University of Sussex

Learning Using Privileged Information

A Unified Approach

Oliver Thomas
Candidate Number: 118448
Supervisor: Dr. Novi Quadrianto

Submitted for the degree of Bachelors of Computer Science
University of Sussex
April 2017

Declaration

This report is submitted as part requirement for the degree of Computer Science at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Signature:

Oliver Thomas

Acknowledgements

Thank you to the members of the SMiLe CLiNiC reading group for their encouraging and inclusive nature. I'd particularly like to thank Novi Quadrianto for his support and Joseph Taylor for his incredible patience. This paper could not have been written without them both.

UNIVERSITY OF SUSSEX

Oliver Thomas

Learning Using Privileged Information
A Unified Approach

Summary

This report revolves around the concept that we can learn better rules for classification if we have more data. The more data we have, the better we can emulate some ground truth. However, this falls down in the real world. When we come to make predictions we don't always have all the data available. Consider trying to diagnose a patient as either healthy or unwell based solely on a scan of their body. Despite having entire patient histories available, current machine learning techniques require us to simply learn from past scans and a label associated with them. This paper examines techniques within the Leaning Using Privileged Information (LUPI) paradigm that take advantage of extra relevant and useful data when training our classifier. This way we can incorporate extra information about our data into our classifier during training, though we can still make predictions about whether the patient is well or not based solely on their body scan. This paper shows how to implement state of the art methods for performing LUPI and proposes a new technique. Furthermore a novel approach to the application of LUPI is introduced and implemented.

Contents

List of Figures

Nomenclature

α	Lagrange Multiplier
$(\mathbf{0})$	A column vector of 0's of length ℓ.
$(\mathbf{0}_M)$	An ℓ by ℓ matrix of 0's.
$(\mathbf{I})$	Identity Matrix
β	Lagrange Multiplier
ℓ	number of training samples
γ	regularisation parameter
$\mathcal{L}$	Lagrange Function
ν	Lagrange Multiplier
$\phi(\mathbf{x})$	mapping from one feature space to another
$\mathbf{x}_i^* \in \mathcal{X}^*$	privileged input in Privileged Input Space
$\mathbf{x}_i^{**} \in X^{**}$	secondary privileged input in Secondary Privileged Input Space
$\mathbf{x}_i \in \mathcal{X}$	input in Input Space
ξ_i	slack Value
ζ	combined slack value across both $\mathbf{X}$ and $\mathbf{X}^*$
b	offset from the origin
b^*	Slack Variable in the Privileged Input Space
C	regularisation parameter
$f(\mathbf{x})$	a function of $\mathbf{x}$
$g(\mathbf{x})$	a function constraining $\mathbf{x}$
h	VC Dimension
$K(\mathbf{x}, \mathbf{z})$	Kernel function over $\mathbf{x}$ and $\mathbf{z}$
N_Z	The number of elements in set Z
O	order of magnitude up to logarithmic factor
X	Collection of inputs
X^*	Collection of privileged inputs

Y	Collection of outputs
$y_i \in \mathcal{Y}$	output in Output Space
Y_{row}	The vector Y represented as a row vector. Y^T

Preface

I feel privileged to have been introduced to LUPI. Not only has it proved to be a challenging topic, but I have found it to be a rewarding and rich area to explore. The initial scope for this project was to understand and write about a recent (2015) paper, with the extension of implementing the technique described. This proved tricky as the paper was incredibly terse and contained some misleading descriptions, which are highlighted in Appendix F. As such I have had to derive all implementations myself. The implementations have all been built from scratch requiring a level of math beyond that which has been required in any computer science module. Understanding how privileged information is incorporated into classification techniques has provided me with a level of understanding about the nature of privileged information. In chapter 4 I try to demonstrate how privileged information effects our classifiers and 'lift the lid' on what's occurring in privileged space graphically. To my knowledge this is the first time that this has been attempted.

Furthermore, the SVM$_\Delta$+ classifier built in this paper has been used by the SMiLe CLiNiC for an upcoming paper, a new classifier has been invented, implemented and demonstrated (see chapter 5), and a novel application for privileged information has been implemented which warrants further exploration (chapter 8).

The code written for this paper has been made publicly available[1] and is referenced in appendix D.

[1] Except where requested

Chapter 1

Introduction

The afterword [1] to the second edition of V.Vapnik's book "Estimation of Dependencies Based on Empirical Data" makes reference to an advanced learning approach which Vapnik calls "Learning Using Hidden Information". This approach takes advantage of a multiple space input to the Support Vector Machine (SVM) and this new SVM was christened the SVM+. In 2009 the paradigm was renamed "Learning Using *Privileged* Information" (LUPI) as it was felt that this better represented the nature of the idea. Privileged implying that the information is of benefit to us and that it will not necessarily be available to others; as opposed to hidden which suggests some sort of hiding of extra information. Along with this re-name came further experimentations into this area, such as the *d*SVM+, a forerunner to the idea of Similarity Control investigated later in this report. This was followed up in 2015 with the paper "Learning Using Privileged Information: Similarity Control and Knowledge Transfer" in which two further techniques for taking advantage of privileged information were introduced. [22]

Privileged information refers to an additional set of features that are available to the classifier at training time, but crucially will not be available to the classifier when classifying new data (hence the multiple input spaces). There are many reasons why this might be the case in real life, the privileged information may be expensive, it may be sensitive in nature, or simply cannot be available at when making predictions. However, a notable feature of LUPI is that it is ubiquitous. For almost any Machine Learning problem there is privileged information that can be provided and as such it is important that we investigate ways to maximise its potential. Consider any problem for which you may wish to seek a machine learning solution. There will almost certainly exist available information that will help with the problem that won't necessarily be available when making predictions in the future. An example may be classifying images as sunny or not sunny. We want to make predictions based solely on the image, but at training time we could use meta-data about the picture to learn a better decision rule[2].

This report is an investigation into the approaches and applications of the Learning Using Privileged Information (LUPI) paradigm. The SVM+ and Similarity Control are demonstrated along the SVMu; an original proposal that seeks to unify the approaches of both Vapnik's Similarity Control and the SVM+. A further method called Knowledge Transfer is also introduced and discussed, though not implemented.

Support Vector Machines (SVM) are one of the most widely used and powerful classifiers and most existing LUPI methods are expansions on this technique. Therefore this report will focus on the SVM and seeks to take the reader from a cursory overview of machine learning (specifically binary classification and supervised learning) through to a more technical definition of LUPI. This is before chapter 3, the background section which gives an overview and implementation of various SVMs which will be required to realise the SVMu. The methodology for testing its performance is explained in

[1] Titled "Empirical Inference Science"

[2] Or a text-description of the picture, or even a video of an interpretive dance piece of how the picture made a viewer feel!

chapter 6, before a discussion of the findings in chapter 7.

1.1 Supervised Machine Learning - Binary Classification

Machine Learning is an umbrella term which encompasses many techniques but this report deals exclusively with supervised learning, specifically investigating binary classification. Supervised learning tries to find a function that maps from a series of training inputs to known outputs. These outputs can be continuous (regression), or discrete (classification). Our goal is then to find the function that generalises with minimum error to previously unseen data. This is different to unsupervised learning where the input data is given without the class label. The goal is then to find some relationships in the data, whether by clustering like points together or finding likely anomalies in the data.

Classification models tend to follow a similar approach to each other. We are given independent and identically distributed (iid) pairs of data of the form $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)$ where $\mathbf{x}_i \in X$ and $y_i \in \{-1, +1\}$. This means that the pairs have the same probability distribution (there are no biases in the data) and are mutually independent from each other. Additionally, it imposes that the structures of the feature vector $\mathbf{x}$ and class label y are always consistent throughout. Using these iid pairs, also referred to as training data, a function is learned ($f(\mathbf{x})$) that minimises incorrect classification. We then use this learned function on new, unseen values of $\mathbf{x}$ to make our best guess at the classification it belongs to. Broadly, this idea can assign a mapping from a set of training inputs to many different class labels, however, we are looking into binary classification where we only consider two possible classes. The binary nature has some useful properties. Not only is it simpler (so quicker to compute), it can also be described using relatively simple linear algebra[3]. Binary classification can also be extended to classify more than the two classes using One vs All (OvA) or One vs One (OvO) techniques. These methods involve breaking a multiple class classification problem into multiple binary classifications, comparing one class against another (OvO), or one class against all the other possible classes (OvA). Whilst there has been an effort to incorporate solving more than two classes in an SVM's objective function [4], they tend to perform no better than OvA or OvO and take much longer to train [15].

1.2 Motivation

With supervised learning, when we learn a function that maps from the input to the output we are left with a limitation. What if we have extra, relevant information that would improve the performance of the function? Traditional machine learning approaches cannot take advantage of this. They simply try to find a mapping from input $\mathcal{X}$ to output $\mathcal{Y}$. This works well when we have all relevant information, such as trying to predict Body Mass Index from the inputs Height, Weight, Age, and Gender, because all the required information for the underlying function is supplied in the inputs. This wouldn't work as well if we only supplied the Height and Weight of users to the prediction function. All the information which is required to generate the closest approximation to the underlying rule isn't supplied. Even if we had this information available for our training data, with our traditional classifiers we simply cannot incorporate this extra information when learning our mapping function. The illustration below of inputs to outputs demonstrates that we are aiming for the same output, just with an additional set of inputs.

Traditional Machine Learning

$$\text{input: } (X, Y) \implies \text{output: } f(x) = y$$

LUPI

$$\text{input: } (X, \text{Extra Information}, Y) \implies \text{output: } f(x) = y$$

[3] Multiclass classification can also be described using linear algebra, it's just more complex.

[4] Weston And Watkins, Vapnik; Lee, Lin and Wahba; Crammer and Singer etc.. - [15]

Why would we want to use privileged information when training a learning model? Surely we're better learning a model in the space that we're going to be making predictions in? The answer comes from the machine learning axiom that more data is better. If the privileged information helps to find an approximation closer to the true function than we could learn ourselves why would we not take advantage of it? Vapnik's inspiration however comes from trying to improve the SVM, particularly in regard to one aspect: the rate of convergence.

1.2.1 Rate of Convergence

The rate of convergence refers to the rate at which we tend towards a minimal generalization error bounded by the empirical error and the functional complexity[5]. Let us consider an SVM where we are trying to build a decision boundary for separable data. Suppose there exists a function $f'(x)$ that can optimally separate our separable data. Vapnik-Chervonenkis theory (VC theory) states that using an SVM, the rate of convergence to this optimal solution will have the order of magnitude $\frac{1}{\ell}$, where ℓ is the number of data points [22]. Simply put, the more data points available, the closer our learned solution $f(x)$ will get to the optimal solution for this problem $f'(x)$.

Now let us consider building an SVM for non-separable data. Suppose there is a function $f''(x)$ that can separate this data. VC theory says that finding the optimal SVM solution will have a rate of convergence with the order of magnitude $\frac{1}{\sqrt{\ell}}$. We need to see more training examples for our learned solution $f(x)$ to get closer to the optimal solution for this problem $f''(x)$. For the same size problem, Vapnik and Izmailov point out [22], the separable case would require 320 training examples to find an optimum solution. The non-separable case on the other hand would require 100,000 training examples (see fig. 1.1). This is caused by the non-linearly separable case having to learn an extra variable per data point called the slack, which is described in section 3.1.2. In fig. 1.1 we can see that generally, we will always asymptotically converge on the optimal solution. However, for a fixed given number of training samples (for example 32 in fig. 1.1), the faster converging classifier will perform better. This gap in the rate of convergence is what Vapnik attempts to close by introducing the concept of a teacher.

1.2.2 Intelligent Teacher

The non-separable case is a sticking point for the SVM. Generally, predictions are very fast, being $O(n)$ where n is the number of training data points[6], but training is slow[7], being $O(n^2)$ when solving the dual form. To counter this we consider what if there were an oracle that could supply us with extra knowledge? An oracle could tell us the distance by which the margin has been violated. This margin violation is called the slack, or $\xi_i, \forall i = 1 \dots \ell$. In addition to the training data points and their labels, the oracle would make the slack available to us at training time. This changes our training data pairs to training data triplets of the form $(x_1, \xi_1, y_1), \dots, (x_\ell, \xi_\ell, y_\ell)$.

If we were to learn an SVM solution on non-separable data where we already knew the slack value, the rate of convergence falls to that of an SVM solution on separable data. This is clearly a massive improvement, but it has the drawback that no such oracle exists. Instead, the best we can do is emulate this oracle. The problem is that getting the slacks for our input space X would involve learning a perfect SVM solution in $\mathcal{X}, \forall x \in X$, which is what we're trying to get to. This would be akin to using or knowledge of "the perfect function" to emulate itself - there would be no learning to be done. Perhaps we could learn the slack value in input space $\mathcal{X}$ from another set of data? The selection of this data then becomes important - if it doesn't help us to find the slack in $\mathcal{X}$, the rate of convergence can't be improved. What we want is data in a new space, $\mathcal{X}^*$, that models what the slack would be in "the

[5] As opposed to the convergence of an optimization problem - the rate that a minima can be found by Gradient Descent for example

[6] When using the 'Kernel Trick' in the dual form. See Appendices A and B

[7] This depends on if one decides to solve the primal problem or its dual. For further detail see appendix A

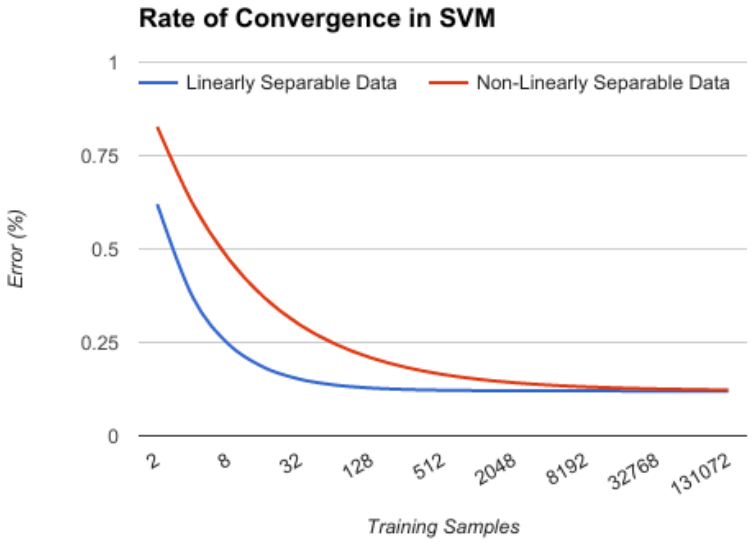


Figure 1.1: Demonstration of Rate of Convergence to an optimal solution (Linearly separable shifted to converge to same rate as Non-Linearly separable data for illustration purposes)

optimal solution" ($f(x)$). The quality in the selection of this extra data, Vapnik argues, differentiates the good 'teachers' from the poor ones [22].

1.3 Introduction to LUPI

LUPI is different to the classical learning model. Instead of training using iid pairs, we generate a function based on iid triplets $(\mathbf{x}_1, \mathbf{x}_1^*, y_1), \ldots, (\mathbf{x}_\ell, \mathbf{x}_\ell^*, y_\ell)$ where $\mathbf{x}_i \in X, \quad \mathbf{x}_i^* \in X^*, \quad y_i \in \{-1, +1\}$. Noticeably, we have extra feature space $\mathcal{X}^*$. This is a new feature space which is separate to $\mathcal{X}$ and crucially, is only available at training time. The aim is for this extra information (X^*) to fulfil the role of the oracle of the previous section and provide the slack in space $\mathcal{X}$. X^* should be drawn from a conditional probability function $P(x^*|x)$. This constrains $\mathcal{X}^*$ to be related to $\mathcal{X}$. When we come to use the classifier we have built we will not have access to this extra information, only the feature vector $\mathbf{x}$. Just as with the oracle, the actual classification function generated is still a mapping from $\mathcal{X}$ to $\mathcal{Y}$; it's just helped by the privileged space $\mathcal{X}^*$ to converge on the optimal solution faster.

Providing this extra information can be compared to the analogy of a classifier during training being a 'student' and at this time the function $P(x^*|x)$ takes on the role of the 'teacher'. By providing the student with examples (X) and answers (Y) the student has to take a brute force approach to learning - guessing without having any idea of context beyond the feature set given. Instead, if the teacher were to give analogy, description and reasoning as to why a decision should be made, the student will understand much quicker. Unfortunately, as in the real world, the teacher will not always be available to the student. Once the student has left the classroom they will have to put their learned knowledge into practice without a teacher to guide them. The student will have to predict an answer to an unseen problem without the context provided by the teacher. In this analogy the privileged information is the background context to help the classifier produce better accuracy. Transferring this analogy back to the LUPI classifier leads us to say the following. In many situations where machine learning is used there is extra information that the classifier isn't trained with because it won't be available at classification time. LUPI allows us to take advantage of this data.

For example, suppose we want to create an application that classifies images of food as either "healthy" or "unhealthy". Instead of training the classifier solely on example images of food and their

class label, we also use the ingredients list that we have available for the images we have provided. This information is only available for the images used during training and will not be available to users of our application when they submit pictures of their food to be classified as healthy or not. Our theory is that there is some rule in the space of the recipes that will help us to make better predictions in classification of the user's photograph. There are many reasons why the recipes wouldn't be available at prediction time, such as wanting to classify images of food sourced from Twitter, or by scraping a particular web site. However, in general, privileged information (by definition) is specialised and specialised data tends to be expensive.

As discussed in section 1.2.1, it's important to note that the goal of LUPI is to converge to the optimal decision function $f(\mathbf{x})$ quicker than more traditional machine learning approaches, rather than to find a different function. This is reflected in the bounds on the rate of convergence which is in the order $O(h/\ell)$ for linearly separable data, where h is the VC-Dimension of the set of functions $f(\mathbf{x})$ belongs to. Non-linearly separable data however has a convergence bound in the order $O(\sqrt{h/\ell})$. As already stated, the initial motivation for LUPI came about from trying to improve the bound on convergence in the non-linearly separable case rather than get to another result. As such, the performance of the classifier is directly comparable to others which do not take advantage of privileged information (see fig. 1.1).

1.3.1 Comparing LUPI to Other Learning Paradigms

LUPI can be compared to other methods that incorporate additional information. Example include

Knowledge-based learning Using domain specific facts to infer predictions

Learning with hints Where hints is an umbrella term ranging from extra examples to giving the underlying function we're trying to learn

Context-sensitive models Where contextual features are those that are not useful when considered alone, but can be useful when combined with other features

Multi-view learning Where models across different spaces are co-trained and predictions can be made in both spaces

Model distillation Where a more complicated model is learnt, then semantically replicated by a simpler model

There have been workshops [8] to try and encourage sharing between researchers in these areas and this is an ongoing challenge, but the primary difference is the motivation. LUPI is motivated theoretically, whereas many other approaches are motivated by finding practical solutions[9]. Privileged information is unique in that we're not simply adding additional data to the problem, we're adding intelligently selected data that can be used to learn the slack in the input space $\mathcal{X}$. The final learnt decision boundary is still decided in $\mathcal{X}$, though is influenced by the slack generated in $\mathcal{X}^*$. Unlike in other paradigms, privileged information ($\mathcal{X}^*$) **must** be about $\mathcal{X}$. Whilst there has been research to show that LUPI classifiers can take advantage of random data [16], this is not an example of privileged information. Privileged information **must** be drawn from a conditional probability function $P(x^*|x)$. This definition helps us to make further reasoning about the paradigm.

Let's consider the example of looking for cancer cells with our input space being biopsy images and our privileged information being a surgeon's notes. We can notice that the biopsy image space, $\mathcal{X}$, is a series of pixels and can be described as universal. We can not only describe biopsy images in this space, but also artwork or text. The input space $\mathcal{X}^*$ is constrained to be a bridge between the input space and the label, meaning the VC-Dimension of admissible rules we can learn in $\mathcal{X}^*$ must be smaller than those admissible in $\mathcal{X}$. Note that this doesn't mean that $\mathcal{X}^*$ has to better describe

[8] http://smileclinic.alwaysdata.net/ijcai16workshop/

[9] Some developing theories as time goes by

$\mathcal{X}$, it can just give relevant information. Crucially though, we should be able to learn a rule in $\mathcal{X}^*$ that emulates the margin violation in $\mathcal{X}$ of the optimal solution. So instead of better describing $\mathcal{X}$, $\mathcal{X}^*$ should be about describing the relevance of $\mathcal{X}$ to the label, $\mathcal{Y}$. In this way, Privileged Information is different to other approaches.

1.4 Project Aims

The aims of this project are to understand, implement, and explore state-of-the-art machine learning techniques, and the effect of different techniques to make use of Privileged Information. The SVM+, Similarity Control (SVM_Δ+ and Margin Transfer) and Knowledge Transfer are implemented from the ground up and explored, and a unifying approach to the SVM+ and Similarity Control techniques is introduced and implemented. A novel application for privileged information is suggested and implemented in chapter 8 which poses an interesting area for further research. All code used has been written from scratch, except library calls, which are explicitly cited.

Chapter 2

Professional Considerations

This research may involve the use use of data that could potentially be of a sensitive nature. As such, only publicly available, anonymous data sets that are compliant with the Data Protection Act 1998 (and the BCS Code of Conduct 3.a) [1] will be used if and when required. This will also give the benefit of being able to ensure results are reproducible by any who so wish, with links to the data sets used available in appendix D of this paper.

In accordance with the BCS Code of Conduct 2.a, the research area is one that the author can claim some competence in having studied the Machine Learning module taught at the University of Sussex. [2]

All claims and viewpoints of relevant parties will be well documented and referenced as to avoid injury to others or their reputation.[3] [4]

If possible, in encouragement and support of the professional development of others, any advancements in implementations will be added to appropriate libraries. [5]

Some extensions of this project are regarding the use of sensitive features (such as race, sex etc) as privileged information (meaning that these features cannot be used at "decision" time). This extension builds upon the work done learning fair classifiers [cite] and will be in direct support of the BCS Code of Conduct section 1.c., helping to remove discriminations based on sensitive features. [6].

[1] BCS Code of Conduct 3.a. "You shall carry out your professional responsibilities with due care and diligence in accordance with the Relevant Authority's requirements whilst exercising your professional judgement at all times."

[2] BCS Code of Conduct 2.a. "You shall only undertake to do work or provide a service that is within your professional competence."

[3] BCS Code of Conduct 2.b. "You shall **NOT** claim any level of competence that you do not possess"

[4] BCS Code of Conduct 2.f. "You shall avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction."

[5] BCS Code of Conduct 4.f. "You shall encourage and support fellow members in their professional development"

[6] BCS Code of Conduct 1.c. "You shall conduct your professional activities without discrimination on the grounds of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age or disability, or of any other condition or requirement"

Chapter 3

Background

For an introduction to duality with regard to finding an SVM solution, see appendix A.

3.1 Support Vector Machines

Our classification problem is defined in terms of finding $f(\mathbf{x})$, a function that 'splits' the data sets into two 'classes' in feature space $\mathcal{X}$ such that incorrect classifications are minimised. One approach to this would be to generate a line (or hyperplane) between the data and amend it until the condition is met, so our data is classified as well as possible, then stop. This would be a perceptron, a simple linear classifier. However, this method is flawed as it doesn't go far enough - the conditions are too relaxed and as such it is too sensitive to fluctuations in the data. This doesn't give the best results on the unseen data that ultimately, we want to perform well on. We usually want a more general function to be learned, so that when classifying we not only perform well in training, but also in practice. When the classifier comes to be used in the future on completely unseen data points, we assume that are going to be classifying varying perturbations of our training samples. There is a balance to be struck between training to our sample data too well (over-fitting, or low bias, high variance) and not training well enough (under-fitting, or high bias, low variance). This balance is known as the bias-variance trade-off. To combat this, what we really need is to add an additional condition to our function, so that instead of just minimising incorrect classifications, we also need to maximise the boundary (margin) to the nearest point. This creates a separating plane with the least possible vulnerability to fluctuations in the data, giving better generalization for our unseen data. A classifier that works in this way is known as a Support Vector Machine (SVM).

Mathematically, SVMs are an optimisation problem - we seek to maximise the margin subject to classifying our training data correctly[1]. The margin is described as the distance (d) from a feature vector $\mathbf{x}$ to the separating hyperplane (H) and is formally described in equation 3.1.

$$d(\mathbf{x}, H) = \frac{\mathbf{w} \cdot \mathbf{x}}{||\mathbf{w}||} \tag{3.1}$$

Thus to compute the maximum distance to the hyperplane, we need to minimise the weight vector $\mathbf{w}$. So whilst we could make the objective function to minimise $\mathbf{w}$ in equation 3.1 the general case to do this is written as in equation 3.2.

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \tag{3.2}$$

The weight vector $\mathbf{w}$ is a vector of weights (or values) that correspond to each feature in $\mathbf{x}$ and can be thought of as determining the 'importance' of each feature. When we train a classification model, we learn the best approximation we can for determining the 'importance' weight of each feature in our inputs. Once the margin is maximised, we treat the distance from the margin to the decision boundary

[1] The non-linearly separable case is considered in section 3.1.2

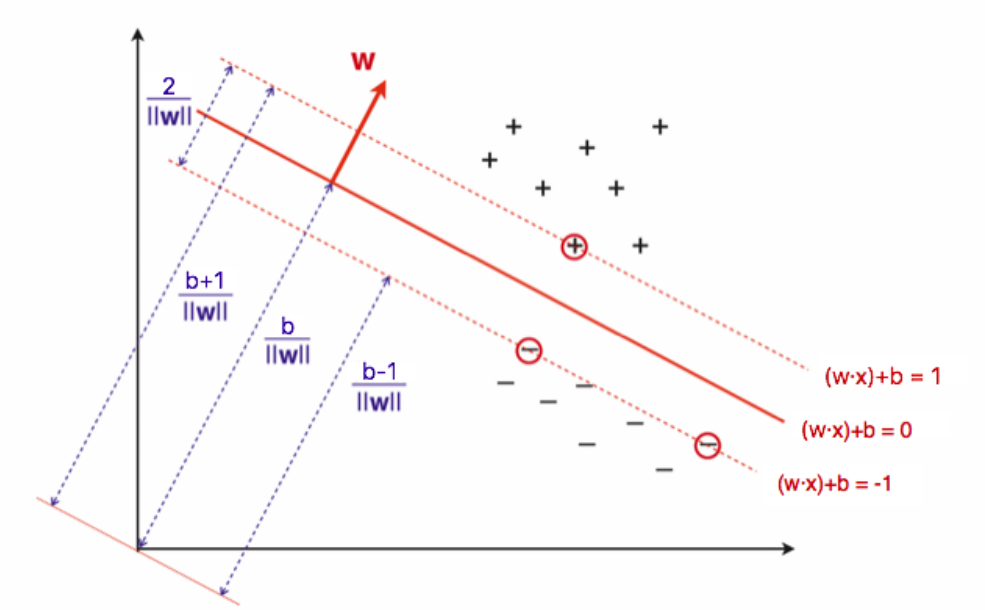


Figure 3.1: Geometry of an SVM showing definitions for the decision boundary and margins.

as 1, so $\frac{1}{||\mathbf{w}||} = 1$. Although this could be any value as long as we were consistent, 1 is chosen by convention [3]. The decision boundary as shown in fig. 3.1 is $((\mathbf{w} \cdot \mathbf{x}) + b)$ where b is the bias (the offset from the origin) and lies equidistant from the support vectors of both classes. The support vectors are the data points that lie one margin distance away from the decision boundary in this linearly separable case. So the support vectors for one class are where the decision (classification) function equal the margin distance $(((\mathbf{w} \cdot \mathbf{x}) + b) = 1)$. For the other class, the support vectors are where the decision (classification) function equals the negative margin distance $(((\mathbf{w} \cdot \mathbf{x}) + b) = -1)$. They are called support vectors because each data point is a set of feature vectors and they define where the decision boundary lies. They 'support' the decision boundary. In fact, you could remove all other data points apart from the support vectors and you would still generate the same boundary. [2]

This leads to quite an elegant solution for ensuring all data points are correctly classified. Suppose $(\mathbf{w} \cdot \mathbf{x}) + b$ produces the scalar value 1.86. To see if this is the correct classification we can multiply it by the classification it was given y. If the classification is correct then the answer will be positive and in excess of the margin (1). Similarly, had the scalar value produced been -1.86, multiplying it by the correct classification (-1) would again produce a positive value in excess of the margin. An incorrect classification produces a negative value. This gives a neat constraint to our objective function. Whilst minimising $\mathbf{w}$ we need to ensure that the decision function multiplied by the classification produces a value greater than the margin (or equal to it for the support vectors). This gives us the constraint $y(\mathbf{w} \cdot \mathbf{x}) + b \geq 1$.

Once the SVM is trained, classification is a simple check. If the dot product of the learned weight vector and the feature vector plus the bias is less than 0 (below the decision boundary), the new data-point considered to be of the negative class (Class -1). If it doesn't then it is of the positive class (Class +1) $(sign((\mathbf{w} \cdot \mathbf{x}) + b))$. Here the *sign* function returns 1 if the value is positive, else returns -1.

Solving the constrained objective function (eq. 3.3) however, is a difficult problem to solve computationally, particularly if the number of features ℓ is large. Instead, we take advantage of duality by converting the problem from its primal form to its Lagrangian dual. Not only does this allow us to approach the problem from another direction, but it also incorporates the constraints into the objective function.

[2] That's not to say that a specific decision boundary can only be described a unique combination of support vectors. For some problems a reduced number of support vectors can be found (Burges 1996)

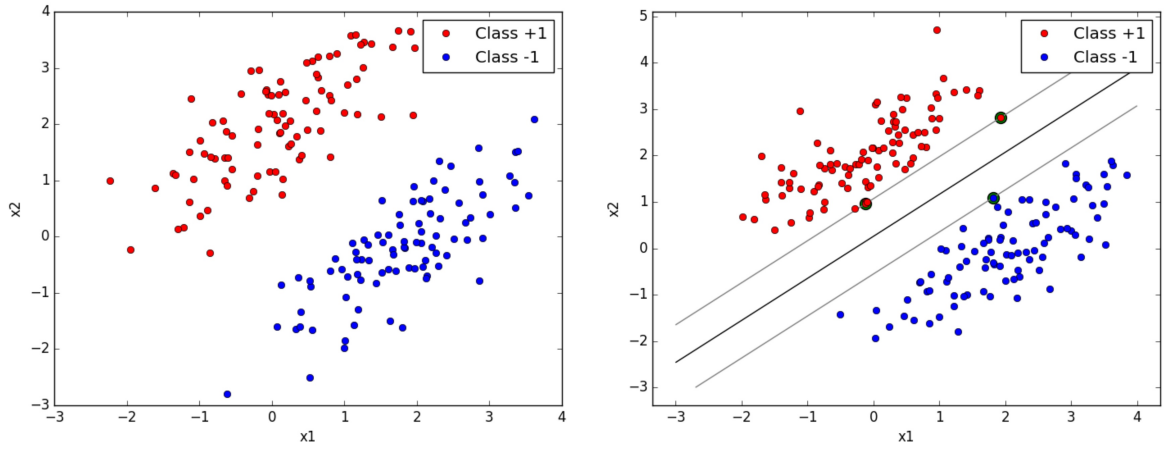


Figure 3.2: (Left) Linearly separable data. We seek to maximise the margin between the two classes. (Right) A Support Vector Machine, classifying the data with the maximum margin between classes. The support vectors are located on the margin and have been highlighted.

3.1.1 Hard Margin SVM

In the previous section we discussed looking for $f(\mathbf{x})$, where the number of incorrect classifications is a minimum and with a maximum margin between classes. To do this, let's take the simplest scenario, which is that our data is linearly separable as in fig. 3.2.

We construct a plane that satisfies

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \end{aligned} \tag{3.3}$$

This is the objective function, the primal form of the Hard Margin SVM. To compute the optimal solution efficiently we need to convert the problem to its Lagrangian Dual. Notes on this can be found in appendix A.

- Primal Problem

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \end{aligned} \tag{3.4}$$

- Dual Problem

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\text{subject to} \quad \alpha_i \geq 0 \\ &\text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{3.5}$$

Where $\alpha_i \forall i = 1, \ldots, \ell$ are Lagrange multipliers. Implementation of the Hard Margin SVM is included in appendix D.

Once we have a solution to the dual problem, we can calculate the weight and the bias.

$$
\begin{aligned}
w &= \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \\
b &= \frac{1}{N_S} \sum_{s \in S} (y_s - \sum_{m \in S} (\alpha_m y_m K(\mathbf{x}_m, \mathbf{x}_s))
\end{aligned}
\tag{3.6}
$$

Where S is the set of support vectors, data points whose corresponding value of α is greater than 0 ($\alpha > 0$).

3.1.2 Soft Margin SVM

In theory, the Hard Margin SVM should work very nicely. However, in theory, theory and practice are the same, in practice they are not. When it comes to real data sets, we rarely achieve linear separability; often data overlaps as in fig. 3.3. As such, we need to consider an SVM where the constraint about maximising the margin is maintained, whilst still allowing for misclassification. After all there may be anomalies in the data, or the data may not be able to represent the underlying decision function entirely. To do this, we consider a "slack" value ξ for each data-point that seeks to correct misclassification, ensuring the first constraint in equation 3.7. The classification function of the SVM ($\sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b$)[3] produces a scalar value. In the Hard Margin SVM we ensure that this is always ≥ 1 because we only consider linearly separable data. As we are now looking at data with some overlap, we have to allow for the classification of the SVM generating a scalar value less than 1. The maximum value of either 0 or 1 minus the scalar value produced is the slack (ξ_i). For every ξ_i there are two scenarios. Either $\xi_i = 0$, because that data point is either on the appropriate margin, or further away from the decision boundary than the margin ($(\sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b) \geq 1$). Or, $\xi_i > 0$ which occurs when the data point lies on the incorrect side of the margin. There are two possible reasons for this margin violation. The data point lies on the correct side of the decision function, but lies inside the margin. Or, the data point lies on the incorrect side of the decision function. In both of these situations (where $\xi > 0$) the beauty is that they can be incorporated into the objective function. We can sum together the total ξ and add this to our function that we're minimising. [4] Usually when we do this we introduce a new variable C which we multiply by the sum of the slack values. This is our 'regularisation' parameter and gives us control over the importance which we place on the margin violations. If we give a high value of C, we are placing such a high value on minimising incorrect classifications that we effectively have a Hard Margin SVM. If on the other hand we give a very low value to C we place little importance on margin violations and end up with a very soft margin indeed. By altering the C parameter we obtain a method of affecting how well our classifier fits to the data.

- Primal Problem

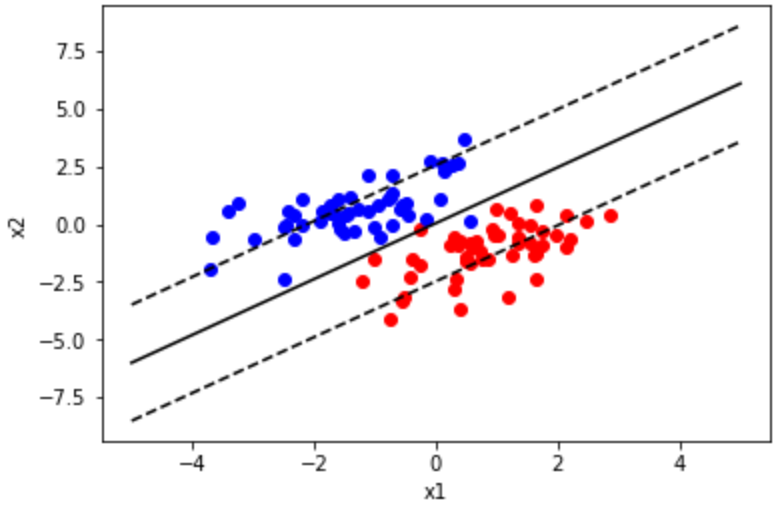
$$
\begin{aligned}
\min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{\ell} \xi_i \\
\text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\
\text{and} \quad & \xi_i \geq 0
\end{aligned}
\tag{3.7}
$$

- Dual Problem

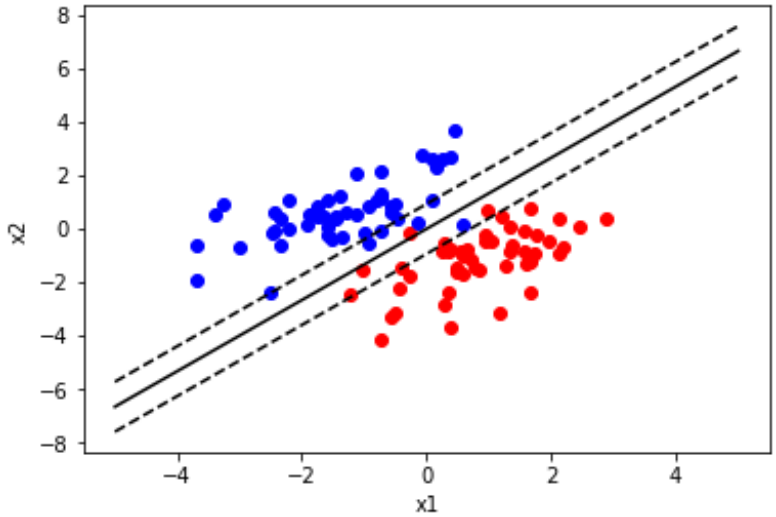
$$
\begin{aligned}
\max_{\alpha} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\
\text{subject to} \quad 0 \leq \alpha_i \leq C \\
\text{and} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0
\end{aligned}
\tag{3.8}
$$

[3] The function $K(x, x)$ refers to the kernel trick. See appendix B.

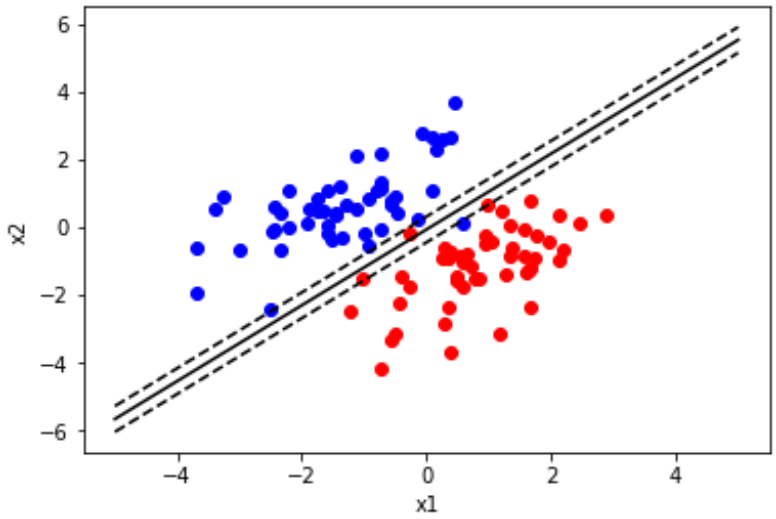
[4] We add the error function 'hinge-loss' and seek to minimise this error. $\xi_i = \max(0, 1 - (y_i(\langle w, x_i \rangle + b)))$



(a) Regularisation Parameter C set to 0.01



(b) Regularisation Parameter C set to 1



(c) Regularisation Parameter C set to 100

Figure 3.3: We have overlapping, but largely linearly separable data. We seek to maximise the margin between the two classes whilst keeping the number of margin violations to a minimum. a-c show a Soft Margin Support Vector Machine, classifying the data with the maximum margin between classes whilst maintaining the smallest margin violations. In these examples the effect that the regularisation parameter C has on the decision boundary is demonstrated.

An implementation of the Soft Margin SVM is included in appendix D.

Once we have a solution to the dual problem, we can calculate the weight and the bias.

$$\begin{aligned} w &= \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \\ b &= \frac{1}{N_S} \sum_{s \in S} (y_s - \sum_{m \in S} (\alpha_m y_m K(\mathbf{x}_m, \mathbf{x}_s)) \end{aligned} \tag{3.9}$$

Where S is the set of support vectors, data points whose corresponding value of α is greater than 0 and less than or equal to C ($0 < \alpha_i \leq C$).

Chapter 4

Implementation of LUPI

4.1 Current Approach

The current approach to LUPI is the SVM+ which is detailed below. Vapnik introduced two new LUPI techniques, Similarity Control and Knowledge Transfer in 2015 [22] and are described later in this chapter. These techniques represent the current state of the art. Later, in chapter 5, a new technique is introduced which unifies the SVM+ and Similarity Control.

4.1.1 SVM+

The SVM+ was introduced to take advantage of additional, privileged data $\mathbf{x}^*$ that is only available at training time. As such instead of just optimising over $\mathbf{w}$ and b in the non-privileged space ($\mathbf{x}$), it also seeks to minimise $\mathbf{w}^*$ and b^* in the privileged space ($\mathbf{x}^*$).

$$\begin{aligned} \min_{\mathbf{w},b,\mathbf{w}^*,b^*} \quad & \frac{1}{2}(||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2) + C\sum_{i=1}^{\ell}((\mathbf{w}^* \cdot \mathbf{x}_i^*) + b^*) \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - ((\mathbf{w}^* \cdot \mathbf{x}_i^*) + b^*) \\ \text{and} \quad & ((\mathbf{w}^* \cdot \mathbf{x}_i^*) + b^*) \geq 0 \end{aligned} \tag{4.1}$$

In the SVM+, γ is introduced as a hyper-parameter. It is similar to C, but inverted with regard to the 'importance' placed on the privileged space. A low γ will place a higher weight on optimising the margin generation function that occurs in the privileged space, $\mathcal{X}^*$. A higher γ places a greater weight on the decision function in the non-privileged space, $\mathcal{X}$.

- Dual Problem

$$\begin{aligned} \max_{\alpha,\beta} \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i \cdot \mathbf{x}_j) - \frac{1}{2\gamma}\sum_{i,j=1}^{\ell}(\alpha_i + \beta_i - C)(\alpha_j + \beta_j - C)(\mathbf{x}_i^* \cdot \mathbf{x}_j^*) \\ \text{subject to} \sum_{i=1}^{\ell}(-\alpha_i - \beta_i + C) = 0 \\ \text{and} \quad \sum_{i=1}^{\ell} y_i\alpha_i = 0 \\ \text{and} \quad \alpha_i \geq 0 \\ \text{and} \quad \beta_i \geq 0 \end{aligned} \tag{4.2}$$

Once we have a solution to the dual problem, we can calculate the weight in both spaces.

$$
\begin{aligned}
w &= \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \\
w^* &= \frac{1}{\gamma} \sum_{i=1}^{\ell} \mathbf{x}_i^* (\alpha_i + \beta_i - C)
\end{aligned}
\tag{4.3}
$$

The bias is a little trickier. By KKT conditions,

$$
\alpha_i \rangle 0 \implies y_i(< w, \mathbf{x}_i \rangle + b) = 1 - (< w^*, \mathbf{x}_i^* \rangle + b^*)
$$

resulting in

$$
\begin{aligned}
\alpha_i > 0, y_i = 1 \implies & (\langle w, \mathbf{x}_i \rangle + b) = 1 - (\langle w^*, \mathbf{x}_i^* \rangle + b^*) \\
& = \langle w, \mathbf{x}_i \rangle + b = 1 - \langle w^*, \mathbf{x}_i^* \rangle - b^* \\
& = b + b^* = 1 - \langle w, \mathbf{x}_i \rangle - \langle w^*, \mathbf{x}_i^* \rangle \\
\alpha_i > 0, y_i = -1 \implies & -(\langle w, \mathbf{x}_i \rangle + b) = 1 - (\langle w^*, \mathbf{x}_i^* \rangle + b^*) \\
& = -\langle w, \mathbf{x}_i \rangle - b = 1 - \langle w^*, \mathbf{x}_i^* \rangle - b^* \\
& = b - b^* = -1 - \langle w, \mathbf{x}_i \rangle + \langle w^*, \mathbf{x}_i^* \rangle
\end{aligned}
\tag{4.4}
$$

Then to find the bias in both spaces

$$
\begin{aligned}
b &= \frac{(b + b^*) + (b - b^*)}{2} = \frac{\frac{1}{N_S} \sum_{m \in S} -2\langle w, \mathbf{x}_m \rangle}{2} \\
b^* &= \frac{(b + b^*) - (b - b^*)}{2} = \frac{\frac{1}{N_S} \sum_{m \in S} 2 - 2\langle w^*, \mathbf{x}_m^* \rangle}{2}
\end{aligned}
\tag{4.5}
$$

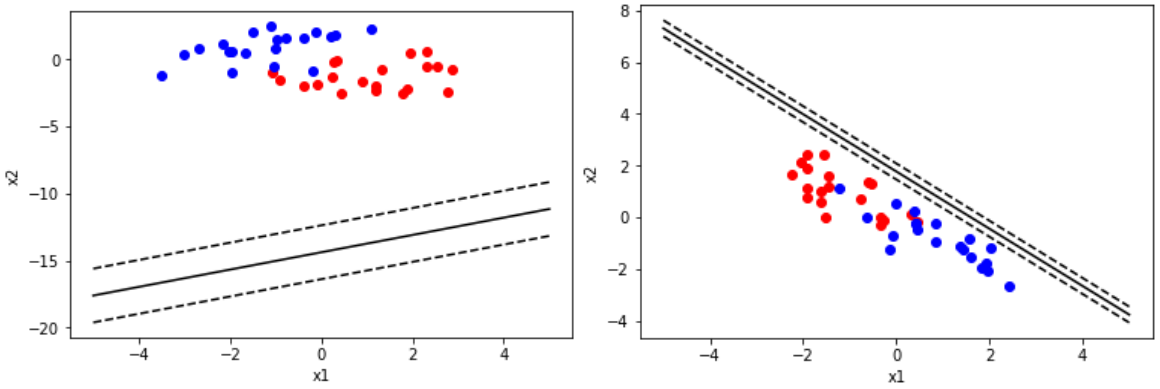
However, we need to consider that we are using the 'kernel trick', so we replace w with the representation found in the dual form, $\sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$. This gives

$$
\begin{aligned}
b &= \frac{(b + b^*) + (b - b^*)}{2} = \frac{\frac{1}{N_S} \sum_{m \in S} -2 \sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_m)}{2} \\
b^* &= \frac{(b + b^*) - (b - b^*)}{2} = \frac{\frac{1}{N_S} \sum_{m \in S} 2 - 2\frac{1}{\gamma} \sum_{i=1}^{\ell} (\alpha_i + \beta_i - C) K(\mathbf{x}_i^*, \mathbf{x}_m^*)}{2}
\end{aligned}
\tag{4.6}
$$

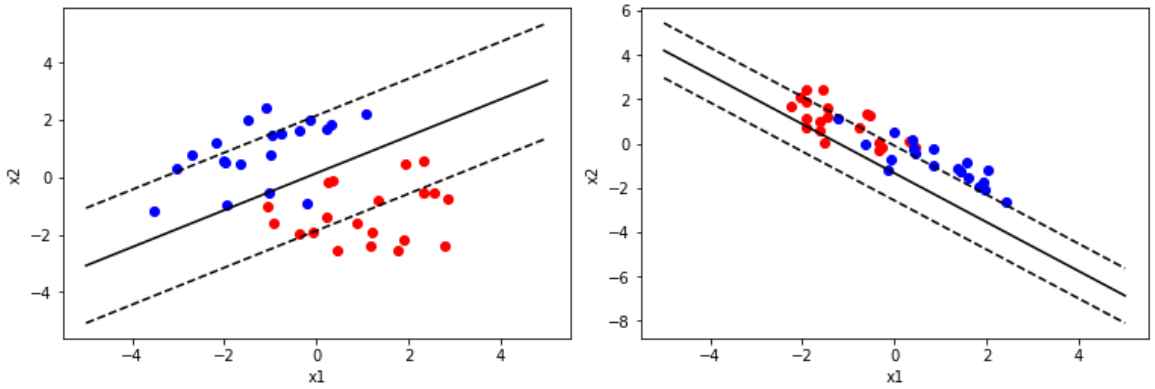
Where S is the set of support vectors, data points whose corresponding value of α is greater than 0. We didn't need to calculate w^* and b^*. No predictions are being made in privileged space $\mathcal{X}^*$. However, by doing this it allows us to visualise what's happening in the privileged space.

In fig. 4.1 we can see that whilst the decision function in X may be of the shape we expect, X^* is showing us something quite different. The key is to remember the purpose of space $\mathcal{X}^*$. We're not trying to learn a decision rule, but rather a margin representation rule. The distance to the boundary in $\mathcal{X}^*$ should represent the margin violation in X. Technically the margin doesn't exist in $\mathcal{X}^*$, however it has been drawn to show the effect of γ. When γ is small, placing more emphasis on the space $\mathcal{X}^*$, the margin is smaller. This happens because the units by which we are measuring the distance from the boundary gets smaller. Therefore, though the decision line in $\mathcal{X}^*$ can stay in the same place, the distance that each data point is to it increases (Though in practice, the boundary line moves to the optimal position). The opposite is true when γ increases, reducing the role of $\mathcal{X}^*$ to represent the margin in X.

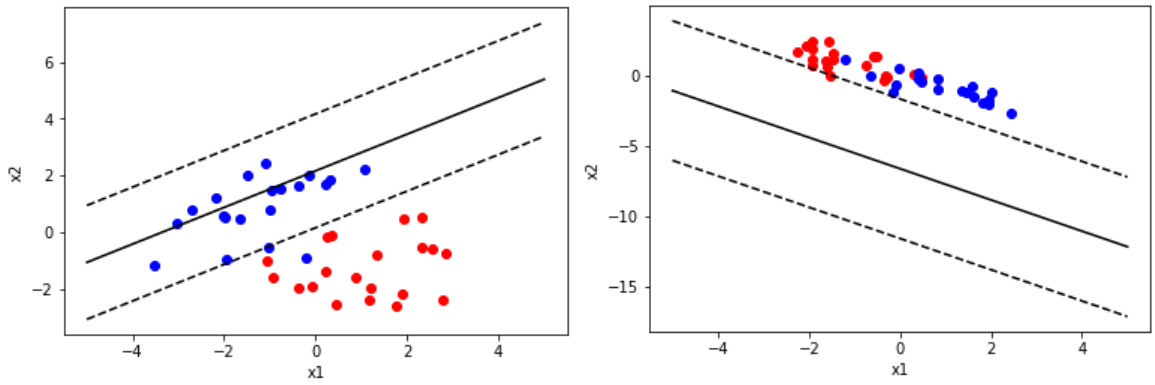
Figure 4.1 demonstrates the effect that γ has on the decision boundary in X. Previously we discussed that γ emphasizes the weight of the margin function in $\mathcal{X}^*$, but let's look a little closer at that. The value of γ effects the unit distance to the 'boundary' in $\mathcal{X}^*$. A smaller γ means a smaller unit distance, so conceptually, points are 'further away'. It's important to remember that the margin being



(a) $\gamma = 0.5$



(b) $\gamma = 1$



(c) $\gamma = 2$

Figure 4.1: (Left Hand Side) The decision space, X. (Right Hand Side) Privileged Space $\mathcal{X}^*$. We have overlapping, but largely linearly separable data. We seek to emulate the optimal dividing boundary in X between the two classes. a-c show SVM+ solutions with varying effects of the hyper-parameter γ. In these examples the regularisation parameter C is set to 1.

learnt in $\mathcal{X}^*$ (the right hand side of fig. 4.1) is only for support vectors in X. If the corresponding data point in X is not a support vector in X, the distance to the margin function boundary in $\mathcal{X}^*$ is ignored. In fig. 4.1(a) the decision boundary in X is a long way from the data points. This is represented by $\mathcal{X}^*$ having a small unit length making all points far from the margin function boundary.

This leads to the question, what makes good privileged information? Though more specifically the question should be, what makes good privileged information for the SVM+, because as we'll see with the introduction of Similarity Control, there is more than one approach to privileged information. For the SVM+ the ideal 'shape' of the privileged information would be with enough flexibility to produce a unique slack for each data point in X. So that would mean the data is well spaced, with high dimensionality, to give as much flexibility to the SVM+ as possible. Although in practicality this is unlikely to be the case, there will be some compromise and due to the closeness of some data points in X^*, some slacks in X will be incorrectly approximated. In effect, this is helping to regularise our data in X, preventing over-fitting to the training data.

The follow up question, is what makes bad data for the SVM+? Crucially, the SVM+ does not pay any regard to the class label in $\mathcal{X}^*$. In this space we end up with all points on one side of the margin boundary (usually[1]). If we had data the opposite of what the SVM+ is looking for, so densely packed, low dimensionality, say as a perfect classification style dataset, then the SVM+ will not be able to deal with it well at all. What we really need is a way of performing classification in the privileged space.

4.2 Similarity Control

During the next two subsections, the two methods of implementing Similarity Control will be described. Understanding and implementing Similarity Control as a mechanism to take advantage of LUPI is one of the principal investigations in this report. Similarity Control was an idea that Vapnik alluded to in his 2009 paper where he introduced the *d*SVM+, a first attempt at treating privileged information in a different way. In his *d*SVM+ Vapnik first learned an SVM solution in $\mathcal{X}^*$. The margin violation for each data point was then used as privileged information in the SVM+. Whilst the 2009 paper showed that this had favourable results it wasn't an elegant solution and showed a weakness in the SVM+, namely an inability to take the class label into account in privileged space.

Ideally what we would like to learn is some SVM solution that maximises the margin of correct classifications in X subject to the constraint that $y(\langle w, x\rangle + b) \geq 1 - [y(\langle w^*, x^*\rangle + b^*)]_+$ where $[u]_+$ denotes the maximum of 0 or u. This problem translates to "correctly classify X so that all points are greater than or equal to the margin minus the distance to the decision boundary of the corresponding point in X^*, which must be correctly classified if X violates the margin". The notation $[y(\langle w^*, x^*\rangle + b^*)]_+$ implies that $\mathcal{X}^*$ doesn't require correct classification for all points, so if a point in X doesn't fall within the margin we don't pay attention to the distance to the boundary in $\mathcal{X}^*$. Because of the notation $[u]_+$, this is no longer a linear optimisation as the constraint is piece-wise linear. As such, we have no way to effectively solve the problem. To combat this we construct a new problem which tries to model our idealized solution. This new problem imagines that there is a slack value in $\mathcal{X}^*$ as well as X and introduces a variable ζ. This variable ζ connects what the slack value would be across both spaces and can be thought of as $\zeta_i = \xi_i + \xi_i^*$. This leads to the constraint $y(\langle w, x\rangle + b) \geq 1 - y(\langle w^*, x^*\rangle + b^*) - \zeta$ where $y(\langle w^*, x^*\rangle + b^*) + \zeta \geq 0$. This is the essence of Similarity Control.

Similarity Control differs from the approach taken by the SVM+ in that it takes into account the class label of the privileged data when generating the distance to the slack in X. The constraint $y(\langle w^*, x^*\rangle + b^*) + \zeta \geq 0$ can have the value 0 for ζ_i when x_i^* is correctly classified. ζ will only be greater than 0 when $y(\langle w^*, x^*\rangle + b^*)\langle 0$ placing x^* on the wrong side of the boundary in $\mathcal{X}^*$. The value of ζ is the mechanism by which Similarity Control is enforced. Providing x_i is correctly classified and doesn't violate the margin in X, we don't pay any attention to privileged space or the value of ζ. It's only when x_i violates the margin in X that we consider $y((w^*, x^*) + b^*) + \zeta$. In this case there are

[1]The combination of hyper-parameters C and γ can allow us to force the SVM+ into some shapes it naturally doesn't want to be in.

two situations. Either x_i^* is correctly classified in $\mathcal{X}^*$, in which case the distance to the boundary in $\mathcal{X}^*$ should emulate the margin violation in X. Or, x_i^* is on the wrong side of the boundary. In this case, the value of ζ needs to compensate for the margin violation in both X and $\mathcal{X}^*$. A regularisation parameter will determine how much we penalise margin violation across both spaces.

Two methods of Similarity Control were introduced in the 2015 paper. They are the SVM$_\Delta$+ and a second approach which is named SVM$_\Delta$+: Simplified Approach. However, this is a long name and the technique is identical in spirit to the proposed "Margin Transfer" technique of Sharmanska et al [18], though the implementation is slightly different[2]. However, for the sake of brevity the terms Margin Transfer and SVM$_\Delta$+: Simplified Approach are used interchangeably throughout this report, though SVM$_\Delta$+: Simplified Approach is what's implemented. It would be an interesting future project to determine if there is any practical difference between the two techniques.

The general difference between Similarity Control and the SVM+ is this. If we consider that the SVM+ was learning a regression in $\mathcal{X}^*$, we can argue that Similarity Control learns a classifier in $\mathcal{X}^*$.

4.2.1 SVM$_\Delta$+

$$\begin{aligned} \min_{\mathbf{w},b,\mathbf{w}^*,b^*} \quad & \frac{1}{2}(||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2) + C\sum_{i=1}^{\ell}[y_i((\mathbf{w}^* \cdot \mathbf{x}_i^*) + b^*) + \zeta_i] + \Delta C\sum_{i=1}^{\ell}\zeta_i \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - y_i((\mathbf{w}^* \cdot \mathbf{x}_i^*) + b^*) - \zeta_i \\ \text{and} \quad & y_i((\mathbf{w}^* \cdot \mathbf{x}_i^*) + b^*) + \zeta_i \geq 0 \\ \text{and} \quad & \zeta_i \geq 0 \end{aligned} \tag{4.7}$$

The variable ζ has been introduced with respect to each data point and represents margin violation across both $\mathbf{X}$ and $\mathbf{X}^*$. Another hyper-parameter, Δ is also introduced. C continues to control the amount by which we punish margin violations. It controls the 'softness' of the margin. Δ is required to control the importance of margin violations across both spaces. From a practical standpoint, optimising over 3 hyper-parameters, C, Δ and γ, plus considering kernels makes the problem challenging. Vapnik suggests setting Δ to be "a sufficiently large value", rather than optimising over it. Let's look at why this would be. In $\mathcal{X}^*$, we're not learning a strict SVM. Instead of a typical SVM hinge-loss error function we're setting ζ to be the error from a hinge loss where the activation begins at 0 (see the second constraint in equation 4.7). Given this is the case it makes sense to try and make the data as classified as can be in $\mathcal{X}^*$. Fortunately we don't have to consider generalization of the classification in $\mathcal{X}^*$, so the bias-variance trade-off in $\mathcal{X}^*$ is not a concern as long as we get good generalization in X. Whilst Vapnik doesn't give a value for the Δ in his paper, for practical purposes[3] the maximum value of the range being searched over in C should be sufficient.

- Dual Problem

$$\begin{aligned} \max_{\alpha,\beta} \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i \cdot \mathbf{x}_j) - \frac{1}{2\gamma}\sum_{i,j=1}^{\ell} y_i y_j(-\alpha_i - \beta_i + C)(-\alpha_j - \beta_j + C)(\mathbf{x}_i^* \cdot \mathbf{x}_j^*) \\ \text{subject to} \quad \sum_{i=1}^{\ell} y_i\alpha_i = 0 \\ \text{and} \quad \sum_{i=1}^{\ell} y_i(-\alpha_i - \beta_i + C) = 0 \\ \text{and} \quad 0 \leq \beta_i \\ \text{and} \quad 0 \leq \alpha_i \leq -\beta_i + C + \Delta C \end{aligned} \tag{4.8}$$

[2]Both learn an SVM solution in $\mathcal{X}^*$, then learn an SVM solution in X using the slack from $\mathcal{X}^*$. How this information is used is where the difference occurs.

[3]With the caveat being that the minimum value of $\Delta C \geq 1$. If $\Delta C < 1$ it can overly constrain our problem

Once we have the solution to the dual problem, we can calculate the weight in both spaces.

$$\begin{aligned} w =& \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \\ w^* =& \frac{1}{\gamma} \sum_{i=1}^{\ell} y_i \mathbf{x}_i^* (\alpha + \beta - C) \end{aligned} \tag{4.9}$$

The bias in $\mathcal{X}$ is derived by KKT conditions.

$$\begin{aligned} \alpha_i \geq 0 &\implies y(\langle w, x \rangle + b + \langle w^*, x^* \rangle + b^*) + \zeta_i - 1 = 0 \\ \beta_i \geq 0 &\implies y(\langle w^*, x^* \rangle + b^*) + \zeta_i = 0 \end{aligned} \tag{4.10}$$

Resulting in

$$\begin{aligned} \alpha_i, \beta_i \rangle 0, y_i = 1 &\implies \langle w, x \rangle + b = 1 \\ &\implies b = 1 - \langle w, x \rangle \\ \alpha_i, \beta_i \rangle 0, y_i = -1 &\implies -\langle w, x \rangle - b = 1 \\ &\implies b = -1 - \langle w, x \rangle \end{aligned} \tag{4.11}$$

So b is the average position, giving

$$b = \frac{\frac{1}{N_S} \sum_{m \in S} 1 - \langle w, x_m \rangle + \frac{1}{N_S} \sum_{m \in S} -1 - \sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}_i, x_m)}{2} \tag{4.12}$$

As with the SVM+ we now to to take into account the 'kernel trick', by substituting w with $\sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$ (see eq. 4.9)

Unfortunately there isn't a deduction of b^* that has been found.

We can now depict what the spaces $\mathcal{X}$ and $\mathcal{X}^*$ look like for 2-D data and show the effect of the various hyper-parameters in fig. 4.2.

4.2.2 SVM$_\Delta$+: Simplified Approach

Margin Transfer aims to approximate the effect of the SVM$_\Delta$+. Whilst the SVM$_\Delta$+ is a new approach, it takes a long time to compute. By approximating the value of the ζ, we can emulate the classifier in practice.

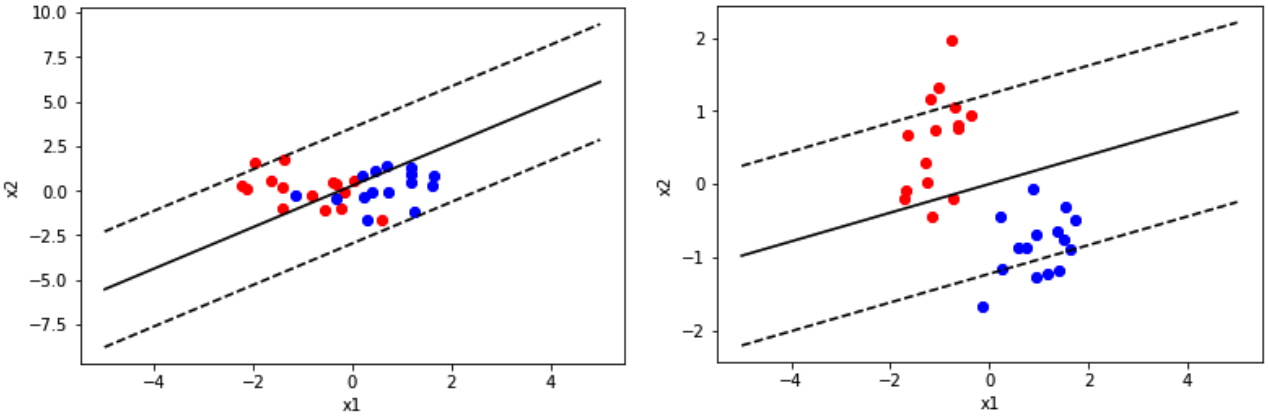
To construct SVM$_\Delta$+: Simplified Approach, we find an SVM solution in $\mathcal{X}^*$. We then use this to get the privileged space slack ξ^*

$$\xi^* = [1 - y(\langle w^*, x^* \rangle + b^*)]_+$$

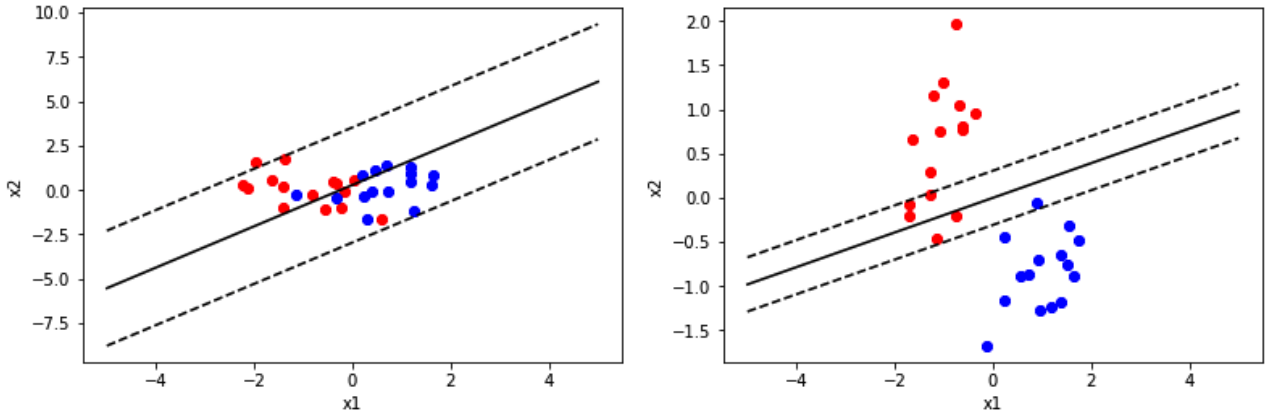
We then learn a modified SVM solution in $\mathcal{X}$

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{subject to} \quad & \sum_{i=1}^{\ell} \alpha_i \xi_i^* \leq C \sum_{i=1}^{\ell} \xi_i^* \\ \text{and} \quad & 0 \leq \alpha_i \leq (1 + \Delta) C \\ \text{and} \quad & \sum_{i=1}^{\ell} \alpha_i y_i = 0 \end{aligned} \tag{4.13}$$

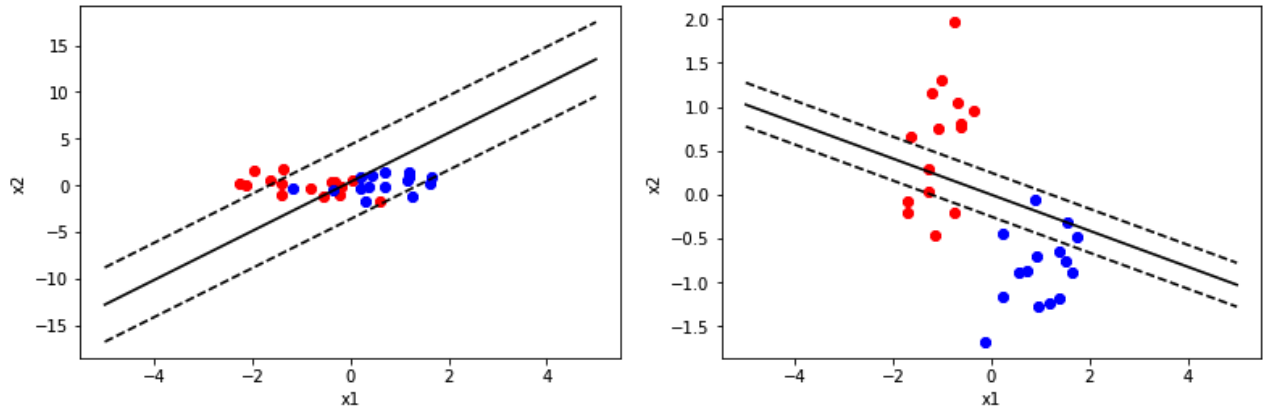
Where the only modification from a regular SVM is incorporating the slack value from the privileged space into the constraints on the SVM's dual form. This new constraint translates to: Ensure the



(a) $\gamma = 1$, $\Delta = 10$. Baseline comparison. For reference, the 4 blue points outside of the margin on the right correspond to the 4 misclassified blue points on the left. Similarly the red point at the top on the right corresponds to the furthest misclassified red point on the left



(b) $\gamma = 0.5$, $\Delta = 10$. By halving γ the unit length from the margin boundary of X^* has shrunk. Though the boundary is in the same place, all points are now considered further away from it



(c) $\gamma = 0.5$, $\Delta = 1$. By reducing Δ the importance of classifying points in $\mathcal{X}^*$ correctly has diminished. As such we place less emphasis on classifying this space correctly

Figure 4.2: (Left a-c) The decision boundary in $\mathcal{X}$
(Right a-c) The margin function in $\mathcal{X}^*$. C=1 throughout.
We have overlapping, but largely linearly separable data in X, but separable privileged information. We seek to maximise the margin between the two classes in $\mathcal{X}$ whilst keeping the number of margin violations to a minimum. a-c show the SVM$_\Delta$+ classifying the data with the maximum margin between classes whilst maintaining the smallest margin violations. In these examples the effect that the hyper-parameters γ and Δ have in both spaces is demonstrated.

sum of the slack values in $\mathcal{X}^*$ multiplied by that point's Lagrange multiplier [4] is less than the sum of all the slack values in $\mathcal{X}^*$ multiplied by the regularization hyper-parameter. When the regularization hyper-parameter (C) is large, the loss function is 'increased'. This heavily punishes the weights for allowing margin violations. As such, a larger C tends to reduce the number of margin violations in $\mathcal{X}$. In general, when the value of C is increased, the number of support vectors in $\mathcal{X}$ falls. When the number of support vectors in $\mathcal{X}$ is reduced, the number of points that can be affected by Margin Transfer are reduced. In the regular SVM, the value of the Lagrange multiplier for each point has an upper bound of C. If this was the case in this approach the first constraint in eq.4.13 would always hold by definition. However, α has an upper bound of $C + \Delta C$. So whilst in general a large C means that few points can be affected, a large Δ can mean that those points that are affected may be affected to a greater extent.

4.3 Knowledge Transfer

Knowledge Transfer is the second LUPI method introduced by Vapnik in his 2015 paper [22]. It has the same overall aim as Similarity Control, in that we want to use privileged information to better classify in our decision space, $\mathcal{X}$. When we implemented the SVM$_\Delta$+ we posited that the method worked best when there was an underlying classification rule in $\mathcal{X}^*$ that could be found, despite there not necessarily being one in $\mathcal{X}$. The motivation behind Knowledge Transfer comes from trying to take advantage of there being a better separation in $\mathcal{X}^*$ and poses the question: What if we could transform our data in $\mathcal{X}$ into data in $\mathcal{X}^*$? To make this tangible, given our example of the biopsy images being our data in $\mathcal{X}$ and our surgeon's notes being the data in $\mathcal{X}^*$. Instead of using our doctors notes to improve classification of the biopsy images, what if we could find a function that turned the images into (a representation of) the surgeon's notes? We can better classify with the surgeon's notes, so if we were able to learn such a function, we'd be better able to classify the image. In Vapnik's paper he suggests that we could use this transformed data to construct triplets $(\phi(x_i), x_i^*, y_i)$ where $\phi(x_i)$ is the transformation of X into X^* and train an SVM$_\Delta$+ on this new triplet.[5]

Given the radically different approach of Knowledge Transfer compared to the SVM+ or Similarity Control methods, we should consider whether we feel comfortable classifying this approach as Learning Using Privileged Information. To do this we have to consider what makes a LUPI technique. The motivation behind other methods of LUPI has been to improve the rate of convergence, by using the privileged information to determine the slack value in $\mathcal{X}$. As such, the classifier is still learnt in space $\mathcal{X}$, though is influenced by $\mathcal{X}^*$. This gives us a check-list of two points to determine if a technique qualifies as LUPI.

1. Does the method improve the rate of convergence to the optimal solution?

2. Does the classification rule remain in non-privileged space $\mathcal{X}$?

Knowledge Transfer will improve the rate of convergence if there is a better classification rule that can be learnt in privileged space $\mathcal{X}^*$. The classification rule in Knowledge Transfer still takes place in non-privileged space $\mathcal{X}$. It may help to think of a Knowledge Transfer classifier 'pre-processing' the data that it receives as input by projecting it into a space learnt in $\mathcal{X}^*$. As such, the decision rule is still found based on the non-privileged data X. Given that Knowledge Transfer satisfies our two conditions, we can satisfactorily call it a LUPI technique.

[4] Ensuring that we only consider slacks that correspond to support vectors in $\mathcal{X}$

[5] To make things complicated, Vapnik describes that it is possible to learn an SVM solution on pairs $(\phi(x_1), y_1), \ldots, (\phi(x_\ell), y_\ell)$ rather than an SVM$_\Delta$+ solution on the triplets mentioned above. However, this is only the case when step three of the implementation ("Find the functions") produces almost 0 error. As such, the more general approach is to use an SVM$_\Delta$+ classifier.

4.3.1 Implementing Knowledge Transfer

An implementation of Knowledge Transfer is linked to in appendix D. The sections below are named following Vapnik's naming conventions.

Three Steps to Knowledge Transfer

- Find the fundamental elements of knowledge $u_1^*, u_2^*, \ldots, u_m^*$ in space $\mathcal{X}^*$.
- Find the "frames of knowledge" $K^*(u_1^*, \mathbf{x}^*), K^*(u_2^*, \mathbf{x}^*), \ldots, K^*(u_m^*, \mathbf{x}^*))$ in $\mathcal{X}^*$.
- Find the functions $\phi_1(x_i), \phi_2(x_i), \ldots, \phi_m(x_i)$ in space $\mathcal{X}$ such that $\phi_k(x_i) \approx K^*(u_k^*, x_i^*)$ holds true for almost all pairs (x_i, x_i^*)

Finding the fundamental elements of knowledge

This means to find the support vectors in $\mathcal{X}^*$. Although as we are after the "fundamental" elements, we strictly want to find the reduced number of support vectors, which is beyond the scope of this paper. To find the support vectors in $\mathcal{X}^*$ requires simply learning an SVM solution in $\mathcal{X}^*$.

Find the frames of knowledge

For each data point in X^*, get the kernel distance between each data point and the support vectors in privileged space. This projects the problem to a different space. Imagine we have two dimensional data in both $\mathcal{X}$ and $\mathcal{X}^*$. If we have learnt an SVM solution in $\mathcal{X}^*$ we may have three support vectors in privileged space. We obtain our 'frames of knowledge' by comparing each data-point in $\mathcal{X}^*$ to each of the three support vectors, resulting in a $3 \times N$ matrix where each column represents a feature vector, each feature being the distance from a data-point to one of the support vectors. These new feature vectors are the 'frames of knowledge'.

Find the functions

This leaves us with a regression problem. Given pairs $(x_i, K^*(u_1^*, x_i^*)), \ldots, (x_i, K^*(u_m^*, x_i^*))$, find the function $\phi_s(x_i)$ for each $s = 1 \ldots m$ such that $\phi(x_k) \approx K(u_k^*, x_i^*)$. We want to find mappings that project $\mathcal{X}$ to each of the 'frames of knowledge'.

We then find an SVM$_\Delta$+ solution using triplets $(\phi(x_i), x_i^*, y_i)$, $\forall i \in 1, \ldots, \ell$. When we come to make predictions we 'transform' $\mathcal{X}$ to the new space using $\phi(x_i)$.

Chapter 5

Unifying Approaches

When we introduced LUPI, we argued that the ability to generate the slack (ξ) in x is determined by the quality of the selection of X^*. This supposes that there is an intelligence in the selection of X^*. That we have some pre-existing knowledge of the conditional probability function from which X^* is drawn. What if we don't know what we're being provided with in X^*? We know that it's related to X, but we're not sure if it is best suited to the SVM+ classifier, or the SVM$_\Delta$+ classifier from Similarity Control. In this situation we would have to try both and their performance may lead us to different reasoning about the X^*. A strong performance by the SVM+ implies that the data in X^* is correlated with high variance (though not necessarily separable), giving us the ability to learn a function that describes the slack in X. A strong performance by the SVM$_\Delta$+ classifier indicates that the data in X^* is separable and that the distance to the margin in X should accommodate the distance to the margin in X^*.

There is a problem though. If both the SVM+ and the SVM$_\Delta$+ perform well, we learn that the data is correlated, with high variance, but also separable. We don't have a mechanism to take advantage of this.

We need a unified approach that considers the two approaches simultaneously. Which is why this paper introduces the SVMu (with the u standing for unified).

5.1 SVMu

We want to learn a classifier where the margin in X is affected by the primal objective of the SVM+ and the primal objective of the SVM$_\Delta$+. However, these are in opposition to each other. The SVM+ has the constraint that $\langle w^*, x^* \rangle + b^* \geq 0$, whilst in the SVM$_\Delta$+ we have the constraint that $y(\langle w^*, x^* \rangle + b^*) + \zeta \geq 0$. Here we have a problem. The SVM+ doesn't care about the class of the data, it just wants to learn a margin such that all values are on the positive side of it, hence the line it wants to learn will be similar to a regression line in that has a similar correlation to the data. The SVM$_\Delta$+ on the other hand wants the data to be correctly classified. The constraints of the SVM$_\Delta$+ will place its margin boundary between the two classes in X^*. Because of this, whilst they can operate in the same space (X^*), they can't both affect w^*.

The choice of how to continue falls down to a choice between two approaches.

1. Operate in X^*. but learn two weights, w^* and w^{**}. Or
2. Increase the input space to X^* and X^{**}, but with the same data in both spaces. In this situation we still learn w^* and w^{**}, but we have the flexibility to learn different kernel methods in both X^* and X^{**}, which might be more suitable to to each task.

It seems clear that the best approach is to increase the input space to X^* and X^{**}. [1]

[1] This may look like an opportunity to learn with different data in X^* and X^{**}, but that wouldn't be appropriate. The privileged information should be replicated so that it is both spaces. To use an additional set of privileged would require increasing the input space again to X^{***} and X^{****}, repeating for more input spaces.

This gives the primal objective function

$$\begin{aligned}
\min_{w,w^*,w^{**},b,b^*.b^{**},\zeta} \quad & \frac{1}{2}[\langle w,w\rangle + \gamma\langle w^*,w^*\rangle + \Gamma\langle w^{**},w^{**}\rangle] \\
& + C\sum_{i=1}^{\ell}(\langle w^*, x_i^*\rangle + b^*) \\
& + C\sum_{i=1}^{\ell}[y(\langle w^{**}, x_i^{**}\rangle + b^{**}) + \zeta_i] + \Delta C\sum_{i=1}^{\ell}\zeta_i \\
\text{subject to} \quad & y(\langle w, x_i\rangle + b) \geq 1 - (\langle w^*, x_i^*\rangle + b^*) \\
\text{and} \quad & (\langle w^*, x_i^*\rangle + b^*) \geq 0 \\
\text{and} \quad & y(\langle w, x_i\rangle + b) \geq 1 - y(\langle w^{**}, x_i^{**}\rangle + b^{**}) - \zeta_i \\
\text{and} \quad & y(\langle w^{**}, x_i^{**}\rangle + b^{**}) + \zeta_i \geq 0 \\
\text{and} \quad & \zeta_i \geq 0, \quad \forall i = 1,\ldots,\ell
\end{aligned}$$

Where γ and Γ are hyper-parameters that control the influence on their respective classifiers (γ the SVM+ and Γ the SVM$_\Delta$+). In the case that the SVMu can't outperform either the SVM+ or SVM$_\Delta$+, the relationship between γ and Γ will be inverse. As one increases, the other decreases. The reason Γ isn't replaced with simply $\frac{1}{\gamma}$ is that we should consider the case that we want to use both techniques on the privileged data.

Transforming this objective function to its dual gives us

$$\begin{aligned}
\max_{\alpha,\beta,\eta,\mu} \quad & -\frac{1}{2}\sum_{i,j=1}^{\ell} y_i y_j K(x_i,x_j)(\alpha_i+\eta_i)(\alpha_j+\eta_j) \\
& -\frac{1}{2\gamma}\sum_{i,j=1}^{\ell} K^*(x_i^*x_j^*)(\alpha_i+\beta_i-C)(\alpha_j+\beta_j-C) \\
& -\frac{1}{2\Gamma}\sum_{i,j=1}^{\ell} y_i y_j K^{**}(x_i^{**}x_j^{**})(\eta_i+\mu_i-C)(\eta_j+\mu_j-C) \\
& +\sum_{i=1}^{\ell}\alpha_i+\eta_i \\
\text{subject to} \quad & \sum_{i=1}^{\ell} y(\alpha_i-\eta_i) = 0 \\
\text{and} \quad & \sum_{i=1}^{\ell}(-\alpha-\beta+C) = 0 \\
\text{and} \quad & \sum_{i=1}^{\ell} y(-\eta_i-\mu_i+C) = 0 \\
\text{and} \quad & \alpha \geq 0 \\
\text{and} \quad & \beta \geq 0 \\
\text{and} \quad & 0 \leq \eta \leq -\mu + C + \Delta C \\
\text{and} \quad & \mu \geq 0
\end{aligned}$$

The steps for deriving the dual for the SVMu can be found in appendix E.

There is a problem though. Tuning 2 kernels and 2 hyper-parameters in the SVM+ and 2 kernels and 3 hyper-parameters in the SVM$_\Delta$+ is expensive, but tuning 3 kernels and 4 hyper-parameters increases the complexity by an order of magnitude. A selection of 3 kernels, and a grid search over

7 values (0.001, 0.01, 0.1, 1, 10, 100, 1000) for each hyper-parameter leads to training 441 different solutions for the SVM+, 3,087 different solutions for the $SVM_{\Delta}+$ and 64,827 different solutions for the SVMu. Whilst the nature of tuning hyper-parameters by grid search is embarrassingly parallel, meaning that there is almost no effort required in separating the problem into parallel tasks[2], it still requires a lot of computational power. We can also take advantage of 'fixing' the value of the *Delta* as we did with the $SVM_{\Delta}+$.

However, hyper-parameter tuning is an active area of research[19]. It would be an interesting extension to find more optimizations for the SVMu if it proves to be a useful technique.

[2]Experiments conducted in this paper have taken advantage of multi-core architectures

Chapter 6

Methodology

This report looks at the performance of 6 classifiers on a publicly available dataset for text classification known as TechTC (http://techtc.cs.technion.ac.il/).

6.1 The Classifiers

The classifiers that are being looked at are below. Each has been implemented from scratch by the author.

1. **SVM** The Support Vector Machine is the basis from which the other classifiers are built on. It is a widely used classifier and the benchmark that we will be judging the performance of the other classifiers against.

2. **SVM+** As described by Vapnik in 2009.

3. **SVM$_\Delta$+** As described in Similarity Control [Vapnik 2015].

4. **Margin Transfer** aka SVM$_\Delta$+: Simplified Approach. An SVM solution is found in X^*. From then a modified SVM solution is found in X such that the margin violation (if any occur) in X^* is taken into account in X.

5. **SVMu** The classifier introduced in this paper.

6. **Knowledge Transfer** A naïve implementation of this classifier has been implemented for comparison purposes. The only hyper-parameter tuned was C. It is possible to tune this classifier further.

6.2 The Data

Five datasets from the TechTC collection of data were chosen to test the classifiers on. The TechTC data consists of data points (X) and labels (Y). Each data point consists of approximately 1500 features. As we want to test the performance of privileged information classifiers, we will generate our own 'privileged information' as suggested by Taylor et al (LUFe) [21]. We perform recursive feature elimination on the Tech TC data to determine the 300 most informative features and make this our input space X for the above classifiers. The remaining 1200 features will form the 'privileged information' X^*. This is valid privileged information. The conditional probability function $P(x^*|x)$ returns the unselected features for a data point given the 300 selected features. The class labels will be as in the TechTC dataset.

6.3 Parameter Selection

To be fair to each classifier we want to consider each one performing at its best. It wouldn't be fair to compare classifiers that aren't tuned to their optimum with each other. However, to do this for a full range of parameters would be beyond the computational power available to the author. As such we conduct a grid search of 3 kernels in each input space (Linear, Quadratic and Gaussian) and a range of 3 logarithmically spaced values for each hyper-parameter (C, γ and Γ as appropriate for each classifier), so $10^{-1}, 10^{0}, 10^{1}$. The value of Δ is fixed as discussed in section 4.2.1, so 10 in this case.

As the range $10^{-1}, 10^{0}, 10^{1}$ is rather small to give a fair representation of the performance of our classifiers, we make the following optimisation steps. Once the reduced hyper-parameter search for each classifier has been completed, we see which combination of kernels performed the best on average across the five datasets. We then run another search fixing the value of the kernels in each space. Say for example that we performed the initial, reduced hyper-parameter search with the SVM+ classifier. If this gave results that the classifier performed best when a Gaussian kernel was used in X space and a Linear kernel was used in X^* space, then we save this information. We then start another search over a wider range of values, but fix the kernels to be the ones that performed best in the initial search. The new range we tune our hyper-parameter over is $[0.001, 0.1, 10, 1000]$, setting $\Delta = 1000$. Whilst this approach will be quicker, different kernel combinations may work better for different datasets. Using one combination of kernels per classifier on each dataset may not give the best results possible for each individual dataset.

To test the effectiveness of our classifiers we use cross-validation. We take our dataset of selected features (X), unselected features (X^*) and class-labels (Y) and create 10 equally sized sets of data (folds). Each classifier will be judged on the average performance after being trained on the data from 9 of the ten folds, then 'tested' on the 10th, repeated 10 times so that each fold becomes the 'test' data exactly once. The classifiers will be judged based on their accuracy and their F-Score. However, this leaves a problem. As discussed, each classifier has a number of hyper-parameters to choose from (C, γ etc.). Which is the best combination of hyper-parameters to use for each classifier? That question goes even deeper. The hyper-parameter selection isn't decided by the classifier choice, but chosen by the data. To select the hyper-parameters we have to cross-validate each set of training folds (see fig. 4.1).

An inner fold is formed by splitting the outer fold's training data (the 9 non-testing folds) into 5 equally sized sets of data. Each possible combination of kernels and hyper-parameters is trained on these inner folds. There are 5 inner folds and they can be viewed as in fig.4 so that they have 1 inner testing set, which is just one of the inner folds, and an inner training set, which consists of the remaining inner 4 folds. This is repeated so that each inner fold becomes the testing set exactly once. Once each possible kernel/hyper-parameter combination has been trained and tested on the inner folds, the average performance across he inner folds is taken. The best performing combination on average across all inner folds is then selected to be the parameters selected for training the classifier on the outer fold. Best performing is a slightly ambiguous term. It's hard to say which result is necessarily better in given situations. though in this experiment we consider accuracy and F-Score, which are described below. Results for accuracy and F-Score are saved. The highest accuracy is chosen as the best performing combination. In the event of a tie, the highest F-Score within the tied group is selected. Again, in the event of a tie, the combination returned is randomly selected by numpy's max function. The results for best combination based on F-Score similarly select the best combination based on F-Score from the inner folds (with the case of a tie being as above, but with Accuracy being the tie breaker).

6.3.1 Accuracy and F-Score

Given the confusion matrix in fig. 6.2 accuracy is the True Positive (TP) and True Negative (TN) divided by the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

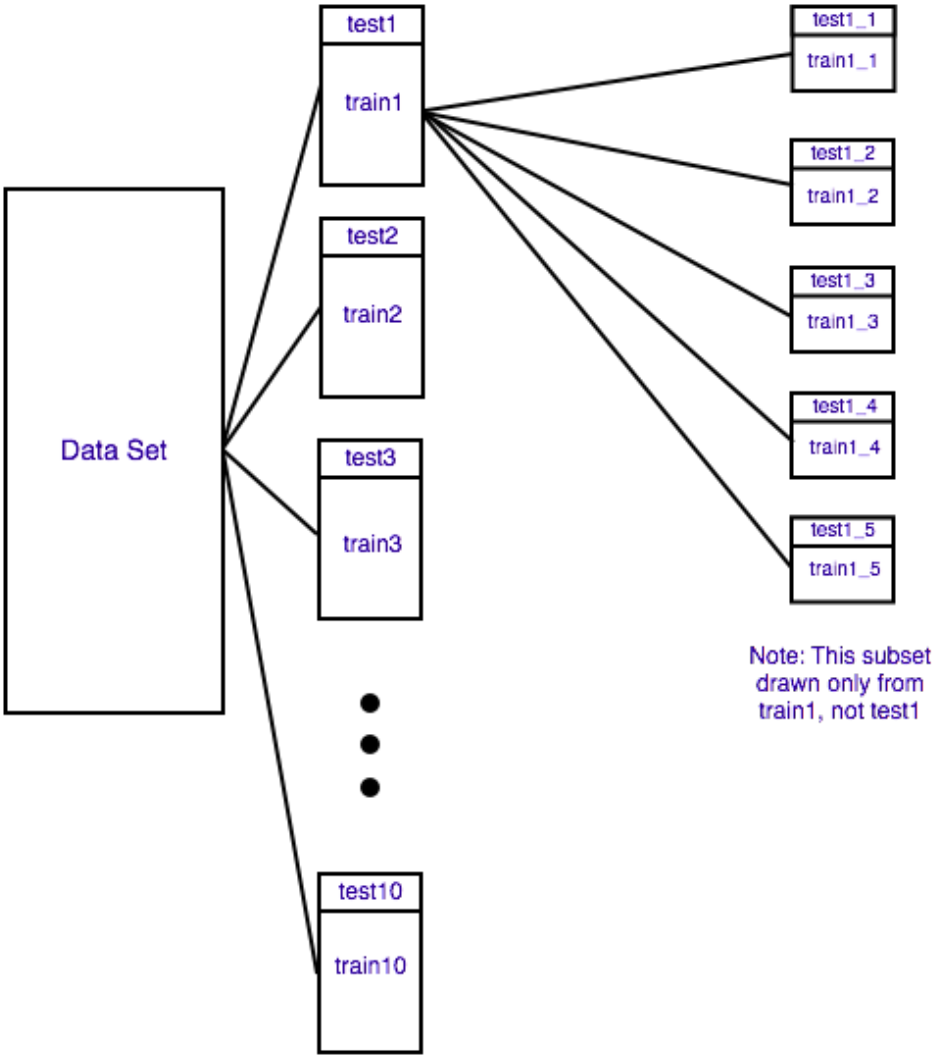


Figure 6.1: Relationship between inner folds (right) to outer folds (middle) to dataset (left).

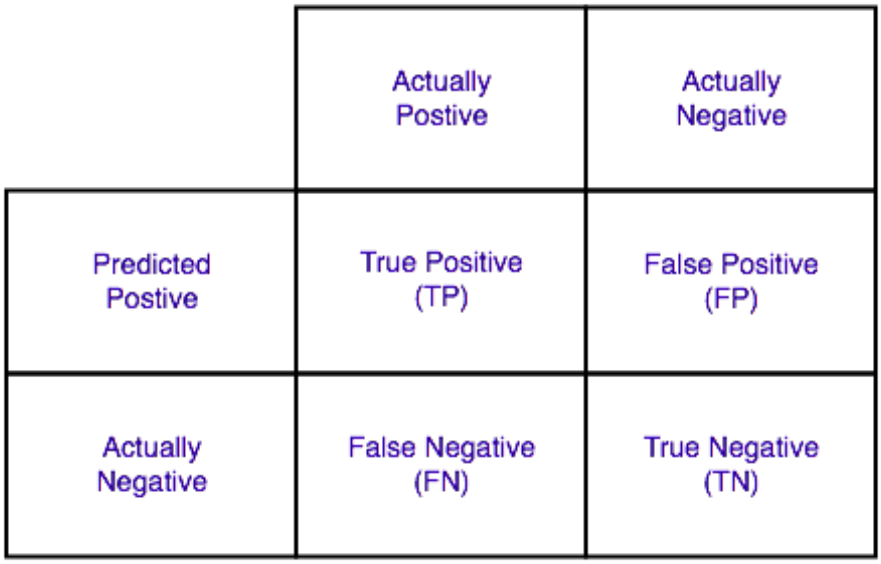


Figure 6.2

However, this can sometime lead to misunderstanding about the data. If the data isn't perfectly split, so you have an uneven number of actual positives and actual negatives, your accuracy result will be affected.

The F-Score considers both the precision (number of selected items that are correct) and recall (number of correct items that were selected) to give.

$$2 \times \frac{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

Whilst this is a more robust measurement than the accuracy, it is less intuitive to understand. A higher F-Score means that we have a high precision **and** high recall for a given class.

Chapter 7

Results and Discussion

In the following charts, the following key is used to describe the classifiers tested.

SVM Regular SVM as implemented in this paper

SVM+ SVM+ as implemented in this paper

MT $SVM_\Delta+$: Simplified Approach aka Margin Transfer as implemented in this paper

SVMd+ $SVM_\Delta+$ as implemented in this paper

SVMu SVMu as introduced and implemented in this paper

KT Knowledge Transfer as implemented in this paper

7.1 Predictions Given The Data

As discussed in previous sections, the 'shape' of the privileged data becomes important in determining which classifier will perform the best. Based on our knowledge of the conditional probability function from which the privileged information is being drawn (the unselected features), we can make a prediction that the SVM+ will be the best performing classifier. This is because we are removing the features which give us the best guide for classification. As such, we can imagine that our privileged information is the 'noise' associated with the data. If either Similarity Control method (and in turn Knowledge Transfer) perform better than the SVM+, then that can only be because there is an underlying classification that can still occur in the unselected features. Such a case would imply that we could have selected more useful features.

In general we anticipate that LUPI should perform better than the non-LUPI method of the regular SVM. The exception to the rule is the Knowledge Transfer method. Given that Knowledge Transfer works by 'transforming' the data from $\mathcal{X}$ to $\mathcal{X}^*$ we are presented with a problem given the dataset. Knowledge Transfer in this situation will 'transform' the 300 most informative features into a representation of the 1,200 least informative features. It will then try to learn a solution in this space. Even in the case where the unselected features are still informative, classifying without the 300 most informative features is still going to give poor results.

7.2 Initial search

The first step considered a full search over all kernels and a limited hyper-parameter range of [0.1, 1, 10] for all hyper-parameters, except Δ which is fixed to 10. These were then used to select the best kernel combination for each classifier on average across all the datasets.

Classifier	$\mathcal{X}$ Kernel	$\mathcal{X}^*$ Kernel	$\mathcal{X}^{**}$ Kernel
SVM	Linear	n/a	n/a
SVM+	Linear	Quadratic	n/a
MT	Linear	Linear	n/a
SVMd+	Linear	Linear	n/a
SVMu	Linear	Quadratic	Gaussian
KT	Linear	Linear	n/a

Table 7.1: Best kernel combination for each classifier in the initial search.

7.3 Selecting Kernels

The consistently best performing kernel combination across the five datasets are shown in table 7.1. These are the combination of kernels that are used whilst searching a wider range of hyper-parameters.

7.4 Full Results

Results are shown in table 7.2, with the best results highlighted in bold. The results are broadly in line with the predictions. Generally, the SVM+ is the best classifier across the datasets. As anticipated, Knowledge Transfer performed poorly. Because of this it is not shown in fig. 7.1 which charts the contents of table table 7.2.

Dataset 137 (fig. 7.1 a) behaves almost as we would expect. The performance of the SVM+ suggests that the unselected features aren't particularly separable. This is corroborated by the performance of the SVM$_\Delta$+, which appears to fall back on an SVM solution rather than make use of the unselected features. Given this, the SVMu can only try and match the performance of the SVM+, which we see that it gets close to doing.

There are a number of surprises. In chapter 4 we theorised that the SVMu would perform best when both SVM+ and SVM$_\Delta$+ performed well compared to the SVM. However, we see that this hasn't always happened. In dataset 174 (fig. 7.1 b) both SVM+ and SVM$_\Delta$+ perform better than the SVM, but the SVMu falls short of the SVM for accuracy (though surpasses it by F-Score) and is less effective than both SVM+ and SVM$_\Delta$+ individually. This is surprising as it is designed to perform at least as well as one of them. Similarly dataset 197 (fig. 7.1 c) shows neither the SVM+ or the SVM$_\Delta$+ surpassing the SVM solution. However, in this case, the SVMu performs extraordinarily well.

Across the five datasets SVM$_\Delta$+: Simplified Approach (Margin Transfer) appears to perform poorly. Just as with Knowledge Transfer, this classifier is not best suited to this kind of data. Whereas the SVM$_\Delta$+ has a mechanism to 'ignore' the privileged information, by changing the value of γ, the SVM$_\Delta$+: Simplified Approach (Margin Transfer) does not. As such, it is being forced to take into account data that is not useful to it.

In general we can see that at least one of the LUPI classifiers performs better than the regular SVM. In the case where the SVM+ didn't perform best, the SVMu performed better. In fact, both the SVM+ and SVMu performed better than the regular SVM in across all datasets in terms of F-Score, and all but one dataset in terms of accuracy. The SVM$_\Delta$+ performs relatively well. However, the privileged information in these datasets is not well suited to the classifier.

Dataset 137		
Classifier	**Accuracy (%)**	**F-Score**
SVM	95.549	0.951
SVM+	**97.219**	**0.972**
MT	92.800	0.930
SVMd+	95.022	0.952
SVMu	96.692	0.966
KT	81.581	0.819

(a)

Dataset 174		
Classifier	**Accuracy (%)**	**F-Score**
SVM	86.274	0.838
SVM+	**91.290**	**0.898**
MT	87.042	0.853
SVMd+	89.479	0.881
SVMu	85.125	0.874
KT	73.607	0.738

(b)

Dataset 197		
Classifier	**Accuracy (%)**	**F-Score**
SVM	90.611	0.899
SVM+	90.167	0.900
MT	90.167	0.889
SVMd+	88.611	0.871
SVMu	**93.889**	**0.939**
KT	78.444	0.695

(c)

Dataset 219		
Classifier	**Accuracy (%)**	**F-Score**
SVM	90.146	0.904
SVM+	**92.895**	**0.932**
MT	92.339	0.928
SVMd+	91.342	0.917
SVMu	92.339	0.927
KT	78.822	0.692

(d)

Dataset 254		
Classifier	**Accuracy (%)**	**F-Score**
SVM	87.225	0.860
SVM+	**88.751**	**0.880**
MT	84.058	0.819
SVMd+	87.307	0.836
SVMu	88.444	0.867
KT	73.336	0.651

(e)

Table 7.2: Full results across all datasets.

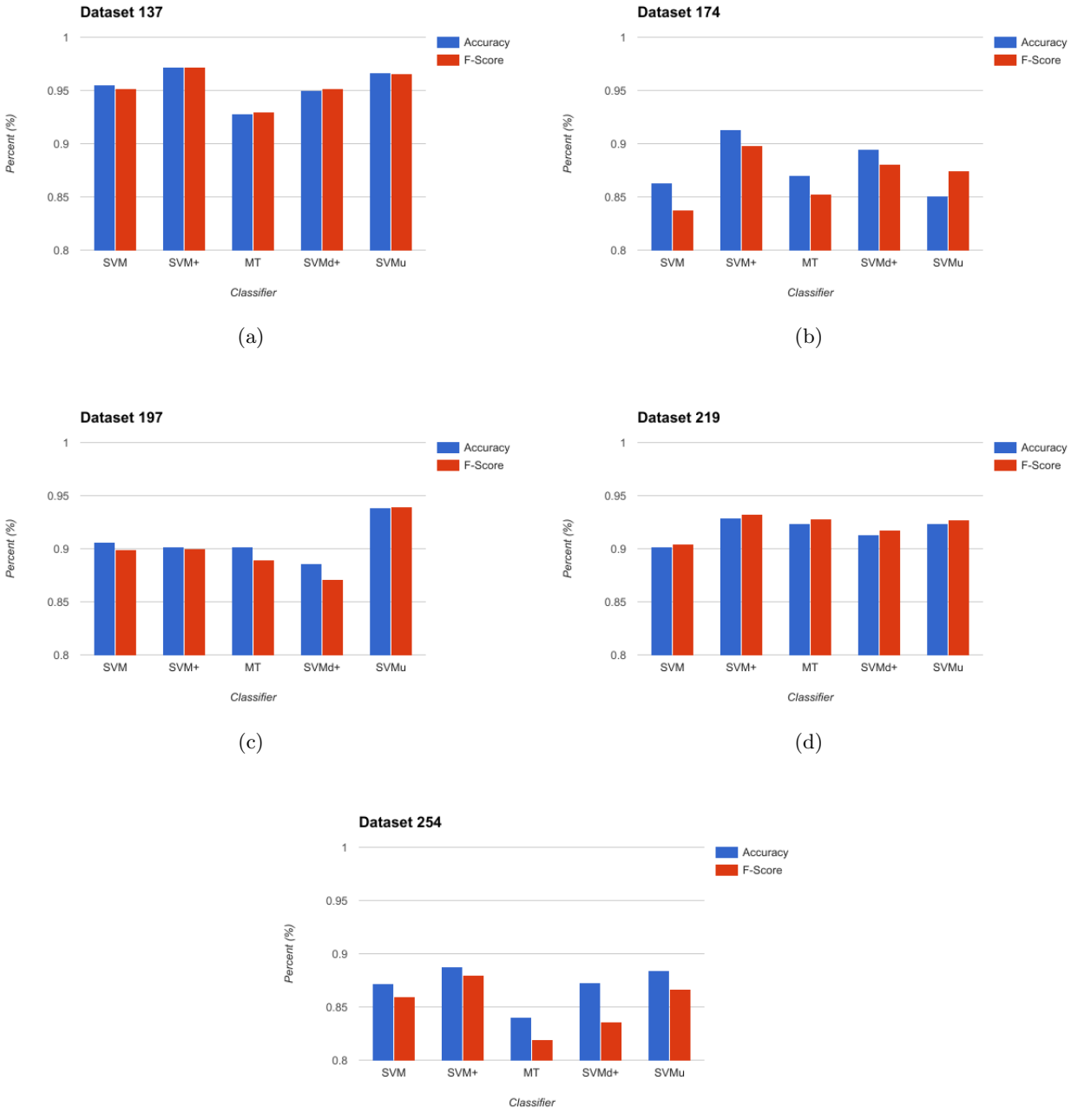


Figure 7.1: Full results across all datasets. Vertical axis shows the range 80% to 100%.

Chapter 8

Applications

This report looks at the performance of six classifiers on a publicly available dataset for text classification (http://techtc.cs.technion.ac.il/). However, the most salient point about privileged information is that it's ubiquitous. For (almost) any machine learning problem, privileged information exists. So let's consider a more practical use, applying privileged information to ethical machine learning.

8.1 Fairness

Fairness in machine learning is the idea that we don't want to build biases into our classifiers or other techniques. As more decisions are being made by algorithms, we want to ensure that the notion of fairness is ingrained into their models. However we should begin by describing our notion of fairness. To do this we will use two definitions of unfairness used in a recent paper on this subject [26], disparate treatment and disparate impact.

Disparate Treatment This is where decisions are (at least partly) made on the features of an individual which are deemed to be sensitive in nature. Examples of sensitive features include gender, race, religion etc. Disparate treatment is faced by an individual. For example, the interviewer immediately decided the candidate was unsuitable for the job is they had a specific sensitive attribute.

Disparate Impact This is where certain groups are disproportionally hurt (or promoted) based on their sensitive features. This is disparate treatment affecting a group a whole. For example, on average less females are senior managers in large companies than you would expect based on the proportions of men and women as a whole.

There is a conflict between making predictions 'fairly' and making predictions 'accurately'. To make prediction accurately means to emulate the decision procedure that we have observed from the training data. Making predictions using our notion of fairness involves learning a rule that minimises disparate impact and treatment based on 'unfair' or 'sensitive' features. This 'fair' prediction is often inaccurate. This is because the underlying rule may be based on on the sensitive features. If the underlying decision rule is influenced by a feature we have determined is unfair or sensitive in nature, then the accuracy of the classifier will be impacted. A naïve approach may be simply removing the sensitive features from our data. We can ensure that we're not taking into account the feature directly, but we may learn that other features represent similar information.

For example, we want to hire a graduates from a diverse range of backgrounds. However, the 'clever' students from a majority group have been disproportionately encouraged to study finance. Similarly 'clever' students from a minority group may have been disproportionately encouraged to study Physics. To ensure that we are hiring from both groups, let's assume we have decided to simply not include membership of either group as a feature. It should be clear that our classifier will learn a correlation between finance students and 'cleverness' and simply select those students who study finance. Despite

removing the sensitive feature, we still have disparate impact to the minority group. Simply removing the sensitive features isn't enough. We need an approach that is not only 'aware', taking into account all features, but ensures fairness and also is able to balance the trade-off between fairness and accuracy.

We want to include a trade-off measure, to maintain predicted behaviour, but also to improve fairness. Such an approach isn't about creating the 'fairest' classifier, but rather a 'fairer' classifier. To implement this, a proposal for amending classifiers to include a "percentage fairness" constraint has been suggested by Zafar et al [26].

The downside is that we have to train our classifier using the sensitive features. As such, we have to make predictions based on data that includes them, directly contradicting discrimination laws.

Given that a significant portion of this thesis has been implementing the LUPI techniques such as the SVM$_\Delta$+, is it possible to incorporate Zafar's 'percentage fairness' constraint into a LUPI classifier?

8.2 Sensitive Features as Privileged Information

The benefit of using sensitive features as privileged information is that the privileged information is not needed at prediction time. We are able to train using the sensitive features, so are able to mitigate unfairness and we can remain compliant by not considering sensitive features when we come to make decisions.

8.3 Implementation

Zafar's fairness constraints are

$$\begin{aligned} \text{minimize} \quad & L(\theta) \\ \text{subject to} \quad & \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})d_\theta(x_i) \leq c \\ \text{and} \quad & \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})d_\theta(x_i) \geq -c \end{aligned} \tag{8.1}$$

Where L is the loss function on some weights θ, c is the covariance threshold, specifying an upper-bound on the covariance between each sensitive attribute (z_i) and the **signed** distance from the feature vector (x_i) to the decision boundary ($d_\theta(x_i)$). For clarity, $\bar{z}$ is the mean average position of the sensitive features z.

We incorporate the extra constraints into the SVM$_\Delta$+. This gives.

8.3.1 Primal Form

$$\begin{aligned} \min_{w,w*,b,b*,\zeta} \quad & \frac{1}{2}[(w,w) + \gamma(w^*,w^*)] + C\sum_{i=1}^{\ell}[y_i((w^*,x_i^*)+b^*) + (1+\Delta)\zeta_i] \\ \text{subject to} \quad & 1 - y_i((w^*,x_i^*)+b^*) - \zeta_i - y_i((w,x_i)+b) \leq 0 \\ \text{and} \quad & -y_i((w^*,x_i^*)+b^*) - \zeta_i \leq 0 \\ \text{and} \quad & -\zeta \leq 0 \\ \text{and} \quad & \frac{1}{\ell}[\sum_{i=1}^{\ell}(z_i - \bar{z})d(x_i)] - v \leq 0 \\ \text{and} \quad & -\frac{1}{\ell}[\sum_{i=1}^{\ell}(z_i - \bar{z})d(x_i)] - v \leq 0 \end{aligned}$$

8.3.2 Dual Form

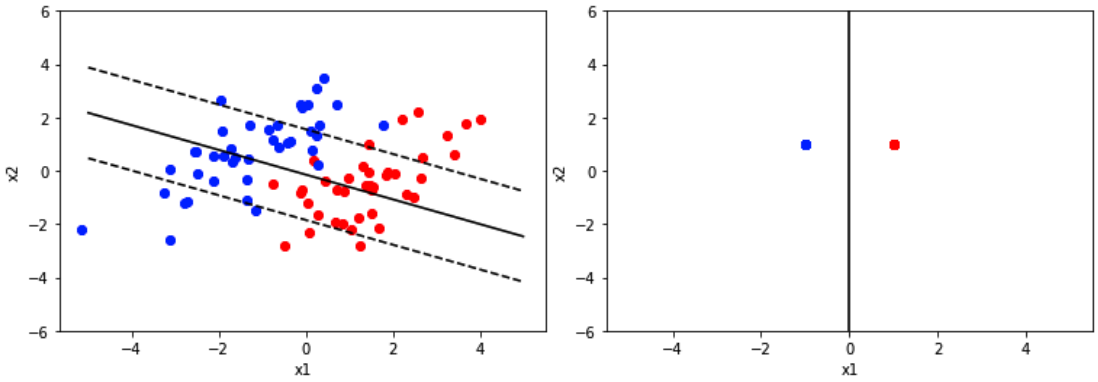
$$\max_{\alpha,\beta} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i=1}^{\ell} \alpha_i\alpha_j y_i y_j (x_i, x_i) - \frac{1}{2\gamma}\sum_{i,j=1}^{\ell} y_i y_j (x_i^*, x_j^*)(\alpha_i + \beta_i - C)(\alpha_j + \beta_j - C)$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i=1}^{\ell} y_i \alpha_i = 0 \\
\text{and} \quad & \sum_{i=1}^{\ell} y_i(-\alpha_i - \beta_i + C) = 0 \\
\text{and} \quad & 0 \leq \alpha \leq -\beta_i + C + \Delta C \\
\text{and} \quad & 0 \leq \beta \\
\text{and} \quad & \sum_{i=1}^{\ell} [\frac{1}{\ell}\sum_{j=1}^{\ell} \alpha_j y_j (x_i, x_j)(z_i - \bar{z})] \leq v \\
\text{and} \quad & -\sum_{i=1}^{\ell} [\frac{1}{\ell}\sum_{j=1}^{\ell} \alpha_j y_j (x_i, x_j)(z_i - \bar{z})] \leq v
\end{aligned}$$

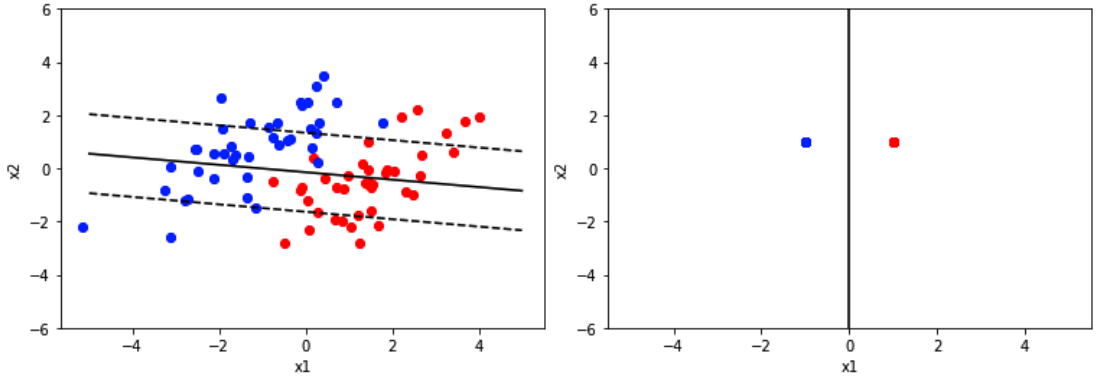
In fig. 8.1 we can see various SVM$_\Delta$+ classifiers with different fairness constraints being learnt across spaces $\mathcal{X}$ and $\mathcal{X}^*$. Whilst the data used is synthetic, we can imagine that the classifier in $\mathcal{X}$ is deciding if a candidate should be hired, or not hired and that the data in $\mathcal{X}^*$ represents a candidate's gender. The features in $\mathcal{X}$ may correspond to x1 being the amount of experience and x2 being education levels. We can see that our hiring policy has been quite biased. The blue data-points correspond to those who were hired, and red are those who weren't. Our previous hiring policy was to hire the blue gender.

The images of classifiers on the left in fig. 8.1 show the boundary and margin that the SVM$_\Delta$+ learns in non-privileged space $\mathcal{X}$. The right side of fig. 8.1 shows the margin function learned in $\mathcal{X}^*$. In (a) Zafar's 'fairness percentage' makes the classifier act fairly. We are taking into account the gender of the applicant and are trying to ensure that both groups are treated equally, so that we hire with minimal disparate impact or treatment the red and the blue gender.

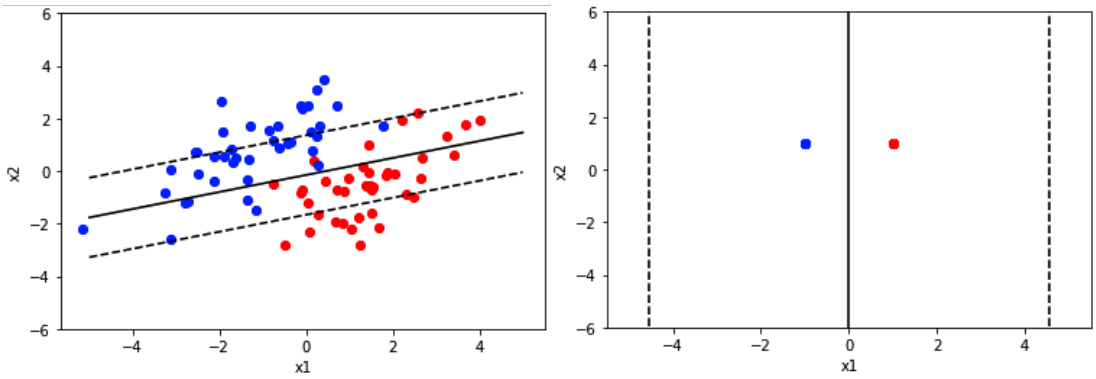
In (b) and (c) the fairness constraint is relaxed and the decision boundary in $\mathcal{X}$ can be seen 'rotating' towards the position shown in (d). As the decision boundary in $\mathcal{X}$ rotates, the mechanism by which $\mathcal{X}^*$ influences the outcome is shown. As the fairness constraint is lessened, the margin in $\mathcal{X}^*$ comes closer to the data-points. We know from chapter 4 that the margin represents one unit distance from the boundary. By bringing the margin closer to the boundary, the influence of the privileged information is greater and we learn a more accurate decision rule in $\mathcal{X}$.



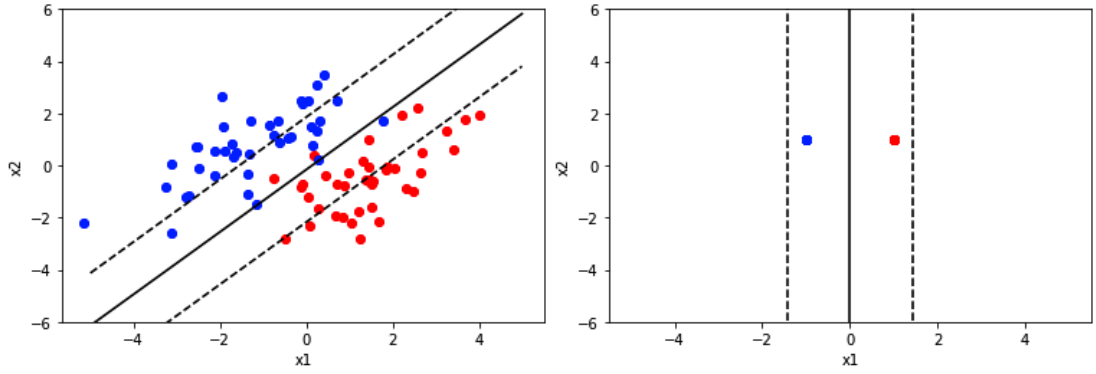
(a) 99% Fair



(b) 70% Fair



(c) 30% Fair



(d) 0.001% Fair

Figure 8.1: The effect of the 'Fairness Percentage' on classifying data.
(Left) The decision boundary in $\mathcal{X}$
(Right) The margin distance function in $\mathcal{X}^*$, where margin lines show one unit distance to the margin as in fig. 4.2

Chapter 9

Conclusion

This has been an interesting area to research. As privileged information is ubiquitous it is important that we learn techniques to best improve its performance. Initial signs from the classifier introduced in this paper, the SVMu are encouraging. Clearly there needs to be further experimentation on a wider range of datasets and this may form the basis of further work. Irrespective of its performance, the process of creating a new classifier was one that was rewarding and informative. The aim of creating a highly generalisable LUPI classifier such as the SVMu warrants further exploration. It may be possible to incorporate further methods into this 'all-in-one' approach, but balancing extra spaces and hyper-parameters against the curse of dimensionality is a difficult challenge. This paper introduced an initial structure for such a classifier and it would be and interesting further project to maximise its potential.

In terms of further work there is a lot more to be done. This is a rich vein of research and one that is fascinating to explore. Though only alluded to in this paper, there appears to be an intrinsic connection between LUPI and Machine Teaching[28]. Further exploration into this connection, particularly with regard to the conditional probability function $P(x^*|x)$. Throughout this paper, knowledge of the conditional probability function, Vapnik's 'Intelligent Teacher' has allowed us to reason about the performance of various LUPI techniques. Formalising this relationship, and putting it in terms of 'what makes *good* privileged information is another challenge.

Other papers have shown Knowledge Transfer to be useful as a model distillation method [10]. Given that Knowledge Transfer is a radically different approach, further research with regard to novel applications seems just. With regard to application, applying LUPI techniques to ethical machine learning is a fascinating one. The implications for this research are far reaching and it is imperative that the ideas developed in chapter 8 are developed further.

Bibliography

[1] A. Caliskan, J. J. Bryson, and A. Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017. ISSN 0036-8075. doi: 10.1126/science.aal4230. URL http://science.sciencemag.org/content/356/6334/183.

[2] C. Campbell and Y. Ying. Learning with support vector machines, 2011.

[3] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines: and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000. 10

[4] M. Faculty. Quadratic programming with python and cvxopt, 2012.

[5] P. Flach. Machine learning: The art and science of algorithms that make sense of data, 2012.

[6] T. Fletcher. Support vector machines explained, 2009.

[7] R. M. Freund. Applied lagrange duality for constrained optimization, 2004.

[8] Jeff M. Phillips and Suresh Venkatasubramanian. A gentle? introduction to the kernel distance, 2011.

[9] D. Lindsay. Matlab workshop 2: An introduction to support vector machine implementations in matlab, 2003.

[10] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. *ArXiv e-prints*, Nov. 2015. 39

[11] Maksim Lapin, Matthias Hein, and Bernt Schiele. Learning using privileged information: Svm+ and weighted svm, 2014.

[12] C. H. Martin. Svm+ / lupi: Learning using privileged information, 2014.

[13] Pechyony, D., Izmailov, R., and Vashist, Vapnik. SMO-style algorithms for learning using privileged information. *International Conference on Data Mining*, pages 235–241, 2010.

[14] R. Nilsson, J. Bjrkegren, and J. Tegnr. A flexible implementation for support vector machines, 2006.

[15] R. Rifkin. Multiclass Classification, 2008. URL http://www.mit.edu/~9.520/spring09/Classes/multiclass.pdf. 3

[16] C. Serra-Toro, V. J. Traver, and F. Pla. Exploring some practical issues of svm+: Is really privileged information that helps? *Pattern Recognition Letters*, 42:40 – 46, 2014. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2014.01.013. URL http://www.sciencedirect.com/science/article/pii/S0167865514000270. 6

[17] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Learning to Transfer Privileged Information. *ArXiv e-prints*, Oct. 2014.

[18] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Learning to transfer privileged information. *arXiv preprint arXiv:1410.0389*, 2014. 19

[19] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. 26

[20] Stephen Boyd and Lieven Vandenberghe. Convex optimization, 2009. http://web.stanford.edu/ boyd/cvxMISC/bv_cvxMISC.pdf.

[21] Taylor, Joseph G., Sharmanska, Viktoriia, Kersting, Kristian, Weir, David, and Quadrianto, Novi. Learning using Unselected Features (LUFe). *IJCAI*, 25:2060–2066, 2016. 27

[22] Vapnik, Vladimir and Izmailov, Rauf. Learning Using Privileged Information: Similarity Control and Knowledge Transfer. *Journal of Machine Learning Research*, 16:2023–2049, 2015. 2, 4, 5, 15, 22, 60

[23] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information, 2009.

[24] X. Xu, J. Tianyi Zhou, I. Tsang, Z. Qin, R. Siow Mong Goh, and Y. Liu. Simple and Efficient Learning using Privileged Information. *ArXiv e-prints*, Apr. 2016.

[25] X. Xu, J. T. Zhou, I. Tsang, Z. Qin, R. S. M. Goh, and Y. Liu. Simple and efficient learning using privileged information. *arXiv preprint arXiv:1604.01518*, 2016. 42

[26] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017. 35, 36

[27] Zemel, Richard, Wu, Yu, Swersky, Kevin, Pitassi, Toniann, and Dwork, Cynthia. Learning Fair Representations. *JMLR W&CP*, 28:325–333, 2013.

[28] X. Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, pages 4083–4087, 2015. 39