University of Sussex

Privileged Learning Using Unselected Features

Joseph Gerard Taylor

Submitted for the degree of Doctor of Philosophy

University of Sussex

October 2018

Declaration

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature:

Joseph Gerard Taylor

UNIVERSITY OF SUSSEX

Joseph Gerard Taylor

Submitted for the degree of Doctor of Philosophy

Privileged Learning Using Unselected Featurers

This thesis proposes a novel machine learning paradigm called Learning using Unselected Features (LUFe), which front-loads computation to training time in order to improve classifier performance, without additional cost at deployment. This is achieved by repurposing and combining techniques from feature selection and Learning Using Privileged Information (LUPI). Feature selection is a means of reducing model complexity, which enables deployment in devices with limited computational power, but this can waste additional resources which may be available at training time. LUPI is a paradigm that allows extra information about the training data to be harnessed by the learner, but this requires an additional set of highly informative attributes. In the LUFe setting, feature selection is used to partition datasets into primary and secondary subsets, instead of discarding the features which are unselected. Both datasets are then passed to a LUPI algorithm, enabling the secondary feature-set to provide additional guidance at training time only, in place of 'privileged' information. Only the selected features are used at train time, maintaining low-cost deployment while exploiting train-time resources.

Experimental results on a large number of datasets demonstrate that LUFe facilitates an improvement in classification accuracy over standard feature selection approaches in a majority of cases. This performance boost is consistent across a range of feature selection approaches, and is largest when the SVM+ algorithm is used for implementation. This effect is shown to be partially dependent on the usage of information in the unselected features, as well as resulting from the presence of additional constraints on the function space searched for the model. The enhancement by LUFe is shown to be inversely correlated with the performance of standard feature selection and mediated by a further reduction in model variance, beyond that provided by standard feature selection. Aside from demonstrating the direct practical benefit of LUFe, this work makes the contribution of broadening the scope of applications for the LUPI framework.

Acknowledgements

I would like to start by thanking my family for all their constant encouragement: my parents Ged and Marian, my sister Dee, my niece Hannah, and my nephews Alfie and Alex. Thank-you to my wonderful partner Clemy who has been amazing and supportive of me. Thanks to everyone I have shared an office with, and last but not least, thanks to my supervisors Novi and David.

Contents

1 Introduction **2**

1.1 Classification with limited resources . . . 2

1.2 The basis for this research: LUPI and feature selection . . . 3

1.2.1 Limitations of current work . . . 4

1.3 Introducing LUFe: hypothesis and research questions . . . 4

1.4 Outline . . . 5

1.5 Summary of contributions . . . 6

1.6 Outline of publications . . . 7

2 Background **8**

2.1 Learning Using Privileged Information . . . 8

2.1.1 Motivation for Learning Using Privileged Information . . . 9

2.1.2 LUPI in context . . . 11

2.1.3 Technical explanation of LUPI . . . 11

2.1.4 SVM . . . 12

2.1.5 Oracle function . . . 16

2.1.6 SVM+ . . . 17

2.1.7 *d*SVM+ . . . 22

2.1.8 $SVM_{\Delta}+$ . . . 23

2.1.9 Margin Transfer . . . 25

2.1.10 Beyond supervised learning: GPC+ . . . 27

2.1.11 Beyond classification: unsupervised LUPI . . . 27

2.1.12 Assessing SVM+ performance . . . 28

2.1.13 Questioning the mechanism of LUPI . . . 29

2.1.14 Comparison to other approaches . . . 30

2.2 Feature Selection . . . 32

2.2.1 Filter methods . . . 33

2.2.2 Wrapper methods . . . 34
2.2.3 Embedded methods . . . 35
2.2.4 Combining methods . . . 36
2.2.5 Comparing approaches in the context of this thesis . . . 36

3 Related work **38**
3.1 Using the variables that machine learning discards . . . 38
3.1.1 Unselected features in multi-task learning . . . 38
3.1.2 Comparison with LUFe . . . 39
3.1.3 Experimentation . . . 39
3.1.4 Concluding remarks . . . 41
3.2 Other approaches to front-loading training . . . 43
3.2.1 Model compression . . . 43
3.2.2 Distillation . . . 44
3.2.3 Unifying distillation and privileged information . . . 45
3.3 Other approaches with different information at train and test times . . . 46
3.3.1 Domain adaptation . . . 47
3.3.2 Transfer learning . . . 49
3.3.3 Multi-view learning . . . 51

4 Learning using Unselected Features **53**
4.1 Overview . . . 53
4.2 Motivation . . . 54
4.2.1 Use cases . . . 55
4.2.2 Feature selection revisited . . . 57
4.2.3 Rethinking feature selection . . . 59
4.2.4 Comparison with LUPI and conventional feature selection . . . 61
4.3 Experimentation . . . 62
4.3.1 Dataset . . . 63
4.3.2 Experimental protocol . . . 64
4.3.3 Implementation . . . 66
4.3.4 Results . . . 66
4.3.5 Discussion of results . . . 76
4.4 Further analysis of results . . . 77
4.4.1 Dataset size and class imbalance . . . 79

4.4.2 Dataset topics and distance . . . 80
4.5 Further experimentation 1:
Altering the number of selected features . . . 84
4.5.1 Experimentation . . . 84
4.5.2 Results and discussion . . . 84
4.6 Further experimentation 2:
Comparison with Existing Method to Utilise Discarded Features . . . 86
4.6.1 Experimental procedure . . . 87
4.6.2 Results . . . 89
4.6.3 Discussion . . . 89
4.6.4 Closing remarks . . . 94

5 Further Investigating Unselected Features 95
5.1 Feature selection methods . . . 96
5.1.1 Comparison of filter and wrapper methods . . . 96
5.1.2 Experimental procedure . . . 97
5.1.3 Results . . . 98
5.1.4 Discussion . . . 101
5.1.5 Further analysis of results . . . 102
5.1.6 Further experimentation A: Comparison with multi-task learning . . . 110
5.1.7 Further experimentation B: RFE step-size parameter . . . 111
5.2 Alternative implementations of LUPI . . . 114
5.2.1 $\text{SVM}_\Delta+$ . . . 114
5.2.2 *d*SVM+ . . . 115
5.2.3 Experimentation with different implementations . . . 116
5.2.4 Experimental procedure . . . 116
5.2.5 Results . . . 116
5.2.6 Discussion . . . 117
5.3 Using subsets of unselected features . . . 119
5.3.1 Methodology . . . 120
5.3.2 Results . . . 121
5.3.3 Discussion . . . 121

6 Investigating the mechanism of action for LUFe 125
6.1 Is LUFe dependent on Unselected Features? . . . 126

6.1.1 Experimentation . . . 126
6.1.2 Results . . . 128
6.1.3 Discussion . . . 131
6.1.4 Further experimentation: Random feature selection . . . 132
6.2 Assessing the value of unselected features . . . 134
6.2.1 Informativeness between feature sets . . . 134
6.2.2 Informativeness of individual feature sets . . . 135
6.2.3 Experimentation . . . 136
6.2.4 Results . . . 138
6.3 How does the LUFe setting improve performance? . . . 148
6.3.1 Learning curves . . . 148
6.3.2 Experimentation . . . 148
6.3.3 Discussion . . . 150

7 Conclusion **153**
7.1 Summary of results . . . 153
7.2 Limitations . . . 154
7.2.1 Limitations to the scope of findings . . . 154
7.2.2 Limitations of applications for LUFe . . . 155
7.2.3 Limitations of assessment methods . . . 155
7.3 Further work . . . 156
7.3.1 Further work to address limitations . . . 156
7.3.2 Other areas for further work . . . 156
7.4 Closing remarks . . . 157

Bibliography **158**

Preface

This thesis is the result of work carried out in the Text Analytics Group and Predictive Analytics Laboratory, in the Department of Informatics at the University of Sussex. An earlier version of Chapter 4 was published as Learning using Unselected Features (LUFe), in the proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016.

Chapter 1

Introduction

1.1 Classification with limited resources

The modern world is a data-rich environment, and suitable tools to understand, categorise and classify this data are more necessary now than ever. One general approach is machine learning. This broad term encompasses methods that enable systems to learn to perform a task from data, without requiring specific instructions. As computing becomes more pervasive, more devices and sensors require the capability to perform 'intelligent' tasks. Training the model may be centralised, and have access to great computational resources, but at deployment, the model may have access to limited resources only (Hinton et al., 2015). The need therefore rises for 'asymmetrical' systems, which allow a model to exploit additional train-time resources, which are not required at deployment time — effectively 'front-loading' the burden of computation.

As an illustrative example, consider the case of a sensor array on an autonomous robot, which performs some initial processing to inform the robot's actions — such as identifying and classifying an object. In this environment, each sensing unit must be able to respond to a constant stream of data and produce output with very low latency. The model in this scenario therefore requires the ability to handle high data velocity with a fast response rate. A smaller, less complex model built on fewer features would typically provide this, but it may be at the expense of accuracy. However, if this lighter-weight model were trained on a larger feature set offline before deployment, perhaps the accuracy could be improved while maintaining the rapid deployment time. It is this insight which motivates the work in this thesis. The purpose of this work, then, is to introduce, validate and examine a new learning paradigm called Learning using Unselected Features (LUFe), which is designed to improve classifier performance without increasing computational cost

at deployment time.

This thesis will focus on the task of classification, which is among the most common types of function that can be accomplished via machine learning. Classification involves learning to correctly assign a label indicating class membership to data points. and can be applied to countless domains, including textual data, images, or financial data (Bishop et al., 2006). Machine learning is typically categorised into two broad categories: *supervised learning* which learns from examples of input-output pairs, and *unsupervised learning*, which finds structure from input data without an associated output. Classification is usually handled as a supervised learning task, which takes pairs of training input feature vectors with output class labels, and learns a function that attempts to map each input vector to its corresponding label. Once learned, the function can then be applied to classify a new data point, by assigning it a label.

Recently, methods such as model distillation (Hinton et al., 2015) and model compression (Buciluǎ et al., 2006) have been developed in order to allow a relatively small neural network to learn from a larger, more complex neural network-based model. These approaches allow the usage of a high-performing model with low deployment cost. This thesis presents a related but novel manner of front-loading the computational cost of a model, with a focus on SVM-based classification.

1.2 The basis for this research: LUPI and feature selection

This thesis builds upon two existing machine learning methods: Learning Using Privileged Information (LUPI) and Feature Selection. Learning Using Privileged Information is a relatively novel paradigm that allows additional, highly informative ('privileged') information about training instances to be harnessed during training, even though equivalent information will not be accessible for unseen instances at deployment time (Vapnik and Vashist, 2009). The privileged information is typically from a different domain and represented in a separate feature space. Given that the privileged information is not available at deployment time, it is harnessed through a 'teacher function' that guides the 'student function' to learn a better decision function in the standard feature space. This is accomplished by improving the student function's convergence rate to the optimal solution. For a given amount of training data, a faster-converging solution will get closer to the optimal solution.

Feature selection is a widely-used approach to reduce model size, that is employed in a range of machine learning tasks including classification (Guyon and Elisseeff, 2003). This

involves selecting a subset of the most relevant attributes from the available dataset, and typically using only this subset in modelling of the problem. Feature selection produces benefits including improvements to human interpretability, to the cost of storage and data collection, and to the computational cost of training and deploying the model. It can also improve model performance by reducing variance, and preventing overfitting to noisy or irrelevant attributes. Approaches to feature selection are usually grouped into 'filter', 'wrapper' and 'embedded' types, based on how they approach this task.

1.2.1 Limitations of current work

Existing research on LUPI and feature selection approaches has certain limitations, which are addressed by the work in this thesis. LUPI is an exciting and powerful method of boosting performance compared to conventional supervised learning, but requires a specific scenario: a supervised learning task to be performed, with a standard dataset, and an additional, highly informative dataset which is available for all training instances. Its application has therefore been limited to those settings where a secondary dataset, providing an alternate 'view', is accessible for training instances. However, as a relatively new learning setting, there remains a lack of empirical work to validate that LUPI actually requires a highly-informative secondary data set, as the literature describes.

Feature selection is effective at reducing model complexity but may lose information contained in the discarded features. This is particularly an issue in filter and wrapper approaches, which discard all attributes that are not included in the selected subset passed to the learner. Feature selection is an NP-hard problem (Weston et al., 2003), so most approaches tend to generate a sub-optimal feature set, and therefore lose information in the unselected features. Even if feature selection were to perform optimally, there remains a trade-off between size of the selected subset and model performance.

1.3 Introducing LUFe: hypothesis and research questions

This setting involves a novel combination of LUPI and feature selection, based on the insights from the previous section that (a) the LUPI framework may be under-used and more broadly applicable and (b) standard feature selection approaches tend to be sub-optimal. This is used to address the previous observation that front-loading computation has practical benefits. To summarise this approach: a standard feature selection algorithm is performed to partition the dataset into 'selected' and 'unselected' subsets, which are both then supplied to an implementation of LUPI. The selected features are supplied as

the primary feature set, to be used at both train time and deployment time, and the unselected features are passed in place of the 'privileged' features, for training data only.

In this manner, both LUPI and feature selection are repurposed: feature selection is now used to split the data into subsets of 'primary' and 'secondary' importance, rather than being used to assign useful and discarded feature sets. The LUPI framework is broadened in scope, and its use-case extended; the secondary dataset may now consist of features which have been assigned as *less* informative attributes of the same domain by the feature selection process, rather than more informative attributes from a different domain.

The benefit of this research is twofold. Firstly, the practical application of LUFe allows model accuracy to be boosted with no additional cost at deployment time. This effectively allows computation to be 'front-loaded', with more computational resources exploited at training time in order to learn a better model, but without additional cost when the model is being deployed. The second benefit is to question the theoretical underpinning of the LUPI framework. The expansion of this paradigm by using a *less* informative secondary feature set to improve performance demonstrates that the LUPI model is not entirely dependent on the availability of highly informative 'privileged' information, as described in the majority of literature on this topic.

The research asks the following questions. Firstly, whether the LUPI framework can be re-purposed to improve model performance by using unselected features — rather than privileged information. Secondly, whether the performance of LUFe is consistent when implemented using different LUPI algorithms, and when the primary and secondary inputs are assigned by different feature selection algorithms. Thirdly, whether any improvement due to LUFe is actually dependent on information contained in the unselected features, and how this benefit is accomplished.

1.4 Outline

This thesis begins with a two-part chapter which summarises the main areas of background work which LUFe builds upon. The first part describes the state of current research on LUPI, describing the motivation for this framework, the technical details of various LUPI implementations, and reporting some experimental results achieved using this paradigm. The second part describes the three main categories of established approaches to feature selection, and describes the benefits and challenges associated with each. The second chapter then continues to review the literature, describing other related work: namely,

existing approaches taken to front-load computation, and more broadly, other work which takes an asymmetric view of the data at train time and deployment time.

The following three substantive chapters each then contain the main contributions of this work. Firstly, the LUFe paradigm is introduced, along with motivation and theoretical justification, and an experimental framework to test its efficacy is described. This is used to provide proof-of-concept experimentation, where LUFe is shown to provide a performance boost in a majority of classification tasks on 295 datasets.

Secondly, the LUFe framework is examined in a wider range of settings: its performance is assessed in combination with a range of feature selection methods, with different implementations of the LUPI paradigm, and with different subsets of unselected features. The performance boost due to LUFe is shown to be robust across a range of different feature selection methods, but performance is affected by the choice of LUPI algorithm for implementation.

The third substantive chapter then investigates the mechanism by which LUFe works. LUFe is compared with performance using 'dummy' datasets to assess the impact of the unselected features. Metrics to predict the LUFe performance boost *a priori* are described and tested. Learning curves are used to investigate the effect of LUFe on the bias-variance trade-off. The implications for the LUPI framework are discussed.

Conclusions, interpretations, limitations, and areas for future work are then discussed in a final chapter.

1.5 Summary of contributions

The contributions of this thesis are summarised as follows:

1. The introduction of the Learning using Unselected Features (LUFe) paradigm, to improve classifier performance in scenarios with limited resources at deployment, by front-loading computation to training time

2. The novel combination of Learning Using Privileged Information (LUPI) with feature selection, which repurposes LUPI algorithms to use a less informative secondary feature set

3. Validation of LUFe performance from experimental results on 295 datasets, demonstrating a robust improvement that is dependent on the implementations of feature selection, and of LUPI

4. Experimentation that investigates the mechanism of action for LUFe, demonstrating a dependence on the efficacy of feature selection

1.6 Outline of publications

The following publication resulted from this thesis:

- Taylor, J., Sharmanska, V., Kersting, K., Weir, D., and Quadrianto, N. (2016). Learning using Unselected Features (LUFe). In Proceedings of the Twenty-Fifth International Joint 144 Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. AAAI Press/International Joint Conferences on Artificial Intelligence. 56

Chapter 2

Background

The work in this thesis bridges a gap between two distinct topics in machine learning: **Feature Selection** and **Learning Using Privileged Information**. The intersection of these fields is a novel area of research, so the background material will be described in terms of these topics as two discrete sections.

2.1 Learning Using Privileged Information

Learning Using Privileged Information (LUPI) is a machine learning paradigm that can improve performance by incorporating some additional information when training a model, which is not available at test time (Vapnik and Vashist, 2009).[1] This section will first provide a high-level description of the motivating concepts for the LUPI framework, with particular focus on its usage in the supervised classification setting which is most relevant to this thesis. Then the technical details underlying LUPI will be tackled in more depth: firstly, the general principles of supervised classification and specifically the support vector machine (SVM) classifier are reviewed. This will lay the groundwork to then explain how the SVM+ instantiation of the LUPI paradigm can employ additional information to improve upon this performance. Alternative LUPI implementations are discussed, such as dSVM+, SVM$_\Delta$+ and GPC+. Finally, some experimental results using LUPI are described.

[1] This kind of learning was first described in the afterword of Vapnik (2006) as 'Learning Using Hidden Information' but is referred to as 'Learning Using Privileged Information' in all subsequent work, to stress the informativeness and limited accessibility of the secondary dataset.

2.1.1 Motivation for Learning Using Privileged Information

Supervised machine learning typically involves learning from training data that consists of paired inputs and outputs. Training consists of learning a function that maps each input instance to its corresponding output; this function can then be applied to unlabelled data to predict an output value. In a supervised classification task, the output consists of a label indicating class membership, whereas in regression, the output is a continuous variable.

The standard supervised learning framework requires that a given set of attributes are available both for the training data, and for the testing data, as the model which is learned on the training data then requires similar input to make predictions. This is restrictive, as additional information which is available for *only* the training data cannot be incorporated into the classification rule. Even if some highly-informative 'privileged information' were available to describe training data, it could not be harnessed to improve performance. Learning Using Privileged Information (LUPI) is a framework that expands the supervised learning setting by lifting this restriction, allowing extra information at training time to be incorporated into the learning process — even though corresponding information will not be available for test data points (Vapnik and Vashist, 2009).

'Privileged information' — defined as highly informative and available only for train data — is ubiquitous in real-world, applied machine learning settings (Vapnik and Vashist, 2009). The feature set which is used to train a model is a single representation of some underlying ground truth, which also can be represented in other modalities. However, these other representations may be difficult or impossible to acquire, for reasons including temporal, computational, or financial constraints. As such, this extra information would only be available for the training data. The following hypothetical examples demonstrate the motivation for LUPI, by illustrating how potential privileged information is frequently available for assistance in binary classification tasks:

- **Task:** Binary image classification task to identify whether there is a cat present in a photograph.
  Standard information: Set of 2D visual-spectrum digital images.
  Privileged information: Additional infra-red images of the same scenes, available for training set only.

- **Task:** Predicting whether a business is in profit after one year.
  Standard information: A numerical dataset listing financial and operational information for each business

Privileged information: An auditor's report written in technical language, available for training data only.

- **Task:** Labelling whether a patient has a particular neurological condition.
  Standard information: Bag-of-words representations of doctor's notes
  Privileged information: an MRI scan of the patient's brain, available for training cases only.

In each of these cases, we can see that the desired function would map from the standard feature-space to the corresponding label, and also see that the additional 'privileged' information is highly informative about the training instances. However, despite being useful, it is not available at deployment time because it requires further efforts to collect, using additional equipment and/or expert knowledge. These use-cases therefore demonstrate the need for a novel method that could incorporate the extra data at training time only. Having seen the motivation for LUPI, we now turn to the next question, of *how* this could be implemented.

Vapnik and Vashist (2009) use the analogy of human learning to explain how LUPI can use the additional information. They propose the concept of a 'teacher function' that could be used to inform the learning process of a 'student function' and improve its rate of convergence to an optimal solution. In this case, the privileged information can be described as analogous to the assistance provided by a teacher: "explanations, comments, comparisons, and so on" which are provided to assist understanding of training cases but are not available at test time (Vapnik and Izmailov, 2015). Expanding on this comparison with human learning, they reference a Japanese proverb to explain the benefit of using a teacher function: "Better than a thousand days of diligent study is one day with a great teacher."

This is the means by which privileged information can be useful despite its limited availability. The predictive model still learns to map from only the standard feature set to an output label, and can therefore still be applied to test cases for which only the standard features are available. However, the additional information is used to *guide* the learning process at train time. In other words, *the privileged information helps a better model to be learned on the standard information.* Just as a human teacher instructs the student during lesson time, but the student has to then apply the learned knowledge without assistance, so too does the teacher function only assist the student at training time; at deployment there is no extra information to serve as a teacher.

2.1.2 LUPI in context

We have seen how LUPI is a method to incorporate extra information at training time, thereby front-loading computation and boosting performance. Two other techniques that take a similar approach are model compression and model distillation, which will be described in more detail in 3.2. Model compression involves training a complex 'target' system on the dataset, to label some unlabelled data, then training simpler 'mimic models' on the target model's predictive output (Ba and Caruana, 2014). In that work, the logits from a target neural network were used, allowing a more robust mimic model to be learned than if it were trained directly on binary output labels.

Model distillation similarly trains a mimic model to learn the generalisations made by a target model (Hinton et al., 2015), but the mimic learns the probabilities from softmax, rather than the logits. Lopez-Paz et al. (2015) unite the concepts of distillation and LUPI into a unified framework, and define a learning learning model which describes both paradigms. In this framework, LUPI and distillation are both defined as a trade-off between two loss functions: firstly between the student model output and the true labels, and secondly between the student model logit and the output from the learner. This defines LUPI's similarity to distillation, and by extension to compression. However, they differ in that the second error term for LUPI is calculated using the privileged features, but for distillation it is the same primary feature set.

LUPI can also be defined in terms of its combination of learning domain (X and X^*) and target domain (X only). More broadly, other machine learning approaches with multiple domains are defined in Section 3.3, and can be compared to LUPI in these terms. Domain adaptation is the field of training in a source domain (X) and deploying in a target domain with the same attributes but different (X^*). Transfer learning trains in source domain (X) with labels (y) and deploys in different domain (X^*) with different labels too (y^*). Multi-view learning trains *and* deploys on two domains: domain X with labels y and domain X^* with labels y^*.

2.1.3 Technical explanation of LUPI

Having seen this brief overview of why the LUPI framework was developed, and how it can be seen to work, the following sections will describe the implementation of this paradigm in more technical detail. In order to contextualise how this works, firstly, the classical supervised learning approach will be summarised, with particular emphasis on the SVM classifier. The LUPI paradigm, and its SVM+ instantiation, can then be described in

terms of how they enhance this.

Note on nomenclature

- Matrices are represented as upper case letters (eg X)
- Vectors are represented as bold lower-case letters (eg $\mathbf{x}$)
- Scalars are represented as lower-case, non-bold letter (eg y)
- Feature spaces are represented as upper-case calligraphic letters (eg $\mathcal{X}$)
- Dataset X consists of n data points.
- Subscript is used to denote the index of a given point within the dataset, $\mathbf{x_1}$,... $\mathbf{x_n}$.
- Superscript is used to denote the index of a given attribute within the dataset
- Each instance has dimensionality d so $\mathbf{x} = x^1...x^d$
- LUPI introduces a secondary dataset X^*, also consisting of n datapoints

2.1.4 SVM

Supervised learning is an area of machine learning concerned with learning a function that maps an input to an output, from a series of input-output pairs (Russell and Norvig, 2016). Supervised classification is a form of supervised learning where the learned output is a label that indicates class membership. Formally, the data consists of iid pairs:

$$(x_1, y_1), ..., (x_l, y_l), \quad x_i \in \mathcal{X}, \quad y_i \in \{-1, +1\}$$

The learning process seeks to find a function $y = f(x, \alpha_*)$ that minimises the probability of incorrect classifications, minimising the risk functional (Vapnik, 1999):

$$R(\alpha) = \tfrac{1}{2} \int |y - f(x, \alpha)| dP(x, y)$$

The expected generalisation error when the classification function applied to new data can be decomposed into three components: bias, variance, and the irreducible error due to the noise in the system (Domingos, 2000). Bias refers to the difference between the expected prediction of the model, and the correct target value; a high-bias classifier is one which *underfits* the data and fails to sufficiently capture the underlying patterns. Variance refers to the variability of prediction for a given data point; a high-variance classifier will *overfit* the data and fail to be applicable to unseen data points from outside the training

set. In order to minimise the risk of error on unseen test data, the classifier needs to balance bias and variance. This *'bias-variance tradeoff'* (or dilemma) is a core problem in machine learning, as a successful model must capture the generalities of the dataset while remaining generalisable to new and unseen instances at test time.

Binary classification tasks typically involve the placement of a separating hyperplane between classes, which is known as the decision boundary. The support vector machine (SVM) (Cortes and Vapnik, 1995)[2] is a very widely-used classifier which takes a "maximum margin classifier" approach, in order to tackle the bias-variance tradeoff. That is, it seeks to place a decision boundary between the classes, such that the distance from the hyperplane to instances of both classes is maximised. In doing so, the model attempts to learn an optimal boundary that minimises training error, while also remaining generalisable. Intuitively, we can see that this boundary will be more robust, and generalisable to unseen instances at deployment. The margins allow 'room for error' so that even if a data point is inside the margin, it may be still be correctly classified, whereas a boundary without margins would be closer to one class and therefore make misclassification more likely. The model is named for the 'support vectors': instances of each class which are closest to the decision boundary.

In the linear case, the decision boundary is defined by a weight vector $\mathbf{w} = \{w_1, ..., w_d\}$ and bias b. Each element of $\mathbf{w}$ is a coefficient that corresponds to a single dimension of the training data. For a data point $\mathbf{z}$, the decision function is $f(\mathbf{z}) = (\mathbf{w}, \mathbf{z}) + b$. The learning process consists of learning the parameters $\mathbf{w}$ and b such that the decision boundary is positioned to correctly partition the feature space so the dataset is split into the two classes. This is equivalent to learning a function $f(\mathbf{z})$ such that $f(\mathbf{z}) > 0$ for items in class $+1$ and $f(\mathbf{z}) < 0$ for items in class -1. This can be succinctly expressed as

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 0, \quad \forall i \in \{1...n\} \tag{2.1}$$

Let us first consider the case of separable data, which is that which can be classified without errors:

$$y_i f(\mathbf{x}_i, \alpha_l) > 0 \quad \forall i = 1...l \tag{2.2}$$

The weight vector is canonically set such that for any given support vector $\mathbf{x}_+$ in class 1, or $\mathbf{x}_-$ in class -1, the classification function $f(\mathbf{x}_+) = 1$ and $f(\mathbf{x}_-) = -1$. The weight vector is orthogonal to the decision boundary, so that for any point $\mathbf{x}_b$ on the boundary, $f(\mathbf{x}_b) = 0$.

[2]The algorithm was referred to as 'support vector network' in this initial publication

It can then be shown that the size of the margin measured from the decision boundary is $\frac{2}{\sqrt{||\mathbf{w}||^2}}$. Therefore, the stated goal of maximising the margins can be achieved by minimising the weight vector $\mathbf{w}$, subject to the correct classification of all datapoints $\{z_1, ..., z_n\}$. This leads to the formulation of the primal problem to find optimal weights $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} = \arg\max_w \frac{2}{||\mathbf{w}||} = \arg\min_w (||\mathbf{w}||^2) \tag{2.3}$$

subject to:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \quad \forall i \in \{1...n\} \tag{2.4}$$

In this optimisation, the objective function simply seeks to minimise the weight (with the numerator set to 2 for numerical convenience) and the constraint enforces correct classification of all datapoints. [3]

The introduction of Langrangian multipliers allows the contstraints to be rewritten as

$$1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq 0, \forall i \in \{1...n\} \tag{2.5}$$

and incorporated into the objective function, providing a convex optimisation problem, to obtain optimal parameters $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} = \arg\max_w \min_\alpha \ ||\mathbf{w}||^2 + \sum_{i=1}^{N} \alpha_i (1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)) \tag{2.6}$$

In the linearly separable case, $\alpha > 0$ only for the datapoints on the margin.

Slack variables

In practice, a dataset is unlikely to be linearly separable so a 'hard margin' classifier which correctly classifies each point is not applicable; it is impossible to place the hyperplane without margin violations. To deal with this case, an alternative 'soft margin' formulation is used, that allows margin violations, but deals with them by assigning a non-negative penalty, ξ_i to each. This 'slack variable' ξ_i is zero for correctly classified data points and positive for those which violate the margin. The slack variables are incorporated into the objective function so that they are minimised — ensuring that the decision boundary is placed to enable maximally correct classification. For canonical margin size = 1, this

[3] For simplicity of notation, we can define an extra feature x_0, equal to 1 for all instances, and then consider the bias as an additional corresponding feature weight w_0. The classification function can then simply be represented as $f(\mathbf{x}) = (\mathbf{w}, \mathbf{x})$.

means that the penalty will be greater than 1 for misclassifications, and $0 \leq \xi_i \leq 1$ for points that are on the correct side of the decision boundary but violate the margin.

The objective function in primal form, then is

$$\begin{aligned} \hat{\mathbf{w}} = &\arg\min_{w,b,\xi} \ \mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{subject to}: \quad &y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i, \ \forall i \in \{1...n\} \end{aligned} \tag{2.7}$$

As before, the model is learned through minimisation of the weight vector, but now the slacks are also minimised, in order to minimise the amount of margin violations. C is a scalar regularisation parameter [4], which sets the relative impact of the slacks and the weight vectors on the objective function. In effect, this controls the bias-variance trade-off of the classifier. A lower C means that the minimisation focuses less on reducing the slacks, and more on reducing the magnitude of $\mathbf{w}$, producing a lower-variance, generalisable classifier that may have more errors on the training set. Conversely, setting a higher C means that the slacks contribute more to the objective function, so a lower bias classifier with smaller training error is produced, but this may be less generalisable to new data.

As with the linearly separable form, the constraints can be incorporated into the Langrangian dual form:

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \ - \sum_{i=1}^{N}\alpha_i[y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b] - 1 + \xi_i] - \sum_{i=1}^{N}\beta_i, \xi_i \tag{2.8}$$

which is solved by: minimising over w and maximising over α and β.

The convergence rate for the non-separable case is much slower than for the separable case, likely because it requires more parameters to be estimated from the same number (N) of training observations (Vapnik and Vashist, 2009). The separable case requires estimation of $d + 1$ weight parameters — one for each feature dimension, and the bias, while the non-separable case needs to learn $d + N + 1$ parameters: a slack for each training instance, in addition to the same number of weights and bias. The guaranteed rate of convergence for the separable case is of order $O(h/N)$, and $O(\sqrt{h/N})$ for the non-separable case one, where h is the VC dimension of the set of admissible hyperplanes.

Non-linearity and the 'kernel trick'

The SVM — among other learning methods — has a key feature referred to as the 'kernel trick' that hugely increases its discriminative capability (Bishop et al., 2006). So far,

[4] By convention this parameter is denoted by upper-case C so this will be adhered to

we have seen the classifier operating in the feature space $\mathcal{X}$, but some problems are not separable in this space. Intuitively, this could be fixed by mapping data an alternative feature space $\mathcal{Z}$, and solving the classification problem in this space. A linear operation in $\mathcal{Z}$ would be equivalent to a non-linear operation in $\mathcal{X}$, and so a more complicated decision function to be learned by the SVM.

This mapping is carried out by applying a function $K(\mathbf{x}_i, \mathbf{x}_j)$ to pairs of data points, and any function that produces a symmetric, positive definite kernel $K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$ may be used. This kernel function may be considered as a kind of 'distance measure' between data points. Example functions include the polynomial kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i, \mathbf{x}_j) + 1)^d \tag{2.9}$$

where d is the degree of the polynomial, and radial basis function (RBF) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/(2\sigma^2)) \tag{2.10}$$

where σ is the width of the Gaussian.

Subsequent descriptions of learning methods will follow the example of Vapnik and Vashist (2009) and refer to data $\mathbf{x}$ in space X mapped to data $\mathbf{z}$ in space Z.

2.1.5 Oracle function

The preceding summary of supervised learning and SVM sets the scene to explain the benefit of Learning Using Privileged Information. This is introduced in the literature (Vapnik and Vashist, 2009) through a kind of thought experiment which describes a hypothetical 'oracle function'. This function ξ could enhance the performance of an SVM-type model on unseparable data, by providing slacks to the learner. The oracle function is defined as follows:

$$\begin{aligned} \xi(x) =& [1 - y_i(\langle \mathbf{w}^0, \mathbf{x} \rangle) + b^0)]_+ \\ \text{which satisfies} \quad & y_i(\langle \mathbf{w}^0, \mathbf{x}_i \rangle + b^0) \geq 1 - \xi_i^0, \quad \forall (\mathbf{x}_i, y_i) \\ \text{where} \quad & \xi_i^0 = \xi(\mathbf{x}_i) \end{aligned} \tag{2.11}$$

Use of the oracle function effectively replaces the data-label pairs $(\mathbf{x}_1, y_1)...(\mathbf{x}_N, y_N)$ with data-slack-label triplets $(\mathbf{x}_1, \xi_1^0, y_1)...(\mathbf{x}_N, \xi_N^0, y_N)$. Only the $d+1$ parameters (weights and bias) now need to learned from the normal data, instead of learning $d + N + 1$ parameters previously in the non-separable case (weights, slacks and bias). This would improve the convergence rate for the non-separable case; only the d weights would need to be estimated, so convergence would return to that of the separable case.

The oracle example illustrates a route through which additional information about training examples can improve the performance of a classifier: by changing the bound on the rate of convergence; "The goal of the LUPI paradigm is to use privileged information to significantly increase the rate of convergence" (Vapnik and Vashist, 2009). By reducing the number of parameters to be learned, the rate of convergence to the optimal solution is improved. This is beneficial because the SVM will converge on the Bayesian optimal solution with sufficient training examples (Cortes and Vapnik, 1995). In practice, however, the amount of available training data is finite and may be insufficient for optimality. Therefore, the rate of convergence is important: if we have a given, limited amount of training data, a faster-converging algorithm will perform better by getting closer to the optimal solution. Given that the slack variables are now provided, the objective function to be minimised reverts back to simply $||\mathbf{w}||^2 + b$, as in the separable case seen in equation 2.3. The significance of convergence rate on classifier performance is illustrated in 2.1.

To summarise, the improved conversion rate that is provided by the oracle function enables an optimal solution to be reached faster, where this is possible. In cases where it is not possible, the improved rate allows a better solution to be reached which is closer to optimal, and thus higher-performing. This thesis focuses on the supervised learning task of classification. In this context, improved conversion is therefore equivalent to improved classification accuracy, and differences in classifier performance will be quantified with this metric.

2.1.6 SVM+

This oracle function does not exist, but serves as a useful allegory to provide the intuition behind the working of the *SVM+*: a classification algorithm that extends SVM, to provide the primary implementation of the LUPI paradigm (Vapnik and Vashist, 2009). We have already seen the ubiquity of privileged information which could prove beneficial to a classifier, and seen how the provision of slack variables could enhance classifier performance. Combining these two observations, the SVM+ classifier uses the additional 'privileged' data at train time to approximate the slack variables.

Whereas the standard learning setting requires *data-label* pairs, and the hypothetical oracle function supplements this, to provide *data-slack-label* triplets, the real LUPI setting instead takes *standard-data–privileged-data–label* triplets as input. The LUPI paradigm allows an additional feature set $\mathbf{x}^*$ for each training data point to be incorporated into the training data, so that it consists of iid triplets:

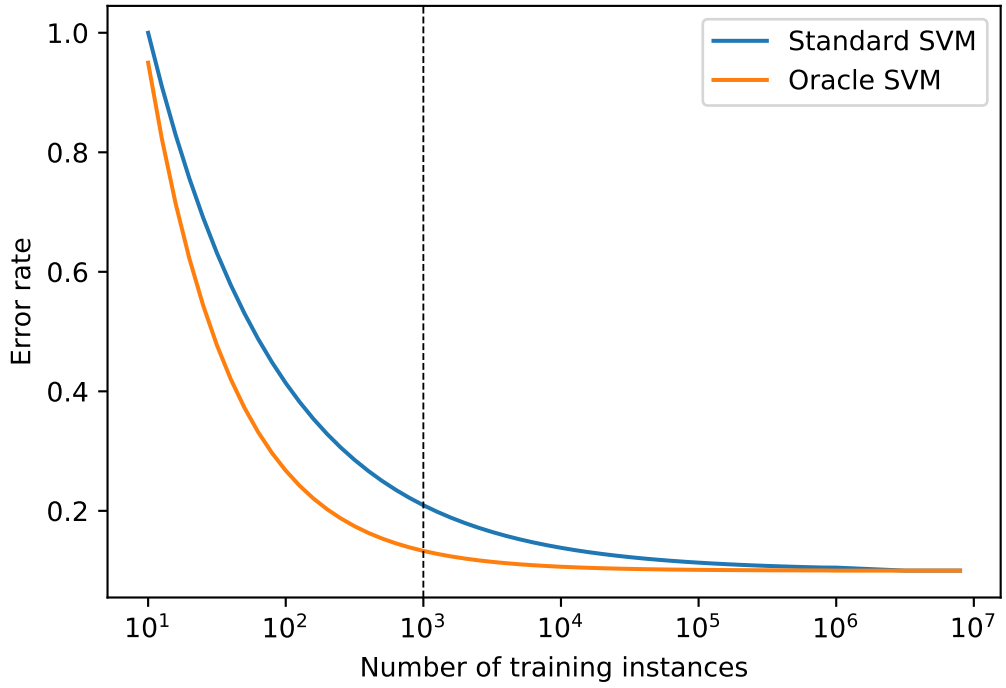

Figure 2.1: Illustrative example demonstrating the effect of convergence rate on classifier performance. Both classifiers ultimately converge on an optimal solution, but the oracle SVM does so faster. Therefore, for a given amount of training data insufficient for optimality, the Oracle SVM outperforms the standard SVM. For example, at 10^3 training examples, marked by dashed line, the faster rate of convergence means that the Oracle SVM has error rate 0.14, whereas standard SVM has error rate 0.22.

$$(\mathbf{x}_1, \mathbf{x}_1^*, y_1)), \ldots, (\mathbf{x}_n, \mathbf{x}_n^*, y_n), \quad \mathbf{x}_i \in \mathcal{X}, \quad \mathbf{x}_i^* \in \mathcal{X}^*, \quad y_i \in -1, +1$$

$\mathbf{x}_i^*$ is the privileged information that is available at training time only, generated by an unknown probability function $P(\mathbf{x}_i^*|\mathbf{x}_i)$; the test data on which the model will be deployed still consists of $(\mathbf{x}_i, y_i)$ pairs. Note that just as vector x in space $\mathcal{X}$ is mapped to vector z in space $\mathcal{Z}$, the privileged vector $\mathbf{x}^*$ in $\mathcal{X}^*$ can be mapped to vector $\mathbf{z}^*$ in space $\mathcal{Z}^*$.

The privileged information is only available at training time, so the model cannot be directly fitted to these features — for example, by concatenating the normal and privileged feature vectors and providing this as input to a standard SVM — because they will not be available at deployment time. The use of this data to estimate the slacks is therefore an appealing method to use the privileged data to inform a model which is still learned in the normal feature space.

The process of learning by the SVM+ can be described as follows: two sets of weights and biases are simultaneously learned; one in the privileged space and one in the primary feature space. The standard space weight vector and bias are denoted as $\mathbf{w}$ and b, as before, and continue to define a decision boundary in the standard feature space. The new parameters, denoted $\mathbf{w}^*$ and b^*, operate in the privileged feature space and also define a hyperplane, but this is not a decision function. Rather, it is used to inform the slack variables in the standard feature space, thereby transferring information from the privileged space. The SVM+ algorithm consists of learning these parameters through the following optimisation:

$$\begin{aligned} R(\mathbf{w}, \mathbf{w}^*, b, b^*) =& \frac{1}{2}[||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2] + C\sum_{i=1}^{N}\xi_i \\ \text{where} \quad & \xi_i = [\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle + b^*] \\ \text{subject to constraints :} \quad & y_i[(\langle \mathbf{w}, \mathbf{z}_i \rangle + b) \geq 1 - [\langle \mathbf{w}^* \mathbf{z}_i^* \rangle + b^*] \quad \forall i = 1...N \\ \text{and} \quad & (\langle \mathbf{w}, \mathbf{z_i} \rangle + b) \geq 0 \quad \forall i = 1...N \end{aligned} \tag{2.12}$$

The equivalent Lagrangian form then is :

$$\begin{aligned} \mathcal{L}(w, b, w^*, \alpha, \beta) =& \frac{1}{2}||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2 + C\sum_{i=1}^{N}[\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle + b^*] \\ & - \sum_{i=1}^{N}\alpha_i[y_i\langle \mathbf{w}, \mathbf{z_i} \rangle + b] - 1 + [\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle + b^*]] \\ & - \sum_{i=1}^{N}\beta_i[\langle \mathbf{w}^*, \mathbf{z_i}^* \rangle + b^*] \end{aligned} \tag{2.13}$$

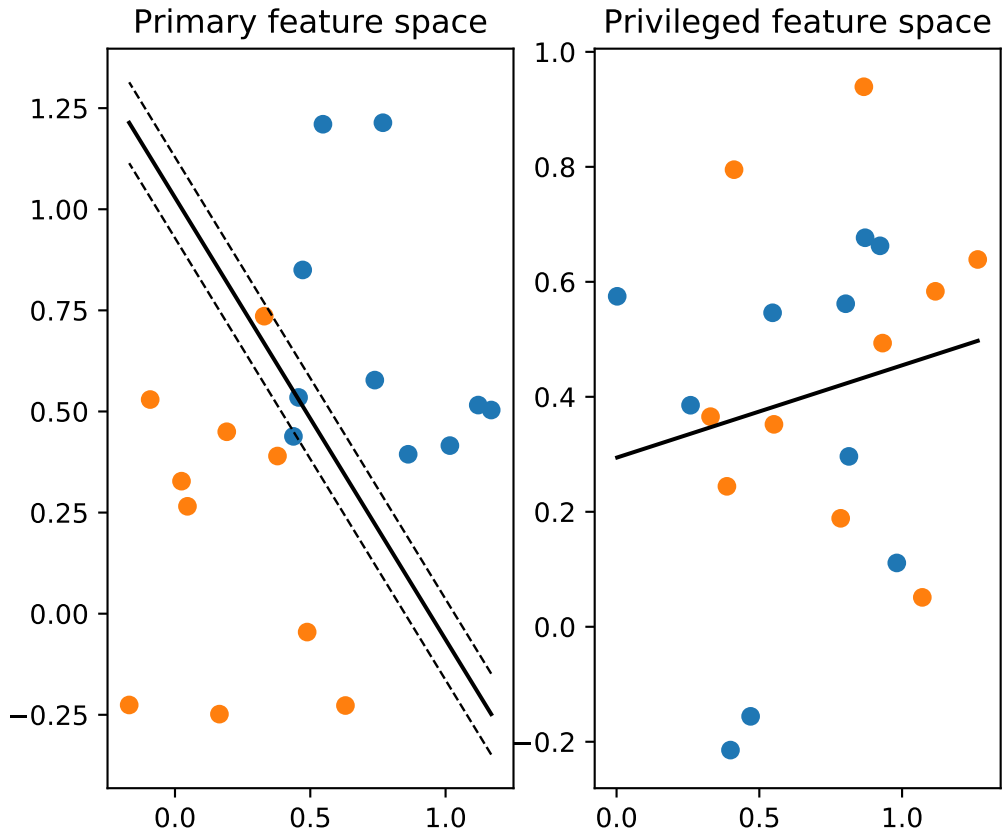

Figure 2.2: Illustrative example of the SVM+ LUPI classifer+, with two separate two-dimensional feature spaces: primary feature space (left) and privileged feature space (right). Each training data point is represented in both spaces. The decision boundary (solid line) is learned in the primary space and attempts to separate the classes, enforcing margins (dashed lines). Learning in the privileged space involves fitting a regression to the data, without taking labels into account. Information is transferred between spaces: distance from the hyperplane in the privileged space is used to inform the margin violation allowed in the primary space.

Similar to the standard SVM, this is solved by minimising with respect to $\mathbf{w}, b, \mathbf{w}^*, b*$ and maximising with respect to the Lagrange multipliers α and β. As in standard SVM, there is minimisation of the norm of the weight vector $\mathbf{w}$, and C is a weighting hyper-parameter to trade off minimising weights vs minimising training error. The first term of Equation 2.12 shows how the new γ parameter 'trades off' the relative contribution of the two weight vectors to the objective function; higher values of γ penalise the large values in the privileged weight vector more heavily, while lower values place more emphasis on minimising the standard weight vector. Note that if one feature space in LUPI is of much greater dimensionality, and so the norm of its weight vector is much larger, then γ may balance the contributions of the two vectors by downweighting it.

The final term of the objective function Equation 2.12 effectively shows a regression in the privileged space. The weights and bias in the privileged space, $\mathbf{w}^*$ and b^*, define a hyperplane, and the minimisation of this term is equivalent to fitting the hyperplane to the privileged features of all training data — of both classes. This minimisation of the distances to the hyperplane replaces the final term in the original SVM Equation 2.7, which directly minimised the slacks $(\xi_1 \dots \xi_n)$.

The distances therefore effectively serve as a data-dependent constraint, enforced in the second constraint of Equation 2.12. This error upper bound is looser for data points which have larger distances to the plane in the privileged space, so the minimisation is less concerned with correctly classifying these points. This is equivalent to providing these points with a larger, privileged data-dependent slack. In this manner, information about the dataset is transferred between feature spaces, and the privileged information is able to inform the learning of the decision boundary in the primary feature space.

Note that the data labels are not taken into account in the privileged space; all data is included in the same regression. The size of the slack in the primary space depends on the extent to which that data point is an outlier in the privileged space. Intuitively, we can take this to mean that a data point which is atypical in the privileged space can be expected to behave similarly in the primary feature space. This data point is assumed to behave in a manner which is not representative of the underlying distribution, and therefore it is difficult to predict class membership for it. Therefore, the learner should place less importance on fitting the decision boundary to correctly classify this instance. In terms of the teacher/student analogy, the privileged data is used to 'teach' the learner in the primary feature space which data points are outliers and which are more typical, thereby allowing the 'student' to focus its efforts of classification on the more typical data,

which may result in a better, more generalisable function.

Note on hyperparameters

Like the SVM, training the SVM+ requires a quadratic optimisation problem to be solved, with similar constraints (Vapnik and Vashist, 2009). However, tuning the increased number of hyperparameters for the SVM+ adds an additional computational overhead to training — for example, by performing grid-search over a larger combination of hyperparameters. In addition to the new trade-off parameter γ, any kernel parameters in the privileged space also need to be selected — for example, if an RBF kernel is used in X^*, the σ^* parameter must be set. However, this hyperparameter optimisation is performed at training time only, so deployment cost remains the same as standard SVM.

2.1.7 *d*SVM+

The *d*SVM+ is another SVM-based LUPI algorithm, proposed by Vapnik and Vashist (2009) alongside the SVM+[5]. The two approaches are algorithmically similar, with the *d*SVM+ differentiated by an additional transformative preprocessing step carried out on the privileged data. A key point of difference is that this additional step takes the labels of training data into account in the privileged space.

Rather than learning the 'correcting function' in the multi-dimensional X^* space, the *d*SVM+ correcting function is defined in a one-dimensional 'd-space' which is constructed for the purpose, as folllows. Firstly, a classifier is trained in X^*, to establish a decision boundary in this privileged space. The slack variables d_i in this space (referred to as 'deviation values') for each training data point are then used as secondary input to the SVM+, in place of using the entire privileged set. In doing so, the algorithm directly "stresses the main goal [of LUPI], to provide information about the slack variables in the simplest form" (Vapnik and Vashist, 2009).

Formally, the first step of the *d*SVM+ is to learn 'deviation values' d_i for each x_i by finding the minimisation of the following:

[5]The SVM+ in this thesis is sometimes referred to as X^*SVM+ by Vapnik and Vashist (2009) to disambiguate it from *d*SVM+. For brevity and consistency, the terminology SVM+ without specification refers to SVM+ which takes the privileged feature set directly as input to the correcting function

$$
\begin{aligned}
R(\mathbf{w}^*, b^*, \xi^*) =& \frac{1}{2}||\mathbf{w}^*||^2 + C\sum_{i=1}^{N}\xi_i^* \\
\text{subject to the constraints:} \quad & y_i[\langle \mathbf{w}^*, \mathbf{x_i}^* \rangle + b^*] \geq 1 - \xi_\mathbf{i}^* \quad \forall i \in \{1...n\} \\
\text{and} \quad & \xi_i^* \geq 0 \quad \forall i \in \{1...n\}
\end{aligned}
\tag{2.14}
$$

These deviation values are then used to supplement the training data, forming triplets $(\mathbf{x}_1, d_1, y_1)...(\mathbf{x}_N, d_N, y_N)$ where

$$
d_i = 1 - y_i[\langle \mathbf{w}_l^*, \mathbf{x}_i^* \rangle + b_l^*] \tag{2.15}
$$

where $\mathbf{w}_l$ and b_l are the solution to equation 2.14. 3 The SVM+ described in 2.12 is then trained using these triplets, with the one-dimensional d-value for each data point used as privileged information, instead of the original X^* feature set itself. The authors note that this outperforms the previous 'X^*SVM+' method in most experimentation.

2.1.8 SVM$_\Delta$+

The SVM$_\Delta$+ is an alternative LUPI classifier that is also based on the SVM (Vapnik and Izmailov, 2015). Like dSVM+, this algorithm also differs from SVM+ by taking labels into account in the privileged space. Whereas SVM+ performs a regression on the privileged features in space Z^*, SVM$_\Delta$+ learns a dividing hyperplane between the two classes in Z^*. This is similar to the process of learning a classifier, but the hyperplane is not used directly as a decision boundary; the test data is not represented in this secondary feature space so there is no need to learn a generalisable decision function here. Furthermore, the boundary that is learned in the privileged space is not a max-margin classifier like the SVM. Instead, the class boundary in the privileged space is used to learn a new non-negative parameter ζ_i for each data point x_i^*.

It would be desirable to use the function in the privileged space as an upper bound to constrain the loss function in the standard feature space; that is: minimise

$$
\begin{aligned}
R(\mathbf{w}, b, \mathbf{w}^*, b^*) =& \frac{1}{2}(||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2) + C\sum_{i=1}^{N}[y_i(\langle \mathbf{w}^*\mathbf{z}_i^* \rangle + b^*)]_+ \\
\text{subject to:} \quad & y_i[\langle \mathbf{w}, \mathbf{z}_i \rangle + b] \geq 1 - [y_i(\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle - b^*)]_+ \\
\text{where} \quad & [u]_+ = max\{0, u\}
\end{aligned}
\tag{2.16}
$$

The use of this 'positive part' function in the final term of the objective function in Equation 2.16 is hinge loss, ensuring positive penalisation for misclassified points in

the privileged space, and no penalisation for correctly classified points. Its usage in the constraint replaces the slack variable ξ_i seen in the original SVM objective function 2.7 and effectively transfers the slack from the privileged space. For datapoints $\mathbf{z}_c$ which are correctly classified in Z, the first constraint is fulfilled, as $y_i[(\mathbf{w}, \mathbf{z}_c) + b] \geq 1$ in these cases; therefore the privileged data for these points does not impact the learning in the normal space and these privileged datapoints are essentially disregarded. However, for those data points $\mathbf{z}_m$ which are misclassified in Z, $y_i[(\mathbf{w}, \mathbf{z}_m) + b] < 1$ so the slack replacement, learned from privileged data as the distance to the boundary in Z^*, comes into play. In short, the slack in Z is captured by the distance to the boundary in Z^* but only for those points which are on the correct side of the boundary.

However, the usages of this 'positive part' function means that the optimisation is non-linear. Learning the SVM$_\Delta$+ therefore consists of the following approximation of Equation 2.16, requiring the minimisation of the following:

$$
\begin{aligned}
R(\mathbf{w}, b, \mathbf{w}^*, b^*) = & \frac{1}{2}(||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2) + C\sum_{i=1}^{N}[y_i(\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle + b^*) + \zeta_i] + \Delta C \sum_{i=1}^{N} \zeta_i \\
\text{Subject to:} \quad & y_i(\langle \mathbf{w}, \mathbf{z}_i \rangle + b) \geq 1 - y_i(\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle + b^*)) - \zeta_i \\
\text{and} \quad & y_i(\langle \mathbf{w}^*, \mathbf{z}_i^* \rangle + b^*) + \zeta_i \geq 0 \\
\text{and} \quad & \zeta_i \geq 0
\end{aligned}
\tag{2.17}
$$

As seen in SVM+, the norms of the weight vectors in both spaces are minimised, with γ controlling the trade-off between the two. The ζ_i parameter can be thought of as a 'privileged slack', which ultimately impacts the decision boundary in the primary feature space. In the standard SVM, a non-margin violating point in Z would normally have little to no effect on the final decision boundary, as it would not be a support vector. However, a correctly classified point in Z, that is on the wrong side of the class boundary in Z^*, can have a big effect on the the class boundary in Z^*, which in turn affects the decision boundary in Z. This effect is mediated by the ζ_i parameter.

The ζ_i in the second constraint of Equation 2.17, serves a similar function to ξ in Equation 2.7, permitting boundary violations to occur when the weights and bias in the privileged space are learned. But unlike ξ, this constraint does not enforce margins in the privileged space, which are not required as a max-margin classifier is not learned in this space; given that these privileged parameters are learned only to inform the decision boundary in the privileged space, and the hyperplane in privileged space is not used for classification, there is no need for margins. However, the boundary is used to allow the

distance to margin to be measured. Like the standard SVM slack, ζ is minimised in the final term of the objective function. This is weighted by the product of C and Δ: a hyperparameter introduced in this formulation, and referred to as the parameter of approximation.

The SVM$_\Delta$+ formulation is intuitively appealing for classification. Whereas SVM+ does not take labels into account when learning in the privileged space, the SVM$_\Delta$+ does. This means that the amount of error permitted for a given instance in the primary feature space depends on how much error it required in the privileged space. Another way to put this is that the slack depends on how much of an outlier it is in its class in the privileged space, rather than how much of an outlier it is compared to the whole dataset.

In order to incorporate the constraints and solve this quadratic optimisation problem, a Lagrangian is constructed:

$$\begin{aligned}\mathcal{L}(\mathbf{w}, b, \mathbf{w}^*, b*, \mathbf{v}, \alpha, \beta) =& \frac{1}{2}||\mathbf{w}||^2 + \gamma||\mathbf{w}^*||^2 + C\sum_{i=1}^{N}[y_i(\langle \mathbf{w}^*, \mathbf{z}_i^*\rangle + b^*) + (1+\Delta)\zeta_i] \\ &- \sum_{i=1}^{N} v_i\zeta_i - \sum_{i=1}^{N}\alpha_i[y_i\langle \mathbf{w}, \mathbf{z}_i\rangle + b] - 1 + [y_i(\langle \mathbf{w}^*, \mathbf{z}_i^*\rangle + b^*) + \zeta_i]] \\ &- \sum_{i=1}^{N}\beta_i[y_i(\langle \mathbf{w}^*, \mathbf{z}_i^*\rangle + b^*) + \zeta_i]\end{aligned} \tag{2.18}$$

where $\alpha_i \geq 0, \beta_i \geq 0 v_i \geq 0, i = 1, \ldots, N$ are Lagrange multipliers.

2.1.9 Margin Transfer

Margin Transfer is an alternative maximum-margin model for LUPI, proposed by Sharmanska et al. (2014). In this model, an ordinary SVM is trained on the privileged data X^*, producing to learn privileged weights $\mathbf{w}^*$ and bias b^*. These learned parameters are then used to calculate the distance ρ_i for each sample $\mathbf{x}_i$ to the decision boundary hyperplane:

$$\rho_i := y_i\langle \mathbf{w}^*, \mathbf{x}_i^*\rangle + b^* \tag{2.19}$$

A second SVM is then trained on the original data X to produce a hyperplane with a "data-dependent margin". That is, for each example $\mathbf{x}_i$, the margin is equal to ρ_i, thereby transferring information from the privileged space; this is formulated in the constraints of the following optimisation problem, the minimisation of:

$$\begin{aligned} R(\mathbf{w}, b, \xi) =& \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq \rho_i - \xi_i \\ \text{and} \quad & \xi_i \geq 0,\ 1 \leq i \leq n \end{aligned} \tag{2.20}$$

It is apparent that "hard-to-classify" examples, which have small or negative ρ_i values, contribute little to the optimization function; the inequality $y_i((\mathbf{w}, \mathbf{x}_i) + b) \geq \rho_i - \xi_i$ is easily fulfilled due to the low ρ_i, so these examples are essentially ignored when setting the decision boundary. Conversely, the "easy-to-classify" cases have higher ρ_i, making the inequality more difficult to satisfy. In short, the decision boundary is set in a manner which only pays attention to easily classifiable examples, and this information is obtained from the privileged domain.

This optimisation problem can be solved using standard SVM packages after reparameterisation. Firstly, the constraints are divided by $\rho_i : \hat{x_i} = \frac{x_i}{\rho_i}$ and $\hat{\xi_i} = \frac{\xi_i}{\rho_i}$. The minimisation problem is then expressed:

$$\begin{aligned} R(\mathbf{w}, b, \hat{\xi}) =& \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\rho_i\hat{\xi}_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \hat{\mathbf{x}}_i\rangle + b\rangle \geq 1 - \hat{\xi}_i \\ \text{and} \quad & \xi_i \geq 0,\ 1 \leq i \leq n \end{aligned} \tag{2.21}$$

Margin Transfer therefore transfers information from $\mathcal{X}^*$ to $\mathcal{X}$ about the ease of classification for each example, as does SVM+. However, Margin Transfer explicitly uses classifier performance in $\mathcal{X}^*$ to guide training in $\mathcal{X}$, while SVM+ does not.

Margin Transfer has shown "comparable" performance to SVM+ in a series of image classification experiments. When the privileged feature set consisted of high-level image attributes, Margin Transfer utilised privileged information better than SVM+, leading to a greater improvement over baseline SVM score. However, when the privileged information consisted of extra image data, the greater improvement was achieved using SVM+. The authors suggest that this may be because SVM+ is more suitable when the original and privileged domains are the same. For the purposes of Learning using Unselected Features then, it can be expected that SVM+ is a more appropriate algorithm to handle two feature subsets which are not only the same domain, but from the same original dataset. Given this lesser suitability, and the fact that this is similar to Vapnik and Izmailov (2015)'s simplified version of SVM$_\Delta$+, this approach will not be used to implement LUFe.

2.1.10 Beyond supervised learning: GPC+

Work by Hernández-Lobato et al. (2014) expanded LUPI beyond the SVM-based classifier, building on the Gaussian Process Classifier (GPC) to produce the GPC+. This is the first Bayesian treatment of privileged information for classification, in contrast with the non-probabilistic SVM-based model seen thus far. The GPC+ uses the privileged information to model the confidence that the classifier has about a particular training example. Examples which are deemed 'easier' based on the representation in priviliged space cause a faster-increasing probit, while more difficult examples have a slower-increasing probit. The model for classification with 'privileged noise' is as follows:

$$\begin{aligned} \text{Likelihood model}: Pr(y_n = 1|\mathbf{x}_n, \tilde{f}) &= \mathbb{I}[\tilde{f}(\mathbf{x}_n) \geq 0], \quad \text{where } \mathbf{x}_n \in \mathbb{R}^d \\ \text{Assume}: \tilde{f} &= (\mathbf{x}_n) = f(\mathbf{x}_n) + \epsilon_n \\ \text{Privileged noise model}: \epsilon_n &\sim \mathcal{N}(\epsilon_n|0, z(\mathbf{x}_n^*) = exp(g(\mathbf{x}_n^*))), \text{where } \mathbf{x}_n^* \in \mathbb{R}^{d^*} \\ \text{GP prior model}: f(\mathbf{x}_n &\sim \mathcal{GP}(0, k_f(\mathbf{x}_n,)) \quad \text{and } g(\mathbf{x}_n^*) \sim \mathcal{GP}(0, k_g(\mathbf{x}_n^*, \cdot)) \end{aligned} \tag{2.22}$$

The assumption in the second line reflects the standard assumption in a GPC that there is some noise term ϵ_n added to the noise-free latent function. However, in this classifier which harnesses the priviliged information, the noise term ϵ_n now has a different variance $z(\mathbf{x}_n^*)$ for each training data point, which is drawn from a distribution that depends on the privileged information for that datapoint. GPC+ was shown to outperform standard GPC

Further work by the same authors (Sharmanska et al., 2016) expands on this Bayesian treatment of privileged information by introducing a further learning algorithm: GPC^{conf}. This takes a novel approach of allowing uncertainty in the labelling of the dataset to be modelled. In the experimentation in this paper, this uncertainty was represented in terms of the disagreement between annotators that had labelled an image dataset with whether or not a given attribute appeared in the images. The level of uncertainty was supplied as privileged information, and incorporating this allowed a boost in performance relative to the standard GPC approach in a majority of datasets.

2.1.11 Beyond classification: unsupervised LUPI

Feyereisl and Aickelin (2012) describe the first usage of privileged information in the unsupervised task of clustering, by introducing the *P-dot* algorithm, as follows. Firstly, a standard clustering algorithm such as k-means is performed twice: on the data points

in the standard space, and separately on the privileged space. This is repeated for i iterations with random initialisations, producing i clusterings. The clustering in standard space which has the highest mutual information with privileged space is selected, as it is assumed to be the most reliable. For each datapoint where there is disagreement between the labels assigned in the standard and the privileged space, a measure of reliability is calculated in both spaces: the ratio between the distance of the point to the assigned centre, and to the next-closest centre. If this confidence value is higher in the privileged than in the normal space, then the privileged cluster assignment is used instead. According to this new labelling, additional attributes are added to the dataset, with values depending on the new labels. Finally, clustering is applied to this new dataset, to produce the final solution for this method.

2.1.12 Assessing SVM+ performance

Vapnik and Vashist (2009) provide proof-of-concept results alongside their introduction to LUPI, comparing SVM+ and dSVM+ performance to that of standard SVM, in three different problem domains. The first set of problems was composed of 80 binary protein classification problems, that required the classifier to assign a label to data, indicating its protein superfamily. Primary feature set X consisted of similarity measures between amino acid sequences, and privileged feature set X^* consisted of similarity measures between 3D protein structures. Both implementations of LUPI outperformed standard SVM; SVM+ achieved the same accuracy in 18 cases, and lower error in the remaining 62 cases, while dSVM+ performed equally in 14 cases and better in 66. Comparing the two LUPI approaches directly, dSVM+ performed better in 36 cases, while SVM+ was better in just 6 cases; the remaining 38 cases were tied.

Similar results were also reported for the second problem domain: a simulated financial-forecasting task. Here, the classifiers predicted whether future steps ($t+T$) would be higher or lower than current step t in a Mackey-Glass time series model (Mukherjee et al., 1997). Here X consisted of values up to point t, and X^* contained 'future' information beyond point t (which of course would not be possible to access at model deployment time in a real-life financial forecasting task; thereby showing a use case for LUPI). Across all 12 different settings, dSVM+ achieved equal or lower error than SVM+, which in turn achieved lower error than standard SVM in all cases.

The final problem domain consisted of handwritten digit classification, and performance was consistent with previous results. The standard data X represented the images

in 100-dimensional pixel space, and X^* was composed of 21-dimensional vectors based on a 'poetic' description of each image, with each dimension measuring an attribute of the image such as 'stability' or 'tilting to the right'. Across all 6 different sizes of training datasets (40-90 instances), SVM+ achieved consistently lower error than SVM, and error by dSVM+ was lower still.

Work by Shiao and Cherkassky (2013) utilised SVM+ as an approach to handle 'right-censored' data in survival forecasting. In this partially incomplete setting, it is known that survival time for some individuals exceeds a certain value, but it is not known by how much. SVM+ with RBF kernel was used to predict survival at a given point as a binary classification, with this partially-unavailable data being supplied as privileged information for training instances. SVM+ accuracy was assessed across four different datasets, along with a novel 'pSVM' that deals with the unavailable data by allowing the notion of uncertainty in class labels. These SVM-based methods were compared with a non-SVM-based baseline: the Cox method, which is an established approach for survival-curve estimation. The SVM-based methods were more accurate in 3 of 4 datasets, with the authors noting their superiority when the survival time does not abide by classical probabilistic methods,or when there is a lot of absent information. In all cases, SVM+ accuracy score was between that of pSVM with a linear kernel, and pSVM with an RBF kernel.

LUPI has also been used to enhance performance in the domain of financial data (Ribeiro et al., 2010). SVM+ with an RBF kernel used to predict whether or not a company undergoes bankruptcy or financial distress. Standard and privileged data were taken from the 'DIANA' database, which describes attributes of French companies. The SVM+ approach outperformed standard SVM, and a multi-task learning approach, in terms of accuracy and F1-score.

2.1.13 Questioning the mechanism of LUPI

Recent work by Serra-Toro et al. (2014) suggests that the SVM+ can improve performance over standard SVM *even if the privileged information is randomly generated and not meaningful*. Two such conditions — one with linearly separable random features, the other non-separable — were compared with SVM+ performance using genuine privileged information, and with a standard SVM. In a handwritten digit-classification task, all three settings produced significant improvement ($p < 0.05$) over standard SVM, across a range of training data sizes. Performance using 'genuine' privileged data was only significantly

better than the non-separable random data in one setting, and the improvements by the LUPI settings were consistent even when size of validation set was altered.

However, if the size of the validation set is traded off for a bigger training set, SVM performance could be boosted to exceed that of the SVM+. The number of training instances was fixed at 90 for SVM+ baseline, and varied from 90-190 for SVM; when training data size was 140 or more instances, SVM performance overtook SVM+. The authors suggest that when privileged information is costly to obtain, it may be more efficient to improve computational and classification performance by providing additional training data, rather than by acquiring additional privileged features, and training the SVM+ with these. The authors state that "one possible direction for gaining insight into the role of random features with LUPI or SVM+ is by modelling the regular features, the privileged information and the class labels as random variables within the information theory" — which will be explored later in this thesis.

These findings are important within the context of this thesis, as they are the only previous work to suggest that 'Learning Using Privileged Information' may enhance model performance even without actual 'Privileged Information'. However, the novelty of this thesis lies in the fact that the secondary features — that is, unselected features — are potentially useful in themselves, and the contribution of these features, compared to the model itself, will be explored later in this thesis.

2.1.14 Comparison to other approaches

Lapin et al. (2014) compare LUPI to weighted SVM (WSVM), as two alternative means of introducing prior knowledge to an SVM binary classification task, which is also the focus of this thesis. They explain that in LUPI, the privileged features for a particular training example parameterise the upper bound on the loss function, and so are used to estimate the loss of an optimal classifier on this example. Outlying data points are permitted higher loss and therefor handled differently to other data points. This is comparable to weighted SVM, where each training instance is given a non-negative weight which controls its contribution to the loss function. Re-weighting allow the loss function to be altered to a large extent, which is similar to the effect of LUPI — "correcting features thus control the maximum influence a data point $(\mathbf{x}_i, y_i)$ can have on the resulting classifier, just like the weights in WSVMs". The authors demonstrate that this weighting can in fact be used to express the same knowledge which is encoded in the privileged information; the SVM+ dual optimisation problem can be used to construct WSVM weights that yield the same

primal solution. Therefore privileged features represent importance weighting of training instances, and SVM+ performance can be replicated using instance-dependent weighting. The authors prove that it is possible to construct weights from any SVM+ solution such that a weighted SVM will have the same solution. However the inverse is not the case. Therefore SVM+ solutions are contained within WSVM solutions. The authors state despite this, "there is no implication that any of the two algorithms is 'better"'.

The LUPI framework has also been compared with model distillation (Hinton et al., 2015) by Lopez-Paz et al. (2015), and the two approaches are explained as belonging to the same general paradigm. This will be described in more detail after introducing distillation in Chapter 3.

2.2 Feature Selection

Machine learning encompasses a variety of different learning tasks that can be performed on a given set of training data, including classification, regression, ranking, and novelty detection. Learning typically involves building a model, based on a set of N instances of training data, each of which is represented as a d-dimensional vector, where each value corresponds to one feature or attribute of that instance. As access to computational power increases, so too does the feasability of processing increasingly high-dimensional datasets. Even by 2003, in a widely-cited review of feature selection, Guyon and Elisseeff (2003) note that domains such as gene selection and text categorisation had increased the expected number of attributes to tens of thousands of features, in contrast with previous work in the field, published a few years earlier (Blum and Langley, 1997; Kohavi and John, 1997), when datasets tended to have fewer than 40 features. However, while the processing of increasingly large datasets is increasingly possible, it is not necessarily the optimal approach to train a model using all available attributes.

The dimensionality of the data (number of features) is typically reduced before learning occurs, for a number of reasons (Kohavi and John, 1997; Molina et al., 2002; Guyon and Elisseeff, 2003; Navot et al., 2005). Firstly, an excessive number of features increases the computational cost of data collection and storage. Secondly, a model of higher dimensionality often makes human interpretation and visualisation of the model more difficult. Thirdly, computational cost and runtime for the classifier will increase with extra dimensionality, both for training and deployment of a model. In addition to these benefits, performance of the model may actually improve when learned only on the selected features, due to a reduction in variance.[6], mediated by preventing fitting to noisy, irrelevant or redundant attributes in the dataset. Feature selection is therefore an important and beneficial step in many data mining and machine learning task, and different use-cases may place different importance on these different purposes.

Feature selection methods are generally grouped into three broad categories: 'filter', 'wrapper', and 'embedded' approaches; this taxonomy was popularised by Guyon and Elisseeff (2003), expanding on the work by Kohavi and John (1997) which introduced wrapper methods in contrast to filter methods. This summary of methods will concentrate on the application of feature selection to classification problems, which are the focus of this thesis.

[6]The bias-variance trade-off was previously discussed discussed in Section 2.1.4

2.2.1 Filter methods

Filter approaches evaluate feature informativeness based on correlation criteria between input data and its labels such as Pearson's Correlation (Van't Veer et al., 2002), mean differences between classes such as t-statistics (Smyth, 2004), or the generalization of those two approaches in terms of non-linear dependency measure between data and labels with Hilbert-Schmidt Independence Criterion (HSIC)(Song et al., 2012).

Variable ranking is the standard univariate approach taken in filter methods of feature selection, wherein each attribute is scored individually by some metric, which can then be used to rank the attributes and select the required top k. This approach is carried out independently of the predictor, and so can be considered a form of pre-processing. Variable ranking methods have the advantage of being scalable; they require simply the computation of d scores, and the sorting of these scores. This contrasts with the iterative and less scalable process of wrapper methods. They are also robust against overfitting through the reduction of variance but may increase bias (Guyon and Elisseeff, 2003).

Disadvantages of filter methods

One potential disadvantage of univariate ranking methods is that by considering each attribute in isolation, the feature selection process does not accurately consider how features can inform the learner *in situ*, particularly in terms of how attributes interact with one another. Guyon and Elisseeff (2003) demonstrate two ways that selecting variable *subsets* with good predictive power can be more advantageous than approaches which rank the predictive power of single variables only. Firstly, a variable which is useless by itself due to having entirely overlapping class-conditional densities is still able to improve separability in conjunction with another variable. Secondly, XOR-type 'clumping' of classes in two-dimensional problems demonstrates a further case where class separability can be achieved through the combination of individual variables, even though they individually have no discriminative power. Both of these cases demonstrate the shortcomings of ranking approaches.

However, while redundancy among variables may appear to be a further issue of filter methods, the authors also demonstrate that seemingly redundant variables can in fact produce better class separation, through noise reduction. Perfectly correlated variables are redundant, but the inclusion of very highly correlated variables may still be complementary.

Selected examples of filter methods

A range of metrics can be used to filter attributes, based on their expected informativeness about a learning task. Some examples are as follows:

1. **Chi-squared** (χ^2) This criterion measures the extent to which the label is dependent on each feature. This is usually used for categorical attributes when used for feature selection.

2. The **analysis of variance** (ANOVA) F-value between feature and label. This measures the ratio between explained and unexplained variance; in this case, the amount that variance in a label is explained by variance in the given feature of interest

3. **Mutual information** (MI) between feature and label. MI measures how much information about the label is contained by the variable by comparing their joint probability $P(\mathbf{x}, y)$ to the product of their marginal probabilities, $p(\mathbf{x})p(y)$

4. **BAHSIC** is a framework for feature selection that makes use of kernels, and the Hilbert-Schmidt independence criterion (HSIC) (Song et al., 2007). This differs from the other listed filter approaches by not being a univariate filter. Instead, this approach generalises the Hilbert-Schmidt norm, which measures dependence through summarising the co-variance matrix between two variables. As a backwards-elimination approach, BASHIC consists of iteratively choosing a kernel and subset of features to remove (the authors suggest 10%) based on maximising HSIC score with the feature set. The authors demonstrate that other widely–used criteria including Pearson's correlation coefficient are special cases of the class of algorithms defined by BAHSIC.

2.2.2 Wrapper methods

Wrapper methods assess the usefulness of features *in situ* by 'wrapping' the learning method of interest (Kohavi and John, 1997) and iteratively assessing its performance on different subsets of features. This approach treats the learner as a 'black box' which can be 'plugged in' to the wrapper method. They can be seen as a computationally expensive 'brute force' method, but searching the space of feature combinations through efficient strategies can alleviate this, and may have the additional benefit of avoiding overfitting. Greedy wrapper approaches are typically categorised as 'forward' or 'backward' feature selection processes, which respectively either start with an empty set and iteratively add

informative features, or start with a full set of features and iteratively delete the least informative ones.

Disadvantages of wrapper methods

Wrapper methods have two main disadvantages. Firstly, when number of training instances N is small, the risk of overfitting increases, as the feature set is being chosen specifically based on task performance. Secondly, this metric requires iterative training and deployment of a classifier, which is computationally expensive — particularly when the number of features d is large.

Selected examples of wrapper methods

The *de facto* example of wrapper techniques is recursive feature elimination (RFE) (Guyon et al., 2002). This approach 'wraps' a learner, and trains and tests it using subsets of the training data, before removing the attributes which correspond to the smallest learned parameters. This process is iterated until the desired number of attributes remains. In the case of classification, the SVM is commonly used as the learner, though RFE can be applied to other tasks, such as regression. The SVM parameter for an attribute controls its impact on the decision function, so removal of smaller values equates to removal of the less-informative features.

2.2.3 Embedded methods

Embedded feature selection methods differ from filter and wrapper methods in that they directly address optimisation of the objective. Typically this involves optimising a trade-off between maximising 'goodness-of-fit' and minimising the number of selected variables (Guyon and Elisseeff, 2003). Sparse regularization methods, such as those based on an L1 regularization term, are an example of embedded techniques. They are generally less computationally expensive and less prone to over-fitting than wrapper methods (Guyon and Elisseeff, 2003).

Weston et al. (2003) introduced the use of embedded methods for feature selection. They approximate the minimisation of the ℓ_0 norm of the weight vector (that is, the number of non-zero elements) by iteratively repeating the following two steps until convergence: training a regular linear SVM, and rescaling the input variables by multiplying them by the absolute value of the resulting weight vector.

Disadvantages of embedded methods

Embedded methods are not without computational challenges; this is mostly caused by the non-smooth nature of the regularization term. However, due to growing interest, efficient solvers have been proposed for logistic regression with L1 regularization (Lee et al., 2006; Koh et al., 2007), support vector machine with L1 regularization (Zhu et al., 2004), and recently using proximal algorithms (Parikh and Boyd, 2014).

For embedded methods, feature selection is performed as part of the statistical estimation procedure. Given that the model performance and the number of selected features are jointly optimised, the required number of features is not explicitly set *a priori*. The joint optimisation also means that feature selection is not performed as a distinct preliminary step. While this is not disadvantageous in itself, it means that the data is not partitioned prior to classification into primary and secondary subsets, which could be used as input to a LUPI algorithm.

2.2.4 Combining methods

It should be noted that filter methods are not necessarily univariate - they may be extended for use of subset selection. Bi et al. (2003) take the approach of using an embedded method *as a filter*. This work trains multiple sparse linear SVMs to generate linear models with good generalisation ability; these models are then used to provide a subset of variables that have non-zero weight. The linear SVM with l1-norm regularization innately performs variable selection by capacity of the SVM model, so this is exploited as a kind of variable selection, with the resulting weights used for ranking the variables. These variables are then used as the input to a final, non-linear model.

Koller et al. (1996) take an information theoretical approach to remove features which do not provide additional information beyond that contained with the remaining features. A set of features M is described as a 'Markov blanket' for a separate feature F_i if F_i is conditionally independent of everything not in M, given M. While this is computationally intractable, it can be heuristically approximated, for example by calculating pairwise KL-divergence for each pair of features, between the class probabilities conditioned jointly on the pair, and conditioned on just one feature.

2.2.5 Comparing approaches in the context of this thesis

Filter and wrapper methods of feature selection can be considered together as approaches which take a 'local view' of the feature set; they perform feature selection as a distinct

combinatorial optimisation task, prior to model learning, and this allocates a particular subset of selected variables. Only this subset is then provided as input to the model. In contrast, embedded methods do not allocate a feature set separately; they take a 'global view' of the feature set while learning, and perform feature selection as part of this procedure. The research in this thesis can be considered as a way of enhancing the 'local view' taken by filter and wrapper methods, by using a 'global view' at training time. This approach assumes a separate and preliminary feature selection process, and therefore research focuses on filter and wrapper methods, rather than embedded approaches which already take a 'global view'.

Chapter 3

Related work

The main novel contribution of this thesis concerns the usage of unselected features when training a model, in order to improve performance. The preceding chapter provided background for this work, by describing Learning Using Privileged Information, and Feature Selection, which are used in combination to implement this approach.

The purpose of this chapter is to contextualise this work by describing previous approaches to similar problems. This falls into three categories, based on similarity with the Learning using Unselected Features paradigm. Firstly, the work which took a similar *approach* of harnessing unselected features will be described in detail. Secondly, a range of prior work which tackled a similar *goal* of front-loading computation to training time will be discussed. Finally, some broader context will be provided, through the discussion of other machine learning techniques that have asymmetrical training and testing.

3.1 Using the variables that machine learning discards

The concept of utilising the features which were discarded by feature selection is relatively unexplored, with the exception of some limited investigation into this topic by Caruana and de Sa (2003). As a solitary precursor to the central problem tackled in this thesis, this previous work will be discussed in depth here.

3.1.1 Unselected features in multi-task learning

Although tackling a similar issue to this thesis, Caruana and de Sa (2003) took a considerably different approach; Multi-task learning (MTL) was used as the method for incorporating extra information into a model. MTL is a learning paradigm used to improve performance on multiple tasks, by training on the tasks in parallel, and learning multiple

input-output mappings simultaneously (Caruana, 1995, 1997). In the case of MTL in neural networks, this is effected by allowing the two sets of outputs to share a hidden layer. This forces the network to learn a shared representation, which is assumed to be a more effective and generalisable representation of the ground truth.

In this case, the primary task consisted of the standard supervised classification task: learning a mapping from the selected features as input, to the labels as output. The secondary task consisted of learning multiple *regressions*, each learning a mapping from the selected features to one of the *unselected features*. In other words, the unselected features were incorporated as a secondary output and the secondary task consisted of attempting to recreate them.

3.1.2 Comparison with LUFe

There are key differences in terms of use case and motivation between this MTL approach, and Learning using Unselected Features as described in this thesis. Caruana and de Sa (2003) examine the setting which uses unselected features that are detrimental when used as part of the primary input. LUFe instead assumes that there is some classification-relevant information in the unselected features, albeit less than the primary feature set, and attempts to utilise this. This contrast justifies the development of the novel LUFe approach; it is needed to address a different situation. The relative performance of these approaches can be expected to depend on the amount of information in the unselected features, which determines the suitability of the approach. If feature selection had produced a subset that contains nearly all of the discriminatory information from the feature set, leaving little in the unselected, then the MTL approach would be expected to out-perform LUFe. Conversely, if feature selection left some information in the unselected feature set which was not in the selected set, then LUFe would be a more effective means of harnessing this.

3.1.3 Experimentation

Caruana and de Sa (2003) tested their approach through a raft of experimentation on synthetic and real-world problems. The authors state "the results in this paper should be viewed more as a proof of concept than as the final word in this area" and indeed the findings show only a small improvement on two 'real' problems (as well as proving to be beneficial on synthetic datasets). However, these results are important within the context of this research, as they demonstrate the possibility of improving performance

using attributes which worsen performance if used as inputs.

Synthetic problems

The synthetic problems were specifically formulated in order to provide a test bed for the proposed framework and investigate whether a variable which is detrimental to performance as an input can nonetheless be beneficial as an extra output. This 'proof of concept' underlies the work on real datasets - where the additional features would be detrimental to performance if used as part of the input feature set. By extension, this can be seen as a proof of concept for LUFe; it demonstrates that through novel handling, less informative features can provide a performance boost.

In the first synthetic problem, input consisted of two random values (A and B), binned and binary-encoded - with each bit providing one input (producing a total of twenty inputs). The task was to learn the output of a sigmoid function on the sum of the two values. A secondary piece of information consisted of the output of the same sigmoid function on the difference between these two values. This experimental setting was designed such that this secondary information is not helpful to performance if it is used as additional input; the sum and difference do not correlate for random values. However, if used as a secondary *output*, as an additional task in the MTL framework, it biases the shared hidden layer to learn the sub-variables A and B — which both tasks require — thereby improving performance in the primary task.

The second synthetic problem demonstrated some extra variables are more useful as inputs when little or no noise is added to them, but they become more useful as outputs as more noise is added. The authors explain that with finite training data, the correlation between noisy inputs and the main task cause the network to learn the main task as a function of the noisy inputs, and passing the noise into the output, and using the noise-free primary input less.

Both of these cases demonstrate the feasibility of harnessing features which would worsen performance if used as input. It should also be noted that unselected features will not necessarily be detrimental to performance if used as input — for example, if there is a strict upper bound on the number of selected features which is smaller than the optimal subset size, then some beneficial features are clearly remaining in the unselected subset. Therefore, these proof-of-concept results are more than sufficient to indicate benefit in re-purposing unselected features.

Real-world problems

Following the above proof-of-concept, the MTL approach to using unselected features was applied to a DNA splice-junction problem: classifying boundaries between nucleotide sequences. The dataset contained 2000 cases, each consisting of 60 nucleotides, encoded in 3 bits, totalling 180 attributes. These features were ranked using the greedy Markov blanket approach described by Koller et al. (1996); this allowed the top 30 features to be used as input, with the next 30 features being designated the 'unselected', secondary feature set. Incorporating this subset of unselected features via the MTL approach improved performance compared to the standard setting, which only used the top 30 features as input. Interestingly, when the selected and unselected subsets were combined into a 60-dimensional input, performance actually worsened compared to the standard setting, and using all 180 features resulted in the lowest accuracy of all. These results therefore demonstrate a real-world classification problem where attributes that are detrimental as input can be beneficial if used in a secondary role.

A final real-world problem applied the MTL setting to a pneumonia risk-prediction task. This required classification of whether a 'dire' outcome was experienced for each patient, based on 192 attributes describing their symptoms and demographic profile. Attributes were again ranked, this time using the top 50 as the 'selected' feature subset, and the next 50 as top unselected features. Again, using all 192 features as input performed the worst, and concatenating the subsets into a top-100 feature input performed worse than just the top 50 subset as input. The highest performance was again achieved when the top 50 unselected were used as secondary output, with the top 50 as input. This demonstrates again that attributes which reduce performance as input, and are rightfully discarded, may still boost performance if used in a different manner.

3.1.4 Concluding remarks

Caruana and de Sa (2003) report modest improvements in synthetic and real tasks, but nonetheless provide significant motivation for work that re-purposes feature selection as a method to generate primary and secondary feature sets. They write that "the benefit of using a variable as an extra output is different from using the variable as an input"; in the cases shown in the paper, the extra information was beneficial to performance in the former case and detrimental in the latter. The results therefore show that there is potential to harness the features which are discarded by feature selection, and use them in a novel way to boost performance — even though using these features conventionally

as an input may actually worsen performance.

The authors also caution that "in order for extra outputs to be more useful than the extra inputs, there must be enough information in the regular inputs to learn the problem"; the feature selection therefore must perform well in order to achieve reasonable performance.

3.2 Other approaches to front-loading training

This thesis describes work which uses unselected features to improve perfomance and front-load computation. The previous section described work which shares the *approach* of harnessing unselected features to improve performance; this section describes other work which shares the *goal* of front-loading computation.

3.2.1 Model compression

Model compression is a machine learning technique which allows relatively inexpensive models to achieve good performance, by harnessing the power of more complex models (Buciluǎ et al., 2006). This approach therefore fulfils a similar role to the work described in this thesis: front-loading computational power; compression also shares the 'teacher/student' function nomenclature with LUPI. Compression, described by Caruana and collaborators, enables a function learned by a complex system to be learned by a simpler system; this is done by training the complex 'target' system on a dataset, and then using it to label some unlabelled data. Simpler 'mimic models' are then trained on this predictive output from the complex system (Ba and Caruana, 2014; Buciluǎ et al., 2006). Compression yields fast, compact models that mimic the performance of high-performance ensemble models, but without their prohibitive requirements at deployment time for storage space or computational power. The mimic models are able to outperform models of the same size that are trained directly on labelled training data, and in some cases, even outperform the complex target model (Buciluǎ et al., 2006). The systems discussed by Ba and Caruana (2014) are neural networks, with shallower models learning the functions from deeper models, (or ensembles of deeper models), and achieving results which previously were only attained by deep nets.

Ba and Caruana (2014) train the mimic models by minimising squared difference to the logit values (logarithms of predicted probabilities) from the target models, before softmax activation. This ensures that the mimic model places equal emphasis on learning predictions for all the target datapoints, and learns the detailed behaviour of the target function. If it were instead trained on the p-values after softmax activation, it would focus on targets with high probability, at the expense of ignoring those with probabilities which are orders of magnitude lower. Training on logits therefore prevents this information loss from passing to probability space.

Ba and Caruana (2014) propose several mechanisms through which the mimic performance can exceed that of models trained on original labels. Firstly, the complex labelling

function could act as a 'filter', both 'censoring' any errors in the original labels, and providing simplified labels for complex regions of the feature space. Secondly, the 'soft' targets provide information about uncertainty to the mimic function, which is enhanced by training on logits; this spread of uncertainty across multiple possible labels is more useful than hard 0/1 labels. Thirdly, the labelling by the teacher model is a function of only the input features, whereas the original labels could depend on other attributes not included in the input space; dependence on unavailable features is eliminated through filtering.

Compression performs successfully because the mimic approximates the function learned by the high performing model, and will not overfit if it is trained on sufficient data. Compression therefore requires large amounts of unlabeled data; if this is unavailable, Buciluǎ et al. (2006) describe a method for generating synthetic data called MUNGE. They note that while pseudo data generation is not computationally expensive, labelling it with a large, complex ensemble may be. "In the worst case, it can be more expensive to label the pseudo data and train the mimic neural net than it was to train the original ensemble library and build the ensemble model". Therefore, model compression is justified in three situations: if high-performing models are required but storage or computation is restricted, if real-time predictions are needed, or if there is a large test set.

3.2.2 Distillation

Distillation is a similar method that develops on compression by using a different and more general compression technique (Hinton et al., 2015), and similarly allows front-loading of computation. Like compression, the goal of this procedure is to teach the smaller model to generalise in the same way that the large model does. The key difference between these approaches is in how this similarity is enforced. While compression is trained using the logits, distillation is trained directly on the probabilities which are produced by the softmax. The softmax output layer computes a probability q_c for a given class c from the logit z_c as follows:

$$q_c = \frac{exp(z_c/T)}{\sum_d \exp(z_d/T)} \tag{3.1}$$

In the above formulation, T is a 'temperature' parameter, usually set at 1. Hinton et al. (2015) use a larger T value in order to 'raise the temperature' of the softmax function and force 'soft targets'. This serves the same purpose as training on logits, and means that valuable information about classes which are *not* chosen is not lost. The authors give the example of digit classification; the relative class probabilities for a 2 that resembles a 3, are different from those of a 2 that resembles a 7 — encodes some information

about generalisation. Raising the temperature and enforcing soft targets means that these probabilities are not vanishingly small, so still contribute to the error function; therefore this information is not lost.

If the correct labels for all or some of the transfer set are known, distillation is also able to take these into account, in order to further improve performance. This can be achieved by taking a weighted average of two objective functions: one of which is based on the cross-entropy with soft targets, and other is based on the cross-entropy with the correct labels.

Initial experimentation on the MNIST digit classification dataset proved effective, with 67 - 74 errors, depending on regularisation strategy. Better results were reported by down-weighting the impact of the true labels, which demonstrates the amount of knowledge hidden in the softmax output. The T value is a hyperparameter and the authors report best results in the range 2.5-8, depending on size of the neural net.

3.2.3 Unifying distillation and privileged information

The similarities between LUPI and distillation are made explicit in work by Lopez-Paz et al. (2015). The authors unify the two concepts into a framework called 'generalised distillation' — which they describe, in turn, as an example of the 'machines-teaching-machines' paradigm.

Generalised distillation consists of three stages: first, teacher function $\hat{f}_t \in \mathcal{F}_t$ is learned from input output pairs $\{(\mathbf{x}_i^*, y_i)_{i=1}^n\}$ using the equation:

$$\hat{f}_t = \underset{f_t \in \mathcal{F}_t}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \sigma(f_t(\mathbf{x}_i))) + \Omega(||f||) \tag{3.2}$$

$\mathcal{F}_t$ is a class of functions that map from $\mathbb{R}^d$ to $\mathbb{R}^c$ where c is the number of classes, and label y is also a c-dimensional vector. Therefore this minimisation finds a function to minimise error on the class probabilties.

Then, 'soft labels' s from the teacher function are calculated using the softmax function σ and the positive-valued 'temperature' parameter T: $\{\sigma(\hat{f}_t(\mathbf{x}_i^*)/T)\}_{i=1}^n$

Finally, the student function $\hat{f}_s \in F_s$ learns from two sets of input-output pairs: $(\mathbf{x}_i, y_i)_{i=1}^n$ and $(\mathbf{x}_i, s_i)_{i=1}^n$, using the equation

$$\begin{aligned} \hat{f}_s =& \underset{f_s \in \mathcal{F}_s}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} [(1-\lambda)\ell(y_i, \sigma(f_s(\mathbf{x}_i))) + \lambda \ell(s_i, \sigma(f_s(\mathbf{x}_i)))] \\ \text{where} \quad s_i =& \sigma(f_t(\mathbf{x}_i)/T) \in \Delta^c \end{aligned} \tag{3.3}$$

This essentially trades off two loss functions: loss between the student function and the labels, and loss between the softmax from student function with the softmax from the teacher; 'imitation parameter' $\lambda \in [0, 1]$ balances this trade-off.

Here $\mathbf{x}^*$ could be privileged information. However if $\mathbf{x}_i^* = \mathbf{x}_i \ \forall i$, and if the capacity of the student function space $\mathcal{F}_s$ is much less than the capacity of the teacher $\mathcal{F}_t$, this reduces to distillation described earlier. Conversely, this reduces to generalised LUPI if $\mathbf{x}_i$ is a privileged description of $\mathbf{x}$ and the student function space has much more capacity. The authors emphasise this difference: $\mathcal{F}_t$ enables a general-purpose, flexible representation of the training data in distillation, but this representation is a more simple and specialised space in LUPI.

3.3 Other approaches with different information at train and test times

The asymmetrical training and testing domains seen in LUPI, compression and distillation are not without precedent in Machine Learning research. This section will discuss various other settings that involve different feature spaces and/or probability distributions between the training and testing domains: *domain adaptation*, *transfer learning* and *multi-view learning*. The techniques discussed in this section are not directly applicable to the case of harnessing unselected features at train time. However, these approaches provide some context for the development of techniques for the LUFe setting. They are summarised, and compared with LUPI and classical machine learning, in Table 3.1.

Setting	Learning domain	Target domain	Source labels	Target labels
Classical	X	X	$\mathbf{y}$	$\mathbf{y}$
LUPI	X and $X*$	X	$\mathbf{y}$	$\mathbf{y}$
Domain Adaptation	X	$X*$	$\mathbf{y}$	$\mathbf{y}$
Transfer Learning	X	$X*$	$\mathbf{y}$	$\mathbf{y*}$
Multi-view learning	X and $X*$	X and $X*$	$\mathbf{y}$ and $\mathbf{y*}$	$\mathbf{y}$ and $\mathbf{y*}$

Table 3.1: **Characteristics of different machine learning settings** In each case, X refers to some basic feature space and X* is a feature space different to X. Similarly, Y and Y* are two different label spaces

3.3.1 Domain adaptation

Domain adaptation (DA) refers to the case where a classifier is trained on data from a 'source' domain with plentiful labelled data, before being applied to data from a different, 'target' domain with little or no labelled target data. The source and target domains have the same feature space, but different marginal distributions. A comprehensive review of DA by Jiang (2008) divides the field into two categories: *unsupervised* domain adaptation, where labelled data is available only in the source domain[1], and *supervised* domain adaptation where there is labelled data in the source and target domains.

Domain adaptation and LUPI have some similarities. Both settings involve performing the same classification task (assigning labels from the same categories) in training and testing, but on different data. However, in DA the features are the same but the distribution is not; conversely, LUPI assumes the same distribution over normal features — the difference lies in the additional privileged features.

A number of simple ways to address a supervised domain adaptation situation are listed by Daume III (2007): simply using only the source or target data, using both sets (optionally giving greater weight given to target data), using the label from the source-trained classifier as an additional feature, and linearly interpolating the predictions of classifiers trained on each domain separately. Of these methods, the only one which could be applied in an unsupervised scenario is to train on source data only. Daume III describes these baseline approaches as 'surprisingly difficult to beat'.

The 'frustratingly easy' approach to supervised DA taken by Daume III (2007) is to create three versions of each feature: one general, one source-specific, and one target-specific. From original feature space $X = \mathbb{R}^F$, this produces the new space $\breve{X} = \mathbb{R}^{3F}$. Source and target instances are mapped to $\breve{X}$ according to ϕ^s and ϕ^t as follows, where $\mathbf{0} = \langle 0, 0, ..., 0 \rangle \in \mathbb{R}^F$:

$$\phi^s(x) = \langle \mathbf{x}, \mathbf{x}, \mathbf{0} \rangle, \phi^t(x) = \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle$$

In this way, the learning algorithm can decide which features are shared between domains, and encode the level of consistency via the weight vector; higher weights are assigned to the appropriate version of a feature, depending on whether it is domain-invariant, or different across the domains. This approach can also be generalised to more than K domains by using $K + 1$ versions of a feature.

Another approach is called *instance weighting* (Jiang, 2008). Supervised learning aims

[1] This setting is also occasionally referred to as *semi-supervised* domain adaptation (Daume III, 2007)

to minimise expected loss on the target data; given that target labels are unknown in unsupervised DA, this performance is approximated with a loss function across source domain data. It is therefore possible to individually weight the training data instances, such that the loss function places greater importance on correctly classifying those which are likely to occur in the target domain. This allows the classifier to be fit according to those source domain instances which resemble the target domain, with less consideration given to differences in distributions.

Blitzer et al. (2006) introduce Structural Correspondence Learning (SCL), which assumes the existence of some 'pivot' features that behave similarly in both the source and target domains. In SCL, these pivot features occur frequently but also must be sufficiently diverse to capture the nuances of the data distribution. A binary classification task is then created for each pivot; all the non-pivot features are used as input to a classifier, which predicts whether or not the pivot in question is contained in each instance. The resulting weight vectors which are trained in each binary classifier encode the covariance of the non-pivot features with the pivots. Singular Value Decomposition is then applied to the weight vector, which produces θ: a projection from the original feature space to $\mathbb{R}^h$, which functions as a shared representation for features of both domains. θ is used to augment the original feature array so that each training instance then contains original features $\mathbf{x}_t$ and shared features $\theta\mathbf{x}_t$; this augmented feature array is used as input to the classifier. The selection of pivots depends on the task; for part-of-speech tagging it was possible to use frequently-occurring words, which tended to correspond to prepositions and determiners, which are good indicators of part-of-speech. However, for sentiment classification, a further stipulation of high mutual information with the source label was added.

SCL has been extended to the semi-supervised case (Blitzer et al., 2007), exploiting a small amount of labelled target data to learn to ignore features which are misaligned between domains. Firstly, labels for the small target dataset are obtained using SCL (trained on the source domain). Then, these labels are used to augment the target dataset, and SCL is applied to the labelled target set with an extra constraint, to minimise the distance between the new weights and those learned on the source data. This technique led to improvement over a baseline which used the label from source domain as a feature, but no SCL features. This was despite using only 50 labelled target instances, compared with 1600 training source instances, and 400 testing target instances.

Pan et al. (2010) propose Spectral Feature Alignment (SFA), which similarly to SCL, induces correspondence among features from different domains, by exploiting their relation

to a subset of 'domain independent' features. The remainder of features are domain-specific, but can then be modelled in terms of co-occurrence with the domain-specific features. In effect, this maps to a common latent space for domain-independent features, in which similar features from different domains are aligned due to their similar patterns of co-occurrence. The full original feature set is augmented with the new feature representation and this augmented feature set is used to train a classifier.

Another approach to domain adaptation involves taking subsets of the data in order to minimise the difference between training and test distributions ($\mathcal{D}$ and $\mathcal{D}'$) (Satpal and Sarawagi, 2007). The distributions are compared in terms of their component-wise means.

3.3.2 Transfer learning

Transfer learning refers to the situation where knowledge is transferred from one learning task to another, where there are different feature spaces or data distributions between the two tasks (Pan and Yang, 2010). This fulfils a similar purpose to domain adaptation: utilising related data and models, to compensate for limited data available for the target domain. The difference is that while DA involves a single task with training in a source domain and application in a target domain, TL has a learning task in the source domain which is used to help learn a second task in the target domain. A different domain may involve a different feature space, or a similar feature space with different marginal probabilities. A different task could be assigning different labels, or different conditional probabilities of labels given certain features.

Pan and Yang (2010) outline four types of information that can be transferred between domains: *instance transfer*, where source domain data is re-weighted for use in the target domain, *feature representation transfer*, where divergence between the feature representations is minimised, *parameter transfer*, which discovers parameters or priors shared between the domains, and *relational knowledge transfer*, which maps knowledge between domains about relations between non-iid data points.

TrAdABoost is an algorithm developed from AdABoost for use on the transfer learning setting, following an instance transfer (Dai et al., 2007b) approach. TraAdaBoost assumes identical features and labels but different data distributions between the domains, and iteratively re-weights the source domain data to better enable learning in the target domain. A similar aim underlies the heuristic approach taken by Jiang and Zhai (2007). This method removes training examples which are less representative of the target domain, based on the difference in conditional probabilities $P(y_T|\mathbf{x}_T)$ and $P(y_S|\mathbf{x}_S)$.

An alternative approach to parameter transfer is feature representation transfer; these methods learn a low-dimensional data representation shared between the source and target tasks, (Argyriou et al., 2007). The algorithm alternates between an unsupervised step where the shared representation is learned, and a supervised step, where task-specific functions are learned using this representation.

This approach is used in *self-taught clustering* problems (Dai et al., 2007a), a type of unsupervised transfer learning situation, where there is no labelled data in either domain. However, it is still possible to transfer information to improve clustering in the target space; this is accomplished by finding a shared feature space between the source and target domains. The amount of information lost by the clustering should be minimised, so self-taught clustering involves solving the following optimisation problem, consisting of minimising the loss function:

$$\underset{\tilde{\mathrm{X}}_T,\tilde{\mathrm{X}}_S,Z}{argmin}(I(X_T,Z)-I(\tilde{\mathrm{X}}_T,\tilde{\mathrm{Z}})+\lambda[I(X_S,Z)-I(\tilde{\mathrm{X}}_S,\tilde{\mathrm{Z}})])$$

where $\tilde{\mathrm{X}}_S$ and $\tilde{\mathrm{X}}_T$ are the clusterings in the source and target domain respectively, I denotes mutual information, and Z is a shared feature space between X_S and X_T.

Self-taught learning (STL) is a paradigm proposed by Raina et al. (2007) and can be seen as a type of transfer learning with feature representation transfer. Firstly, STL uses the unlabelled out-of-class data to learn a sparse, higher-level representation; unlabelled data instances are represented as the product of basis vectors (shared between all instances) and the 'activation vector' (unique to that instance; each element assigns weight to a corresponding basis vector). The following optimisation problem is formulated to obtain a and b, and is convex over a and b separately:

$$\underset{b,a}{minimize}\sum_i ||\mathbf{x}_u^{(i)}-\sum_j a_j^{(i)}b_j||_2^2+\beta||a^{(i)}||_1$$

The first term minimises reconstruction error, while the second enforces sparseness in a. Given the resulting fixed basis vectors b, a new feature representation $\hat{a}(\mathbf{x}_l^{(i)})$ is then computed for each labelled instance $\mathbf{x}_l^{(i)}$:

$$\hat{a}(\mathbf{x}_l^{(i)})=\underset{a^{(i)}}{argmin}||x_l^{(i)}-\sum_j a_j^{(i)}b_j||_2^2+\beta||a^{(i)}||_1$$

The resulting new representation $\hat{T}=\{(\hat{a}(\mathbf{x}_l^{(i)}),y_{i=1}^{(i)})\}_{i=1}^m$ is then passed to a supervised learning algorithm.

STL avoids two limitations of PCA: the new representation ($\hat{a}$) does not need to be a linear function of $\mathbf{x}$, and its dimensionality can exceed that of $\mathbf{x}$. STL is a significantly

different setting from LUPI, even though both leverage extra data at train time. The additional data in STL does not correspond to the same instances, it is unlabelled, and does not even need to belong to the categories from the classification task. The way it is employed is also different; the extra data is used to change the feature representation.

3.3.3 Multi-view learning

Multi-view learning refers to the situation where there are multiple available datasets relating to a single classification task. These datasets may result from different sources or feature subsets. LUPI can almost be considered a form of multi-view learning; the privileged information is, in effect, a second view for the training data. The difference which makes LUPI a distinct field is that the second view is available for testing data in multi-view learning, but not in LUPI.

Simple concatenation of multiple views into a single dataset causes overfitting, particularly on small training sets; therefore specific strategies have been developed for the multi-view case. There are three groups of methods for dealing with this situation (Xu et al., 2013): *co-training*, which train alternately between two views and maximise their mutual agreement, *multiple kernel learning*, which combines multiple kernels that correspond to different views, and *subspace learning* which attempts to find the common latent subspace between views. Xu et al. (2013) describe two principles which underlie all three methods: the *consensus principle* states that the probability of disagreement between two hypotheses (each corresponding to one view) $\geq$ the error rate of either view. This inequality shows that lowering the disagreement rate will also lower the error rate. The *complementary principle* states that multiple views can accurately describe the data because each may contain knowledge that the others do not. These principles may also be applicable to LUFe; the effect of agreement or disagreement between selected and unselected feature sets will be explored later in this thesis.

The first co-training algorithm was proposed for semi-supervised learning, where there is both labelled and unlabelled data (Blum and Mitchell, 1998). Two classifiers are trained on labelled data L from views $\mathbf{x}_1$ and $\mathbf{x}_2$ separately before being applied to 'pool' U', a subset of unlabelled data U. The most confidently classified p positive and n negative examples are then added to L. U' is replenished from U and the process is repeated iteratively. Co-EM expands on this by running expectation maximisation in each view, to assign probabilistic labels which can change each iteration (Nigam and Ghani, 2000). One limitation of Co-EM is that it requires naive Bayes as the underlying classifier. Co-

regression is a co-training style algorithm for semi-supervised regression, that uses two k-nearest neighbour regressors (Zhou and Li, 2005). The regressors label an unlabeled example for each other, according to labeling confidence, estimated according to the consistency of the regressor with the labeled example set. This is repeated iteratively, until the final prediction is made by averaging over the classifiers.

Chapter 4

Learning using Unselected Features

4.1 Overview

The contribution of this thesis is to introduce an approach called Learning using Unselected Features (LUFe), which can improve classifier performance after feature selection has been employed. In a less expressive model with limited dimensionality, error due to classifier bias may increase. LUFe can improve performance by reducing this error due to bias. This is achieved by allowing different features to serve different functions in classification, with feature selection used to assign these roles. In this framework, the most informative features are used directly in setting the decision boundary, while the less informative features play an indirect role in guiding the learning process. The unselected features help by constraining the feasible set which is searched for the decision boundary.

LUFe draws on the work on LUPI and feature selection that is summarised in Chapter 2, and combines these practices in a novel manner. To recap: LUPI allows additional information about training data to be harnessed by a learning system; feature selection is the process of picking a subset of available attributes from a dataset in order to build a predictive model, and the remaining features are typically discarded and not used further. In the LUFe approach, feature selection is performed, but the unselected features are instead used as a secondary input to the classifier, at training time only. This is done by re-purposing the LUPI framework; the selected features are used as the 'standard' feature set and the unselected features are used as the secondary, 'privileged' input; this is summarised in Figure 4.1.

This deviates from almost all feature selection procedure by not 'throwing away' the

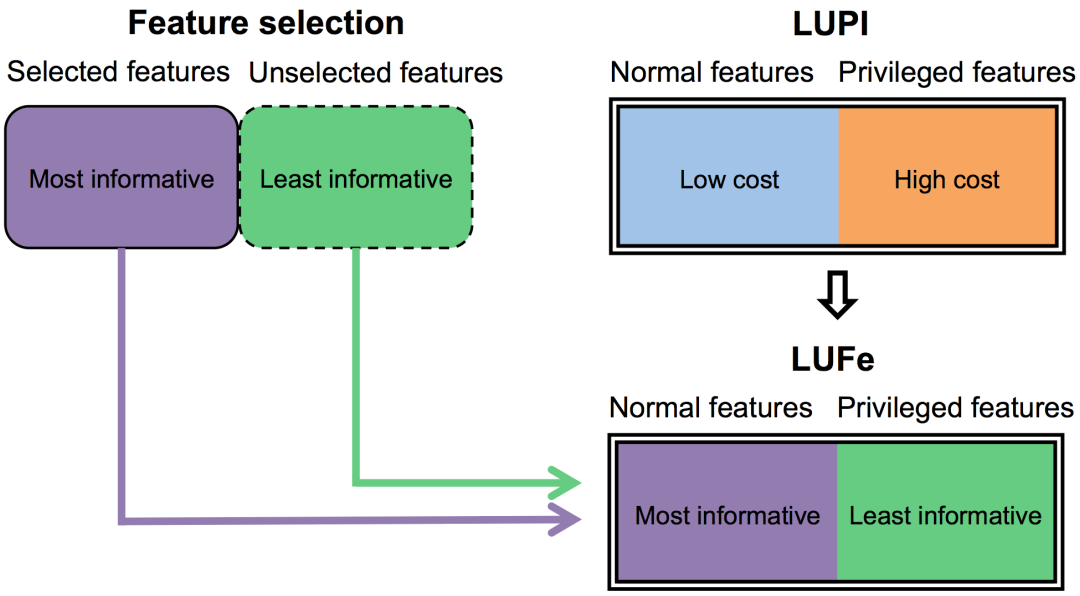

Figure 4.1: LUFe combines feature selection with the LUPI framework. Instead of being used to decide which features are used by the model, and which are discarded, feature selection is repurposed to determine two feature subsets to be used as inputs to the LUPI framework.

unselected features. This also differs from the typical LUPI paradigm in three ways: Rather than using some informative, but high-cost, privileged features, LUFe uses those which were designated as less-informative. Instead of using a secondary feature set which is only available at training, LUFe "chooses" to use a subset of readily available features at training time only, so they do not need to be collected for test data. Rather than being of a different modality, the secondary feature set in LUFe is from the same modality as the primary feature set.

4.2 Motivation

The current technological landscape involves an abundance of data and a growing number of 'smart' devices that can collect and process it. The Internet of Things (IoT) is the paradigm of pervasively present objects with the ability to sense, actuate and interact with each other (Atzori et al., 2010). Although the realisation of this concept is already underway, Gubbi et al. (2013) state that for IoT to fully emerge,"the computing paradigm will need to go beyond traditional mobile computing scenarios that use smart phones and portables, and evolve into connecting everyday existing objects and embedding intelligence into our environment". The same authors note the necessity of storing and using data

intelligently for smart monitoring and actuation, and the requirement to "develop artificial intelligence algorithms which could be centralised or distributed based on the need". This suggests some processing beyond simply acquiring raw sensor data is required on the connected 'things'.

We can consider sensing to involve not just detecting a raw signal, but also to require some initial processing by the sensing device — for example a signal classification to decide whether a detected event is genuine, or spurious. Consider now the situation where such a classification model is being deployed in a device with limited resources for computational power and data storage. These constraints negatively limit the complexity of a model, and so potentially impair its performance.

Feature selection is one widely-used method to reduce model complexity, by deciding which variables are most informative about a task, and then using only these (reviewed in Section 2.2). If the unselected features are instead used only at training time, the benefits of reduced complexity at deployment can still be enjoyed, while still exploiting the greater resources for more expensive model training. The computational load of using a model would be front-loaded to training time, while still minimising the deployment time cost. This is the intuition behind the Learning using Unselected Features approach which is introduced here. It may seem counter-intuitive to use the features designated as less informative to improve performance. However, there is some precedent for this line of research — namely, that by Caruana and de Sa (2003) discussed in 3.1.

4.2.1 Use cases

Now that the principles of the LUFe approach have been described, let us consider the domains which provide a use case for this paradigm. These areas have certain attributes in common. Firstly, the domain must have limited resources on the deployment system, but more resources where the model is being trained. The former must be true for LUFe to be used in place of the entire feature set, and the latter is necessary to provide a means of harnessing the additional features. Secondly, the domain must have high data velocity, and a need for rapid processing, to necessitate the limited model at deployment time which is seen in LUFe.

IoT devices fulfil these criteria as they are are able to access trained models online, but typically have limited resources of their own. Mahdavinejad et al. (2018) define the architecture of "edge computing" for IoT where "processing is run at a distance from the core, toward the edge of the network." This allows data to intially processed at the edge

of the network — on IoT devices — and the authors describe benefits including early filtering and cleaning of data, local data storage for local use, and enhanced security.

Mahdavinejad et al. (2018) describe data processing at the edge of IoT networks in a range of domains such as Smart Traffic, Smart Health, Smart Weather Prediction and Smart Agriculture. The following hypothetical examples demonstrate how edge computing situations can benefit from the LUFe paradigm:

- **Scenario:** Object identification by on-board sensor on autonomous robot with limited computational power

  Data: Sensor array data from robot's surroundings, from multiple directions, with high data velocity

  Application of LUFe: Trained using all sensor array data; deployed using only most informative features; e.g. specific pixels within field, or specific features (edges, corners, ridges) extracted after initial processing

- **Scenario:** Smart kitchen device which activates in response to a voice command

  Data: Constantly monitored audio with a buffer lasting a few seconds which is monitored for activation command and subsequent other voice commands

  Application of LUFe: : Trained using a large feature set obtained after analogue-to-digital conversion of training audio. Deployed using only the most informative of these features.

- **Scenario:** Wearable health monitor for early detection of adverse health outcomes

  Data: A range of biometric attributes such as heart rate, breathing rate, skin conductivity, body temperature, and accelerometers and gyroscopes, monitored at high frequency intervals

  Application of LUFe: Trained using a large set of all features from all available modalities; deployed using only the most informative features, processed onboard the wearable device.

Each of these examples demonstrates a use-case with high volume and velocity of data, but limited processing power — showcasing a need for the LUFe framework. Now that we have seen a general explanation of LUFe, and motivating examples for its application, the paradigm will be formally described. The following sections reconsider feature selection to provide context, before formally introducing the LUFe paradigm.

4.2.2 Feature selection revisited

In supervised learning, we are given input $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and output $Y = (y_1, \ldots, y_N)$. The data points form pairs $(X, Y) = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \subset \mathcal{X} \times \mathcal{Y}$ and we attempt to learn a mapping function $f : \mathcal{X} \to \mathcal{Y}$. In a binary classification task, $\mathcal{Y} = \{+1, -1\}$ and $\mathcal{X} = \mathbb{R}^d$ where d is the dimensionality of the input feature space. We seek to infer a latent binary classification function $f : \mathcal{X} \to \{+1, -1\}$ that is taken from a particular function space $\mathcal{F}$ and will attach a label y_{new} for a new input $\mathbf{x}_{\text{new}}$.[1]

Feature selection is often utilised to decrease data dimensionality in order to reduce storage and computational requirements at deployment time. Let $\mathcal{T}$ be the set of features for dataset X, so $|\mathcal{T}| = d$. Let k be an upper bound on the number of features to be selected, which is set based on the requirements or limitations of the system. Feature selection techniques then attempt to solve the combinatorial problem of selecting an optimal subset of features $\mathcal{S} \subseteq \mathcal{T}$ which contains the maximum amount of information from X without redundancy, and the number of selected features does not exceed k. Let $\mathcal{Q}(\mathcal{S})$ be a functional measuring the informativeness of a feature subset, computed by restricting X to have only the features in $\mathcal{S}$. Feature selection can then be formulated as the following optimisation:

$$\widehat{\mathcal{S}} = \arg\max_{\mathcal{S} \subseteq \mathcal{T}} \mathcal{Q}(\mathcal{S}) \tag{4.1a}$$

$$\text{subject to } |\mathcal{S}| \leq k, \tag{4.1b}$$

The corresponding unselected feature subset is $\widehat{\mathcal{U}} := \mathcal{T} \setminus \widehat{\mathcal{S}}$. Finding a global solution to the formulation in (4.1) is generally an NP-hard problem (Weston et al., 2003). In practice, a greedy approach such as forward feature selection, backward feature selection, or a mixed approach is adopted (Guyon et al., 2002). The feature selection algorithms discussed in 2.2 each explicitly or implicitly have some way of approximating the optimisation problem (4.1) by defining the criterion $\mathcal{Q}(\cdot)$. Standard feature selection methods only use $\widehat{\mathcal{S}}$ in the subsequent regularized risk minimization and discard $\widehat{\mathcal{U}}$ without use. With the exception of Caruana and de Sa (2003), who employed the secondary feature set as an additional output in a multi-task learning problem as described in Section 3.1, this practice has not been disputed.

Feature selection is a non-trivial problem; one way to consider this concerns bias-variance trade-off. Using filter and wrapper feature selection methods involves a trade-off

[1] This work focuses on classification as the primary example, but the same principles can be generalised to perform regression by applying a real-valued output instead of a label.

between the number of features used ($|\widehat{\mathcal{S}}|$) and the regularised training error for this subset. Fitting a model to fewer features (and discarding the rest) reduces the size of the the function class $\mathcal{F}$. This reduces the chance of overfitting the data and thus reduces the model's error due to variance, by requiring fewer parameters to be learned. In this way, feature selection has the benefits of being potentially more accurate, as well as more computationally cost-effective. However, this lower variance is typically gained at the expense of higher error due to bias. This bias-variance trade off is a fundamental problem in machine learning and determines the generalisability of the classifier. The two types of error have to be balanced when training, such that f fits the training data while still being generalisable to unseen instances.

Another way to consider the challenge of feature selection is in terms of the amount of information being taken into account. Filter and wrapper methods of feature selection can be summarised as *combinatorial* methods, because they involve the binary decision to either include or exclude each feature, whereas embedded methods are *continuous*, and involve varying the non-discrete weight of each feature. Combinatorial methods produce a 'local view', where only the selected subset is used to build the classifier, whereas continuous methods produce a 'global view', where the predictive model can access all the features. These two different approaches to feature selection have different benefits and drawbacks. Combinatorial methods have the advantage of being less computationally expensive at test time, as they only consider the selected features. However, this local view means that once a feature is unselected, it will not be considered at all in the subsequent classification. The feature selection process which designates $\widehat{\mathcal{S}}$ makes this decision based on training data and may not generalise to new datapoints when the model is deployed. The subset $\widehat{\mathcal{S}}$ chosen by maximising a given choice of criterion $\mathcal{Q}(\cdot)$ will likely be non-identical to that chosen by a different choice of $\mathcal{Q}(\cdot)$, so the set of selected features is dependent on the feature selection method used. Conversely, although the global view of continuous methods takes the entire feature space into account, it has the disadvantage of higher computational cost. The proposal of Learning using Unselected Features explores the space between combinatorial and continuous approaches. Specifically, this enhances combinatorial feature selection, in order to gain the advantage of a global view, while maintaining lower complexity at test time. This method therefore combines the respective benefits of both combinatorial and continuous feature selection.

4.2.3 Rethinking feature selection

Given that (a) the less complex model produced by feature selection can have more error due to bias, and (b) unselected features are usually discarded, this work questions whether the unselected features can instead be used to reduce bias in the model. This is attempted by taking both selected and unselected features into account in different roles during the regularized risk minimization. Feature selection is re-purposed, to split the dataset into the primary and secondary feature subsets, rather than one subset which is used, and one which is discarded. LUFe can maintain the lower-variance advantage of feature selection, while potentially also limiting the amount of bias. The model still only learns parameters for the selected features, but uses the extra features at training time to constrain the space which is searched for these parameters.

In this situation where feature selection desirably reduces computational and storage costs at deployment time, the unselected features cannot be used as a direct input to the latent function f. Instead, they are used to bias the learning process of a classifier — which is fit to the primary feature set — towards better generalization performance.

The Learning using Unselected Features (LUFe) method achieves this by using the feature criterion for unselected features ($\mathcal{Q}(\widehat{\mathcal{U}})$) to define a data-dependent upper bound on the classifier's loss function in the primary feature space. In effect, the unselected features will constrain the feasible set which is searched for the decision boundary in the primary feature space.

For each data point $\mathbf{x}_i$, the data-dependent upper bound on the classifier's loss incurred when using selected features $\mathbf{x}_i^{\widehat{\mathcal{S}}}$ is defined as $\mathcal{Q}^i(\widehat{\mathcal{U}}) = \left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathcal{Q}(\widehat{\mathcal{U}}) \right\rangle$, where $\mathbf{x}_i^{\widehat{\mathcal{U}}}$ is a data point $\mathbf{x}_i$ restricted to only unselected features and $\mathcal{Q}(\widehat{\mathcal{U}})$ denotes an array of computed feature criterion on all singletons of $\widehat{\mathcal{U}}$. The assumption is made that feature criterion $\mathcal{Q}(\cdot)$ is non-negative. This is fulfilled in almost all criteria of feature selection methods, such as, RFE (Guyon et al., 2002), HSIC (Song et al., 2012), mutual information (Lefakis and Fleuret, 2014), ANOVA and Chi-squared.

For a linear classification function $f(\mathbf{x}) := \langle \mathbf{w}, \mathbf{x} \rangle + b$, the LUFe optimization is:

$$\underset{\mathbf{w},b}{\text{minimize}} \ \|\mathbf{w}\|_{\ell_2}^2 \tag{4.2a}$$

$$\text{subject to,} \quad \forall\, i = 1, \ldots, N,$$

$$\underbrace{1 - y_i[\langle \mathbf{w}, \mathbf{x}_i^{\widehat{\mathcal{S}}} \rangle + b]}_{\substack{\text{classifier's loss} \\ \text{based on selected features}}} \leq \underbrace{\left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathcal{Q}(\widehat{\mathcal{U}}) \right\rangle}_{\substack{\text{data-dependent upper bound} \\ \text{based on unselected features}}}. \tag{4.2b}$$

This resembles the definition of a hard-margin SVM described in section 2.1, but with a key difference: the constraint is altered such that the cost incurred by each datapoint $\mathbf{x}_i^{\widehat{\mathcal{S}}}$ is no longer bounded by 0 and instead is bounded by the term $\left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathcal{Q}(\widehat{\mathcal{U}}) \right\rangle$. This is therefore a data-dependent margin for data point $\mathbf{x}_i^{\widehat{\mathcal{S}}}$, based on corresponding unselected features $\mathbf{x}_i^{\widehat{\mathcal{U}}}$.

Let us consider the intuition behind this formulation, using $\mathcal{Q}^i(\widehat{\mathcal{U}})$ as shorthand for the upper bound $\left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathcal{Q}(\widehat{\mathcal{U}}) \right\rangle$ on the i-th data point. If the feature selection algorithm has done a good job of selecting the subset $\mathcal{S}$ then the informativeness of unselected features $\mathcal{U}$ is relatively low, and relatively high for selected features $\mathcal{S}$. In this case, the classifier can be expected to perform well for this instance and this is reflected by a small data-dependent upper bound $\mathcal{Q}^i(\widehat{\mathcal{U}})$ on the classifier's loss in (4.2b). Conversely, if the informativeness of selected features is not much higher than that of unselected features, the classifier can be expected to perform less well, and the loss for this datapoint is given a larger upper bound $\mathcal{Q}^i(\widehat{\mathcal{U}})$.

This formulation of the LUFe concept uses an array of computed feature criterion values on all singletons $\mathcal{Q}(\widehat{\mathcal{U}})$; this can then be replaced with a second weight vector $\mathbf{w}^\star$, that is $\mathbf{w}^\star := \mathcal{Q}(\widehat{\mathcal{U}})$. The weight vector $\mathbf{w}^*$ is learned from the data, providing the following formulation of LUFe:

$$\underset{\mathbf{w},b,\mathbf{w}^\star}{\text{minimize}} \ \|\mathbf{w}\|_{\ell_2}^2 + \lambda_1 \|\mathbf{w}^\star\|_{\ell_2}^2 + \lambda_2 \sum_{i=1}^{N} \left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathbf{w}^\star \right\rangle \tag{4.3a}$$

$$\text{subject to, for all } i = 1, \ldots, N,$$

$$1 - y_i[\langle \mathbf{w}, \mathbf{x}_i^{\widehat{\mathcal{S}}} \rangle + b] \leq \left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathbf{w}^\star \right\rangle \tag{4.3b}$$

$$\left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathbf{w}^\star \right\rangle \geq 0. \tag{4.3c}$$

The weight vector $\mathbf{w}$ is still a k-dimensional vector, corresponding to each dimension in the primary feature space of selected features $\mathcal{S}$; $\mathbf{w}^\star$ is a $(d-k)$-dimensional vector, corresponding to the unselected features $\mathcal{U}$. λ_1 and λ_2 are scalar trade-off parameters. The second and third terms of the objective function (4.3a) are for regularisation to ensure that the criterion values do not grown unreasonably large. The first constraint again ensures that the error is upper bounded by the criterion on unselected features, and the second constraint ensures this is non-negative. This approach of Learning using Unselected Features (LUFe) is summarised in Algorithm 1.

The optimization problem in (4.3) can be solved in the dual representation using a standard quadratic programming (QP) solver. The Lagrangian dual form of a constrained

optimisation problem is a second, closely-related problem in which the constraints are incorporated directly into the optimisation. Each constrant has a coefficient or 'Lagrange multiplier', α. In this case where the primal form includes a constraint corresponding to each of the N data points, the Lagrangian form includes N new terms, each of which has a new variable α_i. This is easier to solve in the case where the data dimensionality significantly exceeds the number of instances, as the optimisation is now over the N Lagrangian multipliers. The dual form is also beneficial to use when employing the kernel trick (described in 2.1.4), as it removes the need to explicitly compute the mapping of each data point, as is required in the primal form.

4.2.4 Comparison with LUPI and conventional feature selection

It is now apparent that this formulation borrows and re-purposes the SVM+ algorithm for LUPI as explained in section 2.1. However, the LUFe approach deviates from the standard LUPI paradigm in a number of ways. Firstly, the SVM+ as described in Vapnik and Vashist (2009) uses a secondary feature space $\mathcal{X}^*$ with a different data modality to guide the learning in the primary feature space $\mathcal{X}$— for example, with $\mathcal{X}^*$ as pictures and $\mathcal{X}$ as text. Conversely, the LUFe paradigm uses two subsets of the same feature set as $\mathcal{X}$ and $\mathcal{X}^*$ — which are therefore the same modality. For example, if LUFe is used with a bag-of-words text dataset, $\mathcal{X}$ and $\mathcal{X}^*$ both correspond to subsets of words within the corpus.

Secondly, the typical *privileged* information is described as being highly informative, but its availability is limited; the LUPI paradigm uses this to guide learning at train time only out of necessity because it is only accessible for training data. LUFe uses unselected features that were designated less-informative by feature selection in place of privileged information[2].

Finally, given that LUFe does not utilise a highly informative secondary feature space, the interpretation differs for how it harnesses the secondary feature set to guide learning. LUPI uses privileged information to discriminate between easy and difficult examples in the privileged space $\mathcal{X}^\star$ and subsequently transfer this information to the original data space $\mathcal{X}$ (Vapnik and Vashist, 2009; Vapnik and Izmailov, 2015). Conversely, LUFe uses features which have been designated as *less* informative as the secondary data source, therefore the interpretation of easy and hard transfer between privileged and original data

[2]This explains the change in nomenclature; we are no longer Learning Using Privileged Information (LUPI), we are Learning using Unselected Features (LUFe)

Algorithm 1 Learning using Unselected Features (LUFe)

Input a set of data points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^D$, an upper bound on the number of selected features k, and trade-off hyperparameters λ_1 and λ_2

Apply a feature selection method to produce a feature subset $\widehat{\mathcal{S}}$ with $\widehat{\mathcal{S}} \subseteq \mathcal{T}$ and $|\widehat{\mathcal{S}}| \leq \mathrm{K}$

Solve an optimization problem in (4.3)

Return a classifier f that works on selected feature subset $\widehat{\mathcal{S}}$ but does not discard the $\widehat{\mathcal{U}}$ features during training.

does not explain LUFe's behaviour. We instead consider the unselected features as a *data-dependent* upper bound on the classifier's loss function incurred using selected features.

Comparable to the way that LUFe re-purposes a LUPI algorithm, it also re-purposes feature selection. Conventional feature selection aims to identify the optimal subset $\mathcal{S}$ which maximises $\mathcal{Q}(\mathcal{S})$ and uses this to build a model, discarding the remainder, $\mathcal{U}$. LUFe also uses a feature selection algorithm but partitions the dataset into primary and secondary subsets of features — both of which are used in different roles. This is illustrated in 4.1

4.3 Experimentation

The goal of this chapter is to introduce the Learning using Unselected Features paradigm, and its application as a means to improve classifier performance. We have seen the motivation behind the LUFe setting and the theoretical justification for it. This section now describes the experimentation designed to test empirically whether LUFe can enhance classifier performance.

The LUFe approach described so far utilises the SVM+ classifier; an extension of the standard SVM. Therefore, LUFe performance was assessed through comparison with standard SVM techniques which do not employ the unselected features as a secondary dataset. An SVM classifier using standard feature selection procedure, with unselected features being discarded and not used further, is the main baseline against which LUFe performance will be evaluated. If the LUFe setting performs significantly better than the standard non-LUFe methods, this can be taken as evidence of the ability of LUFe to enhance classification performance. As a further baseline, classifier performance using the entire datasets without any feature selection was also assessed. This allows performance of standard feature selection and LUFe to be compared in terms of their improvement over

the all-features baseline (or lack of improvement), as well as in absolute terms.

4.3.1 Dataset

The initial tasks for experimentation with the LUFe method used the Technion Repository of Text Categorization Datasets. This comprises 295 datasets [3] generated from the Open Directory Project, an open-content web directory. A given dataset consists of representations of web pages from two different categories. Each instance is labelled as belonging to one of these categories, so every dataset defines a binary classification problem. Across these 295 tasks, the baseline SVM error rate is uniformly distributed between 0.4 and 0.[4]. Following the pre-processing described in 4.3.2, the number of instances per dataset range between 54 and 280 (mean 196), and the number of features range between 5140 and 60818 (mean 21154).

This collection was chosen for several factors: firstly, there is precedent for using this dataset in work on feature selection, as in Paul et al. (2015); this is partly because standard 'off-the-shelf' feature selection algorithms produce an improvement in the majority of these datasets, when compared to using the entire feature set in each case. Secondly, the dimensionality of the datasets is well-suited to the LUFe paradigm. Each dataset has a high-dimensional feature space, which allows feature selection to provide a significant reduction in computational complexity. Each dataset also comprises relatively few instances which is appropriate for using a LUPI method; as seen in 2.1, the SVM+ improves convergence rates towards the optimal solution, which is beneficial in the case of limited training data. Finally, the size of the collection and the range of baseline error rates allows exploration of the settings in which LUFe can improve performance, across a range of tasks with different levels. For example, the LUFe setting may find it easier to improve performance compared to a lower all-feature baseline, or conversely, datasets with high accuracy score for the all-feature classifier may have more informative unselected features to mediate a greater LUFe boost.

The use of the dataset for this work also has some caveats which should be noted. The motivation for the LUFe paradigm has been based around limitations on deployment-time resources, with particular emphasis on Internet of Things — which produces data characterised by "high volume, fast velocity, and different varieties of data" (Mahdavinejad et al., 2018). Conversely, this dataset collection is from a single domain of website texts,

[3] Some documentation claims there is 300 datasets in this collection, but correspondence with the maintainer of this collection confirmed that 295 is the right number

[4] http://techtc.cs.technion.ac.il/techtc300/techtc300.html

and each dataset has relatively few instances. While there are uses for text classification within the broad field of IoT, the TechTC collection is not representative of the entire field; its bag-of-words datasets particularly differ from sensor data in terms of number of instances, feature density, and feature correlation. As described above, the collection was chosen for this initial exploration because it consists of a large number of datasets with high-dimensionality and a broad range of difficulty levels. If it proves to be successful, then further testing should be applied to a broader range of datasets which are more representative of the IoT domain in which LUFe is expected to be useful.

4.3.2 Experimental protocol

The experimental protocol of Paul et al. (2015) was followed for each dataset: firstly, pre-processing removed all features corresponding to a word of less than 5 characters. Further pre-processing then standardised each variable to have zero mean and unit variance. Feature selection was then applied to partition the resulting feature set into two subsets: the k selected features, and the remaining $d - k$ unselected features, where d is the original dimensionality of the dataset. As per Paul et al. (2015), $k = 300$ and $k = 500$ were investigated. Experimentation in the initial publication of this work (Taylor et al., 2016) was carried out using the subset of 49 datasets used by Paul et al. (2015), but this section will describe further experimentation that expanded the testbed to all 295 datasets.

RFE was selected as the initial feature selection approach to determine the subsets. The resulting LUFe experimental setting is referred to as *LUFe-RFE-SVM+*. The performance of this technique was assessed in terms of classification accuracy across the 295 datasets. This was compared with two baseline settings, referred to as follows:

- *FeatSel-RFE-SVM*; a standard SVM, trained on the top k selected features only
- *ALL-SVM*; a standard SVM, trained on all d features in the dataset

There is some precedent for using RFE in feature selection for text classification, for example work by Luo and Luo (2010), where RFE was used as as second round of feature selection, following initial usage of odds ratio. However, as an iterative process, it can be slow when applied to full high-dimensionality datasets, such as the bag-of-words representations which are used in this experimentation. The step-size parameter in RFE was set to remove 10% of features at each iteration, to balance performance with running time; this is explored further in Section 5.1.7.

The classification accuracy of all three experimental settings was assessed using 10-fold cross-validation, and results are averaged over the 10 folds. Nested within each fold, a further stratified 5-fold cross-validation was used to tune the hyperparameters for each setting. The single SVM trade-off hyperparameter (λ) for *SVM-RFE* and *SVM-ALL* was selected from seven log-spaced values in the range $\{10^{-3}, ..., 10^{3}\}$. The two SVM+ trade-off parameters (λ_1 and λ_2) for *LUFe-RFE-SVM+* were jointly optimised through grid search over the same range of seven parameters; that is, 49 combinations were assessed.

Pairwise comparisons of performance between the three settings were made in two different ways: the mean accuracy of each setting, and the number of different datasets where each metric outperformed the other. Mean accuracy is an easily interpretable measure of performance for the relatively balanced datasets (mean percentage of positive instances in each dataset $= 50\% \pm 4\%$). There is not one specific 'positive' class of interest in each dataset, for which precision or recall could be used to quantify the type of error being made. The number of improvements gives a better indicator of the consistency of improvement; as a count, it is less prone to be influenced by an outlier where one setting performs significantly better or worse on a dataset. The results are primarily reported in terms of these pairwise comparisons for two reasons. Firstly, this directly depicts the point of interest in this research: the relative improvement by feature selection, and how this changes when unselected features are utilised. Additionally, pairwise comparisons are easier to visualise over a large number of datasets than all three side-by-side.

Significance testing

The widely cited protocol by Demšar (2006) for significance testing classifiers over multiple datasets was followed. This recommends non-parametrical tests — namely the Wilcoxon signed-rank test for comparisons between two classifiers and the Friedman test for two or more. These are safer than their parametric equivalents because they do not assume normal distributions or homogeneity of variance, so can be applied to classification accuracies. Also following these recommendations, results of multiple folds are not tested for significance across individual folds:

"Could we also consider the variance, or even the results of individual folds? There are variations of the ANOVA and the Friedman test which can consider multiple observations per cell provided that the observations are independent. This is not the case here, since training data in multiple random samples overlaps. We are not aware of any statistical test that could take this into account" (Demšar, 2006).

4.3.3 Implementation

All code was written in Python v.3.x.[5] Widely used 'industry standard' machine learning library Scikit-Learn was used where available, including for RFE feature selection, implementation of standard SVM, and for cross-validation. The CVXOPT (Python Software for Convex Optimization) package was used for optimisation of the SVM+ objective function in dual form (see 2.13).

4.3.4 Results

The performance of the LUFe approach was assessed in terms of classification accuracy across all 295 datasets. The general trend in the results was that RFE feature selection tended to improve classification accuracy in a majority of datasets, and that LUFe then provided a further boost with even higher accuracy in most cases. These trends are similar across the two experimental settings of $k = 300$ and $k = 500$.

Top 300 features selected

Standard feature selection (*FeatSel-RFE-SVM*) increased mean accuracy by 1.4% relative to *ALL-SVM*. Using unselected features (*LUFe-RFE-SVM+*) more than doubled this improvement, providing a further performance boost of 1.72% relative to *FeatSel-RFE-SVM*; 3.12% higher than *ALL-SVM*.

LUFe-RFE-SVM+ outperformed ALL-SVM in 225 cases (76.3%), tying in a further 7, and outperformed *FeatSel-RFE-SVM* in 212 cases (71.9%), tying in a further 12. *FeatSel-RFE-SVM* improved over *ALL-SVM* in 187 cases (63.3%) (tying in a further 7).

The improvements by *LUFe-RFE-SVM+* compared to *FeatSel-RFE-SVM*, and *ALL-SVM*, were statistically significant ($p < 0.05$). The improvement by standard feature selection was also significant ($p < 0.05$) . These comparisons between all three settings are displayed in Figure 4.2. Pairwise comparisons between the three classifiers are plotted in 4.3 to 4.5, and summarised in Table 4.1.

Comparing all three methods side-by-side, *LUFe-RFE-SVM+* was the best or joint best in 187 cases (63.3%), while *FeatSel-RFE-SVM* was in 65 cases (22%), and *ALL-SVM* was in 60 cases (20.3%). If we exclude ties where two methods performed equally well on a dataset, *LUFe-RFE-SVM+* was the single best method in 177 cases (60%), *FeatSel-RFE-SVM* was the single best in 55 cases (18.6%), and *ALL-SVM* was the single best in

[5] Names of certain programming languages and packages which are commonly typeset in all lower-case, such as "python" and "scikit-learn" are capitalised in this thesis, for clarity

53 cases (18.0%)

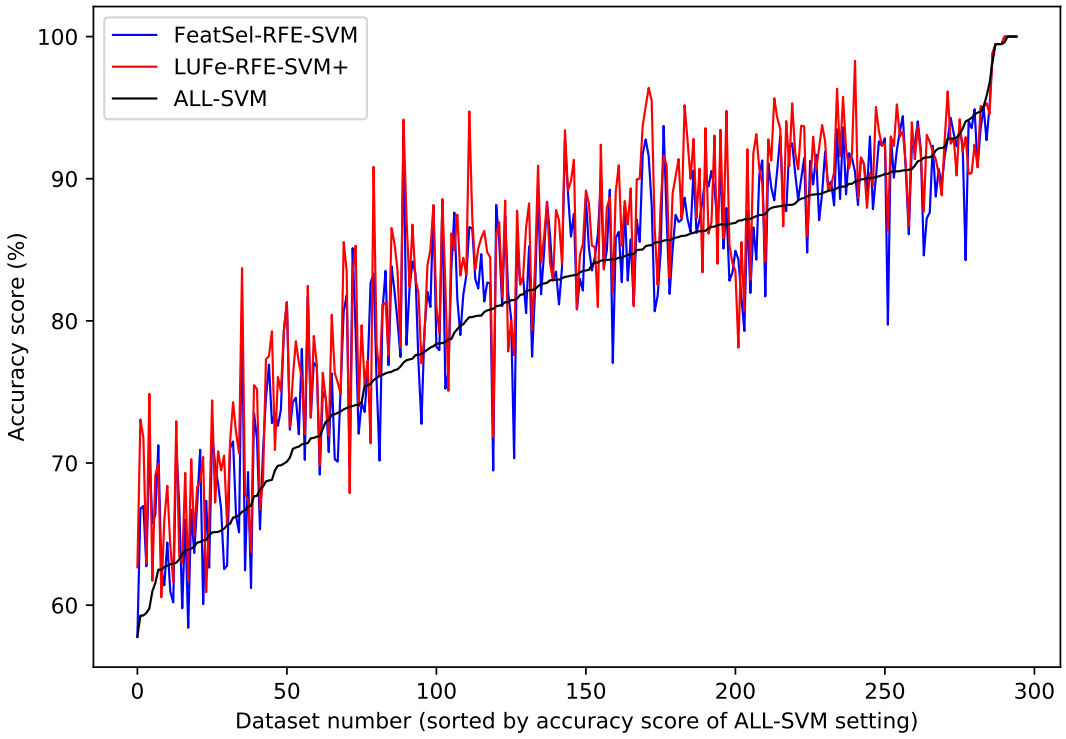

Figure 4.2: Accuracy rates (%) for *ALL*, *Featsel-RFE* and *LUFe-RFE* settings, across 295 datasets, with top 300 features selected (sorted by performance of *ALL* setting).

Top 500 features selected

Very similar results were observed when the number of selected features (k) was increased from 300 to 500.

The standard SVM with feature selection (*FeatSel-RFE-SVM*) performed even better when 500, rather than 300, features were chosen, improving by 2.0% relative to *ALL-SVM*. But there was still significant further improvement by *LUFe-RFE-SVM+*, improving by a further 1.7% (3.7% better than *ALL-SVM*).

LUFe-RFE-SVM+ now outperformed ALL-SVM in 247 cases (83.7%), tying in a further 4, and outperformed *FeatSel-RFE-SVM* in 209 cases (71.9%), tying in a further 14. *FeatSel-RFE-SVM* improved over *ALL-SVM* in 211 of 295 cases (tying in a further 4),

Again, improvements by *LUFe-RFE-SVM+* were statistically significant ($p < 0.05$) compared to both *FeatSel-RFE-SVM*, and *ALL-SVM*. These comparisons between all three settings are displayed in Figure 4.6. Pairwise comparisons between the three classifiers

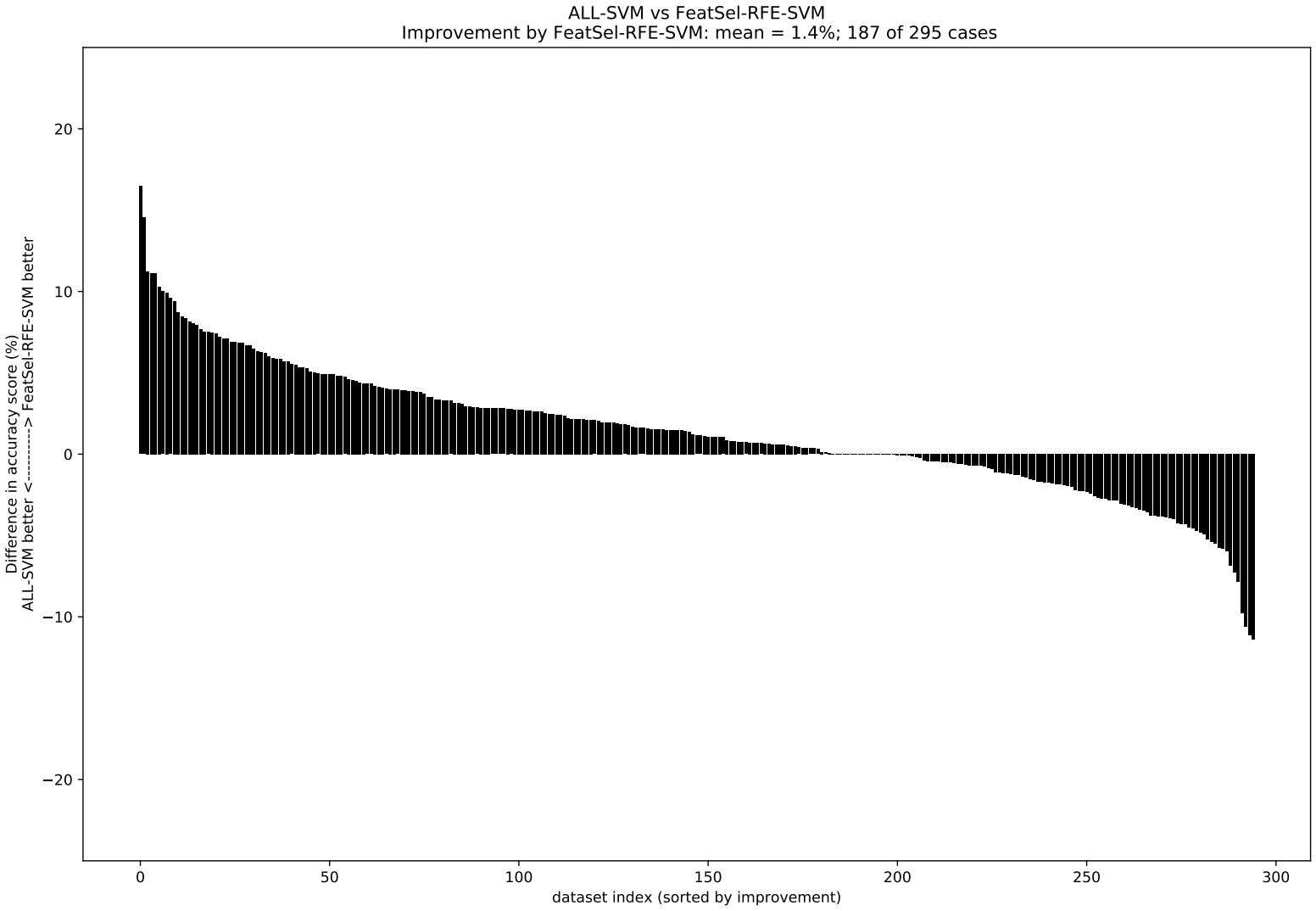

Figure 4.3: Difference in accuracy scores between *ALL-SVM*, and *FeatSel-RFE-SVM* settings, across 295 datasets, with top 300 features selected (sorted by magnitude).

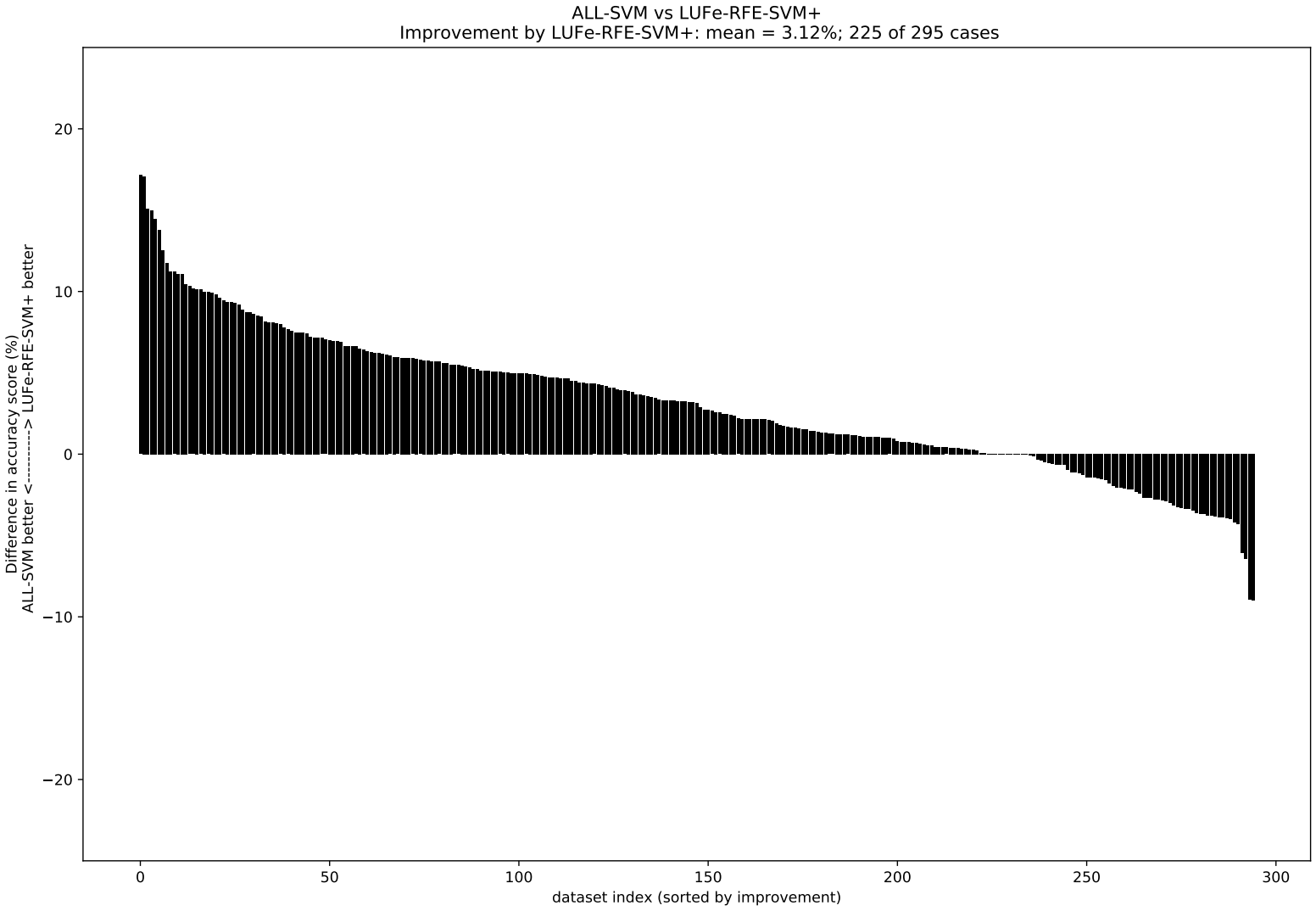

Figure 4.4: Difference in accuracy scores between *ALL-SVM*, and *LUFe-RFE-SVM+* settings, across 295 datasets, with top 300 features selected (sorted by magnitude).

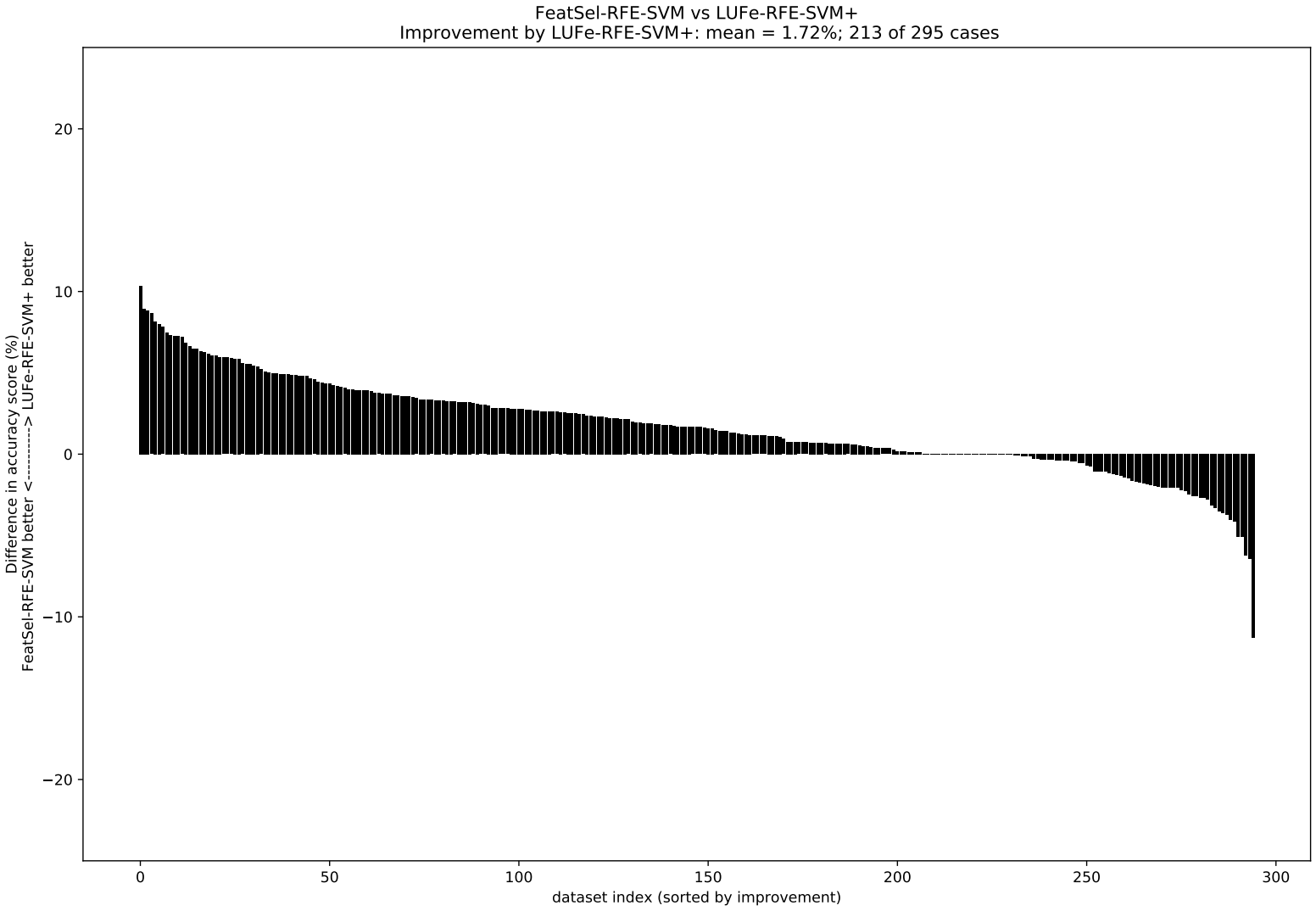

Figure 4.5: Difference in accuracy scores between *FeatSel-RFE-SVM*, and *LUFe-RFE-SVM+* settings, across 295 datasets, with top 300 features selected (sorted by magnitude).

Table 4.1: **Summary of LUFe improvements**

Accuracy scores for different settings, with improvements relative to *ALL*, and to corresponding standard feature selection setting, using the number of datasets where performance improved (out of 295 datasets), and the difference in mean accuracy score. In the setting column, 300 and 500 refer to the number of selected features.

	Improvements				
	vs ALL		vs Feature Selection		Mean
Setting	Num. wins (out of 295)	Mean (%)	Num. wins (out of 295)	Mean (%)	Accuracy (%)
ALL	-	-	-	-	81.2
FeatSel-RFE-300	187	1.4	-	-	82.6
LUFe-RFE-300	225	3.1	212	1.7	84.3
FeatSel-RFE-500	211	2.0	-	-	83.2
LUFe-RFE-500	247	3.7	209	1.7	84.9

are plotted in 4.7 to 4.9, and summarised in table 4.1.

Comparing all three methods side-by-side, *LUFe-RFE-SVM+* was the best in 202 cases (68.5%), *FeatSel-RFE-SVM* was the best in 70 cases (23.7%), and *ALL-SVM* was the best in 38 cases (12.9%). Excluding 'joint-best' ties, *LUFe-RFE-SVM+* was the single best method in 191 cases (64.7%), *FeatSel-RFE-SVM* was in 59 cases (20.0%), and *ALL-SVM* was in 34 cases (11.5%)

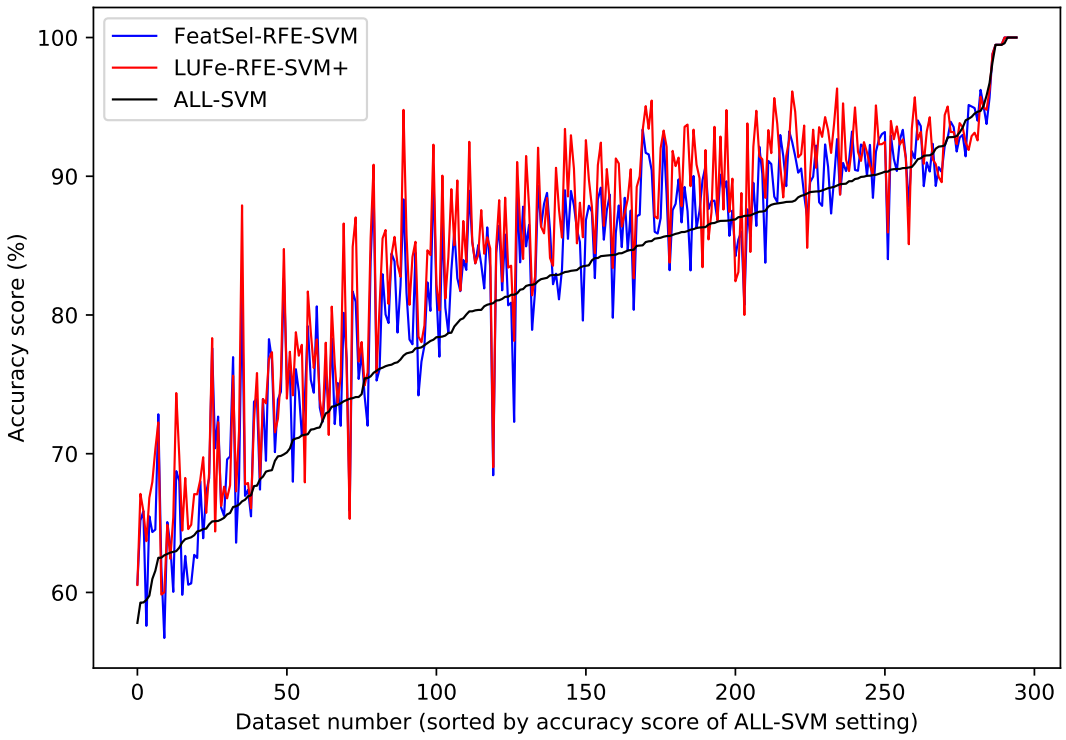

Figure 4.6: Accuracy rates (%) for *ALL*, *Featsel-RFE* and *LUFe-RFE* settings, across 295 datasets, with top 500 features selected (sorted by performance of *ALL* setting).

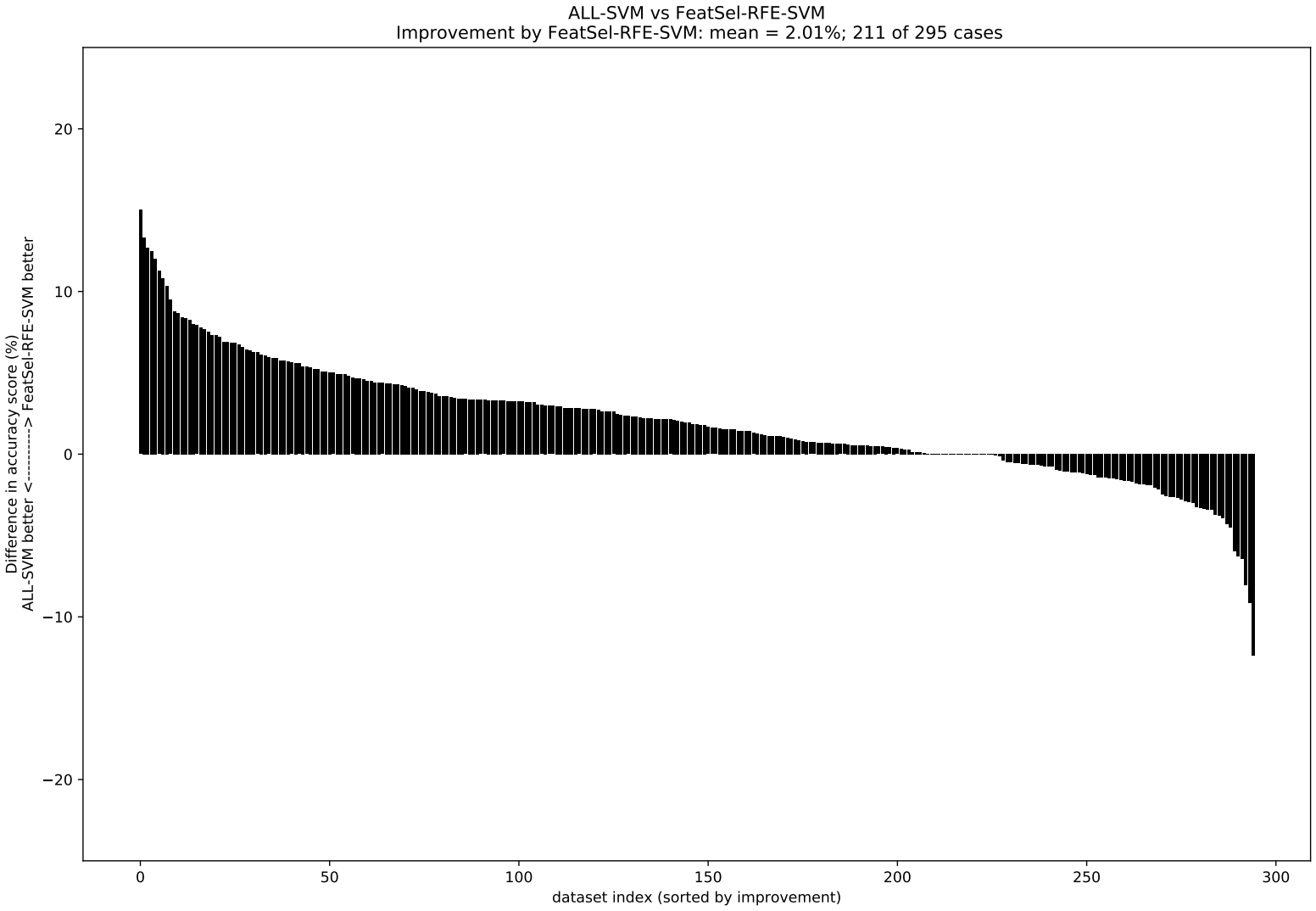

Figure 4.7: Difference in accuracy scores between *ALL-SVM*, and *FeatSel-RFE-SVM* settings, across 295 datasets, with top 500 features selected (sorted by magnitude).

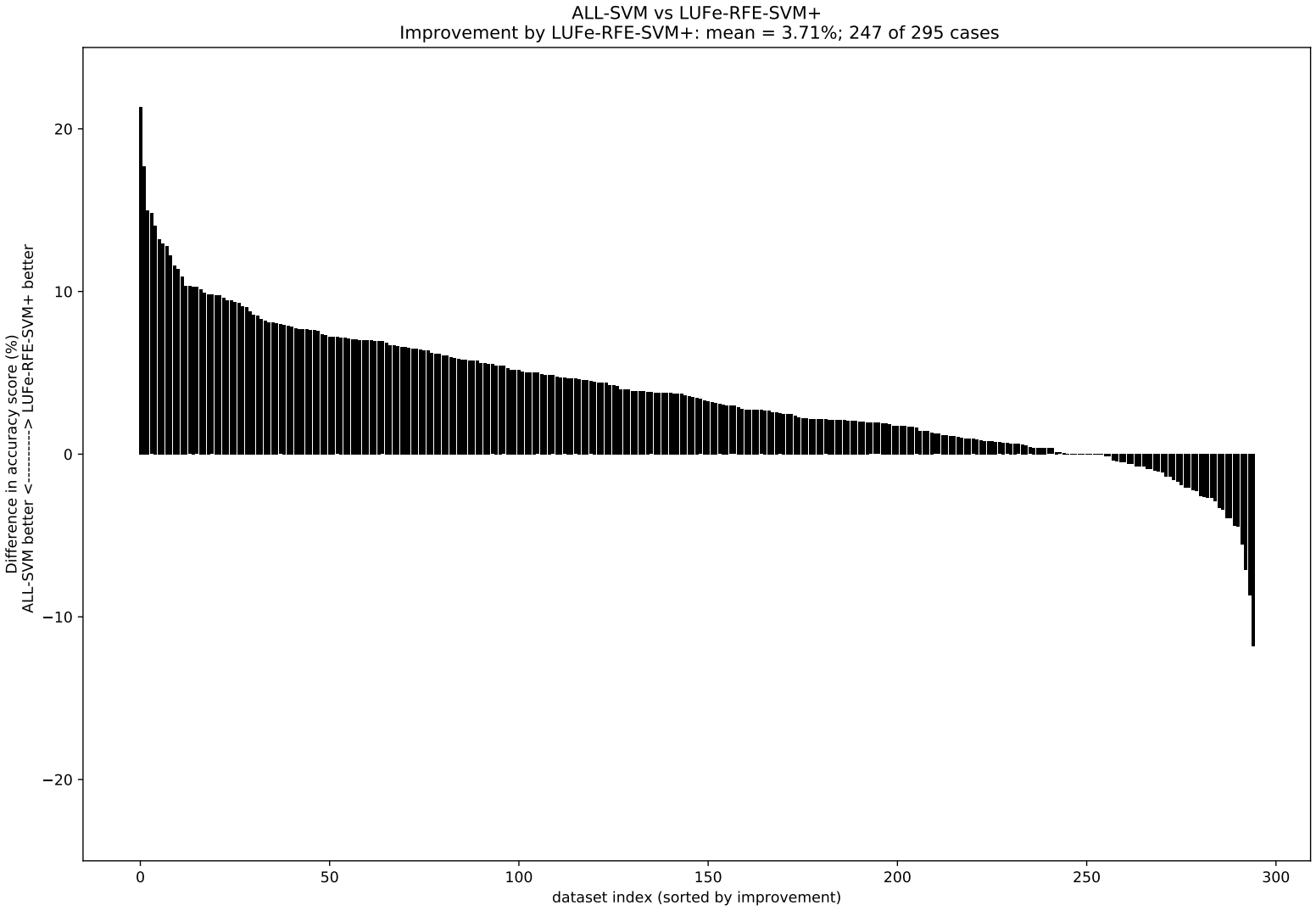

Figure 4.8: Difference in accuracy scores between *ALL-SVM*, and *LUFe-RFE-SVM+* settings, across 295 datasets, with top 500 features selected (sorted by magnitude).

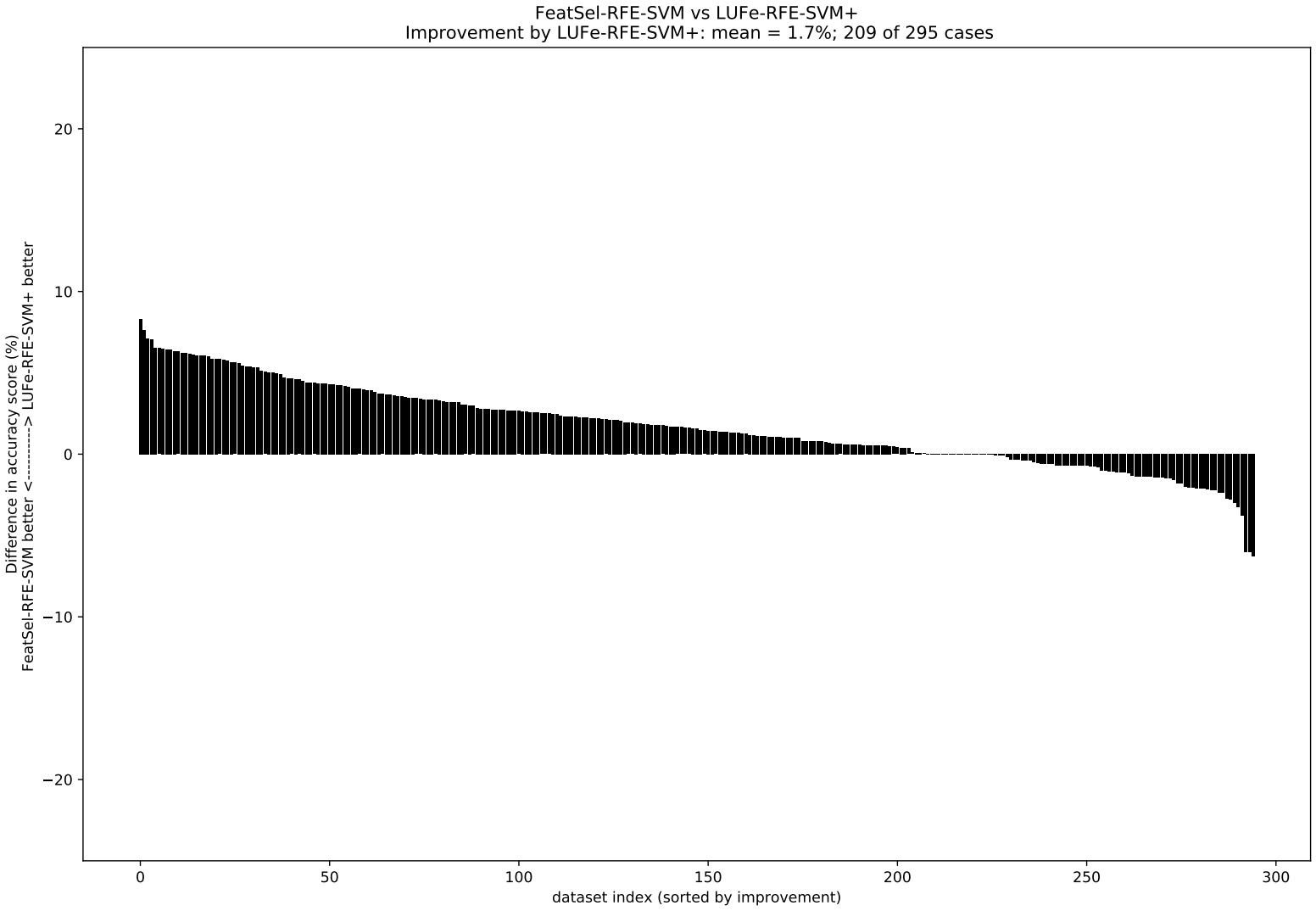

Figure 4.9: Difference in accuracy scores between *FeatSel-RFE-SVM*, and *LUFe-RFE-SVM+* settings, across 295 datasets, with top 500 features selected (sorted by magnitude).

4.3.5 Discussion of results

These initial results provide strong empirical support for the hypothesis that Learning using Unselected Features could allow an additional performance enhancement, beyond that which is provided by conventional feature selection. For $k = 300$, there were 139 cases where feature selection improved on *ALL-SVM*, and then *LUFe-RFE-SVM+* further improved on this. In a further 48 cases, standard feature selection was worse than *ALL-SVM*, but using LUFe compensated for this and beat *ALL*. For $k = 500$, results were similar. There were 156 cases where *FeatSel-RFE-SVM* improved and *LUFe-RFE-SVM+* further improved, and in a further 46, *LUFe-RFE-SVM+* improved versus *ALL-SVM* even when standard feature selection was detrimental. These results show that LUFe can help performance in two ways: (a) further enhancing feature selection which is beneficial to performance and (b) compensating for feature selection which is detrimental to performance. In both cases, LUFe takes a global view of the feature space into account, and in doing so, outperforms the standard feature selection approach with a 'local' view in 70.8%-71.9% of cases (depending on k).

It is worth emphasising that this LUFe enhancement is not simply a result of a brute force approach where all features are included. Of the three settings, *ALL* did the worst. Rather, LUFe incorporates the unselected attributes in a principled manner, as a way to guide a learner that is trained to fit the top k features. We can interpret this as follows: the less informative feature set $\mathcal{U}$ increases variance and impairs performance if used as a direct input to a classifier in concatenation with $\mathcal{S}$, but instead can help performance if their usage is restricted to a secondary feature space and a guiding role. These initial results suggest that the theoretical justification for LUFe is true. The lower-variance model with fewer features is learned, but the additional constraints on the model space $\mathcal{F}$ which is searched allow a better model that compensates for bias. However, further experimentation is required to verify whether this is in fact the mechanism by which LUFe helps performance; this will be performed in Section 6.3.

It is apparent in Figure 4.2 and 4.6 that in general, LUFe provides an additional increase in performance, on top of the improvement gained in most cases by using feature selection. In these plots, total accuracy is plotted for all three settings, for each of the 295 datasets; for clarity, the datasets are sorted by *ALL-SVM* performance, which can be considered a measurement of the "difficulty" of a dataset. This shows the trends of (a) RFE mostly improving performance compared to using the entire feature, and (b) LUFe mostly supplying an additional improvement. We can see that both of these

improvements are more consistent at first, as the baseline, *ALL-SVM* accuracy is low, but the performances level out as *ALL-SVM* performs better. Eventually, for datasets on the right of the plot, the SVM using all features performs very well, so feature selection is no longer beneficial, and neither is LUFe. This suggests a 'ceiling effect' where LUFe cannot improve performance if *ALL* performance is already high and feature selection cannot improve it.

Figure 4.10 further visualises the relationship between the different settings. Looking first at the left hand plots, for k=300 (top row) there is a very strong correlation (r=0.75) between the boost gained by standard feature selection, and the boost gained by LUFe, both compared to the all-features baseline. In other words, for those datasets where standard feature selection improved performance, then LUFe was also similarly beneficial. For k=500 (bottom row), the positive correlation between the two 'boosts' is even stronger (r=0.8). This increased strength of correlation means that when more features are in the primary subset, the LUFe boost is more closely linked to the feature selection boost. A larger proportion of classification-relevant information is in the primary feature set, so there is less variation due to the secondary feature set. Extrapolating from this, it seems likely that as k increases, *LUFe* performance becomes more closely dependent on *FeatSel* performance, as more of the classification-relevant information is included in the primary set and the secondary set has less impact.

The right-hand plot demonstrates a weak inverse correlation (r=0.20) between the improvement by *FeatSel-RFE*, with the additional improvement by LUFe over *FeatSel-RFE*. This demonstrates the 'ceiling effect' mentioned above, where LUFe is less able to improve upon feature selection that performs well. Given that the both axes show an improvement rather than an absolute score, this plot shows that the ceiling effect can be seen as a function of how much feature selection helps, rather than a function of the absolute score of the *FeatSel* setting. However, for k=500, there is no significant inverse trend.

We have seen that the results point to a LUFe boost in a majority of cases — but not in every case. Therefore, the following section analyses the results in greater depth, to understand when LUFe is beneficial.

4.4 Further analysis of results

Analysis of the results thus far has focused on the relative performance of the classifier settings, and how they interact. This section investigates the association between classi-

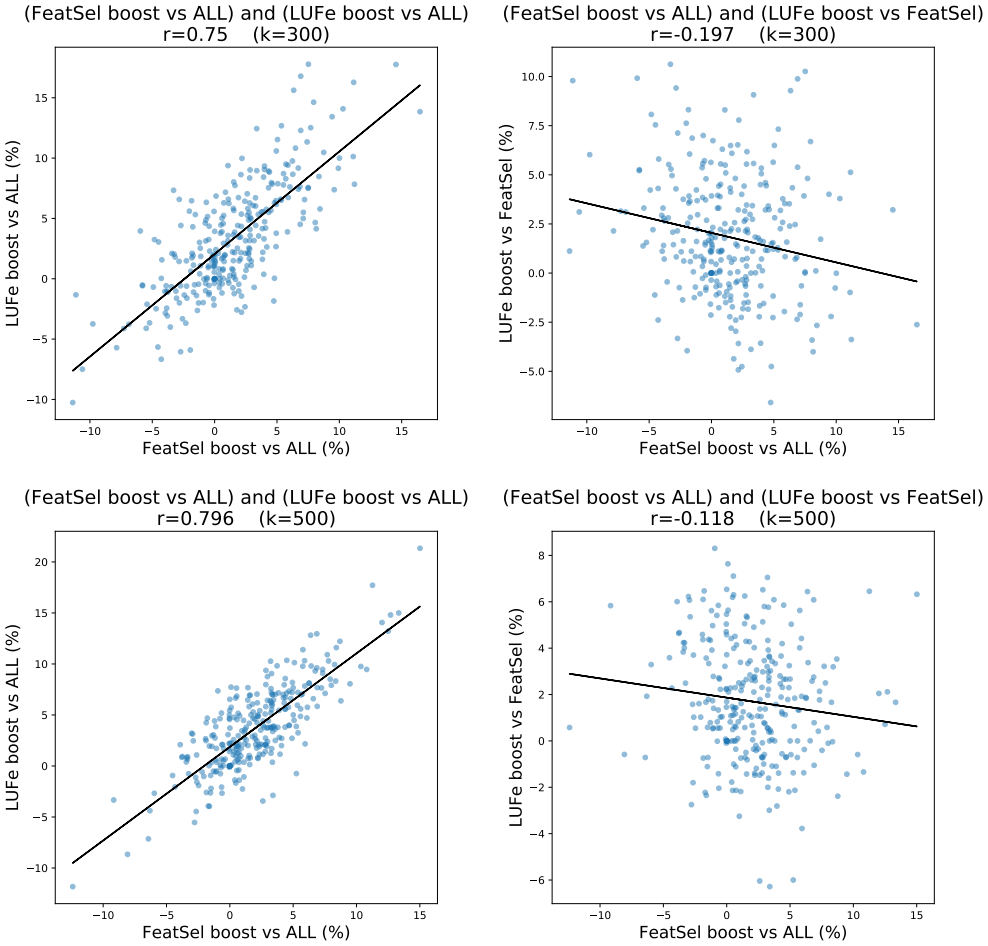

Figure 4.10: Correlation plots. Left: correlation between the improvement of *FeatSel-RFE-SVM* over *ALL-SVM*, and *LUFe-RFE-SVM+* over *ALL-SVM*. Right: correlation between the improvement of *LUFe-RFE-SVM* over *ALL-SVM*, and*FeatSel-RFE-SVM* over *ALL-SVM*. Shown for $k = 300$ (top row) and $k = 500$ (bottom row).

fier performance and the content of the underlying dataset and classification task. This attempts to identify the descriptive attributes of a dataset which make it more likely to be improved by LUFe.

4.4.1 Dataset size and class imbalance

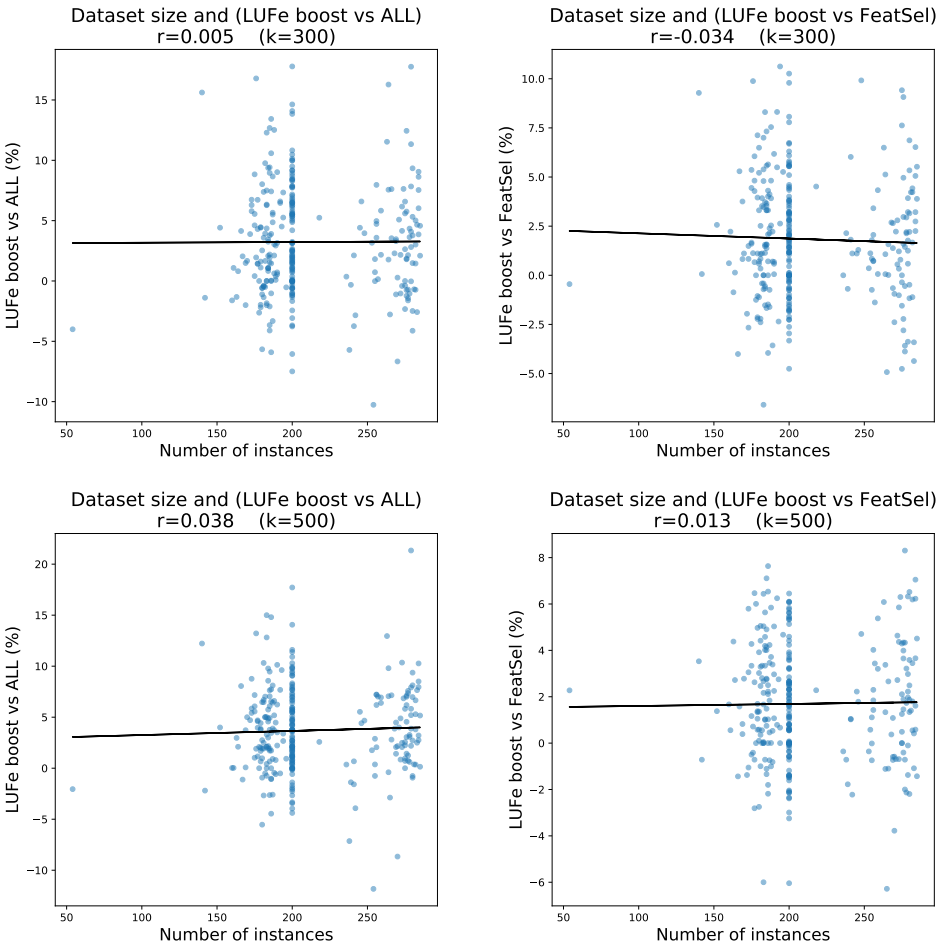

Figure 4.11: Correlation plots. Left: correlation between dataset size and the improvement of *LUFe-RFE-SVM+* over *ALL-SVM*. Right: correlation between class imbalance and the improvement of *FeatSel-RFE-SVM* over *ALL-SVM*. Shown for $k = 300$ (top row) and $k = 500$ (bottom row).

Performance was first compared with the class imbalance and the number of instances for each dataset, to see if it was associated with either factor. LUFe may be more beneficial in datasets with stronger class skew. Standard SVMs can produce suboptimal models

which are biased towards the majority class (Batuwita and Palade, 2013), so LUFe performance relative to standard SVM baselines may improve in more skewed datasets. LUFe may also be more beneficial in smaller datsets, given the original motivation of LUPI as a way to improve convergence rates (Vapnik and Vashist, 2009).

The LUFe boost was again measured relative to the all-feature baseline, and relative to standard feature selection. Class imbalance was measured as the difference between the class sizes, as a percentage, for each dataset. This was found to be very weakly correlated with LUFe improvement over *ALL* (r=022 for k=300 and r=0.24 for k=500), but there was no correlation with LUFe improvement over *FeatSel*. These results are shown in Figure 4.12. There was no correlation with dataset size for either measure of LUFe improvement, as shown in Figure 4.11.

4.4.2 Dataset topics and distance

The Tech-TC collection of datasets was generated from a corpus of web documents, hierarchically organised according to topic. Each task was analysed (a) in terms of which topics were involved, and (b) in terms of the similarity between these two topics. This section will analyse the results for k=300 from this perspective.

Topics and classifier peformance

The topics involved in each classification task were inspected, to look for trends and associations with classifier performance. The dataset was sorted by improvement by LUFe over standard feature selection, as the measure of LUFe benefit. The top and bottom 12 datasets according to this metric are shown in Table 4.2. There is no apparent patterns in which topics are present in each case.

Similarity of topics and classifier performance

The metadata for each dataset includes the graph distance between the two topic categories, indicating how far separated they are in this hierarchical tree structure. This graph distance can be taken as as a measurement of semantic separation between the two pages, and therefore how easy the classification task is likely to be; more distinct topics are reflected in more distinct and easily-distinguishable representations in the bag-of-words model. Graph distance was checked for correlation with classifier accuracy, across the 295 datasets. The correlation coefficient for the baseline *ALL* classifier without feature selection was 0.248; for *FeatSel-RFE-SVM* this dropped to 0.207 and for *LUFE-RFE-SVM+* this

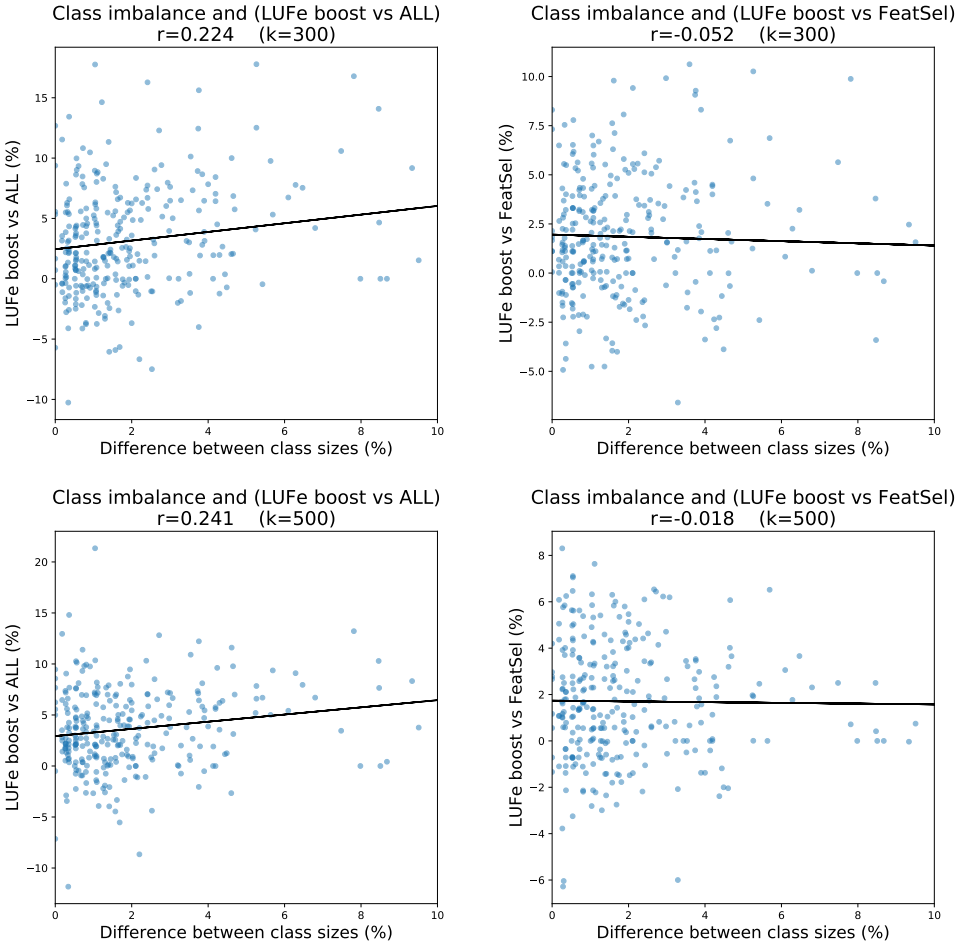

Figure 4.12: Correlation plots. Left: correlation between class imbalance and the improvement of *LUFe-RFE-SVM+* over *ALL-SVM*. Right: correlation between class imbalance and the improvement of *FeatSel-RFE-SVM* over *ALL-SVM*. Shown for $k = 300$ (top row) and $k = 500$ (bottom row).

Table 4.2: **Datasets with largest differences between LUFe and standard feature selection**

The top 12 Tech-TC datasets with the largest improvement due to LUFe, and the bottom 12 datasets with the largest decrease due to LUFe, both measured relative to standard feature selection. The hierarchical class labels have been truncated to the two lowest levels in each case.

Rank	Dataset Index	Lufe Improvement (%)	Class 1	Class 2
1	229	10.6	Kirkland/Business-and-Economy	Washington/University-of-Washington
2	17	10.3	Video-Games/Shooter	Makes-and-Models/Volkswagen
3	215	9.9	Golf/Training-Aids	Dance/Middle-Eastern
4	158	9.9	Makes-and-Models/Japanese	Religion-and-Spirituality/Meditation
5	34	9.8	Pennsylvania/Business-and-Economy	Localities/G
6	145	9.4	People/Image-Galleries	Visual-Arts/Performance-Art
7	155	9.3	Georgia/Government	Localities/B
8	143	9.1	Localities/M	Michigan/Arts-and-Entertainment
9	292	8.3	San-Diego-State-University/Departments-and-Programs	Washington/University-of-Washington
10	225	8.3	Textiles-and-Nonwovens/Resins-and-Chemicals	International/International-Schools
11	187	8.1	Localities/L	D/Dallas
12	50	7.8	Metro-Areas/Miami	Localities/S
284	98	-3.4	Business-and-Economy/Real-Estate	American/General-Motors
285	262	-3.4	Hinduism/Temples	Attack/Decommissioned
286	273	-3.6	Components/Capacitors-and-Resistors	Organizations/Development
287	252	-3.6	Child-Health/Conditions-and-Diseases	Regional/North-America
288	288	-3.9	Styles/Traditional	Latin/Salsa
289	228	-4.0	Competitions/Quiz-Bowl	San-Diego-State-University/Departments-and-Programs
290	7	-4.0	Dance/Ballroom	People/V
291	83	-4.4	Michigan/Arts-and-Entertainment	Localities/L
292	35	-4.8	Localities/S	E/Evansville
293	16	-4.8	Mail/Windows	Localities/L
294	96	-4.9	T/Tacoma	Kung-Fu/Tai-Chi
295	120	-6.6	Physics/Plasma	Agriculture/Forestry

dropped further to 0.195.

These results indicate that the underlying level of separation between classes has the most impact on the baseline without feature selection, and the least impact on the LUFe setting. This can be explained as follows: With feature selection, only the most useful discriminatory attributes are used to build the model. Given that a consistent, k best features are selected, this has the effect of 'levelling the playing field' between different datasets; it doesn't matter so much if the dataset as a whole is similar. However, the baseline model relies more on the entire feature set, which renders its performance more dependent on the underlying dataset similarity. The fact that LUFe is even less correlated than standard feature selection may then seem surprising, as LUFe uses the whole dataset at train time. However, as the full feature set is used only to guide the model fitting to the restricted subset of selected features, so gains the same benefit as standard feature selection. Furthermore the difference in correlations for *FeatSel* and *LUFe* is slight, and unlikely to be significant.

Similarity of topics and LUFe improvement

As described earlier, the benefit of the LUFe approach can be represented as the difference in score between *LUFe-RFE-SVM+* and *FeatSel-RFE-SVM* settings. Similarly, the benefit of standard feature selection is (*FeatSel-RFE-SVM* − *ALL-SVM*) and the combined benefit is given by (*LUFe-RFE-SVM+* − *ALL-SVM*). By looking at how these values correlate with the graph distance, we can see how these improvements are impacted by the difficulty of the dataset.

The correlation coefficient for graph distance with LUFe boost over standard feature selection is -0.08. For the boost due to standard feature selection, it is -0.11, and for the combined boost it is -0.15. As a whole, these results indicate that the difficulty of the task — measured by graph distance — does not significantly impact the resulting benefit by LUFe and/or feature selection.

However, we have already seen how the *FeatSel-RFE-SVM* score is inversely correlated with the LUFe boost, where it was theorised that this was due to two factors: better-performing feature selection (i) provides a harder baseline to improve upon, and (ii) leaves fewer informative features in the unselected set. These additional findings suggest that this effect was mediated by the standard SVM performance, rather than the dataset itself.

4.5 Further experimentation 1: Altering the number of selected features

The results so far provide compelling support for the benefit of LUFe, indicated by pairwise improvement over the corresponding standard feature selection approach with the same number of selected features k. However, it is worth considering that standard feature selection is sensitive to hyperparameter k, so an equivalent improvement might be achievable without LUFe, by simply adjusting k. This experimentation therefore considers a wider range of k for standard feature selection, as further benchmarks to compare LUFe approaches

4.5.1 Experimentation

Earlier results were obtained using $k = 300$ and $k = 500$, with slightly better performance by *FeatSel* and *LUFe* when $k = 500$. For further experimentation, nine more *FeatSel* settings were trained, with k in the range $\{300, 320...500\}$. This range of parameters was selected to maintain low deployment-time cost, while allowing for much broader scope to investigate sensitivity to number of selected features.

Performance was then compared in terms accuracy with the *LUFe-RFE-300* and *LUFe-RFE-500* settings seen earlier. Due to time constraints and the large computational cost of RFE feature selection, a subset of 155 datasets were randomly selected from the 295 datasets total in the collection.

4.5.2 Results and discussion

The results of this further experimentation are shown in Figure 4.13. All 11 standard feature selection approaches performed significantly worse than either of the *LUFe* settings. The new settings all performed better than *FeatSel-RFE-300* as more features were included in the primary feature set, but there was less variation *between FeatSel* settings than compared to the LUFe settings.

This finding further strengthens the viability of the LUFe technique. LUFe has been shown to provide a benefit which cannot be matched by standard feature selection without significantly altering the number of selected features. Given these promising results, the next section compares LUFe performance with the multi-task learning approach by Caruana and de Sa (2003), which was designed to tackle a similar issue.

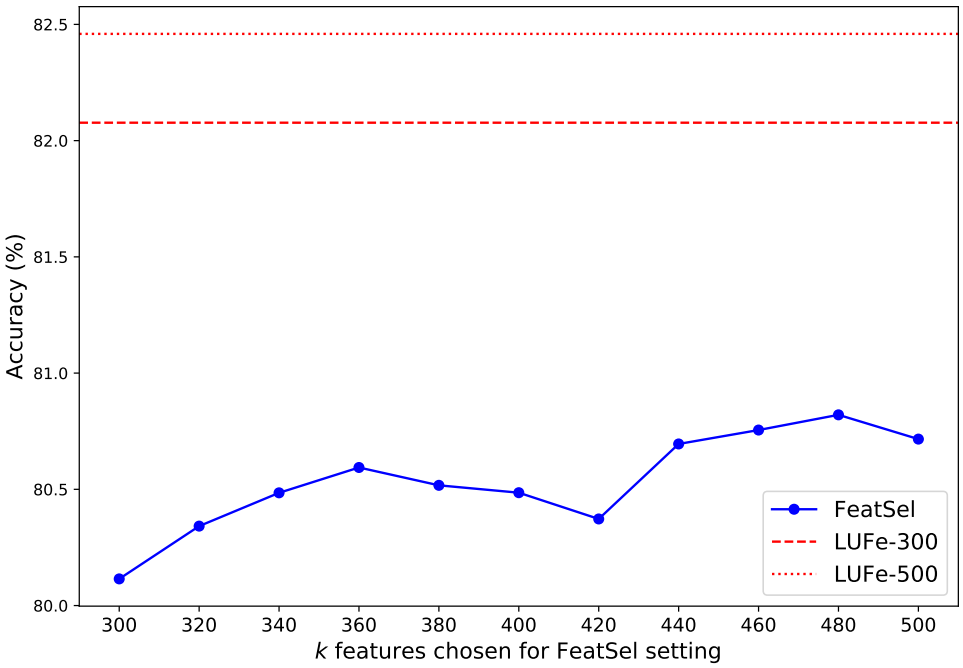

Figure 4.13: Comparison of LUFe settings against a range of standard feature selection settings, in terms of accuracy. *LUFe-RFE-300* and *LUFe-RFE-500* plotted as dashed red lines. *FeatSel-RFE* settings plotted as solid blue line for k in the range $\{300, 320...500\}$.

4.6 Further experimentation 2: Comparison with Existing Method to Utilise Discarded Features

Section 3.1 described the only earlier attempt to utilise unselected features, by Caruana and de Sa (2003). This work took the approach of using discarded variables as a secondary output to be learned by a multi-task learning (MTL) setting — in contrast with the LUFe usage of unselected features as a secondary input. In reconstructing this feature set, the neural network learns a more accurate function on the primary task (which in this case is classification). This strategy can retroactively be considered to fall under the wider umbrella of LUFe, as it similarly involves a feature selection procedure, followed by the use of those which were discarded in a secondary role. Therefore, this approach will be referred to as *LUFe-MTL* for brevity. As the only previous LUFe-style approach, *LUFe-MTL* is a sensible benchmark to compare *LUFe-SVM+* performance. The experimentation earlier in this chapter serve as a proof-of-concept that unselected features can indeed be harnessed to enhance classification performance; this experimentation investigates whether *LUFe-SVM+* is the best way to do so.

In both *LUFe-MTL* and *LUFe-SVM+*, the unused features represent additional information about each training data point, and are used to direct the learner to produce a more accurate classifier in the normal feature space. In both cases, the unselected features only need to be collected for training data, and are not taken into account once the model has been learned. However, the way in which the features are used differs between methods. The unselected features in *LUFe-SVM+* are a secondary input that add further constraints to the function space which is searched during risk minimisation. In *LUFe-MTL*, the unselected features are a secondary output and in learning to reconstruct the unselected features from the selected features, biases the function that maps from inputs to the labels in the 'main task'. Caruana and de Sa (2003) define feature selection as "used to find the subset of variables to use as inputs", whereas LUFe uses feature selection as deciding which are used as *primary* inputs — and which are used as secondary.

The different strategies employed to harness the power of unselected features may result in different performance. Caruana and de Sa (2003) reported only a modest improvement from the *LUFe-MTL*, intended just to supply a proof-of-concept to this approach. The results published on classification in that paper only increased performance relative to the standard feature selection approach by 0.2%. However, this result was on a single dataset

which already achieved 91% accuracy with no feature selection — so was a difficult baseline to improve upon. As previously discussed, the *LUFe-RFe* setting also improved less in datasets where the *ALL* already achieved high accuracy. Therefore, *LUFe-MTL* might similarly lead to a greater performance boost when there is less of a ceiling effect caused by high baseline scores.

4.6.1 Experimental procedure

LUFe-MTL was implemented as a benchmark to compare *LUFe-SVM+* performance across the 295 Tech-TC datasets. Both methods' training consists of two distinct stages: feature selection first allocates selected and unselected subsets, and then a model is trained to fit this selected subset, informed by the unselected subset. This two-stage approach means that any feature selection can be 'plugged in' to both the algorithms. The same selected subset $\mathcal{S}_{RFE}$ was therefore provided as the primary feature set to *LUFe-MTL* and *LUFe-SVM+*, and the two methods of employing extra features compared. The complete unselected subset $\mathcal{U}$ is used in full for LUFe-SVM+ but only a subset of the top k features within $\mathcal{U}$ is used in LUFe-MTL. This is consistent with the methodology described in Caruana and de Sa (2003), and was necessary due to the the computational complexity of running LUFe-MTL with the entire feature set; training with the entire $\mathcal{U}$ as secondary output was an order of magnitude slower than LUFe-SVM+. The same partition of datasets was used to implement 10-fold cross-validation as in previous experimentation, to allow direct comparison between settings. All pre-processing was consistent across settings.

The general architecture for the *MTL* method followed the description from the original implementation of this approach: a shared hidden layer for both tasks, which outputs to two separate, task-specific output layers. A shared representation is therefore learned for both tasks which may better capture the underlying generator of data than either task independently; this is then input to the output layers for a further task-specific data transformation to make predictions. The outputs are a label prediction for the primary task, and a regression on the k unselected features. The multi-task learning neural network was implemented in TensorFlow.

The number of units in the hidden layer was set at 3200; this was consistent with the original implementation of this method, in proportion to the number of input dimensions. Caruana and de Sa (2003) used 30 selected features as input dimensions and achieved optimal performance with 320 hidden units. Some preliminary experimentation using

different numbers of hidden units was carried out on a subset of 5 datasets to set this neural net hyperparameter. Architectures with 300, 1600, 3200 and 6400 units were tested, and while 3200 performed better than using fewer units, there was no significant improvement from further increasing the number beyond 3200. Given the consistency of this result with that seen in the earlier work, this parameter was fixed to avoid further computationally hyperparameter estimation. Using significantly more units than the input dimensionality increases the expressivity of the model learned.

The model learns by minimising the error function, which was a sum of the respective errors for the two tasks. The main task error was calculated using softmax cross-entropy between the logits and the true labels. The secondary task error was calculated using l2 loss between the true value and predicted value for the unselected features, normalised by the number of unselected features.

Using a non-linear kernel

Neural networks are universal approximators and allow any function, including non-linear ones, to be learned on the input data. Conversely, the SVM+ method in *LUFe-RFE-SVM+* utilised a linear kernel, which is disadvantageous in comparison, as it means a more limited function space is searched to learn mapping function f. Therefore, a non-linear version of SVM+ was also used to compare performance with the linear *LUFe-RFE-SVM+* amd *LUFe-RFE-MTL*. An RBF kernel instead of linear kernel was used to transform the data, allowing non-linear models to be learned (Hsu et al., 2003), as described in Chapter 2. This setting is referred to as *LUFE-RFE-RBF-SVM+*.

Hyperparameter selection is more challenging when using an RBF kernel. Two additional hyperparameters (ω and ω^*) involved in making the kernel computation need to be set; further to the γ_1 and γ_2 hyperparameters in the SVM+. A grid search over all four parameters quickly becomes intractable; if the same range of seven values were searched over, 2401 combinations would be searched.

To solve this, the 'median trick' heuristic can be employed to estimate suitable ω_1 and ω_2 values in the SVM+, with ω_1 and ω_2 set based on the median kernel distance of X and X^*, respectively. The λ_1 and λ_2 parameters continued to be chosen using grid search over the same seven values.

4.6.2 Results

There was no significant difference in performance between any of the three settings over the 295 datasets ($p > 0.05$). *LUFe-RFE-SVM+* performed better than *LUFe-MTL* in 142 of 295 cases (48.1%), and *LUFe-MTL* was better in 149 cases (50.5%). The two classifiers achieved equal accuracy in the remaining 4 cases (1.4%). LUFe-MTL achieved 84.6% mean accuracy across all datasets, improving by 0.23% over LUFe-SVM+ (84.3%).

Introducing an RBF kernel in the LUFe SVM+ led to a very slight (0.1%) improvement in mean accuracy over the linear kernel, to 84.5% across all datasets. *LUFe-RFE-RBF-SVM+* beat *LUFe-RFE-SVM+* in 150 cases (50.8%) while the linear setting was better in 141 cases (47.8%). This non-linear SVM+ based method was therefore closer to *LUFe-MTL* performance in terms of mean accuracy; *LUFe-MTL* was 0.1% higher. However, *LUFe-MTL* was better in 158 cases, and the non-linear SVM+ was only better in 133 cases.

4.6.3 Discussion

The two strategies for Learning using Unselected Features performed very similarly, with no significant difference in mean accuracy across all the datasets. Performance was also not significantly impacted by the inclusion of non-linearity for the *LUFe-SVM+* method via the RBF kernel. In terms of classification accuracy then, the novel Learning using Unselected Features approach which was introduced in this chapter is an equally powerful way to boost performance as the previously-reported strategy for using discarded features via multi-task learning.

It is also worth noting that the *LUFe-MTL* method led to a considerably larger mean improvement than that which was seen in the original "proof-of-concept" description of this work. *LUFe-MTL* increased performance by 1.95% compared to using only the selected features — this was over tenfold the improvement to an equivalent setting in a classification task reported by Caruana and de Sa (2003). This increase in performance boost is in part attributable to the fact that the baseline performance was lower in this work, allowing for more room for improvement.

LUFe-MTL only uses a subset of unselected features $\mathcal{U}$ as a secondary output, whereas the *LUFe-SVM+* methods take the entire $\mathcal{U}$ as a secondary input feature set. It is possible that the entire feature set or a larger subset of this could be employed to further enhance *LUFe-MTL* by using a larger number of training outputs in the secondary task. The inverse is true for *LUFe-SVM+*; this used the entire unselected set $\mathcal{U}$ as secondary input

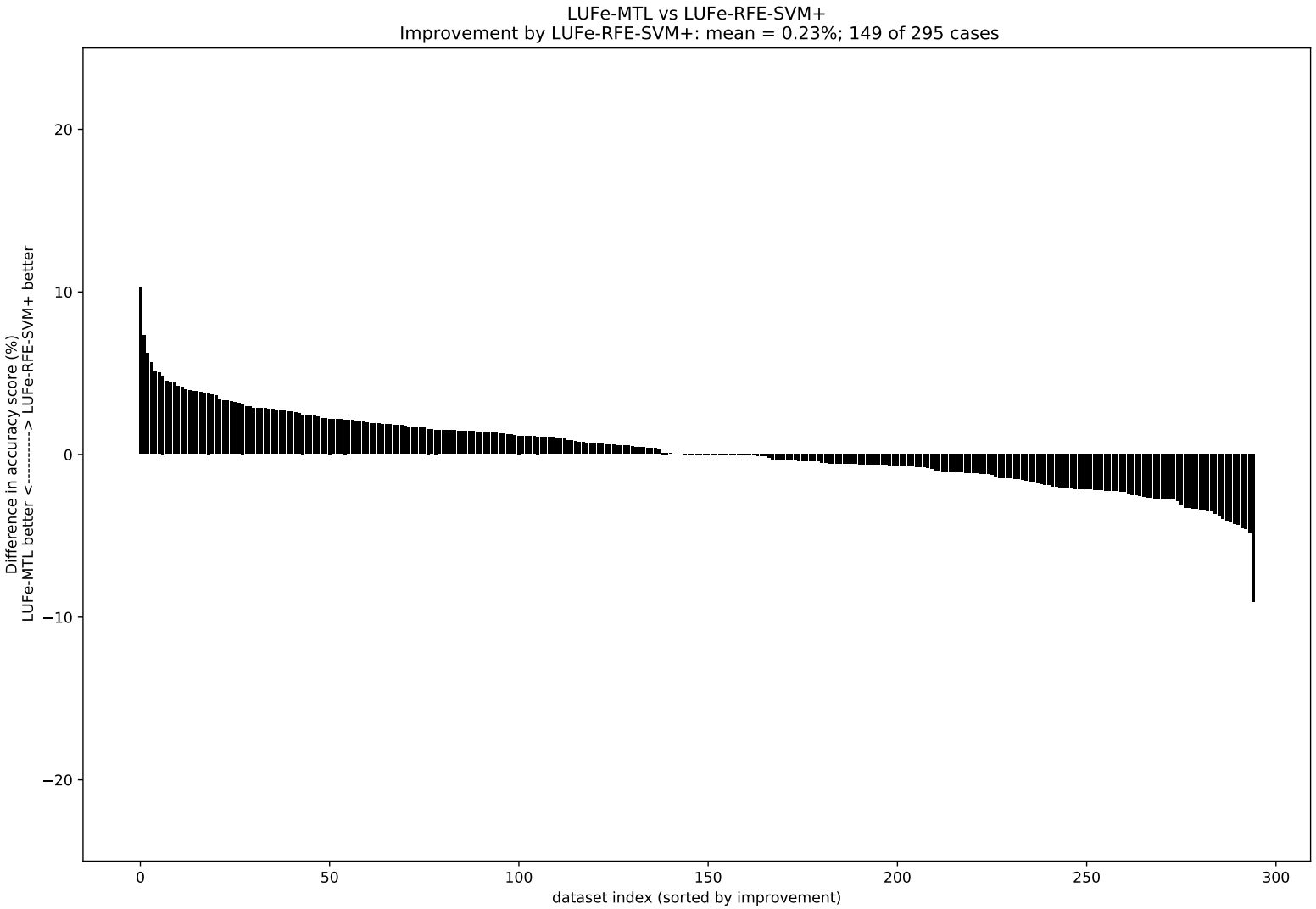

Figure 4.14: Difference in accuracy scores across 295 datasets between between LUFe using Multi-Task Learning, and LUFe using SVM+ with a linear kernel

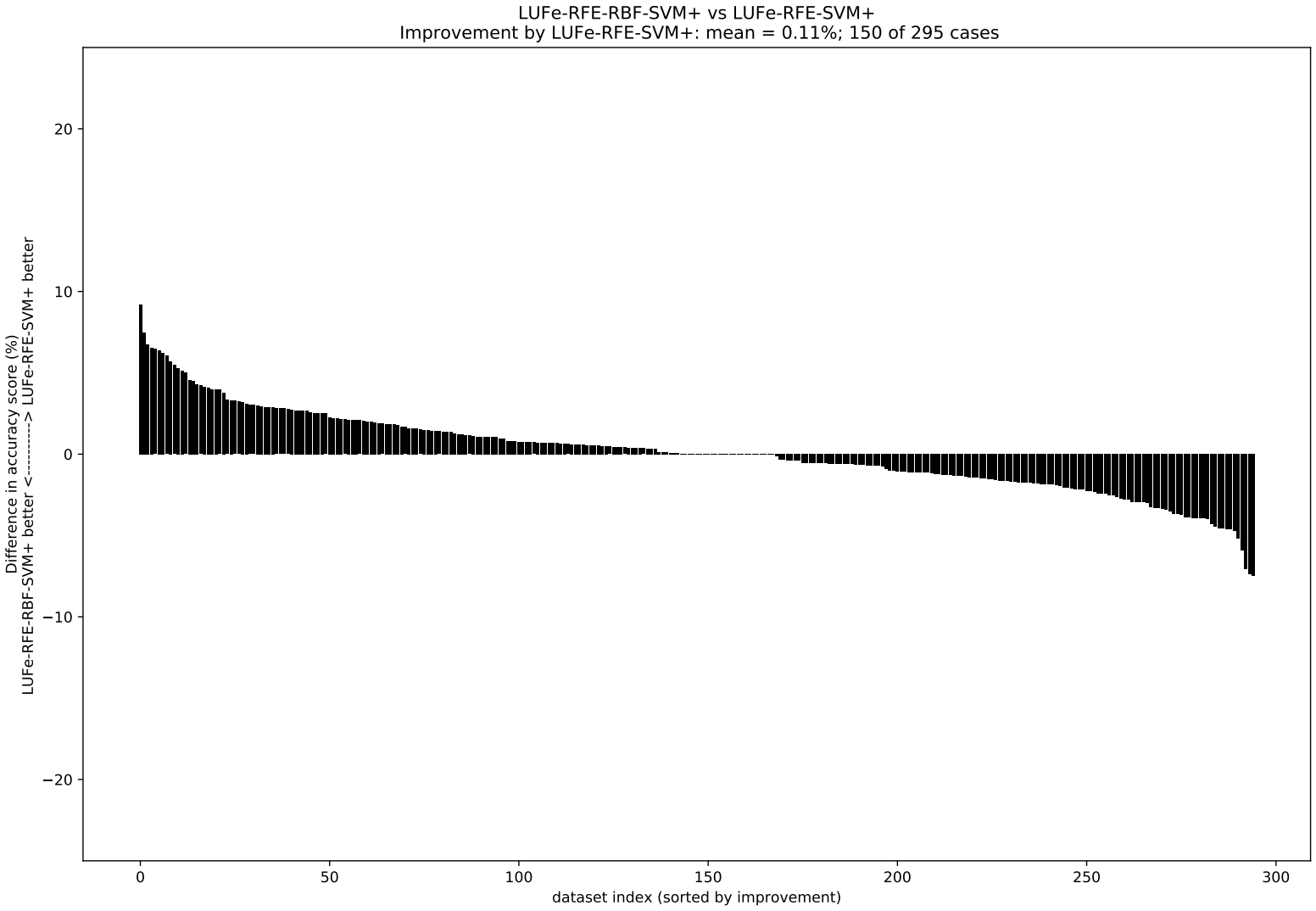

Figure 4.15: Difference in accuracy scores across 295 datasets between two LUFe-RFE-SVM+ settings: one using linear kernel, one using a non-linear (RBF) kernel

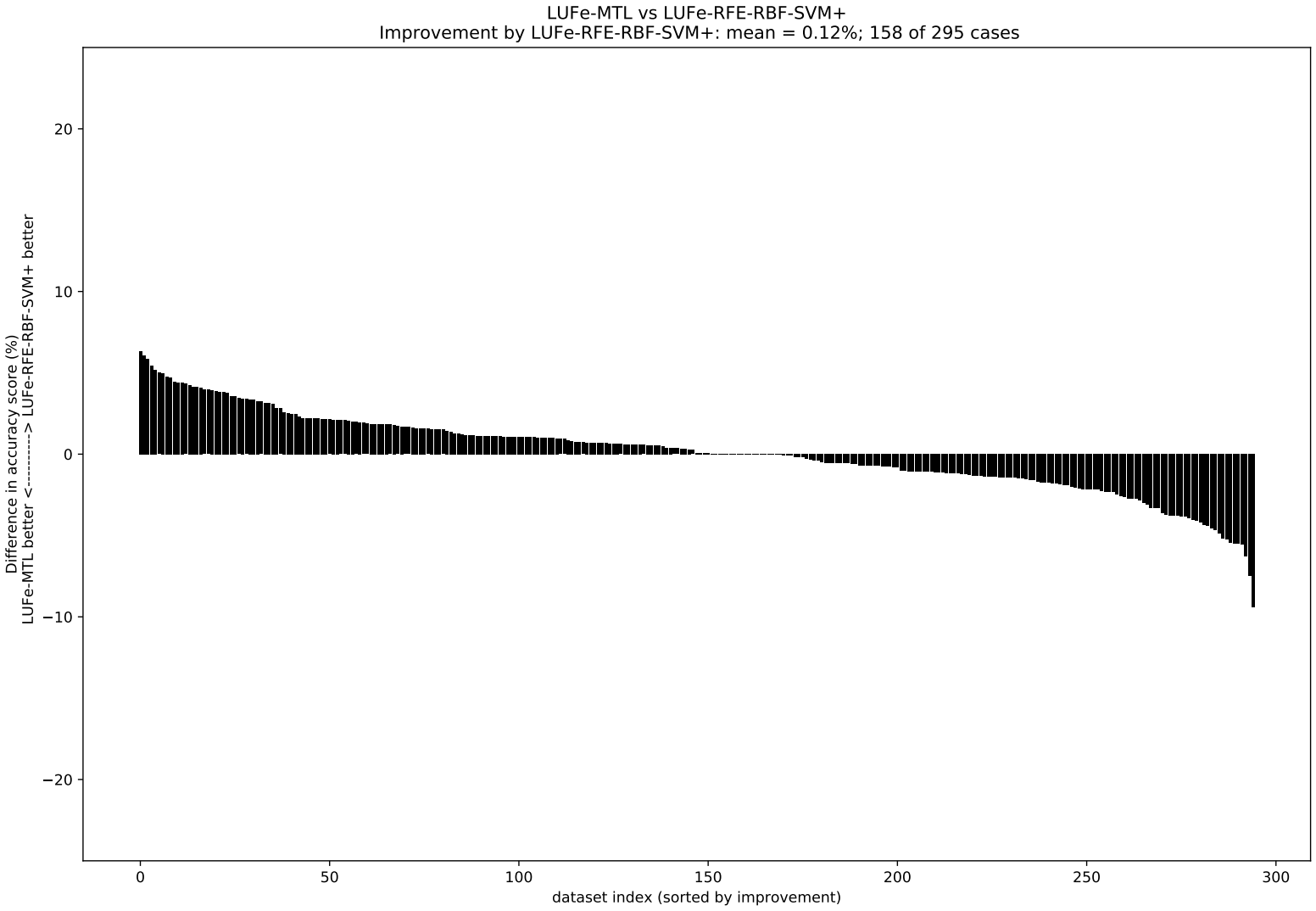

Figure 4.16: Difference in accuracy scores across 295 datasets between between LUFe using Multi-Task Learning, and LUFe using SVM+ with a non-linear (RBF) kernel

and it is possible that using a more task-specific subset would gain better performance. Both of these directions for research, for *LUFe-SVM+* and *LUFe-MTL*, will be explored in subsequent chapters.

It is likely that *LUFe-RFE-RBF-SVM+* performance could be marginally improved with a more extensive hyperparameter estimation procedure. However, computational and temporal limitations restrict experimental settings to the median trick, which produced a classifier that is not significantly different to that using the linear kernel.[6]

The similar results seen for different kernels are in fact in keeping with a very widely cited guide to support vector classification by Hsu et al. (2003), which states that "if the number of features is large, one may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance." They go on to state that the use of the linear kernel is good enough for the high-dimensional situation, and has the further advantage of only requiring search for a single hyperparameter. This general guidance was reflected in the same paper's experimentation, where the cross-validation accuracy was comparable between linear and RBF kernels when the number of features greatly exceed the number of instances. The authors conclude that it may not be necessary to map the data in this case. These results in this thesis are based on bag-of-words datasets, which are high-dimensional with few instances — so the minimal benefit of non-linearity is not surprising. The rest of this work will therefore focus only on using a linear kernel for both SVM and SVM+ based models.

The usage of *LUFe-RFE-SVM+* over *LUFe-MTL* despite their similar performance can be justified as there are some benefits to using an SVM-based method over a neural network-based one. SVM is equivalent to a neural network with a single hidden layer. Neural network classification accuracy tends to improve linearly with an increasing amount of training data, so this approach is preferable when training data is abundant. However, the SVM+ setting is designed to improve convergence rates to compensate for limited training data, so may be advantageous in this situation where neural network approaches are sub-optimal.

This comparison of approaches also opens the possibility for a future approach which combines the two. A fusion approach could be developed that combines the use of both additional input features as in the LUFe setting, and additional output features, as in

[6] As an alternative approach, a second *LUFe-RBF-SVM+* setting was also investigated. Here, all parameters were cross-validated, but using a smaller range of 3 values $\{10^{-2}, 10^{0}, 10^{2}\}$, resulting in a more manageable 81 combinations. Performance of this classifier was very similar, with mean accuracy over 295 datasets 0.1% lower than the fixed median trick described above.

in multi-task learning. The *LUFe-MTL* approach of using only a subset of unselected features could be taken, for example with the top k unselected features used as secondary input, and the next top k used as secondary output.

4.6.4 Closing remarks

These initial results provide a compelling proof-of-concept for LUFe, and demonstrate its potential to improve performance in classification tasks. LUFe provided a boost in a majority of cases, though was subject to a 'ceiling effect' when standard feature selection performed well. There was no evidence of a relationship between LUFe boost and dataset size, or LUFe boost and class imbalance. Given this lack of relationship, it is difficult to predict how LUFe would perform when applied to a domain with radically different characteristics — such as IoT sensors. However, the robust statistical significance of the improvement, and the lack of significant dependency on dataset attributes, suggest that it may transferable to other domains; further experimentation is required to test this.

The limitations to these results prompt other questions for investigation. The first concerns the robustness of these findings in a wider variety of settings. LUFe was shown to improve upon RFE feature selection, but these results do not give any insight into whether LUFe could also help with filter feature selection techniques. Also, these experiments have been conducted using only the SVM+ implementation of LUFe; other LUPI algorithms may or may not be similarly able to harness unselected features. Secondly, further work is required to establish the exact means by which LUFe was able to improve classification performance; these results do not confirm whether the reduced bias theory is in fact true.

The following chapters will address these questions raised by this initial positive result for LUFe. Given the similarity of results between $k = 300$ and $k = 500$, further experimentation will use only the former setting. This is to avoid a combinatorial explosion of different parameter settings which could occur as more experiments are conducted. Using only one setting halves the amount of experimentation required, and the use of fewer features ($k = 300$) at deployment time more closely follows the requirement for limited resources, which motivates the LUFe paradigm.

Chapter 5

Further Investigating Unselected Features

Following the 'proof of concept' for Learning using Unselected Features (LUFe) in Chapter 4, this chapter tests the functionality of LUFe in a wider range of settings. The work performed in Chapter 1 used only recursive feature elimination (RFE) as a method to select features, and then only used SVM+ as a single Learning Using Privileged Information (LUPI) algorithm to instantiate the LUFe paradigm. This raises the question of whether the efficacy of LUFe depends on this specific form of feature selection, and this specific LUPI framework. Alternatively, we can ask whether there is a general underlying benefit that can be gained from using unselected features at train time, which is ambivalent both to the means of feature selection, and to the LUPI algorithm used to exploit the unselected features. If LUFe is indeed robust and stable across different methods, this raises the subsequent question of which methods produce the most benefit, and if this improvement can be further enhanced in other ways — for example, applying selectivity to the secondary feature set.

To address these questions, this chapter expands the usage of LUFe in a number of ways:

- **Using alternative feature selection methods:**
- **Using alternative implementations of LUPI**
- **Using only subsets of the unselected features**

5.1 Feature selection methods

As explained in 4.6, the LUFe paradigm consists of two discrete stages: feature selection and training. This means that different feature selection methods can be 'plugged in', to select a subset $\widehat{\mathcal{S}}$ — consisting of the k most informative features — and assign the remainder as secondary feature set $\widehat{\mathcal{U}}$, before the two subsets are then used in training. Regardless of the metric used, the features which are not selected serve the same purpose of providing a data-dependent upper bound on the error calculated on selected features. Recall the LUFe objective function defined in Chapter 4:

$$\underset{\mathbf{w},b}{\text{minimize}} \ \|\mathbf{w}\|_{\ell_2}^2 \tag{5.1a}$$

$$\text{subject to,} \quad \forall\, i = 1, \ldots, N,$$

$$\underbrace{1 - y_i[\langle \mathbf{w}, \mathbf{x}_i^{\widehat{\mathcal{S}}} \rangle + b]}_{\substack{\text{classifier's loss} \\ \text{based on selected features}}} \leq \underbrace{\left\langle \mathbf{x}_i^{\widehat{\mathcal{U}}}, \mathcal{Q}(\widehat{\mathcal{U}}) \right\rangle}_{\substack{\text{data-dependent upper bound} \\ \text{based on unselected features}}}. \tag{5.1b}$$

Different feature selection methods provide different definitions of the criterion $\mathcal{Q}(\cdot)$, resulting in different $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$. Therefore the model parameters $\mathbf{w}$ are fit to a different $\widehat{\mathcal{S}}$, and the learning process is constrained using a different set of constraints, based on a different $\widehat{\mathcal{U}}$. As described in Chapter 2.2, feature selection methods are broadly classified as filter, wrapper and embedded methods. Filter and wrapper methods were described as providing a 'local view' of the data, as they select a limited subset of features, which is subsequently provided to the predictive model, in place of the entire dataset. This was contrasted with the 'global view' afforded by wrapper methods, where the predictive model can access all the features. The first way to expand the application of LUFe then was to investigate its performance when applied to other types of feature selection. The LUFe setting is a means to enhance the 'local view' taken by wrapper and filter methods, to gain the 'global view' of an embedded method, at training time only. RFE is the *de facto* wrapper method, so four filter methods were also investigated. These methods are based around the same fundamental principle: each attribute is assessed individually, using the metric to score its informativeness about the labels. These scores for each feature can then be used to rank the featureset and thereby choose a subset consisting of the top-ranked t features.

5.1.1 Comparison of filter and wrapper methods

The univariate approach to feature selection has a disadvantage when compared with wrapper approaches, as the features are not assessed *in situ*. RFE iteratively trains a

classifier on a feature set — starting with all features — and progressively eliminates features based on their contribution to the classifier. Therefore RFE is able to take the interaction of different features into account. Univariate approaches are not as effective in this regard. As a simple example, two identical features would be equally informative about the label and therefore ranked consecutively by univariate methods. This would introduce redundancy into $\widehat{\mathcal{S}}$ if both were ranked within the top t features, and therefore a sub-optimal set would be selected. RFE does not share this deficiency.

However, RFE is a greedy algorithm, choosing an optimal feature set at each iteration; therefore it can only approximate a globally optimal solution for the NP-hard problem of feature selection. This means that RFE is sensitive to the 'step-size' parameter which controls the number of features to be eliminated in each iteration. The number of iterations required to reduce the size of the feature set to a given t is inversely correlated with the step-size. However, this impact on feature selection running time must be traded off with the granularity of the approach. Step-size was set at 0.1 in Chapter 4, which increases the likelihood that a sub-optimal subset of features was chosen. The feature selection method chosen is of even more importance when only a subset of unselected features is used. When the top 10% of unselected features are taken, for example as in *LUFe-MTL*, it is necessary that the unselected features be ordered. It is possible that the boost provided by *LUFe-RFE-SVM+* compared to *FeatSel-RFE-SVM* occurs precisely due to the weaknesses of RFE. If LUFe can harness informative unselected features that a sub-optimal feature selection 'should' have selected, as posited in Chapter 4, then the boost is dependent on the weakness of RFE, and the same improvement may not be seen when LUFe is used with univariate feature selection approaches which individually rank the entire feature set. This provides the motivation for the following experimentation: attempting to validate whether LUFe efficacy does not depend on the shortcomings of a single feature selection method.

5.1.2 Experimental procedure

To investigate the impact of feature selection method on the efficacy of LUFe, four different filter feature selection methods were investigated. The four metrics considered were ANOVA, BAHSIC, Chi-squared and Mutual Information, each of which are described in Section 2.2. In a comparison of feature selection methods for text classification, Yang and Pedersen (1997) found chi-squared to perform joint-best, and mutual information to perform worst among the metrics tested. Using these different methods then allows LUFe

performance results to be observed with a range of different efficacies of feature selection. Furthermore, the same study found chi-squared performance to be very closely associated with that of information gain and document frequency, so its use here can be taken as indicative of how these measures would perform.

The same testbed of 295 datasets was used, with identical pre-processing as described in Section 4.3.1. However, chi-squared feature selection is generally used for categorical features, as it requires variables to be non-negative. Therefore, an additional pre-processing step was performed before applying chi-squared feature selection, to ensure that all features fulfilled this requirement: a constant was added to each attribute, equal to the additive inverse of the largest negative value for that feature. Feature selection was performed 'inside' each fold of cross-validation, as Refaeilzadeh et al. (2007) observe that performing feature selection 'outside' the loop is tantamount to peeking at held-out test data, and results in over-optimistic (falsely-inflated) accuracy scores.

Similar procedure was followed as described in Chapter 4.3.2. First, each of the investigated feature selection metrics was used to partition the dataset into two subsets: the top 300 features, $\widehat{\mathcal{S}}$ and the remainder, $\widehat{\mathcal{U}}$. These subsets were then passed to two classifiers, providing two experimental settings for each metric: a standard SVM, trained using only $\widehat{\mathcal{S}}$, and an SVM+ LUFe setting that used $\widehat{\mathcal{S}}$ as normal features, and $\widehat{\mathcal{U}}$ as the secondary feature set. The standard setting is referred to as *FeatSel-*{*feature selection method*}*-SVM*; for example, *FeatSel-ANOVA-SVM*. The LUFe setting is referred to as *LUFe-*{*feature selection method*}*-SVM+*; for example *LUFe-ANOVA-SVM+*. The methodology was consistent with Chapter 4; the 295 datasets from Tech-TC collection were used, with the same pre-processing and the same splits for cross-validation. Performance on these relatively balanced datasets was again assessed in terms of accuracy as explained in 4.3.2, and compared between *LUFe* and *FeatSel* settings, with the SVM trained on all features (*ALL-SVM*) used as a further baseline.

5.1.3 Results

In broad terms, similar patterns of results emerged when using univariate filter feature selection, to what was observed with a wrapper method in Chapter 4. Applying any standard feature selection method significantly improved ($p < 0.05$) mean accuracy score compared to *ALL*, and using LUFe led to a further significant improvement ($p < 0.05$) on top of this, for all feature selection methods. When looking at individual datasets, LUFe improved in at least 67.1% of cases over standard feature selection and in at least 82.4%

Table 5.1: **Summary Results**

Mean accuracy scores for *FeatSel* and *LUFe* settings with five different feature selection metrics. Pairwise improvements in accuracy are shown for *FeatSel* over *ALL*, *LUFe* over *ALL* and *LUFe* over *FeatSel*. Significant pairwise differences are marked with *.

	Mean accuracy (%)		Improvement in mean accuracy		
Feature selection	Feat Sel	LUFe	FS vs ALL	LUFe vs ALL	LUFe vs FS
ALL	81.23%	-	-	-	
RFE	82.62%	84.34%	1.40%*	3.12%*	1.72 %*
ANOVA	83.89%	85.87%	2.66%*	4.65%*	1.99%*
BAHSIC	83.79%	85.92%	2.56%*	4.70%*	2.14%*
CHI^2	83.95%	85.86%	2.72%*	4.63%*	1.91%*
MI	85.00%	86.31%	3.78%*	5.09%*	1.31%*

of cases over the *ALL* baseline. Standard feature selection also improved over the baseline in a majority of cases.

Results: mean scores

Mean scores for all settings are summarised in Table 5.1 and Figure 5.1. All standard univariate feature selection techniques produced a greater improvement in classification accuracy compared to *ALL* than that provided by RFE (1.4%). Using ANOVA as the metric for standard feature selection increased performance by 2.66% . BAHSIC and Chi-squared feature selection provided a similar improvement of 2.56% and 2.72% respectively, compared to *ALL*. Mutual information performed best amongst the standard feature selection techniques, increasing accuracy by 3.78% compared to *ALL*.

For each of these feature selection methods, accuracy score was then further improved by employing the unselected features in the LUFe paradigm; that is, each *LUFe-SVM+* setting outperformed the corresponding *FeatSel-SVM* approach. *LUFe-ANOVA-SVM+* provided a further improvement of 1.99%, totalling 4.65% improvement over *ALL*. *LUFe-BAHSIC-SVM+* provided a further improvement of 2.14%; the largest improvement by any LUFe method relative to the standard feature selection; this was a 4.70% improvement over *ALL*. *LUFe-Chi2-SVM+* provided a further improvement of 1.91%, totalling 4.63% improvement over *ALL*. *LUFe-MI-SVM+* increased accuracy by 1.31%, the smallest provided by any LUFe setting relative to its corresponding standard feature selection.

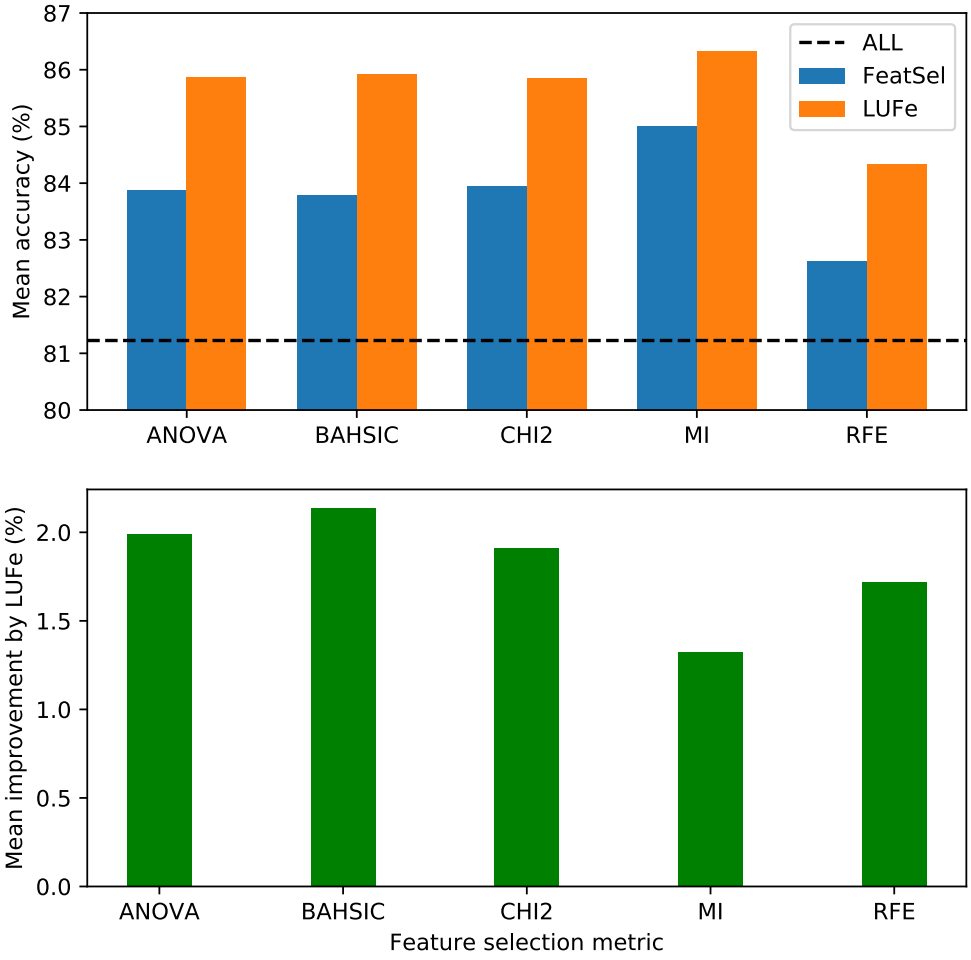

Figure 5.1: Top: mean accuracy scores of *LUFe-SVM+* and *FeatSel-SVM* settings, across 295 TechTC datasets, for five different feature selection techniques. Baseline SVM using all features is marked as dotted line. Bottom: improvement in mean accuracy score by *LUFe-SVM+* over corresponding *SVM* setting, for the five feature selection techniques

However, as the *FeatSel-MI-SVM+* was already high, this totalled 5.09% improvement over *ALL*, and was the best-performing approach among all investigated settings. In each case, these improvements in mean accuracy by LUFe over the corresponding *FeatSel* setting represent a significant difference ($p < 0.05$).

Results: number of datasets

Univariate feature selection methods were more consistently more beneficial than RFE, in terms of the number of datasets where they improved accuracy. These results are summarised in Tables 5.3 and 5.4. *LUFe-ANOVA-SVM+* outperformed *ALL-SVM* in 252 of 295 cases (85.4%), and beat *FeatSel-ANOVA-SVM* in 215 of 295 cases (72.9%). *LUFe-BAHSIC-SVM+* also outperformed *ALL-SVM* in 252 of 295 cases (85.4%), and beat *FeatSel-BAHSIC* in 226 of 295 cases (76.6%). Similarly, *LUFe-CHI2-SVM+* outperformed *ALL-SVM* in 251 of 295 cases (85.1%), and beat *CHI2-SVM* in 210 of 295 cases (71.2%) *LUFe-MI-SVM+* outperformed *ALL-SVM* in 243 of 295 cases (82.4%), and beat *MI-SVM* in 198 of 295 cases (67.1%). For comparison, *LUFe-RFE-SVM+* outperformed *ALL-SVM* in 225 of 295 cases (76.3%), and beat *RFE-SVM* in 212 of 295 cases (71.9%).

5.1.4 Discussion

The results show a consistency across wrapper and filter feature selection methods, which demonstrates that LUFe is not dependent on a single feature selection algorithm. The overall score achieved by each LUFe approach appears to depend on the underlying feature selection approach; MI was the best for LUFe and standard feature selection, RFE was worst for both, and the other three methods performed similarly in both. This indicates that LUFe performance partially depends on the quality of the primary feature set.

However, the smallest improvement by LUFe relative to standard feature selection occurred with MI. This suggests that there is also a 'ceiling effect' where the standard feature selection performed well enough that LUFe found it harder to improve upon. This mirrors the effect seen on the level of individual datasets in Figures 5.2 to 5.5, where LUFe boosts performance less when accuracy is already high. The 'ceiling effect' can be explained by two factors. Firstly, when standard feature selection performs better, it is simply a higher baseline and therefore harder for LUFe to improve upon. Secondly, if the classifier performs better, then feature selection has probably done a better job, so fewer informative features remain in the unselected set whcih can be harnessed to boost performance.

Table 5.2: **Correlations between classifier performance and dataset 'difficulty'** Table shows — for each feature selection method — the correlation coefficient calculated between the graph distance between the two classes in each task, with the corresponding accuracy score of *Feature Selection* and *LUFe* classifiers.

Feature selection	Standard SVM	LUFe
ALL	0.248	
RFE	0.207	0.195
ANOVA	0.204	0.202
BAHSIC	0.201	0.212
CHI^2	0.201	0.202
MI	0.089	0.177

The ranking of the standard feature selection methods contrasts with Yang and Pedersen (1997), where MI was found to be least effective, and Chi-squared the best. The authors explain that Mutual Information has a bias in favour of low-frequency terms, and is sensitive to probability mention errors, but also state that it is task-sensitive. Therefore, the difference in dataset size and dimensionality may be the cause of MI sucess. In another analysis of feature selection for text classification, Forman (2003) state that "it is difficult to beat the performance of SVM using all available features. In fact, it is sometimes claimed that feature selection is unnecessary for SVMs". This also contrasts with the results seen in this work, and is likely due to a high number of redundant variables in this dataset, without which the classifier performs better.

5.1.5 Further analysis of results

As in Section 4.4.2, the results were further analysed, in reference to the underlying classification task which was being performed in each case. The graph distance for each dataset was again used as a proxy measure of task difficulty, and assessed for correlation with *FeatSel-SVM* and *LUFe-SVM+* settings, for each feature selection method. As in Section 4.4.2 the accuracy of both standard and LUFe settings were found to be less correlated than that of all-features baseline; this finding was true across all five feature selection techniques and is summarised in Table 5.2.

Table 5.3: **Summary of comparisons between methods**

Pairwise comparisons of *ALL*, *Feature Selection* and *LUFe* settings for each feature selection method across 295 datasets. Comparisons are made in terms of the number of datasets where one setting achieved higher accuracy, or where they were equal ('tie').

	ALL vs Feat Sel			ALL vs LUFe			Feat Sel vs LUFe		
Feature selection	ALL	tie	FS	ALL	tie	LUFe	FS	tie	LUFe
RFE	101	7	187	63	7	225	71	12	212
ANOVA	72	9	214	36	7	252	71	9	215
BAHSIC	76	4	215	36	7	252	64	5	226
CHI^2	70	9	216	37	7	251	76	9	210
MI	93	4	198	44	8	243	92	5	198

Table 5.4: **Summary of comparisons between methods**

Pairwise comparisons of *ALL*, *Feature Selection* and *LUFe* settings for each feature selection method across 295 datasets. Content is the same as Table 5.3 (above) but results are presented in terms of the percentage of 295 datasets.

	ALL vs Feat Sel			ALL vs LUFe			Feat Sel vs LUFe		
Feature selection	ALL	tie	FS	ALL	tie	LUFe	FS	tie	LUFe
RFE	34.2%	2.4%	63.4%	21.4%	2.4%	76.3%	24.1%	4.1%	71.9%
ANOVA	24.4%	3.1%	72.5%	12.2%	2.4%	85.4%	24.1%	3.1%	72.9%
BAHSIC	25.8%	1.4%	72.9%	12.2%	2.4%	85.4%	21.7%	1.7%	76.6%
CHI^2	23.7%	3.1%	73.2%	12.5%	2.4%	85.1%	25.8%	3.1%	71.2%
MI	31.5%	1.4%	67.1%	14.9%	2.7%	82.4%	31.2%	1.7%	67.1%

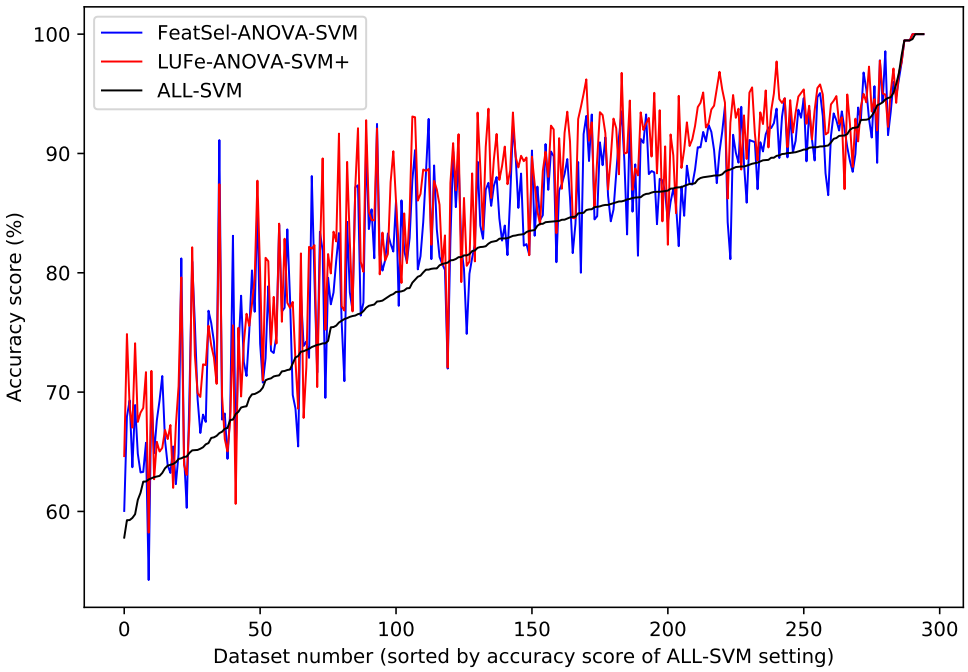

Figure 5.2: **ANOVA accuracy rates** (%) for *ALL*, *FeatSel-ANOVA* and *LUFe-ANOVA* settings, across 295 datasets (sorted by performance of *ALL* setting).

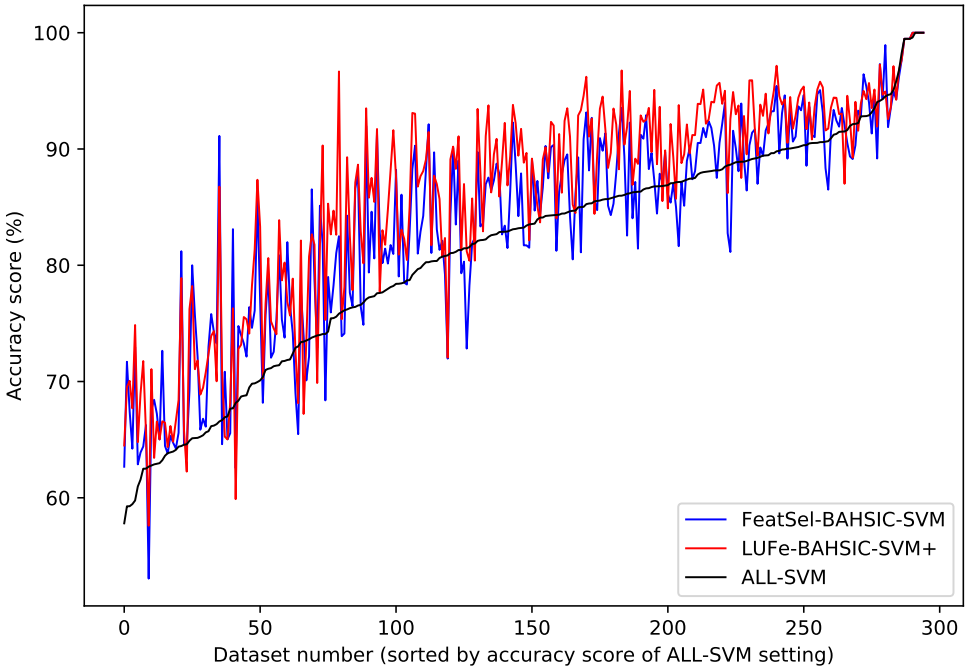

Figure 5.3: **BAHSIC accuracy rates** (%) for *ALL*, *FeatSel-BAHSIC* and *LUFe-BAHSIC* settings, across 295 datasets (sorted by performance of *ALL* setting).

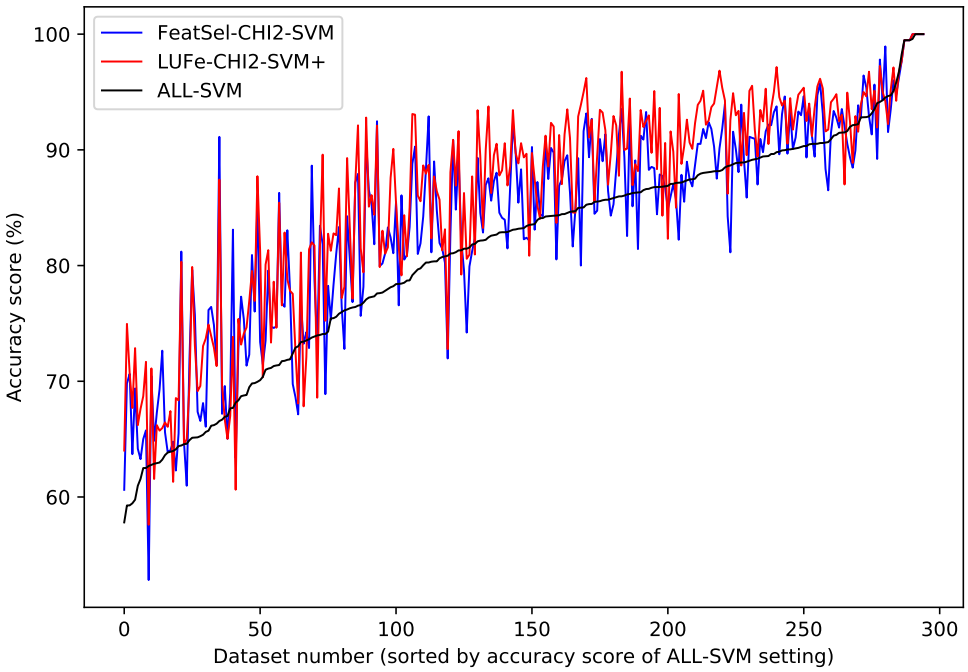

Figure 5.4: **Chi-squared accuracy rates** (%) for *ALL*, *FeatSel-Chi*^{2} and *LUFe-Chi*^{2} settings, across 295 datasets (sorted by performance of *ALL* setting).

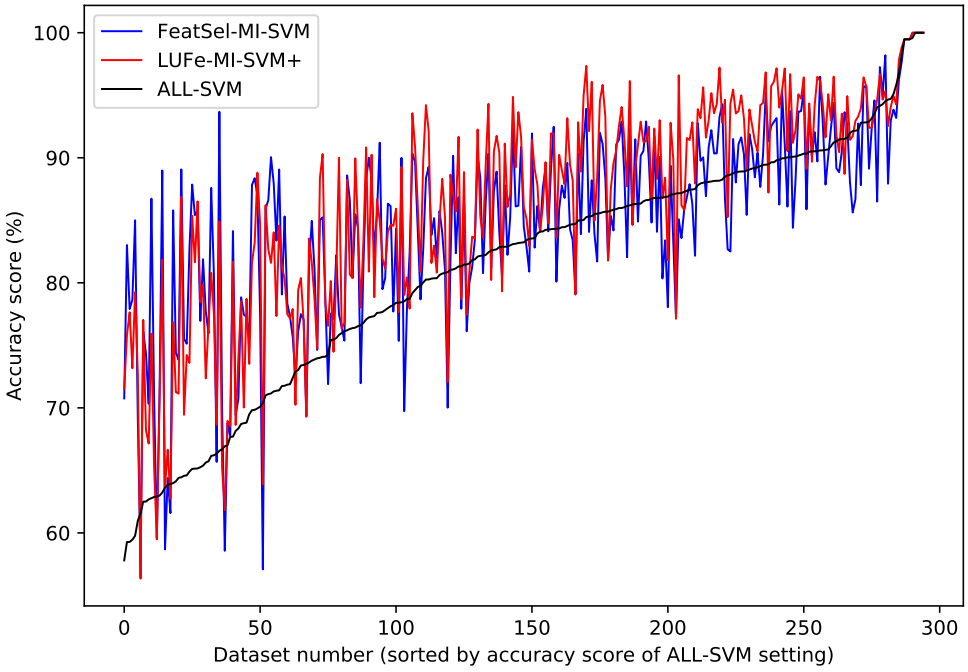

Figure 5.5: **Mutual Information accuracy rates** (%) for *ALL*, *FeatSel-MI* and *LUFe-MI* settings, across 295 datasets (sorted by performance of *ALL* setting).

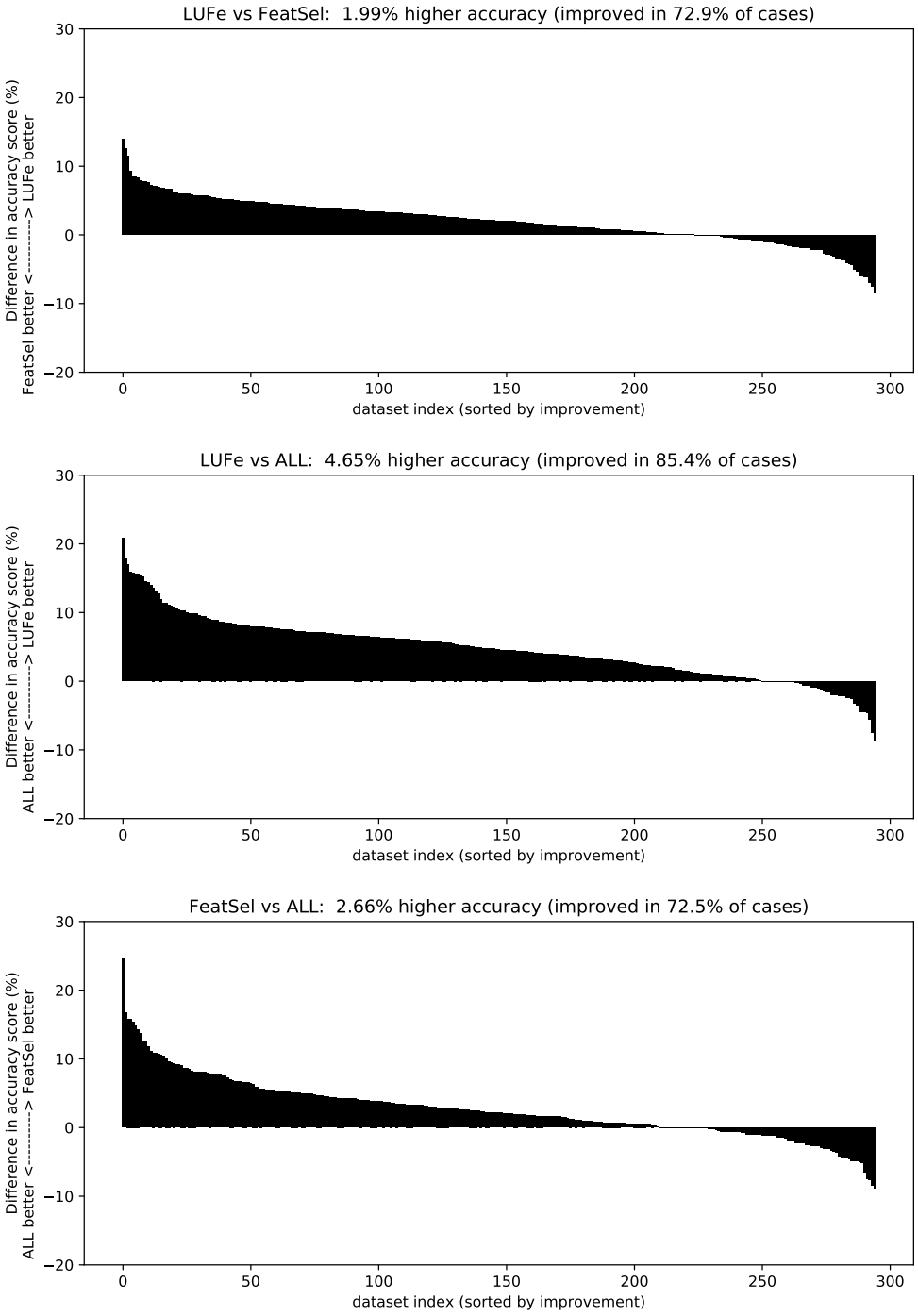

Figure 5.6: Pairwise differences in accuracy scores between *LUFe-ANOVA-SVM+*, *FeatSel-ANOVA-SVM*, and *ALL-SVM* settings, for 295 datasets from the TechTC collection (sorted by magnitude of accuracy difference). Top: *LUFe-ANOVA-SVM+* vs *FeatSel-ANOVA-SVM*, middle: *LUFe-ANOVA-SVM+* vs *ALL-SVM*, bottom: *FeatSel-ANOVA-SVM* vs *ALL-SVM*.

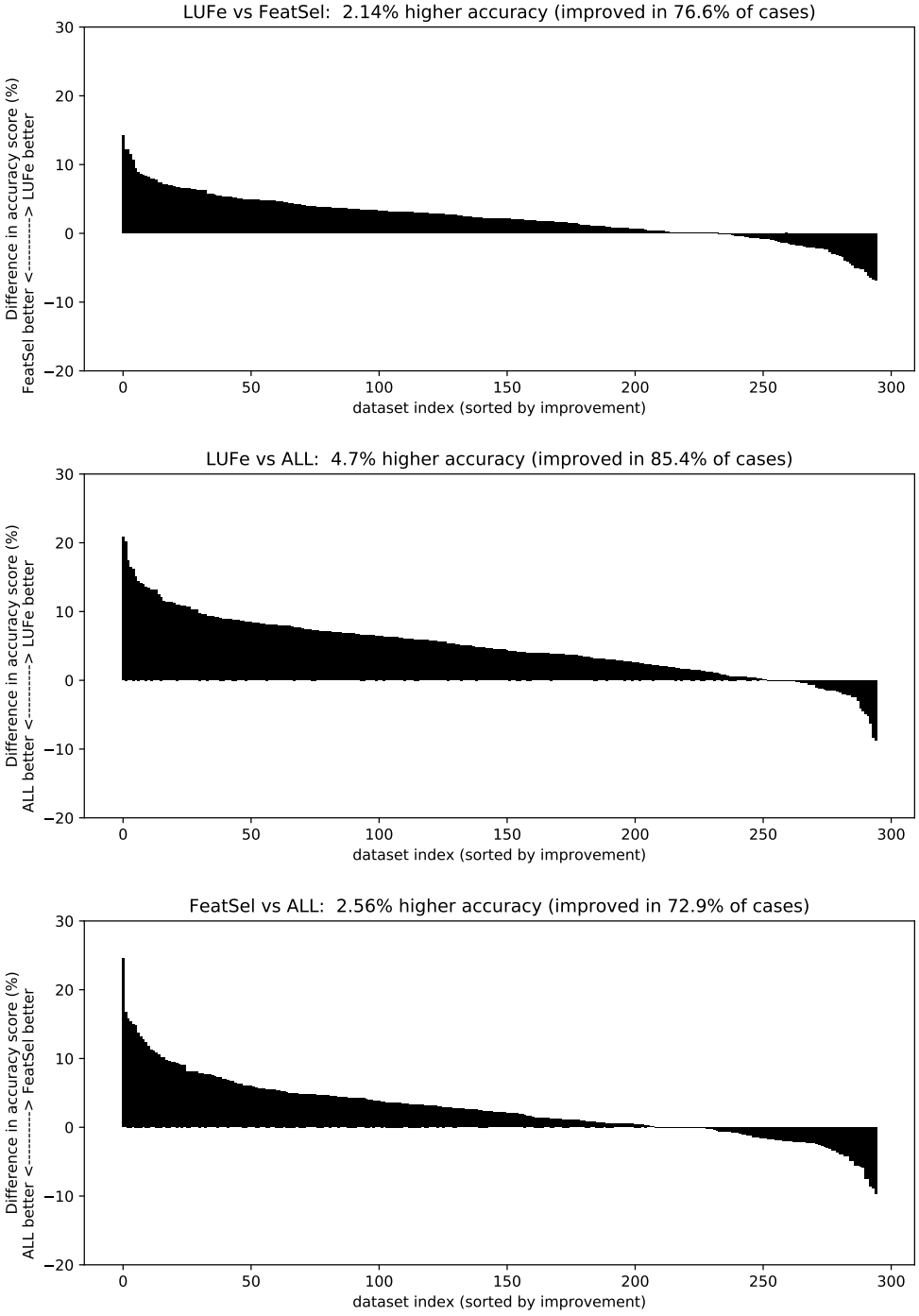

Figure 5.7: Pairwise differences in accuracy scores between *LUFe-BAHSIC-SVM+*, *FeatSel-BAHSIC-SVM*, and *ALL-SVM* settings, for 295 datasets from the TechTC collection (sorted by magnitude of accuracy difference). Top: *LUFe-BAHSIC-SVM+* vs *FeatSel-BAHSIC-SVM*, middle: *LUFe-BAHSIC-SVM+* vs *ALL-SVM*, bottom: *FeatSel-BAHSIC-SVM* vs *ALL-SVM*.

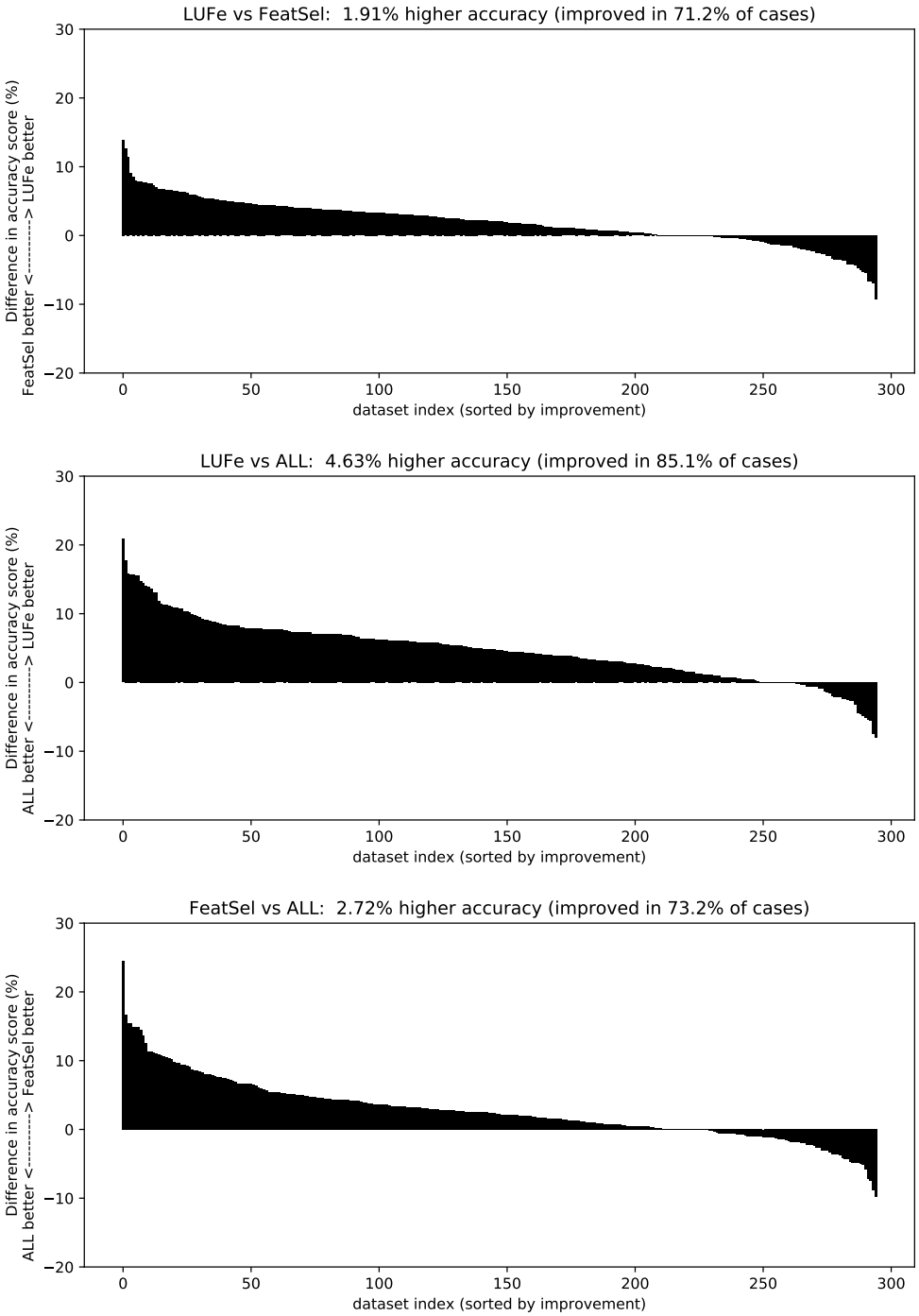

Figure 5.8: Pairwise differences in accuracy scores between *LUFe-CHI2-SVM+*, *FeatSel-CHI2-SVM*, and *ALL-SVM* settings, for 295 datasets from the TechTC collection (sorted by magnitude of accuracy difference). Top: *LUFe-CHI2-SVM+* vs *FeatSel-CHI2-SVM*, middle: *LUFe-CHI2-SVM+* vs *ALL-SVM*, bottom: *FeatSel-CHI2-SVM* vs *ALL-SVM*.

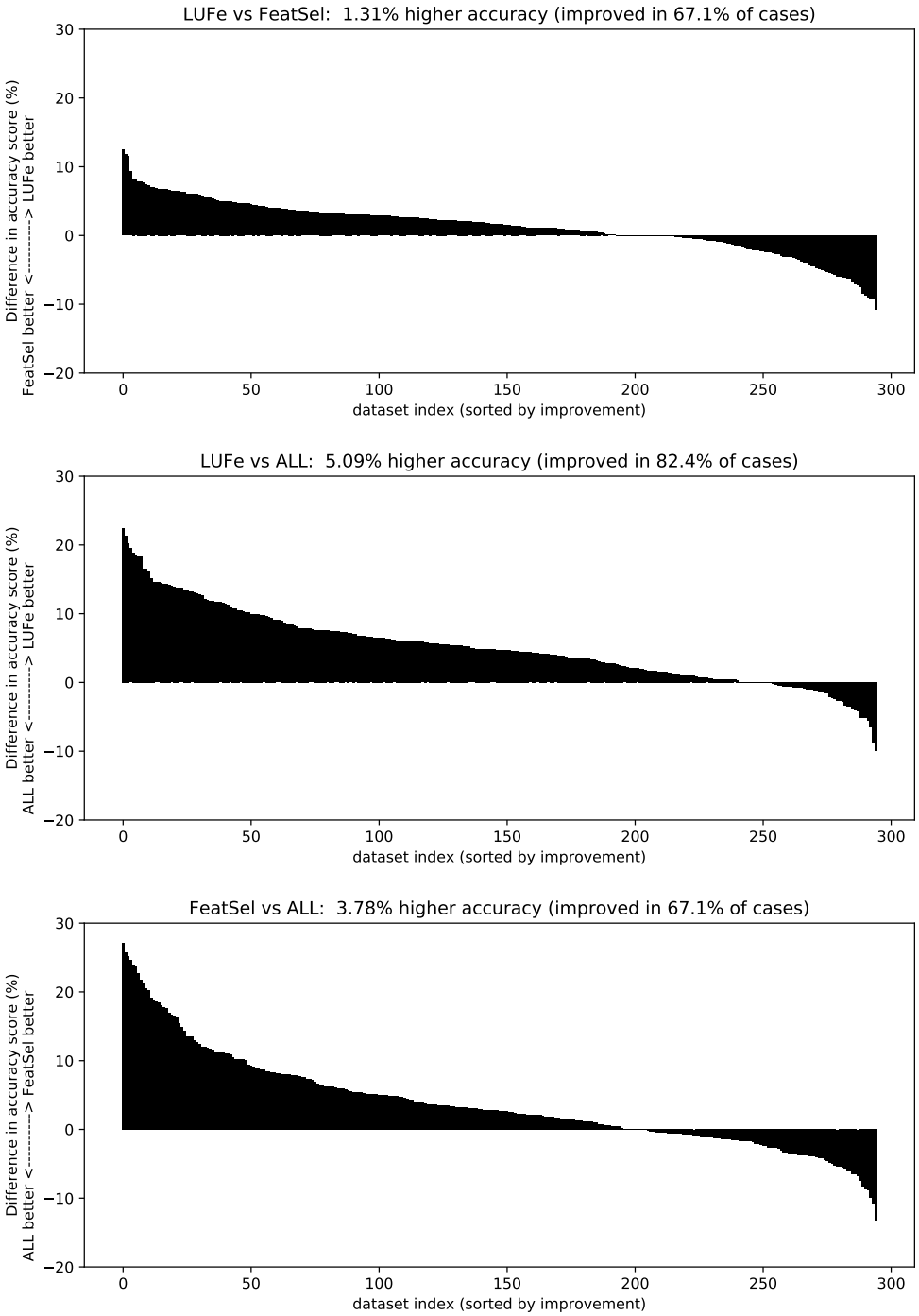

Figure 5.9: Pairwise differences in accuracy scores between *LUFe-MI-SVM+*, *FeatSel-MI-SVM*, and *ALL-SVM* settings, for 295 datasets from the TechTC collection (sorted by magnitude of accuracy difference). Top: *LUFe-MI-SVM+* vs *FeatSel-MI-SVM*, middle: *LUFe-MI-SVM+* vs *ALL-SVM*, bottom: *FeatSel-MI-SVM* vs *ALL-SVM*.

5.1.6 Further experimentation A: Comparison with multi-task learning

A Multi-Task Learning approach to using unselected features (referred to as LUFe-MTL in this work) was described in Section 3.1. LUFe-SVM+ was shown in Section 4.6 to perform at a similar level to this approach at leveraging the features discarded by RFE. Now that LUFe-SVM+ has been shown to also boost performance in conjunction with univariate feature selection methods, it raises the question of whether LUFe-MTL could also be used to gain a similar performance boost. The initial *LUFe-MTL* work by Caruana and de Sa (2003) used a Markov blanket-based feature selection approach and given its demonstrated efficacy in conjuction with RFE, and the fact that LUFe-SVM+ has been shown to be effective with various filter and wrapper feature selection methods, it is reasonable to assume that *LUFe-MTL* will also be applicable to filter feature selection methods.

Experimentation

The experimental procedure was identical to that described in Section 4.6; the *LUFe-MTL* model was trained using a neural network with 3200 hidden units. For a given metric F, input consisted of $\widehat{\mathcal{S}}_F$, and output consisted of label predictions and the top-ranked 300 dimensions of $\widehat{\mathcal{U}}_F$ (where the subscript denotes the dataset was partioned by metric F). Performance of this model was compared with the equivalent *LUFe-SVM+* approach, across the same 295 datasets. Given that the use of a non-linear kernel produced negligible and non-significant improvement for the *LUFe-RFE-SVM+* setting in Section 4.6, the non-linear *LUFe-MTL* models are compared here directly with their corresponding linear *LUFe-SVM+* models.

Results

For all four univariate feature selection methods, both *LUFe-SVM+* and *LUFE-MTL* produced a performance boost compared to standard feature selection. In all cases, the enhancement gained by *LUFE-MTL* was slightly larger, with the difference in mean accuracy between the two LUFe paradigms ranging from 0.03% to 0.61%, shown in Table 5.5. This difference was significant ($p < 0.05$) between the pairs of settings that used ANOVA, BAHSIC and Chi-squared feature selection, but the difference was not statistically significant when RFE and Mutual Information were used.

Table 5.5: Comparison between LUFe-SVM+ and LUFe-MTL methods

	LUFe-SVM+ vs LUFe-MTL			Mean accuracy		
Featsel	SVM+ better	tie	MTL better	SVM+	MTL	MTL improvement
RFE	142	4	149	84.34%	84.58%	0.23%
ANOVA	115	5	175	85.87%	86.43%	0.56%*
BAHSIC	108	5	182	85.92%	86.54%	0.61%*
CHI2	116	5	174	85.86%	86.45%	0.59%*
MI	151	5	139	86.31%	86.34%	0.03%

Discussion

All univariate feature selection methods followed the same pattern as mutual information, where *LUFe-MTL* proved slightly more effective than *LUFe-SVM+* at harnessing unselected features. The difference for all filter methods except for mutual information was larger than that seen with RFE. *LUFe-MI-SVM+* was the best SVM+ implementation of LUFe, and as such, was the closest in accuracy to its corresponding MTL approach.

5.1.7 Further experimentation B: RFE step-size parameter

Contrary to expectations, RFE proved to be the least effective method of feature selection; *FeatSel-RFE-SVM* was the lowest-scoring out of the standard feature selection settings, and *LUFe-RFE-SVM+* was the lowest scoring LUFe setting. As discussed at the beginning of this chapter, one disadvantage of RFE is the sensitivity to the parameter controlling step size: the number of features to be discarded at each iteration. Setting this parameter involves making a trade-off between running time and performance. Initial work in 4 discarded 10% of the feature set at each iteration. Now that we have seen univariate approaches achieve higher accuracy, it is worth considering whether RFE could equal or exceed this performance by means of a more fine-grained procedure with a smaller step-size. If RFE is performing sub-optimally, then it is possible that the LUFe boost in this case is compensating for this by allowing informative but incorrectly-discarded features to contribute to the decision function. This experimentation is to investigate if *LUFe-RFE* still enhances performance if the baseline standard RFE could be improved to the level of univariate methods.

Experimentation

The experimental procedure closely followed that described in Chapter 4, but repeated using a range of logarithmically-spaced step-size parameters: $\{0.1, 0.01, 0.001\}$. Using each value, RFE was performed on the dataset to produce subsets $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$ where $|\widehat{\mathcal{S}}| = 300$. These subsets were used as input for SVM ($\widehat{\mathcal{S}}$ only) and SVM+ ($\widehat{\mathcal{U}}$ and $\widehat{\mathcal{S}}$), to investigate whether varying this parameter resulted in a performance difference in either case.

Results

Results for both settings are depicted in 5.10. As previously stated, for step-size = 0.1, accuracy was 82.6% for standard feature selection, increasing to 84.3% with LUFe. Reducing step-size to 0.01, both improved to 82.7% and 85.1% respectively. Further reducing step-size to 0.001, standard feature selection performance dropped off 81.1% and LUFe slightly decreased to 84.1%.

Discussion

The results demonstrate a somewhat surprising absence of correlation between step-size and error rate. The best LUFe performance was achieved when a medium value of step-size was used. Even this peak result did not attain the same accuracy as the lowest-scoring filter feature selection LUFe method. One possible line of further investigation would be to would further investigate a wider range of step-size parameters. However, the underlying issue persists, that RFE is a greedy algorithm and the challenge of feature selection is not amenable to such a greedy approach.

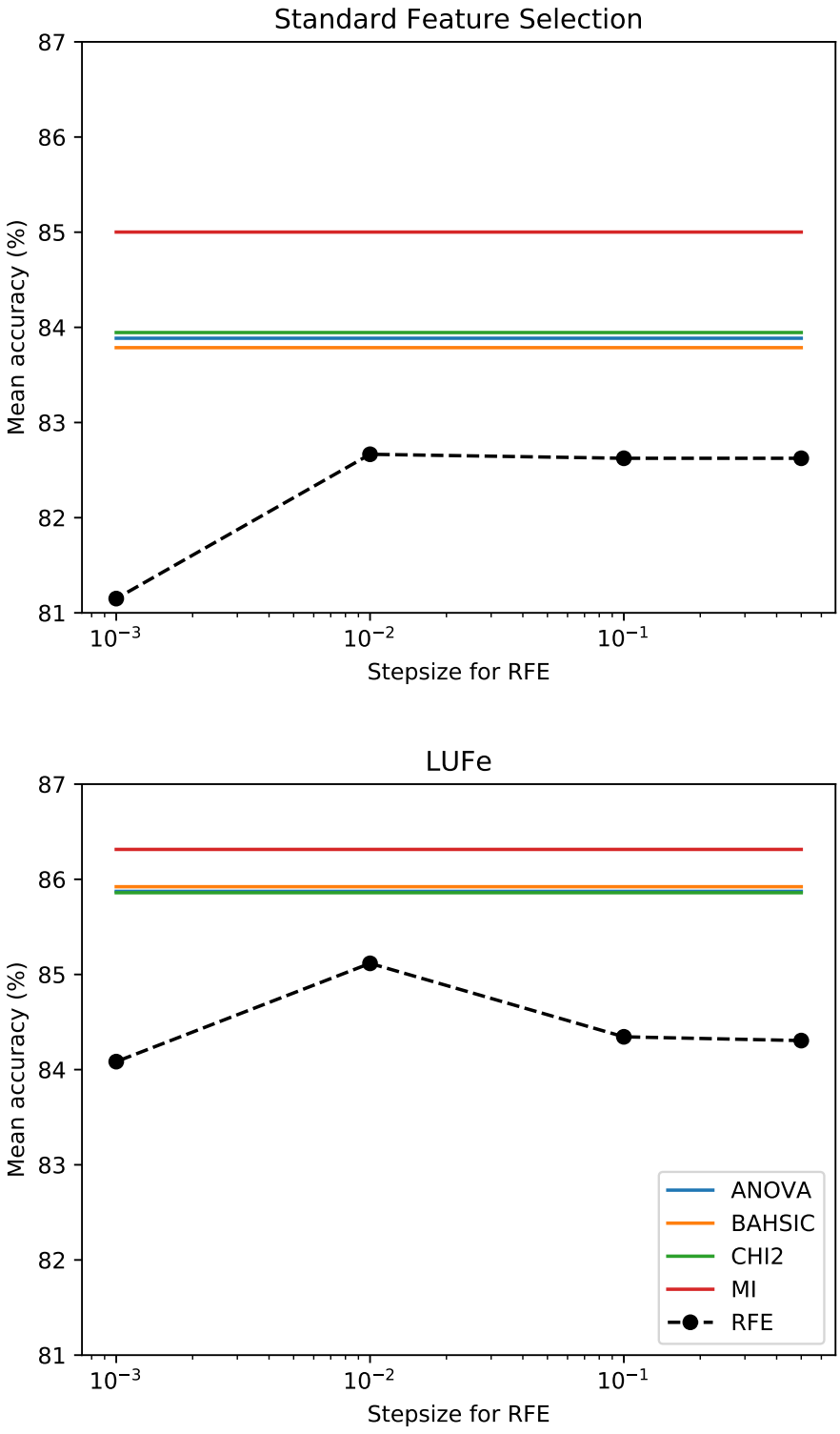

Figure 5.10: Mean accuracy scores of *FeatSel-RFE-SVM* (top) and *LUFe-RFE-SVM+* (bottom) settings, for three different step-size parameters. Also plotted are mean accuracy for ANOVA, BAHSIC, Chi-squared and Mutual Information approaches, which do not have an equivalent parameter.

5.2 Alternative implementations of LUPI

The LUPI paradigm is a general framework that allows an additional feature set describing training data to be incorporated into a learner. Learning using Unselected Features repurposes implementations of this paradigm, by using unselected features as the secondary feature set, instead of the typical highly-informative features. There have been a number of different implementations of the LUPI paradigm since it was first proposed by Vapnik and Vashist (2009); these are summarised in chapter 2.1. The first such classifier, SVM+, was used to instantiate the LUFe framework in 4, but the efficacy of LUFe does not necessarily depend on the use of this particular algorithm. The LUFe model consists of 'plugging in' the two feature subsets obtained by feature selection into a LUPI framework, and the SVM+ implementation that was used in Chapter 4 can be substituted with an alternative implementation — much as the feature selection procedure was 'swapped out' in the previous section.

5.2.1 SVM$_\Delta$+

As described in Section 2.1, the SVM$_\Delta$+ classifier learns a hyperplane in the privileged space that attempts to separate the classes — though this is not a *decision* boundary as it is not used directly for classification. Rather, the 'privileged slacks' ζ_i are learned in this space and used to to inform the corresponding slacks in the standard feature space.

This algorithm therefore assumes that the separability in the privileged space is informative about the separability in the primary space, and that a given data point behaves similarly in both spaces. Data points which are violating the boundary in the privileged space are expected to similarly violate the margins in the primary feature space and to have a large slack value; those which are more easily separable in the privileged space are assumed to also be in the primary feature space, and provided a smaller slack. This contrasts with the SVM+, where class labels are not taken into account in the privileged space (Vapnik and Izmailov, 2015).

Let us now consider how this could be expected to perform in the LUPI and LUFe settings. In the standard LUPI framework, the privileged information is defined as being highly informative about the classification task, and is used only to guide learning at training time due to its limited availability. Therefore, learning a class separation on a set of highly informative privileged features serves as a 'teacher' function, guiding the 'student' to learn a better decision boundary in the primary feature space. It does this by indicating which data points should be expected to be correctly classified, and which are

more acceptable to misclassify.

When used in the LUFe setting, the SVM$_\Delta$+ process translates to learning a class boundary in the space of those features which were deemed by feature selection to be *less* informative about the class separation. It is then apparent that while SVM$_\Delta$+ can improve classifier performance in the standard LUPI paradigm, it may be less suited to usage in LUFe. We have seen that the SVM+ can improve performance when employed for LUFe, but it uses the unselected features without reference to their labels in the privileged space. Taking the labels into account, and using them to place a class boundary in the space of unselected features may be a less effective way of utilising them.

While it is to be expected that the SVM$_\Delta$+ is less suited to LUFe than SVM+, it is possible that it will still improve performance relative to the corresponding standard feature selection approach. We can assume that given the challenge of feature selection is NP-hard, most approaches are sub-optimal, as explained in Chapter 4; therefore there will be classification-relevant information in the unselected features which will be better transferred to the primary feature space by SVM$_\Delta$+.

5.2.2 *d*SVM+

The *d*SVM+ was introduced alongside SVM+ as a way to directly 'provide the slack variables in the simplest form' (Vapnik and Vashist, 2009); this is described in Section 2.1. The *d*SVM+ procedure can be summarised as first training a standard SVM in the privileged space, and then using the privileged slack variables ξ_i as a single-dimensional privileged data for each data point, referred to as a 'deviation value' d_i. This forms triplets (x_i, d_i, y_i) which are used as input to the SVM+, with the primary features using x_i, and d_i taking the role of a single-dimensional privileged information.

When applied to the LUFe paradigm, *d*SVM+ requires a standard SVM to be trained on the unselected feature subset $\widehat{\mathcal{U}}$ that is discarded by feature selection. The slack values learned by this classifier in the $\widehat{\mathcal{U}}$ space for each instance are then taken to be deviation values d_i. The same SVM+ algorithm as used in 4 is then trained, with this one-dimensional d_i as the secondary feature set, with the selected features $\widehat{\mathcal{S}}$ in the primary role as before.

Much like the SVM$_\Delta$+, this approach is contingent on learning a class boundary in the privileged space, and using the slacks learned in this process to transfer information to the primary space and guide the learner. For the same reasons discussed above, this approach may not be transferable from the standard LUPI setting to the LUFe paradigm; the lower informativeness of the secondary featureset means that this data is less separ-

able. Therefore, the original SVM+ approach is likely to be better-suited to harnessing unselected features.

5.2.3 Experimentation with different implementations

LUFe involves feature selection and LUPI techniques. It has been shown in Section 5.1.7 that LUFe is robust to different feature selection approaches, and the experimentation in this section is to determine whether it is equally robust for different feature selection methods. Specifically, this experimentation is to ascertain if the use of LUFe implementations other than SVM+ still provide an additional performance boost beyond standard feature selection, and if so, how that compares with the enhancement that is gained by *LUFe-SVM+*.

Different LUPI classifiers can be 'plugged in' to the LUFe framework, just as the different feature selection algorithms were in the previous section. The feature selection and LUPI methods for LUFe can therefore be 'mixed and matched'. The five feature selection techniques and three LUPI implementations provide a total of 15 combinations. This allows more robust conclusions to be drawn, both in terms of whether feature selection performance is consistent across different classifiers, and the inverse: whether classifier performance is consistent across different feature selection.

5.2.4 Experimental procedure

The testbed of 295 datasets continued to be used, with the same train/test splits, and the same subsets $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$ partitioned for each dataset by each metric. For each metric F, the subsets $\widehat{\mathcal{S}}_{\mathcal{F}}$ and $\widehat{\mathcal{U}}_{\mathcal{F}}$ were used to train two new LUFe classifiers: one using dSVM+ and one using SVM$_\Delta$+. Classifier performance was again assessed in terms of accuracy and compared with the previous settings: *LUFe-SVM+*, *FeatSel-SVM*, and *ALL-SVM*. The LUFe settings are referred to as *LUFe-*{*feature selection method*}*-*{*LUPI method*}; for example: *LUFe-ANOVA-SVM+*, *LUFe-ANOVA-dSVM+*, and *LUFe-ANOVA-SVM*$_\Delta$*+*.

5.2.5 Results

Mean accuracy

Mean accuracy scores across all combinations of LUFe implementations and feature selection methods are summarised in Table 5.6. Relative performance of classifiers (in terms of mean accuracy across all datasets) was very consistent across the different feature selection metrics. LUFe using SVM$_\Delta$+ beat standard feature selection for all five feature selection

techniques. However, for each of the five metrics, SVM+ remained the best-performing classifier and SVM_Δ+ was consistently the second-best classifier.

*d*svm performed the worst of the three LUPI methods. *d*svm accuracy scores were similar to standard feature selection in all four cases where filter selection was used, performing marginally worse for Mutual Information and marginally better in the other three. These differences were not statistically significant. However, when recursive feature elimination was used, *d*svm performed particularly badly, dropping even below the *ALL* baseline.

With this exception, all other fourteen LUFe settings — which used all selected and unselected features in two different roles — performed better than the *ALL* baseline which used all features as a simple input to SVM classifier.

5.2.6 Discussion

The results show that unselected features can be exploited by the LUFe framework even if the SVM+ is not used to instantiate it. The consistent LUFe improvement seen when using SVM_Δ+ demonstrate some robustness to the LUFe paradigm. However, the constant ranking of SVM+ ahead of SVM_Δ+ suggests that SVM+ is the better-suited algorithm for this setting. This can be expected, as the SVM_Δ+ uses class information in the secondary feature space $\mathcal{U}$, which has been designated less useful for classification. The better performance of SVM+ can be explained by the fact it does not use labels in the secondary feature space.

The *d*SVM+ approach was less successful at harnessing unsuccessful features. When compared to the standard SVM+ approach, which simply uses $\widehat{\mathcal{U}}$ directly as the secondary feature set, the *d*SVM+ involves an additional classification step. In the LUFe setting, this is performed in the $\widehat{\mathcal{U}}$ feature space which has been denoted as less informative, so the poor performance of *d*svm is unsurprising. The particularly low performance when *d*SVM+ was combined with RFE makes sense, given that RFE was the worst standard feature selection method. The *d*SVM+ performance suggests that LUFe does depend on information in the unselected features, rather than the constraints simply performing additional regularisation.

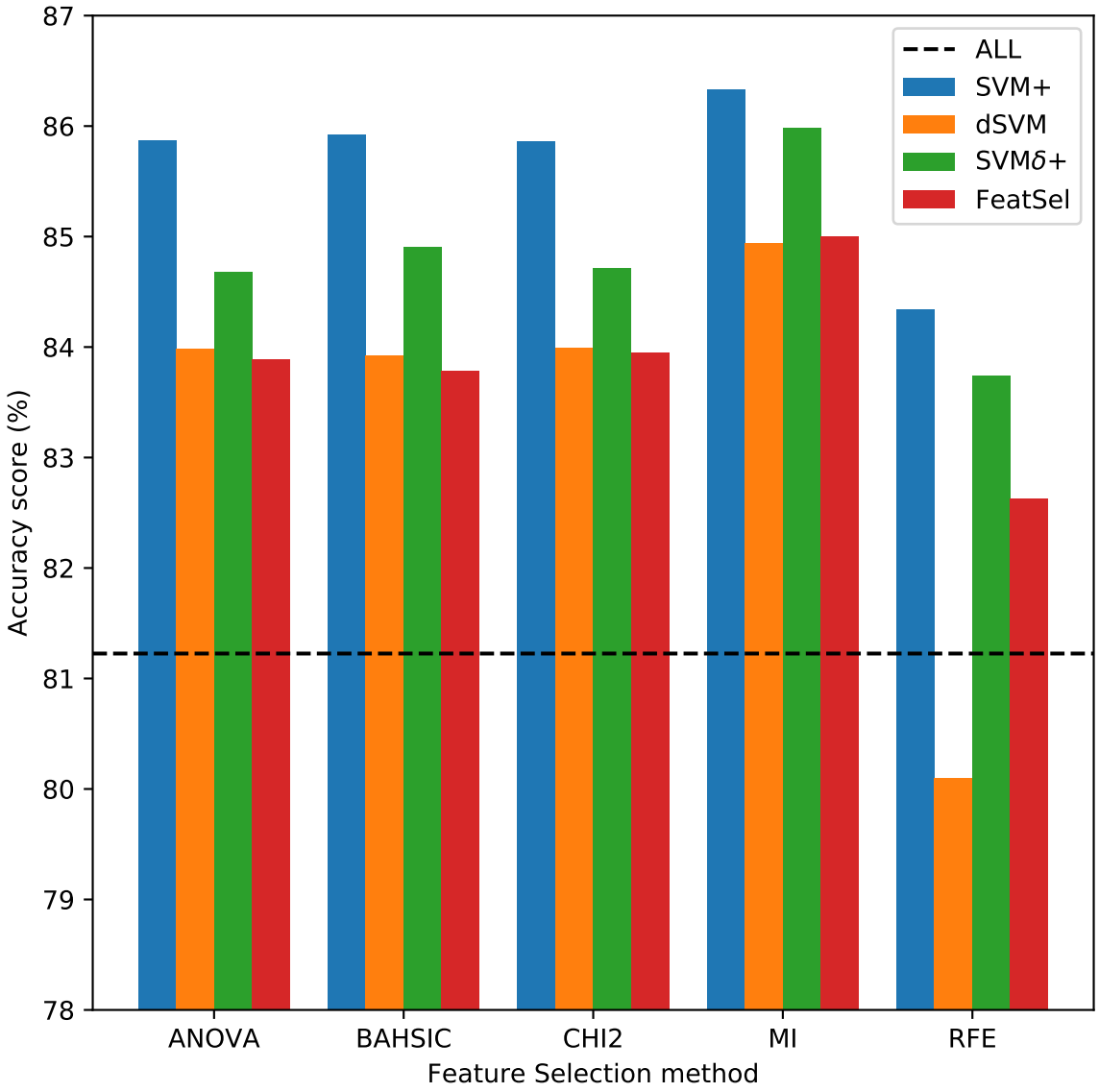

Figure 5.11: Mean accuracy scores across 295 datasets. Results shown for three different implementations of LUFe, plus standard feature selection, for five different feature selection methods. *ALL-SVM* setting with no feature selection is shown as dotted line.

Table 5.6: **Results Summary: Various Feature Selection and LUFe methods** Mean accuracy scores for *LUFe* classifiers across 295 datasets, using five different methods of feature selection, and three different implementations of LUFe. *FeatSel* score for various feature selection methods also shown for comparison; significant improvements over this are marked with * and significant decrease marked with ˆ. Maximum per row shown in bold.

Feature selection	No LUFe	LUFe implementation		
	SVM	SVM+	*d*SVM+	SVM_Δ+
RFE	82.6%	**84.3%***	80.1%ˆ	83.74%*
ANOVA	83.9%	**85.9%***	84.0%	84.7%*
BAHSIC	83.8%	**85.9%***	83.9%	84.9%*
CHI2	84.0%	**85.9%***	84.00%	84.7%*
MI	85.0%	**86.3%***	84.9%	86.0%*

5.3 Using subsets of unselected features

The exploration of LUFe thus far has demonstrated the consistency of its performance boost, across different feature selection metrics and different implementations of LUPI. In all cases, the LUFe setting has used the top k features as the primary feature set, and the remaining $d - k$ features as the secondary set.

In much the same way that feature elimination in the standard feature domain can improve classifier performance, discarding some unselected features may also allow a more generalisable model to be constructed, thereby improving classification performance. The performance enhancement gained through Learning using Unselected Features may be greater if some selectivity was applied to these unselected features that are used. The *LUFe-MTL* approach has been shown to achieve similar improvements to *LUFe-SVM+* even though it uses only a subset of the top k features from $\widehat{\mathcal{U}}$. By taking only a subset of the $d - k$ unselected features to use as the secondary feature set, LUFe becomes a three-way partition of the dataset into the following: (a) the top k features used as primary dataset, as before, (b) the t best unselected features, used as a secondary dataset and (c) the remaining $d - k - t$ features which are neither used in primary or secondary feature set, and discarded as in standard feature selection.

How the 'best' features should be defined is a further question to be considered. Referring back to the SVM+ objective function, we see that the quality functional on unselected

features is used to upper-bound the loss on the selected subset. Recall that when feature selection has produced a much better subset $\mathcal{S}$ than $\mathcal{U}$, the quality score for $\mathcal{Q}^i(\mathcal{U})$ for a given datapoint x_i tends to be lower, producing a tighter bound on error; conversely, if subset $\mathcal{S}$ is not much better than than $\mathcal{U}$, the bound is looser.

If only a further subset of the top t from $(\mathcal{U})$ were to be used, the quality functional would be higher, and this would result in a looser lower bound. This would seem to imply that, counter-intuitively, we should use the *worst* unselected features in order to get a lower-scoring functional and thus secure a tighter lower-bound. However, the data-dependent margin is calculated for each data point. Using a tighter lower-bound on all data points in this manner reduces the contribution of all points in the objective function.

A further motivation for using a subset of unselected features is the reduction in running time at training. The LUFe setting is designed for the scenario of greater computational resources at train time. However, if equal or better performance can be achieved by using only a subset of unselected features, which reduces training time, then the more efficient approach would be preferable.

To empirically investigate the effect of the size and quality of the secondary feature set, the next phase of experimentation used subsets of the unselected features. This intended to address the following questions: does using only a subset of unselected features further boost performance, compared to using the entire subset? If so, does using the best or worst unselected features help more, and what proportion of unselected features is the most beneficial to use?

5.3.1 Methodology

For this experimentation, mutual information and RFE were used as representative filter and wrapper feature selection metrics, respectively. Due to the time constraints involved in training multiple classifiers, no other feature selection metrics were investigated; MI was chosen as the highest-scoring univariate approach in previous experimentation.

Unselected features were ranked using the output from the initial feature selection procedure. Univariate feature selection approaches score each attribute in order to select the top-scoring k features, so the same ranking can be simply used to select the next-best scoring t features from the remaining unselected subset. For wrapper methods, it is less simple. RFE iteratively eliminates attributes with the lowest coefficients. Therefore the top t unselected features were chosen as those attributes which had the highest coefficients among the features that were not selected the final iteration. As discussed previously, RFE

as a greedy algorithm will likely perform sub-optimally and its sensitivity to stepsize may worsen this.

For each metric, eight *LUFe-SVM+* classifiers were trained. In each case, the same top 300 features continued to be used as the primary feature set, but only a subset of the unselected features were used as the secondary feature set. This consisted of either the top t% unselected features, or the bottom b% unselected features, with both t and b parameters varied across the set: {10, 25, 50, 75, 100}. Setting hyperparameters to $t = 100$ or $b = 100$ are equivalent, with the entire unselected feature set being used, and the setting reverts back to the standard *LUFe-SVM+* used previously.

5.3.2 Results

Results are displayed in Table 5.7 and Figure 5.12. For both feature selection methods, LUFe with only the bottom 10% of unselected features was detrimental, resulting in accuracy lower than the standard feature selection approach. However, all other LUFe settings using a subset of top t or bottom b features improved upon standard feature selection.

Different patterns of results were observed for the two feature selection approaches. When using top t% features selected by mutual information, there was no significant difference in performance across different t; performance was consistently between 86.2% accuracy (achieved when 25% used) and 86.5% accuracy (achieved when 10% used). This contrasted with using various amounts of bottom b% features. Here, there was a significant difference between using 10%, and using 50% or more.

There was a similar consistency among results for top t features when these were instead selected using RFE. Performance again was within a 0.3% range and not significantly different. However, using bottom b features according to RFE was a lot less ordered; using only the worst 10% was again the worst-performing setting, at 82.1%, and increased as features were added, up to 84.8% when 75% used. However, once 100% of features were used, performance dropped off slightly to 84.3%.

5.3.3 Discussion

The trends of results seen in using subsections of unselected features according to mutual information and RFE can be explained as follows: LUFe performance boost is partially dependent on having unselected features that are informative about the classification, and partly dependent on the amount of data provided. The deterioration seen in both settings where $b = 10$% demonstrates that learning using unselected features is detrimental if both

the quality and quantity of the secondary dataset is lacking.

However, the highest score seen in all experimentation so far was achieved by using only the best 10% of the features which were not selected by mutual information. This indicates that informative unselected features have a positive impact on performance; in this case, the quality of the top $t = 10\%$ of features more than made up for the smaller number. However, there was little adverse effect from using additional, 'worse' features as well by increasing t. Accuracy score only became very marginally worse — to no significant degree — as further unselected features, of less informativeness, were progressively added. Conversely, using only the 10% of features deemed *least* informative by mutual information performed the worst among all *MI-LUFe* settings — again, suggesting that classification-relevant information is crucial to maximising the gains due to LUFe. As more of the bottom-ranked features were added, performance improved. Performance stabilised and was consistent when 50% to 100% of unselected features were used. This suggests that the inverse of the previous statement is also true: quantity of unselected features can make up for lack of quality.

The results from RFE point to similar conclusions. Given that RFE was the worst standard feature selection metric, in Section 5.1.7, its performance is sub-optimal. In contrast with MI, the top 10% of RFE features did not provide the biggest boost; this is likely because of the sub-optimal selection which did not select the next-best 10% as effectively as MI. Performance improved when more unselected features were used — suggesting that either more informative features were now being used, or that the quantity made up for the lack of quality; in either case, this supports the idea that classification-relevant information is necessary to maximise the improvement due to LUFe.

When only the worst 10% of features according to RFE were used, performance was worse even than standard feature selection — repeating the trend seen when MI was used. The features used in this case are a subset of those rejected in the first iteration of RFE, so can be safely assumed to be uninformative; lending further weight to the idea that unselected features need to be informative for maximal effectiveness of LUFe. As with MI, performance improved as more were added — but then worsened when the final 25% of features were included. This could be attributed to the performance of the RFE, with less relevant features being included in the final 25%, again suggesting the importance of classification-relevant information to the LUFe boost.

Finally, comparison between using the top t% of features and the bottom b% of features underlines the impact of using informative features. *LUFe-RFE-SVM+*, scored 84% when

Table 5.7: **Results Summary: Varying Amounts of Unselected Features**
Mean accuracy scores for *LUFe-RFE-SVM+* and *LUFe-MI-SVM+* classifiers across 295 datasets, in response to changing the percentage (top t% or bottom b%) of unselected information used.

	Top features used		Bottom features used	
% of unselected	RFE	MI	RFE	MI
10%	84.0%	86.5%	82.1%	84.9%
25%	84.2%	86.2%	84.3%	85.2%
50%	84.2%	86.4%	84.7%	86.3%
75%	84.3%	86.3%	84.8%	86.3%
100%	84.3%	86.3%	84.3%	86.3%

using top 10% features, and 82.1% when using the bottom 10% features; a significant difference. Similarly, there was a significant difference between *LUFe-MI-SVM+* with top 10% (84.9%) and bottom 10% (86.5%). These differences can only be attributed to the quality of the secondary dataset in each case.

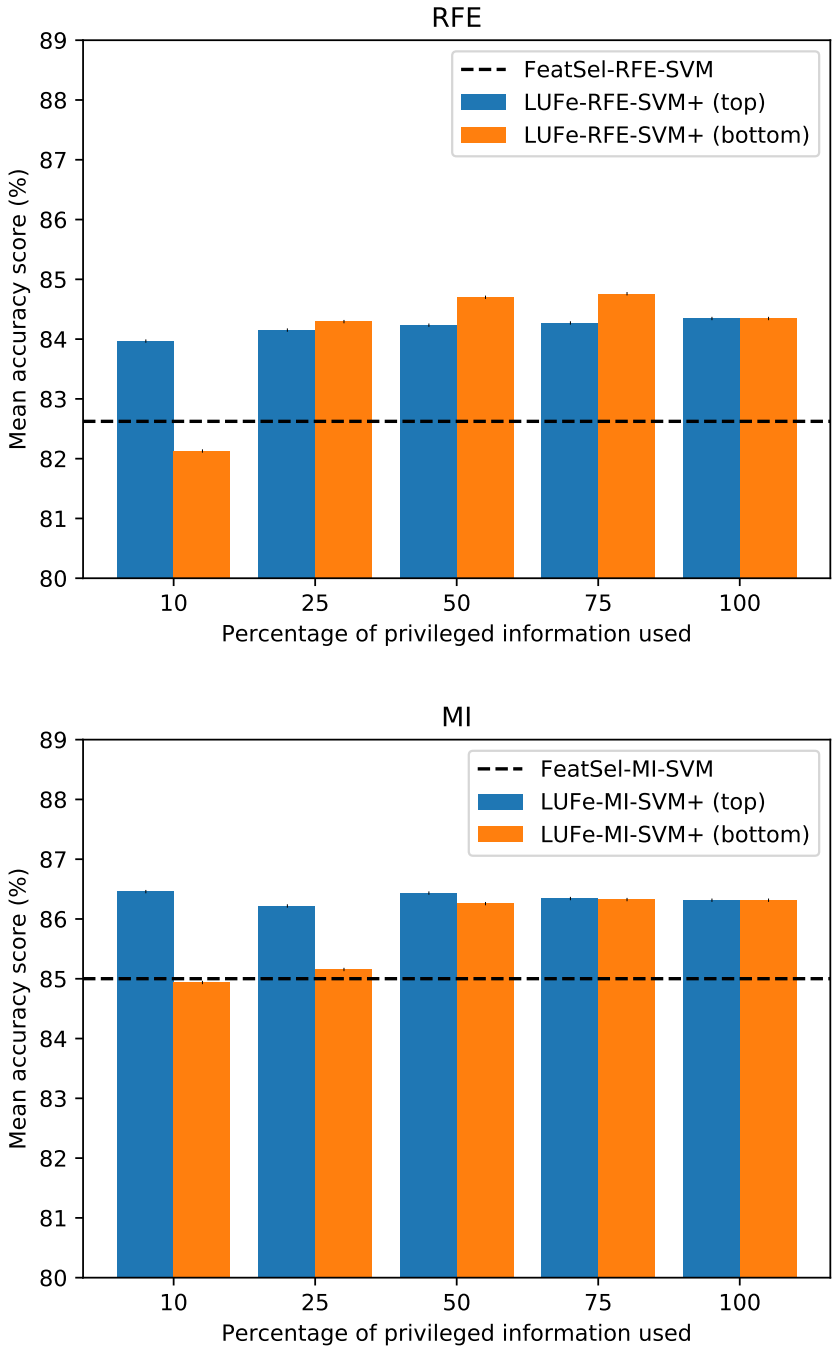

Figure 5.12: Mean accuracy scores for *LUFe-RFE-SVM+* and *LUFe-MI-SVM+* classifiers across 295 datasets, in response to changing the percentage of privileged information used. The corresponding SVM classifiers using no standard feature selection are shown as dotted lines.

Chapter 6

Investigating the mechanism of action for LUFe

The work in Chapters 4 and 5 has shown LUFe to be an effective method of improving accuracy in classification problems, in comparison with standard feature selection. Chapter 4 provided a proof-of-concept for LUFe, and Chapter 5 demonstrated the continued efficacy of LUFe in a wider range of feature selection methods and LUPI methods.

Besides this direct functionality of LUFe, the success of this paradigm has wider implications for the broader LUPI framework. The LUFe setting employs the LUPI framework but violates some theoretical points that underpin it. Namely: the assumption that the secondary feature set is more informative than the first (Vapnik and Vashist, 2009), and the assumption that the capacity of the 'teacher function' learned on the secondary set is smaller than that of the 'student function' (Vapnik and Izmailov, 2015).

So already, Learning using Unselected Features has extended the application for the LUPI framework, and provided further evidence for its practical efficacy, while simultaneously challenging the justification for this paradigm. Only a small amount of prior research has questioned the mechanism by which LUPI improves classification, compared to classical single-domain methods (Serra-Toro et al., 2014); LUFe continues this avenue of exploration.

The purpose of this section then, is to establish **how and why the LUFe performance boost is effected**. This was approached from an empirical angle, with experimentation designed to answer the following questions, which follow from each other:

1. **Is LUFe dependent on using *Unselected* Features?**

   If so,

2. **What properties of this secondary feature set make it beneficial?**

   Given this,

3. **How does the LUFe setting improve performance?**

Each of these will be addressed in the sections of this chapter.

6.1 Is LUFe dependent on Unselected Features?

It is important to consider that the performance enhancement may simply result from the usage of additional constraints on the objective function, and that this effect may not depend on actually utilising the *information* contained in the unselected features.

Work by Serra-Toro et al. (2014) investigated a similar question in reference to the LUPI paradigm. The authors note that 'more research is required both in the theoretical and practical sides of LUPI for a better understanding of its nature as well as its possibilities and limitations'. This work questioned whether the provision of random features in place of 'meaningful a priori' privileged information could still help performance. Three settings were used, each with equal amounts of 'privileged' information: one genuine privileged information, and two with random features: one class separable, the other non-separable, and performance was similar across the three; the random settings achieved similar performance in some tasks. This finding sets a precedent for the following experimentation.

6.1.1 Experimentation

Experimentation was designed to establish the importance of the unselected features themselves for the efficacy of LUFe. In order to investigate whether the LUFe benefit can be attributed to the inclusion of classification-relevant information held by the unselected features, a number of different experimental settings were investigated, that imitated the LUFe paradigm but without actual unselected features. This allows us to isolate the impact of the secondary feature set.

For each feature selection metric, two new experimental settings were produced, that utilise the same algorithms and primary feature sets as the LUFe settings encountered so far, but with the true unselected features swapped out for alternative secondary feature sets of the same dimensionality. [1]

[1] These will be described as 'LUFe' settings, despite not actually using *unselected* features. This is to emphasise the algorithmic similarity to 'genuine' LUFe

Random 'unselected' features

The first setting will be referred to as *LUFe-Random-SVM+*. This used attributes generated from random distributions, each with 0 mean and unit standard deviation. This is consistent with the original unselected feature set (as well as the selected set) which were normalised to have 0 mean and unit standard deviation during pre-processing. For each dataset with d original features, $d - 300$ random features were generated, producing a secondary dataset of equal dimensionality to the 'real' unselected features.

Shuffled unselected features

In addition to this, a further setting referred to as *LUFe-Shuffle-SVM+* used the same secondary feature set as standard LUFe, consisting of the unselected features. However, the order of instances for the secondary feature set is randomly shuffled. This replaces training triplets

$$(x_i, y_i, x_i^*) \quad \forall i \in 1...N \tag{6.1}$$

with new training triplets

$$(x_i, y_i, x_j^*), \quad i \neq j \quad \forall i \in 1...N \tag{6.2}$$

The primary feature set X and labels y continue to correspond to the same ordered training instances and labels; this is unchanged from standard LUFe. However, the secondary feature set X^* is now mismatched with X and y. Each instance of selected features, x_i, has a secondary feature set that consists of the unselected features from a different instance, x_j^*. The LUFe objective function usually uses the secondary feature space to provide a data-dependent upper bound on the error in the primary feature space. In this setting, the upper bound on error for a given instance x_i is instead based on the secondary feature set of a different instance, x_j^*.

This means that the error upper bounds are still 'data-dependent' in a looser sense — they depend on the secondary feature set, but the bound on error for a particular instance does not depend on its corresponding representation in that space. This means that the distribution of errors permitted by the classifier is the same as in standard LUFe, allowing some information to be transmitted from the secondary space about the dataset as a whole. However, this occurs without transmission of individual instance-by-instance information from the unselected feature space to the space of selected features.

The*LUFe-Shuffle* setting can therefore be seen as a midpoint between the standard *LUFe* and the *LUFe-Random* settings. Like standard LUFe, it uses unselected features

to inform the boundary that is learned in the selected feature space. However, like *LUFe-Random* — and differing from standard LUFe — it does not use the unselected features of a single instance to inform the error bound on the selected features of that same instance.

Discussion of experimental settings

The performance of *LUFe-Shuffle* and *LUFe-Random* settings allows insight about the factors that produce an improvement with LUFe. If *LUFe-Random* performs better than standard feature selection, this suggests that the benefit of LUFe can be partially attributed simply to the presence of additional constraints imposed by the classifier. If performance matches standard LUFe, this would suggest that 'Learning using "Unselected" Features' does not in fact require the unselected features, and would be entirely attributable to the presence of additional constraints.

Similarly, if *LUFe-Shuffle* performs better than *LUFe-Random*, this will indicate that there is some benefit to the transfer of information from the space of unselected features to the selected feature space. If the LUFe boost is indeed dependent on genuine data, then comparison of *LUFe-Shuffle* to standard LUFe can gain insight about this how this data-dependent boost is effected. If *LUFe* outperforms *LUFe-Shuffle*, this supports the original hypothesis that the performance is helped by bounding the error for an instance, based on that instance's unselected features. If there is no significant difference between LUFe and *LUFe-Shuffle*, but they are both significantly better than LUFe-Random, then this would suggest that the improvement can be attributed to some general transference of information from the unselected feature space, which is common to *LUFe* and *LUFe-Shuffle*.

6.1.2 Results

Consistent patterns of results occurred across all five feature selection metrics, when comparing standard *LUFe*, *LUFe-Shuffle*, *LUFe-Random* and *FeatSel* settings in terms of accuracy. This is shown in Table 6.1 and Figure 6.1. *LUFe-Shuffle* mean accuracy ranged from 83.61% (with RFE feature selection) to 85.26% (with mutual information). *LUFe-Random* mean accuracy ranged from 83.27% (with RFE feature selection) to 84.71% (with mutual information). This compares with a range of 84.34% to 86.31% in the 'genuine' LUFe approach; the new settings *LUFe-Shuffle* and *LUFe-Random* were outperformed by genuine LUFe for all feature selection metrics. *LUFe-Shuffle* was the second-best method in all five cases, and *LUFe-Random* was the worst of the three LUFe settings for all feature

Table 6.1: **Summary Results**

Mean accuracy scores across 295 datasets for LUFe, LUFe-Shuffle and LUFe-Random settings, using five different feature selection methods

Feature Selection	LUFe	LUFe-Shuffle	LUFe-Random
ANOVA	85.9%	84.9%	84.3%
BAHSIC	85.9%	84.8%	84.2%
CHI2	85.9%	84.9%	84.3%
MI	86.3%	85.3%	84.7%
RFE	84.3%	83.6%	83.3%

selection methods.

When comparing with standard feature selection, *LUFe-Shuffle* was better across all five feature selection metrics. *LUFe-Random* was better than *FeatSel* for four out of five feature selection metrics, but when used in combination with mutual information, *LUFe-Random* actually worsened performance.

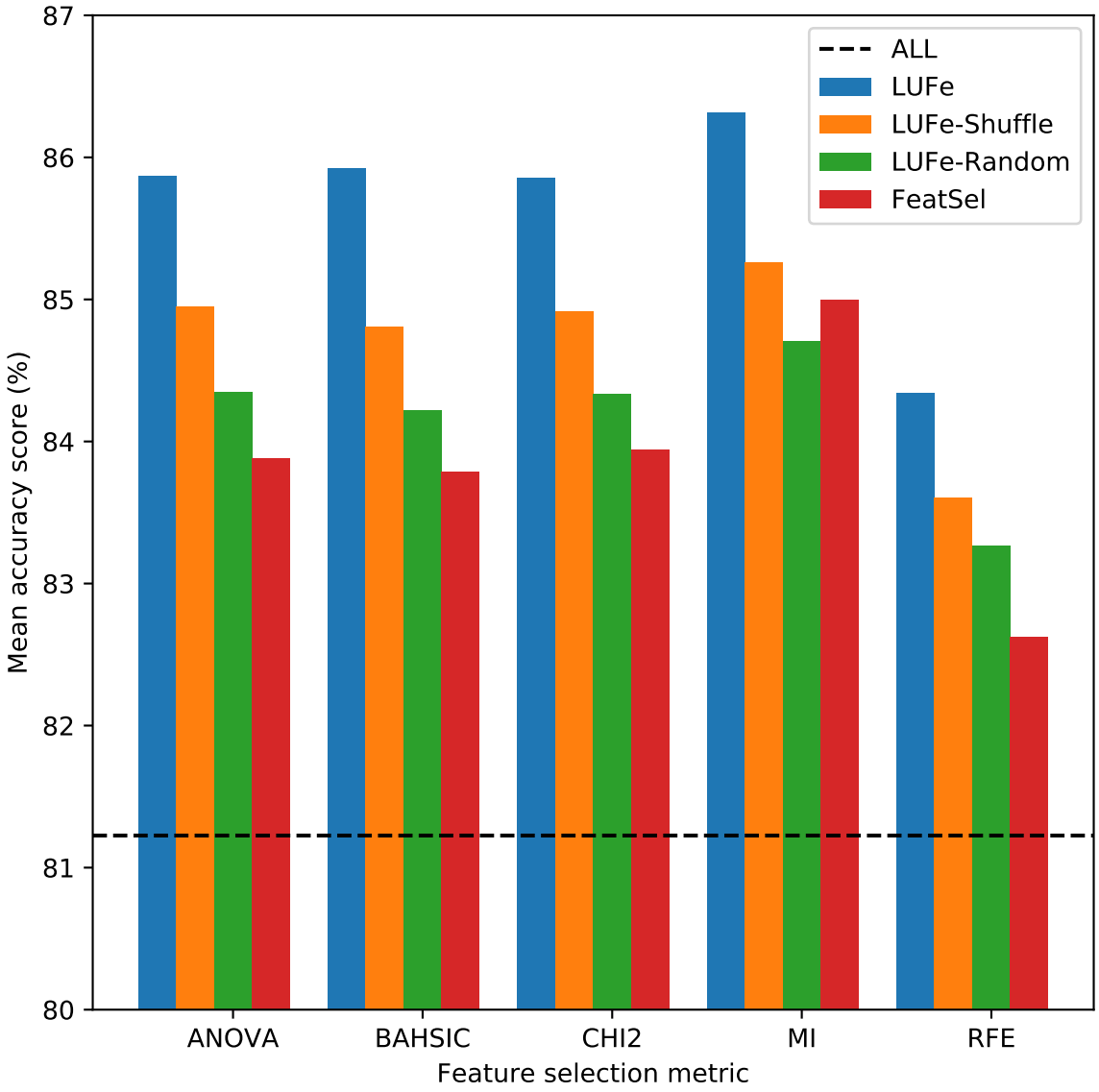

Figure 6.1: Comparison of accuracy scores for *LUFe*, *LUFe-Shuffle* and *LUFe-Random* classifiers across 295 datasets, using five different feature selection methods. The score for each standard feature selection method is also plotted, and the baseline with no feature selection is shown as a dotted line.

6.1.3 Discussion

The results display a clear hierarchy of LUFe methods, with "genuine" *LUFe* at the top, followed by *LUFe-Shuffle*, and then *LUFe-Random*. The first observation to be made from this result is that it strongly points to the data-dependence of the boost due to LUFe. In other words, LUFe works best when using real $\{x_i, y_i, x_i^*\}$ triplets; performance drops off if x_i^* is shuffled so that training triplets are 'mismatched', and it declines further when it is substituted with random data which does not originate in the original dataset. The superiority of both settings that use the real unselected features indicate that LUFe does indeed benefit partly through the transfer of information about the training data which is contained in the secondary feature set. Furthermore, the superiority of standard *LUFe* to *LUFe-Shuffle* shows that this performance is better when information transfer occurs between corresponding selected and unselected features from the same instance. This supports the hypothesis that the effect is mediated by a data-dependent upper bound on error, with a single instance x_i being provided with an upper-bound on error based on the unselected features x_i^* from that same instance.

In *LUFe-Shuffle* and *LUFe-Random*, the additional constraints of the SVM+ classifier continue to provide the upper bounds for the error in the primary feature space, but they are not learned using the corresponding unselected features. However, the provision of these parameters still improves conversion rate to an optimal solution. As discussed in Chapter 2.1, by providing slack variables, N fewer parameters need to be learned. These results show that even the provision of sub-optimal bounds allows the model to learn better overall, by requiring fewer parameters to be learned from a given amount of train data.

The second key observation, complementary to the first, is that the LUFe improvement is not *entirely* attributable to the use of genuine, paired, selected features. The improvement by LUFe-Random over four out of five standard feature selection approaches shows that the extra constraints of the SVM+ classifier can improve performance relative to an equivalent SVM model trained on just the primary feature set, even if its secondary feature set is meaningless artificially-generated data. Even using the 'wrong' error bounds still constrains the function space $\mathcal{F}$ which searched for the classification model. This supports the evidence from Serra-Toro et al. (2014) that randomly generated features can perform similarly to meaningful a priori privileged information in some problems. The relative success of *LUFe-Shuffle* demonstrates that some informative generalities of the secondary feature set may be conveyed through the LUFe setting. For example, the error

bounds used in *LUFe-Shuffle* are calculated using the same set of secondary features as in standard LUFe, so are drawn from the same distribution.

To examine the circumstances under which LUFe settings improve performance, we can observe the results from the settings which use mutual information as a feature selection metric. MI was the only standard feature selection technique to outperform its corresponding *LUFe-Random* setting. The smallest relative improvements by *LUFe-Shuffle* and standard LUFe were also observed in combination with mutual information, which was the highest-scoring standard feature selection method. This further demonstrates the 'ceiling effect' where a standard feature selection method that produces a high-performance classifier is harder to improve upon by LUFe methods. The extreme case of this is the *LUFe-Random* setting, which was consistently the worst LUFe method, and so was not even able to achieve equal performance as standard MI feature selection. The benefit of being provided slacks and having fewer parameters to estimate was insufficient to counter the inaccuracy of the slacks, as the standard SVM classifier already did a good job of estimating.

Let us now consider these results in terms of the 'teacher function' and 'student function, described in the literature for the original LUPI paradigm. Originally, this described a highly-informative function learned on a secondary feature set, analogous to one expert human teaching another. This experimentation has demonstrated that a 'confused' teacher, in the case of *LUFe-Shuffle* — or even an 'unqualified' teacher, for *LUFe-Random* — can help the student by giving some values, and leaving the student with less to worry about.

This work has illustrated that the inclusion of extra unselected information can benefit LUFe, but not fully explain its effects. Therefore, the following section looks into what causes some datasets to benefit from LUFe more than others.

6.1.4 Further experimentation: Random feature selection

Until now, the work has been based around comparison between three types of setting: the all-features baseline, a feature selection method which tends to improve upon this, and a LUFe setting which tends to improve further. This leaves an open question of how LUFe performs in comparison with feature selection which is *not* better than the baseline.

Two further experimental settings were considered to address this question, by using a random approach to feature selection, in contrast with the filter and wrapper style methods seen thus far. In both cases, a subset of $k = 300$ features was randomly selected to be

the primary feature set, instead of using an informed feature selection approach. The first setting, *FeatSel-RandomSelection* trained an SVM on this subset, and the second setting, *LUFe-RandomSelection* trained an SVM+ classifier, using this same subset as the primary feature set, and the remaining $d - k$ features as the secondary feature set.

Experimentation

The standard experimental procedure was followed, averaging results over 10-fold cross-validation and 295 datasets, with parameter estimation performed using the same range of hyperparameters.

Results and Discussion

FeatSel-RandomSelection achieved 54.4% accuracy and *LUFe-RandomSelection* achieved 54.5% accuracy — both around the level of random decision making. The first comparison to be made is that both these *RandomSelection* settings performed massively worse than even the *ALL* baseline (81.2%), and therefore worse than all *FeatSel* and *LUFe* settings which used principled feature selection, where results were in the range 82.6% - 85% and 84.3% - 86.3% respectively (depending on feature selection method). This is unsurprising; the improvement seen by standard, principled feature selection indicates that the full feature sets have irrelevant attributes, without which the model performs better. Alternatively, a large amount of classification-relevant information is contained within a relatively small subset of features, so taking a random subset of the entire feature set, is unlikely to contain informative features.

The second comparison is between the two *RandomSelection* settings; in contrast with pairs of classifier that used principled feature selection, there was no significant improvement by LUFe — even though there was ample room for improvement due to the low score by *LUFe-RandomSelection*. This is interesting for two reasons. Firstly, it was observed in 4.3.5 that LUFe is not simply a brute-force approach of throwing more information at the problem; this is again demonstrated, as it does much worse than the 'brute force' *ALL* baseline. Secondly, the low score by *FeatSel-RandomSelection* also further demonstrates that LUFe performance is strongly dependent on the underlying feature selection.

6.2 Assessing the value of unselected features

We have seen in the preceding section that gaining the full benefit of LUFe appears contingent on using additional information about the training data, contained in the unselected features. In other words, the secondary feature set itself influences how the LUFe setting performs. Experimentation in this section investigates which characteristics of a dataset make it likely that including its unselected subset via LUFe will be beneficial. It also seeks to define a metric to judge the value of LUFe on a given dataset. Specifically, this section investigates whether the informativeness of either the primary or secondary subset — about each other, or about the labels — can predict the usefulness of LUFe.

6.2.1 Informativeness between feature sets

True unselected features, correctly matched with the corresponding selected feature set, were shown to be more beneficial than shuffled or random features. Therefore, one candidate metric to predict the usefulness of using unselected features concerns the amount of information shared *between* $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$. In their review of multi-view (MV) learning, Xu et al. (2013) describe two unifying concepts that underlie different MV techniques: the *consensus principle*, which maximises the mutual agreement between views, and the *complementary principle*, which states that multiple views contain different and complementary information to each other. As discussed in 3, LUPI may be considered a special case of multi-view learning, with a secondary "view" that is available at train-time only. LUFe may also be defined in such a way, with the unselected features supplying a secondary train-time view, so we can apply these same principles from MV learning when considering the circumstances where learning where using unselected features is beneficial.

It is possible that, in accordance with the consensus principle, LUFe will perform better when the unselected features are informative about the selected features. It was shown in 6.1 that "LUFe" using random information — which is correlated to neither labels nor $\widehat{\mathcal{S}}$ — performs worse than using genuine unselected features. This supports the principle that LUFe will perform better when using subsets which 'agree' and describe similar distributions of labelled training data. Given that algorithms such as SVM+ make the assumption that a given datapoint behaves similarly in primary and secondary feature spaces, and work by transferring slack information from one to the other, then it is reasonable to assume that better values will be transferred if the datasets are dependent on one another. This hypothesis is also supported by the superior results from the genuinely informative LUFe in the previous section.

Conversely, the complementary principle suggests that LUFe may perform better if $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$ are *not* informative about each other, and instead give different 'views' of the training data. LUFe requires the selection of a subset $\widehat{\mathcal{S}}$ of attributes, which is likely to be a sub-optimal solution of the NP-hard problem of feature selection. Given this, it is likely that a subset $\widehat{\mathcal{U}}$ which is less informative about $\widehat{\mathcal{S}}$ contains additional unique and beneficial information which the original feature selection procedure discarded. The presence of this additional and complementary information may then be reflected in better LUFe performance.

6.2.2 Informativeness of individual feature sets

Informativeness of unselected features

We have seen that unselected features that describe the distribution of the original labeled training dataset are more beneficial than those that do not. Based on this, we can further ask whether the features which are informative about *classification labels* are more beneficial than those which do not; alternatively, whether this informativeness of unselected feature set $\widehat{\mathcal{U}}$ can predict its utility as a secondary input to LUFe. The original LUPI uses a highly-informative secondary feature set. Although LUFe recasts this to use less-informative features, there is still justification for why a more informative secondary feature set would provide a better boost to performance, as described in 5.3. It was also shown in 6.1.4 that LUFe performs better when secondary features are informative about the dataset, and even better when secondary features are corresponding to the primary feature set. These principles could be generalised to state that the informativeness of a secondary feature set about labels indicates its usefulness.

Informativeness of selected features

It was observed in Chapter 4 that the improvement due to LUFe was more consistent among datasets where the baseline *FeatSel* performance was lower. Given that we evaluate the benefit of LUFe according to the additional improvement beyond standard feature selection, this means that the better feature selection performs, the harder it is for LUFe to be beneficial. This "ceiling effect" which was discussed in Section 4.3.5 and 5.1.4 means that the informativeness of the primary feature set may be negatively correlated with the performance increase due to LUFe.

Measuring similarity: HSIC

Hilbert-Schmidt Independence Criterion (HSIC) is proposed by Gretton et al. (2005) as a criterion to measure the difference between two multivariate random variables, and is therefore well-suited to the purpose of measuring dependence of two feature subsets, $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$. HSIC is 0 for independent variables, and a large value for dependent variables. HSIC uses the theorem that two random variables are independent if any bounded continuous function of the two random variables are uncorrelated (have zero covariance). As the basis for BAHSIC feature selection, HSIC was also discussed in Section 2.2.

6.2.3 Experimentation

The experimentation in this section is designed to test whether the benefit of LUFe can be predicted by either: the informativeness of either subset $\widehat{\mathcal{S}}$ or $\widehat{\mathcal{U}}$ about labels y, or the informativeness of $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$ about each other. These different kinds of informativeness were then assessed for correlation with the extent to which LUFe helped.

Measures of informativeness

The informativeness of $\widehat{\mathcal{S}}$ about y has of course already been measured; the standard *FeatSel-SVM* classifier uses only $\widehat{\mathcal{S}}$ to make a prediction about y. In other words, this uses $\widehat{\mathcal{S}}_{train}$ and y_{train} to learn a generalisable mapping $\widehat{\mathcal{S}}$ to y. This is assessed by applying the learned model to new data, and the resulting accuracy score reflects how well the model captures the data. This accuracy score for *FeatSel-SVM* indirectly measures how informative the selected features are about classification labels y.

A similar approach was taken to provide a proxy measure of the informativeness of $\widehat{\mathcal{U}}$. Classifiers were trained using $\widehat{\mathcal{U}}$ as the primary feature set, and the accuracy achieved by the resulting classifier was then used as a metric to judge the informativeness of $\widehat{\mathcal{U}}$. This *SVM-Reverse* setting simply trains a standard SVM classifier on $\widehat{\mathcal{U}}$ instead of $\widehat{\mathcal{S}}$. The pre-existing assignment into selected and unselected subsets was used so that for a given feature selection method, subset $\widehat{\mathcal{U}}$ is identical as in previous experimentation. In order to provide an additional metric that assesses the informativeness of $\widehat{\mathcal{U}}$ — relative to $\widehat{\mathcal{S}}$ — the accuracy score of *FeatSel-SVM* was then subtracted from *SVM-Reverse*.

As a measure for the informativeness of $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$, HSIC was calculated between these feature sets to measure their dependency. HSIC also was used as an alternative measure of informativeness between data and labels. By measuring the dependency between a feature set and labels, the HSIC score measures the extent to which the feature set can predict

the labels. HSIC was therefore calculated between $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$, $\widehat{\mathcal{S}}$ and y, and $\widehat{\mathcal{U}}$ and y.

To summarise, we consider the following three measures which may serve as indications of the usefulness of a feature set, each with its own justification. Each measure may be evaluated via one or more metrics, which are also listed below.

- **Shared information between $\widehat{\mathcal{S}}$ and y**:
  Due to the "ceiling effect", the informativeness of the primary feature set may be negatively correlated with the performance increase due to LUFe. Evaluated via:
  - HSIC calculated between $\widehat{\mathcal{S}}$ and y
  - Performance of *FeatSel-SVM* classifier
- **Shared information between $\widehat{\mathcal{U}}$ and y**:
  In keeping with the original LUPI theory that requires a highly informative secondary feature set. Evaluated via:
  - HSIC calculated between $\widehat{\mathcal{U}}$ and y
  - *SVM-Reverse* classifier performance
  - (*SVM-Reverse* classifier performance) - (*FeatSel-SVM* classifier performance)
- **Shared information between $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$**
  This may be either positively or negatively correlated with the benefit of LUFe, respectively in keeping with either the *consensus principle* or the *complementary principle* discussed above. Evaluated via:
  - HSIC calculated between $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$

Measuring the effect of informativeness

The experimental protocol then consists of comparing these similarity metrics with a measure of LUFe performance, I where:

$$I = A_{LUFe-SVM+} - A_{FeatSel-SVM} \tag{6.3}$$

where A is the accuracy achieved by the classifier denoted in subscript. Therefore, I is a measure of improvement by LUFe over standard feature selection in a given dataset, This measure of improvement (rather than absolute score) was used to better indicate directly the benefit achieved by using unselected features.

This process was repeated for each of the five feature selection methods used in 5.1.7. Pearson correlation coefficient (PCC or r) was used to measure this. If a metric is strongly

positively correlated with LUFe improvement, then its increasing value would be seen to be associated with increased benefit of LUFe; if r shows a strong inverse correlation then an increase in the metric would be associated with worse LUFe performance.

6.2.4 Results

All results are summarised in Table 6.2 and Figures 6.2 to 6.7, and will be discussed in turn.

Table 6.2: **Pearson's correlation coefficients**

PCC between various usefulness metrics, with improvement by *LUFe* over *FeatSel*

	Metric correlated with LUFe improvement					
	Classifier Accuracy			HSIC value		
Feature Selection	FeatSel-SVM Improvement	SVM-Reverse Accuracy	SVM-Reverse Improvement	$\widehat{\mathcal{S}}$ and y	$\widehat{\mathcal{U}}$ and y	$\widehat{\mathcal{S}}$ and U
ANOVA	-0.38	0.07	0.32	-0.13	-0.12	-0.11
BAHSIC	-0.38	0.06	0.30	-0.14	-0.10	-0.08
CHI2	-0.40	0.07	0.33	-0.14	-0.12	-0.12
MI	-0.63	0.46	0.63	-0.03	-0.04	-0.03
RFE	-0.24	0.16	0.04	-0.13	-0.05	-0.02

Shared information between $\widehat{\mathcal{S}}$ and y

The informativeness of the *FeatSel* setting (measured as improvement over the *ALL* baseline) was shown to be negatively correlated with the improvement due to LUFe (shown in Figure 6.2). In other words, when standard feature selection performed better, there was a smaller improvement by LUFe, and when it performed worse, LUFe was more beneficial. This finding was consistent across all five feature selection metrics, with PCC ranging from -0.239 to -0.633.

The dependence of y on $\widehat{\mathcal{S}}$ was also measured using HSIC. As shown in Figure 6.3, PCC was consistently negative between this value, and the improvement by LUFe over feature selection. However, these negative correlation scores were very low, ranging from -0.029 to -0.138.

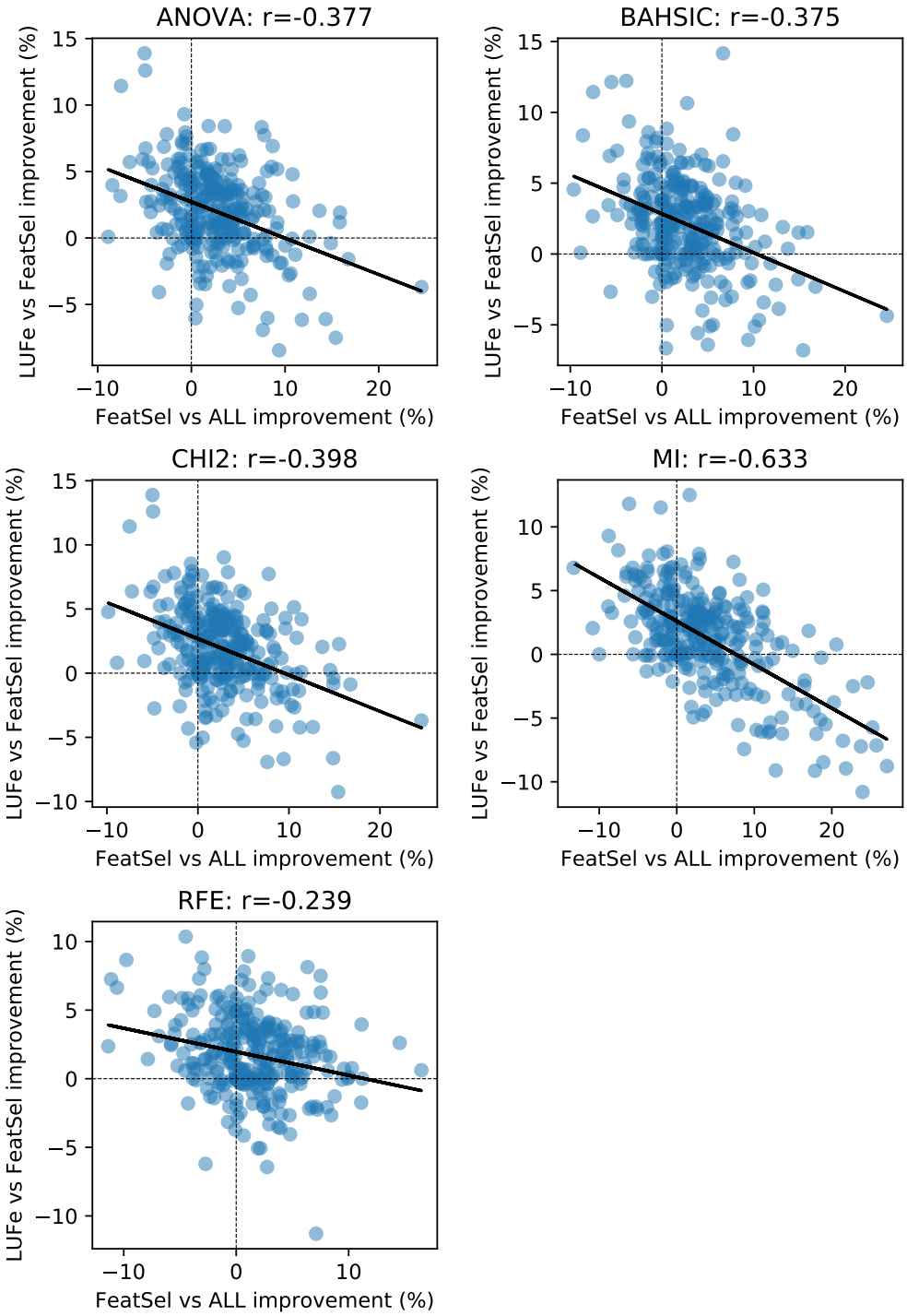

Figure 6.2: Scatter plots showing improvement by feature selection compared to all-features baseline on horizontal axis, plotted against *LUFe* improvement in accuracy over corresponding *FeatSel* on vertical axis. Correlation coefficients indicated in sub-plot title. Each sub-plot uses a different feature selection method; each point represents a single dataset.

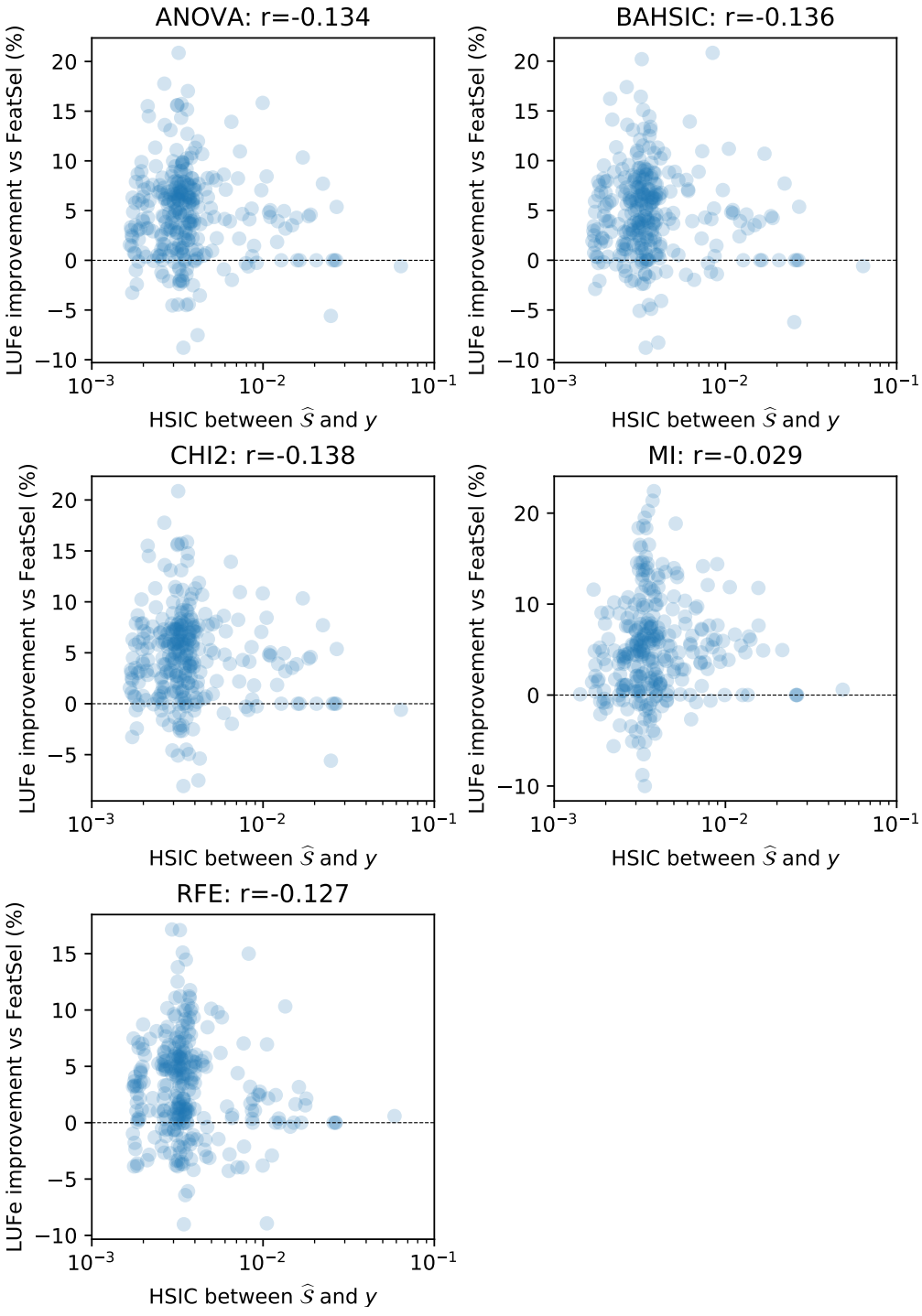

Figure 6.3: Scatter plots: showing HSIC score between selected features $\widehat{\mathcal{S}}$ and labels y on horizontal axis (log scaled), plotted against *LUFe* improvement in accuracy over corresponding *FeatSel* on vertical axis. Correlation coefficients indicated in sub-plot title. Each sub-plot uses a different feature selection method; each point represents a single dataset.

Shared information between $\widehat{\mathcal{U}}$ and y

The informativeness of $\widehat{\mathcal{U}}$ about y was first assessed using the performance of the *SVM-Reverse* classifier (shown in Figure 6.4). For four of the five feature selection methods, there was surprisingly very little correlation between the accuracy score of *SVM-Reverse* setting that used $\widehat{\mathcal{U}}$ as input, and the performance of the *LUFe* setting that used these same unselected features as a secondary dataset. However, there was an apparent positive correlation when mutual information was used for feature selection. PCC ranged between -0.162 to 0.460 across the five feature selection methods. However, the second metric based off *SVM-Reverse*, that is, $A_{SVM-Reverse} - A_{FeatSel}$ was consistently positively correlated with LUFe improvement, with significant correlation in 4 out of 5 cases (PCC between 0.038 to 0.631).

The alternative method for label-informativeness of $\widehat{\mathcal{U}}$ — the HSIC score between $\widehat{\mathcal{U}}$ and y — was also found to not correlate with the improvement by *LUFe* over *FeatSel*, over all five feature selection methods. The correlation coefficients range from -0.047 to -0.125, which are not indicative of a linear relationship between the variables. This is shown in Figure 6.6.

Shared information between $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$

Dependence of $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$ was measured using HSIC. This was not found to have any strong correlation with a LUFe improvement, with PCC ranging from -0.023 to -0.118 across the five feature selection methods. This correlation was negative in all cases but too small to be likely to be significant.

Discussion

The clearest conclusion that can be drawn from this research was the confirmation that a ceiling effect limits the benefit of LUFe in cases where standard feature selection has performed well. There are two explanations for this: firstly, that LUFe utilises classification-relevant information in informative features which were erroneously discarded by feature selection. In this case, if feature selection does a better a job, and maximises the amount of non-redundant information in $\widehat{\mathcal{S}}$, there is less remaining useful information for LUFe to exploit. An alternative explanation is that simply the better baseline is harder to improve upon. Caruana and de Sa (2003) speculate that additional constraints in their method of incorporating unselected features may simply 'push' the model to go further; however if the model is already close to converging on an optimal learned vector of coefficients for the

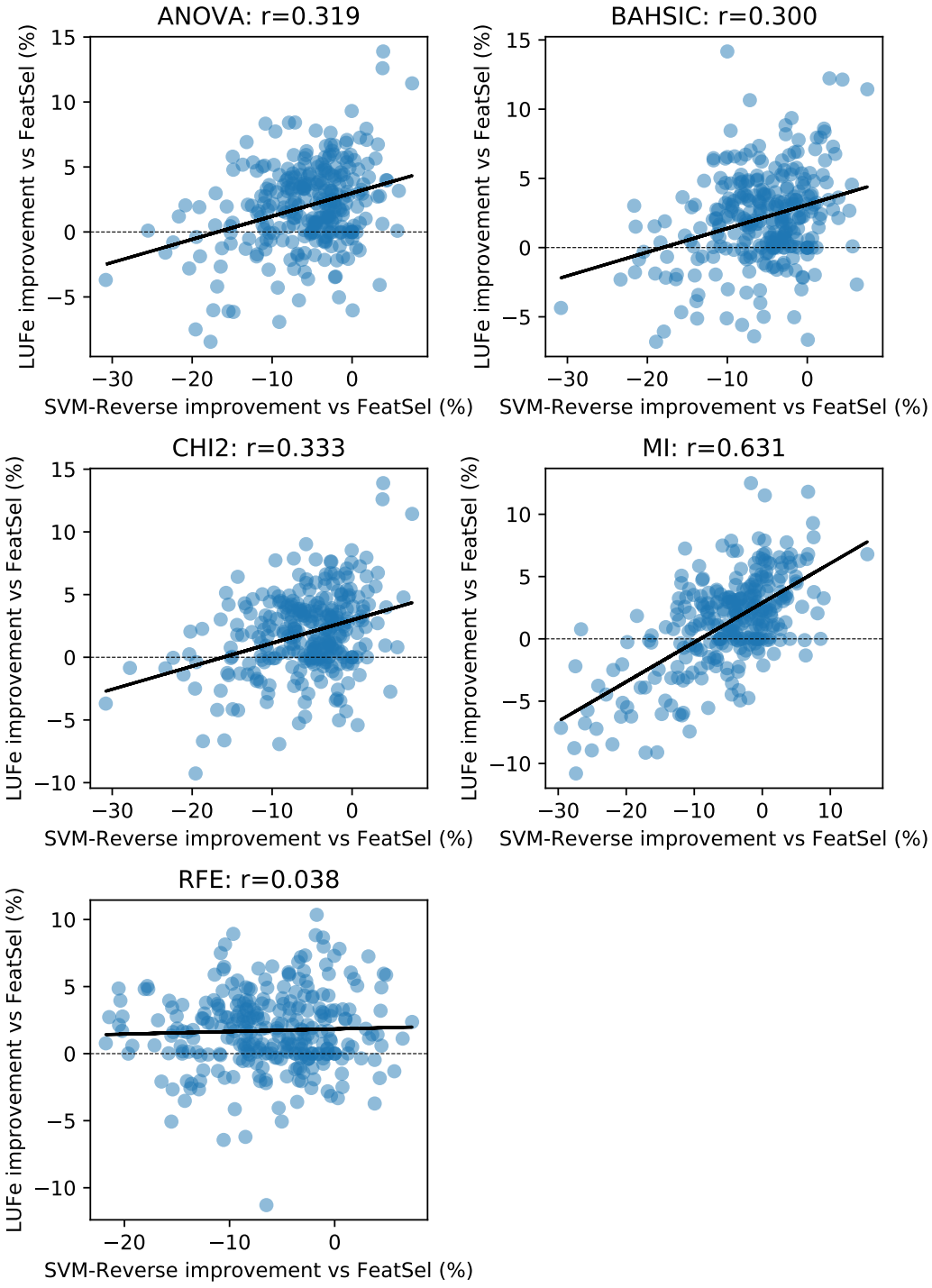

Figure 6.4: Scatter plots: showing improvement by *SVM-reverse* compared to *FeatSel* on horizontal axis, plotted against *LUFe* improvement in accuracy over corresponding *FeatSel* on vertical axis. Correlation coefficients indicated in sub-plot title. Each sub-plot uses a different feature selection method; each point represents a single dataset.

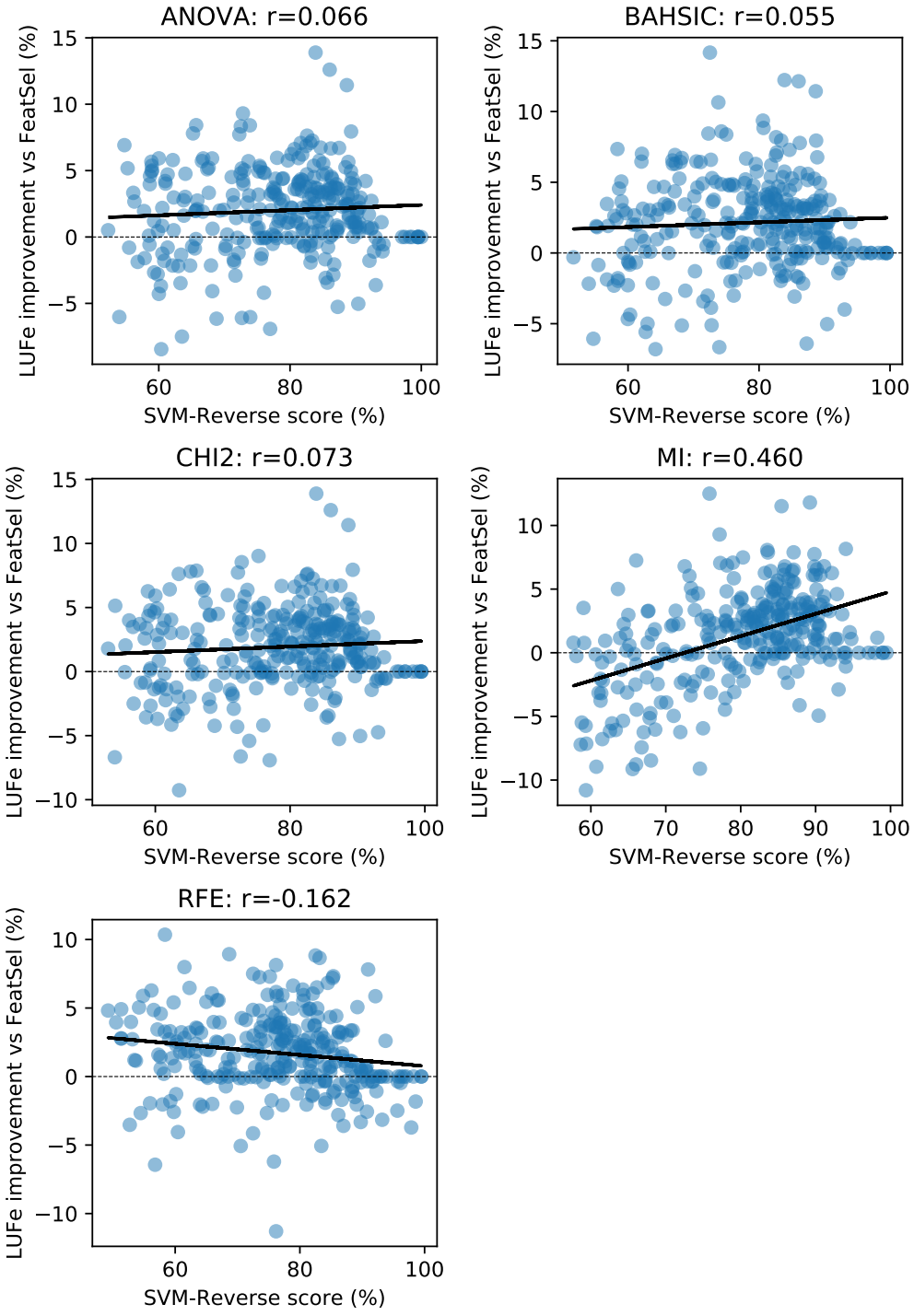

Figure 6.5: Scatter plots: showing accuracy score by SVM-reverse on horizontal axis, plotted against *LUFe* improvement in accuracy over corresponding *FeatSel* on vertical axis. Correlation coefficients indicated in sub-plot title. Each sub-plot uses a different feature selection method; each point represents a single dataset.

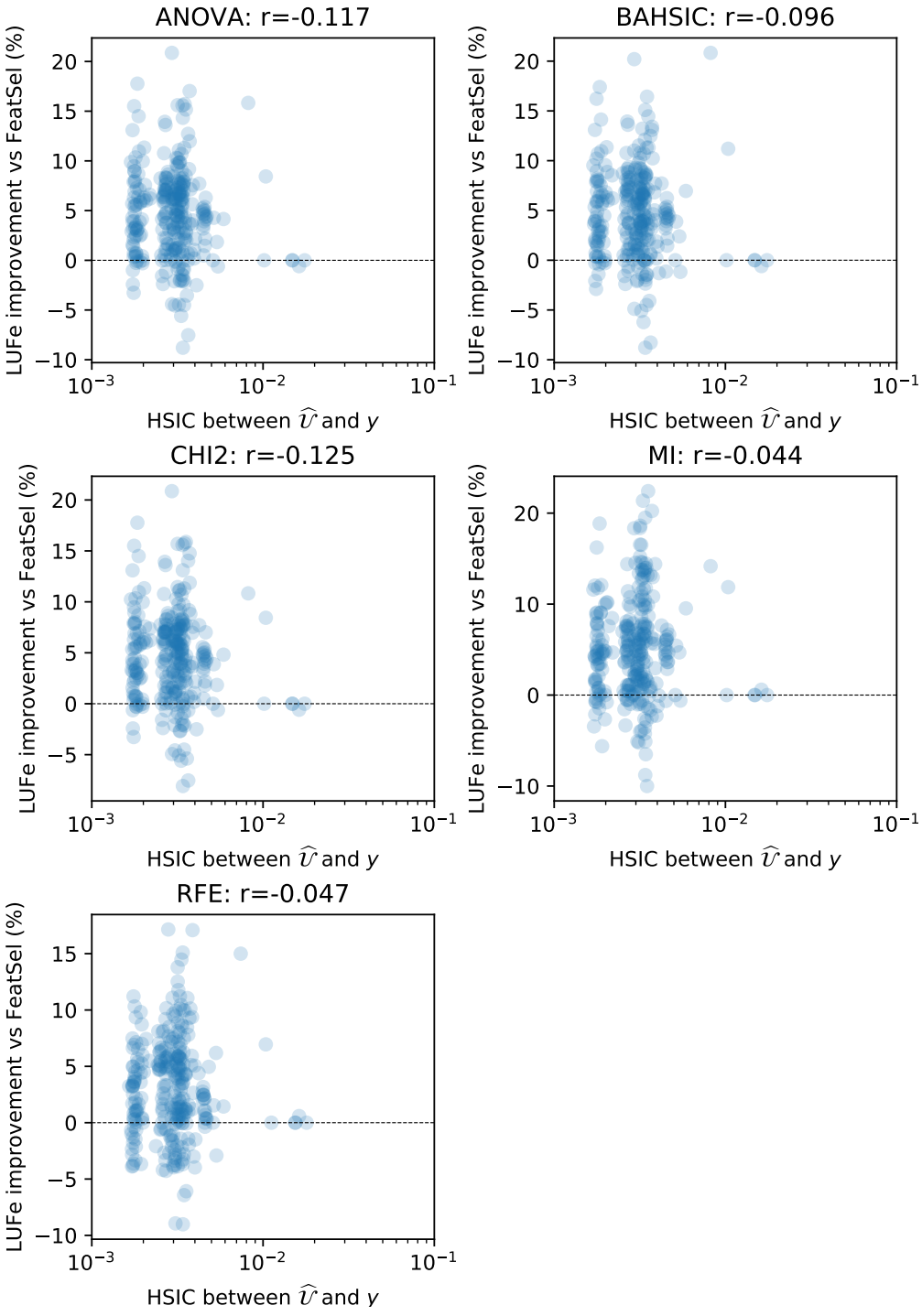

Figure 6.6: Scatter plots: showing HSIC score between unselected features $\widehat{\mathcal{U}}$ and labels y on horizontal axis (log scaled), plotted against *LUFe* improvement in accuracy over corresponding *FeatSel* on vertical axis. Correlation coefficients indicated in sub-plot title. Each sub-plot uses a different feature selection method; each point represents a single dataset.

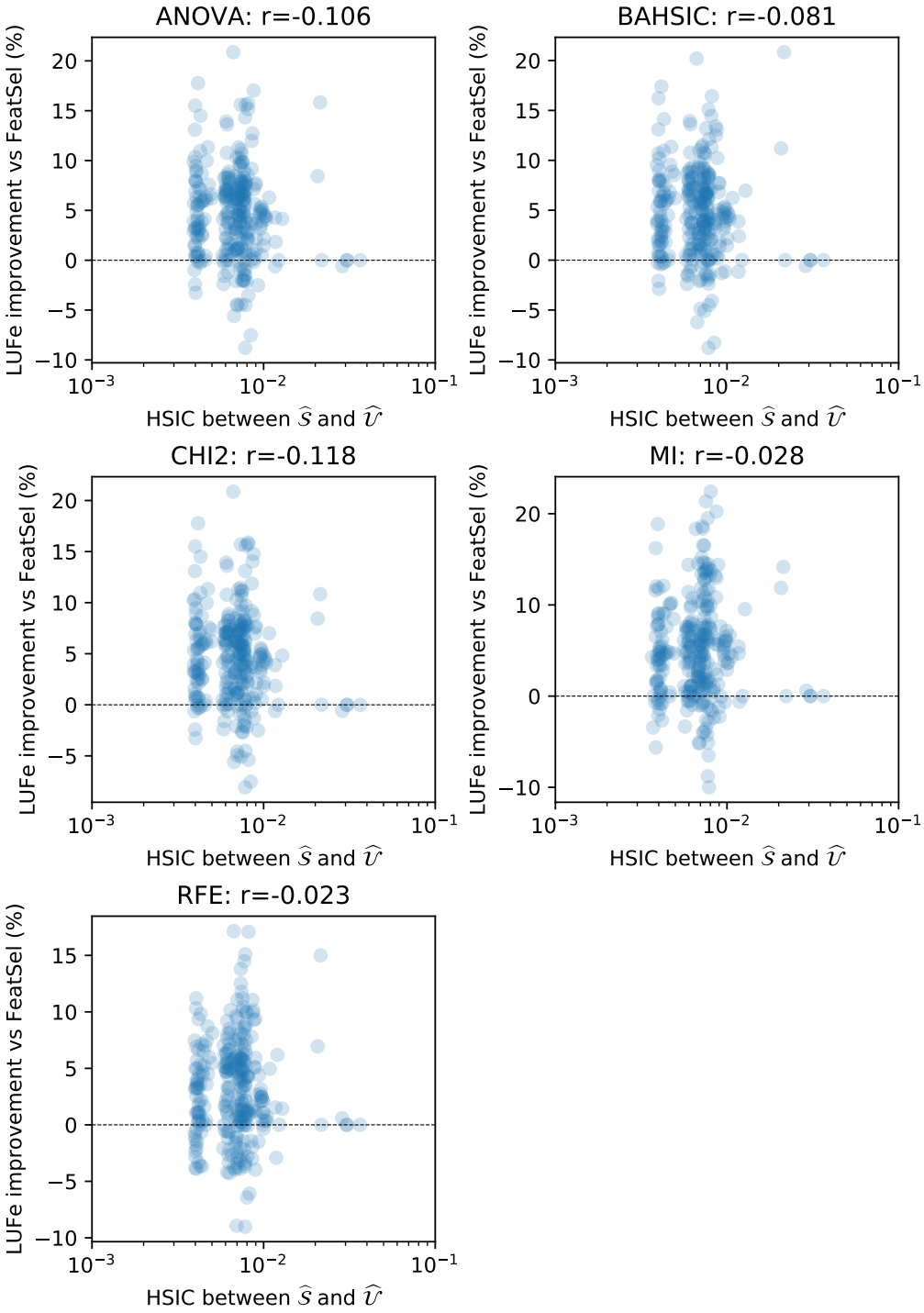

Figure 6.7: Scatter plots: showing HSIC score between selected features $\widehat{\mathcal{S}}$ and unselected features $\widehat{\mathcal{U}}$ on horizontal axis (log scaled), plotted against *LUFe* improvement in accuracy over corresponding *FeatSel* on vertical axis. Correlation coefficients indicated in sub-plot title. Each sub-plot uses a different feature selection method; each point represents a single dataset.

selected features, then there is less room to squeeze better performance out of the model.

The lack of consistent correlation between *SVM-Reverse* performance, and *LUFe* improvement I, was somewhat surprising. Given that $A_{SVM-Reverse}$ was intended as a proxy for $\widehat{\mathcal{U}}$ informativeness, then this result can be interpreted as stating the $\widehat{\mathcal{U}}$ informativeness does not affect magnitude of improvement. However, PCC indicates the strength of a linear relationship so it remains possible that a non-linear relationship has gone undetected; further experimentation is required. Furthermore, the relatively large number of samples (295) reduces the likelihood of a large correlation coefficient. The strongest positive correlation ($\rho = 0.194$) was found when mutual information was used to partition the subsets; the strongest negative correlation ($\rho = -0.163$) was found when RFE was used.

There was more consistent correlation between $(A_{SVM-Reverse} - A_{FeatSel-SVM})$ and I. Three feature selection methods (ANOVA, BAHSIC, Chi^2) had a weak positive correlation and a fourth (MI) had a strong correlation. It may seem that the improvement of *SVM-Reverse* over standard feature selection is a promising way to use attributes of the secondary set to predict its usefulnes. However, this metric looks at the difference between using selected features $\widehat{\mathcal{S}}$ only, compared with using unselected features $\widehat{\mathcal{U}}$ only, so depends on both of these settings. On closer inspection, we can see this correlation is closely related to the previously-noted ceiling effect; it is more linked to the performance of *FeatSel-SVM*. Through comparison of 1st and 3rd columns in 6.2 we can see that the magnitude of PCC for both metrics is similarly ordered across feature selection methods, and that they are of similar absolute values. Therefore, given the lack of association between direct *SVM-Reverse* and I, this $(A_{SVM-Reverse} - A_{FeatSel-SVM})$ metric essentially resembles the negative of *FeatSel-SVM*, with added noise, so has limited usefulness as a metric to predict the value of a subset $\widehat{\mathcal{U}}$.

HSIC score between feature sets $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{U}}$ did not exhibit any positive *or* negative correlation with LUFe improvement, so did not display the effects of either *consensus principle* or *complementary principle*. This was another surprising failure to reject the null hypothesis. One possible explanation is that both of these principles were in play, and that a secondary dataset may be beneficial due to either conveying complementary information, or by being more informative about the primary set.

It was observed in the previous chapter (Section 5.3.3), when using a subset of the 'best' t% or 'worst' b% features, that the quantity of features used appears to make up for the quality. This may further be a factor in the difficulty of predicting the benefit of an unselected feature set.

This section attempted to investigate *which* feature sets may be harnessed by the LUFe paradigm to improve classification performance. The next section will attempt to discover *how* this effect occurs.

6.3 How does the LUFe setting improve performance?

6.3.1 Learning curves

The bias-variance trade-off was discussed in Chapter 4 to explain the behaviour of LUFe. To recap: it was hypothesised that the LUFe benefit occurs by harnessing the lower variance of a less complex model that fits to fewer features, while also exploiting the unselected features as a means of guiding the search space and learning a model with lower bias. This section intends to verify whether this explanation sufficiently explains the behaviour seen.

The bias and variance for a classifier can be visualised in a learning curve, which typically plots the error achieved by a given model — on both the training set and the test set — as a function of the size of the training set. A high-bias model can be expected to be achieve a small training error when the size of the train set is small; the model sufficiently captures the data. However, as the number of training examples grows, training error increases, as the model is less capable of capturing the data. The error on a test set (or from cross-validation in this case where a separate test set is unavailable) is also high in a high-bias model. Conversely, a high-variance model tends to over-fit to training data, and therefore achieve low training error (but this will worsen as the amount of data increases). This also means that the model does not generalise well to new instances, so performs badly on the test set. There is therefore a larger gap between the errors in a high-variance model, whereas they may converge in a high-bias model.

6.3.2 Experimentation

Learning curves are plotted in Figure 6.8 for the *ALL-SVM*, *FeatSel-MI-SVM*, and *LUFe-MI-SVM+* classifiers[2], showing performance on train and test sets. The results are also summarised in Table 6.3. Classifiers were trained with various amounts of training data, ranging from 20-100% in 20% increments. The subsets of training data were randomly selected from each class for each dataset, such that class balance was maintained from each original dataset; these subsets were consistent across different classifiers.

The learning curve shows that performance on the test set improves as more training data is added, both for LUFe and for the standard SVM approaches. At each increment,

[2]Producing a learning curve can be computationally intensive, involving the repeated training of a classifier with different amounts of data. For this reason, learning curves were plotted only for *FeatSel-MI-SVM* and *LUFe-MI-SVM+* settings, used because they are the highest performing standard feature selection and LUFe methods

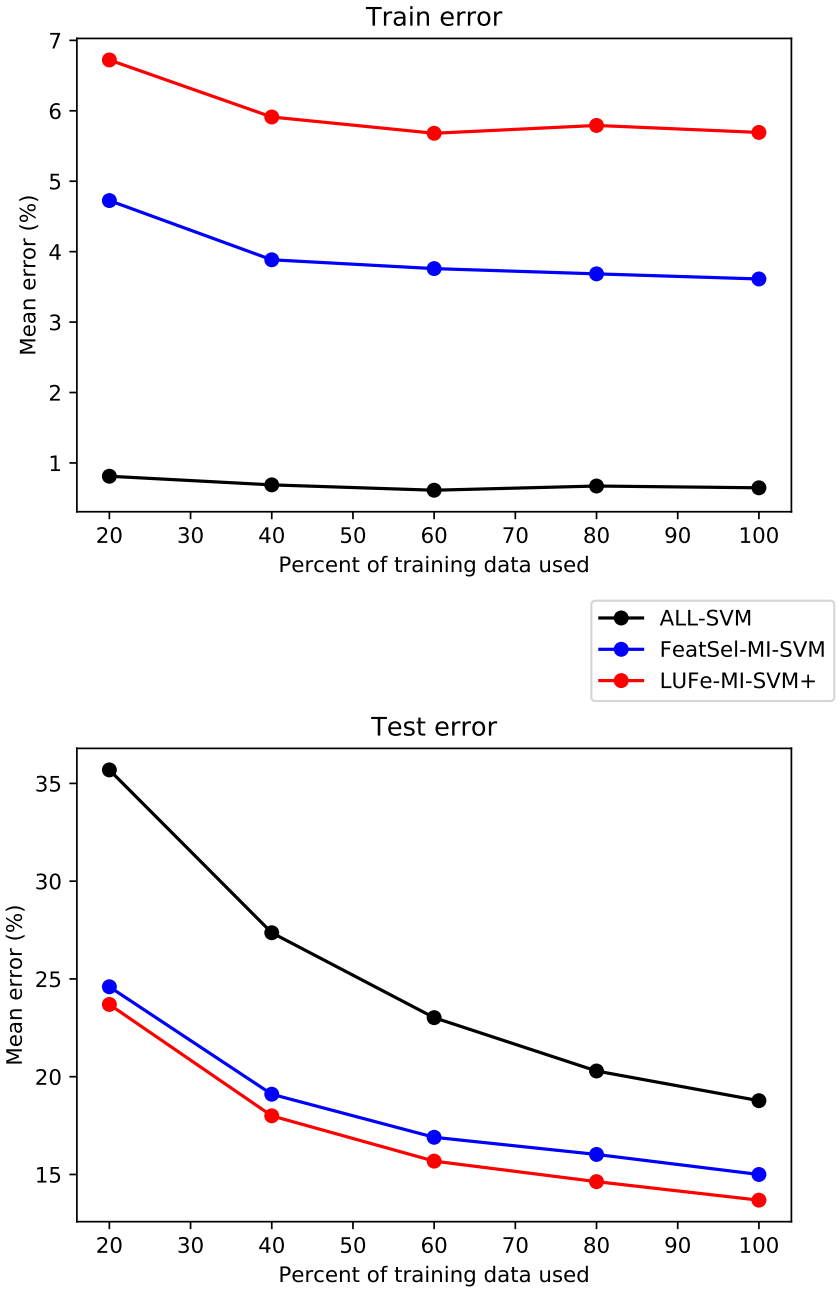

Figure 6.8: Learning curves for *LUFe-MI-SVM+*, *FeatSel-MI-SVM*, and *ALL-SVM* classifiers, showing mean error rates over 295 datasets, when deployed on training data (top) and testing data (bottom).

Table 6.3: Performance of *ALL-SVM*, *FeatSel-MI-SVM*, and *LUFe-MI-SVM+* classifiers, showing mean accuracy score when deployed on training data and testing data, over 295 datasets

	Test score			Train score		
% Training data	All	FeatSel	LUFe	All	FeatSel	LUFe
20%	64.3%	75.4%	76.3%	99.2%	95.3%	93.3%
40%	72.6%	80.9%	82.0%	99.3%	96.1%	94.1%
60%	77.0%	83.1%	84.3%	99.4%	96.2%	94.3%
80%	79.7%	84.0%	85.4%	99.3%	96.3%	94.2%
100%	81.2%	85.0%	86.3%	99.4%	96.4%	94.3%

LUFe-MI-SVM+ outperforms *FeatSel-MI-SVM*, which in turn outperforms *ALL-SVM*. Conversely, when classifying the training set, this order is reversed: using all features consistently produces the highest train accuracy, while LUFe is consistently the worst, behind standard feature selection. All three settings achieve very low train error which changes very little as more data is added.

6.3.3 Discussion

Let us consider the results in terms of the hypothesised method of action for LUFe. Firstly, *ALL-SVM* performance fits the profile of a high-variance classifier, which is able to very closely fit the training data but is not generalisable to the test data. This produces the low train error that slowly increases with m, and a considerably higher test error. This high-variance classifier results in the observable large difference between train error and test error, which reduces as the amount of training data is increased.

We can next verify that feature selection on these datasets performs as expected, through comparison with *ALL-SVM*: the train error is 3.0-3.9% higher for *FeatSel-MI-SVM* and test error is 3.8-11.1% lower — with a smaller gap as m increases. This is indicative of beneficial feature selection that reduces the variance in a high-variance classifier.

Let us now consider whether this variance reduction is at the expense of higher bias. A high-bias model tends to have similarly high error both on train and test sets; the model fails to capture the training data, and therefore also cannot be successfully applied to unseen instances. Train error tends to increase as the training set grows and becomes

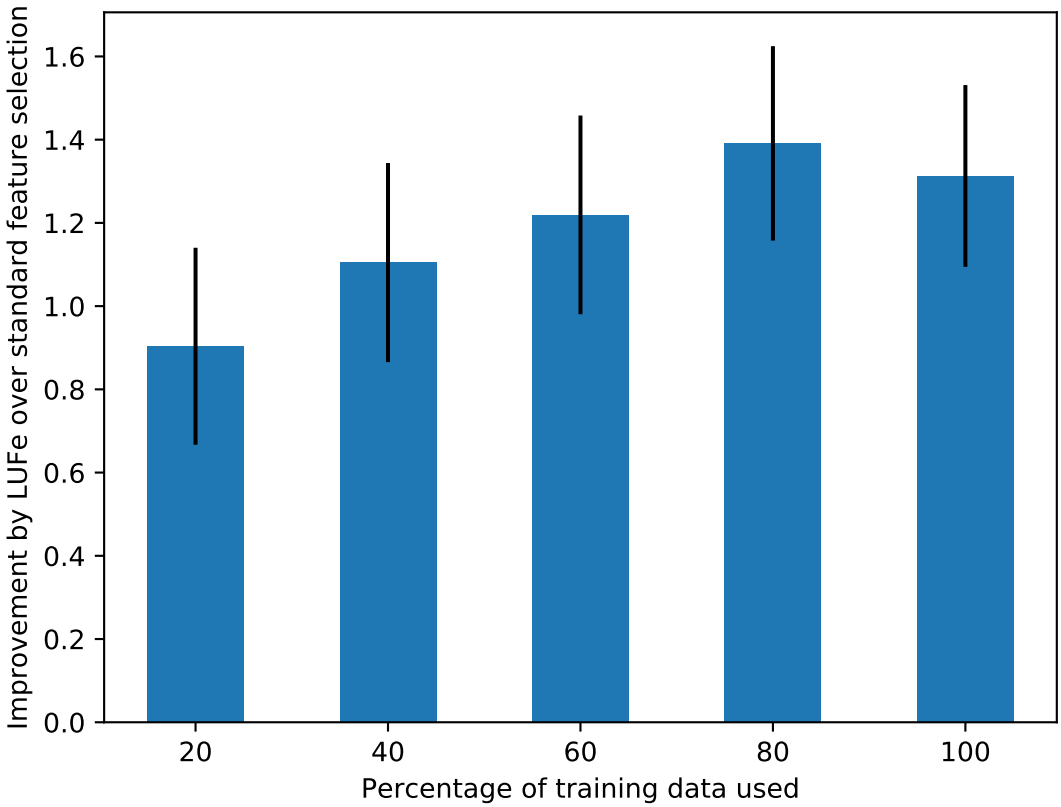

Figure 6.9: Mean differences in accuracy score on test data between *LUFe-MI-SVM+* and *FeatSel-MI-SVM* classifiers, over 295 datasets. Error bar shows SEM.

increasingly difficult for the high-bias model to capture. Test error tends to initially drop before plateauing, because the high-bias model fails to harness the additional data for improvement. This is not the case for *FeatSel-MI-SVM*, which performs consistently well on the training set, and whose test error continues to improve as additional training data is supplied.

Comparison between *FeatSel-MI-SVM* and *LUFe-MI-SVM+* further confutes the 'reduced bias theory'. *LUFe-MI-SVM+* performs 0.9-1.4% better on test data at each point on the learning curve, but 1.9-2.1% worse on train data. These performance differences mimic those seen between *ALL-SVM* and *FeatSel-MI-SVM*, which were characteristic of reduced *variance*: *LUFe-MI-SVM+* produces a smaller gap between train and test error, which is indicative of lower variance. This surprisingly implies that the beneficial effect of LUFe is not mediated through bias-reduction, but rather due to a further reduction in variance, beyond that provided by standard feature selection.

Let us now consider LUFe in terms of the original LUPI theory. This posited that

SVM+ would improve performance compared to standard SVM, by providing the slack variables, which reduces the number of parameters to be estimated and thereby enables faster convergence on an optimal solution. The learning curves in 6.8 appear to bear this out. Interestingly, the difference in test errors between *FeatSel-MI-SVM* and *LUFe-MI-SVM+* tend to widen as increasing amounts of training data are supplied, between $m = 20\%$ and $m = 80\%$, before narrowing again at $m = 100\%$. (These differences are displayed in 6.9). This initial increased distance between the curves illustrates that LUFe, like standard LUPI, enables better performance via faster convergence; the narrowing of gaps once all data is used reflects that standard feature selection performance begins to catch up once LUFe performance starts 'levelling off' and improves more slowly with additional data.

Chapter 7

Conclusion

7.1 Summary of results

As the prevalence of 'smart' devices and pervasive intelligence has exploded in recent years, so too has the need for 'front-loaded' machine learning approaches, which exploit greater resources at training time, but allow low-cost models at deployment time. This thesis introduced a learning paradigm, movitated as a solution to this problem, called Learning using Unselected Features (LUFe). LUFe was presented as a means of combining Learning Using Privileged Information (LUPI) with feature selection. This was intended to achieve the reduced model complexity and deployment cost of feature selection, but with an additional boost to model performance. In doing so, the LUPI framework was repurposed, to allow less-informative information to be harnessed as a secondary dataset.

Initial experimental work confirmed the validity of this approach. LUFe was implemented for classification, using the SVM+ LUPI algorithm, as a means to harness feature subsets that had been partitioned by recursive feature elimination (RFE). In 71.9% of the 295 datasets, LUFe improved performance over standard feature selection, which already tended to improve performance over using the entire feature set. This LUFe performance was similar to that of an existing Multi-Task Learning method for incorporating unselected features, and demonstrated that the LUPI framework can enhance performance even without highly-informative data from another domain.

The application of the LUFe framework was then extended in a number of ways, and its benefit was shown to be consistent when using different feature selection methods. Four filter methods were used in combination with LUFe (again implemented using SVM+), and in each case LUFe continued to boost classification accuracy in 67.1—73.2% of datasets, compared to the standard feature selection baseline. The use of different implementations

was then found to have a more varied effect on performance. SVMΔ+ implementations of LUFe consistently increased accuracy score compared to standard feature selection in all cases, but to a consistently smaller degree than the SVM+ implementation. *d*SVM+ implementations of LUFe were not beneficial, and in some cases detrimental to performance. These results were attributed to the consideration of data labels in the privileged space by SVMΔ+ and *d*SVM+. This approach for handling secondary features is less appropriate to the LUFe paradigm than the original LUPI setting.

LUFe performance was shown to be relatively consistent when only a subset of the top t% features was used, but more variable when the bottom b% features were used. This was interpreted as showing that LUFe is beneficial when the unselected feature set is informative, but that the addition of less useful attributes does not worsen performance.

Building upon this, the impact of the unselected features themselves was assessed through comparison between settings that used alternative secondary datasets — which were either 'shuffled' or randomly generated. While the genuine LUFe setting performed best for all feature selection methods, mean accuracy was still generally boosted by the alternative settings. Taken together, this hierarchy of results demonstrated that while the LUFe boost effect benefits from harnessing information in unselected features, it is also partially attributable to the model itself, and its additional constraints.

The performance of standard feature selection was shown to be the strongest predictor of LUFe performance, out of a range of metrics which were assessed. This reflected a 'ceiling effect', due to high-performing feature selection being harder to improve upon, and less useful information remaining in the unselected features. Finally, the mechanism of LUFe's action was concluded to be mediated through a further reduction in variance, beyond the initial reduction provided by standard feature selection.

7.2 Limitations

7.2.1 Limitations to the scope of findings

The first general limitation of this research concerns the scope of findings, and the range of settings in which LUFe could improve performance. This is largely due to the dataset collection which was used for experimentation. The Tech-TC collection was chosen for a number of factors discussed in Section 4.3.1: the precedent for its usage, the high dimensionality and low number of training instances, and the range of 'difficulty levels' in class separability. Furthermore, the large number of datasets allowed statistically significant

conclusions to be drawn from the aggregated results. However, the relative homogeneity of the datasets — in terms of both domain and dimensionality — limits the universality of the findings. There may be some statistical similarities between the datasets, or some characteristics which are shared by them, that make them particularly amenable to the LUFe setting in a way which other datasets may or may not be.

In particular, the datasets differ from most domains encountered within Internet of Things settings, particularly in terms of volume and velocity of data. They will also not represent this broad field in terms of number of instances, feature density, and feature correlation, so some caution should be applied when extrapolating results to IoT. For example, the bag-of-words datasets in this collection have much sparser features than general IoT datasets.

The scope of findings is also constrained by the range of hyperparameter settings that were tested. The number of features selected, k, is a hyperparameter that requires adjustment. In this work, $k = 300$ was used throughout, with the additional $k = 500$ setting also employed in the initial proof-of-concept work. These values were chosen to follow the protocol described in Paul et al. (2015), and to significantly reduce the storage and computational requirements; 300 features represents 0.5 — 5.8% of the full dimensionality, depending on dataset. Due to the wider range of other variables being tested, this parameter was fixed in order to avoid a combinatorial explosion of settings. However, it is possible that the size of the selected or unselected subsets — or their relative sizes — is particularly conducive to benefitting from the LUFe framework.

7.2.2 Limitations of applications for LUFe

A second limitation concerns the applicability of the approach. The LUFe setting was introduced as a method to front-load computational effort to training time, by fitting a model to the top k selected features only. For this use-case to be fulfilled, and LUFe to be worthwhile, the initial dataset dimensionality, and the data velocity in deployment, must sufficiently exceed the computational requirements of the deployment system.

7.2.3 Limitations of assessment methods

Performance was reported in this thesis using only the single metric of accuracy. Other methods such as precision, recall, F-score and the area under the receiver operating characteristic curve could also be considered (Refaeilzadeh et al., 2007). Each of these metrics emphasises a different aspect of model performance, and the choice is task-dependent. For

example, precision is preferable in cases when false positives need to be avoided, while recall places more importance on avoiding false negatives. In this work the classes were reasonably balanced, and particular importance was not placed on one particular kind of error, so accuracy is sufficiently descriptive of the findings, but the usage of other metrics would provide a more holistic view of performance. Furthermore, if this paradigm were to be expanded to other tasks, for example with more unbalanced classes, then other metrics may also be required.

7.3 Further work

7.3.1 Further work to address limitations

The limitations discussed in the previous section should be addressed in future work. Firstly, the LUFe paradigm should also be tested on a wider variety of datasets from a range of domains. Although the usage of LUFe produced statistically significant improvement on those datatsets which were tested, these datasets were all from a similar domain: website classification. The LUFe setting should be tested on, for example, image processing tasks to assess whether its value extends beyond this setting. Further work should be conducted to assess LUFe in a wider variety of datasets that are represenative of the IoT field at large. The k hyperparameter should also be tested more widely to investigate whether LUFe is particularly effective for specific ranges of dimensionality of $\mathcal{U}$ and $\mathcal{S}$, or for ratios between these sizes. Furthermore, these assessments should be carried out using a range of different performance metrics.

7.3.2 Other areas for further work

Chapter 5 broadened the scope of the framework and used two alternative LUPI implementations (dSVM+ and SVMΔ+) with mixed results. One exciting area for expansion of the LUFe paradigm is to extend beyond these SVM-based implementations described in this work. As discussed in the literature review, LUPI has now been extended to neural network based systems, and the Gaussian Process Classifier (GPC) has been extended to incorporate privileged information in the novel GPC+ format. Both of these approaches could be employed by the LUFe framework to harness unselected features.

Future work could also broaden the scope of LUFe beyond the simple case of binary classification. This was the focus of this work as the majority of work in the field of LUPI has been carried out on supervised classification tasks, with many LUPI algorithms

being an analogue or extension of a corresponding single-domain classifier. One avenue of research would be the case of applying LUFe to multi-class classification. More broadly, the LUFe paradigm could be expanded to a wider range of tasks. In its essence, it simply requires a task which LUPI algorithms can be applied to, and a dataset of sufficient dimensionality to be split by feature selection. Therefore it may be applied to any task for which LUPI methods exist; for example, clustering (Feyereisl and Aickelin, 2012).

Neural networks and deep learning are a current focal point for machine learning research, and combination with these approaches is a logical next step for LUFe. Feature selection can be automatically performed by neural networks, for example using an error function that forces larger differences in weights between relevant features and irrelevant ones (Verikas and Bacauskiene, 2002). The features which are discarded in this process could be utilised in some secondary regularisation role, as in LUFe. Representation learning (or feature learning) is a field which attempts to learn representations of raw data from which useful information can be easily extracted for predictive modelling (Bengio et al., 2013). This is another area which could be combined with LUFe, with a secondary representation of support features also learned, or with some selectivity applied to the learned representation to partition it into primary and secondary feature sets. A combined LUFe and LUFe-MTL setting could also be investigated, which would use subsets of unselected features in both input and output roles.

7.4 Closing remarks

Machine learning is a dynamic field of research, and even during the research and writing of this thesis, it has progressed quickly — both in terms of technical development, and breadth of application. These two spheres of growth complement each other; technological advances allow machine learning to permeate new areas of technology. It is hoped that on the technical side, this work has presented a useful new paradigm, which allows better accuracy when using a selected subset of features, and that on the practical side, this work will have application in settings with limited resources at deployment. It is also hoped that this work has a further technical benefit, and raises some questions about the mechanism of LUPI that may inspire further research.

Bibliography

Argyriou, A., Evgeniou, A., and Pontil, M. (2007). Multi-task feature learning. *Advances in neural information processing systems*, 19:41. 50

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805. 54

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662. 11, 43

Batuwita, R. and Palade, V. (2013). Class imbalance learning methods for support vector machines. 80

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828. 157

Bi, J., Bennett, K., Embrechts, M., Breneman, C., and Song, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3(Mar):1229–1243. 36

Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, volume 1. springer New York. 3, 15

Blitzer, J., Dredze, M., Pereira, F., et al. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447. 48

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics. 48

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM. 51

Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271. 32

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM. 3, 43, 44

Caruana, R. (1995). Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems*, pages 657–664. 39

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28:41–75. 39

Caruana, R. and de Sa, V. R. (2003). Benefitting from the variables that variable selection discards. *Journal of Machine Learning Research (JMLR)*, 3:1245–1264. 38, 39, 41, 55, 57, 84, 86, 87, 89, 110, 141

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. 13, 17

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007a). Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM. 50

Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007b). Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM. 49

Daume III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. 47

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30. 65

Domingos, P. (2000). A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238. 12

Feyereisl, J. and Aickelin, U. (2012). Privileged information for data clustering. *Information Sciences*, 194:4–23. 27, 157

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305. 102

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer. 136

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660. 54

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research (JMLR)*, 3:1157–1182. 3, 32, 33, 35

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422. 35, 57, 59

Hernández-Lobato, D., Sharmanska, V., Kersting, K., Lampert, C. H., and Quadrianto, N. (2014). Mind the nuisance: Gaussian process classification using privileged noise. In *Advances in Neural Information Processing Systems*, pages 837–845. 27

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *stat*, 1050:9. 2, 3, 11, 31, 44

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification. 88, 93

Jiang, J. (2008). A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*. 47

Jiang, J. and Zhai, C. (2007). Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271. 49

Koh, K., Kim, S.-J., Boyd, S., and Lin, Y. (2007). An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research (JMLR)*, 8:1519–1555. 36

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(12):273 – 324. 32, 34

Koller, D., Sahami, M., et al. (1996). Toward optimal feature selection. In *ICML*, volume 96, pages 284–292. 36, 41

Lapin, M., Hein, M., and Schiele, B. (2014). Learning using privileged information: Svm+ and weighted svm. *Neural Networks*, 53:95–108. 30

Lee, S.-I., Lee, H., Abbeel, P., and Ng, A. Y. (2006). Efficient l1 regularized logistic regression. In *AAAI Conference on Artificial Intelligence (AAAI)*. 36

Lefakis, L. and Fleuret, F. (2014). Jointly informative feature selection. In *Artificial Intelligence and Statistics (AISTATS)*. 59

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. (2015). Unifying distillation and privileged information. *stat*, 1050:19. 11, 31, 45

Luo, M. and Luo, L. (2010). Feature selection for text classification using or+ svm-rfe. In *2010 Chinese Control and Decision Conference*, pages 1648–1652. IEEE. 64

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., and Sheth, A. P. (2018). Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 4(3):161–175. 55, 56, 63

Molina, L., Belanche, L., and Nebot, A. (2002). Feature selection algorithms: a survey and experimental evaluation. In *IEEE International Conference on Data Mining (ICDM)*. 32

Mukherjee, S., Osuna, E., and Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 511–520. IEEE. 28

Navot, A., Gilad-Bachrach, R., Navot, Y., and Tishby, N. (2005). Is feature selection still necessary? In *Subspace, Latent Structure and Feature Selection, Statistical and Optimization*. 32

Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM. 51

Pan, S. J., Ni, X., Sun, J.-T., Yang, Q., and Chen, Z. (2010). Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM. 48

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359. 49

Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Foundations and Trends in Optimization*, 1:127–239. 36

Paul, S., Magdon-Ismail, M., and Drineas, P. (2015). Feature selection for linear SVM with provable guarantees. In *Artificial Intelligence and Statistics (AISTATS)*. 63, 64, 155

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM. 50

Refaeilzadeh, P., Tang, L., and Liu, H. (2007). On comparison of feature selection algorithms. In *Proceedings of AAAI workshop on evaluation methods for machine learning II*, volume 3, page 5. 98, 155

Ribeiro, B., Silva, C., Vieira, A., Gaspar-Cunha, A., and das Neves, J. C. (2010). Financial distress model prediction using svm+. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–7. IEEE. 29

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,. 12

Satpal, S. and Sarawagi, S. (2007). Domain adaptation of conditional probability models via feature subsetting. In *Knowledge Discovery in Databases: PKDD 2007*, pages 224–235. Springer. 49

Serra-Toro, C., Traver, V. J., and Pla, F. (2014). Exploring some practical issues of svm+: Is really privileged information that helps? *Pattern Recognition Letters*, 42:40–46. 29, 125, 126, 131

Sharmanska, V., Hernández-Lobato, D., Miguel Hernandez-Lobato, J., and Quadrianto, N. (2016). Ambiguity helps: Classification with disagreements in crowdsourced annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2194–2202. 27

Sharmanska, V., Quadrianto, N., and Lampert, C. H. (2014). Learning to transfer privileged information. *Computer Vision and Image Understanding (submitted)*. 25

Shiao, H.-T. and Cherkassky, V. (2013). Svm-based approaches for predictive modeling of survival data. In *Proceedings of the International Conference on Data Mining (DMIN)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). 29

Smyth, G. K. (2004). Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3(1). 33

Song, L., Bedo, J., Borgwardt, K. M., Gretton, A., and Smola, A. (2007). Gene selection via the bahsic family of algorithms. *Bioinformatics*, 23(13):i490–i498. 34

Song, L., Smola, A., Gretton, A., Bedo, J., and Borgwardt, K. (2012). Feature selection via dependence maximization. *Journal of Machine Learning Research (JMLR)*, 13(1):1393–1434. 33, 59

Taylor, J., Sharmanska, V., Kersting, K., Weir, D., and Quadrianto, N. (2016). Learning using unselected features (lufe). In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. AAAI Press/International Joint Conferences on Artificial Intelligence. 64

Van't Veer, L. J., Dai, H., Van De Vijver, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., van der Kooy, K., Marton, M. J., Witteveen, A. T., et al. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536. 33

Vapnik, V. (2006). *Estimation of dependences based on empirical data*. Springer Science & Business Media. 8

Vapnik, V. and Izmailov, R. (2015). Learning using privileged information: Similarity control and knowledge transfer. *JMLR*, pages 2023–2049. 10, 23, 26, 61, 114, 125

Vapnik, V. and Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural Networks*, pages 544–557. 3, 8, 9, 10, 15, 16, 17, 22, 28, 61, 80, 114, 115, 125

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999. 12

Verikas, A. and Bacauskiene, M. (2002). Feature selection with neural networks. *Pattern Recognition Letters*, 23(11):1323–1335. 157

Weston, J., Elisseeff, A., Schölkopf, B., and Tipping, M. (2003). Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research (JMLR)*, 3:1439–1461. 4, 35, 57

Xu, C., Tao, D., and Xu, C. (2013). A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*. 51, 134

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35. 97, 102

Zhou, Z.-H. and Li, M. (2005). Semi-supervised regression with co-training. In *IJCAI*, volume 5, pages 908–913. 52

Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2004). 1-norm support vector machines. In *Neural Information Processing Systems (NIPS)*. 36