that a sigmoid function, which is also called an S-curve. An S-curve or sigmoid function $s(x)$ is also called a squashing function because for an input which is high or infinite on either the positive or negative side the sigmoid function squashes the value down to +1 or -1 respectively, i.e., $s(\infty) = +1$ and $s(-\infty) = -1$.

A sigmoid kernel has two inputs $k(x, z)$ but always uses them together as a product $k(x, z) = s(xz)$. I consider only scalar inputs not vector inputs, initially, for ease of illustration. The intuition behind the Mercer sigmoid is that squashing the product xz can be approximated by the product of squashing x and z separately. That is, $s(xz)$ can be approximated by $s(x)\, s(z)$. For a vector, the squashing is done in each dimension.

The Mercer sigmoid kernel $k_{\text{MSig}}(\mathbf{x}, \mathbf{z})$ is positive definite and is defined by:

$$k_{\text{MSig}}(\underline{x}, \underline{z}) \triangleq \frac{1}{n}\sum_{i=1}^{n} \tanh\left(\frac{x_i - d}{b}\right) \cdot \tanh\left(\frac{z_i - d}{b}\right), \quad \underline{x} \in \mathbb{R}^p \tag{4.33}$$

where the parameter b specifies the kernel width, d specifies a horizontal shift and n is the number of features in $\underline{x}$ or $\underline{z}$. The normalization by p has (virtually[4]) no effect on SVM classification but is included for better interpretability in plots and proper weighting in composite and multiple kernel learning.

I can also define a non-uniform Mercer sigmoid kernel with d_i in place of d, and b_i in place of b:

$$k_{\text{MSigNU}}(\underline{x}, \underline{z}) \triangleq \frac{1}{n}\sum_{i=1}^{n} \tanh\left(\frac{x_i - d_i}{b_i}\right) \cdot \tanh\left(\frac{z_i - d_i}{b_i}\right), \quad \underline{x} \in \mathbb{R}^p \tag{4.34}$$

4.8 Matching new and existing kernels to nominals and presence-only binary data

Two kernels approximately match **nominals** converted to binary indicators and **presence-only binary** data.

My proposed new Mercer sigmoid kernel, when scaled and shifted, i.e., $\frac{1}{2}(k_{\text{MSig}}(\underline{x}, \underline{z}) + 1)$, approximately matches the requirement separately from other data with hyperparameters $b < \frac{1}{3}$, $d = 0$. It must be applied to other data separately or else data are weighted unevenly—i.e., it may be applied within a disjoint composite kernel as in 4.8.4.

[4] intuitively it should have no effect but empirically results are not exactly the same

Also, the existing power distance kernel when scaled, $\frac{1}{8} \cdot k_{\text{Pwr}}(\underline{x}, \underline{z})$, approximately matches the requirement separately from other data with hyperparameter $\beta = 2$. It must be applied to other data separately or else data are weighted unevenly—i.e., it may be applied within a disjoint composite kernel.

4.8.1 Insensitive sigmoid variant kernels (proposed)

This kernel is a precursor to the next kernel which applies to positive match requirements. The intuition behind the insensitive sigmoid variant (ISV) kernel is to create a more flexible version of a sigmoid kernel—a more general form with the possibility to fit data better as a consequence. Whereas a sigmoid kernel is based on an S-curve or sigmoid function, I define an ISV function which generalizes that to a family of curves between an S-curve and a double S-curve. A double S-curve looks like an S-curve but with a flat plateau in the center (Figure 4.11).

The Mercer sigmoid kernel was born out of this more complex kernel. In classification with binary targets, the ISV kernel matches the requirement to reduce the effect of points in the region of class overlap, whose labels are less certain, and therefore may represent label error or class noise. The ISV kernel, like the Mercer sigmoid kernel, matches binary data or binary data imputed as continuous values.

In binary classification, if data are balanced and centered then the region of overlap is at the origin of the kernel. If data are imbalanced then allowing the kernel to shift its origin with the d hyperparameter is prudent. I mitigate the class noise by treating points in the region of interest with no weight — that is, I design a kernel which outputs values of zero with a flat plateau (at the centre of the region). Instead of assigning no weight with a flat plateau, I can assign less weight, i.e., a kernel with low height from a gradual slope in the centre of the region.

I design the required kernel geometry by first creating a flat region in a one dimensional sigmoid function. In the literature, there is a function that is a sigmoid with a plateau—a double S-curve, however it does not have a flexible width parameter separate from the overall scale and slope, and it is piece-wise defined with a $\text{sign}(\cdot)$ function that is not differentiable at all points (i.e., at the origin).

To achieve the shape desired I take a sigmoid basis function (with parameters b and d for horizontal shifting and scaling respectively),

$$\sigma_{bd}(x) \triangleq \tanh\left(\frac{x-d}{b}\right)$$

and add to it, a continuous bimodal odd-symmetric function which cancels out the sigmoid in the center to create a flat (or zero slope) region when $a = 1$,

$$\mathcal{N}'_{a,b,d}(x) = -\frac{(x-d)}{ab} exp\left(-\left(\frac{x-d}{b}\right)^2\right)$$

In this case the function is a Gaussian derivative (with the same horizontal shift and scale parameters b and d). It is also has a vertical scaling parameter $\frac{1}{a}$, which causes a positive slope in the center for $a > 1$, or zero slope for $a = 1$ and degenerates to a sigmoid with no plateau (the most positive slope achievable) when $a \geq \exp(1)$.

The resulting model is used as a basis function within each dimension (denoted with a "g" for its Gaussian derivation),

$$\phi_{\mathrm{g}}(x) = \tanh\left(\frac{x-d}{b}\right) - \frac{(x-d)}{ab} exp\left(-\left(\frac{x-d}{b}\right)^2\right)$$

This function is continuously defined and parameterized in a very flexible manner. I use this function to define a p-dimensional vector,

$$\vec{\phi}_{\mathrm{g}}(\mathbf{x}) = \begin{bmatrix} \phi_{\mathrm{g}}(x_1) & \phi_{\mathrm{g}}(x_2) & ... & \phi_{\mathrm{g}}(x_p) \end{bmatrix}^T$$

for the kernel k_{ISVg},

$$\begin{aligned} k_{\mathrm{ISVg}}(\mathbf{x}, \mathbf{z}) &\triangleq \frac{1}{p} \cdot \left\langle \vec{\phi}_{\mathrm{g}}(\mathbf{x}),\ \vec{\phi}_{\mathrm{g}}(\mathbf{z}) \right\rangle \\ &= \frac{1}{p} \sum_{i=1}^{p} \phi_{\mathrm{g}}(x_i) \cdot \phi_{\mathrm{g}}(z_i) \\ &= \frac{1}{p} \sum_{i=1}^{p} \left\{ \tanh\left(\frac{x_i-d}{b}\right) - \frac{(x_i-d)}{ab} exp\left(-\left(\frac{x_i-d}{b}\right)^2\right) \right\} \cdot \left\{ \tanh\left(\frac{z_i-d}{b}\right) \right. \\ &\qquad \left. - \frac{(z_i-d)}{ab} exp\left(-\left(\frac{z_i-d}{b}\right)^2\right) \right\} \end{aligned}$$

I may similarly define a kernel, k_{ISVh} instead of k_{ISVg}, by using a function $f_{\mathrm{h}}(x)$ instead of $\mathcal{N}'_{a,b,d}(x)$ — that is, $f_{\mathrm{h}}(x)$, which is the second derivative of a hyperbolic tangent, also has the required shape as a continuous bimodal odd-symmetric function:

$$f_{\mathrm{h}}(x) = -\tanh\left(\frac{x-d}{b}\right) \frac{1}{a} \mathrm{sech}^2\left(\frac{x-d}{b}\right)$$

I add this function $f_{\mathrm{h}}(x)$, to flatten or cancel out slope in tanh, and arrive at the following

basis function (denoted with an "h" for its hyperbolic tangent derivation),

$$\begin{aligned}\phi_{\mathrm{h}}(x) &= \tanh\left(\frac{x-d}{b}\right) - \tanh\left(\frac{x-d}{b}\right)\frac{1}{a}\mathrm{sech}^2\left(\frac{x-d}{b}\right) \\ &= \tanh\left(\frac{x-d}{b}\right)\left(1 - \frac{1}{a}\mathrm{sech}^2\left(\frac{x-d}{b}\right)\right)\end{aligned}$$

This function is also continuously defined and parameterized in a very flexible manner. I use this function to define a p-dimensional vector,

$$\vec{\phi}_{\mathrm{h}}(\mathbf{x}) = \left[\begin{array}{cccc} \phi_{\mathrm{h}}(x_1) & \phi_{\mathrm{h}}(x_2) & \ldots & \phi_{\mathrm{h}}(x_p) \end{array}\right]^T$$

for the kernel k_{ISVh},

$$\begin{aligned}k_{\mathrm{ISVh}}(\mathbf{x},\mathbf{z}) &\triangleq \frac{1}{p}\cdot\left\langle \vec{\phi}_{\mathrm{h}}(\mathbf{x}),\ \vec{\phi}_{\mathrm{h}}(\mathbf{z})\right\rangle \\ &= \frac{1}{p}\sum_{i=1}^{p}\phi_{\mathrm{h}}(x_i)\cdot\phi_{\mathrm{h}}(z_i) \\ &= \frac{1}{p}\sum_{i=1}^{p}\left\{\tanh\left(\frac{x_i-d}{b}\right)\cdot\left(1-\frac{1}{a}\mathrm{sech}^2\left(\frac{x_i-d}{b}\right)\right)\right\}\cdot\left\{\tanh\left(\frac{z_i-d}{b}\right)\right. \\ &\qquad\qquad \left.\cdot\left(1-\frac{1}{a}\mathrm{sech}^2\left(\frac{z_i-d}{b}\right)\right)\right\}\end{aligned}$$

The kernel hyperparameters are as follows:

- b scales the kernel horizontally in each dimension, where the sigmoid is twice as wide for $b = 2$
- d shifts the kernel in the horizontal plane along the $x_i = z_i$ axis in each dimension, toward the positive orthant for $d > 0$. If the features are properly centered then $d = 0$ should be optimally aligned with the kernel's geometry and behaviour, i.e., this hyperparameter should be unnecessary.
- a controls the slope of the central region by increasing or decreasing the effect that flattens the sigmoid into a plateau. For $a = 1$ there is plateau with zero slope, while $a > 1$ causes a positive slope, and $a \geq \exp(1)$ minimizes the effect so that the kernel degenerates to a Mercer sigmoid with no plateau.

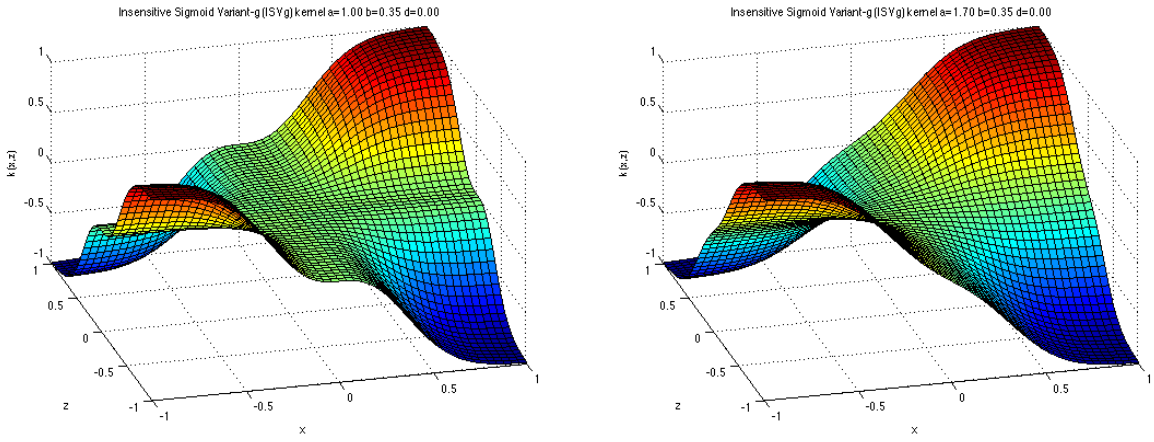

Figure 4.11: Insensitive sigmoid variant g kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with $a = 1$ for zero slope at the center of marginals, and $a = 1.7$ for positive slope at the center of marginals, respectively.

4.8.2 Orthant insensitive sigmoid variant kernels (proposed)

The orthant insensitive sigmoid variant (OISV) kernels $k_{\mathrm{OISVg}}(\mathbf{x},\, \mathbf{z})$ and $k_{\mathrm{OISVh}}(\mathbf{x},\, \mathbf{z})$ are kernels which meet requirements for asymmetric match weighting (as discussed in 3.4.2) with class noise mitigation, when $c = 1$, versus the insensitive sigmoid variant (ISV) kernels which are match kernels with class noise mitigation. They are derived from the ISV kernels by adding the hyperparameter c to the basis function,

$$\phi_{\mathrm{g},c}(x) = \tanh\left(\frac{x-d}{b}\right) - \frac{(x-d)}{ab} exp\left(-\left(\frac{x-d}{b}\right)^2\right) + c$$

The hyperparameter $c \in [-1, +1]$ controls how the kernel is balanced between sensing positive matches in the all positive orthant ($x_i, z_i > 0$) versus negative matches in the all negative orthant ($x_i, z_i < 0$). For $c = -1$ it is a negative match kernel with class noise mitigation, and for $c = 0$ it degenerates to an ISV kernel. For values in between the kernel places more weight on one type of match versus the other.

In p dimensions the vector

$$\vec{\phi}_{\mathrm{g},c}(\mathbf{x}) = \left[\begin{array}{cccc} \phi_{\mathrm{g},c}(x_1) & \phi_{\mathrm{g},c}(x_2) & \ldots & \phi_{\mathrm{g},c}(x_p) \end{array}\right]^T$$

is used to create the kernel $k_{\mathrm{OISVg}}(\mathbf{x},\, \mathbf{z})$,

$$k_{\mathrm{OISVg}}(\mathbf{x}, \mathbf{z}) \triangleq \frac{1}{2(|c|+1)} \cdot \left\langle \vec{\phi}_{\mathrm{g},c}(\mathbf{x}),\, \vec{\phi}_{\mathrm{g},c}(\mathbf{z}) \right\rangle + n\left(1 - c^2\right)$$

The kernel $k_{\text{OISVh}}(\mathbf{x},\ \mathbf{z})$ is similarly derived:

$$\phi_{\text{h},c}(x) = \tanh\left(\frac{x-d}{b}\right)\left(1 - \frac{1}{a}\text{sech}^2\left(\frac{x-d}{b}\right)\right) + c$$

$$\vec{\phi_{\text{h},c}}(\mathbf{x}) = \left[\begin{array}{cccc} \phi_{\text{h},c}(x_1) & \phi_{\text{h},c}(x_2) & \ldots & \phi_{\text{h},c}(x_p) \end{array}\right]^T$$

$$k_{\text{OISVh}}(\mathbf{x},\mathbf{z}) \triangleq \frac{1}{2(|c|+1)} \cdot \left\langle \vec{\phi}_{\text{h},c}(\mathbf{x}),\ \vec{\phi}_{\text{h},c}(\mathbf{z}) \right\rangle + n\left(1 - c^2\right)$$

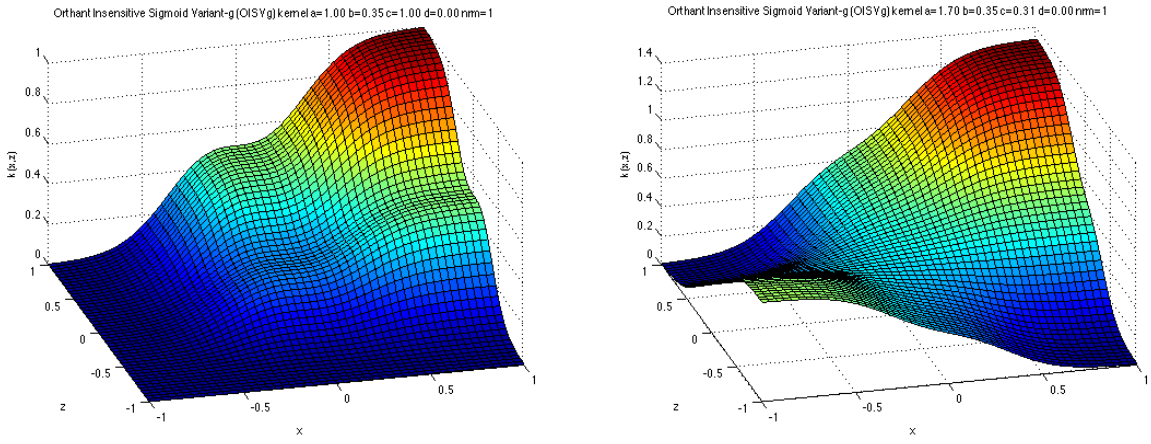

Figure 4.12: Orthant insensitive sigmoid variant g kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with $a = 1$ for zero slope at the center of marginals, and $a = 1.7$ for positive slope at the center of marginals, respectively. Also, $c = 1$ for positive-match weighting and $c = 0.31$ for asymmetric positive and negative match weighting, respectively.

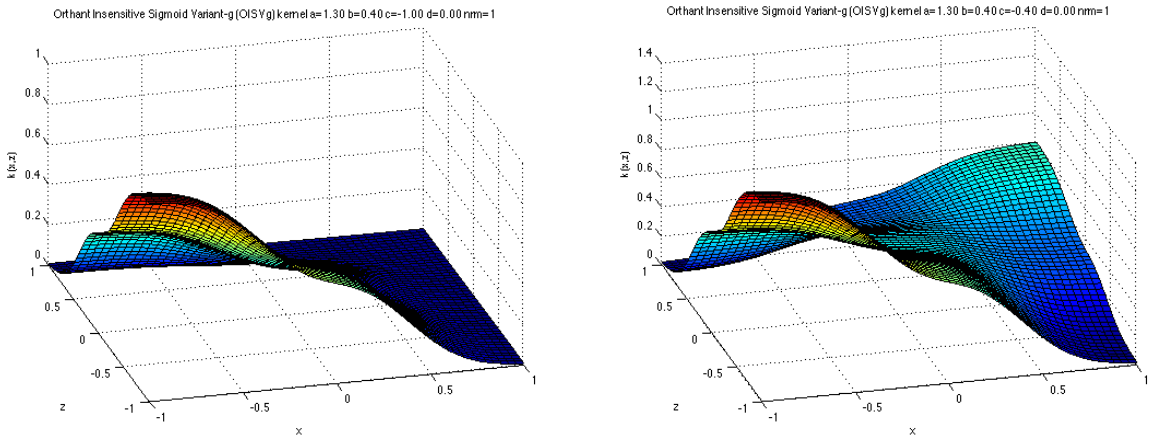

Figure 4.13: Orthant insensitive sigmoid variant g kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with $c = -1$ for negative-match weighting and $c = -0.4$ for asymmetric negative and positive match weighting, respectively.

4.8.3 Gaussian derivative kernel (proposed)

Based on a Gaussian RBF kernel, I create a kernel k_{GD} with basis functions $\vec{\phi}_{\mathrm{GD}}(\mathbf{x})$ that are bimodal in each of p dimensions (4.35). One mode pertains to data belonging to the negative class, with a negative height, and the other mode pertains to the positive class with a positive height. This geometry is simply achieved by the taking the derivative of a Gaussian (omitting a factor of 2 on the outside which has negligible effect on a basis function or kernel):

$$\phi_{\mathrm{GD}}(x) = \mathcal{N}'_{b,d}(x) = -\left(\frac{x-d}{b}\right) \cdot \exp\left(-\left(\frac{x-d}{b}\right)^2\right) \quad b > 0;\ b, d \in \mathbb{R} \tag{4.35}$$

$$\vec{\phi}_{\mathrm{GD}}(\mathbf{x}) = \begin{bmatrix} \phi_{\mathrm{GD}}(x_1) & \phi_{\mathrm{GD}}(x_2) & \dots & \phi_{\mathrm{GD}}(x_p) \end{bmatrix} \tag{4.36}$$

$$k_{\mathrm{GD}}(\mathbf{x},\ \mathbf{z}) = \frac{1}{p}\left\langle \vec{\phi}_{\mathrm{GD}}(\mathbf{x}),\ \vec{\phi}_{\mathrm{GD}}(\mathbf{z}) \right\rangle \tag{4.37}$$

$$= \frac{1}{p}\sum_{i=1}^{p} \phi_{\mathrm{GD}}(x_i) \cdot \phi_{\mathrm{GD}}(z_i) \tag{4.38}$$

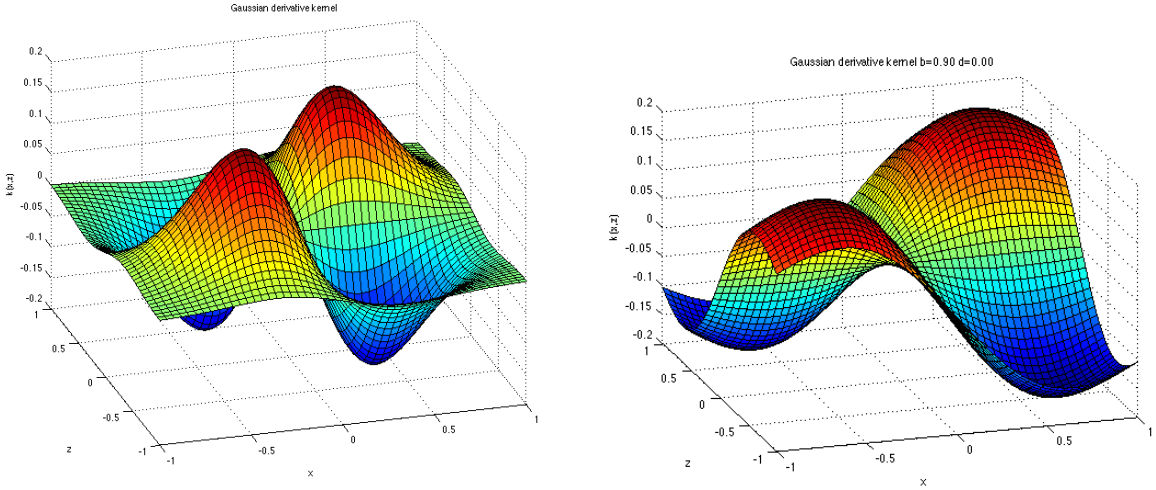

Figure 4.14: The Gaussian derivative kernel in $\mathbb{R}^1 \times \mathbb{R}^1$ with $b = 0.5$ and $b = 0.25$.

4.8.4 Orthant Mercer sigmoid binormal composite sum kernel (proposed)

I also propose the following disjoint composite kernel [256] (versus globally composite kernels [44]): the orthant/Mercer sigmoid binormal composite sum (OMBcs) kernel.

Table 4.2: Kernel components matched to data types in the OMBcs kernel

Kernel component	Data type abbreviation	Data type description
O	bin+	converted nominals or presence-only binary data
M	bin	binary data
B	real	reals, integers, ordinals, dates

The OMBcs kernel is a composite sum (cs) of an Orthant sigmoid kernel (OSig), a Mercer sigmoid kernel (MSig), and a Bayesian binormal kernel (BBN), denoted by "O", "M" and "B" respectively. It is the sum of three component kernels, where each is applied to a disjoint subset of features (Table 4.2).

The OMBcs kernel is defined as a weighted sum,

$$k_{\text{OMBcs}}(\mathbf{x},\, \mathbf{z}) \triangleq k_{\text{OSig}}\left(\{\mathbf{x},\, \mathbf{z}\}_{\text{bin+}}\right) + n_{\text{bin}} \cdot k_{\text{MSig}}\left(\{\mathbf{x},\, \mathbf{z}\}_{\text{bin}}\right) + n_{\text{real}} \cdot k_{\text{BBN}}\left(\{\mathbf{x},\, \mathbf{z}\}_{\text{real}}\right)$$

where n_{bin} is the number of binary dimensions and n_{real} is the number of real/numeric dimensions. The Orthant sigmoid kernel k_{OSig} is normalized within each feature or dimension, but not across features or dimensions. So it has a weight proportional to the number of binary+ features or dimensions. The Mercer sigmoid kernel and Bayesian binormal kernels are normalized overall, so I must apply weights to them based on the number of dimensions: n_{bin} and n_{real} (assuming that all of the n_{bin} and n_{real} dimensions should be treated equally). If one wishes to apply weights based on the correlation of features with the target, that may be done instead.

4.9 Summary

In this chapter I defined a kernel data modeling framework to address seven gaps in kernel requirements (Section 3.5). To support this framework I defined the new class of explicit Mercer kernels, needed to create transparent and atomic kernels and discussed the implications of that class. I then derived, designed and matched a variety of newly proposed kernels (and in some cases existing kernels) to various data types and distributions. The seven requirement gaps are addressed in detail (Table 4.3 on page 108) and their fulfillment are summarized as follows:

1. High: **Twelve** out of fifteen of my proposed (Mercer) kernels meet the need for admissible dot product kernels with a meaningful and finite minimum and maximum.

2. High: **Fourteen** of my proposed (Mercer) kernels meet the need for kernels with a finite feature space for transparency, like the linear and polynomial kernels, and **six** of these consistently perform better (Section 5.3) than the linear and polynomial kernels.

3. High: **Twelve** of my proposed (Mercer) kernels meet the need for explicit Mercer kernels for transparency and interpretability, in addition to the previously identified linear and polynomial kernels.

4. High: **All fifteen** of my proposed (Mercer) kernels, **and two** existing kernels meet/-match the need for kernels for heterogeneous data, whereas the Gaussian RBF kernel **does not** match the need for nominals.

   (a) In this context, I am interested in the heterogeneity between reals, binary (bin), nominals (nom), and presence-only binary (+bin) data, where there is room for improved fit and accuracy. I am **not** interested in the heterogeneity between reals, integers and ordinals where I assume a kernel for reals performs sufficiently well.

   (b) The existing kernels which meet the need are: the **linear kernel** that matches real and binary data, and the **power distance** kernel with $\beta = 2$ that matches reals, nominals and presence-only binary data.

5. Medium: **Three** of my proposed kernels—the Mercer sigmoid kernel and two insensitive sigmoid variant kernels—can be used as a safe (Mercer) alternative to the sigmoid kernel which is not admissible.

   (a) My proposed kernels have good accuracy on clinical data—sometimes better than the Gaussian RBF kernel with statistical significance and they are transparent/interpretable.

6. Medium: **Nine** of my proposed kernels have uniform functions for interpretability.

7. Medium: **Six** of my proposed kernels have asymmetric match weighting for converted nominals and presence-only binary data.

Table 4.3: My proposed kernels resolve the gaps in kernel requirements, in various combinations. I assess 15 kernels in 11 columns.

Gaps resolved	Numeric								Non-numeric	
	BBN, JPBN	BD, JPD	MSig	OSig	OISVgc, OISVhc	ISVgc, ISVhc	OLin	OMB	DC	Pos
	D	D	D	D	D	D	D	D	E	H
1.Admissible dot product with min/max meaning/finite	✓	✓	✓	✓	✓	✓		✓		
2.Finite $\mathcal{F}$ for transparency	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3.Interpretable explicit Mercer	✓	✓	✓	✓	✓	✓	✓	✓	n/a	n/a
4.Matches heterogeneous data	real bin	real bin	real bin	real nom +bin	real nom +bin	real bin	real bin	real bin nom +bin	bin nom +bin	nom +bin
5. Sigmoid alternative			✓			✓				
6. Uniform			✓	✓	✓	✓	✓		✓	✓
7. Asymmetric match weighting				✓	✓		✓	✓		✓

In contrast to the standard kernel selection method (Figure 3.1 on page 57) and alternative kernel selection methods (Figure 3.2 on page 58), I proposed a new kernel selection method (Figure 4.1 on page 72) at the beginning of this chapter, which is I can now update (Figure 4.15) below.

My proposed kernels which have a transparent form/formula, have transparent geometry in the feature space and can be matched geometrically to requirements in that feature space. Common and uncommon kernels on the other hand, can only be matched by their formulas or their geometry in the input space—where in my experience, matching is more challenging.

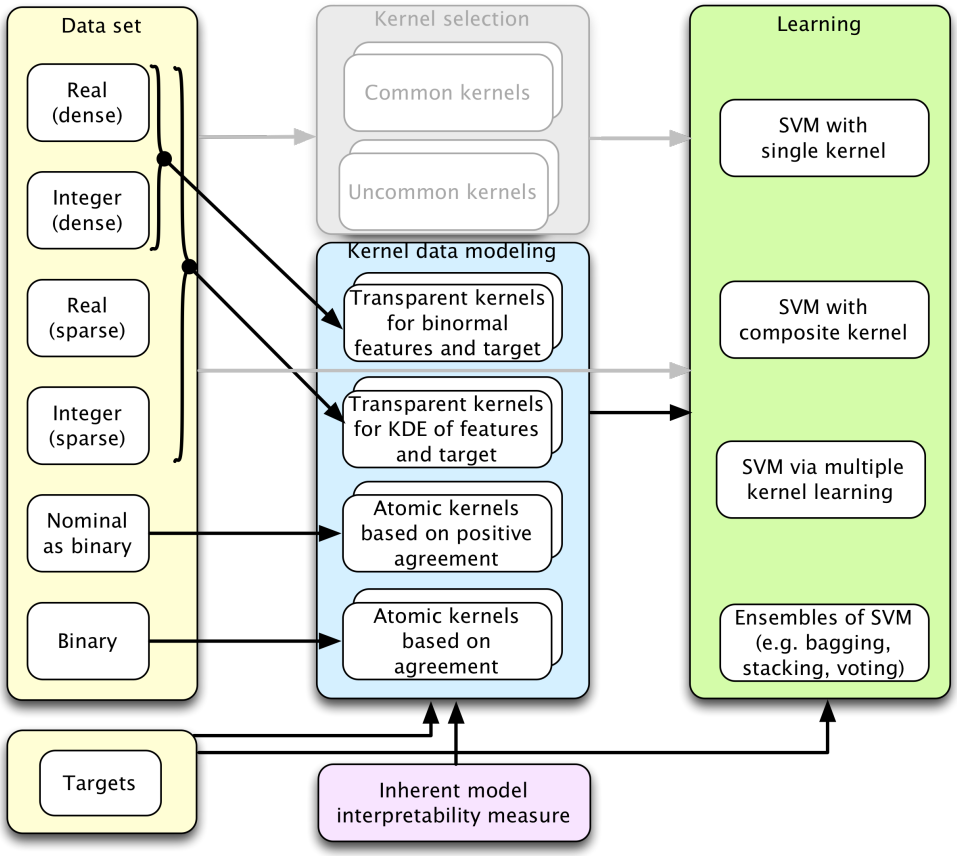

Figure 4.15: I propose the approach illustrated above (in blue and pink) for model/kernel selection in SVM.

Chapter 5

Effect of kernel data modeling

5.1 Overview of experimental data

My hypothesis applies to classification of atomic data types (4.2) in health care data. I examine the literature for benchmark health care data with mixed (heterogeneous) data.

Applicable work with mixed data, not limited to health care include the following. Aradhye and Dorai [8] use two data sets from the well-known University of California at Irvine (UCI) repository: the Mushrooms and Splice data sets; as well a data set of their own. Wong *et al.* [291] use two industry data sets (that are not health related) and three data sets from the UCI repository: the Iris Plants, Adult (Income) and Mushrooms data sets. Berrado and Runger [22] use a number of UCI data sets: Iris Plants, Ionosphere, Statlog Heart, Pima Indians, Statlog Satellite Image, Thyroid and others.

For my experiments I use publicly-available data sets (Table 5.1) commonly used in the literature as a baseline for comparison, e.g., from the University of California at Irvine (UCI) machine learning repository [11] and other sources associated with papers. These data sets, with characteristics summarized in Table 5.1, are: Skin cancer [73], Heart (Statlog, not Cleveland) [145], Pima Indians diabetes [11], Bupa liver [11], Hepatitis [11], Colon tumor [6], Prostate cancer [235].

I select commonly used (i.e., reputable) and interpretable/meaningful health care data sets, discarding data sets with conflicting information (e.g., the Thyroid data set), lack of interpretation (e.g., the Cancer data set), unclear target variables (e.g., the Diabetes data set, not to be confused with Pima Indians diabetes data set) and features which have been transformed or processed too much (e.g., the ecoli data set). I also did not use data sets

with more than two class labels (multiclass problems) although these can be transformed into permutations of two class problems.

After conducting a number of experiments, I realize two further aspects of two data sets I selected. First, the Pima Indians diabetes database has disguised missing data [204] and this is tolerable for experiments on accuracy and interpretability, but not end-use/actual explanation (since the disguised missing data will throw off interpretation). Knowing the phenomenon it can be accounted for during interpretation, and I do not expect the interpretability of models with data to be grossly affected in this case. Predicting on the data set as-is allows me to benchmark my performance against other results in the literature (which in part, provides a measure of validation for the correctness of my implementation).

Second, the Bupa liver data set is classically misused [186] to predict a meaningless target with just 4 of 59 papers using it correctly. A variable that is meant to indicate a training/test data set split is classically thought to be the target of prediction indicating a diseased/non-diseased state. Since the explanation in the machine learning repository's description was a little ambiguous, I corroborated my understanding or assumption with other authors (some of whom I had previously read). Unfortunately their interpretations were wrong.

Used incorrectly the study/problem is not clinical nor very meaningful—although by achieving 70% cross-validated prediction accuracy there is clearly a pattern to be learned in the incorrect target, which is a hand-selected indicator to assign patients to a group for training prediction methods, or a group for testing them.

McDermott *et al.* [186] recommend using a threshold on the variable that is commonly misused, like Turney [271] and Tang et al [257] who both use $q_6 \geq 3$ to predict persons with drinking problems from blood tests of their liver function. McDermott et al prefer $q_6 > 3$, but I choose the former for benchmark purposes. Interestingly, it is worthwhile to report on the difference in results between the correct clinical study/problem and the classically incorrect study/problem, because that difference is consistent with other observations.

Another lesson learned after conducting a number of experiments, courtesy of a clinician, is that a number of the data sets I use have a high prevalence (or incidence) of events, which is not representative of data clinicians may experience in practice. In one case, the skin cancer data from Dr. Ehrsam, consists of skin lesion data from patients referred to a dermatologist—hence it is the referred population (with the greatest prevalance), not the population visiting a general practitioner, nor the general population at large (with the least prevalence). As a result, the absolute values in results may not have specific meaning for clinical purposes, but two aspects offset that: (1) my purpose is not to arrive at clinical

Table 5.1: My experiments use up to eight data sets below, described by name, number of instances N and class ratio $N^+ : N^-$, number of features before n and after conversion n_c, missing data, public availability of data, percentage of binormal (%n BN) features, the presence of image pixel or coordinate data (img), and the number of features in each data type—reals, dates, integers (int), ordinals (ord), binary (bin), nominals (nom) and presence only binary data (+bin). No integer features in my data are coarse/sparse: i.e., 10 or less possible values.

Dataset							Number of features per data type						
name	N $N^+ : N^-$	n (n_c)	miss -ing	pub -lic	%n BN	img	real	fine int	crs int	date	ord	bin	nom, +bin
Skin cancer (B5)	60 20:40	10 (100)	y	y/n		n	1	1	0	0	0	6	1_6, 1_{86}
Heart (statlog)	270 120:150	13 (20)	n	y	60%	n	1	4	1	0	1	3	$1_4 + 2_3$, 0
Hepatitis	155 32:123	18	y	y	12%	n	2	3	0	0	0	13	0, 0
Bupa liver (classic)	345 200:145	6	n	y	10%	n	1	5	0	0	0	0	0, 0
Bupa liver (corrected)	345 176:169	**5**	n	y		n	1	**4**	0	0	0	0	0, 0
Pima Indian diabetes	768 268:500	8	**y**	y	0%	n	8	0	0	0	0	0	0, 0
Colon tumor	62 40:22	2000	n	y	83%	n	2000	0	0	0	0	0	0, 0
Prostate cancer	102 52:50	12600	n	y	35%	n	12600	0	0	0	0	0	0, 0

findings but instead to test the relative (not absolute) performance of kernels and (2) I test with multiple data sets with a range of class imbalance from nearly balanced to highly imbalanced (a 1:4 ratio of events to negatives) and my results are consistent across that range, with the exception of the one "non-clinical" problem per the classically misused Bupa liver data set.

5.1.1 Skin cancer data set

The skin cancer data set [73] is extracted from the website of Dr. Eric Ehrsam, a dermatologist in France who publishes de-identified skin lesion data. The extract has 60 patients from March 26, 2009 to October 16, 2013 and I use the following target of prediction: "malignant melanoma or melanoma *in situ*" versus other diagnoses or conditions.

The extract contains 30 features, however I use only the following 10 relevant features:

Breslow index in mm (real), age (integer), gender (binary), change in lesion size in the last 1 year (binary), lesion present since birth or for the last 10 years (binary), excised or biopsy (binary), differential pressure applied with dermascope (binary), physician (binary), body location (nominal with 6 levels), observed lesion characteristics (86 presence-only binary values). Since these patients have been referred to the dermatologist the prevelance of the condition is much higher than for patients seen by a general practitioner or the prevalence in the general population. The Breslow index in mm has missing data for 49 out of 60 observations, coded as 0. No other features have missing data.

5.1.2 Statlog Heart data set

The Statlog heart data set [66] has 270 patients with the target of prediction being the presence vs. absence of heart disease. It has the following 13 features: age, sex, chest pain type (4 levels), resting blood pressure, serum cholestoral in mg/dl, fasting blood sugar $>$ 120 mg/dl, resting electrocardiographic results (3 levels), maximum heart rate achieved, exercise induced angina, oldpeak = ST depression induced by exercise relative to rest , the slope of the peak exercise ST segment, number of major vessels (0-3), colored by flouroscopy, thal (3 levels: 3 = normal; 6 = fixed defect; 7 = reversable defect). There is no missing data in this data set.

5.1.3 UCI Hepatitis data set

The UCI hepatitis data set [66] has 155 patients with the target of prediction being survival. This data set has 18 features after discarding one feature called protime (43% missing). Five of the features are: bilirubin (real), albumin (real), age (integer), alk phosphate (integer with 19% missing), serum glutamic oxaloacetic transaminase (SGOT) (integer). The remaining 13 features are binary: sex, steroid, antivirals, fatigue, malaise, anorexia, liver big, liver firm, spleen palpable, spiders, ascites, varices and histology. Aside from the missing data already highlighted, nine other features have up to 10% missing data. The missing data were imputed using 1 round of multiple imputation with the monte carlo Markov chain.

5.1.4 UCI Bupa liver data set

The Bupa liver data set [66] has 345 patients with the target of prediction being the number of half-pint equivalents of alcoholic beverages consumed per day, i.e., can we reliably predict

the behaviour from the symptoms? It has one real feature—mean corpuscular volume—and four integer features: alkaline phosphotase (alkphos), alamine aminotransferase (sgpt), aspartate aminotransferase (sgot), gamma-glutamyl transpeptidase (gammagt). See also the prior discussion about this data set in the overview (Section 5.1).

5.1.5 UCI Pima Indians diabetes data set

The UCI Pima Indians diabetes data set [66] has 768 patients with the target of prediction being the presence of diabetes. It has eight real features: number of times pregnant, plasma glucose concentration at 2 hours in an oral glucose tolerance test, diastolic blood pressure (mm Hg), triceps skin fold thickness (mm), 2-hour serum insulin (mu U/ml), body mass index (weight in kg/(height in m)^2), diabetes pedigree function, age (years).

5.1.6 Colon tumor data set

The colon tumor data set [6] has 62 patients with the target of prediction being the presence or absence of cancerous colon tissue. The features are the real valued intensities of 2,000 genes.

5.1.7 Prostate cancer data set

The prostate cancer data set [235] has 102 patients with the target of prediction being the presence or absence of prostate cancer. The features are the real valued intensities of 12,600 genes.

5.2 Experimental method for accuracy

I validate the accuracy of my proposed kernels relative to common and uncommon kernels, with clinical studies/problems and data (and in one case with a non-clinical study/problem) to test my overall hypothesis.

I perform 10 experiments for each classification problem and data set. Each experiment uses 3 stratified cross-validation folds, i.e., sampling with replacement, like bootstrapping,

but also stratified, i.e., selected to have a distribution similar to the entire dataset. Two-thirds of the data are used for training and one-third for testing. This enables comparison with the vast majority of benchmark results, viewed either as a number (N_h) of 3-fold cross-validation experiments or as $3 \cdot N_h$ train/test sets or bootstraps

Hence, I obtained results for 30 different folds or bootstraps, which meets the rule of thumb to have at least 25-30 results [125] for a normal distribution of those results. This ensures that the mean, standard deviation, prediction intervals or other statistics calculated on those results are meaningful to detect statistically significant differences, e.g., between the mean accuracy of SVM with one kernel versus another kernel.

I note that an ideal test method, if there are enough instances in data, is to use a combination of cross-validation (or bootstrapping) with a separate hold-out test set—however, I did not use this approach, as it does not permit benchmarking since it was only used by 2 out of 61 authors/papers with benchmark results.

Table 5.2: Some of the hyperparameter ranges used in hyperparameter search, as an illustration rather than exhaustive listing. The lower and upper limits delimit the uniform distribution used for random sampling. The limits, initially derived from literature [18, 167] are extended based on experience. I denote $\varepsilon = 10^{-15}$ as a small value approaching zero from the positive side.

	Kernel (note: $\varepsilon = 10^{-15}$)						SVM
	Poly	RBF	Sig		MSig		
Limit	d	$\log_{10}\sigma$	a	r	b	d	$\log_{10} C$
Lower	2	-4	ε	-5	-3	-2	-3
Upper	5	+6	20	-ε	+6	+2	+6

The data sets are centered and normalized such that $\pm 3\sigma$, i.e., the third standard deviation, becomes ± 1, following guidance in the literature [18]. Hyperparameter values are searched or generated as random variables [20] with a uniform distribution (e.g., Table 5.2), called random search, which is better [20] than grid search as a common method. Newer hyperparameter search methods now include Bayesian search optimization and particle swarm optimization (PSO), however I deem random search to be sufficient[1].

For each fold I test with 60 sets of hyperparameters, i.e., 60 points in the multidimensional hyperparameter space. Any number of points will never seem like enough given the curse of dimensionality, unless the number is increased by orders of magnitude.

[1]With PSO, Yamada *et al.* [295] (Section A.9) shows that my proposed Mercer sigmoid kernel, classifies well relative to the Gaussian RBF kernel on data sets with atomic data types, similar to my results without PSO.

I perform tests with SVM using four common kernels, four uncommon kernels and eight kernels which I proposed.

I test the following common kernels: the linear (Lin), polynomial (Poly), Gaussian RBF (RBF) and sigmoid (Sig) kernels. I test the following uncommon kernels: the normalized sigmoid (SigN), power distance (Pwr), logarithmic (Log) and inverse multiquadric (IMQ) kernels. I test the following proposed kernels: the Mercer sigmoid (MSig), Gaussian derivative (GD), orthant linear (OLin), Bayesian binormal (BBN) and joint probability binormal (JPBN) kernel—and three versions of an insensitive sigmoid variant (ISV) kernel, denoted ISVgc, OISVgc and OISVhc[2].

I use Matlab's standard implementations of SVM to calculate class-specific soft-margin parameters C_+ and C_- from C to achieve a balanced success rate with imbalanced data [18] and I use built-in implementations of three common kernels: the linear, polynomial and Gaussian RBF kernels. In earlier versions of Matlab the sigmoid kernel was also built-in, but is no longer (presumably because it is not an admissible kernel and therefore could create liability). I use sequential minimal optimization (SMO) for SVM in Matlab as it is faster and just as reliable and with the same guarantees as the classical quadratic programming (QP). I use the default optimization solver for SMO.

I obtain results of accuracy and interpretability (Table 5.3) and report on key measures (see next section), while using other measures for analysis and discussion where informative.

[2]"O" refers to orthant, "g" refers to a Gaussian derivative, "h" refers to a hyperbolic tangent and "c" refers to contraints.

Table 5.3: Accuracy (a) and interpretability (U) types of measures

Scope	Measure type (a or U), names and abbreviations	
For a specific model and specific TP/FP trade-off, e.g., $\text{SVM}_{C=1}, \text{RBF}_{\sigma=1}$	a	Accuracy (a), sensitivity/recall (sens), specificity (spec), positive predictive value/precision (ppv), F_1 measure, negative predictive value (npv), balanced accuracy (ba), balanced predictive value (bpv)
	U	Number of support vectors (sv), SVM cost of error (C), kernel width (e.g., σ), percent folds converged (cnv), elapsed time (et), relevant dimension estimate (rde), rde noise estimate (noise), model interpretability ($\tilde{U}_\partial$, U_{sv}), Gram matrix rank, Gram matrix condition and sparsity, model interpretability for limited use (U^*_{rdeT}, U^*_{rdeL}), simplicity of sensitivity with limited use ($U_{H\text{st}}$)
For a family of models at all trade-offs for a set $\{C\}$, $\{\sigma\}$ or $\{C, \sigma\}$, re $\text{SVM}_{\{C\}}, \text{RBF}_{\{\sigma\}}$	a	Area under the curve (auc), average precision (ap, ppv) area under the precision recall curve (auprc)
	U	Number of training instances (N), number of features (n), model transparency (U_∂)

5.3 Experimental results for accuracy

In this section I report my classification results using key accuracy-related measures. In the next section, I include a measure of model interpretability and discuss model selection based on accuracy and interpretability using that measure.

An overview of results is provided by averaging the measures for each kernel and ranking those results for each dataset (Table 5.4). While standardizing each measure within each data set for ranking is the most robust approach, my approach is simple and effective—with only average precision (ap) having greater variance than other measures.

From the ranks (Table 5.4) I **observe** that:

1. Three of my proposed kernels, based on an insensitive sigmoid (OISVgc, OISVhc, ISVgc) and a fourth kernel I propose, called the Gaussian derivative (GD) kernel, rank better on accuracy-related measures than the common kernels, and rank better than the median for all clinical data sets, i.e., a rank less than 8.5.

   (a) These four proposed kernels achieve better accuracy than all of the common kernels. They are also more transparent.

2. The normalized sigmoid kernel [45], an uncommon kernel, has the most top 5 kernel rankings, ***but it is not an admissible kernel*** [45, 167] and therefore not suitable

Table 5.4: Kernels (split across two tables) ranked by an average of accuracy-related measures in classification for five clinical and one non-clinical (nc) data sets. The top 5 ranks are highlighted with a green cell colour.

Data set	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
Hepatitis	8	17	14	15	7	18	16	11
Heart	16	13	11	18	17	10	12	9
Liver	9	11	14	13	8	15	12	10
Skin	9	13	12	11	5	14	10	8
Diabetes	12	10	14	17	6	8	1	3
Colon	10	17	7	15	2	18	16	5
Liver(nc)	15	5	3	13	4	6	2	1

Data set	Proposed kernels							
	MSig	OSig	OISVgc (OISVhc)	ISVgc	BBN (JPBN)	OMBcs	OLin	GD
Hepatitis	9	13	4 2	1	5 3	12	6	10
Heart	7	6	4 3	2	8 1	14	15	5
Liver	5	4	1 1	1	16 18	16	7	6
Skin	6	5	1 3	1	18 17	4	10	7
Diabetes	11	9	5 4	2	16 18	15	13	7
Colon	9	4	3 1	6	13 14	12	8	11
Liver(nc)	11	10	8 7	9	17 16	17	14	12

for health care or other high reliability applications [45]. The sigmoid kernel is also not admissible [45, 167].

   (a) The Mercer sigmoid kernel, proposed as a substitute for both of them, ranks worse than the (uncommon) normalized sigmoid but better than the sigmoid, and ranks better than the median rank on most clinical data sets—hence it is a reasonable substitute.

3. The power distance kernel and log kernels, both uncommon kernels, rank better than the Gaussian RBF kernel on most clinical data sets.

   (a) The predominant use of the Gaussian RBF kernel, or considering only common kernels, falls short of achieving the best accuracy with clinical data.

4. The Gaussian RBF and polynomial kernels rank noticeably well/better on the ***classic misused*** (Bupa) ***liver*** study/problem and data. I consider it a ***non-clinical*** (nc) study/problem because the prediction target is not clinical—it is the researchers' attempt to randomly pick two classes by hand which involves more randomness, subtlety and complexity in patterns than a clinical study/problem.

   (a) For this non-clinical study/problem, the Gaussian RBF and polynomial kernels with feature interactions of second order or more, are more accurate than the proposed transparent kernels which are limited in feature space complexity/dimensionality when the number of features $n_c < N$ are smaller than the number of instances.

5. Based on the match between kernels and requirements of specific data types, I find that performance with kernels that match the data types, is better than or equal to kernels that do not match them. I include proviso "or equal to" since there are many factors at play such as the significance of a feature and the number of kernel hyperparameters to search.

6. For **converted nominals** and **presence-only binary data** found in the heart and skin cancer data sets (the only data sets with those data types):

   - The orthant sigmoid (OSig) kernel and related variants (OISVgc and OISVhc) designed for nominals are better than or equal to the Mercer sigmoid (MSig) kernel in accuracy, as expected—where the MSig kernel is a good baseline since it is a specific case of the other three kernels.
     - Note: From the formula and shape of the MSig kernel, I know that it may act like an "orthant" kernel, if it is shifted and scaled so that the data only

use of half of it—the positive matching "hill" with similar shape—while the other half, the negative matching "hill" is outside of the data's range.

- The orthant sigmoid (OSig) kernel and related variants (OISVgc and OISVhc) designed for nominals are better than or equal to the Gaussian RBF kernel in accuracy, as expected. The RBF kernel is a reasonable baseline as the most common kernel—and it is not unfair (more accurate) than the Mercer sigmoid (MSig) kernel as a baseline.
  - Note: The Gaussian RBF kernel matches a single nominal, but with more features (nominals or otherwise) its match is not known.
- The orthant linear (OLin) kernel designed for nominals is better than or equal to the linear (Lin) kernel in accuracy, as expected.

I **conclude** the following:

- My kernels which are designed to match data types for justification, accuracy and interpretability, with a transparent form, achieve the same or better accuracy as sought in my hypothesis.
  - However, kernels which are not intentionally designed to match certain data types, may fully or partially match them out of coincidence. For example, the Mercer sigmoid kernel may act like an orthant sigmoid kernel, if it is shifted and scaled so that the data only use of half of it—the positive matching "hill" with similar shape—while the other half, the negative matching "hill" is outside the data's range. Another example is the power density kernel with $\beta = 2$ whose formula fully matches converted nominals and presence-only binary data without specific intent in its design.

Table 5.5: Accuracy-related measures of prediction performance (split across two tables) for **hepatitis** data in thirty stratified folds in ten SVM experiments using 60 random hyperparameters. The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
a	85.0±0.01	81.8±0.01	83.8±0.01	83.0±0.01	85.3±0.01	81.7±0.01	83.2±0.01	84.7±0.01
bal a	75.3±0.02	64.0±0.03	72.3±0.02	71.0±0.02	74.8±0.02	56.3±0.02	66.4±0.02	72.7±0.02
AUC	77.3±0.01	68.2±0.02	74.5±0.02	73.4±0.02	77.1±0.02	64.0±0.01	70.3±0.01	75.1±0.01
AP	42.7±0.06	39.3±0.05	45.7±0.03	39.1±0.04	44.9±0.04	43.6±0.03	46.8±0.03	48.6±0.03
F_1	62.6±0.02	50.0±0.03	59.2±0.03	56.9±0.02	62.2±0.02	43.1±0.03	53.8±0.02	60.2±0.02
avg	68.6±0.01	60.7±0.02	67.1±0.02	64.7±0.02	68.9±0.02	57.7±0.02	64.1±0.02	68.3±0.02
rank	8	17	14	15	7	18	16	11

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	85.0±0.01	85.2±0.01	85.5±0.01 85.5±0.01	86.0±0.01	85.0±0.01 84.1±0.01	84.8±0.01	85.2±0.01	84.8±0.01
bal a	74.8±0.01	75.4±0.01	75.7±0.02 75.7±0.02	76.4±0.02	74.3±0.02 75.1±0.01	74.1±0.01	75.6±0.01	73.5±0.02
AUC	76.9±0.01	77.3±0.01	77.7±0.01 77.8±0.01	78.2±0.02	76.5±0.02 76.8±0.01	76.4±0.01	75.6±0.01	75.6±0.01
AP	42.7±0.06	39.7±0.05	43.5±0.06 44.4±0.06	45.4±0.05	47.8±0.05 48.3±0.05	43.9±0.02	43.8±0.05	46.8±0.03
F_1	62.1±0.01	62.9±0.02	63.2±0.02 63.1±0.02	64.9±0.02	61.8±0.03 61.9±0.02	61.6±0.01	62.7±0.02	61.1±0.02
avg	68.6±0.01	68.1±0.02	69.1±0.02 **69.3±0.02**	**70.2±0.02**	66.3±0.01 69.2±0.02	68.2±0.01	69.0±0.02	68.4±0.01
rank	9	13	4 2	1	5 3	12	6	10

Table 5.6: Accuracy-related measures of prediction performance (split across two tables) for **heart** data in thirty stratified folds in ten SVM experiments using 60 random hyper-parameters. The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
a	84.8±0.01	82.7±0.01	83.9±0.01	79.3±0.02	84.3±0.01	82.1±0.00	82.9±0.00	84.1±0.01
bal a	84.0±0.01	81.4±0.01	83.0±0.01	78.9±0.01	83.7±0.01	80.9±0.00	82.0±0.01	83.2±0.01
AUC	85.7±0.01	82.9±0.00	84.5±0.01	80.8±0.02	85.9±0.01	83.4±0.01	84.2±0.00	85.2±0.01
AP	53.0±0.09	66.3±0.01	64.4±0.08	56.6±0.07	53.3±0.14	72.4±0.13	67.0±0.13	67.3±0.10
F_1	82.2±0.01	79.2±0.01	81.0±0.01	77.0±0.01	81.8±0.01	78.5±0.01	79.9±0.01	81.2±0.01
avg	78.0±0.02	78.5±0.01	79.4±0.0	74.5±0.0	77.8±0.0	79.5±0.0	79.2±0.03	80.2±0.02
rank	16	13	11	18	17	10	12	9

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	85.1±0.01	85.1±0.01	86.4±0.01 86.1±0.00	85.6±0.01	84.8±0.01 85.7±0.01	84.1±0.01	84.7±0.01	84.9±0.01
bal a	84.2±0.01	84.2±0.01	85.4±0.01 85.1±0.00	84.8±0.01	83.9±0.01 84.9±0.01	83.2±0.01	83.9±0.01	84.2±0.01
AUC	85.6±0.01	85.4±0.01	86.5±0.00 86.4±0.00	86.4±0.00	85.3±0.01 86.0±0.01	84.6±0.01	85.8±0.01	85.4±0.01
AP	70.4±0.06	71.4±0.03	72.6±0.04 73.9±0.02	77.9±0.02	68.8±0.15 79.6±0.02	58.3±0.07	54.3±0.14	74.1±0.02
F_1	82.4±0.01	82.3±0.01	83.8±0.01 83.4±0.01	83.0±0.01	82.0±0.01 83.2±0.01	81.2±0.01	82.1±0.01	82.3±0.01
avg	81.5±0.02	81.7±0.01	82.9±0.01 83.0±0.01	**83.5±0.01**	80.9±0.03 **83.9±0.01**	78.3±0.02	78.2±0.03	82.2±0.01
rank	7	6	4 3	2	8 1	14	15	5

Table 5.7: Accuracy-related measures of prediction performance (split across two tables) for **correct liver** data in thirty stratified folds in ten SVM experiments using 60 random hyperparameters. The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
a	64.1±0.01	63.5±0.01	63.8±0.00	62.6±0.01	63.9±0.01	63.6±0.01	63.7±0.01	64.0±0.01
bal a	62.3±0.01	61.4±0.01	62.0±0.01	61.2±0.01	62.5±0.01	62.5±0.01	62.5±0.01	62.5±0.01
AUC	65.5±0.01	64.9±0.01	64.9±0.01	64.6±0.01	65.7±0.01	64.5±0.01	64.7±0.01	65.3±0.01
AP	59.7±0.01	60.8±0.02	56.2±0.01	59.6±0.01	60.1±0.01	56.1±0.01	57.5±0.01	58.5±0.01
F_1	67.4±0.00	66.5±0.02	67.4±0.00	67.3±0.00	67.4±0.00	67.4±0.00	67.3±0.00	67.3±0.00
avg	63.8±0.0	63.4±0.0	62.9±0.0	63.0±0.0	63.9±0.0	62.8±0.0	62.8±0.0	63.1±0.0
rank	9	11	14	13	8	15	12	10

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	63.8±0.01	63.9±0.0	64.7±0.00 64.8±0.01	64.6±0.00	61.5±0.01 60.8±0.01	61.5±0.01	64.2±0.01	63.9±0.01
bal a	62.6±0.01	62.9±0.0	64.0±0.01 63.9±0.00	64.0±0.00	58.5±0.01 59.8±0.01	58.5±0.01	62.1±0.01	62.8±0.01
AUC	66.8±0.01	66.9±0.01	67.5±0.01 67.6±0.01	67.5±0.00	62.6±0.01 61.9±0.01	62.6±0.01	65.7±0.01	66.1±0.01
AP	62.6±0.01	63.0±0.01	64.3±0.01 64.3±0.01	64.3±0.01	59.9±0.01 57.0±0.01	59.9±0.01	60.3±0.01	60.4±0.02
F_1	67.4±0.00	67.4±0.00	67.4±0.00 67.4±0.00	67.4±0.00	67.3±0.00 67.4±0.00	67.3±0.00	67.5±0.00	67.5±0.00
avg	64.6±0.01	64.8±0.01	**65.6±0.00** **65.6±0.00**	**65.6±0.04**	62.0±0.01 61.4±0.01	62.0±0.01	64.0±0.00	64.2±0.00
rank	5	4	1 1	1	16 18	16	7	6

Table 5.8: Accuracy-related measures of prediction performance (split across two tables) for **skin cancer** data in thirty stratified folds in ten SVM experiments using 60 random hyperparameters. The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
a	84.5±0.01	81.0±0.01	84.5±0.01	80.2±0.02	84.3±0.01	78.8±0.01	82.2±0.01	84.2±0.02
bal a	79.0±0.02	72.1±0.02	78.6±0.02	76.8±0.03	78.2±0.02	66.4±0.02	73.9±0.01	78.4±0.02
AUC	80.5±0.02	74.7±0.02	80.0±0.02	79.2±0.03	79.8±0.02	70.8±0.01	76.3±0.01	80.1±0.02
AP	60.8±0.04	60.7±0.03	55.9±0.07	54.5±0.04	59.6±0.06	61.5±0.02	64.4±0.03	65.8±0.03
F_1	74.1±0.03	66.0±0.03	73.9±0.02	69.6±0.03	69.6±0.03	59.4±0.02	68.4±0.02	73.4±0.03
Avg	75.8±0.02	70.9±0.02	74.6±0.02	72.0±0.02	75.1±0.02	67.4±0.01	73.0±0.02	76.4±0.02
Rank	9	13	12	11	5	14	10	8

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	88.7±0.02	89.0±0.02	91.5±0.01 91.2±0.02	91.0±0.01	73.7±0.01 79.5±0.03	92.0±0.01	84.7±0.01	88.0±0.01
bal a	79.0±0.02	84.8±0.02	86.9±0.02 86.8±0.02	86.2±0.02	49.0±0.04 69.0±0.04	87.4±0.01	78.9±0.02	82.1±0.02
AUC	80.5±0.02	86.1±0.02	88.0±0.02 88.2±0.02	87.2±0.02	61.6±0.02 76.8±0.03	88.4±0.01	80.3±0.02	84.0±0.02
AP	54.1±0.09	56.1±0.09	53.4±0.06 53.3±0.07	56.5±0.09	57.1±0.06 44.8±0.04	47.1±0.01	60.4±0.05	57.4±0.08
F_1	81.2±0.03	82.0±0.03	85.5±0.03 85.2±0.03	84.7±0.02	38.2±0.05 63.3±0.04	86.4±0.01	74.2±0.02	79.3±0.03
avg	64.6±0.01	64.8±0.01	**81.1±0.02** 80.9±0.02	**81.1±0.03**	55.9±0.03 66.7±0.03	80.2±0.02	75.7±0.02	78.2±0.02
rank	6	5	1 3	1	18 17	4	10	7

Table 5.9: Accuracy-related measures of prediction performance (split across two tables) for **diabetes** data in thirty stratified folds in ten SVM experiments using 60 random hyperparameters. The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
a	77.9±0.01	77.2±0.00	78.1±0.00	76.6±0.01	77.9±0.00	77.5±0.00	78.1±0.01	78.1±0.01
bal a	71.3±0.00	69.7±0.00	71.1±0.00	68.1±0.01	71.2±0.01	71.9±0.01	71.9±0.01	72.0±0.01
AUC	73.8±0.00	72.8±0.00	73.3±0.00	71.2±0.00	74.0±0.01	74.6±0.01	74.4±0.01	74.6±0.01
AP	57.5±0.00	60.8±0.03	52.3±0.08	44.5±0.08	65.2±0.01	50.5±0.16	69.5±0.01	60.2±0.06
F_1	64.2±0.00	62.3±0.00	64.1±0.00	60.2±0.00	63.8±0.00	64.3±0.02	64.7±0.01	65.0±0.01
avg	68.3±0.01	69.3±0.01	67.4±0.01	65.6±0.02	69.7±0.00	69.4±0.01	71.0±0.01	70.6±0.01
rank	12	10	14	17	6	8	1	3

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	85.1±0.0	85.1±0.0	86.4±0.0 86.1±0.0	85.6±0.0	84.8±0.0 85.7±0.0	84.1±0.0	84.7±0.0	84.9±0.0
bal a	84.2±0.0	84.2±0.0	85.4±0.0 85.1±0.0	84.8±0.0	83.9±0.0 84.9±0.0	83.2±0.0	83.9±0.0	84.2±0.0
AUC	85.6±0.0	85.3±0.0	86.5±0.0 86.4±0.0	86.4±0.0	85.3±0.0 86.0±0.0	84.6±0.0	85.8±0.0	85.4±0.0
AP	70.4±0.0	71.4±0.0	72.6±0.0 73.9±0.0	77.9±0.0	68.8±0.0 79.6±0.0	58.3±0.0	54.3±0.0	74.1±0.0
F_1	82.4±0.0	82.3±0.0	83.8±0.0 83.4±0.0	83.0±0.0	82.0±0.0 83.2±0.0	81.2±0.0	82.1±0.0	82.3±0.0
avg	68.9±0.00	69.4±0.00	70.3±0.00 70.3±0.00	**70.7±0.00**	66.0±0.01 65.0±0.02	66.0±0.01	68.2±0.01	69.5±0.01
rank	11	9	5 4	2	16 18	15	13	7

Table 5.10: Accuracy-related measures of prediction performance (split across two tables) for **colon cancer** data in thirty stratified folds in ten SVM experiments using 60 random hyperparameters. The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green. Something anomalous occurs with the inverse multiquadric (IMQ) kernel on this data set.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Logx	Pwr
a	85.7±0.02	63.8±0.02	85.5±0.01	75.6±0.03	85.7±0.01	64.5±0.00	76.2±0.02	85.5±0.01
bal a	83.6±0.02	58.7±0.04	84.0±0.02	69.5±0.04	83.9±0.02	00.0±0.0	61.9±0.05	83.8±0.02
AUC	84.4±0.02	61.3±0.03	84.8±0.02	72.5±0.03	84.9±0.01	50.0±0.00	68.5±0.03	85.0±0.02
AP	75.7±0.01	65.8±0.01	77.0±0.02	72.2±0.02	77.8±0.02	64.5±0.00	69.4±0.01	77.1±0.02
F_1	89.0±0.01	75.1±0.02	88.7±0.01	82.7±0.02	89.2±0.01	78.4±0.00	83.9±0.01	89.0±0.01
avg	83.7±0.01	65.0±0.02	84.0±0.01	74.5±0.02	84.3±0.01	51.5±0.00	72.0±0.03	84.1±0.01
rank	9	17	6	15	3	18	16	5

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	84.9±0.01	85.7±0.01	86.2±0.01 86.7±0.02	86.0±0.01	82.1±0.03 80.6±0.03	82.1±0.03	86.0±0.01	84.9±0.01
bal a	83.0±0.02	83.7±0.01	84.8±0.02 85.4±0.02	84.1±0.02	79.6±0.03 77.7±0.03	79.6±0.03	84.3±0.02	82.9±0.02
AUC	84.2±0.02	85.0±0.02	85.7±0.02 86.2±0.02	84.8±0.02	80.6±0.03 78.3±0.03	80.6±0.03	84.8±0.02	83.5±0.01
AP	76.6±0.02	76.4±0.01	76.3±0.01 76.8±0.01	75.2±0.01	74.7±0.02 75.7±0.03	74.7±0.02	76.0±0.01	71.4±0.02
F_1	88.5±0.01	89.1±0.01	89.4±0.01 89.8±0.01	89.3±0.01	86.5±0.02 85.3±0.02	86.5±0.02	89.2±0.01	88.4±0.01
avg	83.5±0.01	84.0±0.01	**84.5±0.01** **85.0±0.01**	83.9±0.01	80.7±0.02 79.5±0.03	80.7±0.02	84.1±0.01	82.2±0.01
rank	10	6	2 1	8	13 14	12	4	11

Table 5.11: Accuracy-related measures of prediction performance (split across two tables) for **classic misused liver** data in fifteen stratified folds for five SVM experiments using 60 random hyperparameters (or six folds in two experiments for some kernels*). The top 2 results are highlighted in bold and the top five kernels (per the average over all measures) are highlighted in green.

	Common kernels				Uncommon kernels			
	Lin	Poly	RBF	Sig	SigN	IMQ	Log	Pwr
a	69.4±0.01	72.8±0.01	72.6±0.01	68.1±0.01	73.1±0.00	72.1±0.01	72.8±0.01	73.4±0.01
bal a	65.8±0.01	71.0±0.01	70.2±0.01	66.1±0.01	70.6±0.01	70.0±0.01	70.6±0.01	71.1±0.01
AUC	67.7±0.01	72.6±0.01	71.8±0.01	68.2±0.01	72.2±0.01	72.0±0.01	72.4±0.01	72.8±0.01
AP	55.6±0.02	57.9±0.01	62.2±0.02	56.9±0.02	59.4±0.03	60.9±0.04	61.9±0.04	63.5±0.02
F_1	59.6±0.01	66.4±0.01	65.1±0.01	60.8±0.02	65.5±0.01	64.9±0.01	65.6±0.01	66.2±0.01
avg	63.6±0.01	68.1±0.01	68.4±0.01	64.0±0.01	68.2±0.01	68.0±0.01	**68.7±0.01**	**69.4±0.01**
rank	15	5	3	13	4	6	2	1

	Proposed kernels							
	MSig	OSig	OISVgc OISVhc	ISVgc	BBN JPBN	OMBcs	OLin	GD
a	72.0±0.01	71.9±0.01	73.2±0.01 73.2±0.01	73.1±0.01	59.3±0.01 63.9±0.01	59.3±0.01	69.4±0.01	70.3±0.01
bal a	68.4±0.01	68.3±0.01	69.5±0.01 69.6±0.01	69.4±0.01	37.4±0.04 58.6±0.02	37.4±0.04	65.4±0.01	66.5±0.01
AUC	70.1±0.01	70.2±0.01	71.7±0.01 71.8±0.00	71.5±0.01	52.9±0.01 61.3±0.02	52.9±0.01	67.6±0.01	68.3±0.01
AP	58.2±0.02	58.9±0.02	60.9±0.01 60.6±0.01	60.0±0.01	46.5±0.02 49.8±0.02	46.5±0.02	57.8±0.03	57.1±0.02
F_1	62.6±0.01	62.5±0.01	64.1±0.01 64.2±0.01	63.9±0.01	24.4±0.04 50.6±0.02	24.4±0.04	59.0±0.01	60.3±0.02
avg	66.3±0.01	66.3±0.01	67.9±0.01 67.6±0.01	67.6±0.01	44.1±0.02 56.8±0.02	44.1±0.02	63.8±0.01	64.5±0.01
rank	11	10	8 7	9	17 16	17	14	12

5.4 Experimental method to validate theory

I validate my theoretic derivations, designs and matching regarding **asymmetric match weighting** for **converted nominals** and **presence-only binary** data, by testing a series of two data sets with exclusion to increasing inclusion of those data types. These same experiments also serve to validate my claims regarding kernels suited to heterogeneous data.

I also validate my theoretic derivations, designs and matching regarding kernels for **binary** data, by testing two data sets with binary data types included and excluded. These same experiments also serve to validate my claims regarding kernels suited to heterogeneous data.

5.5 Experimental results to validate theory

For **converted nominals** and **presence-only binary data** found in the heart and skin cancer data sets (the only data sets with those data types), I tested the effect of the orthant kernels on classification accuracy when I omit these data types and then include them incrementally.

1. The idea is to isolate the effect to only the difference in the orthant hyperparameter/design of a kernel versus an otherwise same or similar kernel. Confounding factors in this experiment are varying significance of features (e.g., introducing a relatively insignificant feature may have no effect), random differences in data sets and other hyperparameters of kernels if not fixed. I fixed most, but not all other hyperparameters.

2. The result is that in five out of six tests there were no statistically significant differences to prove nor disprove my hypothesis that the orthant hyperparameter/design causes better performance for OSig versus MSig, for OISVgc versus ISVgc and for OLin versus Lin.

3. In one test however, ISVgc versus OISVgc on Heart data, the former is better in that the means do not fall within each other's confidence intervals, however a paired t-test is not performed.

5.6 Innate views and data from SVM

SVM is **semi-transparent** because it innately offers **interpretable views** of the **input feature space** (Figure 2.3 on page 18 at left) relevant to the input and output, **but not of the feature space** (Figure 2.3 on page 18 at right) **except in the case of transparent kernels** which are not innate.

SVM also innately offers **interpretable data about the influence of support vectors (training instances)** in the model—i.e., the **global** influence (or significance) of a support vector per its model weight α_i and the **local** influence (or significance) of a support vector α_i on instance $\underline{v}_j$ per the similarity-weighted model weight, $\alpha_i \cdot k\left(\underline{x}_i, \underline{v}_j\right)$. This is expected since SVM is an instance-based method.

For kernels which are not transparent, features are confounded by the model and the feature space is not visible. Features are also often correlated or confounded in real-life, apart from the model's behaviour.

In the input space, I **can visualize instances** and the **class boundary** and I **can observe/test how specific decisions change** with changes in input, by interrogating the model as a black box or oracle, and I can visualize partial derivatives and/or Taylor series derivatives. However partial derivatives are of **limited local use** when inputs are correlated/confounded in real-life and by the model.

In the big picture, from the input space, I **cannot fully understand or summarize** how the model **makes or changes decisions *in general*** with different inputs or changes in input (Section 5.6.1). **Static plots are prone to misinterpretation** and instead multiple plots are needed for understanding.

To reiterate Feynman: if I understand a concept I must be able to describe it at a freshman level, which often requires simplification or reduction (i.e., summarization), otherwise I don't really understand it [101]. I want understanding so that I can **fully predict outputs** and **fully characterize the model**—to analyze, change and improve, the model itself, or its use/applicability for subgroups seen or not yet seen. I want **transparent views** (Section 5.7) of the feature space, arising from **transparent kernels**, so that I can **fully view/visualize**, **understand**, predict and characterize distance, closest points, the class boundary as a hyperplane and classification decisions (Section 5.6.1). To understand a point of interest I want understanding from one or a few plots, not many plots nor a plot for each conjecture about the point of interest.

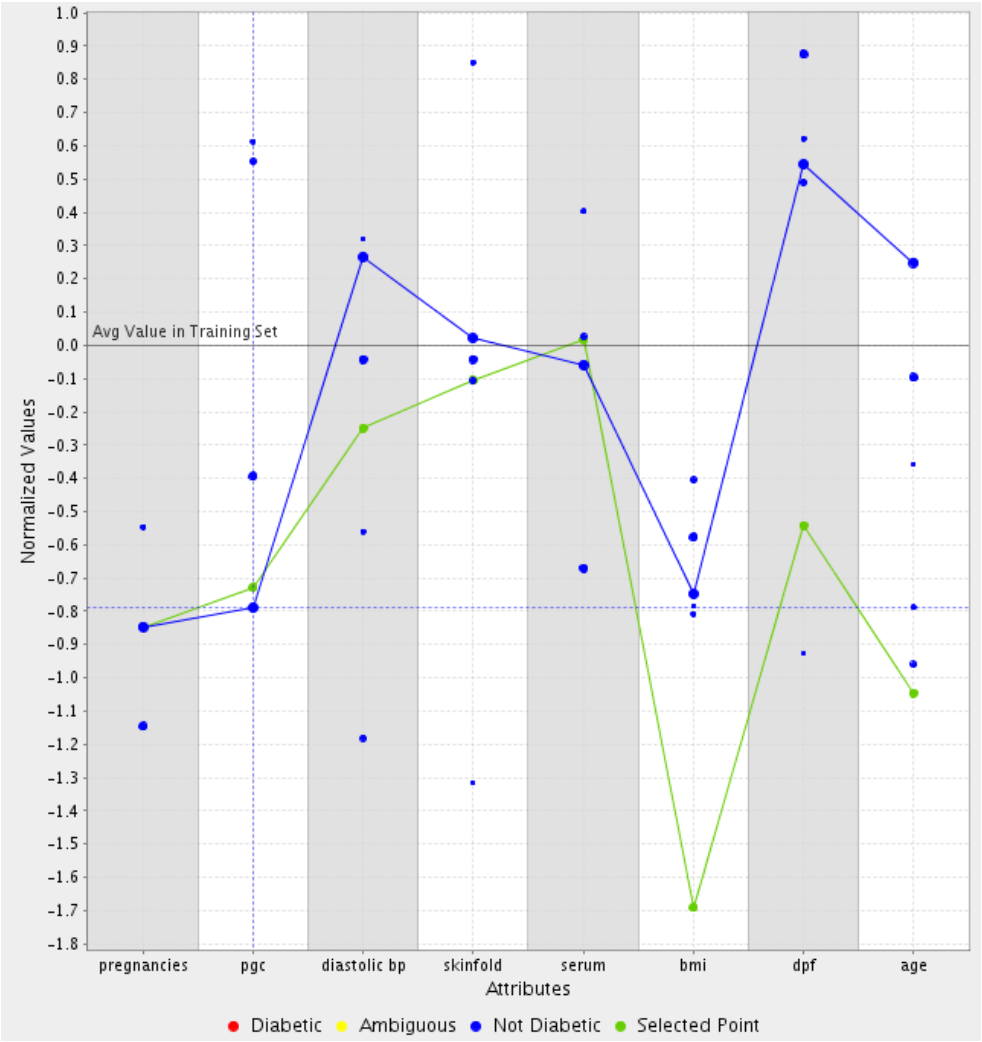

Figure 5.1: This figure from Barbella *et al.* [15] shows an innate view of SVM classification results with a Gaussian RBF kernel with the Pima Indians diabetes data set. In this view, the effect of each column is confounded with others, since the output of the Gaussian RBF kernel confounds the input features.

5.6.1 Innate views of SVM from Barbella *et al.*

Barbella *et al.* [15] provide good contributions to innate views of SVM, albeit with some shortcomings. They focus on an example using the Gaussian RBF kernel as the most common SVM kernel. Their method is an interactive exploratory tool not for automation. Two of their key contributions are as follows, with shortcomings also described (and in the following section my improvements).

1. **They compute and visualize the top 5 most influential instances** (Figure 5.1 on page 130), i.e., support vectors, on the decision for an instance or point of interest. They visualize the multi-dimensional input space as a set of columns for each feature

in an exploratory interactive tool. They colour code the visualized support vectors according to the class they belong to, or as ambiguous if it is in the margin. The visualization is effective but has some shortcomings:

- **The plot and interactive tool are only accurate around the point of interest**, as I explain in subsequent items, except for explicit Mercer kernels, i.e., transparent kernels. To answer questions about what occurs at a different point the user must interactively change the point of interest. Hence, **understanding cannot be derived from a single plot**—it requires a plot for each point of interest.
- **In the plot, it looks like the distances within each feature (column) are independent of each other, but they are not**, in general—their effects on the kernel and classifier output are confounded for any kernel that is not explicit Mercer. For example, with the Gaussian RBF kernel, it only takes one feature to be sufficiently distant, relative to the kernel width, for the overall result to be near zero, while all features must be sufficiently proximal for the overall result to be near one.
- **The plot does not describe the class boundary** [3], i.e., the tipping points, of the model—e.g., it is possible that some (or all) features are not influential enough to change the classification for the point of interest.
- **The plot also does not include the SVM bias**, which, along with class imbalance, may be the reason why no features are influential enough to change the classification for some points of interest.
- **The plot does not indicate how much or how quickly a feature influences changes in classification, nor which features are most influential.** The plot does not show the class boundary nor the decision surface, where the latter is needed because a linear distance in the plot does not correspond to a linear change in kernel output nor classification score.
- **In the plot, a reader might infer that changing a feature's value to that of another plotted point achieves that classification, but it does not, in general.** The classification depends on all of the feature values, in general. Also, the example plot does not show support vectors from both classes.

2. **They find one of the closest points on the class boundary to the point of interest.** Shortcomings are as follows:

[3]Note: the bias or y-intercept for an SVM classifier may increase with increasing class imbalance

- **The objective of finding a point on the class boundary, in the input space, is not sufficiently realistic and stable.** Barbella *et al.* [15] articulate a goal of doing better than inverse classification. Whereas inverse classification finds the closest point in the opposite class, they seek to find a point on the class boundary itself which is closer, on the unstated assumption that a closer point is better. The assumption is not true in two out of three cases, and it depends on the data and model.
  - If a point is on the class boundary or too close to the boundary, it has uncertainty and the classification is unstable with respect to future changes, model error and measurement noise. Even a point found in inverse classification may be too close, or it may be just right, or it may not be close enough—it depends on the data and model.
- **Immutable features and feature cost factors were not considered.** If the purpose is to improve a patient's health in order to change their outcome or prognosis or ability to qualify for certain procedures, then it is notable that the patient's demographics are immutable and that some features are harder to change than other features, hence a better distance measure in the input space is a weighted Euclidean distance.
- **As they recognize, the closest point on the class boundary, in the input space, is not unique, in general**. For example, classification with a Gaussian RBF kernel with a sufficiently small width, creates a class boundary that is spherical in n-dimensions, or partially spherical (e.g., Figure 5.2 on page 133) with an infinite number of class boundary points which are closest in the input space.

  As another example, in classification with any kernel that produces a closed boundary, e.g., stationary kernels, there are a set of points within that boundary which are equidistant to at least two points on the class boundary (e.g., points on the black line in Figure 5.3 on page 133). Open curves, e.g., from a Mercer sigmoid kernel, also have a set of points with at least two closest points on the class boundary.

5.6.2 My improvements to innate views of SVM

I make several additions and improvements (Figure 5.4 on page 134) to the innate view proposed by Barbella *et al.* [15] (Figure 5.1 on page 130) as follows:

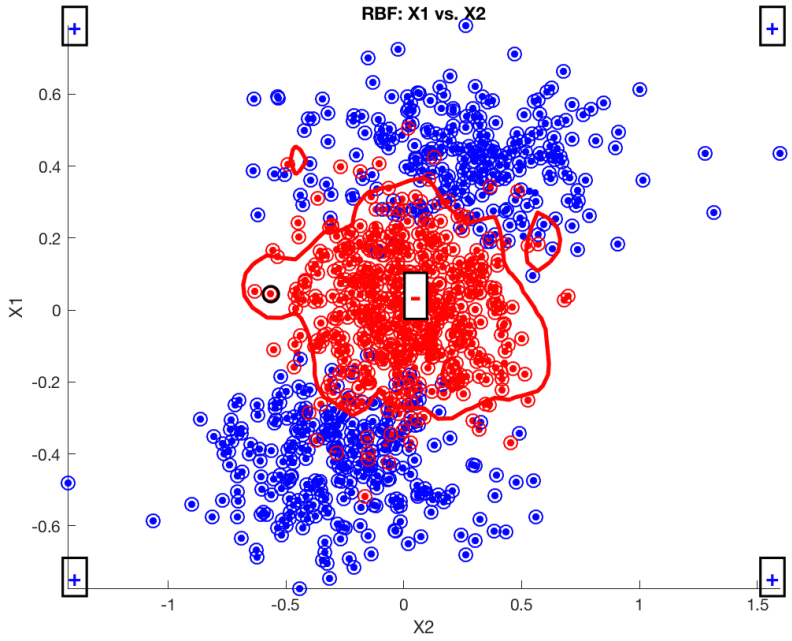

Figure 5.2: This Gaussian RBF kernel has a sufficiently small kernel width such that its spherical behaviour about the point circled in black is evident.

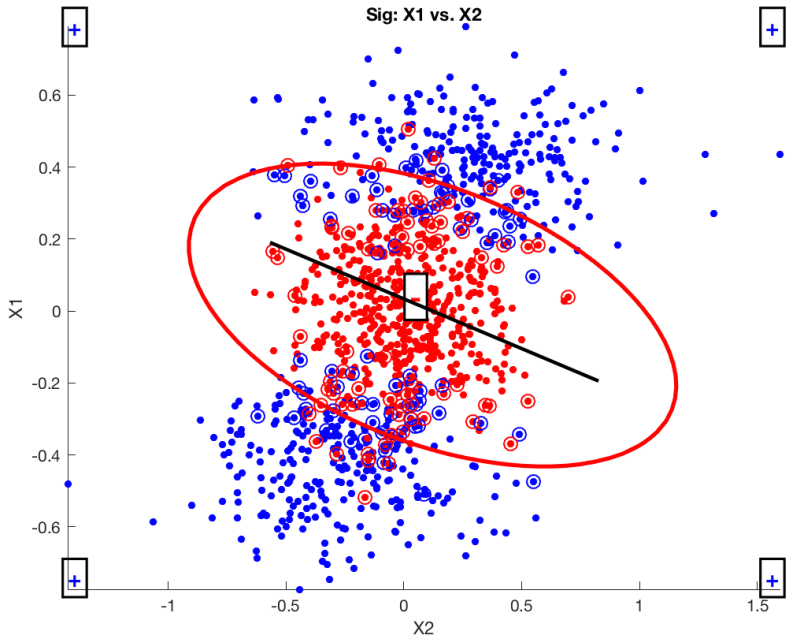

Figure 5.3: This sigmoid kernel has at least two points on the class boundary which are closest to each point on the black line.

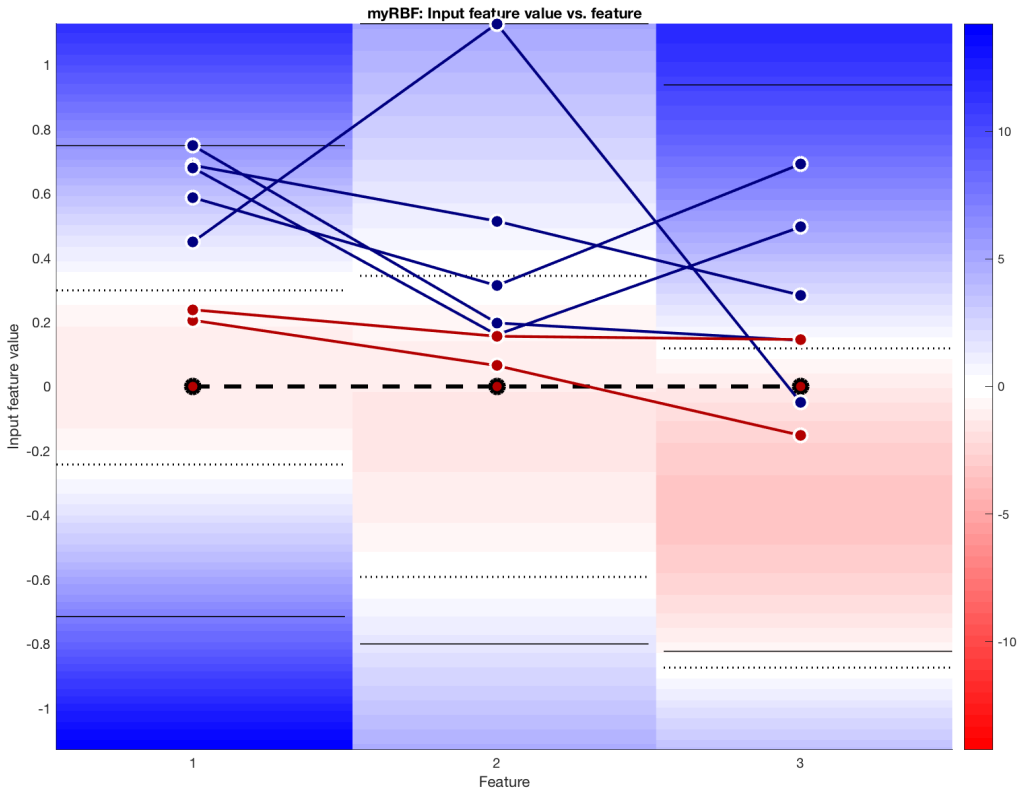

Figure 5.4: My proposed improvement to the innate view from Barbella *et al* [15]. The chosen point of interest (POI) is plotted as a dashed black line, while the support vectors (instances) which contribute most to the POI's classification are plotted as blue and red lines representing the positive and negative class. The class boundary is shown as thin dotted horizontal lines in each feature, while the maximum and minimum values for each feature in the data are shown as thin horizontal lines.

1. I **add a colour gradient which shows, the classification decisions around the point of interest, and how the quickly and significantly the decision changes**, if the input feature changes, one at a time. Note: When features are correlated, changing one feature at a time is not realistic (resolved by item #5 for interactive views, but not static views). The gradient also tells us which features are more influential than others—i.e., which have darker blues and reds. I also show the minimum and maximum values in the data for each feature and the class boundary, and I mark instances with an x, if it was predicted incorrectly.

2. I **add/ensure that support vectors of both classes are plotted to inform the reader with counterfactual information** as a more complete view on how the model makes classification decisions. Counterfactual information helps the reader evaluate the truthfulness/faithfulness of the model [164].

3. I **recommend finding points in the input space which are sufficiently stable and close to the point of interest, rather than finding the closest point on the class boundary**, for points whose classification I wish to change. Points are stable if they have a sufficiently high (or low) classification score toward the desired class and a sufficiently high velocity and acceleration toward the desired class.

4. I **recommend using a weighted Euclidean distance in the input space** to account for features which are immutable (zero weight) and features which have different costs—i.e., are harder or easier to change.

5. I **recommend automatically synchronizing changes in feature values which are confounded/dependent in real-life.** When two features are correlated, and I change one of them, I make a corresponding change in the other according to the correlation—and I call this a first order interaction. Any second order changes/interactions resulting from first order interactions are also applied to the extent of correlation between features. This ensures that my analyses are based on realistic points of interest.

5.6.3 Innate data from SVM: the local and global influence of instances

The SVM classifier's formula (eqn) is the sum of influences on a single test instance, from multiple training instances, with the sign of the target label, and with a weight based on proximity $k(\cdot, \cdot)$ multiplied by a weight α_i learned for each instance.

Table 5.12: Local (L) and global (G) measures of the influence of instances in SVM are innately provided by the support vectors α_i and kernel output $k(\cdot, \cdot)$.

Measure	Normalized measure
$L_V(\hat{z}_m \mid \underline{x}_i) = \text{abs}\left(k\left(\underline{x}_i, \underline{v}_m\right) \cdot \alpha_i / C\right)$	$L_{V\mathbf{n}}(\hat{z}_m \mid \underline{x}_i) = L_V(\hat{z}_m \mid \underline{x}_i) / G_X(\underline{x}_i)$
$G_X(\underline{x}_i) = \frac{1}{N} \sum_{j=1}^{N} \text{abs}\left(k\left(\underline{x}_i, \underline{x}_j\right) \cdot \alpha_i / C\right)$	$G_{X\text{n}}(\underline{x}_i) = \frac{1}{N} \sum_{j=1}^{N} \text{abs}\left(k\left(\underline{x}_i, \underline{x}_j\right) \cdot \alpha_i / C\right) / G_X(\underline{x}_i)$

Hence, we can look at the **local influence** of one training instance on one test instance, by ignoring the sum and the sign aspects of the equation.

From that, we can also determine the **global average influence** of a single training instance on a representative sample of the population—for which we choose training data rather than test data since it is usually a larger sample.

We propose formulas (Table 5.12) to measure these influences or effects, with and without normalization.

- L_V is the **local influence** of training instance $\underline{x}_i$ on test/validation prediction $\hat{z}_m$ for $\underline{v}_m$.
- G_X is the **global average** (over X) influence of training instance $\underline{x}_i$ on a representative sample of predictions $\underline{z}$
- $L_{V\text{n}}$ is the **normalized local influence** of training instance $\underline{x}_i$ on test/validation prediction $\hat{z}_m$ for $\underline{v}_m$
- $G_{X\text{n}}$ is the **normalized global average** (over X)

5.7 Non-innate views and data for SVM

Views or data are not innate to SVM, if they are only available for explicit Mercer kernels (or transparent kernels more generally) or if they require significant computation that is not innate to the SVM process. Three non-innate views or data for SVM are as follows:

1. The global and local influence of features are non-innate data, derived from the sensitivity-based computations performed in my proposed measure of simplicity of sensitivity.

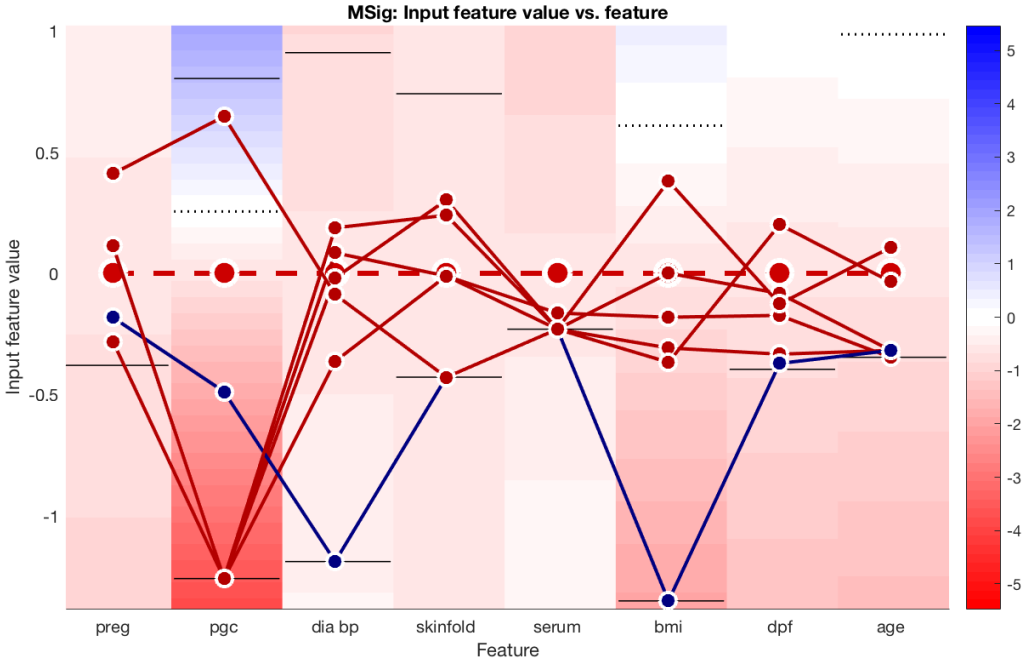

Figure 5.5: A Mercer sigmoid kernel, as an explicit Mercer kernel is shown in my proposed improvement to Barbella *et al.*'s innate view of SVM results. The chosen point of interest (POI) is plotted as a dashed red line, while the support vectors (instances) which contribute most to the POI's classification are plotted as blue and red lines representing the positive and negative class. The class boundary is shown as thin dotted horizontal lines in each feature, while the maximum and minimum values for each feature in the data are shown as thin horizontal lines.

2. In the innate static view of SVM from Barbella *et al.* [15] and my improvements, explicit Mercer kernels allow the view (Figure 5.5 below) describe how decisions change when any one or more features change at the same time—because features act independently in the kernel/model. Whereas for other kernels, the view only informs how decisions change one feature at a time.

3. A feature space plot.

5.8 Summary

My proposed kernels perform better than common kernels on the clinical studies/problems and data in my scope and conversely do not perform as well as the common kernels on the one non-clinical study/problem.

This is consistent with my previous findings [45] and it is also consistent with other experiments in the literature [295] (summarized in Section A.9) with the MSig kernel versus common kernels, on atomic data types like my clinical studies/problems versus complex data types with image pixel or coordinate data.

Chapter 6

Measuring model interpretability

For machine learning (ML) models, data and results, there is a demand for transparency, ease of understanding and explanations [126] to satisfy a citizen's "right to explanation" in the European Union [100] and to meet health care requirements for justification and explanation [21, 106].

Without quantitative measures of transparency and understandability, doctors (or users) will select models which maximize accuracy but may unnecessarily or unintentionally neglect or sacrifice transparency and understandability, or they will choose models in an ad hoc manner to try and meet all criteria. I refer to the transparency and understandability of models as ***inherent model interpretability***—defined further in Section 6.2.

I propose criteria and measures of inherent model interpretability to help a doctor select ML models (Table 6.1 on page 140 steps 1 and 2) which are more transparent and understandable, in a quantitative and objective manner. More transparent models can offer additional views of results (Table 6.1 on page 140 step 3) for interpretation. My measures facilitate the inclusion of better models as candidates and the selection of better models for use.

Table 6.1: Measures of inherent model interpretability facilitate model selection (bold text) in steps 1 and 2.

Step	Task	Basis for task
1	The doctor **selects candidate models** for learning and testing based on...	Data types and distributions, Inherent model interpretability (transparency of model)
2	The machine learns model weights for optimal accuracy with various parameters. The doctor **selects the model to use** based on...	Accuracy, Inherent model interpretability (transparency of model and understandability of results)
3	The doctor uses the model to classify new data. The doctor understands and interprets the result and model based on...	Theory, Views of results, Additional views of results
4	The doctor explains the result and model to a patient or peer based on...	Selected interpretations, Theory

Some of my proposed measures are specific to support vector machines (SVM), as one popular ML method. I perform experiments to validate the SVM measures against a set of propositions and evaluate their utility by concordance or matched pair agreement.

Notably, the proposed measures **do not** provide an interpretation or explanation. They also **do not** indicate how useful or meaningful a model is in the context of data. For example, a model that always classifies patient data as belonging to the positive class is very understandable (interpretable). I can easily construct the explanation of the model and result—all patients are classified as positive—but that does not mean that the model is useful, meaningful, appropriate, or unbiased. Accuracy and common sense address the latter issues. The proposed measures only indicate how understandable a model is, i.e., how likely I am **able** to provide an interpretation, as the necessary basis for subsequent explanation.

Making ML more interpretable facilitates its use in health care because there is a perception that ML is a black box [169] lacking interpretability which inhibits its use. Greater use is important because for a good number of health care problems and data, ML methods offer better accuracy in classification [47, 58, 198] than common alternatives among statistical methods, decision trees and rule-based methods and instance-based methods. Interpretable ML also facilitates research on models and model fit.

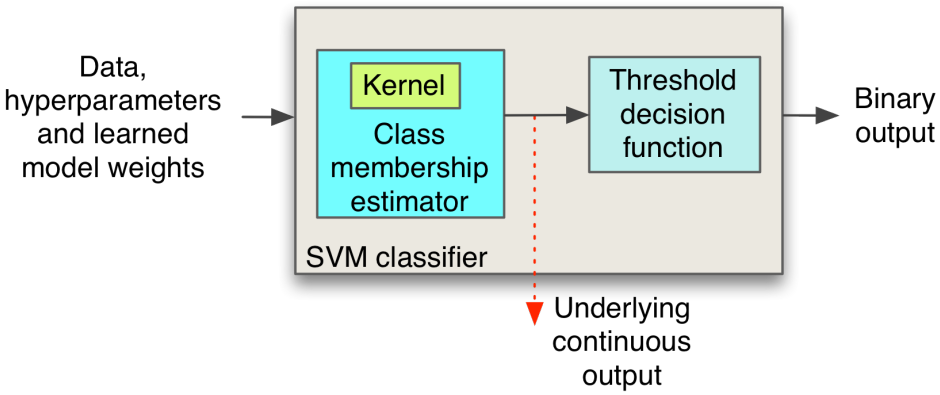

Figure 6.1: A model consists of a learning method, SVM in this case, and all of its associated parts as depicted above. Most machine learning and statistical models (or classifiers) have an underlying continuous output that most accurately describes the model's behaviour.

6.1 Models

I refer to a **posterior model** (e.g., Figure 6.1), or simply **model**, as a learning method (e.g., SVM, neural networks) with all of its associated learning/estimation functions (e.g., kernels and transfer functions), hyperparameters, structure (e.g., layers, connections, components in a composite kernel), constraints and learned model weights, *in the context of specific data*. A model only learns from, and has meaning in, the context of specific data.

I refer to an **initial model** as a model in the context of specific data with initial model weights prior to learning/iteration.

I refer to a **family of models**, or a **prior model**, as the set of models possible when hyperparameters are variables (not specified)—e.g., SVM with a Gaussian RBF kernel with unspecified box constraint and kernel width.

The prior, initial and posterior models are available at different points in the process of machine learning and/or statistical learning process (Figure 1.4).

Other notation is introduced in the context of discussion.

6.2 Inherent model interpretability

I propose the concept of **inherent model interpretability** as distinguished from an individual's understanding and I propose two measures for any learning method or model with numeric inputs.

Feynman said [101] that if I understand a concept I must be able to describe it at a freshman level, which often requires simplification or reduction, otherwise I don't really understand it. Badii *et al.* [13] express that complexity is closely related to understanding and that understanding comes from accurate models which use condensed information or reduction schemes.

Hence, I posit that the simpler a model is, the easier it is to describe, understand and interpret, with all other aspects of the model being equal. This leads to the following general measure.

6.2.1 Red herrings in transparency and interpretability

A red herring is said to be a logical fallacy that leads a reader toward a false conclusion. In the field of model interpretability there are several devices used by papers which are red herrings, such as:

- explaining a model with a separate externally-constructed model [4, 5, 216]
- using a measure of distance that differs from the kernel [15, 216]
- using perturbations which are non-local for a local model [160, 216]
- recovering a sample in the input data space from the interpretable data space [216]
- claiming that a model has fidelity for explanation based on agreement of binary classification decisions [216]. This allows construction of a post-facto explanation, however it is based on empirical results without methodology or theory to understand the model's behaviour.

Authors of papers with red herrings may have the best of intentions however substitutions and models based on empirical results fail to provide any understanding and insight, distracting users from better and more proper objectives. These red herrings also cause doctors to distrust ML methods and tools.

6.2.2 General measure of inherent model interpretability

As stated above, the simpler a model is, the more interpretable it is, inherently. Formally, I propose the following definition.

Definition 6.2.1. Inherent model interpretability (or understandability) U, is a measure with range $[0, 1]$ based on either: a measure of model transparency T in the same range,

the inverse of semi-infinite model complexity H_∞, or the inverse of finite model complexity H_b, respectively as follows:

$$U = \begin{cases} T & \text{(i) } T \quad \in [0,\ 1] \\ \frac{1}{1+(H_\infty - a)} & \text{(ii) } H_\infty \in [a,\ \infty) \quad a \in \mathbb{R}^+;\ a < \infty \\ 1 - \left(\frac{H_b - a}{b - a}\right) & \text{(iii) } H_b \in [a,\ b] \quad a,\ b \in \mathbb{R}^+;\ a,\ b < \infty \end{cases} \tag{6.1}$$

where:

- H_∞ and H_b are measures of model complexity based on parts [13] in the categories of information, entropy, code length or dimension [171],
- *inherent* indicates that the measure is independent of an individual, e.g., their specific learning and forgetting curves [210], and
- the multiplicative inverse [166] in (6.1).ii or additive inverse [272] in (6.1).iii are applied as needed for **absolute** or **relative** measure respectively according to the comparison required. The relative measure is preferred where applicable since it is more intuitive and interpretable (not shown).
  - e.g., to compare a set of models where the range $[a,\ b]$ is known to encompass them all, a relative measure (iii) is fine, however, to compare them to any future model where the maximum b is not known, use an absolute measure (ii), i.e., let $b = \infty$.

The separation of model interpretability into at least two parts, one part that is inherent to the model (and data) and another part that depends on the individual, aligns with the functionally-grounded approach [70].

In order to use this general measure, one must further define T, H_∞ or H_b, as I do in subsequent sections. I note also that measurement may be performed prior to, initially at, or posterior to, optimizing the model weight (Figure 6.2).

6.2.3 Simplicity of output sensitivity measure

I consider the continuous underlying output of a classifier (e.g., Figure 6.1 on page 141) to be the most accurate representation of a classifier's behaviour. It is available most learning classifiers, in machine learning or statistical learning, such as, neural networks,

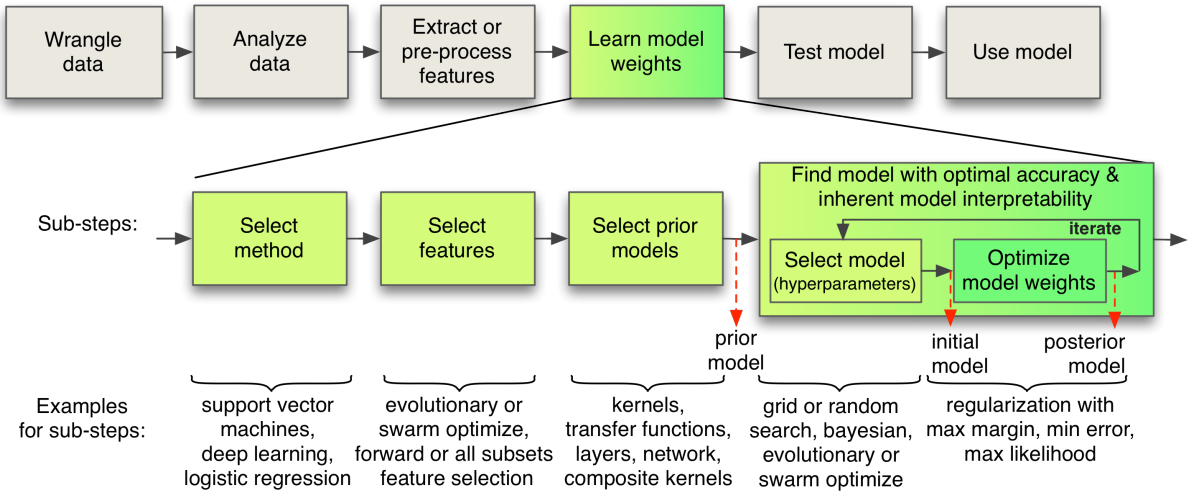

Figure 6.2: I measure the inherent model interpretability of prior, initial and posterior models available at several different points (dashed red arrows) in the process of machine/statistical learning (partially derived from [138]). Note: some steps may not apply to some methods and models.

SVM, logistic regression and naive bayes. It is also facilitated by most implementations, e.g., for SVM it is available in Matlab, R, Python, SPSS, Weka, libsvm and Orange, where the output may be the probability of the positive class or a non-probabilistic value, e.g., "classification score".

Some measure or analyze a classifier's behaviour based on its binary output instead [216]—this approach lacks fine-grained behavioural information. Others measure classifier behaviour by modeling its responses with a separate explanation model that provides a continuous output [216, 14]—this post hoc approach may not meet untested legal, assurance or business requirements.

I use the underlying continuous output, and the logic similar to the previous measure to posit that:

If a model is **uniformly sensitive** in its output to changing values in input features and instances, then its ***sensitivity*** is simple to describe, understand and interpret (as one value). Conversely, a model that is **differently sensitive** to each feature and instance is more difficult to describe, understand and interpret, in those terms or from that perspective. Formally, I propose the following definition:

Definition 6.2.2. The simplicity of output sensitivity U_{Hs} is a measure of inherent model interpretability. It describes the simplicity of the sensitivity of the model's continuous output (e.g., Figures 6.1) to changes in input. It is specified as the inverse of Shannon

entropy H_s with a finite range ((6.1) part iii), repeated below:

$$U_{H\text{s}} = 1 - \left(\frac{H_\text{s}}{H_\text{max}} \right) \qquad H_\text{s} \in [0,\ H_\text{max}] \tag{6.2}$$

$$H_\text{s} = -\sum_i f_i\left(s\right) \log f_i\left(s\right), \quad i = 1 \ldots N_s \tag{6.3}$$

$$H_\text{max} = -\sum_{i=1}^{|s|} \frac{1}{|s|} \log \frac{1}{|s|} \tag{6.4}$$

where s is the set of sensitivities $S_{j,q}$ of the model's continuous output $\hat{y}_\text{c}$ (the value which is underlying for a classifier) to small changes $\varepsilon = (0.1) \cdot 3\sigma$ in each input instance j, one feature q at a time,

$$s = \{S_{j,q}\} \tag{6.5}$$

$$S_{j,q} = \frac{\hat{y}_\text{c}\left(\underline{x}_j + \underline{\varepsilon}_q\right) - \hat{y}_\text{c}\left(\underline{x}_j - \underline{\varepsilon}_q\right)}{2\varepsilon} \tag{6.6}$$

$$\underline{\varepsilon}_q = \begin{bmatrix} \ldots & 0 & \varepsilon & 0 & \ldots \end{bmatrix}^\text{T} \qquad \varepsilon \text{in q}^{th} \text{ cell}$$

and where N_s is the number of bins according to standard binning methods for histograms [82, 230, 251].

I use entropy to measure the global complexity of sensitivities across the space for input data. In the literature, entropy has been applied quite differently to measure the information loss of perturbed features, to indicate their influence — I use entropy instead to measure the complexity of influence with perturbed features.

My measure uses a first-order central difference (first derivative approximation) as a standard and easy to understand approach to sensitivity that does not require knowing or differentiating the model's formulas. I can generalize this idea to second and third-order differences/derivatives, and so on, like the derivatives in deep Taylor decomposition [192] — but the latter requires a model's formulas and derivatives. Whereas [192] examines the local behaviours of a model, I do that and compute the complexity of the values.

I treat the entries $S_{j,q}$ as a set or random variable s (6.5) because I am measuring model interpretability overall, across features and instances, not within a feature nor within an instance.

Table 6.2: I identify criteria for model interpretability in the literature and translate these into proposed criteria which are objective rather than subjective.

Term	Criteria in the literature	ID	Proposed criteria
Interpretable [172] Decomposable [168]	Each calculation has an intuitive explanation [168].	(a)	The feature space is known/ explicit.
		(b)	The feature space has a finite number of dimensions.
	Inputs are interpretable, not anonymous or highly-engineered [168]. Generalized additive models are interpretable [172]	(c)	The model is generalized additive ***with*** known/ explicit basis/shape functions.
	Generalized linear models are interpretable [172]. The contributions of individual features in the model, are understandable [172].	(d)	The model is generalized linear [172]
		(e)	The model is multiplicative, e.g., probabilistic, ***with*** known/explicit basis/ shape functions.
	n/a	(f)	Model parts are uniform in function.
Transparent algorithm [168]	The training algorithm converges to a unique solution [168].	(g)	Model weights are learned by convex optimization or direct computation.

I note that instead of Shannon entropy, it may be possible to apply other types of entropy, such as Renyi entropy, Tsallis entropy, effective entropy or total information [214, 267, 93] and/or Kullback-Leibler (K-L) divergence [56], however such a change would require validation. Prior to this dissertation I experimented with discrete Kullback-Leibler (K-L) divergence as implemented by four measures in the ITK toolkit [255, 254], as an alternative to Shannon entropy, however, my experimental results with K-L divergence did not sufficiently match my expectations, so I focused on Shannon entropy as a more popular and credible measure.

I also implemented differential entropy [56], which is the continuous version of entropy and is defined as the K-L divergence from a uniform probability density function (pdf) to the pdf of interest, but put that aside based on the previously mentioned K-L divergence results and also because it was more compute intensive as it required a kernel density estimate.

Finally I note that the sensitivity portion of my measure (i.e., entropy aspect aside) differs from how other authors compute sensitivity globally across both instances and features [160].

6.3 Criteria for model transparency and a measure for SVM

I identify criteria for model transparency from the literature (Table 6.2) for any model, and propose new criteria in most cases, which are objective, not subjective, and thus suitable for a (quantitative) measure of model transparency.

I apply the proposed criteria (Table 6.2) for any model, to create a measure specific to kernel methods or support vector machines (SVM).

Table 6.3: For kernel methods, e.g., SVM, I propose the following Dirac (binary) measures ∂ of model transparency T. Let $\mathcal{X}_{\mathrm{T}}$ be the space of transparent features derived from simple transforms of the original features $\mathcal{X}$ which are not highly engineered: i.e., given data $\mathcal{X} = \{x\}$, let $\mathcal{X}_{\mathrm{T}} = \left\{x,\, -x,\, \frac{1}{x},\, \log(x),\, \tanh(x),\, \min(c_{\mathrm{top}}, x),\, \max(c_{\mathrm{bottom}}, x)\right\}$.

Name of measure and criterion met	Symbol for measure	Conditions for measure to be true
Explicit symmetric separable (a)	$\partial_{\mathrm{essep}}$	$k(\underline{x}, \underline{z}) = \phi(\underline{x})\,\phi(\underline{z}),\ \phi$ known $x_i, z_i \in \mathcal{X}_0,\ \mathcal{X}_0 \subseteq \mathcal{X}_{\mathrm{T}},\ \phi \in \mathcal{F},\ \phi : \mathbb{R}^n \to \mathbb{R}$
Finite (b)	∂_{fin}	$\dim(\mathcal{F}) < \infty$
Explicit Mercer (c)	∂_{eM}	$k(\underline{x}, \underline{z}) = \underline{\phi}(\underline{x})^{\mathrm{T}}\,\underline{\phi}(\underline{z})$ $. \quad = \sum_q \phi_q(x_q)\,\phi_q(z_q),\ \phi_q$ known $x_i, z_i \in \mathcal{X}_0,\ \mathcal{X}_0 \subseteq \mathcal{X}_{\mathrm{T}},\ \phi_q \in \mathcal{F},\ \phi_q : \mathbb{R} \to \mathbb{R}$
Explicit multiplicative (e)	$\partial_{\times}$	$k(\underline{x}, \underline{z}) = \prod_q \phi_q(x_q)\,\phi_q(z_q),\ \phi_q$ known $x_i, z_i \in \mathcal{X}_0,\ \mathcal{X}_0 \subseteq \mathcal{X}_{\mathrm{T}},\ \phi_q \in \mathcal{F},\ \phi_q : \mathbb{R} \to \mathbb{R}$
Uniform (f)	∂_{uni}	ϕ_q known and uniform e.g., (c) or (e) with $\phi_q = \phi\ \ \forall q$
Admissible (g)	∂_{adm}	k is positive definite (p.d.) [187] or k is conditionally p.d. (c.p.d.) [30]

I use the seven proposed criteria for inherent prior model interpretability (Section 6.3) to

define six Dirac (binary) measures for SVM (Table 6.3) meeting each criterion without overlap, except for criterion d (since all SVM kernels are generalized linear models).

I define an overall measure as follows:

$$\check{U}_{\partial} = 1/6\,(\partial_{\text{essep}} + \partial_{\text{fin}} + \partial_{\text{eM}} + \partial_{\times} + \partial_{\text{uni}} + \partial_{\text{adm}})$$

A benefit of this measure is that while independent of the data, it requires little computation and it informs model selection prior to optimization.

6.4 Other SVM model interpretability measures

In this section I propose measures specific to SVM.

Support vectors: In SVM, a subset of the patients in the data set are key to defining the model. They are known as support vectors since they support the definition of the model's class boundary and decision surface. For example, the decision regarding whether a patient has a disease or not, is determined by a subset of patients, e.g., 5 out of 200 patients, the model learned/picked as positive and negative examples of disease.

The more support vectors there are, the more complex the model is, with all other things being equal: $H_{sv} = sv$. SVM models have at least three support vectors in general — at least two to define the line, curve, hyperplane or surface that is the class boundary, and at least one to define the margin, so $sv \geq 3,\ sv \in \mathbb{N}$.

To select a model for one data set, or to compare results between two data sets, I know the maximum number of patients N, so $sv \leq N$, and I apply (equation 6.1 part iii) to obtain a relative measure, $U_{sv,r}$. Or to obtain an absolute measure $U_{sv,a}$, to compare against any current or future data set, I assume $N = \infty$ and apply (equation 6.1 part ii).

Degrees of freedom: Akaike includes all method and kernel hyperparameters and weights as among the degrees of freedom [244]. I calculate the prior complexity measure $\check{H}_{dof}$ with three terms comprised of: the number of SVM hyperparameters, e.g., 1 for C, the number of kernel hyperparameters, e.g., 1 for the kernel width for a Gaussian RBF kernel, the number of independent inputs, e.g., 1 for a Gaussian RBF kernel or stationary kernel, 2 otherwise. I calculate the posterior complexity measure H_{dof} with an additional term for the support vectors and apply the general measure for model interpretability.

$$\check{H}_{dof} = \check{dof} = d_{\text{SVM\_hyp}} + d_{\text{kernel\_hyp}} + d_{\text{input}}$$
$$H_{dof} = dof = d_{\text{SVM\_hyp}} + d_{\text{kernel\_hyp}} + d_{\text{input}} + sv$$

Relevant dimensionality estimate: The relevant dimensionality estimate (rde) [35] provides a way to measure the complexity of the SVM feature space induced by a kernel. There are two complexity measures $H_{rde\mathrm{T}}$ and $H_{rde\mathrm{L}}$ corresponding to two rde methods: the two-component model and the leave-one-out method, respectively.

6.5 Experiments to validate measures

I validate my proposed measures with sanity checks on formulas (not shown) and by agreement with propositions that describe my expectations and knowledge about model complexity and interpretability.

I create propositions based on expected relationships between measures, and check/test the propositions with a statement $\mathbf{P}$ and its inverse $\mathbf{P}^{-1}$ such as the following,

$$\mathbf{P}: \ \check{dof}_1 \leq \check{dof}_2 \ \overset{\text{usually}}{\rightarrow} \ U^*_{rde1} \geq U^*_{rde2} \tag{6.7}$$

$$\mathbf{P}^{-1}: \ \check{dof}_1 > \check{dof}_2 \ \overset{\text{usually}}{\rightarrow} \ U^*_{rde1} < U^*_{rde2} \tag{6.8}$$

where $\overset{\text{usually}}{\rightarrow}$ is a notation for: implies the majority of the time. I measure how much my results agree (Tables 6.4 and 6.5) with these propositions using either Kendall's W coefficient of rank correlation [143] or matched pair agreement [231], where the latter is applied to control for confounding factors.

If a proposition is robust, then the percentage of the concordance coefficient or matched pair agreement indicates how correct and useful the measure is, from that perspective. A measure has some utility, if it is correct the majority of the time, for different models/kernels and data sets.

I validate my propositions using two types of experiments (#1 and #2 as below). I run each experiment five times on each of three data sets from the University of California at Irvine repository: the Statlog Heart, Hepatitis and Bupa Liver data sets. Missing data in the Hepatitis data set are imputed with Stata, taking one of three multiple imputations with Monte Carlo Markov Chains. Bupa Liver data is used for these experiments with the common (misused) target [186] rather than the clinically meaningful target.

- Experiment Type #1: For each of 90 points chosen randomly in the hyperparameter space, I choose a pair of models, matched pairs [231], that differ by one hyperparameter/*dof* that is fixed in one and free in the other, and check propositions as

the percentage truth of the propositions. I use 3 pairs of kernels that differ by a single *dof*, e.g., a polynomial kernel of varying degree versus a linear kernel, a Gaussian RBF kernel with/without a fixed kernel width and a Mercer sigmoid kernel [45] with/without a fixed horizontal shift.

- Experiment Type #2: From the experiment type #1 I identify three points in the hyperparameter space which perform well for each kernel. For each of 3 fixed points, I choose 30 values of C equally spaced (as logarithms) throughout the range from 10^{-3} to 10^{6} and check propositions as the concordance of the left-hand side with the right-hand side in the propositions, using Kendall's W coefficient of rank correlation [143]. If the right-hand side should have opposite rank to the left-hand side then I apply a negative to the measure on the right-hand side for concordance to measure agreement of rank. I use the following kernels: linear, polynomial, Gaussian RBF and Mercer sigmoid kernel [45].

6.5.1 Propositions

Proposition 6.5.1.

The majority of the time I expect that a model with less degrees of freedom $\check{dof}_1$*, with all other things being equal when compared to another model with* $\check{dof}_2$*, will be simpler and have a relevant dimensionality estimate (rde) [35] that is less than or equal to the other model and therefore be more interpretable/understandable* (U^*_{rde})*:*

$$\textbf{1a}: \ \check{dof}_1 \leq \check{dof}_2 \quad \overset{usually}{\rightarrow} rde_1 \leq rde_2 \tag{6.9}$$

$$\textbf{1b}: \ \check{dof}_1 \leq \check{dof}_2 \quad \overset{usually}{\rightarrow} U^*_{rde1} \geq U^*_{rde2} \tag{6.10}$$

This applies to rde with the two-component model (rdeT) and the leave-one-out method (rdeL).

Proposition 6.5.2.

In SVM, the hyperparameter C is called the box constraint or cost of error. Authors have remarked [233, remark 7.31] that C is not an intuitive parameter, although it has a lower bound for use $C \geq \frac{1}{N}$ and its behaviour suggests $C \doteq \frac{1}{\nu N}$, where ν is a proportion of support vectors. I therefore expect that a model with a higher value C_1 versus a second model with C_2

will have less support vectors (sv) and consequently be more interpretable/understandable (U_{Hs}):

$$\mathbf{2a}: \ C_1 \geq C_2 \ \overset{usually}{\rightarrow} \ sv_1 \leq sv_2 \tag{6.11}$$

$$\mathbf{2b}: \ sv_1 \leq sv_2 \ \overset{usually}{\rightarrow} \ U_{Hs1} \geq U_{Hs2} \tag{6.12}$$

$$\mathbf{2c}: \ C_1 > C_2 \ \overset{usually}{\rightarrow} \ U_{sv,a1} \geq U_{sv,a2} \tag{6.13}$$

$$\mathbf{2d}: \ C_1 > C_2 \ \overset{usually}{\rightarrow} \ U_{Hs1} \geq U_{Hs2} \tag{6.14}$$

This applies to simplicity of sensitivity U_{Hs} with any binning method.

My experiment uses three binning methods: Scott [230] $U_{H\text{sc}}$, Freedman-Diaconis [82] $U_{H\text{fd}}$ and Sturges [251] $U_{H\text{st}}$.

Proposition 6.5.3.

The majority of the time I expect that, if a prior measure is useful, then it reflects the same rankings as the posterior measure,

$$\mathbf{3}: \ U^*_{Hs1} \leq U^*_{Hs2} \ \overset{usually}{\rightarrow} \ U_{Hs1} \leq U_{Hs2} \tag{6.15}$$

Proposition 6.5.4.

I expect that the linear kernel is the simplest of all kernels with greater transparency than other kernels such as the polynomial, Gaussian RBF kernel, sigmoid and Mercer sigmoid kernels, whereby,

$$\mathbf{4}: isLinear(k_1) > isLinear(k_2) \ \rightarrow \ \check{U}_{\partial 1} > \check{U}_{\partial 2} \tag{6.16}$$

Table 6.4: The results from propositions using experiment #2 validate the support vector measure U_{sv} and simplicity of sensitivity measure with Sturges binning $U_{H\text{st}}$.

Proposition	Measure & Result	Agreement %	Comment
2a	sv	82 ± 2.3	C validates sv, supports B3
2b	$U_{H\text{sc}}$	53 ± 3.3	$U_{H\text{sc}}$ not distinguished by sv
	$U_{H\text{fd}}$	48 ± 3.7	$U_{H\text{fd}}$ not distinguished by sv
	$U_{H\text{st}}$	62 ± 3.5	sv validates $U_{H\text{st}}$
2c	U_{sv}	81 ± 2.3	C validates U_{sv}
2d	$U_{H\text{sc}}$	54 ± 3.3	C validates $U_{H\text{sc}}$
	$U_{H\text{fd}}$	49 ± 3.7	$U_{H\text{fd}}$ not distinguished by sv
	$U_{H\text{st}}$	64 ± 3.2	C validates $U_{H\text{st}}$

Legend: Green = affirmative result. Yellow = inconclusive result. Red = contrary result.

Table 6.5: The results from propositions using experiment #1 validate the relevant dimensionality measures rdeT and rdeL, the initial model interpretability measures based on relevant dimensionality $U^*_{rde\text{T}}$ and $U^*_{rde\text{L}}$, the use of prior measures of simplicity of sensitivity as proxies for posterior measures, and the measure of kernel transparency $\check{U}_\partial$. $\check{U}_\partial$.

Proposition	Measure & Result	Agreement %	Comment
1a	rdeT	63 ± 5.0	$\check{dof}$ validates rdeT, supports A2
	rdeL	59 ± 5.2	$\check{dof}$ validates rdeL, supports A2
1b	$U^*_{rde\text{T}}$	62 ± 5.0	$\check{dof}$ validates $U^*_{rde\text{T}}$
	$U^*_{rde\text{L}}$	59 ± 5.2	$\check{dof}$ validates $U^*_{rde\text{L}}$
3	$U^*_{H\text{sc}}$ as a proxy	72 ± 3.1	$U_{H\text{sc}}$ validates $U^*_{H\text{sc}}$ as a proxy
	$U^*_{H\text{fd}}$ as a proxy	76 ± 3.5	$U_{H\text{fd}}$ validates $U^*_{H\text{fd}}$ as a proxy
	$U^*_{H\text{st}}$ as a proxy	80 ± 3.2	$U_{H\text{st}}$ validates $U^*_{H\text{st}}$ as a proxy
4	$\check{U}_\partial$	100 ± 0	k_{Lin} vs. others, validates $\check{U}_\partial$

Legend: Green = affirmative result. Yellow = inconclusive result. Red = contrary result.

6.6 Experimental results

I summarize the results of my validation tests (Table 6.4 and 6.5) as follows: I recommend $\check{U}_\partial$ and U_{sv} as good measures. I find that U^*_{rdeT}, U^*_{rdeL} and $U_{H\text{st}}$ are measures which are of limited use, because they may be wrong one third of the time when providing guidance on decisions. $U_{H\text{sc}}$ and $U_{H\text{fd}}$ are not distinguished from chance by my propositions and

thus not recommended. Finally, if $U_{H\text{st}}$ is validated to a greater degree in the future, then the initial measure $U^*_{H\text{st}}$ has been shown to be a good proxy for it, incurring some loss of information.

My proposed measure of kernel transparency $\check{U}_{\partial}$, a prior measure, scored 100% agreement. This is a good measure that may be used a priori, but it is high-level and not specific to the match between a model and data. No surprises or complexities arose regarding the attributes of kernels.

The general measure based on the number of support vectors, U_{sv}, scored $81 \pm 2.3\%$ agreement—this is a good measure.

My proposed simplicity of sensitivity measure with Sturges binning [251] $U_{H\text{st}}$ scored $64 \pm 3.2\%$ and $62 \pm 3.5\%$, which is of limited use.

The same measure with Scott binning [230] ($U_{H\text{sc}}$), however, is barely distinguishable from chance in one test, and not distinguishable in another, and with Freedman-Diaconis binning [82] ($U_{H\text{fd}}$) it is not distinguishable from chance in both tests. I recommend further validation to examine the role of confounding factors such as kernel width/scale along with C per [65, 18].

If the simplicity of sensitivity measure $U_{H\text{st}}$ can be validated to a greater degree in the future, then the initial measure $U^*_{H\text{st}}$ which scores $80 \pm 3.2\%$ agreement with it, may be used in its place to avoid optimization, or to gain an initial estimate prior to optimization. For now, the prior measure is not advisable since $80\% \times 64\% = 51\%$.

The general measure based on the relevant dimensionality [35] of the feature space, U^*_{rdeT} and U^*_{rdeL} scored $62 \pm 5.0\%$ and $59 \pm 5.2\%$ agreement, respectively. These are of some use. I did not include Braun's noise estimate [35], which in hindsight should improve the measure.

Table 6.6: Result for $\check{U}_{\partial}$ confirm that the linear kernel is more transparent than other kernels.

Dirac measure	Description	Linear	Polynomial	Gaussian RBF	Sigmoid	Mercer Sigmoid
∂_{essep}	Explicit Sym. Sep.	✓	×	×	×	×
∂_{fin}	Finite	✓	✓	×	×	✓
∂_{eM}	Explicit Mercer	✓	×	✓[55]	×	✓
$\partial_{\times}$	Multiplicative	×	×	×	×	×
∂_{uni}	Uniform	✓	×	×	×	✓
∂_{adm}	Admissible	✓	✓	✓	×	✓
$\check{U}_{\partial}$ (%)		83	33	33	0	67

Legend: Green = top result. Light green = second best result.

6.7 Model selection with accuracy and inherent model interpretability

I apply model interpretability to results in a toy problem (Figure 6.7). When I select results for maximum accuracy with the Gaussian RBF kernel, I find that the top result in my sorted list of results achieves 100% accuracy (rounded to no decimal places) with 51 support vectors, while the second best result also achieves 100% accuracy with 40 support vectors and the fifth best result according to the list also achieves 100% accuracy with 25 support vectors.

Selecting results for maximum interpretability $U_{sv,r}$, I find the top result uses 9 support vectors for 99% accuracy and the fourth best result uses 10 support vectors for the same accuracy.

I plot the results (Figure 6.3 on page 155) of accuracy versus interpretability $U_{sv,r}$ (above 80% in each) and find that there are many results which are highly accurate and highly interpretable, i.e., above 96% in both. These results indicate that there is not a trade-off between accuracy and model interpretability based on support vectors in this data set.

I also plot the results of accuracy versus interpretability $U_{sv,r}$ for other data sets (Figure 6.4 on page 155 and Figure 6.5 on page 156) and it is clear that there is no trend in all points showing a trade-off between accuracy and model interpretability, although this trend may be present at the pareto front. A trade-off trend would show as an inverse correlation, a trend line running from the top left to the bottom right—instead, high interpretability is

consistently achievable with high accuracy, i.e., there are points toward the top right of a bounding box for all points.

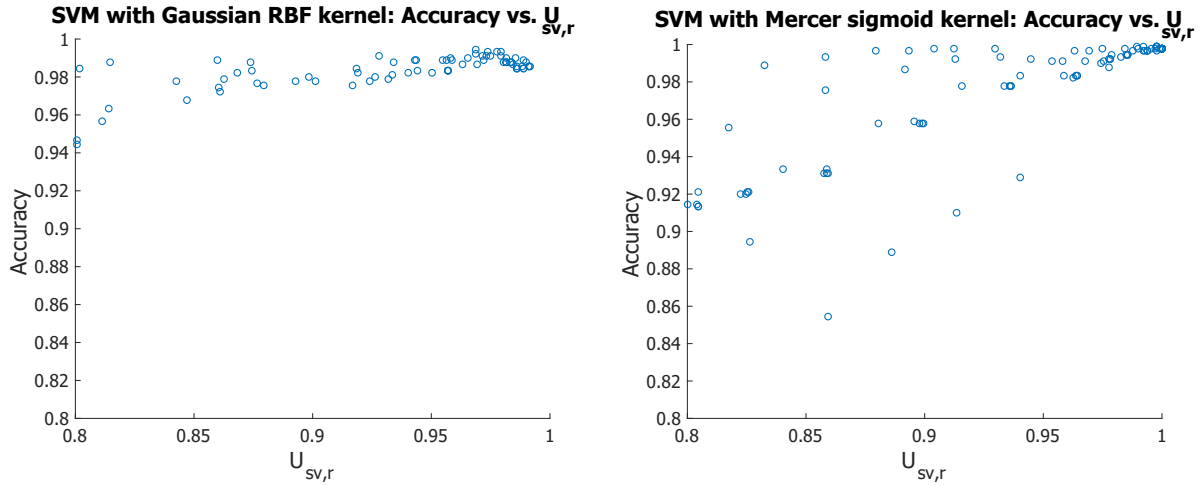

Figure 6.3: In classification for the toy problem, there are many results with high accuracy and high inherent model interpretability, with almost no sacrifice in the latter for maximum accuracy.

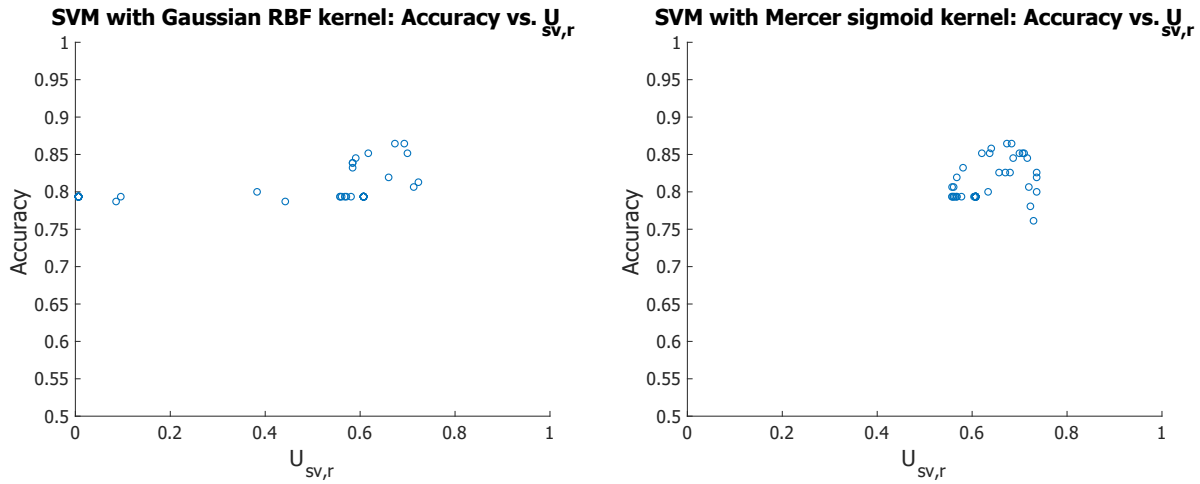

Figure 6.4: In classification with the Hepatitis data set there is a less than 5% sacrifice in inherent model interpretability for the highest accuracy.

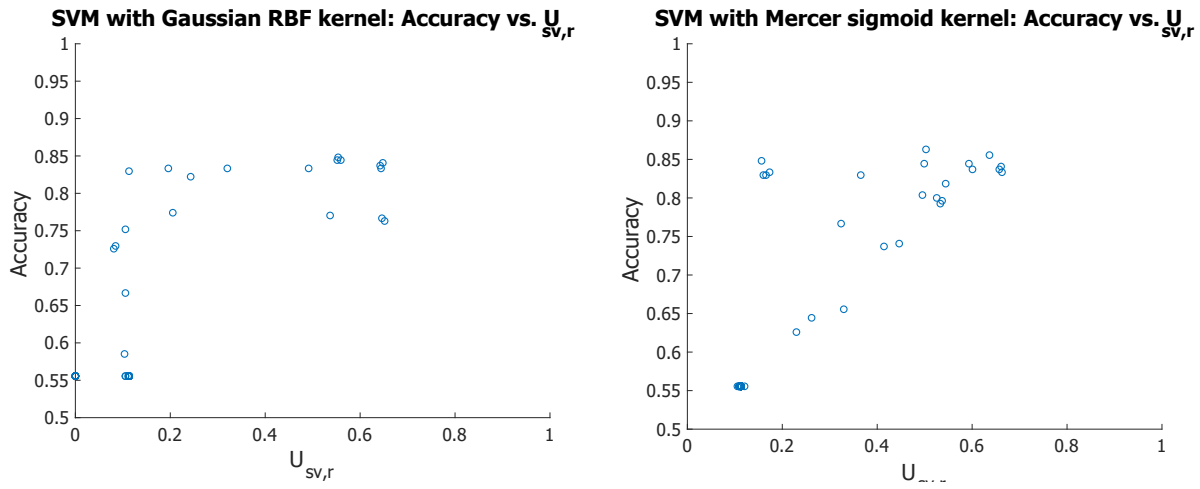

Figure 6.5: In classification with Statlog Heart data there are points with high accuracy and high inherent model interpretability, with minimal sacrifice, 1% and 2%, respectively.

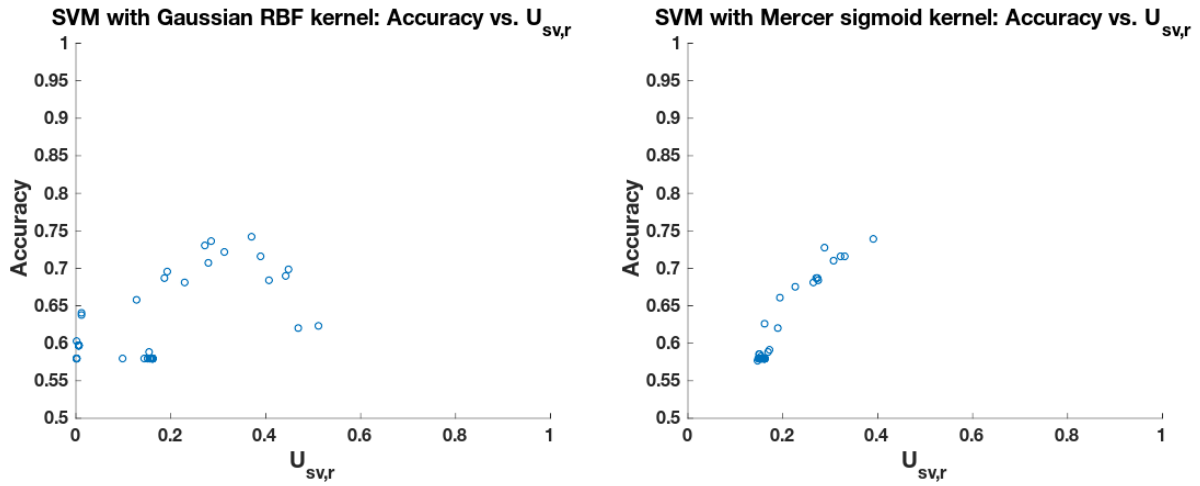

Figure 6.6: In classification with the Bupa liver data set there is a 20% and 0% sacrifice, respectively, in inherent model interpretability for the highest accuracy.

6.7.1 Study

I support the validity of model interpretability measures by performing tests related to two general hypotheses I expect to be true, as applied to two toy problems in classification. My two general hypotheses are:

Hypothesis 6.7.1. *SVM* ***will not*** *fit/separate data well if the kernel's class boundary* ***does not*** *match the instances of data, and conversely*

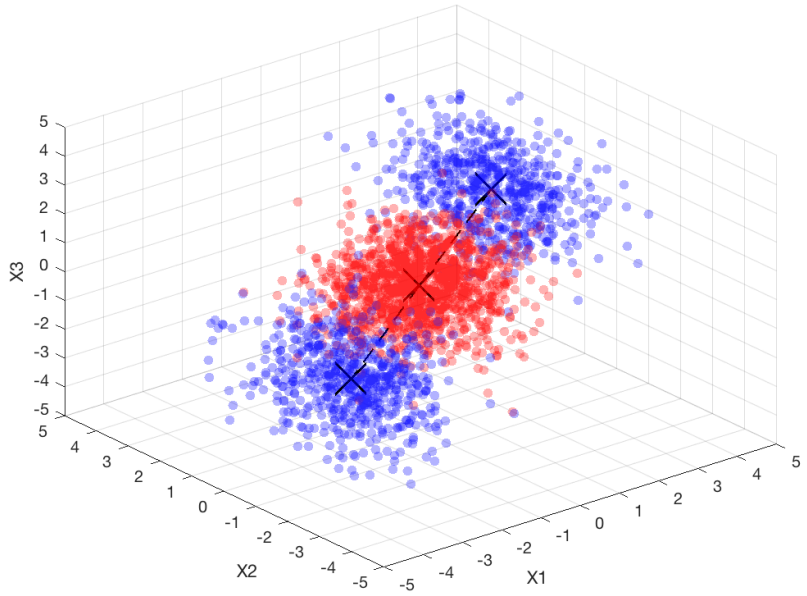

Figure 6.7: Toy problem #1: Classification data for two classes, blue and red, which are not linearly separable — before normalization.

Hypothesis 6.7.2. *SVM **will** fit/separate data well if the kernel's class boundary **matches** the instances of data.*

When I apply these hypotheses to toy problem #1 they become (Figure 6.7):

Hypothesis 6.7.3. *The linear kernel's class boundary matches data which can be separated by a hyperplane, so it is not a good fit to toy problem #1 (Figure 6.8).*

Hypothesis 6.7.4. *The Gaussian RBF kernel creates a circular class boundary around each support vector (dot surrounded by a circle) (Figure 6.9), so it matches or fits any distribution of data including toy problem #1.*

Before I can test these hypotheses, I provide background on the problem, discuss related work and propose measures, further hypotheses and tests. I then perform the tests, discuss the results and then examine a second toy problem. I then provide conclusions and areas for future work.

6.7.2 Revised hypotheses

I revisit the hypotheses from the introduction as applied to toy problem #1 as follows, and add new comments on accuracy:

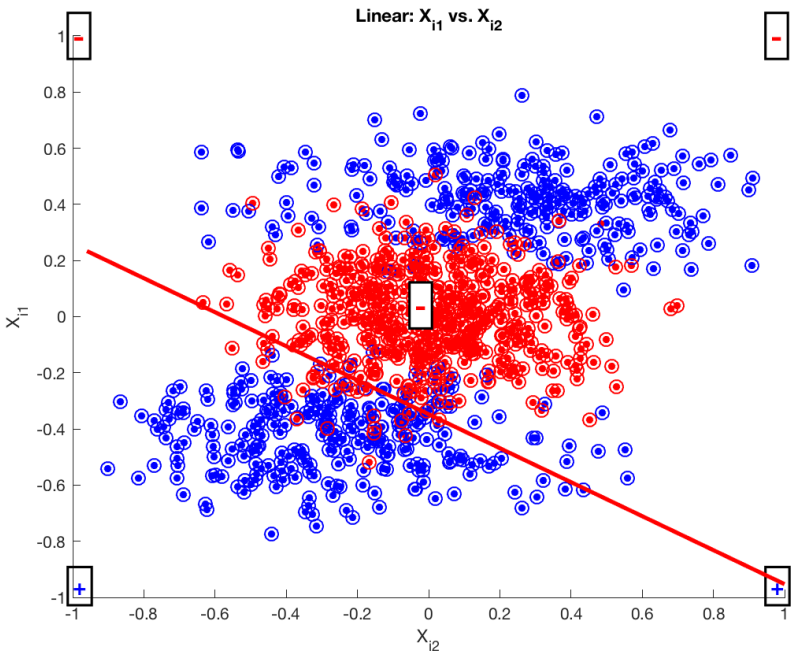

Figure 6.8: SVM with a linear kernel **does not** fit/separate toy problem #1 data well, per the class boundary (red line). The boxes at the center and corners of the plot show the predicted class as blue (positive) or red (negative) versus the actual class labels, blue or red for each point. Circled points are support vectors — every point in this plot. I show the fit for the most accurate result in two dimensions for clarity.

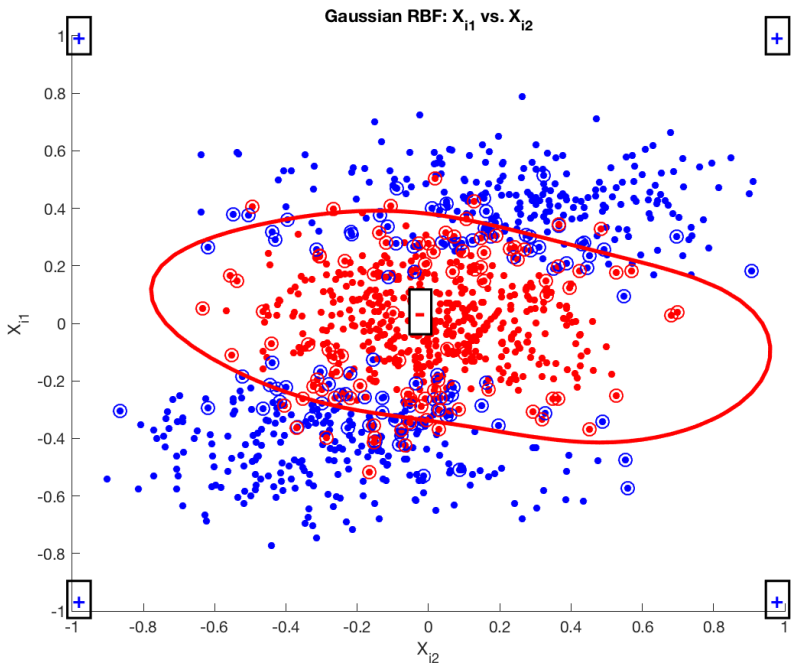

Figure 6.9: SVM with a Gaussian RBF kernel **fits/separates** toy problem #1 data well, per the class boundary (curve red line). Circled points are support vectors.

Hypothesis 6.7.5. *I expect that the linear kernel **does not** quantitatively fit/separate toy problem #1 well — i.e., low accuracy a, on average.*

Hypothesis 6.7.6. *I expect that the Gaussian RBF kernel **does** quantitatively fit/separate toy problem #1 well — i.e., high accuracy a, on average.*

I would like to compare the linear and Gaussian RBF kernels at a point of similarly high accuracy, but the linear kernel never achieves high accuracy. One may consider measuring accuracy per unit complexity to achieve a fair comparison, but there is no way to know what the proper scaling or relationship should be in that trade-off.

So instead, I can compare the accuracy of the two kernels at a point of similarly high model interpretability — i.e., high simplicity or low complexity:

Hypothesis 6.7.7. *If SVM with a linear kernel creates a class boundary (geometry) that **does not sufficiently match** the distribution of instances in the data, while SVM with a Gaussian RBF kernel **does**, then for similarly high model interpretability $U_{r:sv}$, the latter will achieve better goodness of fit, i.e., higher accuracy a.*

Since SVM with a linear kernel does not fit well I expect it to use many support vectors to try to achieve a good fit (Figure 6.8).

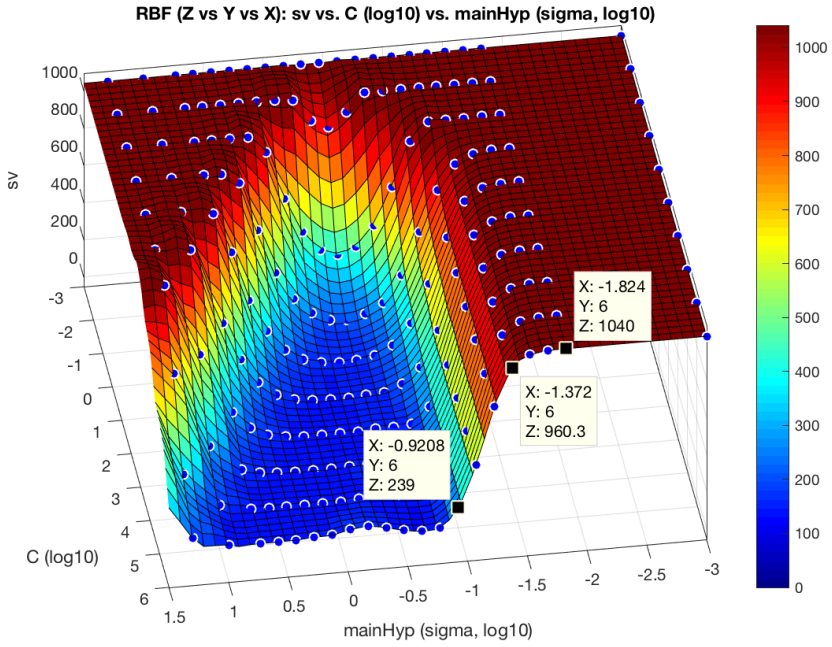

Figure 6.10: For toy problem #1: the Gaussian RBF kernel has a substantive range of hyperparameters which yield few support vectors corresponding to high model interpretability in the next figure.

In contrast, a Gaussian RBF kernel should **not need** more support vectors (complexity) to achieve its best fit/accuracy over a moderately suboptimal fit. It should **not need** to sacrifice model interpretability $U_{\text{r:sv}}$ (simplicity) for accuracy a. This is the notion of a naturally good fit:

Hypothesis 6.7.8. *If SVM with a Gaussian RBF kernel creates a class boundary (geometry) that* ***matches*** *the distribution of instances in the data, then for a wide range of parameter values σ and C, the Gaussian RBF kernel will achieve high model interpretability $U_{r:sv}$ and high accuracy a.*

I expect hypotheses C and D above to be true, and I use them to validate or test if my proposed measure $U_{\text{r:sv}}$ meets my expectations about model interpretability and accuracy. Hypothesis C is relative while hypothesis D is absolute.

6.7.3 Method

I perform an SVM classification experiment using C-SVM (not ν-SVM) since C-SVM is the most common implementation used (e.g., by Matlab, Weka). I generated data using

Gaussian functions for each of three clusters (Figure 6.7). Seven values of the kernel width σ were chosen, to match the data:

$$\sigma \in \left\{ \begin{array}{ccccccc} 0.06 & 0.085 & 0.12 & 0.17 & 0.42 & 0.71 & 1.0 \end{array} \right\}$$

Despite the geometric scale of the handpicked kernel widths, when I pick σ randomly I do so using a base ten logarithmic scale from 10^{-4} to 10^{6}. This scale is almost the same scale I use for randomly picking values for the SVM cost of error C: 10^{-3} to 10^{6}, however for this experiment I handpick the following:

$$C = 10^{x} \mid x \in \left\{ \begin{array}{ccccccccccc} -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \right\}$$

This results in a total of seven by nine, or 63 models, applied to three folds of data.

6.7.4 Toy problem #1 discussion and results

For the Gaussian RBF kernel classifying toy problem #1 data, there is smooth blue floor of few support vectors (Figure 6.11, left) that corresponds to a smooth red ceiling of inherent high model interpretability (Figure 6.11, middle) and a smooth red ceiling of high accuracy (Figure 6.11, right) which is wider for reasons explained below. I refer to the plateaus that overlap in all three as the "good fit" region.

At the back boundary of the "good fit" region where the cost of error C is smaller, as C decreases, e.g., $\log_{10} C \leq -1$, errors are not given much weight, so the model under-fits the data resulting in lower accuracy (Figure 6.11, right) and more support vectors (Figure 6.11, left).

At the right boundary of the "good fit" region where the kernel width σ is smaller, as σ decreases, e.g., $\log_{10} \sigma \leq -1$, more support vectors are needed (Figure 6.11, left) to maintain a contiguous, smooth and accurate class boundary from one support vector to the next. Accuracy begins to drop at $\log_{10} \sigma = -1.37$ (Figure 6.11, right) as the number of support vectors saturates to the maximum and it continues dropping in the region of saturation.

At the left boundary of the "good fit" region where the kernel width σ is larger, there is a diagonal ridge (Figure 6.11, left) showing a trade-off relationship between σ and C described by [18] where similar curvature in the class boundary and similar accuracy is achieved for tuples of (σ, C) whereby an increase in σ can decrease curvature that can be offset by increasing C to increase curvature.

My results confirm hypothesis C for toy problem #1 because for comparably high model interpretability $U_{sv} \approx 77.6\%$ and 78.4% the Gaussian RBF kernel achieves $a \approx 87\%$ and 91% (Figure 6.11) versus the linear kernel's $a \approx 53\%$ and 43% (Figure 6.13). This lends support for the measure U_{sv}.

My results confirm hypothesis D for the Gaussian RBF kernel in toy problem #1 because there is a wide portion of the parameter space (σ, C) that yields high model interpretability $U_{sv} \approx 75\%$ to 84% (Figure 6.11) and high accuracy $a = 85\%$ to 92% (ibid). This also lends support for the measure $U_{\text{r:sv}}$.

For the Gaussian RBF kernel which **fits** the data in toy problem #1 well, there is **no trade-off** between model interpretability (re support vectors) and accuracy within the kernel (with different hyperparameters). For the linear kernel which **does not fit** the data well, there **is** a **small trade-off** between model interpretability (re support vectors) and accuracy.

6.7.5 Toy problem #2 definition, discussion and results

For toy problem #2 (Figure 6.15), I generate classification data in two classes with two real-valued features, which are not linearly separable and therefore not suited to a linear kernel.

The data in toy problem #2 are geometrically and intuitively suited to a Gaussian RBF kernel which creates a class boundary suitable for any shape of training data (Figure 6.15, middle) and a Mercer sigmoid kernel (Figure 6.15, right) which forms an L-shaped class boundary for a quadrant (or orthant) at any point of origin.

My results confirm hypothesis #1 for toy problem #2 because for comparably high model interpretability $U_{sv} \approx 96.2\%$ and 79% the Gaussian RBF kernel achieves $a = 99.6\%$ and 97.1% (Figure 6.17) and the Mercer sigmoid kernel achieves $a = 99.7\%$ and 91.8% (Figure 6.19) versus the linear kernel's $a = 87.2\%$ and 89.1% (Figure 6.21). This lends support for the measure U_{sv}.

My results confirm hypothesis #2 for the Gaussian RBF kernel in toy problem #2 because there is a wide portion of the parameter space (σ, C) that yields high model interpretability $U_{sv} \approx 79\%$ to 98.5% (Figure 6.17) and high accuracy $a = 97\%$ to 99.6% (ibid). This also lends support for the measure U_{sv}.

My results confirm hypothesis #2 for the Mercer sigmoid kernel in toy problem #2 because there is a wide portion of the parameter space (σ, C) that yields high model interpretability

$U_{sv} \approx 77\%$ to 97.3% (Figure 6.19) and high accuracy $a = 91\%$ to 99.7% (ibid). This also lends support for the measure U_{sv}.

For the Gaussian RBF and Mercer sigmoid kernels which fit the data in toy problem #2 well, there is **no** trade-off between model interpretability (re support vectors) and accuracy, either within the kernel (with different hyperparameters) or between the two kernels, where the Mercer sigmoid kernel is more transparent (Section Section 6.6). For the linear kernel which **does not** fit the data well, there **is** a **small** trade-off between model interpretability (re support vectors) and accuracy.

6.7.6 Study conclusion

I proposed a measure of model interpretability U_{sv} and based on two hypotheses I expect to be true, I confirmed that the measure U_{sv} meets my expectations regarding two toy problems. I observed that within and between kernels which fit the data well, there is no loss of accuracy, i.e., no trade-off, to achieve model interpretability (re support vectors), whereas with the linear kernel which does not fit the data well, there is a small trade-off.

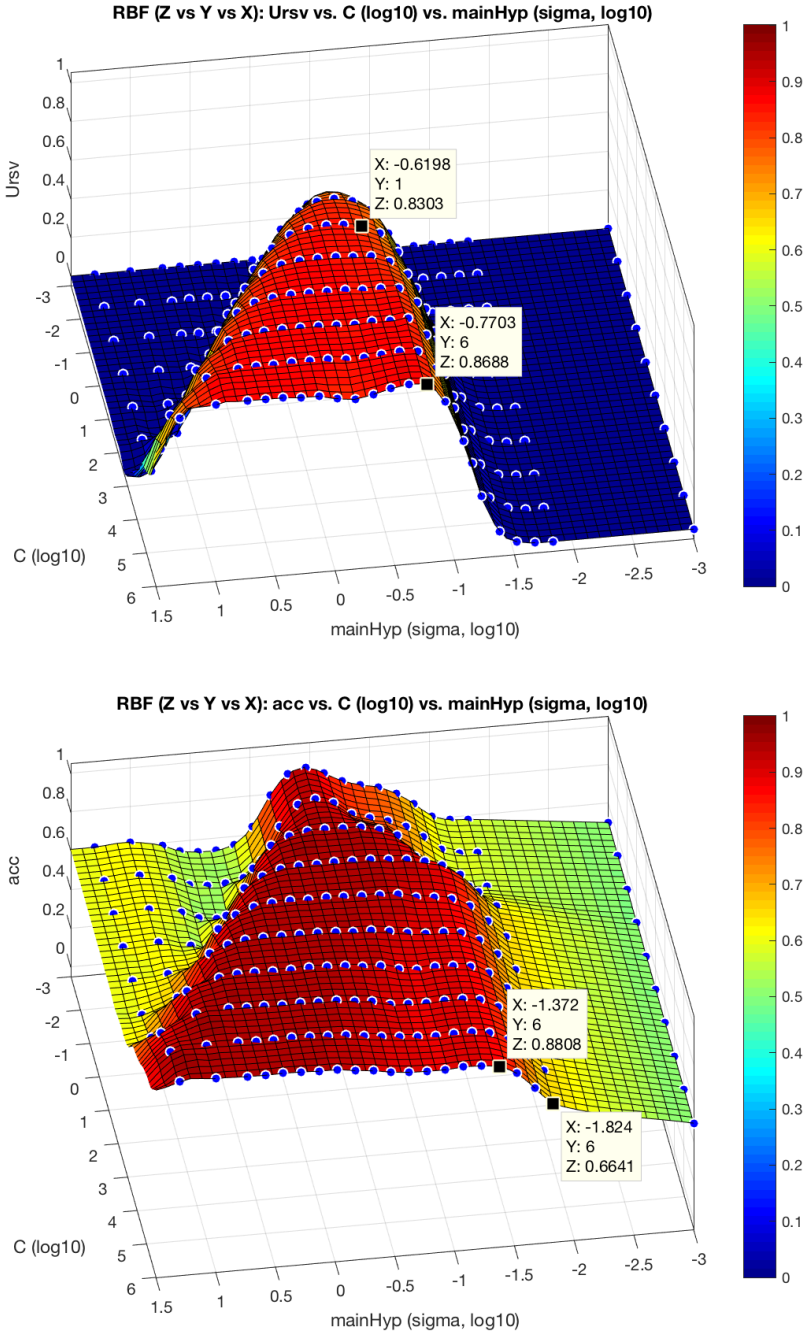

Figure 6.11: For toy problem #1: the Gaussian RBF kernel has a substantive range of hyperparameters which yield high model interpretability U_{sv} (top) and high accuracy a or acc (bottom), which confirms hypothesis C. For this kernel which fits the data well, there is **no** trade-off between model interpretability (re support vectors) and accuracy.

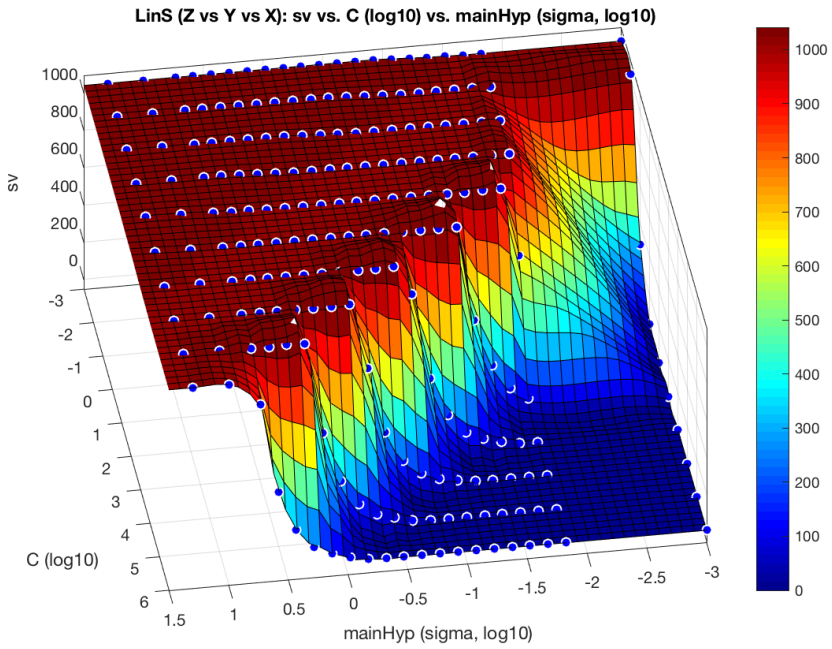

Figure 6.12: For toy problem #1: the linear kernel has a substantive range of hyperparameters which yield few support vectors (corresponding to high model interpretability in the next figure).

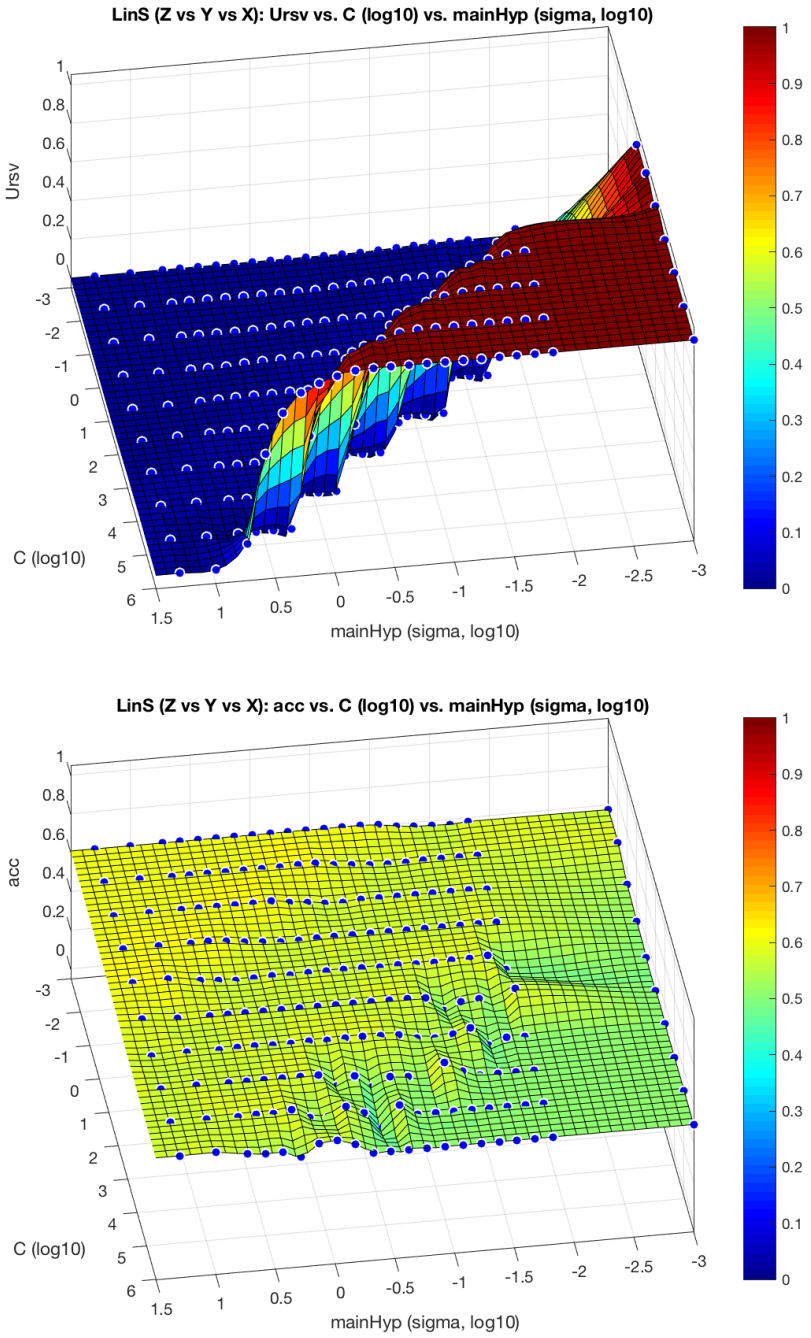

Figure 6.13: For toy problem #1: the linear kernel has a wide range of hyperparameters which yield high model interpretability U_{sv} (top) but low accuracy a or acc for all hyperparameters. For approximately the same U_{sv} as the Gaussian RBF kernel, the linear kernel's $U_{sv} \approx 77.8\%$ and 78.9% (top) corresponds to $a \approx 53\%$ and 43% (bottom), which in comparison to the Gaussian RBF confirms hypothesis #1. For this kernel which **does not** fit the data well, there **is** a **small** trade-off between model interpretability (re support vectors) and accuracy.

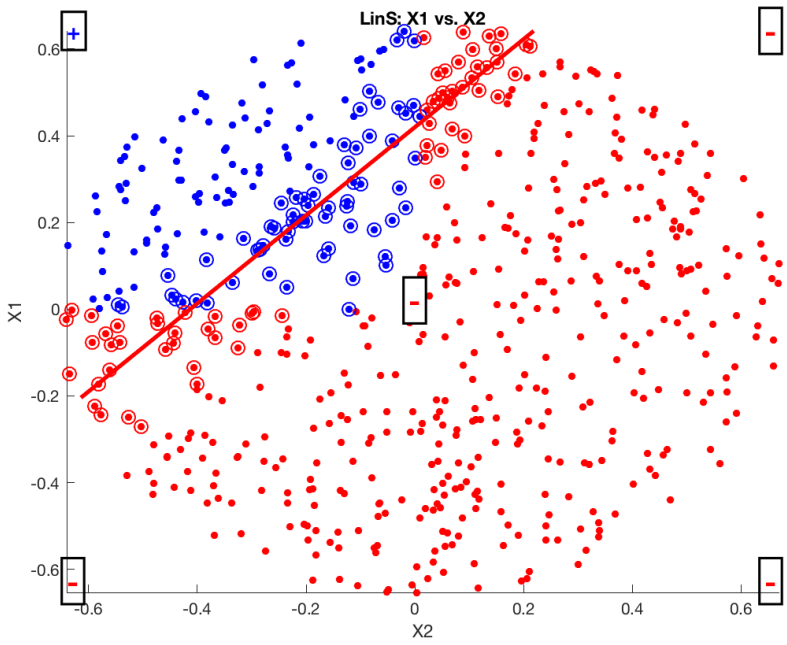

Figure 6.14: Toy problem #2: classification data in two classes with two real-valued features. The data are not linearly separable and therefore not suited to classification by an SVM with a linear kernel. Circled points are support vectors.

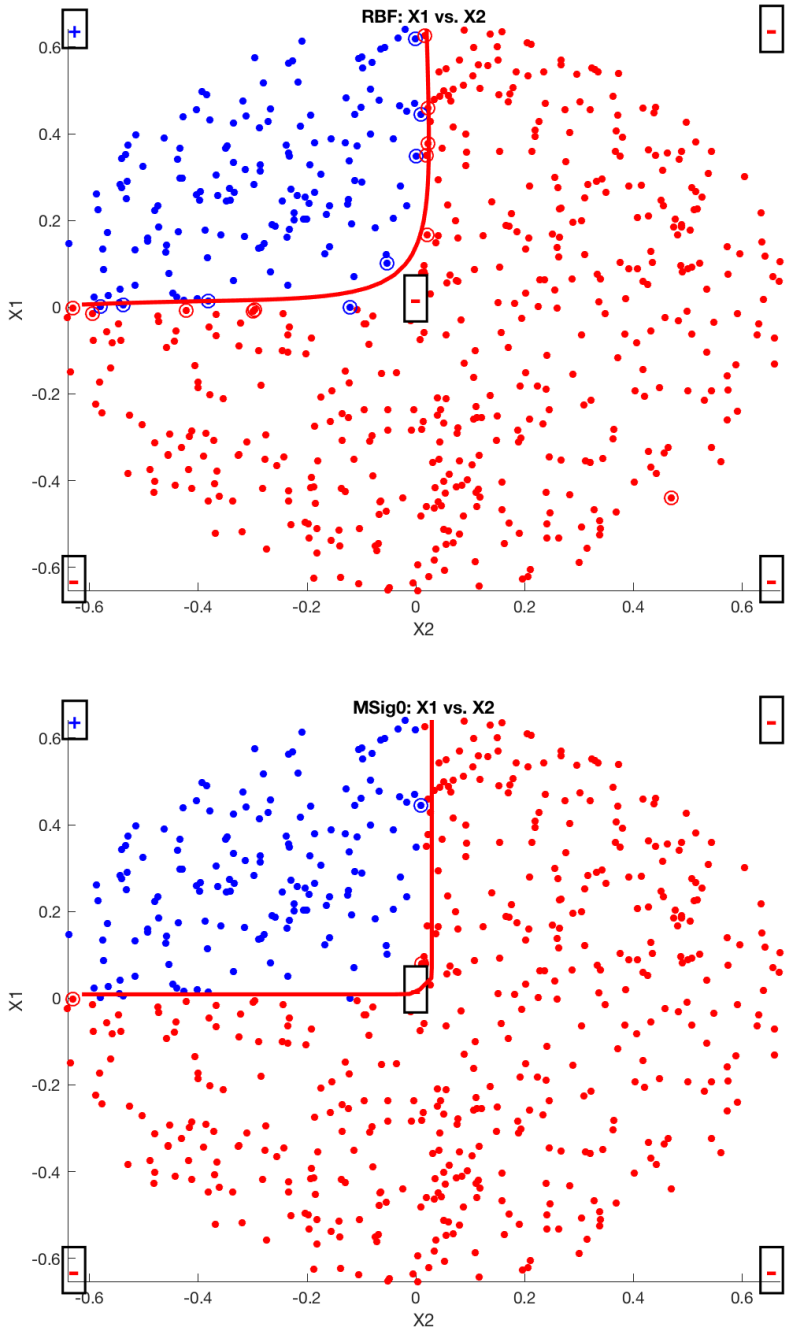

Figure 6.15: Toy problem #2: The data are suited to classification by an SVM with a Gaussian RBF kernel which uses 19 support vectors (top) as well as a Mercer sigmoid kernel which uses 3 support vectors (bottom). Circled points are support vectors.

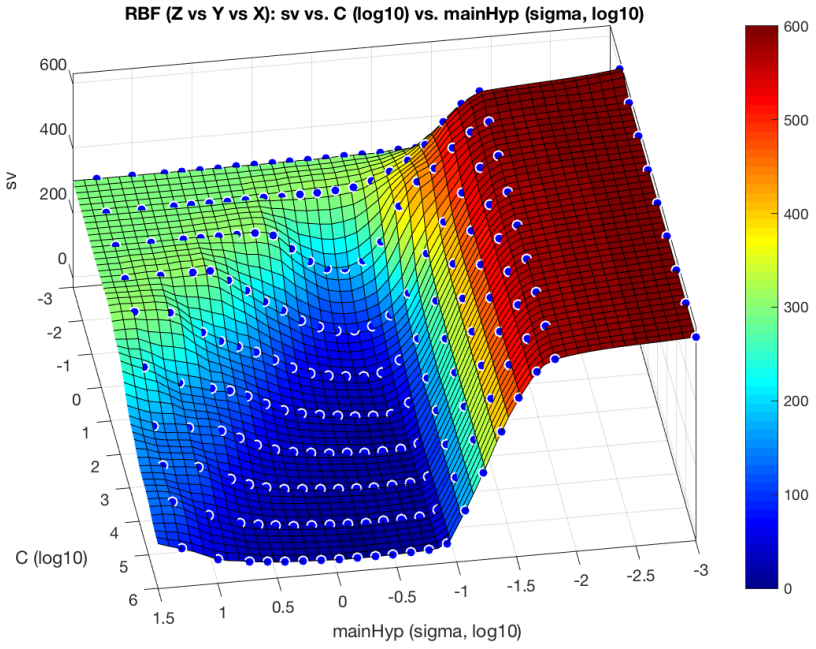

Figure 6.16: For toy problem #2: the Gaussian RBF kernel has a substantive range of hyperparameters which yield few support vectors.

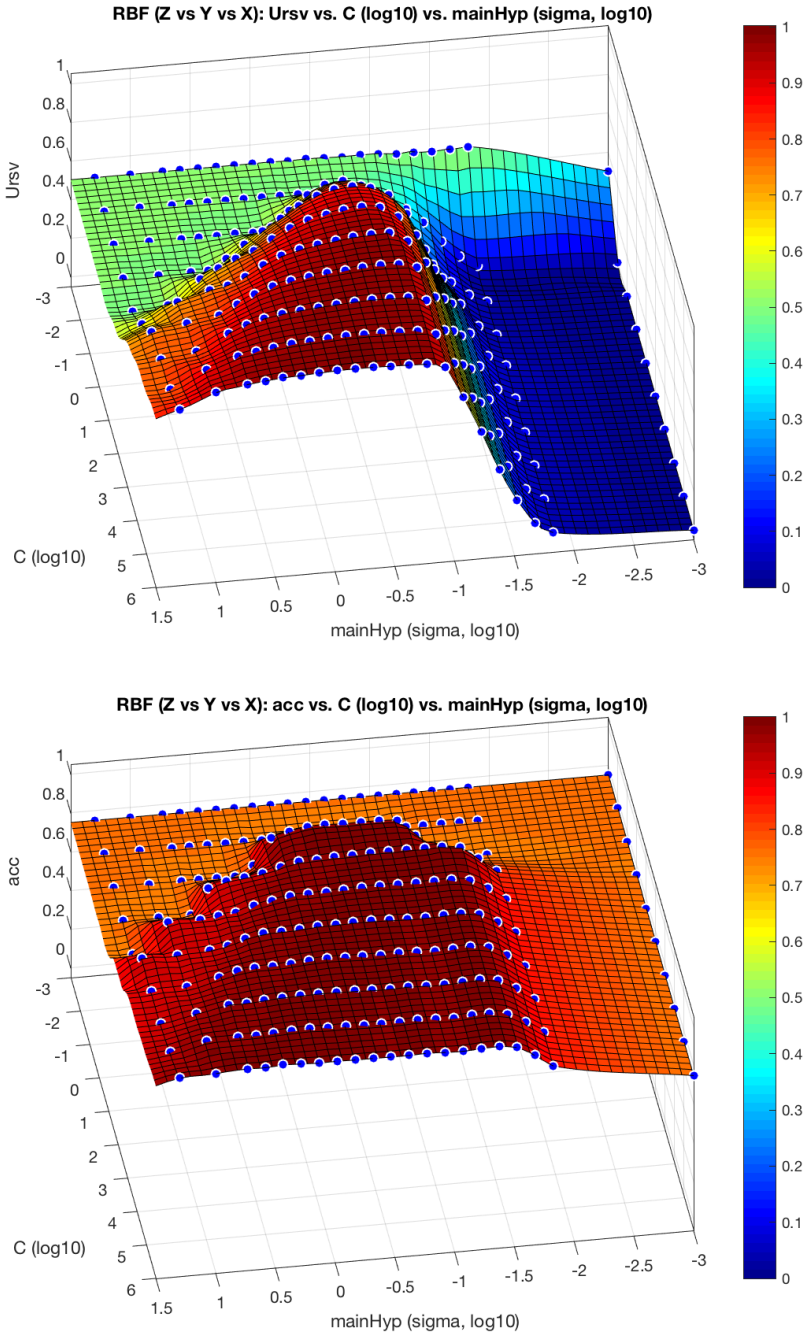

Figure 6.17: For toy problem #2: the Gaussian RBF kernel has a substantive region of hyperparameters which yield high model interpretability U_{sv} (top) and high accuracy a or acc (top), which confirms hypothesis #2. At $U_{sv} \approx 96.2\%$ and 79% (top), $a = 99.6\%$ and 97.1% (bottom), which when compared with the linear kernel which confirms hypothesis #1. For this kernel which fits the data well, there is **no** trade-off between model interpretability (re support vectors) and accuracy.

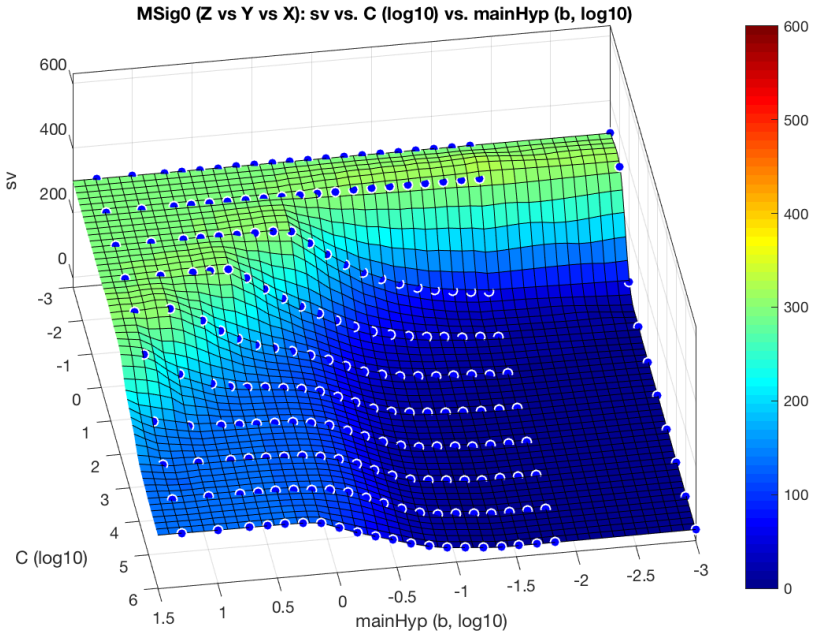

Figure 6.18: For toy problem #2: the Gaussian RBF kernel has a substantive range of hyperparameters which yield few support vectors.

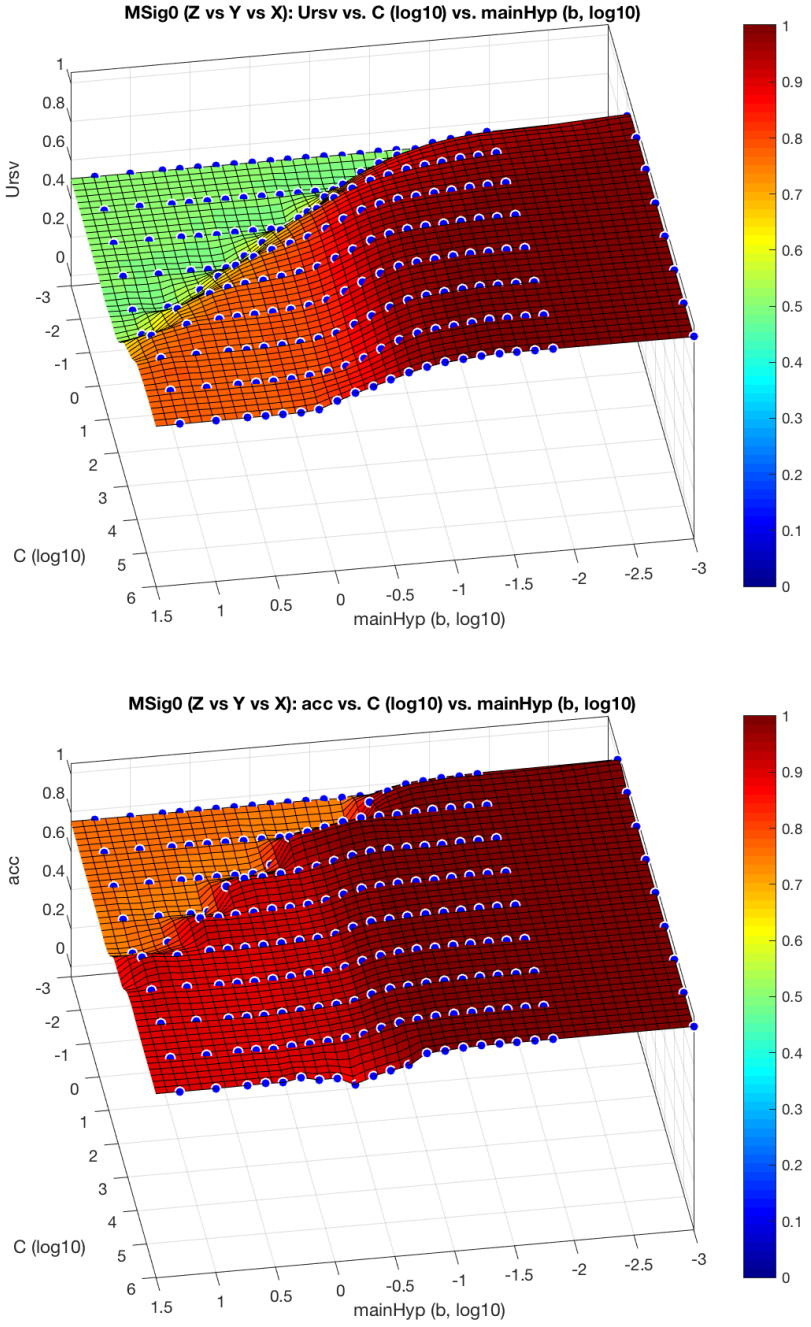

Figure 6.19: For toy problem #2: The Mercer sigmoid (MSig) kernel has a substantive range of hyperparameters which yield high model interpretability U_{sv} (top) and high accuracy a or acc (bottom), which confirms hypothesis #2. For approximately the same U_{sv} as the Gaussian RBF kernel, the Mercer sigmoid kernel's $U_{sv} \approx 96.9\%$ and 79.4% (top) corresponds to $a = 99.7\%$ and 91.8% (bottom), which when compared with the linear kernel confirms hypothesis #1. For this kernel which fits the data well, there is **no** trade-off between model interpretability (re support vectors) and accuracy.

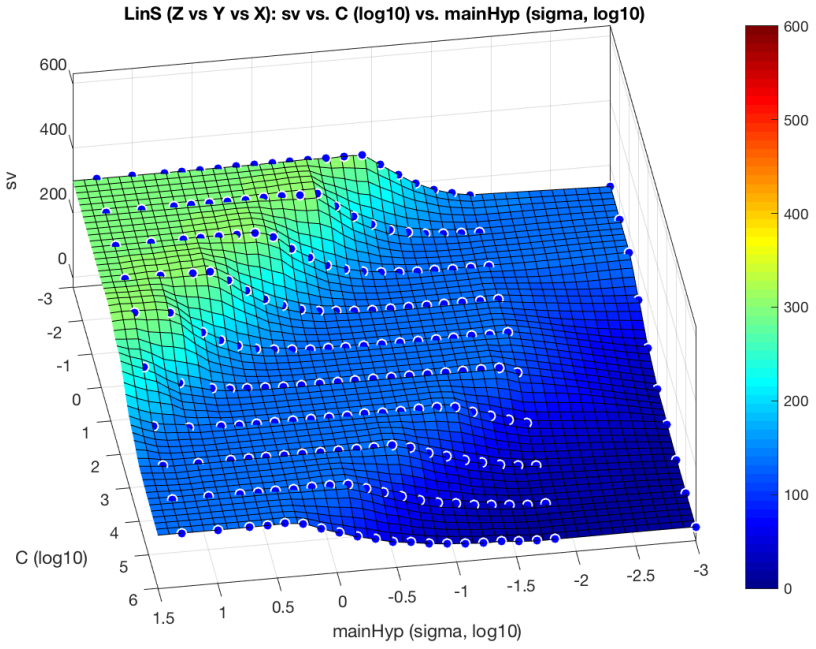

Figure 6.20: For toy problem #2: the linear kernel has three ranges of hyperparameters or plateaus.

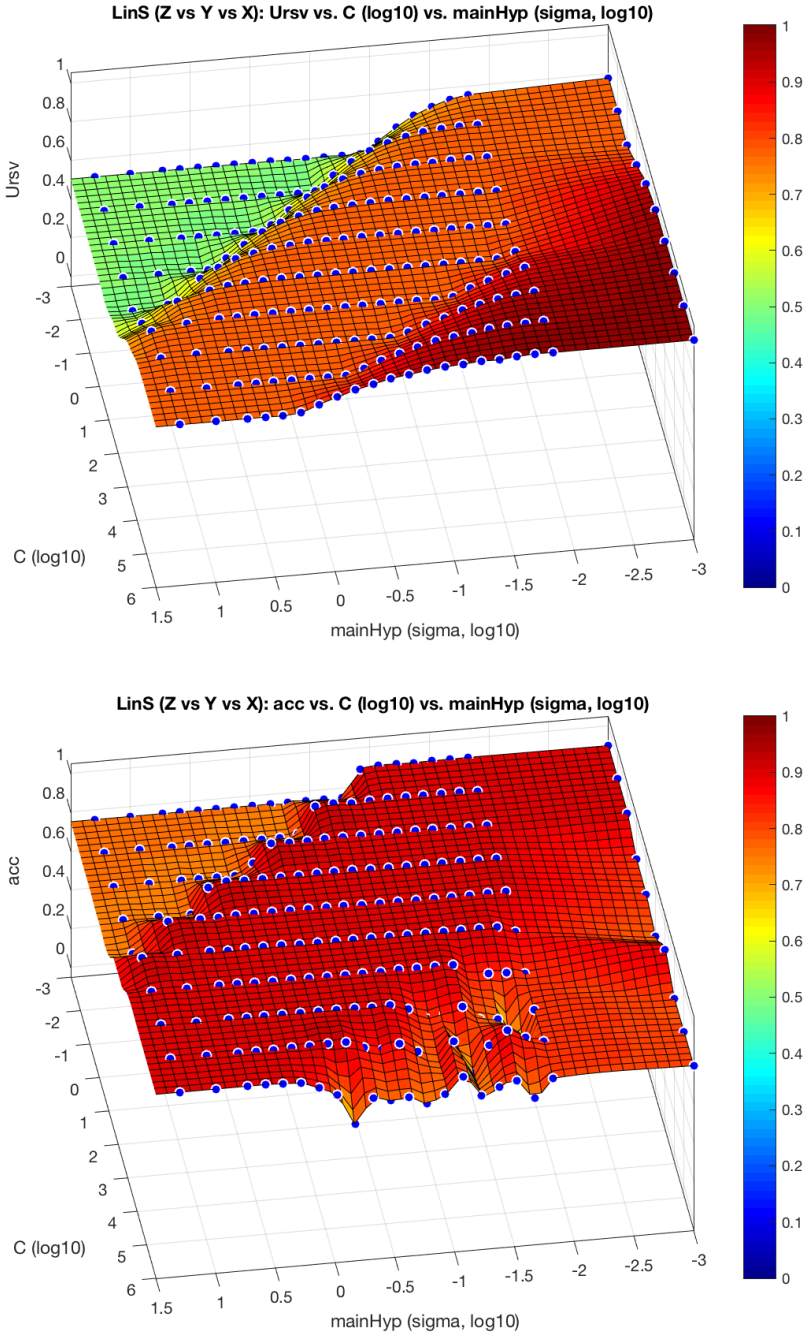

Figure 6.21: For toy problem #2: For approximately the same U_{sv} as the Gaussian RBF kernel, the linear kernel's model interpretability $U_{sv} \approx 96.3\%$ and 79.2% (top) corresponds to accuracy $a = 87.2\%$ and 89.1% (bottom). For this kernel which **does not** fit the data well, there **is** a **small** trade-off between model interpretability (re support vectors) and accuracy.

6.8 Summary

I developed and validated measures for inherent model interpretability to enable automatic model selection and ongoing research. Two measures are recommended: my proposed kernel transparency measure $\check{U}_{\partial}$ which is an inexpensive prior measure, and a posterior measure based on support vectors U_{sv}. Three other measures, U^*_{rdeT}, U^*_{rdeL} and $U_{H\text{st}}$ were found to be of limited use but may be further validated by future work.

I also contributed ideas as a foundation for these measures: the concept of inherent model interpretability, a general measure, a simplicity of sensitivity measure, and measurement of interpretability at different points in the learning process, i.e., via prior, initial and posterior models.

I applied my measure to model selection and demonstrated that choosing a model based on a sorted list of accuracy alone can result in models with substantively less inherent model interpretability despite the consistent availability of models with high accuracy and high interpretability in multiple data sets. The notion of a trade-off between accuracy and interpretability does not hold for these data sets, since in 9 of 10 instances there is minimal to no sacrifice in either measure.

Chapter 7

Conclusions and future work

Einstein said, "everything should be made as simple as possible, but no simpler" [172], which (ironically) is a simplified and incomplete version of Occam's razor or parsimony [244]. Parsimony requires a sufficient or best explanation first, i.e., most accurate model(s), then among such equivalent models, the simplest as the most likely, i.e., the most interpretable and transparent. This thesis fulfills that philosophical and practical goal, achieving equivalent (or better) accuracy with greater transparency, and equivalent or greater interpretability.

This is achieved for data with atomic data types—reals, nominals, binary, ordinals, integers and presence-only binary data—as differentiated from complex data types (images, text, spectra or time series / waveforms).

The contributions of this thesis, are as follows, for binary classification of health care data with atomic data types using support vector machines:

1. There is no accuracy versus transparency trade-off between kernels (among all kernels tested)
   (a) Explicit Mercer kernels are more transparent, because
      - Feature significance is meaningful, innate views are transparent and summarial, and non-innate views of the input space and feature space are made available. Future work (discussed in the next section) with human experiments is required to measure the impact.
   (b) Explicit Mercer kernels are at least as accurate, because

- Explicit Mercer kernels of rank n are sufficient, and implicit kernels of rank N are not necessary. In fact, some of the explicit Mercer kernels, e.g., the orthant insensitive sigmoid variant Gaussian constrained (OISVgc) kernel, are more accurate than common kernels with statistical significance. While a small percentage gain in accuracy for binary classification may only affect a small percentage of patients, my goal was to be at least as accurate while achieving greater transparency—which was achieved.
- Although outside the scope of my thesis, classification of complex data in the form of image pixel/coordinate data, performed by others and myself, show that explicit Mercer kernels are slightly inferior in accuracy on some data sets, which may be an appropriate tradeoff for the sake of greater transparency and inherent model interpretability as I define it. Future work can explore this.

2. There is no accuracy versus inherent model interpretability trade-off within or between kernels (for the Gaussian RBF and Mercer sigmoid kernels), using the support-vector based measure

   (a) There is no trade-off within kernels because there is no **overall** negative linear trend or negative exponential trend in each kernel's plot

   (b) There is no trade-off between kernels, from visual inspection of plots, because the Gaussian RBF and Mercer sigmoid kernels achieve similar accuracy and similar but slightly different interpretability.

   (c) Future work (discussed in the next section) is required to confirm what may appear to be a negative linear trend in one small region of the accuracy versus interpretability plots.

3. I proposed new kernels based on a new kernel class (explicit Mercer kernels) which were derived for, designed for and matched to specific atomic data types and distributions. Kernel requirements and gaps (some new) were identified regarding data, similarity, classification and transparency—and these requirements and gaps were addressed by my new kernels. My approach and framework provides improved rationale and measurably improved transparency. Philosophically my approach hybridizes the two cultures Breiman [39] referred to: the data modelling culture in statistics and the algorithmic modelling culture in machine learning.

   (a) I also clarified the advantages of my framework, approach and kernels over feature shaping, and I clarified the appropriateness of explicit kernels in the context of the kernel trick and generalized additive models (as well as kernel rank and complexity).

4. I provided improvements to an innate view (plot) of SVM results and demonstrated the improved interpretability of my proposed new kernels in the improved innate view. I also provided new measures of feature significance in SVM.

5. I developed new concepts and quantitative measures of transparency for justification of model selection prior to testing, i.e., with prior models, in SVM. I also developed new concepts and measures of inherent model interpretability for initial and posterior model selection, with some measures validated in general and some measures only suggested for limited use at this time. With these measures kernels can be selected quantitatively, i.e., automatically based on accuracy, transparency and inherent model interpretability.

In summary, my thesis takes a different yet fruitful approach to kernels for binary classification with atomic data types in SVM. My thesis highlights the characteristics and shortcomings of existing kernels while proposing new kernels that enable greater transparency and interpretability to meet requirements in health care.

That said, one limitation of my thesis pertains to the focus on accuracy, transparency and interpretability as key requirements to optimize, without observations of calibration (e.g., goodness of fit in each subgroup). Calibration requires a clinician's specification of clinically meaningful groups/subgroups and their granularity. Future work (discussed in the next section) may consider calibration.

On a similar note, external validation (or transferability) is also a desired requirement, but was not possible with the benchmark data sets used. Future work (discussed in the next section) can select multi-population data sets for that criterion, e.g., Canadian versus American kidney transplant data, or multi-centre data within each.

7.1 Future work

Arising from this thesis are various items for future work, as follows.

Future work is required to confirm what may appear to be a negative linear trend in one region of accuracy versus interpretability plots—the top-right portion of the bounding box, at the optimal (pareto) front of accuracy versus interpretability where the two objectives are somewhat balanced.

My thesis focuses on data sets with atomic data types. Work outside the scope of my thesis, by myself [45] and others (Appendix Section A.9), indicates that complex data

types—image pixel or coordinate data in particular—are not as accurately predicted by the explicit Mercer kernels I propose, as compared to the Gaussian RBF kernel. It would be of benefit to confirm the difference, its extent and its cause.

Future work with human experiments would be required to measure the impact of this work on (1) human (non-inherent) model interpretability and understanding, (2) decision-making from accuracy, transparency, interpretability (and calibration), (3) explanations and (4) changes in trust for, and adoption of machine learning methods, such as SVM. For these experiments, the participants would need to learn and become proficient with two aspects: (a) the interpretation of SVM results from its innate transparency and (b) the interpretation of SVM results from the additional transparency and interpretability of my proposed kernels (or any kernel in the class of explicit Mercer kernels).

Two logical next steps from my work, which pertain to the practice of machine learning with SVMs are: to produce a user guide and to publish reference implementations of my proposed kernels. Improving the practice and accuracy of SVM with either or both of these measures, improves research with SVMs. The user guide in particular is also necessary for the participants in the aforementioned human experiments to learn the required skills.

Future work may consider calibration [250] (e.g., goodness of fit in each subgroup) based on subgroups specified by a clinician which are validated as clinically meaningful. Such findings may then also be more fully integrated with considerations for model selection along with accuracy, transparency and interpretability. On a similar note, external validation [250] (or transferability) may also be addressed by selecting data sets which contain multi-population data, such as Canadian (CoRETRIS) [75] versus American (SRTR) [184, 114] kidney transplant data, or multi-centre data within each.

Future work is required to investigate and confirm that the effect of each element of kernel design achieves its intended objective, in full isolation to avoid confounding from other possible effects.

Future work may examine the following question. For optimal accuracy is it sufficient to use a uniform Gaussian kernel or a uniform Mercer sigmoid, i.e., with a single width, or with a single width and shift, respectively, for all dimensions? Or should one use the non-uniform (generalized) versions of those kernels, i.e., with different hyperparameters for each input feature? Testing was not conducted with the non-uniform Mercer sigmoid kernel that was defined herein.

Future work may explore and validate my observation that asymmetric match weighting also applies to zero-coded missing values. That is, in binary classification, assuming the

class boundary is at the origin, values near the origin are less certain and therefore giving them less weight in accordance with their uncertainty is appropriate. Similarly, a zero-coded missing value is completely uncertain and assuming a kernel is aligned or shifted such that the origin has zero or very low weight then the missing value is accordingly given little to no weight.

Future work may test the truncated Gaussian RBF kernel [278] (an explicit Mercer kernel which is finite) to observe its accuracy and interpretability with atomic data types and complex (e.g., image) data types. In the literature the kernel was tested on a single pregnancy data set with simple data types and the results seem to support my conclusions about the sufficiency of explicit Mercer kernels for atomic data types. I found this particular kernel too late in my experimental process to include it.

Related to the truncated Gaussian RBF kernel [278], I developed and sought the concept of a truncated Gaussian early in my work, and based on Cotter *et al.* [55], I implemented and tested a kernel, but the results were not as accurate as I expected them to be. I requested but did not obtain a reference implementation from Cotter *et al.* to rule out errata and/or misinpretations. I subsequently created and tested, a Gaussian derivative kernel (Section 4.8.3) as a non-stationary variation of a Gaussian based on explicit Mercer kernels, using the most relevant of its infinite features for covariance—the first derivative.

Future work could examine the effectiveness of composite kernels as a next step. I proposed the OMB composite kernel (Section 4.8.4), however this is just an exemplar, the beginning for such kernels designed to deal with multiple data types, and partitioning the input data space.

Given that kernels are used beyond classification, for regression and dimensionality reduction, what is the effect of my kernels in these areas? Are they useful? My testing with kernel supervised principal components analysis (KSPCA) ([16]) for dimension reduction prior to binary classification (results not shown) reveal that better accuracy is achieved by using my proposed kernels in KSPCA (e.g., the OISVgc kernel in Section 4.8.2) versus other kernels (e.g., the Gaussian RBF kernel) with atomic data types in kidney transplant data.

Future work may compare SVM with my proposed kernels to other state-of-the-art classification methods. It does not bear upon this thesis, because the present work must show the effect of kernels isolated from the effect of methods. My proposed kernels may be used in other state-of-the-art methods such as kernel ridge regression and kernel supervised principal components analysis.

Appendix A

Supplemental details

The sections in this appendix are for the most part not related to each other. Rather than burden the background further, the following text and concepts which may supplement the readers understanding are too long for a glossary and if included in the body of the thesis would impair its readability. Hence they are included here, as standalone definitional text in most cases, or supplementary analyses in other cases.

A.1 Statistical methods

Classical/traditional statistical methods expect two major tasks to be performed: data analysis and formal inference [62]. Data analysis identifies plausible parametric and semi-parametric models of data by examining the data, the processes that generated the data, summary statistics (e.g., mean, standard deviation), outliers and plots (e.g., of the data, distribution and diagnostics) [62]; and it may also test the goodness-of-fit for candidate models in order to narrow down the list. Formal inference then seeks to determine which of the candidate models best reflects the true state of nature [62].

Based on literature, the following classifiers are **common** classical/traditional statistical methods used in health care: logistic regression, naive Bayes, linear discriminant analysis or Fisher's linear discriminant, quadratic discriminant analysis and Bayesian (or belief) networks.

Alternative or less common statistical methods, excluding machine learning methods, are: regularized logistic regression, kernel density based classifiers [2], Bayes classifier or maximum a posteriori (MAP) classifier [24], generalized additive model (GAM) technique [115], hybrid (machine learning/statistics) classifiers, kernel ridge regression classifier.

A.2 Instance and rule-based learning methods

Based on literature, the following classifiers are **common** classical/traditional instance and rule-based methods used in health care: k-nearest neighbours [293], decision trees (including C4.5, J48, CART) [293] and fuzzy logic.

Alternative or less common instance and rule-based methods, excluding machine learning techniques, are: evolutionary algorithms (EA), genetic algorithms (GA), artificial immune systems, partial decision trees, one-level decision trees.

A.3 Feature extraction, dimension reduction and visualization

Feature extraction, dimension reduction and/or visualization may be achieved with principal components analysis (PCA) or a variety of other similar techniques. I summarize a few methods below and then provide a comprehensive list of such techniques. This section (unlike most other sections in the thesis) is intended for readers familiar with this subject.

PCA [127] is the most well known method. It decomposes a matrix of data into **orthogonal** components, finding the first component as the direction within data with maximum variance, and then an orthogonal direction that is the next maximum in variance, and so on. Differently from PCA, independent component analysis (ICA) [119, 182] decomposes a matrix into **independent** components [140], solving what is known as the cocktail party problem. Independent subspace analysis (ISA) [48] is similar to ICA but solves a problem known as conversation clustering, by clustering components together into independent groups, one for each conversation.

Maximum variance unfolding (MVU) [287] is another decomposition technique. The idea of MVU is that data may be organized in less dimensions than they first appear. If I consider data like "beads on a necklace" where the necklace is in a heap, bunched up together in three dimensions, then I only need to pull the necklace taut to see the one-dimensional arrangement of the beads [287]. That is, three dimensional data may lie on a one dimensional manifold or subspace. Similarly I can consider data like "beads on a lattice" to find a two dimensional manifold or subspace.

A comprehensive list of methods for feature extraction, dimension reduction and visualization is provided in several parts below, omitting domain and task specific methods.

These include the following methods for dimension reduction and visualization: t-distributed stochastic neighbourhood embedding (t-SNE) [175]; locally linear embedding (LLE) [218]; Isomap [259, 260] as a non-linear generalization of multi-dimensional scaling [96, 149, 150]; semi-definite embedding (SDE) [287]; Laplacian eigenmaps (LEM) [17]; action respecting embedding (ARE) [31]; stochastic neighbourhood embedding (SNE) [122]; and Sammon mapping [224].

These also include a change of basis, or transformation, to a linear space or manifold, such as: factor analysis (FA) [261]; principal components analysis (PCA) [127] as a special case of FA according to Gharamani [95] and also called Partial least squares (PLS) [92, 195]; independent component analysis (ICA) [32, 129] as a generalization of FA according to Gharamani [95]; and multidimensional scaling (MDS) [150, 264].

They also include a change of basis to non-linear manifolds, such as: Fourier transforms; wavelet transforms; Gabor transforms; Fourier descriptors; kernel principal components analysis (KPCA) [228] which according to Ghodsi [96] generalizes PCA, FA, ICA, MDS, Isomap, LLE, LEM and SDE; and kernel supervised principal components analysis (KSPCA) [16] as a supervised version of KPCA.

Finally, these also include other changes of basis: non-negative matrix factorization (NMF) [158, 159]; non-negative tensor factorization (NTF) [43]; probabilistic latent component analysis (PLCA) [237, 236]; probabilistic latent semantic analysis (PLSA) [124]; independent subspace analysis (ISA) [48]; zero-phase component analysis (ZCA) as the opposite of PCA; vector quantization (VQ) [104]; whitening transforms [130] and random projection [23].

In addition to the above, there are methods specific to particular domains and types of data—i.e., domains/types such as: images (e.g., SIFT [173], Zernicke moments [258]), time-series data and waveforms (e.g., filters [239]), text (e.g., tagging [246]), spectra (e.g., trace quantitative analysis [33]), etc.

A.4 Simple data types in ISO 21090

ISO 21090 [134] is an implementation of abstract data types based on prior standards from ISO, OMG's UML, Health Level 7's Version 3, CEN 13606 and openEHR. According to the National Cancer Institute it is "a culmination of a large-scale joint effort among standards bodies such as HL7 and ISO, and has been reviewed by experts in the field." [131]. I

provide a diagram of the simple data types within ISO 21090 for reference's sake (Figure A.1).

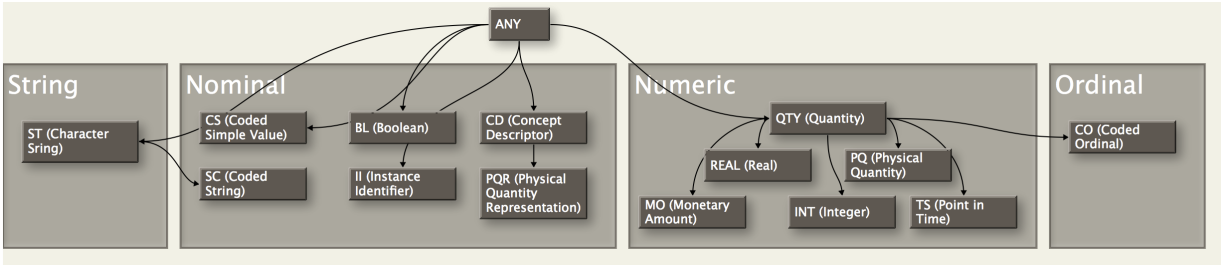

Figure A.1: Overview of ISO 21090 data types for health care

A.5 Complex data types or documents

In contrast to simple data types, complex data types (Figure A.2) , also called documents, are collections of simple data types in a structure, matrix, grammar or ontology—e.g., text, images, video and audio. This category is supported by the literature on data types [123]. These types have domain-specific needs, tasks, measures of performance and specialized kernels/models associated with them.

I depict complex data types or documents in the ISO 21090 standard (Figure A.3) and I provide other examples of images and matrices as complex data types (Table A.1).

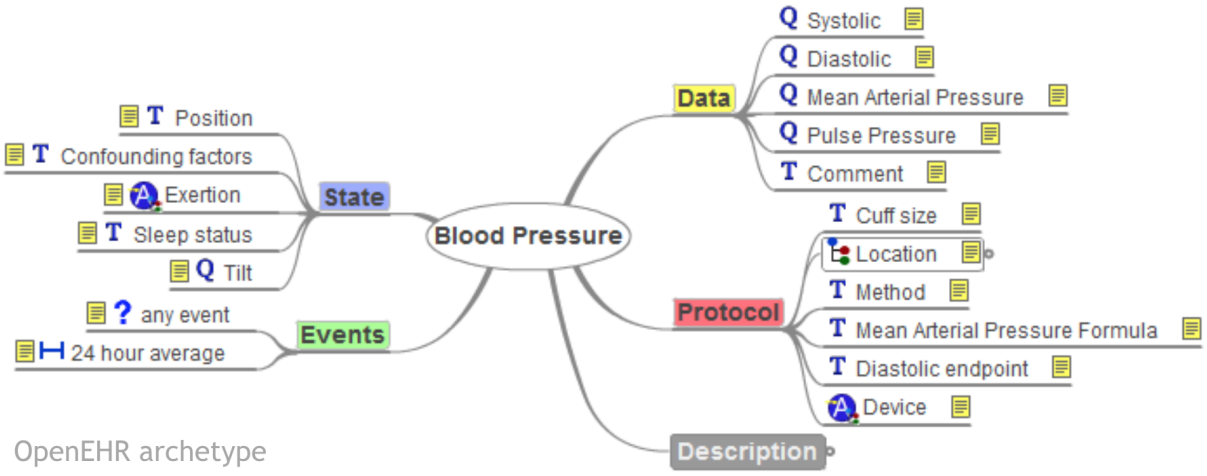

Figure A.2: Complex data types have single or multiple values with an underlying structure, grammar or ontology.

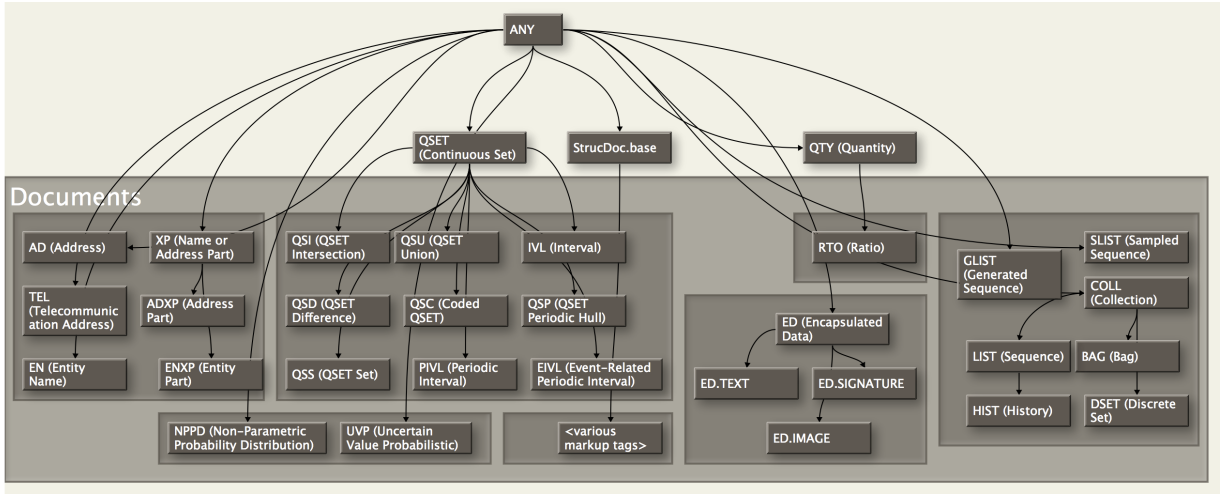

Figure A.3: Overview of ISO 21090 document types for health care

Table A.1: More examples of complex data types in health care

Complex data type or document		Illustrative examples	Independent (structure) dimension	Dependent (value) dimension
Table		Incidence of cancer by region and gender	n-D	n-D
	Sequence	DNA: GATTACA; Peptides: DLGEEHFK	1D: count	1D: nominal
	Waveform or Spectrum	ECG, EKG, EMG, reference signals	1D	1D
		Mass spectrum (MS, MS-MS)	1D	1D
	Map	Brain atlas; Flu activity map; POS location	2D	1D
	Image	Medical photograph (incl. epiluminescence)	2D: pixel	3D: RGB
		Digital microscopy/pathology	2D	3D: RGB
		Grayscale radiograph: X-ray, CT, MRI	2D,3D:voxel	1D: intensity
		Colour radiograph: PET, SPECT, fMRI	2D,3D:voxel	3D: RGB
	Video	Video (activity recognition, fall detection)	3D: pixel, t	1D,3D
Graph		XML, HTML, OID Tree	n/a	
	Semantic standard	HL7v3 CDA or messaging, ICD-09 ontology SNOMED-CT ontology	n/a	
Delimited Document		HL7v2 messaging	n/a	
Structured Document		Text document with standard sections	n/a	
Freeform Document		Text document or field	n/a	

A.6 Meta data

Binary meta data include (*per feature): outliers-trimmed*[1], top-coded*, bottom-coded*, non-negative* (e.g., $\mathbb{R}^+$ or counts $\mathbb{N}$), normalized, centered (including binary), positive-match*, complete, complete-case-analysis, imputed*, imputation-constrained* (i.e., to valid range), binary-imputed-as-continuous*, missing-coded*, binned*, discretized*, rounded*, de-identified, feature-dominant (i.e., $n \gg N$), instance-dominant (i.e., $N \gg n$), censored, imbalanced (e.g., $\frac{N^+}{N^-} > 2$ or $\frac{N^-}{N^+} > 2$), small events (n.b., this term is not well defined, but perhaps I can refer to events per variable $N^+ < 10 \cdot n$, or $N^+ \ll N$).

Four nominal meta data are (*per feature): de-identification, missingness*, normalization and censoring. De-identification has values: case-deletion, binned, k-anonymized, l-diversified, t-closeness. Missingness has values: MAR, MCAR, MNAR (Appendix Section A.8). Normalization has values: standardization, standardization of the third standard deviation, min/max normalization. Censoring has values: left, right or interval.

Two numeric meta data (*per feature) are: missing code*, class noise* (i.e., Bayes error of the marginal density estimated with kernel density estimation).

A.7 A review of literature applying SVM to health care

I review 22 academic papers which conduct experiments using SVM in health care to observe the rationale given for using common kernels with atomic data types (4.2). My review includes 8 papers on SVM for health care in general and 14 papers on SVM for melanoma detection. The review is summarized below.

- 16 papers do not provide any rationale (or a reference which could infer a rationale) for the kernels they use (and 11 of these papers use a single kernel):
  - 4 do not indicate what kernel is used [108, 185, 303, 305]
  - 5 use a Gaussian RBF kernel only, without rationale [91, 177, 190, 283, 292]
  - 1 uses a linear kernel only, without rationale [144]
  - 1 uses a polynomial kernel only, without rationale [298]
  - 1 uses polynomial and Gaussian RBF kernels, without rationale [141]

[1] prior to normalization

  - 1 uses a polynomial kernel, a Gaussian RBF kernel, and two other kernels (Generalized Gaussian and Chi-Squared), without rationale[265]
  - 3 use a linear, polynomial and Gaussian kernel, without rationale[139, 176, 207]

- 2 papers do not state any rationale, but their citations could be construed as providing rationale, based on precedent and abstract theory respectively.
  - 1 uses a linear kernel only [78], and cites several papers on text classification in which the linear kernel is a precedent.
  - 1 uses polynomial kernels only [253], and cites Burges [40] who discusses the Gaussian RBF, polynomial and sigmoid kernels. The only rationale Burges provides for choosing a kernel, is that higher VC dimension indicates the potential for greater complexity in the class boundary.

- 4 papers provide some rationale—based on precedent, popularity, and in two cases theory with anecdotes.
  - 1 refers to the precedent of linear, polynomial and Gaussian RBF kernels [252] as "reasonable kernels" citing Hsu et al [128], although Hsu et al does not discuss the polynomial kernel.
  - 1 refers to popularity and precedents [154] for the polynomial, Gaussian RBF, Generalized Gaussian RBF and Chi-Squared kernels as rationale. It reasons that the Gaussian RBF and polynomial kernels are popular; and that the two other kernels are newly proposed for visual recognition by Chappelle.
  - 1 uses a 5th order polynomial kernel [42] and mentions that various kernels may be applied with reference to Burges [40] who discusses the polynomial, Gaussian RBF, and sigmoid kernels. Neither the authors nor the cited resource provide rationale for choosing the polynomial kernel over others, or choosing the 5th order specifically.
  - 1 uses a polynomial, sigmoid, Gaussian RBF and k-MOD-decreasing kernel, with rationale [97] by reference to supplemental material which references Ben-Hur et al [18] in turn.

A.8 Converting atomic data types to reals

If I select a machine learning method and model that handles continuous values, then I can treat features of any atomic data type as continuous. This treatment requires three steps:

1. Treat missing data.

   (a) Impute missing data for reals, integers, datetimes, dates and ordinals, using whichever method meets requirements—e.g., multiple imputation with Monte Carlo Markov chain (MCMC) or expectation maximization (EM) are common; see Appendix A.10 for further information.

   (b) Impute missing data for nominals using the mode, i.e., the most frequent level.

   (c) Impute missing binary data with a method that will produce continuous values and which is appropriate for binary distributions. I refer to the output as continuously-imputed binary data.

2. Convert nominals to binary indicators, one for each level.

3. Center and normalize data

   (a) For continuously-imputed binary data, bottom-code and top-code the data to the limits, then min-max normalize the data to the range [-1,+1] for SVM or [0,1] for neural networks and logistic regression.

   (b) For binary data, min-max normalize the data to the set {-1,+1} for SVM or {0,1} for neural networks and logistic regression. This data will be treated as reals by the methods/models, but {-1,+1} makes more sensible use of the symmetric kernel geometry in SVM than {0,1}.

   (c) For all other data types, center and normalize each feature using z-score normalization (or scalar variations based on 2 or 3 sigma instead of 1 sigma).

Now all of the data are ready to be treated as reals by the methods/models.

A.9 Accuracy of Gaussian RBF versus Mercer sigmoid in literature

I extract the results (Table A.2) for mean accuracy of the Gaussian RBF kernel compared with the Mercer sigmoid kernel from Yamada *et al.* [295], whose paper focuses on particle swarm optimization (PSO) for hyperparameter optimization (Figure 1.4). They present results for SVM with standard PSO (S-PSO), their proposed method (CFA-PSO) and a Bezier method (SEB).

n classification with twelve data sets (Table A.2) the Mercer sigmoid kernel (MSig) is better than the Gaussian RBF kernel with statistical significance in three, the same (statistically) in five, and worse (statistically) in four image data sets (with pixel or coordinate image data).

For data sets with atomic data types (Section 4.2) as the focus of my thesis, e.g., not image data, the Mercer sigmoid kernel is better. In general, in this Yamada et al's selection of data sets, the Mercer sigmoid kernel is useful for accuracy, and particularly useful for accuracy with transparency.

Table A.2: In classification with twelve data sets the Mercer sigmoid kernel (MSig) is better than the Gaussian RBF kernel with statistical significance in three, the same (statistically) in five, and worse (statistically) in four image data sets.

				Gaussian RBF Kernel					Mercer sigmoid kernel					Winner and	Statistical	Polynomial kernel				
	Image pixel / coord data	Mixed data types (hetero)	Features extracted from images	S-PSO	CFA-PSO	S-PSO	CFA-PSO	SEB	S-PSO	CFA-PSO	S-PSO	CFA-PSO	SEB	difference between MSig & RBF	significance of difference between MSig & RBF (2 st.dev)	S-PSO	CFA-PSO	S-PSO	CFA-PSO	SEB
				50	50	20	20		50	50	20	20				50	50	20	20	
Landsat Satellite	Y	integer	-	93.47	93.56	93.41	93.55	93.51	91.85	91.84	91.76	91.78	91	RBF 1.71	**Y (0.77)**	92.66	92.63	92.5	92.6	92.39
Svmguide1	-	real	-	94.95	95	95	94.99	94.94	96.01	95.9	95.82	95.8	94.91	MSig 1.01	**Y (0.82)**	95.2	95.17	95.06	95.01	95.04
Occupancy Detection	-	real	-	98.27	98.26	98.21	98.23	98.09	98.98	98.97	98.32	98.72	98.03	MSig 0.71	**Y (0.20)**	98.33	98.33	98.2	98.21	98
MAGIC Gamma Telescope	-	real	Y	80.82	80.78	80.75	80.77	80.82	81.55	81.62	81.15	81.19	80.58	MSig(2nd) 0.81	N (1.50)	81.68	81.61	81.51	81.47	81.5
Page Blocks Classification	-	real, integer	Y	93.29	93.39	93.25	93.37	93.31	93.96	93.86	93.09	93.67	93.06	MSig 0.57	N (1.04)	93.58	93.6	93.52	93.54	93.54
Statlog Shuttle	-	integer	-	97.38	97.3	97.27	97.2	97.12	97.73	96.19	96.26	95.09	95.06	MSig 0.35	N (0.94)	96.85	96.85	96.72	96.71	96.55
Wine Quality	-	real	-	79.5	79.44	79.39	79.38	79.43	79.71	79.5	79.25	79.25	79.29	MSig 0.21	N (0.75)	79.49	79.52	79.39	79.39	79.39
Credit	-	real, integer, binary, nominal	-	79.26	79.25	79.25	79.28	79.27	80.69	79.45	79.95	78.75	79.62	MSig 1.41	**Y (1.36)**	79.33	79.41	79.35	79.35	79.38
Pen Digit	Y	integer	-	95.48	95.51	95.35	95.48	95.44	87.6	87.52	87.28	87.25	85.24	RBF 7.91	**Y (1.43)**	95.15	95.18	95.17	95.11	95.01
Opt Digit	Y	integer	-	95.7	95.69	95.64	95.62	95.61	90.36	90.35	90.29	90.3	89.25	RBF 5.34	**Y (1.56)**	95	95.04	95.01	95.01	95.02
Letter	Y	integer	Y	72.66	73.67	72.33	73.32	73.64	70.36	70.28	70.04	70.07	69.62	RBF(2nd) 2.30	**Y (2.17)**	71.13	71.23	71.06	71.03	71.1
Usps	Y	integer	-	93.59*	93.59*	93.3	93.57	93.44	92.1	92.07	91.82	92.01	90.28	RBF 1.49	N (2.07)	93.03	93.05	92.99	93.01	92.89

A.10 Missing values and other characteristics of health care data

Data in health care have a few characteristics that must be accounted for in machine learning: data may have values that are missing, censored, correlated, top-coded or binned (for de-idenfication or anonymization). Handling correlated data can be achieved by performing attribute selection to select only uncorrelated (or less correlated) input variables [290], prior to learning with any algorithm.

Missing data (or values) are typically handled by a sequential method [107], where the missing values are resolved prior to the main learning step. Missing data may also be handled within the main learning step, which is known as a parallel method [107].

One approach to handling missing data is to delete the rows, instances or subjects that have missing values, and this is known as listwise deletion or complete case analysis [107]. This approach, while common, usually causes biased results and is only suitable when data is missing completely at random (MCAR) [69], e.g., a test tube is randomly broken — there is no reason behind why the data are missing.

In epidemiological studies however, it is far more common to have data that is missing at random (MAR) than it is to have MCAR data [69]. MAR data refers to a value in one explanatory variable that is missing whenever another explanatory variable has a certain value (or range of values) [69]. Consider, for example, MAR data from patients with chronic kidney disesase: some patients are on dialysis, because their kidneys have failed and thus I do not need to measure their kidney function; whereas for other patients, who are not on dialysis, I do measure their kidney function via creatinine level.

To handle missing data, I typically have to fill them in, i.e., impute them [69], using values present in existing data or estimated therefrom. Imputation methods range from simple to advanced: direct replacement, mean imputation, hot deck imputation [193], maximum likelihood estimation [174], expectation maximization, single imputation (i.e., multiple linear regression) and multiple imputation. For data which is MAR, simple methods (e.g., mean imputation) yield biased results whereas advanced methods (e.g., multiple imputation) produce unbiased results [69, 220]. The more sophisticated methods are computationally intensive [193]. While this matter is typically handled prior to the (main) learning step, I may also employ learning methods which can handle missing values (typically with effort).

For other characteristics such as censoring the reader is referred to the literature [250, 197].

A.11 Derivation details for the orthant sigmoid kernel

I start with the basic formula which vertically shifts the terms within a Mercer sigmoid by $c \in [-1, +1]$:

$$k_1(\mathbf{x},\, \mathbf{z}) = \sum_{i=1}^{p} \left(\left[\tanh\left(\frac{x_i - d}{b} \right) + c \right] \cdot \left[\tanh\left(\frac{z_i - d}{b} \right) + c \right] \right) \tag{A.1}$$

Within each of p dimensions:

For $c \geq 0$, the maximum height is in the positive match orthant $x, z = +1$ and the minimum height is in either one of the mismatch orthants, e.g., $x = 1, z = -1$. To normalize I find an expression for the range (maximum - minimum) in height and divide by that. Since tanh in (A.1) has a minimum and maximum of $\{-1, +1\}$ for $\{-1, +1\}$ respectively, i.e., $\tanh(x) = x$ at these values, the range in height is:

$$\begin{aligned}
\left| (x+c)(z+c)|_{x,z=+1} - (x+c)(z+c)|_{x=1,z=-1} \right| &= \left| (c+1)^2 - (c+1)(c-1) \right| \\
&= \left| c^2 + 2c + 1 - (c^2 - 1) \right| \\
&= |2c + 2| \\
&= 2\,|c + 1| \\
&= 2\,||c| + 1| \quad \text{since } c \geq 0 \\
&= 2\,(|c| + 1)
\end{aligned}$$

Similarly, for $c \leq 0$, the output range is as follows:

$$\begin{aligned}
\left| (x+c)(z+c)|_{x,z=-1} - (x+c)(z+c)|_{x=1,z=-1} \right| &= \left| (c-1)^2 - (c+1)(c-1) \right| \\
&= \left| c^2 - 2c + 1 - (c^2 - 1) \right| \\
&= |-2c + 2| \\
&= 2\,|-c + 1| \\
&= 2\,||-c| + 1| \quad \text{since } -c \geq 0 \\
&= 2\,||c| + 1| \\
&= 2\,(|c| + 1)
\end{aligned}$$

The maximum output range in each dimension is the same in both cases. Therefore, for any $c \in [-1, +1]$, I normalize by the common factor, $2\,(|c| + 1)$, to achieve a range of unit

size:

$$k_2(\mathbf{x},\ \mathbf{z}) = \frac{1}{2(|c|+1)} \sum_{i=1}^{p} \left(\left[\tanh\left(\frac{x_i - d}{b} \right) + c \right] \cdot \left[\tanh\left(\frac{z_i - d}{b} \right) + c \right] \right)$$

I can also vertically shift the output range so that the minimum, which occurs in a mismatch orthant, is at 0:

$$\begin{aligned} |(x+c)(z+c)| &= |(c+1)(c-1)| \\ &= \left|c^2 - 1\right| \\ &= -\left(c^2 - 1\right) \quad \text{since } \left(c^2 - 1\right) \leq 0 \text{ for } c \in [-1, +1] \\ &= 1 - c^2 \end{aligned}$$

Resulting in the definition of an orthant sigmoid kernel with an output range of $[0, 1]$ in each dimension:

$$k_{\text{OSig}}(\mathbf{x},\ \mathbf{z}) = \frac{1}{2(|c|+1)} \sum_{i=1}^{p} \left(\left[\tanh\left(\frac{x_i - d}{b} \right) + c \right] \cdot \left[\tanh\left(\frac{z_i - d}{b} \right) + c \right] + \left(1 - c^2\right) \right)$$

Glossary

Accuracy: The proportion of correct or true results in the system. Accuracy = (TP + TN) / (TP + TN + FP + FN) = 1 - Misclassification Rate.

Alpha α: See type I error.

Attribute: See feature.

Balanced Accuracy: The geometric mean (or Gmean) of sensitivity and specificity. Balanced Accuracy = $\sqrt{\text{Sensitivity} \times \text{Specificity}}$

Basis function: See feature map.

Beta β: See type II error.

Bilinear: A function of two variables is bilinear if it is linear in both variables, e.g., the linear kernel. Please refer to Wikipedia or any standard text on the definition of a bilinear function, for functions of two variables.

Calibration: The agreement between the frequency of observed classes (or outcomes) and the frequency of predicted classes (or outcomes), within subgroups or time intervals, or for different samples[250].

Covariate: See feature.

Data matrix: An $N \times n$ matrix of data with N instances, each with n features.

Dependent variable: See target.

Discrimination: How well a classifier or prediction model discriminates between data of different classes — i.e., how well it classifies them[250]. There are various measures of discrimination such as accuracy, sensitivity, specificity, etc. Discrimination is often accompanied by calibration[250].

Event: A positive target value (outcome). The outcome of interest.

Explanatory variable: See feature.

Feature matrix: See data matrix.

F-Score: PPV $\times$ Sensitivity (or Precision $\times$ Recall)

False discovery rate: Also called q-Value. The chance of not having the condition among those that test positive. The chance of satisfying the null hypothesis among those that reject the null hypothesis. q = FP / (FP + TP) = 1 - PPV.

False negative (FN): The individual has the condition but tests negative for the condition. The individual does not satisfy the null hypothesis but the test accepts the null hypothesis.

False omission rate: The chance of having the condition among those that test negative. The chance of not satisfying the null hypothesis among those that accept the null hypothesis. x = FN / (FN + TN) = 1 - NPV.

False positive (FP): The individual does not have the condition but tests positive for the condition. The individual satisfies the null hypothesis but the test rejects the null hypothesis.

Feature: Also called an explanatory variable, attribute, covariate or independent variable. Features are random variables which I hope are correlated with a target of prediction, so that I can use them to predict the target.

Feature map: Also called a basis function or kernel map. Tbd.

iff: if and only if

Independent variable: See feature.

Input: See data matrix.

Instance: An instance is each occurrence of an object, where a set of instances constitutes a sample, drawn as a subset from a population. Each draw is an instance, e.g., a patient (or their indicators and observations).

Kernel (in kernel methods): A kernel, in the context of kernel methods, refers to a similarity function that takes two inputs of any type (e.g., atomic data type or complex data type) and outputs a real-valued number that represents the similarity between the inputs,

where the function must meet other specific requirements. A kernel must be symmetric (i.e., switching the inputs yields the same output) and must be admissible for its purpose (e.g., p.d. or c.p.d. for SVM).

This definition is distinguished from 10 others: kernels in kernel density estimation (see below), kernels in image processing, kernels for "machine learning in the cloud" as in Kaggle, convolution kernels in convolution neural networks, kernelization in algorithms and complexity theory, kernels in operating systems and graphical processing unit (GPU) computation, kernels in geometry, kernels in set theory, kernel category theory and kernels in linear algebra.

Kernel (in kernel density estimation): If I want to estimate the probability density function (p.d.f) for a feature from a set of data then I use kernel density estimation (KDE) to do so, where the basic idea of KDE is that the p.d.f. is a landscape made up as a sum of hills at each point in the data set, such that overlapping hills can lead to mountains and plateaus.

Kernel map: See feature map.

Kernel methods: Kernel methods are methods in statistics, machine learning or otherwise which use kernels, such as: support vector machines, support vector regression, kernel logistic regression, kernel k nearest neighbour, kernel fisher discriminant, gaussian processes, kernel principal components analysis, kernel supervised principal components analysis and import vector machines,.

Linear: Please refer to Wikipedia or any standard text on definitions of a linear function, for functions of one-variable, or the definition of a bilinear function, for functions of two variables.

Misclassification rate: The proportion of incorrect or false results in the system. MR = (FP + FN) / (TP + TN + FP + FN) = 1 - Accuracy.

Negatives: A negative target value (or outcome) representing the (negative) class, in binary classification.

Negative predictive value (NPV): The chance of not having the condition among those that test negative. The chance of satisfying the null hypothesis among those that accept the null hypothesis. NPV = TN / (TN + FN) = 1 - False Omission Rate.

Observations: See instances.

Outcome: See target.

Output: See target.

p-Value: See Type I error.

Patient: A person that is under the care of a health care provider, for (or about) whom I wish to make predictions, estimations or decisions. An instance of data represents a patient's observations.

Positives: A positive target value (or outcome) representing the (positive) class of interest, in binary classification.

Positive predictive value (PPV): Also called precision. PPV = TP / (TP + FP) = 1 - False Discovery Rate. The chance of having the condition among those that test positive. The chance of not satisfying the null hypothesis among those that reject the null hypothesis.

Power: For statistical power, see sensitivity.

Precision: See positive predictive value.

Random variable: A random variable refers to a variable that I may measure, which takes on (or appears to take on) values randomly from its probability density/distribution function (for continuous/discrete values respectively). In this context it refers to a feature. See feature.

Response variable: See target.

Recall: See sensitivity.

Sample: A subset of a population. Instances (or observations) drawn from a population or distribution. See also, data matrix.

Sensitivity: Also called statistical power, recall or true positive rate. The chance of testing positive among those with the condition. The chance of rejecting the null hypothesis when/given the null hypothesis is false. $Sensitivity = {}^{TP}/_{(TP+FN)} = 1 - Error_{\text{TypeII}}$.

Specificity: Also called selectivity or true negative rate. The chance of testing negative among those without the condition. The chance of accepting the null hypothesis among those that satisfy the null hypothesis. $Specificity = {}^{TN}/_{(TN+FP)} = 1 - Error_{\text{TypeI}}$.

Statistical power: See sensitivity.

Support vector machine (SVM): A classification method which is variously referred to as a machine learning method, an instance-based method, a maximum-margin technique and a kernel method. By default, I refer to soft-margin SVM, which can handle errors resulting from data which are not linearly separable; and in that case SVM has two optimization criteria: maximum margin and minimum error. By default, I also refer to SVM with kernels (as a kernel method)—not just linear SVM. See section 2.2 for more information.

Support vector regression (SVR): A regression method which is variously referred to as a machine learning method and an instance-based method. SVR is insensitive to errors within a tube surrounding the estimate.

Target: Also called outcome or dependent variable (because its value depends on the prediction model and its input). The value being predicted, estimated or decided. A target may be continuous as in regression, or binary as in binary classification, or nominal (categorical) as in multi-class classification, or less commonly ordinal.

True negative (TN): The individual does not have the condition and tests negative for the condition. The individual satisfies the null hypothesis and the test accepts the null hypothesis.

True positive (TP): The individual has the condition and tests positive for the condition. The individual does not satisfy the null hypothesis and the test rejects the null hypothesis.

Type I error: Also called α (alpha), p-Value or false positive rate. The chance of testing positive among those without the condition. The chance of rejecting the null hypothesis among those that satisfy the null hypothesis. α = FP / (FP + TN) = 1 - Specificity.

Type II error: Also called β (beta) or false negative rate. The chance of testing negative among those with the condition. The chance of accepting the null hypothesis among those that do not satisfy the null hypothesis. β = FN / (FN + TP) = 1 - Sensitivity.

References

[1] Shigeo Abe. Training of support vector machines with mahalanobis kernels. In *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 571–576. Springer, 2005. 2.9.4

[2] John Aitchison and Colin GG Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976. A.1

[3] Enrique Alba, Jose Garcia-Nieto, Laetitia Jourdan, and El-Ghazali Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 284–290. IEEE, 2007. 4.4.3

[4] Shawkat Ali and Kate A Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, 2006. 2.3, 1

[5] Shawkat Ali and Kate A Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing*, 70(1-3):173–186, 2006. 1.1, 2.4, 1

[6] Uri Alon, Naama Barkai, Daniel A Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999. 5.1, 5.1.6

[7] Douglas G Altman and J Martin Bland. Statistics notes: Diagnostic tests 2: predictive values. *Bmj*, 309(6947):102, 1994. 8

[8] Hrishikesh Aradhye and Chitra Dorai. New kernels for analyzing multimodal data in multimedia using kernel machines. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 37–40. IEEE, 2002. 2.9.5, 3.4.2, 3.4.2, 5.1

[9] Benjamin Auder and Bertrand Iooss. Global sensitivity analysis based on entropy. In *Safety, reliability and risk analysis-Proceedings of the ESREL 2008 Conference*, pages 2107–2115, 2008. 2.3

[10] Peter C Austin and Ewout W Steyerberg. Interpreting the concordance statistic of a logistic regression model: relation to the variance and odds ratio of a continuous explanatory variable. *BMC medical research methodology*, 12(1):82, 2012. 4.5.1, 4.5.1

[11] K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`. 5.1

[12] Andreas Backhaus and Udo Seiffert. Quantitative measurements of model interpretability for the analysis of spectral data. In *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, pages 18–25. IEEE, 2013. 2.3

[13] Remo Badii and Antonio Politi. *Complexity: Hierarchical structures and scaling in physics*, volume 6. Cambridge University Press, 1999. 6.2, 6.2.1

[14] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010. 6.2.3

[15] David Barbella, Sami Benzaid, Janara M Christensen, Bret Jackson, X Victor Qin, and David R Musicant. Understanding support vector machine classifications via a recommender system-like approach. In *DMIN*, pages 305–311. Las Vegas, NV, 2009. 1.1, 1.1, 2.4, 5.1, 5.6.1, 2, 5.6.2, 5.4, 2, 2

[16] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, 2011. 3.4.2, 3.4.2, 4.6.1, 7.1, A.3

[17] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. A.3

[18] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. In *Data mining techniques for the life sciences*, pages 223–239. Springer, 2010. 1, 2.3, 2.8.1, 2.8.2, 2.8.3, 3.2, 5.2, 6.6, 6.7.4, A.7

[19] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *Information Theory, IEEE Transactions on*, 44(4):1407–1423, 1998. 2.6, 2.6, 3.4.2

[20] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012. 5.2

[21] Eta S Berner. *Clinical Decision Support Systems*. Springer Science+ Business Media, LLC, 2007. 1.1, 1.2, 4, 5, 3, 6

[22] Abdelaziz Berrado and George C Runger. Supervised multivariate discretization in mixed data with random forests. In *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, pages 211–217. IEEE, 2009. 5.1

[23] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001. A.3

[24] Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006. 1, 6, 2.1, 2.2, 2.2, 2.2.2, 2.2.2, 2.7.1, 2.7.1, 2.2, 2.3, 2.4, 2.8.1, 2.8.2, 2.8.3, 2.8.4, 2.9.2, 2.10, 3.2, 4.4, 4.4.2, A.1

[25] J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995. 5

[26] D Blokh, N Zurgil, I Stambler, E Afrimzon, Y Shafran, E Korech, J Sandbank, M Deutsch, et al. An information-theoretical model for breast cancer detection. *Methods Inf Med*, 47(4):322–327, 2008. 2.6.2

[27] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. *red*, 30(2):3, 2008. 2.6, 2.6.3, 2.6.3, 3.4.2

[28] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Abstractions in process mining: A taxonomy of patterns. In *Business Process Management*, pages 159–175. Springer, 2009. 2.6.2

[29] Remco R Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald, and David Scuse. Weka manual for version 3-7-8, 2013. 4.6

[30] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. Conditionally positive definite kernels for svm based image recognition. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 113–116. IEEE, 2005. 1.1, 2.7.3, 2.9.1, 2.9.1, 3.4.4, 3.4.4, 6.3

[31] Michael Bowling, Ali Ghodsi, and Dana Wilkinson. Action respecting embedding. In *Proceedings of the 22nd international conference on Machine learning*, pages 65–72. ACM, 2005. A.3

[32] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964. A.3

[33] Robert K Boyd, Cecilia Basic, and Robert A Bethem. *Trace quantitative analysis by mass spectrometry*. John Wiley & Sons, 2011. A.3

[34] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 2.1

[35] Mikio L Braun, Joachim M Buhmann, and Klaus-Robert MÃžller. On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research*, 9(Aug): 1875–1908, 2008. 3.4.1, 6.4, 6.5.1, 6.6

[36] Pavel Brazdil, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer, 2008. 1, 2.1, 3.2

[37] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 4, 2.1

[38] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 4, 2.1

[39] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001. 2.3, 3

[40] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. 2.2, 2.8.4, 3.2, A.7

[41] TP Burnaby. On a method for character weighting a similarity coefficient, employing the concept of information. *Journal of the International Association for Mathematical Geology*, 2(1):25–38, 1970. 3.4.2

[42] Evgeny Byvatov, Uli Fechner, Jens Sadowski, and Gisbert Schneider. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of Chemical Information and Computer Sciences*, 43(6):1882–1889, 2003. A.7

[43] Gustavo Camps-Valls. *Kernel methods in bioengineering, signal and image processing*. Igi Global, 2006. A.3

[44] Gustavo Camps-Valls, Luis Gomez-Chova, Jordi Muñoz-Marí, Joan Vila-Francés, and Javier Calpe-Maravilla. Composite kernels for hyperspectral image classification. *Geoscience and Remote Sensing Letters, IEEE*, 3(1):93–97, 2006. 2.8, 2.10, 4.8.4

[45] Andre M Carrington, Paul W Fieguth, and Helen H Chen. A new mercer sigmoid kernel for clinical data classification. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 6397–6401. IEEE, 2014. 1.1, 2.8.4, 3.2, 3.4.2, 3.4.3, 4.6.3, 4.7.1, 2, 5.8, 6.5, 7.1

[46] André M Carrington, Paul Fieguth, and Helen H Chen. Measures of model interpretability for model selection. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2018. 1.1

[47] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168. ACM, 2006. 1, 1.2, 2.2, 6

[48] Michael A Casey and Alex Westner. Separation of mixed audio sources by independent subspace analysis. In *ICMC*, pages 154–161, 2000. A.3

[49] Sung-Hyuk Cha, Sungsoon Yoon, and Charles C Tappert. Enhancing binary feature vector similarity measures. Technical report, Ivan G. Seidenberg School of Computer Science, Pace University, 2005. 2.6, 2.6.2, 2.6.2, 2.6.2, 2.6.2, 2.6.2, 2.6.3, 3.4.2, 3.4.2

[50] Yi-Hsiang Chao. Using lr-based discriminant kernel methods with applications to speaker verification. *Speech Communication*, 57:76–86, 2014. 4.5.2

[51] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014. 2.1

[52] Youngmin Cho. *Kernel Methods for Deep Learning*. PhD thesis, University of California, San Diego, 2012. 2.2

[53] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010. 3.4.2, 4.7

[54] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Two-stage learning kernel algorithms. In *ICML*, pages 239–246, 2010. 2.10

[55] Andrew Cotter, Joseph Keshet, and Nathan Srebro. Explicit approximations of the gaussian kernel. *arXiv Preprint arXiv:1109.4603*, 2011. 1.1, 4, 4.4.3, 6.6, 7.1

[56] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012. 6.2.3

[57] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000. 2.1, 2.2, 4.4.4

[58] Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer Informatics*, 2:59–78, 2006. 1.1, 2.2, 6

[59] Anneleen Daemen and Bart De Moor. Development of a kernel function for clinical data. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 5913–5917. IEEE, 2009. 2.9.5

[60] Anneleen Daemen, Dirk Timmerman, Thierry Van den Bosch, Cecilia Bottomley, Emma Kirk, Caroline Van Holsbeke, Lil Valentin, Tom Bourne, and Bart De Moor. Improved modeling of clinical data with kernel methods. *Artificial intelligence in medicine*, 54(2):103–114, 2012. 2.9.5

[61] Justin HG Dauwels. *On graphical models for communications and machine learning: algorithms, bounds, and analog implementation*. PhD thesis, Citeseer, 2006. 3.4.2

[62] Laurie Davies. Data analysis and approximate models. *Monographs on Statistics and Applied Probability*, 133, 2014. A.1

[63] Vincent Delaitre, David F Fouhey, Ivan Laptev, Josef Sivic, Abhinav Gupta, and Alexei A Efros. Scene semantics from long-term observation of people. In *European conference on computer vision*, pages 284–298. Springer, 2012. 2.9.3

[64] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006. 5

[65] Olivier Devos, Cyril Ruckebusch, Alexandra Durand, Ludovic Duponchel, and Jean-Pierre Huvenne. Support vector machines (svm) in near infrared (nir) spectroscopy: Focus on parameters optimization and model interpretation. *Chemometrics and Intelligent Laboratory Systems*, 96(1):27–33, 2009. 6.6

[66] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 5.1.2, 5.1.3, 5.1.4, 5.1.5

[67] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998. 5

[68] Anca Doloc-Mihu. Kernel method for improving image retrieval performance: a survey. *International Journal of Data Mining, Modelling and Management*, 3(1): 42–74, 2011. 2.9.3

[69] A Rogier T Donders, Geert JMG van der Heijden, Theo Stijnen, and Karel GM Moons. Review: a gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, 59(10):1087–1091, 2006. A.10

[70] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. 2017. 6.2.2

[71] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002. 1.2, 4, 6, 2.1, 2.2.2

[72] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification and scene analysis 2nd ed. 1995. 2.1, 2.1, 3.1, 3.4.2

[73] Dr. Eric Ehrsam. Dermoscopy. `http://dermoscopic.blogspot.com`, 2007. URL `http://dermoscopic.blogspot.com`. 5.1, 5.1.1

[74] Margaret J Eppstein, Jeffrey D Horbar, Jeffrey S Buzas, and Stuart A Kauffman. Searching the clinical fitness landscape. *PloS one*, 7(11):e49901, 2012. 2.6.2

[75] Olusegun Famure, Nicholas Anh-Tuan Phan, and Sang Joseph Kim. Health information management for research and quality assurance: the comprehensive renal transplant research information system. In *Healthcare management forum*, volume 27, pages 30–36. SAGE Publications Sage CA: Los Angeles, CA, 2014. 7.1

[76] Christine Fennema-Notestine, I Burak Ozyurt, Camellia P Clark, Shaunna Morris, Amanda Bischoff-Grethe, Mark W Bondi, Terry L Jernigan, Bruce Fischl, Florent Segonne, David W Shattuck, et al. Quantitative evaluation of automated skull-stripping methods applied to contemporary and legacy images: Effects of diagnosis, bias correction, and slice location. *Human brain mapping*, 27(2):99–113, 2006. 2.6.2

[77] Jonathan T Foote. Content-based retrieval of music and audio. In *Voice, Video, and Data Communications*, pages 138–147. International Society for Optics and Photonics, 1997. 2.6.4, 3.4.2, 3.4.2

[78] George Forman and Ira Cohen. Learning from little: Comparison of classifiers given little training. In *Knowledge Discovery in Databases: PKDD 2004*, pages 161–172. Springer, 2004. 4.4.4, A.7

[79] George Forman, Martin Scholz, and Shyamsundar Rajaram. Feature shaping for linear svm classifiers. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 299–308. ACM, 2009. 4.5.2

[80] Mike Fornefett, Karl Rohr, and H Siegfried Stiehl. Radial basis functions with compact support for elastic registration of medical images. *Image and vision computing*, 19(1):87–96, 2001. 3.4.2

[81] Martin Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004. 2.5

[82] David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4): 453–476, 1981. 6.2.2, 6.5.1, 6.6

[83] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, volume 96, pages 148–156, 1996. 4, 2.1

[84] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001. 4.4.1

[85] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 4, 2.1

[86] Krzysztof Fujarewicz and Małgorzata Wiench. Selecting differentially expressed genes for colon tumor classification. *International Journal of Applied Mathematics and Computer Science*, 13:327–335, 2003. 4.4.3

[87] Kenji Fukumizu. Theory of positive definite kernel and reproducing kernel hilbert space. Graduate University of Advanced Studies, 2008. URL `http://www.ism.ac.jp/~fukumizu/H20_kernel/Kernel_7_theory.pdf`. 2.9.2

[88] Kenji Fukumizu. Kernel method: Data analysis with positive definite kernels. Graduate University of Advanced Studies, 2010. URL `http://www.is.titech.ac.jp/~shimo/class/fukumizu/Kernel_elements_2.pdf`. 2.9.2

[89] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000. 4.4.3

[90] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137–144, 2015. 2.1

[91] Rahil Garnavi, Mohammad Aldeen, and James Bailey. Classification of melanoma lesions using wavelet-based texture analysis. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 75–81. IEEE, 2010. A.7

[92] Paul Geladi and Bruce R Kowalski. Partial least-squares regression: a tutorial. *Analytica chimica acta*, 185:1–17, 1986. A.3

[93] Murray Gell-Mann and Seth Lloyd. Information measures, effective complexity, and total information. *Complexity*, 2(1):44–52, 1996. 6.2.3

[94] Marc G Genton. Classes of kernels for machine learning: a statistics perspective. *The Journal of Machine Learning Research*, 2:299–312, 2002. 1.1, 2.1, 2.7.1, 2.2, 2.3, 2.4, 2.7, 2.8, 2.9.2, 2.9.3, 2.9.4, 2.9.5, 3.4.2, 4.4, 4.4, 1, 4

[95] Zoubin Ghahramani. Unsupervised learning. In *Advanced lectures on machine learning*, pages 72–112. Springer, 2004. A.3

[96] Ali Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 37:38, 2006. A.3

[97] Stephen Gilmore, Rainer Hofmann-Wellenhof, and H Peter Soyer. A support vector machine for decision support in melanoma recognition. *Experimental dermatology*, 19(9):830–835, 2010. A.7

[98] King-Shy Goh, Edward Chang, and Kwang-Ting Cheng. Svm binary classifier ensembles for image classification. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 395–402. ACM, 2001. 2.9.2

[99] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011. 2.10

[100] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". In *1st Workshop on Human Interpretability in Machine Learning, International Conference of Machine Learning*, 2016. 1.2, 6

[101] David L Goodstein and Judith R Goodstein. *Feynman's lost lecture: the motion of planets around the sun*, volume 1. WW Norton & Company, 1996. 5.6, 6.2

[102] John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871, 1971. 2.6, 2.6.2, 3.4.2

[103] John C Gower and Pierre Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of classification*, 3(1):5–48, 1986. 2.6

[104] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984. A.3

[105] DM Green and JA Swets. Signal detection theory and psychophysics. 1966. *New York*. 4.5.1, 4.5.1

[106] Robert A Greenes. *Clinical decision support: the road ahead*. Academic Press, 2011. 1.1, 1, 2, 4, 5, 3, 6

[107] Jerzy W Grzymala-Busse and Witold J Grzymala-Busse. Handling missing attribute values. In *Data mining and knowledge discovery handbook*, pages 37–57. Springer, 2005. A.10

[108] Harsha Gurulingappa, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. Identification of adverse drug event assertive sentences in medical case reports. In *First international workshop on knowledge discovery and health care management (KD-HCM), European conference on machine learning and principles and practice of knowledge discovery in databases (ECML PKDD)*, pages 16–27, 2011. A.7

[109] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003. 2.1

[110] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46 (1-3):389–422, 2002. 4.4.3

[111] James A Hanley. The robustness of the" binormal" assumptions used in fitting roc curves. *Medical Decision Making*, 8(3):197–203, 1988. 4.5.1, 4.5.1

[112] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. 4.5.1

[113] Kenneth M Hanson and François M Hemez. *Sensitivity Analysis of Model Output: Proceedings of the 4th International Conference on Sensitivity Analysis of Model Output (SAMO 2004): Santa Fe, New Mexico, March 8-11 2004*. Los Alamos National Laboratory, 2005. 2.3

[114] A Hart, JM Smith, MA Skeans, SK Gustafson, DE Stewart, WS Cherikh, JL Wainright, A Kucheryavaya, M Woodbury, JJ Snyder, et al. Optn/srtr 2015 annual data report: kidney. *American Journal of Transplantation*, 17:21–116, 2017. 7.1

[115] Trevor Hastie and Robert Tibshirani. *Generalized additive models*. Wiley Online Library, 1990. 2, 4.4.1, A.1

[116] Trevor Hastie and Robert Tibshirani. Generalized additive models for medical research. *Statistical methods in medical research*, 4(3):187–196, 1995. 2, 4.4.1

[117] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5(Oct):1391–1415, 2004. 2.2.1, 4

[118] David Haussler. Convolution kernels on discrete structures. Technical report, Citeseer, 1999. 2.8, 3.4.2

[119] M Heikkinen, A Sarpola, H Hellman, J Rämö, and Y Hiltunen. Independent component analysis to mass spectra of aluminium sulphate. *Spectrum*, 200:000, 2007. A.3

[120] Matthias Hein and Olivier Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *AISTATS*, pages 136–143, 2005. 2.9.3

[121] Francis B Hildebrand. *Methods of applied mathematics*. Courier Corporation, 2012. 2.8

[122] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003. A.3

[123] Charles Antony Richard Hoare. *Chapter II: Notes on data structuring*. Academic Press Ltd., 1972. 2.5, A.5

[124] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999. A.3

[125] Robert V Hogg and Elliot A Tanis. *Probability and statistical inference*. Prentice Hall Inc., 7 edition, 2005. 5.2

[126] Andreas Holzinger, Chris Biemann, Constantinos S Pattichis, and Douglas B Kell. What do we need to build explainable ai systems for the medical domain? *arXiv preprint arXiv:1712.09923*, 2017. 1.2, 2.3, 6

[127] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. A.3

[128] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003. 3.2, 4.6, A.7

[129] Aapo Hyvärinen. Independent component analysis: recent advances. *Phil. Trans. R. Soc. A*, 371(1984):20110534, 2013. A.3

[130] Aapo Hyvärinen, Jarmo Hurri, and Patrik O Hoyer. Principal components and whitening. In *Natural Image Statistics*, pages 93–130. Springer, 2009. A.3

[131] National Cancer Institute. ISO 21090 Wiki. URL `https://wiki.nci.nih.gov/display/ISO21090/ISO+21090+Wiki`. A.4

[132] International Organization for Standardization. International standard: Information technology – general-purpose datatypes (iso/iec 11404). Geneva, Switzerland, 2007. 2.5

[133] International Organization for Standardization. Systems and software engineering-software life cycle processes (iec 12207). Geneva, Switzerland, 2008. 4.4.3

[134] International Organization for Standardization. International standard: Health informatics – harmonized data types for information interchange (iso 21090). Geneva, Switzerland, 2011. 2.5, A.4

[135] Tommi Jaakkola, Mark Diekhans, and David Haussler. Using the fisher kernel method to detect remote protein homologies. In *ISMB*, volume 99, pages 149–158, 1999. 3.4.2, 4.5.2

[136] Tony Jebara and Risi Kondor. Bhattacharyya and expected likelihood kernels. In *Learning theory and kernel machines*, pages 57–71. Springer, 2003. 2.9.3

[137] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *Journal of Machine Learning Research*, 5(Jul):819–844, 2004. 2.9.3

[138] Marvin Ed Jernigan and Paul Fieguth. *Introduction to Pattern Recognition*. University of Waterloo, 2004. 1.4, 6.2

[139] N Karami and A Esteki. Analysis of complexity features of dermatological images, effective tool for automated diagnosis of melanoma. In *Biomedical Engineering (ICBME), 2011 18th Iranian Conference of*, pages 43–47. IEEE, 2011. A.7

[140] Yohei Kawaguchi, Masahito Togami, Hisashi Nagano, Yuichiro Hashimoto, Masuyuki Sugiyama, and Yasuaki Takada. Ica-based acceleration of probabilistic latent component analysis for mass spectrometry-based explosives detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2795–2799. IEEE, 2013. A.3

[141] Hannes Kazianka, Raimund Leitner, and Jürgen Pilz. Segmentation and classification of hyper-spectral skin data. In *GfKl*, pages 245–252. Springer, 2007. A.7

[142] Vojislav Kecman. Support vector machines–an introduction. In *Support vector machines: theory and applications*, pages 1–47. Springer, 2005. 2.9.1

[143] Maurice G Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3): 239–251, 1945. 6.5

[144] Aditya Khosla, Yu Cao, Cliff Chiung-Yu Lin, Hsu-Kuang Chiu, Junling Hu, and Honglak Lee. An integrated machine learning approach to stroke prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 183–192. ACM, 2010. 2.2, A.7

[145] Ross D. King, Cao Feng, and Alistair Sutherland. Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence an International Journal*, 9(3):289–333, 1995. 5.1

[146] Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, 2004. 2.1

[147] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Lp-norm multiple kernel learning. *The Journal of Machine Learning Research*, 12:953–997, 2011. 4

[148] SB Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(3):249–268, 2007. 2.6.4, 2.8.2, 2.8.3, 3.4.2, 3.4.2

[149] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. A.3

[150] Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964. A.3

[151] Brian Kulis et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013. 2.1

[152] Satish Kumar. *Neural networks: a classroom approach*. Tata McGraw-Hill Education, 2004. 2.8.2, 2.8.3

[153] Sun-Yuan Kung and Man-Wai Mak. Pda-svm hybrid: A unified model for kernel-based supervised classification. *Journal of Signal Processing Systems*, 65(1):5–21, 2011. 4.4.3

[154] Elisabetta 'La Torre', Barbara Caputo, and Tatiana Tommasi. Learning methods for melanoma recognition. *International Journal of Imaging Systems and Technology*, 20(4):316–322, 2010. A.7

[155] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004. 2.1, 2.10

[156] Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013. 4.4.2

[157] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436, 2015. 4, 2.1

[158] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. A.3

[159] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001. A.3

[160] Vincent Lemaire, Raphael Féraud, and Nicolas Voisine. Contact personalization using a score understanding method. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 649–654. IEEE, 2008. 2.3, 2.4, 3, 6.2.3

[161] Mi Li, Xiaofeng Lu, Xiaodong Wang, Shengfu Lu, and Ning Zhong. Biomedical classification application and parameters optimization of mixed kernel svm based on the information entropy particle swarm optimization. *Computer Assisted Surgery*, 21(sup1):132–141, 2016. 1.1, 1.1, 3.4.2

[162] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul Vitányi. The similarity metric. *Information Theory, IEEE Transactions on*, 50(12):3250–3264, 2004. 2.6, 2.6, 3.4.2

[163] Qi Li and Jeff Racine. Nonparametric estimation of distributions with categorical and continuous data. *journal of multivariate analysis*, 86(2):266–292, 2003. 4.5

[164] Percy Liang. Provenance and contracts in machine learning. In *Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, 2016. 2.3, 2

[165] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning*, 40(3):203–228, 2000. 1.2

[166] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998. 2.6, 2, 3, 4, 5, 6, 7, 8, 9, 2.6, 2.6.4, 3.4.1, 3.4.2, 6.2.1

[167] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *submitted to Neural Computation*, pages 1–32, 2003. 1.1, 2.8.4, 3.2, 1, 3.4.3, 4.7.1, 5.2, 2

[168] Zachary C Lipton, David C Kale, Charles Elkan, Randall Wetzell, Sharad Vikram, Julian McAuley, Randall C Wetzell, Zhanglong Ji, Balakrishnan Narayaswamy, Cheng-I Wang, et al. The mythos of model interpretability. *IEEE Spectrum*, 2016. 1.1, 2.3, 3.4.4, 1, 3.4.4, 1, 3.4.4, 6.2

[169] Paulo JG Lisboa. Interpretability in machine learning–principles and practice. In *International Workshop on Fuzzy Logic and Applications*, pages 15–21. Springer, 2013. 1, 6

[170] Huibin Liu, Wei Chen, and Agus Sudjianto. Relative entropy based method for probabilistic sensitivity analysis in engineering design. *Journal of Mechanical Design*, 128(2):326–336, 2006. 2.3

[171] Seth Lloyd. Measures of complexity: a nonexhaustive list. *IEEE Control Systems Magazine*, 21(4):7–8, 2001. 6.2.1

[172] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158. ACM, 2012. 1.1, 2.3, 3.4.4, 3.4.4, 2, 2a, 2a, 4.4.1, 6.2, 7

[173] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. A.3

[174] Julián Luengo, Salvador García, and Francisco Herrera. A study on the use of imputation methods for experimentation with radial basis function network classifiers

handling missing attribute values: The good synergy between rbfns and eventcovering method. *Neural Networks*, 23(3):406–418, 2010. A.10

[175] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. A.3

[176] Ilias Maglogiannis and Dimitrios I Kosmopoulos. A system for the acquisition of reproducible digital skin lesions images. *Technology and Health Care-European Society for Engineering and Medicine*, 11(6):425–442, 2003. A.7

[177] Ilias Maglogiannis, Elias Zafiropoulos, and Christos Kyranoudis. Intelligent segmentation and classification of pigmented skin lesions in dermatological images. In *Advances in Artificial Intelligence*, pages 214–223. Springer, 2006. A.7

[178] Subhrajyoti Maji, Alexander C Berg, and Jagannath Malik. Efficient classification for additive kernel svms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):66–77, 2013. 4.4, 4

[179] Subhransu Maji and Alexander C Berg. Max-margin additive classifiers for detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 40–47. IEEE, 2009. 2.8

[180] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2.9.3

[181] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989. 3.4.2

[182] Dante Mantini, Francesca Petrucci, Piero Del Boccio, Damiana Pieragostino, Marta Di Nicola, Alessandra Lugaresi, Giorgio Federici, Paolo Sacchetta, Carmine Di Ilio, and Andrea Urbani. Independent component analysis for the extraction of reliable protein signal profiles from maldi-tof mass spectra. *Bioinformatics*, 24(1):63–70, 2008. A.3

[183] David Martens and Bart Baesens. Building acceptable classification models. In *Data Mining*, pages 53–74. Springer, 2010. 2.3

[184] AJ Matas, JM Smith, MA Skeans, B Thompson, SK Gustafson, MA Schnitzler, DE Stewart, WS Cherikh, JL Wainright, JJ Snyder, et al. Optn/srtr 2012 annual data report: kidney. *American Journal of Transplantation*, 14(S1):11–44, 2014. 7.1

[185] John F McCarthy, Kenneth A Marx, Patrick E Hoffman, Alexander G Gee, Philip O'Neil, M L Ujwal, and John Hotchkiss. Applications of machine learning and high-dimensional visualization in cancer detection, diagnosis, and management. *Annals of the New York Academy of Sciences*, 1020(1):239–262, 2004. A.7

[186] James McDermott and Richard S Forsyth. Diagnosing a disorder in a classification benchmark. *Pattern Recognition Letters*, 73:41–43, 2016. 5.1, 6.5

[187] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, pages 415–446, 1909. 2.2, 2.7.1, 2.5, 2.6, 3.4.4, 3.4.4, 4.4, 6.3

[188] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint arXiv:1706.07269*, 2017. 1.1, 2.3

[189] Tim Miller, Piers Howe, and Liz Sonenberg. Explainable ai: Beware of inmates running the asylum. In *IJCAI-17 Workshop on Explainable AI (XAI)*, page 36, 2017. 2.3

[190] Simone Mocellin, Alessandro Ambrosi, Maria Cristina Montesco, Mirto Foletto, Giorgio Zavagno, Donato Nitti, Mario Lise, and Carlo Riccardo Rossi. Support vector machine learning model for the prediction of sentinel node status in patients with cutaneous melanoma. *Annals of surgical oncology*, 13(8):1113–1122, 2006. A.7

[191] Michael Mongillo. Choosing basis functions and shape parameters for radial basis function methods. *SIAM Undergraduate Research Online*, 4:190–209, 2011. 2.6

[192] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017. 1.1, 2.3, 2.4, 6.2.3

[193] Teresa A Myers. Goodbye, listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data. *Communication Methods and Measures*, 5(4):297–310, 2011. A.10

[194] Jesmin Nahar, Shawkat Ali, and Yi-Ping Phoebe Chen. Microarray data classification using automatic svm kernel selection. *DNA and cell biology*, 26(10):707–712, 2007. 1.1, 2.3, 2.4

[195] Danh V Nguyen and David M Rocke. Partial least squares proportional hazard regression for application to dna microarray survival data. *Bioinformatics*, 18(12): 1625–1632, 2002. A.3

[196] Diarmuid Ó Séaghdha and Ann Copestake. Semantic classification with distributional kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 649–656. Association for Computational Linguistics, 2008. 2.9.2

[197] Lucila Ohno-Machado. Modeling medical prognosis: survival analysis techniques. *Journal of biomedical informatics*, 34(6):428–439, 2001. A.10

[198] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1):36, 2017. 1, 1.2, 2.2, 6

[199] Cheng Soon Ong, Xavier Mary, Stéphane Canu, and Alexander J Smola. Learning with non-positive kernels. In *Proceedings of the twenty-first international conference on Machine learning*, page 81. ACM, 2004. 2.9.5, 2

[200] Cheng Soon Ong, Alex Smola, and Bob Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1045–1071, 2005. 2.10

[201] Margarita Osadchy, Yann Le Cun, and Matthew L Miller. Synergistic face detection and pose estimation with energy-based models. *The Journal of Machine Learning Research*, 8:1197–1215, 2007. 3.4.2

[202] Topon Kumar Paul and Hitoshi Iba. Gene selection for classification of cancers using probabilistic model building genetic algorithm. *BioSystems*, 82(3):208–225, 2005. 4.4.3

[203] Jennie L Pearce and Mark S Boyce. Modelling distribution and abundance with presence-only data. *Journal of applied ecology*, 43(3):405–412, 2006. 3.4.2

[204] Ronald K Pearson. The problem of disguised missing data. *ACM SIGKDD Explorations Newsletter*, 8(1):83–92, 2006. 5.1

[205] Pedro Santoro Perez, Sérgio Ricardo Nozawa, Alessandra Alaniz Macedo, and José Augusto Baranauskas. Windowing improvements towards more comprehensible models. *Knowledge-Based Systems*, 92:9–22, 2016. 1.1, 2.3, 2.4

[206] Pietro Perona. Deformable kernels for early vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(5):488–499, 1995. 4

[207] Weronika Piatkowska, Jerzy Martyna, Leszek Nowak, and Karol Przystalski. A decision support system based on the semantic analysis of melanoma images using multi-elitist pso and svm. In *Machine Learning and Data Mining in Pattern Recognition*, pages 362–374. Springer, 2011. A.7

[208] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 689–690. IEEE, 2011. 2.1

[209] Brett Poulin, Roman Eisner, Duane Szafron, Paul Lu, Russell Greiner, David S Wishart, Alona Fyshe, Brandon Pearcy, Cam MacDonell, and John Anvik. Visual explanation of evidence with additive classifiers. In *Proceedings Of The National Conference On Artificial Intelligence*, volume 21, page 1822. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006. 1.1, 2.3, 2.4, 3, 4.4.3

[210] Martin V Pusic, Kathy Boutis, Rose Hatala, and David A Cook. Learning curves in health professions education. *Academic Medicine*, 90(8):1034–1042, 2015. 2.3, 6.2.1

[211] Jeffrey Scott Racine. *Nonparametric econometrics: A primer*. Now Publishers Inc, 2008. 4.5

[212] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20:1177–1184, 2007. 2.9.2

[213] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004. 2.1

[214] Alfréd Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961. 6.2.3

[215] Vicent Ribas Ripoll, Enrique Romero Merino, Juan Carlos Ruiz Rodríguez, Alfredo Vellido Alcacena, et al. A quotient basis kernel for the prediction of mortality in severe sepsis patients. 2013. 2.10, 4.5.2

[216] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016. 2.3, 1, 2, 3, 4, 5, 6.2.3

[217] Peter H Roe, GN Soulis, and VK Handa. *The discipline of design*. Allyn and Bacon, 1967. 4.4.3

[218] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000. A.3

[219] Patrick Royston, Gareth Ambler, and Willi Sauerbrei. The use of fractional polynomials to model continuous risk variables in epidemiology. *International journal of epidemiology*, 28(5):964–974, 1999. 4.2.2

[220] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. A.10

[221] Mark Ruzon, Carlo Tomasi, et al. Color edge detection with the compass operator. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999. 3.4.2

[222] Hichem Sahbi, Jean-Yves Audibert, and Renaud Keriven. Context-dependent kernels for object classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):699–708, 2011. 4.5.2

[223] Gerard Salton. Developments in automatic text retrieval. *Science*, 253(5023):974–980, 1991. 2.6.4

[224] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969. A.3

[225] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990. 2.1, 2.1

[226] Bernhard Scholkopf. The kernel trick for distances. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 301. MIT Press, 2001. 1.1, 2.7.3, 2.9.1, 2.9.1

[227] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 1.1, 1.1, 3.4.2

[228] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer, 1997. 2.9.5, A.3

[229] Bernhard Schölkopf, Patrice Simard, Alex J Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. In *Advances in neural information processing systems*, pages 640–646, 1998. 2.7.3, 2.7.4, 2.6, 2.9.3

[230] David W Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979. 6.2.2, 6.5.1, 6.6

[231] Steve Selvin. *Statistical analysis of epidemiologic data*. Oxford University Press, 2004. 6.5

[232] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52:55–66, 2010. 2.1

[233] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004. 4, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2.2, 2.2.2, 2.2.2, 2.7, 2.7.1, 2.7.1, 2.2, 2.3, 2.4, 2.7.3, 2.8.3, 2.10, 4.4, 4.4.2, 4.4.2, 4.4.3, 6.5.2

[234] Patrice Simard, Yann LeCun, and John S Denker. Efficient pattern recognition using a new transformation distance. In *Advances in neural information processing systems*, pages 50–58, 1993. 2.6.4

[235] Dinesh Singh, Phillip G Febbo, Kenneth Ross, Donald G Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A Renshaw, Anthony V D'Amico, Jerome P Richie, et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209, 2002. 5.1, 5.1.7

[236] Paris Smaragdis and Bhiksha Raj. Shift-invariant probabilistic latent component analysis. *Journal of Machine Learning Research*, 2007. A.3

[237] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. A probabilistic latent variable model for acoustic modeling. *Advances in models for acoustic processing, NIPS*, 148:8–1, 2006. A.3

[238] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine Learning*, pages 1–32, 2013. 2.1

[239] Steven W Smith et al. The scientist and engineer's guide to digital signal processing. 1997. A.3

[240] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*. Citeseer, 1998. 2.2, 2.2, 4.4, 4, 4.4.2, 4.4.4

[241] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004. 2.2, 2.7.3, 2.7.4, 2.6

[242] Alex J Smola, Bernhard Schölkopf, and Klaus-Robert Müller. The connection between regularization operators and support vector kernels. *Neural networks*, 11(4): 637–649, 1998. 1.1, 2.5, 2.6, 3.4.4

[243] Alex J Smola, Zoltan L Ovari, and Robert C Williamson. Regularization with dot-product kernels. *Advances in neural information processing systems*, pages 308–314, 2001. 2.7.1, 2.7, 4.4

[244] Elliott Sober. Parsimony and predictive equivalence. *Erkenntnis*, 44(2):167–197, 1996. 2.3, 6, 6.4, 7

[245] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1):271–280, 2001. 2.3

[246] Irena Spasic, Sophia Ananiadou, John McNaught, and Anand Kumar. Text mining and ontologies in biomedicine: making sense of raw text. *Briefings in bioinformatics*, 6(3):239–251, 2005. A.3

[247] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. 2.8.3

[248] Ashok N Srivastava et al. Mixture density mercer kernels: A method to learn kernels directly from data. SIAM. 2.8

[249] Stanley Smith Stevens. On the theory of scales of measurement, 1946. 2.5, 2.1

[250] Ewout W Steyerberg. *Clinical prediction models: a practical approach to development, validation, and updating*. Springer, 2009. 1, 1, 2, 3, 7.1, A.10, 7.1

[251] Herbert A Sturges. The choice of a class interval. *Journal of the american statistical association*, 21(153):65–66, 1926. 6.2.2, 6.5.1, 6.6

[252] Grzegorz Surówka. Supervised learning of melanocytic skin lesion images. In *Human System Interactions, 2008 Conference on*, pages 121–125. IEEE, 2008. A.7

[253] Grzegorz Surowka and Katarzyna Grzesiak-Kopec. Different learning paradigms for the classification of melanoid skin lesions using wavelets. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 3136–3139. IEEE, 2007. A.7

[254] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Undercomplete blind subspace deconvolution. *Journal of Machine Learning Research*, 8:1063–1095, 2007. 6.2.3

[255] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Separation theorem for independent subspace analysis and its consequences. *Pattern Recognition*, 45:1782–1791, 2012. 6.2.3

[256] Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. *Machine learning*, 79(1-2):73–103, 2010. 2.10, 3.4.2, 4.8.4

[257] Yuchun Tang, Bo Jin, Yi Sun, and Yan-Qing Zhang. Granular support vector machines for medical binary classification problems. In *Computational Intelligence in Bioinformatics and Computational Biology, 2004. CIBCB'04. Proceedings of the 2004 IEEE Symposium on*, pages 73–78. IEEE, 2004. 5.1

[258] Michael Reed Teague. Image analysis via the general theory of moments. *JOSA*, 70 (8):920–930, 1980. A.3

[259] Joshua B Tenenbaum. Mapping a manifold of perceptual observations. In *Advances in neural information processing systems*, pages 682–688, 1998. A.3

[260] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000. A.3

[261] Louis Leon Thurstone. Multiple factor analysis. 1947. A.3

[262] Tatiana Tommasi, Elisabetta La Torre, and Barbara Caputo. Melanoma recognition using representative and discriminative kernel classifiers. In *International Workshop on Computer Vision Approaches to Medical Image Analysis*, pages 1–12. Springer, 2006. 2.2

[263] Muchenxuan Tong, Kun-Hong Liu, Chungui Xu, and Wenbin Ju. An ensemble of svm classifiers based on gene pairs. *Computers in biology and medicine*, 43(6):729–737, 2013. 4.4.3

[264] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952. A.3

[265] Elisabetta La Torre, Tatiana Tommasi, and Barbara Caputo. Kernel methods for melanoma recognition. In *Medical Informatics in Europe (MIE)*, number LIDIAP-CONF-2006-019, 2006. A.7

[266] Huy Dat Tran and Haizhou Li. Probabilistic distance svm with hellinger-exponential kernel for sound event classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2272–2275. IEEE, 2011. 2.9.3

[267] Constantino Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, 52(1):479–487, 1988. 6.2.3

[268] Ivor W Tsang, James T Kwok, C Bay, and H Kong. Distance metric learning with kernels. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 126–129. Citeseer, 2003. 2.6.4

[269] Koji Tsuda and William Stafford Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(suppl 1):i326–i333, 2004. 3.4.2

[270] Koji Tsuda, Motoaki Kawanabe, Gunnar Rätsch, Sören Sonnenburg, and Klaus-Robert Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, 2002. 4.5.2

[271] Peter D Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of artificial intelligence research*, 2:369–409, 1994. 5.1

[272] Alan Tussy and R Gustafson. *Elementary Algebra*. Nelson Education, 2012. 1.1, 2.6, 2.6.4, 6.2.1

[273] Amos Tversky and Itamar Gati. Similarity, separability, and the triangle inequality. *Psychological review*, 89(2):123, 1982. 2.6

[274] The United States. Executive Office of the President and John Podesta. *Big data: seizing opportunities, preserving values*. 2014. 1.2

[275] Kiichi Urahama and Satoshi Kawakami. Modified deformable model for bijective topology preserving map. *IEICE TRANSACTIONS on Information and Systems*, 77(10):1186–1188, 1994. 3.4.2

[276] Roberto Valerio and Ricardo Vilalta. A data complexity approach to kernel selection for support vector machines. In *AAAI*, pages 3138–3139, 2014. 1.1, 2.4

[277] Roberto Valerio and Ricardo Vilalta. Kernel selection in support vector machines using gram-matrix properties. In *Proceedings of the 27th International Conference on Advances in Neural Information Processing Systems. Workshop on Modern Nonparametrics: Automating the Learning Pipeline, NIPS*, volume 14, 2014. 1.1, 2.4

[278] Vanya Van Belle and Paulo Lisboa. Automated selection of interaction effects in sparse kernel methods to predict pregnancy viability. In *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, pages 26–31. IEEE, 2013. 1.1, 2.9.3, 7.1

[279] Raf Vandebril, Marc Van Barel, and Nicola Mastronardi. *Matrix computations and semiseparable matrices: linear systems*, volume 1. JHU Press, 2007. 2.8, 2.9.3, 4.4, 4.4, 1

[280] Bram Vanschoenwinkel and Bernard Manderick. Appropriate kernel functions for support vector machine learning with sequences of symbolic data. In *Deterministic and statistical methods in machine learning*, pages 256–280. Springer, 2005. 2.7.1

[281] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998. 2.2, 2.2.1, 3.2

[282] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3): 480–492, 2012. 2.9.3, 4

[283] Thierry Verplancke, Stijn Van Looy, Dominique Benoit, Stijn Vansteelandt, Pieter Depuydt, Filip De Turck, and Johan Decruyenaere. Support vector machine versus logistic regression modeling for prediction of hospital mortality in critically ill patients with haematological malignancies. *BMC Medical Informatics and Decision Making*, 8(1):56, 2008. A.7

[284] Andrew J Vickers and Elena B Elkin. Decision curve analysis: a novel method for evaluating prediction models. *Medical Decision Making*, 26(6):565–574, 2006. 8

[285] Jun Wang, Huyen T Do, Adam Woznica, and Alexandros Kalousis. Metric learning with multiple kernels. In *Advances in neural information processing systems*, pages 1170–1178, 2011. 2.1

[286] Jonathan Stuart Ward and Adam Barker. Undefined by data: a survey of big data definitions. *arXiv preprint arXiv:1309.5821*, 2013. 2.1

[287] Kilian Q Weinberger and Lawrence K Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, volume 6, pages 1683–1686, 2006. A.3

[288] Andrew Gordon Wilson. Covariance kernels for fast automatic pattern discovery and extrapolation with gaussian processes. *University of Cambridge*, 2014. 1.1, 1.1, 5, 3.4.2

[289] D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, pages 1–34, 1997. 2.6.4, 3.4.2

[290] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005. 2.1, 2.1, 2.5, A.10

[291] Andrew KC Wong, Bin Wu, Gene PK Wu, and Keith CC Chan. Pattern discovery for large mixed-mode database. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 859–868. ACM, 2010. 5.1

[292] Jionglin Wu, Jason Roy, and Walter F Stewart. Prediction modeling using ehr data: challenges, strategies, and a comparison of machine learning approaches. *Medical Care*, 48(6):S106–S113, 2010. 1, 1.2, 2.2, A.7

[293] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008. 1, 2.2, A.2

[294] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013. 2.1

[295] Shinichi Yamada and Kourosh Neshatiant. Hyper-parameter search in support vector machines using pso with cellular fitness approximation. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–8. IEEE, 2017. 1.1, 3.2, 1, 5.8, A.9

[296] Jieping Ye, Kewei Chen, Teresa Wu, Jing Li, Zheng Zhao, Rinkal Patel, Min Bae, Ravi Janardan, Huan Liu, Gene Alexander, et al. Heterogeneous data fusion for alzheimer's disease study. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1025–1033. ACM, 2008. 2.10

[297] Eunseog Youn and Myong K Jeong. Class dependent feature scaling method using naive bayes classifier for text datamining. *Pattern Recognition Letters*, 30(5):477–485, 2009. 4.5.2

[298] Xiaojing Yuan, Zhenyu Yang, George Zouridakis, and Nizar Mullani. Svm-based texture classification and application to early melanoma detection. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 4775–4778. IEEE, 2006. A.7

[299] Lotfi Asker Zadeh. Similarity relations and fuzzy orderings. *Information sciences*, 3 (2):177–200, 1971. 2.6, 1

[300] Jerrold H Zar. Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578–580, 1972. 2.6.5

[301] Chongsheng Zhang, Changchang Liu, Xiangliang Zhang, and George Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, 2017. 1, 1.2, 2.2

[302] Dengsheng Zhang and Guojun Lu. Evaluation of similarity measurement for image retrieval. In *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, volume 2, pages 928–931. IEEE, 2003. 2.6.4

[303] Lei Zhang, Dimitris Samaras, Dardo Tomasi, Nora Volkow, and Rita Goldstein. Machine learning for clinical diagnosis from functional magnetic resonance imaging. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 1211–1217. IEEE, 2005. A.7

[304] Li Zhang, Weida Zhou, and Licheng Jiao. Wavelet support vector machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(1):34–39, 2004. 2.7.3, 2.7.4, 2.6, 2.9.3

[305] Yulei Zhang, Yan Dang, Hsinchun Chen, Mark Thurmond, and Cathy Larson. Automatic online news monitoring and classification for syndromic surveillance. *Decision Support Systems*, 47(4):508–517, 2009. A.7

[306] Xiao-Hua Zhou, Donna K McClish, and Nancy A Obuchowski. *Statistical methods in diagnostic medicine*, volume 569. John Wiley & Sons, 2009. 4.5.1

[307] Mu Zhu. Kernels and ensembles: Perspectives on statistical learning. *The American Statistician*, 62(2):97–109, 2008. 2.1

[308] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006. 2.1

[309] Mark H Zweig and Gregory Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39(4):561–577, 1993. 4.5.1, 4.5.1