Model Combination in Multiclass Classification

by

Samuel Robert Reid

B.S., University of New Mexico, 2000

M.S., University of Colorado, 2003

M.S., University of Colorado, 2005

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2010

This thesis entitled:
Model Combination in Multiclass Classification
written by Samuel Robert Reid
has been approved for the Department of Computer Science

Michael C. Mozer

Prof. Greg Z. Grudic

Prof. Richard H. Byrd

Prof. James H. Martin

Prof. François G. Meyer

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Reid, Samuel Robert (Ph.D., Computer Science)

Model Combination in Multiclass Classification

Thesis directed by Prof. Michael C. Mozer

Multiclass classification is an important machine learning problem that involves classifying a pattern into one of several classes, and encompasses domains such as handwritten character recognition, protein structure classification, heartbeat arrhythmia identification and many others. In this thesis, we investigate three issues in combining models to perform multiclass classification. First, we demonstrate that ridge regularization is essential in linear combinations of multiclass classifiers. Second, we show that when solving a multiclass problem using a combination of binary classifiers, it is more effective to share hyperparameters across models than to optimize them independently. Third, we introduce a new method for combining binary pairwise classifiers that overcomes several problems with existing pairwise classification schemes and exhibits significantly better performance on many problems. Our contributions span the themes of model selection and reduction from multiclass to binary classification.

Acknowledgements

Many people are responsible for my productive and enjoyable learning experience at the University of Colorado and for the ideas and results in this dissertation research. My first machine learning class was taught by Mike Mozer in Spring 2001, and his infectious excitement sparked my interest in machine learning. Mike's insight and intuition make him an excellent teacher and an invaluable research advisor. Mike improved every aspect of this dissertation, and has always pushed me to strive for the best. Many thanks to Greg Grudic for his support and assistance with the ideas in the early stages of this dissertation research, particularly concerning the proposal and the chapter on regularized linear models in stacked generalization. Thanks to Rich Caruana, Richard Byrd, James Martin and François Meyer for their service on the committees. Many thanks to PhET Interactive Simulations for funding my education and providing hardware for the experimental studies. Thanks to the reviewers of the 2009 Multiple Classifier Systems conference proceedings for providing recommendations for our publication. Thanks to all the groups that made their data sets publicly available, including the UCI Repository and the Turing Institute at Glascow, Scotland. This work was also supported by NSF Science of Learning Center grant SBE-0542013 (Garrison Cottrell, PI), and by NSF BCS-0339103, BCS-720375, and SBE-0518699. Thanks to my parents Robert and Kathy who exemplify encouragement and support. Many thanks to my wonderful wife Ingrid for feedback on rough drafts and practice talks and for her endless encouragement and confidence in me.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

1.1 Introduction

Multiclass classification is a ubiquitous machine learning problem, encompassing diverse domains such as handwritten letter recognition, heartbeat arrhythmia monitoring, image segmentation, protein binding site prediction and many others. Several algorithms have been proposed and studied for solving multiclass classification problems [50, 75, 30]. Early work in machine learning focused on using a single classifier for each problem, but recent work has shown the advantage of training many classifiers for each problem and combining their predictions [101, 62, 11, 89, 31, 48, 17]. Multiclass classifiers can be combined directly using voting, averaging or other linear or nonlinear combination techniques to improve classification performance. Another approach is to combine binary classifiers to solve multiclass problems, with each binary classifier solving a different subproblem. In this thesis, we investigate three previously unexplored issues in model combination for multiclass classification: we study regularization in linear combinations of multiclass classifiers (Chapter 3), we explore model selection in binary subproblems (Chapter 4) and we present a novel pairwise classification approach (Chapter 5). Our contributions span the related themes of model selection and solving multiclass problems with binary classifiers. Each contribution of this thesis is self-contained so that each of the contribution chapters 3-5 can be read independently of the rest of the dissertation. Some methodological descriptions and background materi-

als are duplicated to facilitate independence of the chapters. This introductory chapter provides a high level overview of the contributions and common themes between them. Chapter 2 provides background and literature review of classifier combination. Chapters 3-5 contain the contributions of this thesis, and Chapter 6 concludes with a summary of our work and recommendations for future work.

1.1.1 Regularized Linear Models in Stacked Generalization

In Chapter 3, we focus on combining multiclass classifiers using a linear combination function, under the framework of stacked generalization. We use many types of classification algorithms with many different hyperparameter settings, then use their predictions on unseen (validation) data in order to train a linear combination function. We study several regularization techniques and show that proper regularization of the combiner function is essential to improve performance. The standard linear least squares regression can be regularized with an L2 penalty (ridge regression), an L1 penalty (lasso regression) or a combination of the two (elastic net regression). We study a linear model that applies one weight per classifier prediction rather than one weight per classifier, which allows classifiers to focus on different implicit subproblems corresponding to different classes. This chapter was published in the conference proceedings and presented at the Multiple Classifier Systems conference in 2009 [84].

1.1.2 Model Selection in Binary Subproblems

Some machine learning algorithms were designed for solving binary classification problems (e.g. support vector machines or AdaBoost). A popular and effective way to solve a multiclass problem using binary classifiers is to transform the multiclass classification problem into a set of binary classification problems, solve them using binary classification algorithms and combine the predictions of the binary classifiers. For example, the one-vs-all method creates one binary subproblem for each class, separating

it from the remaining classes. Another common reduction method is called pairwise classification (or all-pairs), in which each subproblem separates one class from another class. In Chapter 4, we focus on the particular issue of how to perform model selection when reducing a multiclass classification problem to a set of binary subproblems. As opposed to monolithic approaches that solve the entire multiclass problem at once and are regularized as a unit, techniques that reduce multiclass problems to binary subproblems introduce the new flexibility to perform model selection in each subproblem. In Chapter 4, we perform experimental studies that show that shared-hyperparameter selection is more effective than independent optimization because subproblems typically share similar structure. Conversely, we construct a synthetic data set with differing decision boundary shapes, and show that independently optimizing subproblem models is more effective in that case. We also rule out several confounding factors such as selection of incorrect models due to insufficient validation data, or a mismatch between the validation and test metrics.

1.1.3 Probabilistic Pairwise Classification

Pairwise classification (all-pairs) [38, 40, 56] has been criticized because it relies on classifiers that must make predictions over distributions that were unseen during training[51, 23]. In Chapter 5, we address this issue with a new pairwise classification technique called probabilistic pairwise classification (PPC) that uses probabilistic predictions for pairwise discrimination and weights each pairwise prediction with an estimated probability that the instance belongs to the pair. The technique is derived from the Theorem of Total Probability, and relies only on the assumption that each instance is assigned exactly one label. Our method is conceptually simpler and easier to implement than other pairwise classification methods that incorporate and produce probabilities. Experimental studies indicate that our proposed technique performs better than other pairwise classification techniques on real world data sets, at the cost of

increased computational demands. We also show that our proposed method is capable of improving multiclass classifiers—for example, the random forest classifier is capable of making multiclass predictions, but embedding it within PPC as a binary classification algorithm can improve performance.

1.2 Thesis Statement

The contributions described in the preceding sections can be summarized in our three-part thesis statement:

- Ridge regularization is essential in linear combinations of multiclass classifiers.
- When reducing a multiclass classification problem to a set of binary classification problems, it is more effective to constrain subproblems to share hyperparameters than to optimize each subproblem independently.
- Probabilistic pairwise classification has many advantages over previously proposed pairwise classification approaches because it explicitly estimates probability of membership in each pair.

Each element of this thesis statement is corroborated by experimental studies over many real world data sets, comparison with similar methodologies, investigation of behavior under synthetic data sets, and discussion of related theories. Before presenting our contributions, we provide an overview of the supervised learning framework and related work in model selection and combination for supervised classification.

Chapter 2

Background

This chapter introduces supervised classification (Section 2.1), and the related concepts of model selection (Section 2.2) and model combination (Section 2.3). We discuss the two methodologies for performing model combination, namely *commensurate model combination*, in which each classifier provides an estimate of the same target function (Section 2.3.1) and *complementary model combination*, in which each classifier provides a different part of the solution (Section 2.3.2).

2.1 Supervised Classification

Given a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i), i = 1..N\}$, where $\boldsymbol{x}$ is a d-dimensional vector comprised of continuous (numerical) and/or discrete (nominal) parameters (attributes), the goal is to induce a model $\hat{y}(\boldsymbol{x})$ that minimizes a loss function $L(y, \hat{y})$ over the distribution of unseen data. Training and test points are assumed to be drawn i.i.d. (independent and identically distributed) from the same full joint probability distribution $q(\boldsymbol{z})$, where $\boldsymbol{z} = (\boldsymbol{x}, y)$. In nearly all real world problems, the full joint distribution is unknown; instead a finite number of observations $\mathcal{D}$ are generated by the full joint distribution. Generative techniques seek to model the full joint probability distribution for the purpose of making predictions whereas discriminative techniques directly model the class boundaries. In this thesis, we restrict our focus to discriminative modeling techniques.

A classifier, also known as a model, hypothesis or expert, is a function $\hat{y}(\boldsymbol{x}) : \mathbb{K}^d \Rightarrow \Omega$, where each dimension in $\mathbb{K}$ refers to a continuous or discrete attribute, also known as a variable or feature, and Ω is the set of possible class labels $\Omega = \{\omega_1 ... \omega_k\}$. A classification algorithm is a function that produces a classifier, given a training data set and a set of hyperparameters: $\hat{f}(\mathcal{D}, \boldsymbol{\theta}) = \hat{y}(\boldsymbol{x}) : \mathbb{K}^{d \cdot N} \Rightarrow (\mathbb{K}^d \Rightarrow \Omega)$. The hyperparameters $\boldsymbol{\theta}$, also known as learning parameters, are settings used to govern the learning algorithm, for instance, regularization hyperparameters in linear regression, the learning rate or momentum parameters in artificial neural network models or the $\{C, \gamma\}$ learning parameters used in Gaussian support vector machines (SVMs). Each classification algorithm entails a search through an implicit or explicit hypothesis space to find the preferred model structure and/or parameters [75]. For classification problems, we consider classification algorithms that predict probability distributions or confidences, also known as as discriminant values or support values.

2.2 Model Selection

Model selection refers to the identification of a suitable algorithm and/or set of hyperparameters for a particular data set. We refer to a model as any function that produces a classifier when evaluated on a given labeled training data set. Typically, a model is the combination of a learning algorithm (e.g. SVM or AdaBoosted decision stumps) and an associated set of hyperparameters (also known as learning parameters or metaparameters) such as $\{\gamma, C\}$ for Gaussian SVM or {number of iterations} for AdaBoosted decision stumps. Parameters, as opposed to hyperparameters, refer to data components of the trained classifier rather than the mechanism used to obtain it. If the algorithm is selected before seeing any labeled training data, then model selection simply refers to the search over learning hyperparameters for the given classification algorithm. Search techniques such as grid search or binary search are often used to identify an appropriate set of hyperparameters, given an algorithm and a labeled training

set. While parametric model selection methods have been proposed, such as minimum description length [47], it is more common to estimate the performance of a model by evaluation on validation data over many resamplings [50].

2.3 Model Combination

Classifier combination techniques, also known as model combination techniques, ensemble methods, committees or opinion pools, combine predictions from multiple models in order to make the final prediction. The underlying classifiers are commonly referred to as the *base classifiers*. Formally, the prediction is taken to be a function of the base classifier predictions $\hat{y}(\boldsymbol{x}) = f(\hat{y}_1(\boldsymbol{x}), ..., \hat{y}_L(\boldsymbol{x}))$, where $\hat{y}(\boldsymbol{x})$ is the ensemble prediction, $\hat{y}_i(\boldsymbol{x})$ is the prediction from the ith base classifier (out of L base classifiers), and $f(.)$ is the combination function. Multiple classifier systems can also allow the combination function to depend on the input vector $f(.) = f(\hat{y}_1(\boldsymbol{x}), ..., \hat{y}_L(\boldsymbol{x}), \boldsymbol{x})$, though this technique is less common.

The base classifiers $\hat{y}_i$, $i = 1..L$ may be constructed to each solve the same problem (*commensurate models*), or each base classifier may be trained to solve a different subproblem (*complementary models*). In bagging, for example, each of the base classifiers is trained on a resampling of the original problem (which is an approximation or perturbation of the original problem), and the base classifiers are combined using voting [11]. In boosting and error-correcting output coding, different subproblems are constructed to be solved by the different base classifiers; when the base classifiers solve different problems, the combination function is typically more complex than averaging or voting. In this thesis, Chapter 3 focuses on a method that combines many commensurate models. Chapters 4-5 focus on methods that split multiclass classification problems into a number of disjoint complementary models.

2.3.1 Commensurate Model Combination

In commensurate model combination, each classifier is an approximation of the same (target) function. In Chapter 3, we use many classification algorithms and hyperparameter settings to approximate the target function, then combine the models using a linear regression technique. This section provides further background in commensurate model combination.

2.3.1.1 Generating Base Classifiers

There are many techniques for generating different base models for classification or usage in an ensemble. A classification algorithm A takes a dataset $\mathcal{D}$ and produces a classifier C. Therefore, perturbations can be made to either the algorithm A or the dataset $\mathcal{D}$ in order to obtain different classifiers. A central issue in generating different base classifiers is algorithm stability; an algorithm is said to be unstable if a minor change to A or $\mathcal{D}$ produces a large change in the resulting classifier C [11]. Unstable classification algorithms are vital for obtaining improved ensembles under some types of perturbations. In this thesis, we mainly focus on the usage of different algorithms and associated hyperparameters to generate diversity among the base classifiers. Furthermore, base classification algorithms may include one or more of the following approaches (for example, a random forest classifier can be used as a base classifier). Many publications have focused on isolating and evaluating the particular dimensions for perturbation described below; an important line of future research would be to combine many of these types of perturbations to attempt to maximize diversity and thus classification performance.

Modifying the Training Data Set Modification of the original training data set $\mathcal{D}$ to produce a perturbed dataset $\mathcal{D}'$ typically produces a different classifier. The main techniques for producing a perturbed dataset are to: (1) subsample examples, (2)

subsample feature sets and (3) generate novel data.

Subsampling When there is enough data so that classifiers can be trained on disjoint subsets of the training data set, the resulting classifiers can exhibit reduced correlation and thus increased predictive performance [19].

Resampling Another method is to train several classifiers using a single classification algorithm on bootstrap samples of the original dataset; this technique is called bagging (for bootstrap aggregating) [11]. Friedman and Popescu [37] point out that there is nothing inherently advantageous about the bootstrap; different problems will benefit from different resampling strategies in general. Boosting assigns higher weight to difficult examples to produce classifiers that correctly label the difficult examples.

Different Feature Sets For each classifier, a random subset of features from the original problem are selected. This technique is known as the random subspace method, and has been explored with decision tree classifiers [52]. This technique fails for problems in which all features are required to attain sufficiently high classification accuracy [61].

Generating Novel Data A new dataset can be constructed by synthesizing novel data based on the original data distribution. For example, in DECORATE, novel data is generated and different classifiers are trained using that data with different label assignments [72]. This technique can also be used productively in semi-supervised methods, in which there is ample unlabeled data; in this case, no data synthesis step is necessary.

Adding Noise Similarly, changing the class labels assigned to examples is one way of producing different classifiers. For some datasets and classification algorithms, this technique has shown to improve performance [14, 13].

Modifying the Classification Algorithm Aside from modifying the training data set to produce different classifiers, it is also possible to modify the algorithm used to train the classifier.

Different Learning Algorithms Different learning algorithms entail different inductive biases [75]; by using different learning algorithms, different classifiers are obtained.

Different Learning Parameters Many classification algorithms have hyperparameters that must be tuned in order to match properties of the dataset at hand. Using different settings for these hyperparameters generally results in the production of different classifiers. For example, backpropagation neural networks can have varying momentum, learning rates, hidden neurons, hidden layers, etc.

Randomized Algorithms Some classification algorithms rely on internal randomization to produce individual classifiers. For example, random forests [14] construct each classifier by sampling i.i.d. from a specified distribution over decision tree classifiers. This randomization increases the diversity of the classifiers without significantly decreasing their accuracy, and subsequently, the ensemble has improved classification accuracy.

Perturbation of the Loss Function When the loss function for a classification algorithm is perturbed, the result is a novel classification algorithm. Freidman and Popescu [37] discuss perturbation of the loss function to produce diverse classifiers.

2.3.1.2 Advantages of Commensurate Model Combination

In 2000, Dietterich [26] identified three distinct problems that can be overcome by classifier combination:

(1) The statistical problem: Several classifiers may yield the same validation set accuracy. Combining predictions from such classifiers produces smoothing in the output space, and reduces the risk of choosing a single poor classifier.

(2) The computational problem: Many classification algorithms entail a search that is susceptible to local optima. For example, optimal training is known to be

NP-hard for neural networks [57] and decision trees [8]; therefore suboptimal techniques are typically used instead (e.g. greedy search for decision trees or gradient descent for neural networks). By combining the results of multiple runs, the ensemble may produce a better prediction than any of the constituent classifiers.

(3) The representational problem: The combination of classifiers may produce a decision boundary that is impossible to represent with any single base classifier (for instance, consider that two right triangular decision boundaries may be combined to produce a single square decision boundary). Many modern classification algorithms are known to be universal, i.e. able to construct an arbitrary set of decision boundaries. For example, a neural network with a single hidden layer and a sufficient number of hidden neurons can represent any continuous function [53]. However, when trained on a finite amount of training data, classification algorithms are only capable of exploring a finite region of the classifier space. It is possible to expand the hypothesis space by asserting that the final decision boundary is a combination of individual classifier boundaries.

Techniques for combining classifier outputs range from simple (e.g. majority voting) to complex (e.g. nonlinear combination functions). For any commensurate multiple classifier technique to succeed, the base classifiers must exhibit nonrandom accuracy and some degree of independence.

2.3.1.3 Independent Classifier Combination

When classification algorithms are applied to overlapping (or equal) data samples or share similar inductive biases, the resulting classifiers will probably make similar predictions. Nevertheless, it is useful to analyze the simpler case of independent classifiers in order to understand the mechanism and benefits of classifier combination [63, 92].

When classifiers have identical accuracy and make independent predictions on a binary classification problem, then the probability of a correct classification is given by the probability mass function of the binomial distribution [48][1] :

$$P_{maj} = \sum_{k=\lfloor L/2 \rfloor + 1}^{L} \binom{L}{k} p^k (1-p)^{L-k} \tag{2.1}$$

Table 2.1 shows an example of the performance of ensembles as a function of base classifier accuracy and number of members in the group. These curves are also depicted in Figure 2.1 for better-than-random classifiers. The behavior as the number of classifiers increases can be characterized based on the individual classifier accuracy [21] (here we assume a Bayes optimal rate of 1.0):

(1) $p > 0.5$ When each classifier predicts the correct class with probability greater than 0.5, then as classifiers are added, the ensemble accuracy increases monotonically. Furthermore, as the number of classifiers approaches infinity, the accuracy of the ensemble approaches 1.

(2) $p < 0.5$ When each classifier predicts the correct class with probability less than 0.5, then as classifiers are added, the ensemble accuracy decreases monotonically. Furthermore, as the number of classifiers approaches infinity, the accuracy of the ensemble approaches 0.

(3) $p = 0.5$ When each classifier predicts the correct class with probability equal to 0.5, then as classifiers are added, the ensemble accuracy remains at 0.5.

For instance, in Table 2.1, the row for which $p = 0.8$ shows that after adding just 8 more independent classifiers, the accuracy of the group under majority vote increases to 0.98, a relative accuracy gain of more than 20%. It is possible to improve upon this benefit by constructing negatively correlated classifiers rather than uncorrelated classifiers [69].

[1] Here we assume a strict majority is necessary for a correct group prediction; ties are considered an error.

However, classifiers will typically be at least somewhat correlated, and the benefits will fall short of those identified in the table. It is also possible to characterize the behavior of independent classifiers for multiclass classification problems. In this case, sums are made over appropriate subsets of the multinomial probability mass function.

Table 2.1: Ensemble accuracy is depicted for a given number of base classifiers L, and individual accuracy p.

p	$L = 1$	$L = 3$	$L = 5$	$L = 7$	$L = 9$
0.51	0.510	0.515	0.519	0.522	0.525
0.6	0.600	0.648	0.683	0.710	0.733
0.7	0.700	0.784	0.837	0.874	0.901
0.8	0.800	0.896	0.942	0.967	0.980
0.4	0.400	0.352	0.317	0.290	0.267

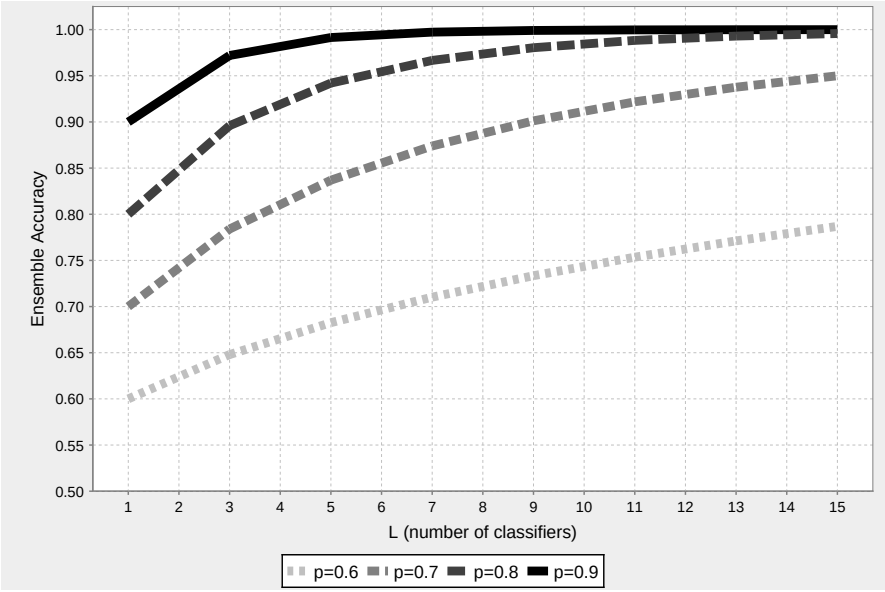


Figure 2.1: Ensemble accuracy is depicted as a function of the number of independent classifiers, for a specified individual classifier accuracy.

2.3.1.4 Dependent Classifier Combination

When classifiers are trained using overlapping data or are produced by algorithms that share a similar inductive bias, the classifiers are likely to be at least somewhat

correlated. Clemen and Winkler [20] and Jacobs [58] show that there is an upper limit on the precision of information that can be obtained by consulting dependent experts. In particular, they show that m dependent experts are worth the same as k independent experts, where $k <= m$. This result is obtained under the assumption that experts provide unbiased point estimates for a regression variable, and that the joint probability of the experts' errors is normally distribution with mean zero and covariance matrix Σ. The combiner function is Bayes' rule, treating the expert predictions as observations in a new space (see Section 2.3.1.5). Therefore, combining dependent classifiers can still lead to benefits, though not as significant as would be attained by combining independent classifiers.

2.3.1.5 Probabilistic Model Combination

Early work on model combination in the statistics community focused on predictions from human experts, so many of the ideas in this literature are referred to as expert combination or opinion pooling, and the combiner function is called the decision maker. Preliminary work involved the search for a combination technique that satisfied normative (axiomatic) constraints, which are intuitive properties one might expect to find in a combination technique [79]. However, when just a few seemingly reasonable axioms are asserted, this search can be shown to be unsatisfiable, suffering from impossibility theorems in the same vein as Arrow's impossibility theorem [4]. Further work showed the advantage of viewing the set of individual predictions as data, and many authors have advocated the usage of Bayes' theorem on the meta-level data in the so-called supra-Bayesian framework [76, 58]. A few authors have investigated particular forms for the likelihood function, with hyperparameters that are inferred from data [46, 60]; however, the computational demands of these techniques are large and grow rapidly with the addition of classifiers. Before discussing probabilistic models, we address a common misconception that Bayesian model averaging behaves as a model

combination technique.

Bayesian Model Averaging is not Model Combination Bayesian model averaging marginalizes over different models

$$p(y|\mathcal{D}) = \sum_{k=1}^{L} p(y|M_k, \mathcal{D})p(M_k|\mathcal{D})$$

That is, classifier predictions are weighted by the probability that the model is correct, given the dataset. This technique appears to be a natural method for combining models, and some have been tempted to treat it as such [28]. However, as is pointed out [74, 46], Bayesian model averaging is not model combination, but rather a form of soft model selection. As more data is observed, Bayesian model averaging will assign more and more probability to the most probable model, and as the amount of data tends to infinity, weights for all other models approach zero. Bayesian marginalization is appropriate when the base classifiers M_i are mutually exclusive and exhaustive and the true data generating model is one of the models under consideration. In the overproduce-and-choose paradigm, neither of these requirements will generally be true.

Linear Opinion Pools A natural way of combining expert opinions is with a linear combination $\hat{y}(\boldsymbol{x}) = \sum_{k=1}^{L} \alpha_k y_k(\boldsymbol{x})$, called a linear opinion pool. Typically the weight for an expert is chosen as a function of his/her (perceived) accuracy for the particular domain. In 1981, McConway [71] showed that the linear opinion pool is the only combination scheme that satisfies the marginalization property. Consider the the probability distribution for a number of output variables $\boldsymbol{y} \in \mathbb{R}^m$. Combination schemes that satisfy the marginalization property produce the same group decision whether the marginal predictions are combined or the joint predictions are combined then marginalized over. Marginalization is an intuitive property, however, there are two main problems. First, there is no foundational approach for how the weights should be allocated. Many studies have been devoted to find theoretically and empirically motivated means for choosing expert weights [44], but this is still an open problem. Second,

axiomatic approaches have been criticized for their failure on particular straightforward examples. Lindley shows, in particular, that the marginalization property ignores important information [68]. Another argument for linear opinion pools was made by Genest and Schervish [45], who showed that the supra-Bayesian paradigm reduces to a linear opinion pool when the decision maker asserts a value only for the mean of the marginal distribution of expert predictions.

Supra-Bayesian Methods In his 1971 thesis, Morris showed that the appropriate way to combine predictions from multiple experts under uncertainty is to treat the predictions as data in a new feature space, and to use Bayes' rule to update the decision maker's prior distribution [76]. This technique was later referred to as the *modeling approach*, since it is necessary to model the joint predictive distribution of all experts, and later called the *supra-Bayesian approach* [58]. In the supra-Bayesian framework, predictions from base classifiers are treated as data by the meta-classifier, and Bayes' rule is used to compute the ensemble decision. The ensemble decision is therefore

$$p(y|H, P_1, ..., P_m) \propto p(P_1, ..., P_m|y, H)p(y|H) \tag{2.2}$$

where $P_i = p_i(y|H)$ is the i^{th} classifier's probability distribution for y when its knowledge is H. That is, the posterior probability distribution for the class label given the combiner's knowledge H and the predictions of each base model $p(y|H, P_1, ..., P_m)$ is proportional to the product of the likelihood $p(P_1, ..., P_m|y, H)$ of the experts' decisions given the true class, the combiner's knowledge H and the prior distribution $p(y|H)$ of the combiner.

The main problem with supra-Bayesian methods was pointed out by Morris [77] and Jacobs [58]; the difficulty is in specifying the likelihood function for the experts' opinions given the data; the likelihood function must account for individual classifier calibration as well as classifier correlation. Furthermore, evaluating the likelihood function can be computationally prohibitive since it is a high-dimensional distribution. Some re-

cent research has focused on specifying a parametric model for the likelihood function, and learning it by observing expert behavior on validation data. The discriminative analog of the supra-Bayesian method is stacked generalization, discussed in Section 2.3.1.6.

Graphical Models Graphical models represent the relationships between inputs, classifiers and predictions using a probabilistic model described by a graph.

Garg et al. describe a simple graphical model for performing classifier combination [43]. The classifiers are assumed to be conditionally independent of one another given the ensemble classification. The Bayesian network is a tree of depth 2, with the root being the ensemble classification and the leaves being the individual classifiers.

Ghahramani and Kim [46] describe a graphical model for combining classifiers, called Bayesian classifier combination (BCC). They start with a simple graphical model which assumes classifiers are independent (IBCC) and show how it can be embellished to account for dependencies between classifiers. They describe a model called the enhanced Bayesian classifier combination model, which uses separate graphs for easy and difficult data points. A Markov network is added in the dependent BCC model to model the correlations between classifiers directly. Finally, they combine dependent and enhanced models to obtain a Markov network with the easy/difficult graph separation. Empirical results are demonstrated for satellite, UCI digit and DNA datasets. The advantage of BCC is shown to be greater when combining multiple different base classifiers trained on the entire training set rather than combining different base classifiers trained on disjoint subsets of the training set.

2.3.1.6 Stacked Generalization

In 1992, Wolpert introduced a general multiple classifier technique called stacked generalization [101], known informally as stacking. The predictions of individual classifiers are viewed as data in a new meta-feature space, and any classification algorithm

can be trained on this new problem. Wolpert also points out that the meta-feature space can be augmented with the original inputs or with other relevant measures. Wolpert restricted his focus to regression problems, but Ting and Witten later demonstrated that stacked generalization can be used for classification problems as well, when using probabilistic predictions (confidence-levels) outputted by the classifiers [95].[2] Since stacked generalization allows the usage of any method as the meta-classifier, it is possible to simulate many discriminative multiple classifier techniques using stacked generalization, including model selection, majority voting, weighted averaging and nonlinear combination functions [101, 91]; in fact, Wolpert refers to model selection by cross-validation as "just a (relatively uninteresting) special case of stacked generalization, corresponding to an extraordinarily dumb level 1 generalizer". Wolpert also showed that any classifier combination technique can use embedded cross-validation to efficiently re-use all training data as validation data for training the combiner function. In this thesis, we refer to stacked generalization as any discriminative technique that views the predictions of base classifiers as data in a new feature space. Below, we formalize the idea of stacked generalization.

Given a set of L classifiers $\hat{y}_i(\boldsymbol{x}|\boldsymbol{\theta}), i = 1..L$, a meta-level dataset[3] is constructed by aggregating the predictions of each classifier on a validation dataset $\mathcal{D}_{val}$, and combining them with the known labels. Table 2.2 shows an example of a meta-level dataset. The meta-level dataset is used as input to the combiner classification algorithm. In order to make a prediction on a new data point, each classifier makes its prediction, and these predictions are input to the combiner function. The original problem of mapping $\mathbb{K}^d \Rightarrow \Omega$ is therefore passed through an intermediate space with dimension equal to the number of classifiers $\mathbb{K}^d \Rightarrow \mathbb{K}^L \Rightarrow \Omega$.

[2] Ting and Witten [95] mistakenly report that Wolpert's original formulation addressed a classification task rather than a regression task.

[3] Wolpert referred to this as the level-1 dataset to emphasize that stacked generalization can be extended to higher levels

Table 2.2: A meta-level stacking dataset for three classes, two base classifiers and five examples, using Wolpert's model [101].

$\hat{y}_0$	$\hat{y}_1$	y
ω_1	ω_1	c_1
ω_1	ω_1	c_1
ω_2	ω_3	c_3
$\boldsymbol{\omega_1}$	$\boldsymbol{\omega_1}$	$\boldsymbol{c_1}$
ω_1	ω_1	c_2

Ting and Witten [95] addressed the classification problem, using probability distributions as inputs to the combiner function rather than class predictions. In this case, each classifier outputs normalized discriminant values for each of the possible classes. The set of discriminant values is passed as an input vector to the combiner function, which produces the final ensemble prediction. In Ting and Witten's stacked generalization representation, the original problem of mapping $\mathbb{K}^d \Rightarrow \Omega$ becomes the new problem: $\mathbb{K}^d \Rightarrow \mathbb{K}^{L \cdot c} \Rightarrow \Omega$. Table 2.3 shows an example of stacking using Ting and Witten's representation of the meta-feature space. Ting and Witten also pointed out the need for regularization for the combiner function. They recommend using a simple combiner function, such as multi-response linear regression (MLR) in order to prevent overfitting. MLR trains a linear model for each of the classes, and at prediction time chooses the class for which the linear model outputs the highest value.

Experiments in Stacked Generalization In this section, we summarize some prominent experiments with stacked generalization.

Wolpert, 1992 [101] When Wolpert introduced stacked generalization in 1992, he investigated a series of toy problems for a 1-dimensional piecewise linear function. This toy problem used a single classifier, and included the original features in the meta-feature space, so that the meta-feature space was 2-dimensional. Wolpert's second numerical experiment involved the NETTalk text-to-speech program. The inputs are 7

Table 2.3: A meta-level stacking dataset for three classes, two base classifiers and five examples, using Ting and Witten's model [95]. ω_i refers to the ith discriminant value, $\hat{y}_i$ is the ith model and c_i is the predicted class. A single row (e.g. shown in bold) corresponds to a meta-data point.

$\hat{y}_0$			$\hat{y}_1$			
ω_0	ω_1	ω_2	ω_0	ω_1	ω_2	y_i
0.8	0.1	0.1	0.6	0.3	0.3	c_1
0.9	0.0	0.1	0.7	0.2	0.1	c_1
0.2	0.7	0.1	0.0	0.1	0.9	c_3
0.7	**0.1**	**0.2**	**0.7**	**0.2**	**0.1**	$\boldsymbol{c_1}$
0.9	0.1	0.0	0.6	0.4	0.0	c_2

letters, and the output is a 21-dimensional output characterising an English phoneme. The stacked generalization experiment combined three classifiers that each made predictions for a single letter. Wolpert points out that the purpose of the experiment wasn't to improve on the performance of existing methods, but to identify whether stacked generalization could productively combine separate pieces of incomplete information.

Breiman, 1996 Breiman investigated the Housing and Ozone datasets in his 1996 paper [12]. 50 CART$^{\text{TM}}$(Classification And Regression Trees)regression trees were used as the base classifiers in the first experiment. CART pruning yields subtrees, so each of the 50 trees were nested subtrees. When using a regularized error function for the combiner, Breiman found that a small number of combiner values had nonzero weights; only a small number of models were being combined in the stacked regression. Breiman also recommends using ridge regression to more efficiently regularize.

Ting and Witten, 1999 [95] Ting and Witten investigate 10 problems: Led24, Waveform, Horse, Credit, Vowel, Euthyroid, Splice, Abalone, Nettalk(s) and Coding. They use C4.5, NB (a re-implementation of Naive Bayes) and IB1, a variant of the K-Nearest Neighbor algorithm as the base learners. They used C4.5, NB, IB1 and MLR as combiner functions. No model selection is done at the base or meta-levels. They show that MLR using confidence-level predictions beats model selection

by cross validation in all datasets, significant at over two standard errors. They also show that stacked generalization with MLR has eight significant wins and two losses (with insignificant differences) against majority vote. They also mention employing a multilayer perceptron as the combiner function, and report that it had the same error rate as MLR while taking approximately 1700 times longer to train.

2.3.1.7 Bi-Level Stacking

Schaffer [87] extended stacked generalization to *bi-level stacking* in which the meta-classifier is trained on a feature set that includes classifier predictions as well as the original input features. This technique has been re-invented a few times in the machine learning community; Chan and Stolfo [19] described this model as the *class-attribute combiner* in their study of scalability for classifier combination techniques. In 2006, Torres-Sospedra et al. called this technique *stacked generalization plus* in their comparative study of combination techniques [96]. Bi-level stacking has the potential to address the tradeoff between classifier selection and classifier combination. Since the combiner function has access to the original inputs, it can select the classifier (or a combination of classifiers) known to be more accurate in that region of the input space. Poor results have been reported for bi-level stacking in the above references, which may be due to the following reasons:

- Insufficient search for an appropriate combiner function, including parameter tuning as well as algorithm tuning. For example, Schaffer's original investigation of bi-level stacking combined three base level classifiers with a decision tree, rule set induction or a neural network; no parameter tuning was performed for the neural network (including a static stopping criterion of 1000 epochs).

- Increased dimension of the meta-feature space. When using probability predictions from base classifiers as well as the original input features, the meta-feature

space has dimension $L \cdot c + d$, which may lead to a sparsely-populated meta-data space in which it is easy to overfit.

2.3.1.8 Theory for Classifier Combination

In a 2001 technical report, Breiman says (in a section titled "My Kingdom for Some Good Theoretical Explanations") [15] "The area of ensemble algorithms is filled with excellent empirical results, but the understanding of how they work is a scarce commodity." In the statistics literature, axiomatic approaches are theoretically motivated, but proposed axioms inevitably entail stronger and more restrictive implications than intended, and tend to exhibit counterintuitive behavior on simple examples [100]. Winkler [100] says "The problem I have with the axiomatic approach (and this does not apply to Morris) is that it is sometimes done in the spirit of a search for a single, all-purpose, "objective" combining procedure. Such a search is futile..." The modeling approach (also known as the supra-Bayesian approach) in which expert predictions are taken as data and combined under Bayes' rule is theoretically well-founded, but to apply this approach in practice requires modeling a high-dimensional joint likelihood distribution over expert predictions which is often difficult or impossible [58]. In this section, we identify other proposed theoretical models for explaining ensemble behavior.

Chebychev's Inequality Breiman gives a bound for the generalization error of random forests in [14] based on Chebychev's inequality. The margin function for a random forest is defined to be $m(\boldsymbol{x}, y) = p_\Theta(y(\boldsymbol{x}, \Theta) = y) - \max_{j \neq y} p_\Theta(h(\boldsymbol{x}, \Theta) = j)$, where Θ is drawn i.i.d. from a probability distribution used to guide induction of the decision trees. Informally, the margin function measures the expectation value of how many more votes are cast for the correct class than for the next highest voted class. By defining the strength of a set of classifiers to be the expectation value of its margin function $s = E_\mathcal{D}[m(\boldsymbol{x}, y)]$, it is possible to put an upper bound on the generalization error of the random forest in terms of the strength and correlation of the classifiers.

Chebyshev's inequality is $p(|x - \mu| \geq k\sigma) \leq \frac{1}{k^2}$, where σ^2 is the finite sample variance. Informally, this inequality states that in any probability distribution, nearly all values are close to the mean. The generalization error of a voted ensemble is given as the probability that the margin function is negative: $E^* = p_{\mathcal{D}}(m(\boldsymbol{x}, y) < 0)$. Therefore, Chebyshev's inequality bounds the ensemble error as $E^* \leq \frac{\sigma_m}{s^2}$, where σ_m is the variance of the margin function. Breiman goes on to show that for random forests, or indeed any set of classifiers whose construction is guided by i.i.d. random sampling, the variance of the margin function can be written as $\sigma_m = \bar{\rho}(1 - s^2)$, where $\bar{\rho}$ is the mean value of the classifier correlation. Therefore an upper bound for the generalization error is given by $E^* \leq \bar{\rho}(1 - s^2)/s^2$. Breiman points out that this bound is likely to be a loose upper bound, but that it explains the success of particular types of ensembles and motivates the search for accurate and diverse classifiers.

Bias-Variance-Covariance Decomposition of Ensemble Error The error of a single classifier for a regression problem can be decomposed into contributions from bias and variance [50]. Some work has been done to generalize this result to classification problems [59]. The error in a single classifier prediction under the squared error loss can be written as $E(\boldsymbol{x}) = \sigma_\epsilon^2 + [E\hat{y}(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})]^2 + E[\hat{y}(\boldsymbol{x}) - E[\hat{y}(\boldsymbol{x})]]^2$. The first term on the right hand side is the irreducible (Bayes optimal) error. The second term is the square of the bias and the final term is the variance. In the regression case for ensembles, a decomposition in terms of the average bias, average variance and average covariance is given as [98]: $E^* = E[v\bar{a}r(\boldsymbol{x})/L + (1 - 1/L)c\bar{o}v(\boldsymbol{x}) + b\bar{i}as(\boldsymbol{x})^2] + \sigma_\epsilon$. A similar result is shown for classifier ensembles. These results show the tradeoff between bias, variance and covariance in the ensemble setting.

Added Error of the Ensemble Tumer and Ghosh [97] describe a mathematical framework that gives the relationship between classifier correlation and ensemble error for the average combiner by approximating the decision boundaries as linear near the Bayes optimal decision boundary. They show that when the classifiers are uncor-

related, the reducible error is reduced by a factor of L; that is, $E_{ens} = \frac{1}{L}E_k$. For the case of correlated classifiers, a similar analysis shows that $E_{ens} = \frac{1+\rho(L-1)}{L}E_k$, where $\bar{\rho}$ is the average classifier correlation.

Diversity Measures One potential route to understanding ensemble behavior is by identification of an appropriate diversity measure. By combining a formal definition of diversity with a model for the benefit due to diversity, we may obtain an understanding of ensemble behavior. The above theoretical models have depicted the ensemble accuracy in terms of the classifier correlation; however, recent work has searched for other diversity measures with desirable properties. Diversity measures can be roughly categorized as either pairwise or non-pairwise measures. For pairwise measures, the pairwise diversity is averaged across all $L(L-1)/2$ pairs of classifiers in the ensemble. A simple pairwise measure known as the disagreement measure is the probability that two classifiers disagree on their decisions

$$d = p(y_{i-} + y_{j+}) + p(y_{i+} + y_{j-}) \tag{2.3}$$

where y_{i-} indicates that classifier i is incorrect and y_{i+} indicates that classifier i is correct. Nonpairwise measures include the entropy measure, Kohavi-Wolpert variance, measure of interrater agreement, measure of difficulty, generalized diversity, coincident failure diversity, as well as others [62]. The entropy measure estimates the amount of disagreement in the votes, with a maximum when the votes are very nearly split. The Kohavi-Wolpert variance is based on an error decomposition formula for a single classifier, and has been shown to differ from the averaged disagreement measure by a coefficient [62].

2.3.1.9 Ensemble Pruning

Ensemble pruning is the process of selecting a subset of classifiers to participate in the group decision [108, 107]. There are two main reasons to prune ensembles: to reduce

the computational requirements (both storage space and prediction time) and to increase performance. However, a complete search through all possible subsets of classifiers is prohibitive; if there are L classifiers, then there are $\sum_{k=1}^{L} \binom{L}{k}$ subsets of classifiers. For example, for ensembles with 10 classifiers, there are 1023 possible subsets, but for 100 classifiers, there are over 10^{30} classifier subsets. Therefore suboptimal search techniques are typically employed. Flexible combination techniques, such as stacked generalization (described in Section 2.3.1.6) with a trainable nonlinear combiner function have the potential to learn which classifiers are most appropriate to combine for a problem, but are prone to overfitting. Ensemble pruning, on the other hand, is a discrete technique that attempts to discard as many irrelevant or redundant classifiers as possible. The algorithms under investigation in Chapter 3 are an example of ensemble pruning. Many other techniques have been proposed for pruning classifier ensembles; we discuss the most prominent techniques below.

Generalized Ensemble Method Many of the early ensemble approaches, and ensemble pruning approaches, come from the neural network ensemble literature. Perrone and Cooper [80] studied the case of neural networks for regression, noting that in the regression case it is possible to derive a closed form solution for weights in a weighted combination model. They also address the issue of network pruning; the weights for a linear combination of outputs is computed using the method of Lagrange multipliers to be $\alpha_i = \frac{\sum_j C_{ij}^{-1}}{\sum_k \sum_j C_{kj}^{-1}}$, where C_{ij} is the symmetric correlation matrix $E[f_i(\boldsymbol{x}) f_j(\boldsymbol{x})]$. The authors point out that linearly dependent rows or columns indicate dependent classifiers, and they recommend subsampling the population of neural networks to assure that C has full rank.

Ensemble Pruning by Semidefinite Programming A recent proposed technique for ensemble pruning reformulates the problem as a semidefinite program-

ming problem [107], where the objective function is

$$min(x^T Gx)$$

$$s.t. \sum_i x_i = k$$

$$x_i \in \{0, 1\}$$

The matrix G is constructed to reflect classifier errors on the diagonal and correlation off the diagonal. An ad hoc scheme for measuring the correlation of classifiers on validation data is used; the authors report that they tried several ad hoc measures and arrived at similar results. The authors used this technique to prune ensembles produced by Adaboost, and report that they are able to maintain the same accuracy with a dramatic reduction in the number of classifiers, in some cases obtaining improved predictive accuracy.

Overproduce and Choose All ensemble pruning techniques overproduce a set of classifiers and choose which will participate in the ensemble decision. However, in the literature, the *overproduce and choose* paradigm typically refers to a set of heterogeneous classifiers (different classifier types built from a variety of learning algorithms and learning parameters). The idea is that usage of a wide variety of algorithms will increase the probability that there will be good models that are amenable to combination, and diversity can be fostered by using different algorithms rather than subsampling from the dataset.

In 2000, Sharkey and Sharkey [93] reviewed work that focused on constructing many classifiers, then trying to identify the most appropriate subset to participate in the ensemble decision [80, 78, 49], referring to them collectively as *test and select* methods. This idea was further refined and generalized by Roli et al. in 2001 [86], and dubbed the *overproduce and choose* paradigm. In the overproduce phase, many classifiers are trained on the training data, using different algorithms, learning parameters, feature subsets or resamplings of the dataset. Next, classifiers are selected in order to optimize

the accuracy of the ensemble; this requires choosing classifiers that are individually accurate, but also different from one another. Roli et al. categorize selection rules as heuristics, diversity measures, clustering and search methods. By analogy with feature selection, we identify the first three techniques as filter methods, since they can be computed without evaluation of the ensemble. Search methods, on the other hand, are similar to wrapper methods, since estimates of the ensemble performance are used to guide the search, and the search technique can be "wrapped around" any combiner function.

Wrapper Methods for Ensemble Selection Wrapper methods for ensemble selection evaluate classifier subsets together with the combiner function and overall performance metric in order to choose classifier subsets. The wrapper method can be contrasted with the filter method, which uses a statistical measure in order to perform subset selection. This definition of filter and wrapper methods can be applied to feature selection, ensemble selection and meta-feature selection. Ensemble Selection from Libraries of Models [17] is a wrapper method for ensemble selection that uses a forward stepwise approach to construct the ensemble from a library of models. Classifiers are selected with replacement, which produces a weighting of the final linear combination function. Since this is a wrapper method, it is possible to use any metric to tune the ensemble. Other stepwise selection procedures are possible such as backward selection, hillclimbing and best-first approaches [61].

2.3.1.10 No-Free-Lunch in Multiple Classifier Systems

The No-Free-Lunch Theorem [102] states that any two machine learning algorithms will have identical performance averaged across all problems. We may be tempted to assume that this theorem merely applies to base-level classifiers, and that model selection or classifier combination methods will be able to circumvent the No-Free-Lunch Theorem. However, as described by Schaffer [88], meta-learning methods (including

cross-validation) are subject to the same kind of bias as base-level classifiers. When available, we should use domain-specific knowledge from a problem to identify a suitable algorithm and model structure, including its inductive bias. When domain-specific knowledge is unavailable, we must acknowledge that the implicit or explicit biases entailed by our base- or meta-learning algorithms may be a poor match for the problem at hand.

The previously discussed methodologies focus on combining commensurate classifiers which each estimate the same (target) function. In the next section, we discuss methods for constructing and combining classifiers that each estimate different functions.

2.3.2 Complementary Model Combination

An alternative to commensurate model combination is to combine models that solve different parts of the overall problem. This includes the separate paradigms of sensor fusion [3], classifier selection [62] (including mixture of experts techniques) and techniques that reduce a multiclass problem to a set of binary classification problems. In these cases, the classifiers are no longer estimating the same target function, but are providing complementary components to the whole solution. In this thesis, we study methods for solving multiclass classification problems by reducing them to a set of binary classification problems (Chapters 4-5). In this section, we provide introductory and background information on various techniques that are used to solve multiclass classification problems with binary classifiers.

2.3.2.1 One-vs-All

One of the simplest and most widely used techniques for reducing a multiclass problem to a set of binary subproblems is known as the one-vs-all reduction (OVA), also known as unordered class binarization [40] or one-vs-rest (or 1vr) [105]. Using

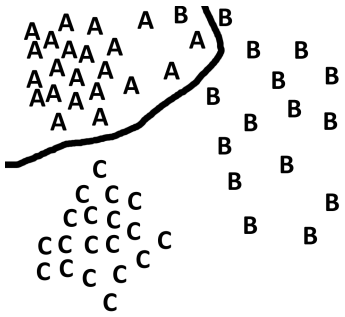


Figure 2.2: Illustration of an A-vs-BC decision boundary in a 2D, 3-class example of the One-vs-All reduction.

the one-vs-all reduction, a k-class classification problem is decomposed into k binary classification subproblems, one for each class (see Figure 2.2). In the ith subproblem, the classifier is trained to distinguish whether the instance belongs to class i or not. At prediction time, the classifier with the highest output is chosen (alternatively, voting or Hamming decoding can be used, as in error-correcting output coding). There is some disagreement in the literature about the terminology for the one-vs-all reduction; Rifkin and Klautau [85] use the term one-vs-all to indicate winner-take-all with continuous outputs (i.e. choosing the classifier with the maximum output); other research such as Beygelzimer et al. [5] refer to one-vs-all as it is used with Hamming decoding (e.g. discrete outputs as in error-correcting output coding [27], and randomizing over the selected classes). Here, we use the term OVA to refer to continuous winner-take-all one-vs-all and we refer to the discrete version as OVA with Hamming decoding. Rifkin and Klautau studied the one-vs-all technique, and compared it to other methods for reducing multiclass to binary and to other SVM techniques that provide direct optimization on the entire multiclass problem [85]. Their main thesis is that it is essential to perform model selection and that under appropriate model selection, one-vs-all tends to perform as accurately as other multiclass SVM methods. Rifkin and Klautau's technique for model selection is to choose one set of hyperparameters for all subproblems, rather

than trying to optimize each subproblem independently or trying to optimize differing hyperparameters for all subproblems jointly. Rifkin and Klautau don't explicitly state that they use the shared-model paradigm for model selection, but it is implied since one set of regularization hyperparameters is reported for each multiclass problem.

2.3.2.2 All-Pairs

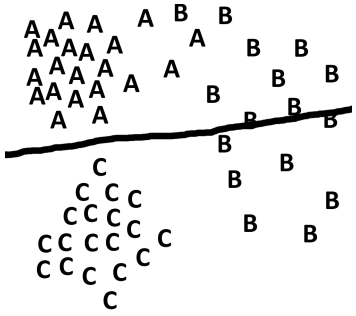


Figure 2.3: Illustration of an A-C decision boundary in a 2D, 3-class example of the All-Pairs reduction.

In the all-pairs reduction, also known as pairwise classification[51], all-vs-all (or AVA)[85], round-robin classification[40] and 1-against-1 (or 11)[105]), a k-class classification problem is decomposed into $\frac{k(k-1)}{2}$ problems, one for each pair of classes (see Figure 2.3). At prediction time, each binary classifier votes for one class, and the class with the most votes is selected as the multiclass prediction. Note that in contrast to continuous winner-take-all one-vs-all, the original all-pairs methodology ignores predicted probabilities or confidence predictions from each binary classifier, instead using only a discrete vote from each, though it is possible to use the all-pairs encoding with a decoding function other than Hamming decoding in Loss Based Decoding. Friedman shows that Bayes optimal binary classifiers combine to produce a Bayes optimal multiclass classifier, and therefore each binary subproblem can be solved independently and as accurately as possible [38].

The Bayes optimal decision is given by

$$\hat{y}(\boldsymbol{x}) = \underset{k \in K}{\operatorname{argmax}}\, p(\omega = k | \omega \in K, \boldsymbol{x})$$

where K is the set of possible labels, ω is the true label, $\boldsymbol{x}$ is the input feature vector and $\hat{y}$ is the predicted label. This is equivalent to

$$\hat{y}(\boldsymbol{x}) = \underset{k \in K}{\operatorname{argmax}} \sum_{i \in K} 1(\frac{p_k}{p_k + p_i} > \frac{p_i}{p_k + p_i})$$

where $1(x) = 1$ if x is true and 0 otherwise. This reduces to:

$$\hat{y}(\boldsymbol{x}) = \underset{k \in K}{\operatorname{argmax}} \sum_{i \in K} 1(p(\omega = k | \omega \in \{i, k\}, \boldsymbol{x}) > p(\omega = i | \omega \in \{i, k\}, \boldsymbol{x}))$$

Therefore, given reliable $p(\omega = k | \omega \in \{i, k\}, \boldsymbol{x})$, binary reduction under the All-Pairs reduction is equivalent to the true Bayes optimal decision. Note that this analysis assumes that $p(\omega = k | \omega \in \{i, k\}, \boldsymbol{x})$ can be determined exactly for each subproblem, whereas in practice, it would be difficult or impossible to accurately obtain this distribution given a finite sample size.

Friedman argues that one-vs-all subproblems must be tuned simultaneously, since the outputs from each model must be commensurate with one another.

2.3.2.3 Other Techniques in Reducing Multiclass to Binary

While one-vs-all and all-pairs are the most widely studied and employed techniques for reducing multiclass to binary, they are only two cases within the more general framework known as loss-based decoding, which is itself an extension of error-correcting output coding. Though we focus our experimental studies on the one-vs-all and all-pairs reductions, we also describe these other frameworks, since they must also address the issue of model selection and they are generalizations of the methods under our study.

Error-Correcting Output Coding The error-correcting output coding (ECOC) framework was proposed by Dietterich and Bakiri in 1995 [27]. This scheme is named for its similarity to error correcting codes in information theory, with the analogy that

the instance's class is a message to be transmitted, and that error correcting codes are employed to encode the message in order to make the transmission (or classification) more tolerant of errors. ECOC requires all classes to appear in each subproblem, and allows an arbitrary specification of how classes are reassigned to subproblems. The data structure used to specify how classes are reassigned to subproblems is called the coding matrix. For example, for a 5-class problem, in a particular subproblem, classes $1, 3, 4$ might be assigned to the positive indicator class. This would correspond to a row in the coding matrix equal to $\{+1, -1, +1, +1, -1\}$. At prediction time, each subproblem classifier votes for or against membership in the positive indicator class, and the class with the most votes is selected as the multiclass prediction, breaking ties randomly.

The number of unique and nontrivial binary splits (codewords) for a set of k classes is $2^{k-1} - 1$ [62]. Of these splits, k correspond to the one-per-class dichotomies. The other splits are different binary problems constructed from the original class labels. Dietterich and Bakiri [27] proposed using as many of these dichotomies as computationally feasible in order to improve the multiclass prediction. At prediction time, each base classifier is evaluated, and the class label with the minimum Hamming distance to the predicted codeword is used as the multiclass prediction. Dietterich and Bakiri recommend using all possible dichotomies when the number of classes is 7 or less; when there are more classes, a random sampling of dichotomies is typically used. The error-correcting output coding technique explicitly designs classifiers to focus on different composite subproblems.

Loss Based Decoding In 2000, Allwein et al. generalized the ECOC framework to to Loss Based Decoding, which (a) accounts for continuous instead of discrete classifier outputs and (b) allows some subproblems to optionally ignore some of the classes in the data set [2]. Incorporating continuous output makes it possible to represent the One-vs-All technique in the Loss Based Decoding framework (as we show below), and allowing subproblems to omit subsets of data points makes it possible to

represent the All-Pairs technique. Loss Based Decoding was further generalized by Crammer and Singer in 2000 to Continuous Output Coding [22], in which each class in a subproblem has some continuous weight $w \in \mathbb{R}$ rather than $w \in \{-1, +1\}$ as in ECOC or $w \in \{-1, 0, +1\}$ as in Loss Based Decoding.

One-vs-All in Loss-Based Decoding While loss-based decoding [2] was shown to encompass a variety of previous methods, including voted pairwise classification and Hamming-decoding one-vs-all, a representation for continuous winner-take-all one-vs-all has been lacking. In this section, we show that using the loss function $L(z) = (1 - z)^2$ in one-vs-all loss-based decoding yields the same predictions as continuous winner-take-all one-vs-all. Our experimental studies use this result in implementing the continuous one-vs-all method.

2.4 Conclusion

This concludes our review of background material in supervised learning, model selection and commensurate and complementary model combination. The next chapter investigates linear combinations of commensurate multiclass classifiers.

Chapter 3

Regularized Linear Models in Stacked Generalization

Chapter Abstract

Stacked generalization is a flexible method for multiple classifier combination; however, it tends to overfit unless the combiner function is sufficiently smooth. Previous studies attempt to avoid overfitting by using a linear function at the combiner level. This chapter demonstrates experimentally that even with a linear combination function, regularization is necessary to reduce overfitting and increase predictive accuracy. The standard linear least squares regression can be regularized with an L2 penalty (ridge regression), an L1 penalty (lasso regression) or a combination of the two (elastic net regression). In multiclass classification, sparse linear models select and combine individual predicted probabilities instead of using complete probability distributions, allowing base classifiers to specialize in subproblems corresponding to different classes. Our experimental studies show that the dense ridge regularization is much more effective than the sparse lasso regularization.

3.1 Introduction

Multiple classifier systems combine the predictions of many classifiers to produce the ensemble prediction [26, 86, 62]. Simple techniques such as voting or averaging can improve predictive accuracy by combining diverse classifiers [14]. More sophisticated ensemble techniques, such as ensemble selection, train a combination function in order

to account for the strengths and weaknesses of the base classifiers and to produce a more accurate ensemble model [18].

Stacked generalization is a flexible method for multiple classifier systems in which the outputs of the base-level classifiers are viewed as data points in a new feature space, and are used to train a combiner function [101][1] . Ting and Witten [95] applied stacked generalization to classification problems, and found that a multiple response linear combiner outperformed several nonlinear combiners for their problem domains and selection of base classifiers. They also showed that in classification problems, it is more effective to combine predicted posterior probabilities for class membership than class predictions.

Caruana et al. [18] evaluated stacked generalization with logistic regression with thousands of classifiers on binary classification problems, and reported that stacked generalization tended to overfit, resulting in poor overall performance. In this chapter, we remedy this overfitting and improve overall generalization accuracy through regularization.

Regularization attempts to improve predictive accuracy by reducing variance error at the cost of slightly increased bias error—this is known as the bias-variance tradeoff [50]. In this chapter, regularization is applied to linear stacked generalization for multiclass classification in order to improve predictive accuracy. In particular, ridge regression [50], lasso regression [50], and elastic net regression [109] are used to regularize the regression model by shrinking the model parameters. Lasso regression and some settings of elastic net regression generate sparse models, selecting many of the weights to be zero. This result means each class prediction may be produced by a different subset of base classifiers.

In our experiments, many classification algorithms and many parameter settings

[1] Wolpert introduced the ideas of internal cross-validation and trainable combiner functions together in his article; we use the term 'stacked generalization' to refer to the latter.

are used to build a library of base models as in Caruana et al. [17]. We also perform resampling at the ensemble level in order to obtain more statistically reliable estimates of performance without the expense of retraining base classifiers. We look at the correspondence between performance on subproblems and overall classifier performance, and interpret the behavior of sparse linear models in stacked generalization.

This chapter is organized as follows: Section 3.2.1 formally describes stacked generalization, including usage of indicator functions to transform the multiclass problem into several regression problems and the class-conscious extension, StackingC. Section 3.2.2 describes linear regression, ridge regression, lasso regression and elastic net regression, which are used to solve the indicator subproblems in stacked generalization. Section 3.3 describes empirical studies that indicate the advantage of regularization. Section 3.4 discusses the results and Section 3.5 concludes with a summary and future work.

3.2 Model

3.2.1 Stacked Generalization

Given a set of L classifiers $\hat{y}_i(\boldsymbol{x}|\boldsymbol{\theta})$, $i = 1..L$, the predictions of each classifier on a validation dataset $\mathcal{D}_{val}$ are aggregated and combined with the known labels to create a meta-level training dataset $\mathcal{D}'_{val}$. The combiner function is then trained on this meta-level validation dataset. Given a test point, the predictions of all base-level classifiers are combined to produce a new data point $\boldsymbol{x}'_i$. The combiner function is evaluated at the new data point $\boldsymbol{x}'_i$, and its output is taken as the ensemble output. Formally, the ensemble prediction of stacked generalization is given by $sg(\boldsymbol{x}) = c(y_{11}(\boldsymbol{x}), ..., y_{1K}(\boldsymbol{x}), ..., y_{L1}(\boldsymbol{x}), ..., y_{LK}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the test point, c is the classifier combiner function and y_{lk} is the posterior prediction of the l^{th} classifier on the k^{th} class. Following Ting and Witten, a regression function can be used at the meta-

level by constructing one regression subproblem per class with an indicator function; this is the so-called multi-response linear regression (MLR) formulation [95]. At prediction time, the class corresponding to the subproblem model with the highest output is taken as the ensemble output. A more general discussion of reducing classification to linear regression problems is given in Hastie et al. [50].

The most general form of stacked generalization includes all outputs from all base classifiers. To simplify the problem, Seewald recommends using a class-conscious approach in which each indicator model is trained using predictions on the indicated class only, called StackingC [90]. Formally, the StackingC class prediction is given by $sc(\boldsymbol{x}) = \text{argmax}_k\, r_k(y_{1k}(\boldsymbol{x}), ..., y_{Lk}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the test point, $k = 1..K$ is an index over classes, r_k is the regression model for the indicator problem corresponding to the k^{th} class and y_{lk} is the posterior prediction of the l^{th} classifier for the k^{th} class. Ting and Witten report that StackingC gives comparable predictive accuracy while running considerably faster than stacked generalization [95], and Seewald also reports increased predictive accuracy. Based on these arguments, the experiments in this chapter use StackingC rather than complete stacked generalization. With Ting and Witten's MLR, the predictive model is $\hat{p}_j(\boldsymbol{x}) = \sum_{i=1..L} w_{ij} y_{ij}(\boldsymbol{x})$, where $\hat{p}_j(\boldsymbol{x})$ is the predicted probability for class c_j, w_{ij} is the weight corresponding to classifier y_i and class c_j, and $y_{ij}(\boldsymbol{x})$ is the i^{th} classifier's output on class c_j. An example of this model is illustrated in Figure 3.1.

3.2.2 Linear Models and Regularization

The least squares solution is given by $\hat{y} = \hat{\boldsymbol{\beta}}\boldsymbol{x}$, where $\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$. Here $\hat{\boldsymbol{\beta}}$ is the vector of model parameters determined by the regression, N is the number of training data points, y_i is the true output on data point i, x_{ij} is the jth feature of the ith data point, and p is the number of input dimensions for the problem. In StackingC, the features are predicted probabilities from base classifiers. When the

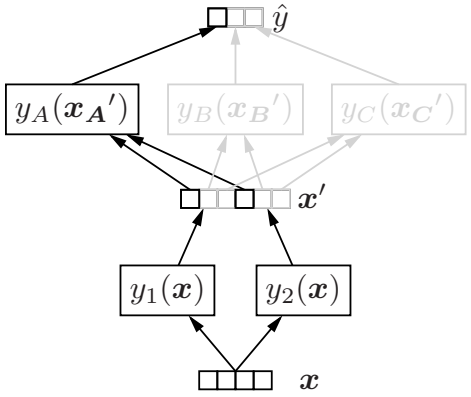


Figure 3.1: Example illustration of the StackingC and Multi-Response Linear Regression model used in our experiments for a 4-dimensional input vector in a 3-class classification problem. The prediction for class c_A is highlighted.

linear regression problem is underdetermined, there are many possible solutions. This situation can occur when the dimensionality of the meta-feature space L is larger than the effective rank of the input matrix (at most N), where L is the number of classifiers and N is the number of training points. In this case, it is possible to choose a basic solution, which has at most m nonzero components, where m is the effective rank of the input matrix.

Ridge regression augments the linear least squares problem with an L2-norm constraint: $P_R = \sum_{j=1}^{p} \beta_j^2 \leq s$. This has the effect of conditioning the matrix inversion problem by adding a constant k to the diagonal: $\boldsymbol{\beta} = (X^T X + kI)^{-1} X^T \boldsymbol{y}$. There is a one-to-one correspondence between s and k [50].

Lasso regression augments the linear least squares problem with an L1-norm constraint: $P_l = \sum_{j=1}^{p} |\beta_j| \leq t$. The L1-norm constraint makes the optimization problem nonlinear in y_i, and quadratic programming is typically used to solve the problem. Unlike ridge regression, lasso regression tends to force some model parameters to be identically zero if the constraint t is tight enough, thus resulting in sparse solutions.

Zou and Hastie describe a convex combination of the ridge and lasso penalties

Table 3.1: Data sets used in the experimental studies, and their properties

Dataset	*Attributes*	*Instances*	*Classes*
balance-scale	4	625	3
glass	9	214	6
letter	16	4000	26
mfeat-morphological	6	2000	10
optdigits	64	5620	10
sat-image	36	6435	6
segment	19	2310	7
vehicle	18	846	4
waveform-5000	40	5000	3
yeast	8	1484	10

called the elastic net [109]. The penalty term is given by $P_{en}(\boldsymbol{\beta}|\alpha) = (1-\alpha)\frac{1}{2}||\boldsymbol{\beta}||_{l2}^2 + \alpha||\boldsymbol{\beta}||_{l1}$, where $0 \leq \alpha \leq 1$ controls the amount of sparsity. The elastic net is particularly effective when the number of predictors p (or classifiers in StackingC) is larger than the number of training points n. The elastic net performs groupwise selection when there are many correlated features (unlike the lasso, which instead tends to select a single feature under the same circumstances). When there are many excellent classifiers to combine, their outputs will be highly correlated, and the elastic-net will be able to perform groupwise selection.

3.3 Experimental Studies

For empirical evaluations, we selected publicly available datasets with numerical attributes and $k \geq 3$ classes. Table 3.1 indicates the datasets and relevant properties. For the 26-class *letter* dataset, we randomly subsampled a stratified selection of 4000 points.

Approximately half the data points (with stratified samples) in each problem are used for training the base classifiers. The remaining data is split into approximately equal disjoint segments for model selection at the ensemble level (e.g. stacking training data or select-best data) and test data, again in stratified samples. In a real-world

application, the base classifiers would be re-trained using the combination of base-level data and validation data once ensemble-level hyperparameters are determined, but this additional training is not done in this study due to the expense of model library construction.

Previous studies with $L \geq 1000$ classifiers obtain one sample per problem, with no resampling due to the expense of model library creation, such as in Caruana et al. [18]; we partially overcome this problem by resampling at the ensemble training stages. In particular, we use Dietterich's 5x2 cross-validation resampling [25] over the ensemble training data and test data. We use the Wilcoxon signed-rank test for identifying statistical significance of the results, since the accuracies are unlikely to be normally distributed [24].

We generate around 1000 classifiers, including neural networks, support vector machines, k-nearest neighbors, decision stumps, decision trees, random forests, and AdaBoost.m1 and bagging models.

3.3.1 Base Classifiers

We generate about 1000 classifiers for each problem. For each classification algorithm, we generate a classifier for each combination of the parameters specified below. All implementations are in Weka except for the Random Forest (R), for which we used the R port of the Breiman-Cutler code by Andy Liaw, available through CRAN.

(1) Neural Network decay={true, false} momentum={0.1, 0.5, 0.9}
learningRate={0.5, 0.75, 0.9} trainingTime={100, 500, 1000}
numHiddens={2, 4, 16, 32}

(2) Support Vector Machine (C-SVM) kernelType={linear, polynomial, rbf, sigmoid} coef0={-1, 1} cost={0.1, 1.0, 10, 100, 1000} degree={1, 2, 3}
eps={0.001, 0.01} gamma={0.1, 0.3, 0.8}

(3) K-Nearest Neighbor k={1, 2, 4, 16, 32, 64}

(4) Decision Stump

(5) Decision Tree (J48) binarySplits={true, false} confidenceFactor={0.25, 0.5, 0.75} reducedErrorPruning={false, true} unpruned={true, false}

(6) Random Forest (Weka) numTrees={1, 2, 30, 50, 100, 300, 500}

(7) AdaBoost.M1 numIterations={10, 50, 100, 500} classifier={J48 binarySplits={true, false}, Decision Stump}

(8) Bagging classifier={J48 binarySplits={true,false}} numBags={5, 10, 50}

(9) Random Forest (R) numTrees={1, 2, 30, 50, 100, 300, 500}

3.3.2 Ensemble Techniques

As a baseline for comparison, we select the best classifier as identified by accuracy on the held-out ensemble training set (*select-best*). We also compare our linear models to voting (*vote*) and averaging (*average*) techniques. The StackingC approaches are denoted *sg-linear*, *sg-ridge* and *sg-lasso*.

For the majority of our datasets, there are more linear regression attributes p (same as the number of classifiers L) than data points n (equal to the number of stacking training points, roughly $\frac{N}{4}$)[2] . To solve this underdetermined system without resorting to the typical ridge regularization solution, we choose a basic solution as implemented in the Matlab *mldivide* function, which provides a sparse solution based on the QR factorization.

In order to select the ridge regression penalty, we search over a coarse grid of $\lambda = \{0.0001, 0.01, 0.1, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$ using cross-validation, then use all validation data to train the ridge regression model with the selected penalty

[2] The *waveform*, *letter*, *optdigits* and *sat-image* datasets are exceptions.

Table 3.2: Accuracy of each model for each data set. Entries are averages over the 10 samples from Dietterich's 5x2 cross-validation at the ensemble level. Variances are omitted based on arguments in Demšar [24]. See Section 3.3 for a description of the methods and Section 3.4 for discussion.

Dataset	*select − best*	*vote*	*average*	*sg − linear*	*sg − lasso*	*sg − ridge*
balance-scale	**0.9872**	0.9234	0.9265	0.9399	0.9610	0.9796
glass	0.6689	0.5887	0.6167	0.5275	0.6429	**0.7271**
letter	0.8747	0.8400	0.8565	0.5787	0.6410	**0.9002**
mfeat-m	0.7426	0.7390	0.7320	0.4534	0.4712	**0.7670**
optdigits	0.9893	0.9847	0.9858	0.9851	0.9660	**0.9899**
sat-image	0.9140	0.8906	0.9024	0.8597	0.8940	**0.9257**
segment	0.9768	0.9567	0.9654	0.9176	0.6147	**0.9799**
vehicle	0.7905	0.7991	0.8133	0.6312	0.7716	**0.8142**
waveform	0.8534	0.8584	**0.8624**	0.7230	0.6263	0.8599
yeast	**0.6205**	0.6024	0.6105	0.2892	0.4218	0.5970

parameter. We use the Matlab implementation of ridge regression from the Matlab Statistics Toolbox. Parameters are selected by cross-validation for each subproblem rather than choosing a single λ for all subproblems. For example, the regularization hyperparameter for the first indicator problem λ_1 may differ from λ_2. For lasso regression, we use the LARS software by Efron and Hastie [32], and search over a grid of $fraction = 0$ to 1 (where $fraction$ is the proportion of the saturated coefficients) in increments of 0.01 to select the regularization penalty term by cross-validation for each subproblem. We search over a finer grid in *sg-lasso* than in *sg-ridge* since model selection is much more efficient in LARS. For the elastic net, we use the glmnet package written by Friedman, Hastie and Tibshirani and described in the corresponding technical report [36].

3.4 Results

The test set accuracies of all ensemble methods are shown in Table 3.2. Each entry in this table is an average over 10 folds of Dietterich's 5x2 cross-validation [25] over ensemble training/validation data. According to the pairwise Wilcoxon signed-

ranks test [24], ridge regression StackingC outperforms unregularized linear regression StackingC at $p \leq 0.002$. *Select-best* outperforms both unregularized and lasso regression StackingC at $p \leq 0.002$. Ridge regression StackingC outperforms *select-best* at $p \leq 0.084$, and has more wins than any other algorithm. On two problems, *select-best* outperforms all model combination methods. On all problems, *sg-linear* and *sg-lasso* perform less accurately than *sg-ridge*; this result suggests that it may be more productive to assign nonzero weights to all posterior predictions when combining several base classifiers. A possible explanation for the superiority of ridge regularization over lasso regularization is that lasso is often more effective when it is able to perform feature selection by discarding irrelevant inputs or inputs that are negatively correlated with the target values; however, the base classifiers are predominantly positively correlated with the target prediction values, so lasso regularization is forced to throw away good predictors. In other words, lasso regularization actively searches for uncorrelated input models; however, since all models are estimates of the same target value, the lasso attempts to combine correct models with incorrect models.[3] Further experimental studies would be necessary in order to validate this hypothesis.

To study the effect of regularization on each subproblem, we plot the root mean squared error for a particular indicator subproblem as a function of the regularization penalty hyperparameter. Computation of a reasonable composite value over all data sets is difficult due to incommensurability of the problems, so we restrict our focus to a particular subproblem. (Results are qualitatively similar for other subproblems.) Figure 3.2(a) shows the root mean squared error in the first subproblem in the *sat-image* dataset[4] . As the ridge penalty λ increases from 10^{-8} to 10^{3}, the error decreases by more than 10%. With such a small penalty term, the error at 10^{-8} roughly corresponds to the error that would be obtained by unregularized linear regression. For individual

[3] Thanks to Abhishek Jaiantilal for pointing out this explanation.

[4] The root mean squared error is used instead of the accuracy because the subproblem in multi-response is a regression problem.

subproblems, therefore, regularization dramatically improves performance.

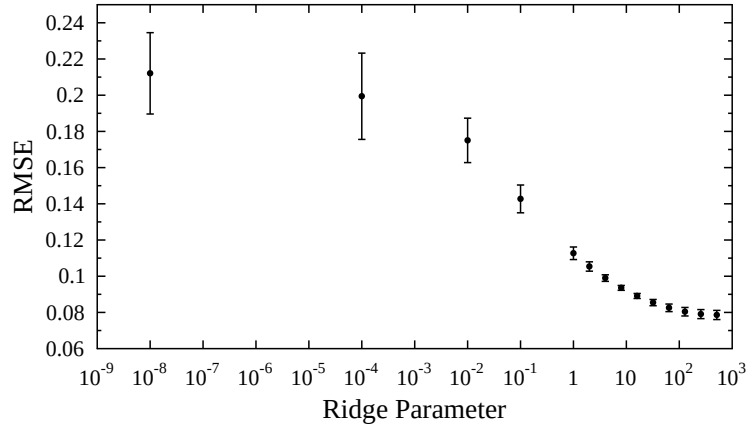
Figure 3.2(b) shows the overall accuracy of the multi-response linear regression with ridge regularization for the *sat-image* dataset. Regularization increases the accuracy of the overall model by about 6.5%, peaking around $\lambda = 10^3$. As the penalty is increased beyond 10^3 (not pictured), the accuracy decreases, reaching 0.24, the proportion of the predominant class, around $\lambda = 10^8$. Please note that in this figure, λ is the same for all subproblems.

Figure 3.2(c) shows the correlation between the accuracy of the overall multi-response linear regression system and the root mean squared error on the first subproblem. The fit is approximately linear, with $a = -0.408e + 0.957$, where a is the accuracy of the multiclass classifier and e is the RMSE of the classifier on the first indicator subproblem.

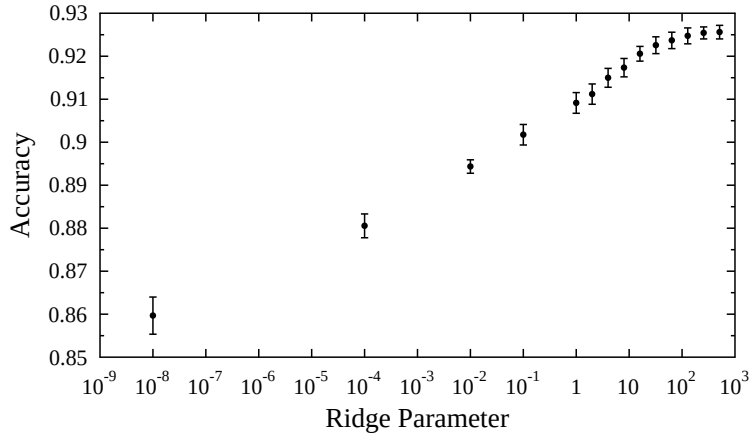
Figure 3.3(a) shows the overall accuracy of the multi-response linear regression system as a function of the penalty term for lasso regression for the *sat-image* problem. Standard errors over the 10 folds are indicated. As in the ridge regression case, λ is the same over all subproblems in this figure. The accuracy falls dramatically as the penalty increases beyond 0.2, stabilizing after $\lambda = 0.50$ at an accuracy of 0.24, the proportion of the predominant class.

In order to view the effect of the elastic net's mixing parameter α on the accuracy of the multi-response system, accuracy vs penalty curves are plotted in Figure 3.3(b) for $\alpha = \{0.05, 0.5, 0.95\}$. The $\alpha = 1.0$ curve indicated in Figure 3.3(a) is highly similar to the $\alpha = 0.95$ curve, and therefore omitted from Figure 3.3(b) for clarity. With a small penalty term $\lambda \leq 10^{-1}$, the curves are constant, and within one standard deviation of the *select-best* curve. As the penalty increases, the accuracy reaches a maximum that is dependent on α, with higher α values yielding higher accuracy at smaller penalty values. In this case, fine-tuned regularization increases accuracy by about 1.5%.

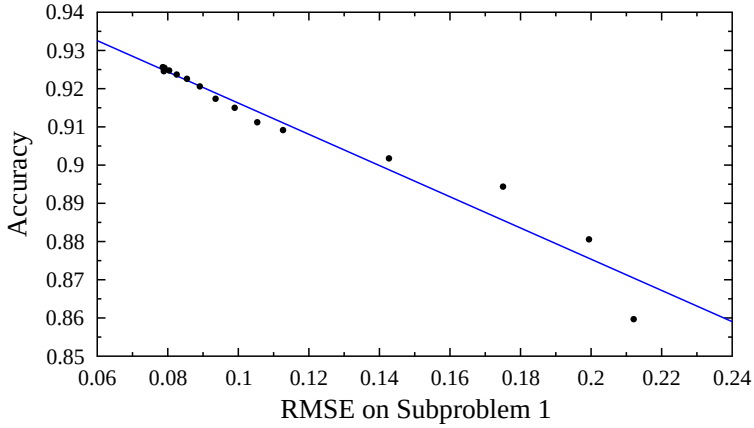
In ridge-regularized stacked generalization, all predictors are given some portion



(a)

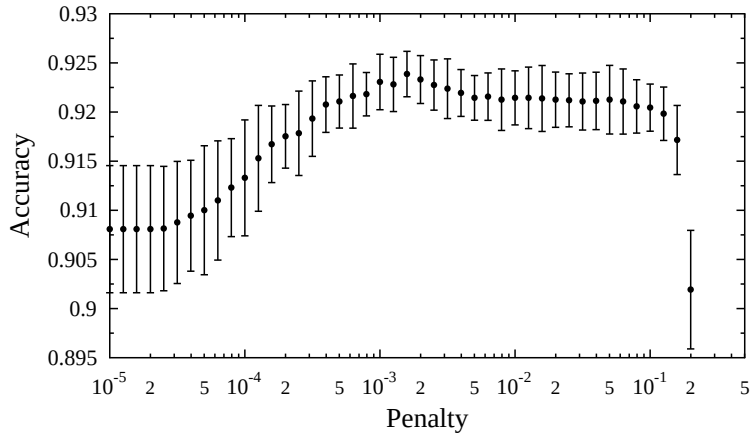


(b)

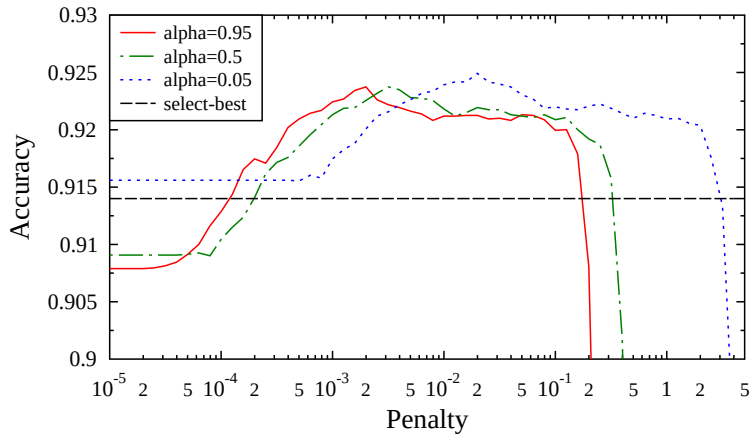


(c)

Figure 3.2: Figure 3.2(a) shows root mean squared error for the indicator problem for the first class in the *sat-image* problem. Figure 3.2(b) shows overall accuracy as a function of the ridge parameter for *sat-image*. The error bars indicate one standard deviation over the 10 samples of Dietterich's 5x2 cross validation. Figure 3.2(c) shows the accuracy of the multi-response linear regression system as a function of mean squared error on the first class indicator subproblem for ridge regression.



(a)



(b)

Figure 3.3: Overall accuracy of the multi-response linear regression system as a function of the penalty term for lasso regression for the *sat-image* problem, with standard errors indicated in Figure 3.3(a). Figure 3.3(b) shows accuracy of the multi-response linear regression system as a function of the penalty term for $\alpha = \{0.05, 0.5, 0.95\}$ for the elastic net for *sat-image*. The constant line indicates the accuracy of the classifier chosen by *select-best*. Error bars have been omitted for clarity, but do not differ qualitatively from those shown in Figure 3.3(a).

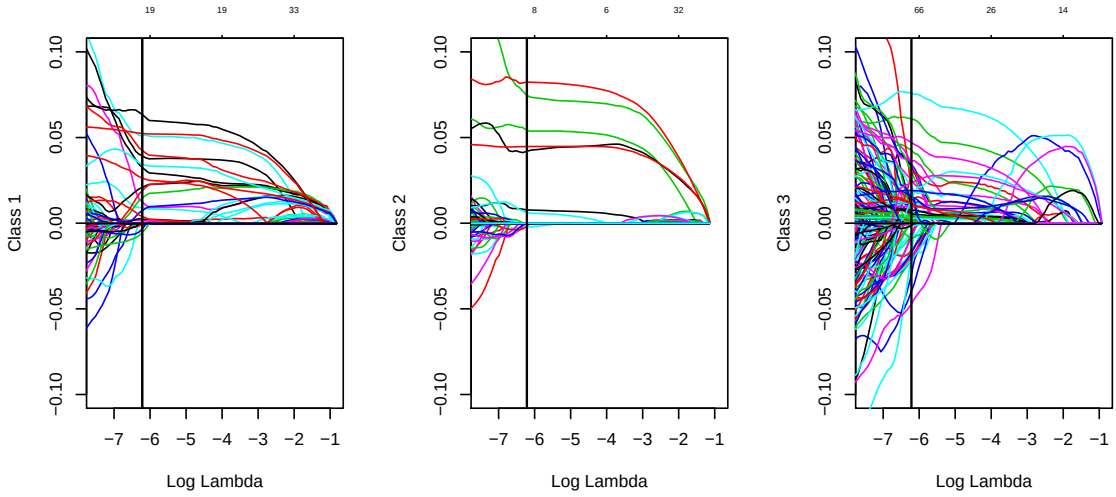


Figure 3.4: Coefficient profiles for the first three subproblems in StackingC for the *sat-image* dataset with elastic net regression at $\alpha = 0.95$, over a single partition of ensemble training and testing data.

of the total weight. In lasso regression and some settings of elastic net regression, it is possible to obtain a sparse model in which many weights are identically zero. The sparse model reduces computational demand at prediction time and makes it possible to identify a small subset of base classifiers and predictions that are responsible for making the overall prediction. Figure 3.4 shows the coefficient profiles for the *sat-image* dataset, for classes 1-3 with elastic net regularized StackingC with $\alpha = 0.95$. The optimal value of λ according to overall classification accuracy is shown as a vertical line. At $\lambda = \lambda_{opt}$, only 244 of the 999 classifiers are assigned weight for any of the subproblems. Table 3.3 shows the 6 classifiers with the highest total sum of weights for all classes. Sparse models obtained by L1-regularized linear regression can choose different classifiers for each class—that is, classwise posterior predictions are selected instead of complete classifiers. For instance, the classifier assigned the most total weight is $k = 1$-*nearest neighbor*, which contributes to the response for classes 3-6, but doesn't appear in the predictions for classes 1 or 2. The base classifier that makes the largest contribution to the *class-1* prediction is boosted decision trees run for 500 iterations.

Table 3.3: Selected posterior probabilities and corresponding weights for the *sat-image* problem for elastic net StackingC with $\alpha = 0.95$. Only the 6 models with highest total weights are shown here. *ann* indicates a single-hidden-layer neural network, and corresponding momentum, number of hidden units, and number of epochs in training.

$Classifier$	$class-1$	$class-2$	$class-3$	$class-4$	$class-5$	$class-6$	$total$
adaboost-500	0.063	0	0.014	0.000	0.0226	0	0.100
ann-0.5-32-1000	0	0	0.061	0.035	0	0.004	0.100
ann-0.5-16-500	0.039	0	0	0.018	0.009	0.034	0.101
ann-0.9-16-500	0.002	0.082	0	0	0.007	0.016	0.108
ann-0.5-32-500	0.000	0.075	0	0.100	0.027	0	0.111
knn-1	0	0	0.076	0.065	0.008	0.097	0.246

Thus each classifier is able to specialize in different class-based subproblems rather than being required to predict accurate probabilities for all classes.

3.5 Conclusion

Stacked generalization has a tendency to overfit; overfitting is even more likely when using many highly correlated, well-tuned models. In order to avoid overfitting and to improve prediction accuracy, it is necessary to perform regularization at the combiner level, even when using a linear combiner. Regularization can be performed by penalization of the L2 norm of the weights (ridge regression), L1 norm of the weights (lasso regression) or a combination of the two (elastic net regression). L1 penalties yield sparse linear models; in stacked generalization, this means selecting from a small number of classifier posterior predictions. We showed that ridge regularization was significantly more effective than the sparse linear lasso penalty, and suggested that this result is because many of the classifier outputs are well correlated with the target prediction value, causing lasso to select from both correct and incorrect models.

An interesting extension of this work would be to examine the full Bayesian solutions (under Gaussian and Laplacian priors for regularization), instead of the single-point maximum likelihood estimates implicit in the ridge (Gaussian prior) and lasso (Laplacian prior) regularizers. Other work could study additional regularization by

(a) selecting a single regularization hyperparameter for use in all subproblems or (b) constraining the weights to be non-negative for each subproblem.

Chapter 4

Model Selection in Binary Subproblems

Chapter Abstract

Model selection is critical in building effective classifiers. Support vector machines (SVMs) are a popular and effective classification technique, and model selection for Gaussian SVMs is performed by tuning the cost (C) and Gaussian width (γ) hyperparameters. SVMs were originally designed for binary classification, but have been extended and adapted to address multiclass classification problems. A simple, popular and effective method for solving multiclass classification problems using binary SVMs is to reduce a given multiclass classification problem to a set of binary classification subproblems, solve the binary subproblems, and combine the predictions from the binary classifiers. This raises the question of how to perform model selection; should the same model and hyperparameters be used on all subproblems, or should subproblems be tuned independently? The predominant technique is to constrain all subproblems to share the same hyperparameters; this enables performing model selection on the target (multiclass) metric, but requires the assumption that a single hyperparameter set works well on all subproblems and allows suboptimal subproblem performance. Our experimental studies indicate shared hyperparameter selection outperforms independent optimization for a variety of binary reductions, and has similar performance in one case. We show two situations in which independent optimization is more effective than shared hyperparameter optimization: (a) when the subproblems have very different structure

and (b) when Hamming decoding is used and there is enough validation data to decrease the probability of choosing suboptimal subproblem models. Though we focus on using SVMs as the binary classification algorithm, the issues and results identified in this paper are applicable to any tunable binary classifier used with a multiclass-to-binary reduction method.

4.1 Introduction

Multiclass classification is an important machine learning problem, encompassing domains such as handwritten text recognition, protein structure prediction [73], heartbeat arrhythmia monitoring, and many others. Support vector machines were introduced as a binary classification algorithm, and several techniques have been proposed for adapting them to address multiclass classification problems [2, 38, 22, 81, 67, 9]. One simple, effective and widely-used technique is to reduce the multiclass classification problem to a set of binary classification problems. The binary classification problems are solved and predictions from the binary classifiers are combined to produce the multiclass prediction [2]. In this chapter, we focus on the issue of model selection in the induced subproblems for a variety of multiclass-to-binary reductions. In particular, we experimentally investigate whether it is more effective to perform model selection on each binary subproblem independently or to perform model selection on the multiclass problem by sharing hyperparameters across subproblems. We also perform experimental studies that illustrate the correlation between binary subproblem performance and multiclass performance, and the comparative performance of the one-vs-all reduction vs. the all-pairs reduction. We present results for two reduction methods (one-vs-all and all-pairs), two decodings (Hamming decoding and squared-error decoding) and two metrics (accuracy and the Brier score, a probability calibration metric). In Section 4.1.1, we describe the one-vs-all and all-pairs reductions and previous research. In Section 4.1.2, we focus on the particular issue of model selection in the binary subproblems,

and discuss theory correlating multiclass accuracy to binary accuracy and consistency results in Section 4.1.4. In Section 4.2, we present experimental studies showing the advantage of the constraint that hyperparameters are shared across subproblems. In Section 4.2, we show that shared optimization has an advantage because model selection tends to choose wrong solutions and subproblems in a multiclass problem are often similar with respect to model selection. We analyze the results and perform control studies in Section 4.3, report on supplementary results in Section 4.4 and conclude in Section 4.5.

4.1.1 Reducing Multiclass to Binary

4.1.1.1 One-vs-All

One of the simplest and most widely used techniques for reducing a multiclass problem to a set of binary subproblems is known as the one-vs-all reduction (OVA), also known as unordered class binarization [40] or one-vs-rest (or 1vr) [105]. Using the one-vs-all reduction, a k-class classification problem is decomposed into k binary classification subproblems, one for each class (see Figure 4.1). In the ith subproblem, the classifier is trained to distinguish whether the instance belongs to class i or not. At prediction time, the classifier with the highest output is chosen. An alternative scheme uses voting, or Hamming decoding, to combine the predictions, with ties broken randomly. There is some disagreement in the literature about the terminology for the one-vs-all reduction; Rifkin and Klautau [85] use the term one-vs-all to indicate winner-take-all with continuous outputs (i.e. choosing the classifier with the maximum output); other research such as [5] refer to one-vs-all as using Hamming decoding. Here, we use the term OVA to refer to continuous winner-take-all one-vs-all, and refer to the discrete version as OVA with Hamming decoding. Rifkin and Klautau studied the one-vs-all technique, and compared it to other methods for reducing multiclass to binary and to

other SVM techniques that provide direct optimization on the entire multiclass problem [85]. They show that model selection is essential and that under appropriate model selection, one-vs-all tends to perform as accurately as other multiclass SVM methods. Rifkin and Klautau's technique for model selection is to choose one set of hyperparameters for all subproblems, rather than trying to optimize each subproblem independently or trying to optimize differing hyperparameters for all subproblems jointly. Rifkin and Klautau don't explicitly state that they use the shared-model paradigm for model selection, but it is implied since one set of regularization hyperparameters is reported for each multiclass problem.

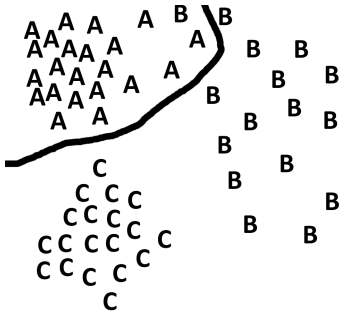


Figure 4.1: Illustration of an A-vs-BC decision boundary in a 2D, 3-class example of the One-vs-All reduction.

4.1.1.2 All-Pairs

In the all-pairs reduction, also known as pairwise classification [51], all-vs-all (or AVA) [85], round-robin classification [40] and 1-against-1 (or 11) [105]), a k-class classification problem is decomposed into $\frac{k(k-1)}{2}$ problems, one for each pair of classes (see Figure 4.2).

At prediction time, each binary classifier votes for one class, and the class with the most votes is selected as the multiclass prediction. Note that in contrast to continuous winner-take-all one-vs-all, the original all-pairs methodology ignores predicted

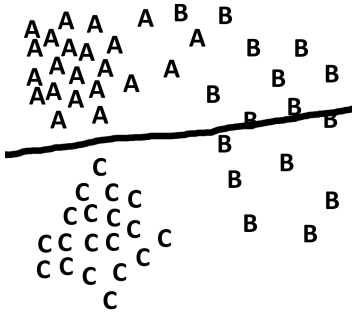


Figure 4.2: Illustration of an A-C decision boundary in a 2D, 3-class example of the all-pairs reduction.

probabilities or confidence predictions from each binary classifier, instead using only a discrete vote from each, though it is possible to use the all-pairs encoding with a decoding function other than Hamming decoding in loss-based decoding. Subsequent research in pairwise classification has shown how to incorporate continuous outputs and also to produce a probability distribution instead of a discrete vote [51, 103]. Friedman shows that Bayes optimal binary classifiers combine to produce a Bayes optimal multiclass classifier, and therefore each binary subproblem can be solved independently and as accurately as possible [38]. This proof is repeated here for for completeness.

The Bayes optimal decision is given by

$$\hat{y}(\boldsymbol{x}) = \underset{k \in K}{\operatorname{argmax}}\, p(\omega = k | \omega \in K, \boldsymbol{x})$$

where K is the set of possible labels, ω is the true label, $\boldsymbol{x}$ is the input feature vector and $\hat{y}$ is the predicted label. This is equivalent to

$$\hat{y}(\boldsymbol{x}) = \underset{k \in K}{\operatorname{argmax}} \sum_{i \in K} 1(\frac{p_k}{p_k + p_i} > \frac{p_i}{p_k + p_i})$$

where $1(x) = 1$ if x is true and 0 otherwise. This reduces to

$$\hat{y}(\boldsymbol{x}) = \underset{k \in K}{\operatorname{argmax}} \sum_{i \in K} 1(p(\omega = k | \omega \in \{i, k\}, \boldsymbol{x}) > p(\omega = i | \omega \in \{i, k\}, \boldsymbol{x}))$$

Therefore, given reliable $p(\omega = k | \omega \in \{i, k\}, \boldsymbol{x})$, binary reduction under the all-pairs reduction is equivalent to the true Bayes optimal decision. Note that this analysis assumes

that $p(\omega = k|\omega \in \{i, k\}, \boldsymbol{x})$ can be determined exactly for each subproblem, whereas in practice, it would be difficult or impossible to accurately obtain this distribution given a finite sample size. Friedman argues that unlike subproblems in the all-pairs reduction, subproblems in one-vs-all must be tuned simultaneously, since the outputs from each model must be commensurate with one another.

4.1.1.3 Other Techniques in Reducing Multiclass to Binary

While one-vs-all and all-pairs are the most widely studied and employed techniques for reducing multiclass to binary, they are only two cases within the more general framework known as loss-based decoding (LBD), which is itself an extension of error-correcting output coding (ECOC). Though we focus our experimental studies on the one-vs-all and all-pairs reductions, we also describe these other frameworks, since they also face the issue of model selection.

Error-Correcting Output Coding The error-correcting output coding (ECOC) framework was proposed by Dietterich and Bakiri in 1995 [27]. This scheme is named for its similarity to error correcting codes in information theory, with the analogy that an instance's class is a message to be transmitted, and error correcting codes are employed to encode the message in order to make the transmission (or classification) more tolerant of errors. ECOC requires all classes to appear in each subproblem, with an arbitrary specification (called a coding matrix) of how classes are reassigned to subproblems. For example, for a 5-class problem, classes $1, 3, 4$ might be assigned to the positive indicator class in a particular subproblem, corresponding to a row in the coding matrix of $r_i = \{+1, -1, +1, +1, -1\}$. At prediction time, each subproblem classifier votes for or against membership in the positive indicator class, and the class with the most votes is selected as the multiclass prediction, with ties broken randomly.

The number of unique and nontrivial binary splits (codewords) for a set of k classes is $2^{k-1} - 1$ [62]. Of these splits, k correspond to the one-vs-all dichotomies. The

other splits are different binary problems constructed from the original class labels. Dietterich and Bakiri [27] proposed using as many dichotomies as computationally feasible in order to improve the multiclass prediction. At prediction time, each base classifier is evaluated, and the class label with the minimum Hamming distance to the predicted codeword is used as the multiclass prediction. Dietterich and Bakiri recommend using all possible dichotomies when the number of classes is 7 or less; when there are more classes, a random sampling of dichotomies is typically used. The tradeoff is that codes with better error correcting properties typically correspond to more difficult subproblems.

Loss Based Decoding In 2000, Allwein et al. generalized the ECOC framework to to loss-based decoding, which (a) incorporates continuous instead of discrete classifier outputs and (b) allows some subproblems to optionally ignore some of the classes in the data set [2]. Incorporating continuous output makes it possible to represent the one-vs-all technique in the loss-based decoding framework (as we show below), and allowing subproblems to omit subsets of data points makes it possible to represent the all-pairs technique. Loss-based decoding was further generalized by Crammer and Singer to continuous-output coding [22], in which each class in a subproblem has some continuous weight $w \in \mathbb{R}$ rather than $w \in \{-1, +1\}$ as in ECOC or $w \in \{-1, 0, +1\}$ as in loss-based decoding.

One-vs-All in Loss-Based Decoding While loss-based decoding [2] was shown to encompass a variety of previous methods, including voted pairwise classification and Hamming-decoding one-vs-all, a representation for continuous winner-take-all one-vs-all has been lacking. In this section, we show that using the loss function $L(z) = (1 - z)^2$ in one-vs-all loss-based decoding yields the same predictions as continuous winner-take-all one-vs-all. Our experimental studies use this result in implementing the continuous one-vs-all method.

Theorem 1. *Using the loss function $L(z) = (1-z)^2$ in one-vs-all loss-based decoding yields the same predictions as continuous winner-take-all one-vs-all for a problem with labels in set K and base classifier predictions $f(\boldsymbol{x})$ on test point $\boldsymbol{x}$.*

Proof. The prediction made by OVA Loss Based Decoding is

$$\hat{y} = \operatorname*{argmin}_c L(f_c(\boldsymbol{x})) + \sum_{a \in K \neq c} L(-f_a(\boldsymbol{x}))$$

Adding $\sum_{a \in K} -L(-f_a(\boldsymbol{x}))$ to each term, we have

$$\hat{y} = \operatorname*{argmin}_c L(f_c(\boldsymbol{x})) - L(-f_c(\boldsymbol{x}))$$

Choosing the loss function to be $L(z) = (1-z)^2$, we have

$$\hat{y} = \operatorname*{argmin}_c (1 - f_c(\boldsymbol{x}))^2 - (1 + f_c(\boldsymbol{x}))^2$$

$$\hat{y} = \operatorname*{argmin}_c -4 f_c(\boldsymbol{x})$$

$$\hat{y} = \operatorname*{argmax}_c f_c(\boldsymbol{x})$$

□

This concludes the proof that winner-take-all one-vs-all classification can be implemented by setting the loss function as $L(z) = (1-z)^2$ in loss-based decoding with the one-vs-all coding scheme. In our experimental studies, we also employ this loss function with the all-pairs reduction scheme for completeness.

4.1.2 Model Selection in Reducing Multiclass to Binary

There are several differing views regarding subproblem model selection when using binary classifiers to solve multiclass classification problems. Before discussing the different types of model selection, we identify the terminology used here and throughout the chapter. We refer to a model as any function that produces a classifier when evaluated on a labeled training data set. Typically, a model is the combination of a learning

algorithm (e.g. SVM or AdaBoosted decision stumps) and an associated set of hyperparameters (also known as learning parameters or metaparameters) such as $\{\gamma, C\}$ for Gaussian SVM or {number of iterations} for AdaBoosted decision stumps. Parameters, as opposed to hyperparameters, refer to data components of the trained classifier rather than the mechanism used to obtain it. A base classifier is a classification algorithm used with a higher order classifier, such as a multiclass-to-binary reduction technique.

If the algorithm (but not the hyperparameter set) is selected before training, then model selection refers to the search over learning hyperparameters for the given classification algorithm. Search techniques such as grid search or binary search are often used to identify an appropriate set of hyperparameters, given an algorithm and a labeled training set.

As opposed to multiclass classification algorithms that solve the entire multiclass problem at once (and are regularized as a unit), techniques that reduce multiclass problems to a set of binary subproblems introduce the new issue of model selection in subproblems. Four approaches have been explored in the literature:

(1) no model selection: No model selection is done on subproblems or on the overall multiclass problem; the algorithm and hyperparameters (if any) are selected independently of the training data set.

(2) shared-model: A single algorithm and set of hyperparameters is used in all binary subproblems, typically selected by cross-validation or another resampling method on the multiclass problem. When only one classification algorithm is used, this method can be referred to as shared-hyperparameter model selection.

(3) independent optimization: Each binary subproblem is optimized independently of the others.

(4) full-joint optimization: A separate model is used for each subproblem, determined by optimizing all models simultaneously by validation on the multiclass

problem metric.

We examine each of these methods below, and point out prominent lines of research based on each technique. There is some disagreement about the 'classical' way to perform model selection for reducing multiclass to binary in SVMs; for example, Lebrun et al. report that "the classical way to achieve optimization of multiclass schemes is an individual model selection for each related binary sub-problem" [65]. However, often cited works such as Rifkin et al. [85] select a single set of hyperparameters that are used in all subproblems. Lorena provides an overview and discussion of these methods and related studies [70].

4.1.2.1 No Model Selection

Some research in machine learning reduction uses untuned (unregularized) base models, with the implicit assumption that behavior will transfer to the regularized case. For example, Allwein et al. report usage of polynomial SVMs with polynomials of degree 4, with no mention of whether or how this model was selected [2]. Beygelzimer et al. report "We do not perform any kind of parameter optimization such as tuning the regularization parameters for support vector machines or the pruning parameters for decision trees. Our objective is simply to compare the performance of the two reductions under the same conditions." [7]. Zadrozny reports usage of the boosted naive Bayes algorithm with 10 rounds of boosting [106], selected a priori without any search over algorithms or hyperparameter space. Avoiding model selection generally results in suboptimal performance [50], and there is no guarantee that results will be transferable to situations with model selection.

4.1.2.2 Shared Hyperparameters

The earliest and most pervasive view is that a single set of hyperparameters should be used for all subproblems. Rifkin and Klautau follow this paradigm, and perform two

independent one-dimensional grid searches for model selection [85].

Duan et al. take the $\{C, \gamma\}$ of each of the binary classifiers within a multi-category method to be the same, tuned based on the multiclass classification performance over a coarse and fine grid [29, 9].

Hsu and Lin [55] take the hyperparameter set for each subproblem to be the same, and again use the shared-hyperparameters paradigm to parallel the shared-hyperparameters paradigm inherent in monolithic SVM multiclass methods [54].

Platt et al. report one set of hyperparameters for each set of binary subproblems, indicating that they performed shared-hyperparameter model selection [81].

When addressing a problem with a domain-specific metric, shared hyperparameter model selection may be more appropriate than independent optimization because there may not be a corresponding binary loss function. Furthermore, shared hyperparameter optimization has the advantage that the tuning is performed by evaluation on the actual training data set for which a predictive model is desired, rather than tuning artificial subproblem models as in independent optimization.

4.1.2.3 Independent Optimization

In independent optimization, model selection is performed independently on each subproblem. More specifically, one model selection routine is applied to each subproblem, and (potentially) different models (algorithms and/or hyperparameter sets) are selected for each subproblem. Independent optimization removes the constraint in shared-hyperparameters that the same model must be applied to all subproblems, and therefore allows more flexibility. However, as in all variance-increasing tradeoffs, this increases the possibility of overfitting, or, more generally, may not entail an inductive bias that is suitable for the problem at hand.

Liepert reported that independent optimization was not a significant improvement over shared-hyperparameter optimization [66].

Szepannek reported on performing independent model selection on each subproblem using heterogeneous classifiers for the all-pairs reduction [94]; that is, potentially using different classification algorithms and hyperparameter sets on each subproblem. However, results on real world data sets reported in [94] are not statistically compelling; only four data sets are used, and Wilcoxon signed ranks tests indicate that there is no statistically significant difference between the proposed method and the baseline algorithms at the predetermined value of $p \leq 0.05$ (actual p-values are $p \leq 0.125$ and $p \leq 0.25$). Furthermore, this work used only naive Bayes and LDA as the base classification algorithms, and these results may not generalize to more powerful classification algorithms. Independent optimization has the potential to increase multiclass predictive performance by finding a separate model suited to each binary subproblem.

4.1.2.4 Full Joint Optimization

Another perspective is that if each of l subproblems have n hyperparameters, then all l^n hyperparameters should be determined simultaneously by optimizing the overall multiclass classification performance. Due to the large search space, evolutionary algorithms are often employed to search for good solutions.

De Souza et al. introduces a particle swarm optimization for searching over all subproblem models simultaneously [33]; however, they conclude that this technique did not statistically significantly improve performance.

Lebrun et al. introduce an evolutionary algorithm for performing a joint optimization over all subproblems simultaneously [65]. In this article, statistical claims are also problematic, since only three datasets are used, and Wilcoxon signed ranks tests don't indicate a statistically significant result at the predetermined value of $p \leq 0.05$ (actual p-value is $p \leq 0.25$.)

Xu and Chan claim that all binary problems need to have different parameters, but that it is insufficient to optimize each individually [104]. They propose an algorithm

that starts by searching over a 13 x 13 grid with shared hyperparameters, then uses genetic algorithms to fine-tune the individual subproblem parameters.

In theory, it would be optimal to perform a full joint optimization (if it were tractable); however, with so much flexibility in the model and given finite sample sizes, there is a larger risk of overfitting to the validation set.

In this thesis, we focus on the shared-hyperparameters and independent-optimization techniques since they are computationally tractable and significantly more effective than avoiding model selection. For purposes of investigation, we introduce and evaluate a new methodology that uses shared hyperparameters that are selected by optimizing average binary performance.

4.1.3 Computational Demand

The previously discussed techniques for model selection differ in their computational demands. At the low extreme, avoidance of model selection requires no additional computational power beyond training the classifier itself because the learning algorithm and learning hyperparameters are specified independently of the data. At the opposite end of the spectrum, full-joint optimization requires evaluation of all l^n combinations of hyperparameter sets, where l is the number of subproblems and n is the cardinality of the hyperparameter set (e.g. two for Gaussian SVMs). Independent-optimization and shared-model model-selection have identical computational requirements (as long as the binary and multiclass metric have similar computational demands), since each requires training and evaluation once for each combination of subproblem and model.

4.1.4 Consistency in Reducing Multiclass to Binary

An important feature of a machine learning reduction method is consistency, which means that optimal binary classifiers guarantee an optimal multiclass classifier. As pointed out in [5], it is sufficient to minimize regret (difference from Bayes error

rate) rather than absolute error since the Bayes error rate may be nonzero. When using a consistent reduction technique, optimal binary classifiers guarantee an optimal multiclass classifier. It has been shown that one-vs-all (or any other ECOC encoding) under Hamming decoding is inconsistent, while the all-pairs reduction and one-vs-all (with continuous outputs) is consistent [5, 6]. This result suggests that independent-optimization model selection should work well for all-pairs and one-vs-all, since independent optimization focuses on identifying the best model for each subproblem and has the flexibility to choose different models for each subproblem. On the other hand, shared-hyperparameter optimization may be more effective than independent optimization for one-vs-all with Hamming decoding, since it is inconsistent. For some consistent reductions, the multiclass regret is bounded by a function of the average binary regret—this motivates solving each subproblem as accurately as possible.

4.2 Main Experimental Results

In this section, we experimentally investigate whether it is better to perform shared-hyperparameter optimization or independent optimization. In Section 4.2.1, we describe the datasets, algorithms and statistical methods used in our studies. The main experimental results are reported in Section 4.2.2 with discussion and analysis in Section 4.3.

4.2.1 Setup

4.2.1.1 Data Sets

Experiments are performed over 20 publicly available datasets. The datasets, number of classes, number of features and number of instances are indicated in Table 4.1. We formalize our data set selection decision procedure below to rule out bias in data set selection. We downloaded collections of preprocessed datasets in Weka's

ARFF format from `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`, omitting any collection that consisted solely of artificial, ordinal or regression datasets, and filtered them based on the following criteria:

(1) classes ≥ 3

(2) $5 \leq$ numeric attributes ≤ 500

(3) instances ≥ 200

The first rule (classes ≥ 3) selects multiclass classification problems, ignoring the binary case $k = 2$. The second rule ($5 \leq$ numeric attributes ≤ 500) ensures there are sufficiently many but not too many attributes. The remaining criterion ensures that there is a sufficient number of data points. Uniquely identifying attributes that completely specify the identity of an instance (such as ID or index attributes) are discarded, specifically: the *counter* attribute in *collins*, the *BookID* attribute in *authorship* and the *ID* attribute in *dj30-1985-2003*. Classes with less than 20 instances are deleted, along with corresponding instances. The free parameters in the above rules were hand-tuned until 20 datasets were selected to facilitate statistical analysis. After deletion of classes, any duplicate instances (based on attribute values, not class values) are deleted. Data set selection was performed before evaluation of algorithms in order to avoid bias.

Stratified subsampling is used to reduce the total number of instances for large problems in order to reduce computational demands, while maintaining a distribution over class labels commensurate with the original sample. For data sets with $N \geq 450$ instances, random draws are sampled with $N_{tr} = 300$ training points and $N_{ts} = 150$ test points. For data sets with $N < 450$, random draws are taken with 2/3 of the instances used for training and the remaining approximate 1/3 points for testing. Missing values are filled in with the mean of non-missing values for each attribute. Datasets from similar domains are discarded in order to improve tests for statistical significance of

results, as prescribed by Demšar [24]. In particular, *pendigits* was discarded because of its similarity to *optdigits*, only one of the *mfeat-* series was selected, and *anneal.ORIG*, *heart-h* and *cars-with-names* were discarded due to similarity with other data sets. For data sets with $k > 20$ (*letter* and *dj30-1985-2003*), 1/3 of the classes are removed to decrease computational demands, and further stratified subsampling removes 1/3 of the instances.

Table 4.1 indicates the datasets used in our experiments, and their relevant properties. The column labeled *entropy* refers to the normalized entropy (in base 2) of the class distribution, $e = \frac{-1}{log_2 k} \sum_{i=1}^{k} p_i log_2(p_i)$, where p_i is the proportion of instances with the class label c_i and k is the number of classes. For instance, the entropy is 1 for a class with an even distribution of class labels $p_1 = ... = p_k = 1/k$ and 0 for a distribution that has only instances with one label, i.e. $p_i = 1, p_{j \neq i} = 0$. To summarize, the number of classes varies from 3 to 20, with entropy varying between 0.4819 and 0.9976. The smallest training sample size (after subsampling) is 133, and the number of attributes ranges from 6 to 254.

Because classes without a sufficient number of instances have been discarded, and the data has optionally been subsampled, not all results on these datasets will correspond identically to those found in the literature. However, we are able to perform comparative studies since our experiments include a number of baseline algorithms. Also note that a few of these data sets have classes that would be more appropriately modeled as ordinal attributes, but that in these experiments we omit any ordering information and treat classes simply as distinct nominal values.

4.2.1.2 Methods

We use a modified version of LibSVM with probability estimates enabled according to the algorithm presented in [103], using Weka's adapter. LibSVM is non-deterministic when probability estimates are produced, so we modified LibSVM to use

Table 4.1: Properties of the 20 data sets used in our experimental studies.

dataset	classes	entropy	numeric	nominal	train	test	sampled-from
anneal	4	0.6282	6	23	300	150	878
arrhythmia	5	0.7311	198	56	257	129	386
authorship	4	0.9363	70	0	300	150	841
autos	5	0.9328	15	10	134	68	202
cars	3	0.8693	6	1	270	136	406
collins	11	0.9543	19	0	300	150	451
dj30-1985-2003	20	0.9936	6	0	133	67	138123
ecoli	4	0.9008	6	0	204	103	307
eucalyptus	5	0.9725	14	5	300	150	736
halloffame	3	0.5010	15	1	300	150	1340
hypothyroid	3	0.4819	6	20	300	150	3707
letter	18	0.9920	16	0	136	68	18668
mfeat-morphological	10	0.9911	6	0	300	150	1888
optdigits	10	0.9971	58	0	300	150	5620
page-blocks	5	0.5945	10	0	300	150	5393
segment	7	0.9927	18	0	300	150	2086
synthetic-control	6	0.9976	60	0	300	150	600
vehicle	4	0.9923	18	0	300	150	846
vowel	11	0.9910	10	3	300	150	990
waveform	3	0.9942	40	0	300	150	5000

a deterministic seed instead of seeding from the system clock to enable reproducible runs. The one-vs-all and all-pairs methods are implemented in Scala under the loss-based decoding framework[1] using a squared error decoding for one-vs-all to implement continuous-output winner-take-all and Hamming decoding for the all-pairs reduction to implement voting.

In order to compare shared-hyperparameter selection to individual optimization model selection, we evaluate both techniques on the 20 datasets identified in Section 4.2.1.1. Results are averaged over 10 random splits.

Support Vector Machines (SVM-121) We use the SVM algorithm as implemented in LibSVM [103], using the Gaussian kernel. To search over the hyperparameters $\{c, \gamma\}$, we perform a search over the coarse 7×7 grid of $\{-5, -1.666, 1.666, 5.0, 8.333, 11.666, 15.0\}^2$ to first determine a value for the cost hy-

[1] The default Weka implementation of One-vs-All or ECOC uses a loss-oriented output, in which probabilities are summed (not using either Hamming decoding or continuous winner-take-all).

perparameter c. Then a separate search is performed over a finer grid of 72 samples (ranging from -40 to 15) at the previously determined c-value to obtain the value for γ. We used this sampling scheme since there was much more sensitivity to the γ hyperparameter than to the c hyperparameter, and so that our scheme would take a total of 121 samples, as done in many other grid searches, such as LibSVM [103]. This fine-granularity 1-d search for γ also facilitates visualization of the results. Platt scaling is used to fit a sigmoid to each of the SVM models to improve probability estimates [82], and Wu et al.'s technique for improving probability estimates by applying their second method for pairwise coupling to the binary predictions [103].

Accuracy Metric The accuracy metric counts the number of correctly classified instances, ignoring probabilistic predictions: $a = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}(\boldsymbol{x}) = y(\boldsymbol{x}))$, where $\hat{y}(\boldsymbol{x}) = \text{argmax}_i\, p_i(\boldsymbol{x})$ is the predicted class, $\boldsymbol{x}$ is the input attribute vector, and $y(\boldsymbol{x})$ is the true class. In this chapter, we primarily focus on classifier using the accuracy metric, though some results with the Brier metric are presented in Section 4.4.2. The accuracy metric has the advantage that accuracy can be computed on both the binary subproblems and on the original multiclass problem; other multiclass metrics may not have a corresponding binary metric (or vice-versa).

4.2.1.3 Statistical Techniques

Since the scores are unlikely to be normally distributed, we use the Wilcoxon signed-ranks test for identifying statistical significance of the results [24]. Also, Demšar recommends using 5+ datasets to obtain reliable results [24]; we use 20 independent data sets to improve statistical confidence in our results.

The rest of this section is organized as follows. Section 4.2.2 compares shared-hyperparameter to independent optimization. In order to understand the results, we investigate the structure of the binary subproblems and of the multiclass problems and the relationship between the two in Section 4.2. We analyze the results and perform

control studies in Section 4.3.

4.2.2 Results

Figure 4.3 indicates the performance of shared-hyperparameter optimization and independent optimization for one-vs-all, all-pairs, one-vs-all with the Hamming loss function and all-pairs with the squared error loss function. In each case, our results show that shared-hyperparameter optimization attains a higher accuracy than independent model optimization.

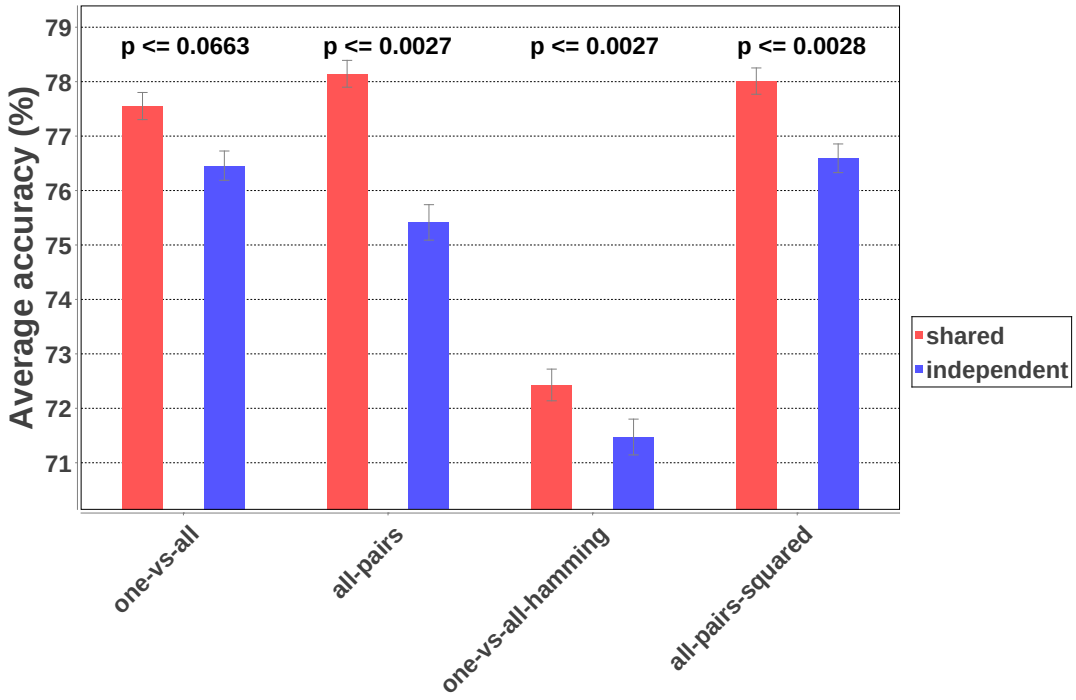


Figure 4.3: Average accuracy values comparing independent to shared hyperparameter selection for each reduction, averaged over 20 data sets. The advantage of shared over independent in the one-vs-all reduction is not statistically significant at a $p \leq 0.05$ level. One standard error is indicated for each method for the 10 runs.

For the four combinations of {one-vs-all, all-pairs} $\times$ {squared, Hamming}, shared-hyperparameter optimization outperforms independent optimization (see Table 4.2). For continuous one-vs-all, the win is not statistically significant at $p \leq 0.05$, with a value of $p \leq 0.0663$; for all other cases the win is statistically significant. For continuous

	one-vs-all	all-pairs	one-vs-all-hamming	all-pairs-squared
accuracy	shared 0.0663	**shared** 0.0027	**shared** 0.0027	**shared** 0.0028

Table 4.2: Winning strategy for each combination of reduction and metric. Statistically significant wins (at $p \leq 0.05$) are highlighted. P-values from the Wilcoxon signed-ranks test are indicated after the winning strategy.

one-vs-all, shared hyperparameter optimization attains an average rank of 1.3 (compared to 1.7 for independent optimization) and an average accuracy of 77.55%, compared to 76.46% for independent optimization. Fine-grained results, including performance on individual data sets, are reported in the appendix (see Section 4.6.1). Explanations for the mechanism behind these results are discussed in Section 4.3.

4.3 Analysis

In this section, we investigate the mechanism behind the effectiveness of shared-hyperparameter optimization. We start by showing that shared-hyperparameter optimization is advantageous because subproblems often have similar structure (Section 4.3.1). Conversely, we show that when subproblem decision boundaries have differing shapes that independent optimization is necessary (Section 4.3.2). We also discuss the relationship between binary and multiclass accuracy, showing a strong correlation (Section 4.3.3). We also perform control studies that rule out other explanations. In Section 4.3.4, we invalidate the hypothesis that shared hyperparameter optimization is more effective because it uses the true target metric for optimization instead of a binary metric. In Section 4.3.5, we show that choosing optimal classifiers on subproblems (as judged by the oracle) favors independent optimization for reductions that use Hamming decoding, but still favors shared-hyperparameters for the one-vs-all reduction.

4.3.1 Subproblem are Similar

In this section, we show that subproblems in multiclass classification often have a similar structure with respect to model selection and, subsequently, that they often share similar optima. This subproblem similarity makes shared-hyperparameter optimization more productive than independent optimization.

For algorithms like one-vs-all, it is not unexpected for subproblems to have similar structure since portions of one decision boundary may appear in several problems. For example, consider a 5 class classification problem. The first one-vs-all subproblem is to discriminate class c_1 from $\{c_2, c_3, c_4, c_5\}$. The second subproblem is to discriminate class c_2 from the union of classes c_1, c_3, c_4, c_5. Both of these subproblems share c_3, c_4, c_5 in the negative indicator class, and therefore, the subproblems will share similar portions of the decision boundary.

Another explanation of why subproblems might have similar structure is that the generative distribution that produces elements from each class may have similar physical, statistical, or noise properties. For instance, if each class in a data sample is produced by a Gaussian distribution with an identity covariance matrix, then all subproblems in the all-pairs reduction will be optimally fit with linear discriminants.

In the remainder of this section, we show that subproblems in real-world data sets are structurally similar, and that this causes shared hyperparameter optimization to work well since a single hyperparameter value can often be chosen that performs well on many subproblems. As an example, we select the four class problem *vehicle* to illustrate the relationship between subproblems. Composite results over many data sets are discussed in Section 4.3.1.3.

4.3.1.1 Case Study: The *Vehicle* Dataset

We selected the *vehicle* data set for particular investigation since it illustrates many of the important issues in subproblem similarity. The *vehicle* data set is a 4-class problem, and therefore has four one-vs-all subproblems and 6 all-pairs subproblems.

One-vs-All Figure 4.4 indicates the model selection curves for the four subproblems in one-vs-all for the *vehicle* data. First, note that there seem to be two types of behaviors; the upper pair of curves for class c_2 and class c_3 and the lower pair for classes c_0 and c_1. It is interesting to note that the lower curves correspond to the classes *opel* and *saab* and the upper curves correspond to *bus* and *van*, showing that these conceptually similar classes have similar structures under model selection. Second, note that even though the pairs of curves have significantly different shape, they have optima in nearly the same region of the $log_2(\gamma)$ dimension, around $log_2(\gamma) = -5.0$. This indicates that the shared hyperparameter would be effective for this problem since all four classes, though different, peak at the same hyperparameter setting. Table 4.7 indicates that independent optimization actually averages a 0.8% higher average accuracy, though this difference is probably not significant compared to the standard deviation of about 3.0%.

All-Pairs Figure 4.5 indicates the model selection curves for the 6 subproblems in the all-pairs reduction for the *vehicle* data set. Note that 5 of the subproblems (upper part of the chart) have a similar structure, with broad peaks in the range of $-13 \leq log_2(\gamma) \leq -3$. The unique lower subproblem plot is between the two sedans *opel* and *saab*. Again, despite the difference between these types of series, they have a similar peak, around $log_2(\gamma) = -6.0$. Table 4.8 indicates that shared-hyperparameter optimization averages about 1.3% higher average accuracy, again probably not significant compared to the standard deviation of about 3.0%.

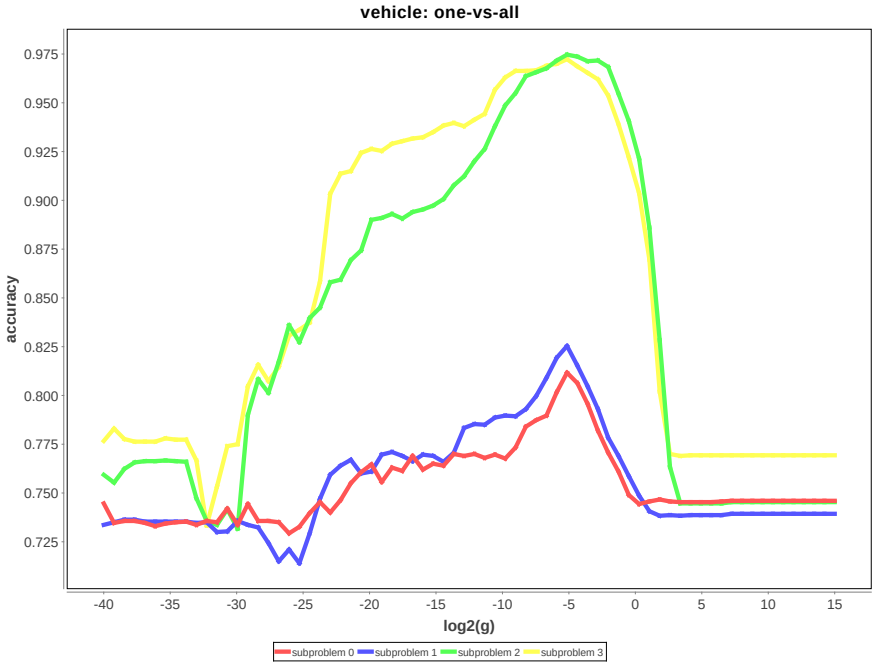


Figure 4.4: Independent model selection curves for the four one-vs-all subproblems in the *vehicle* data set.

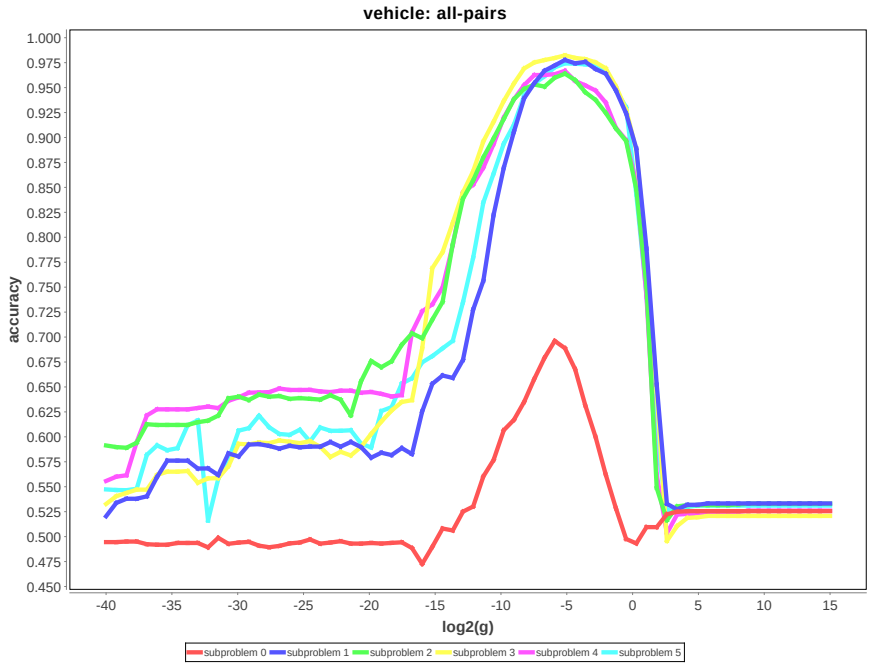
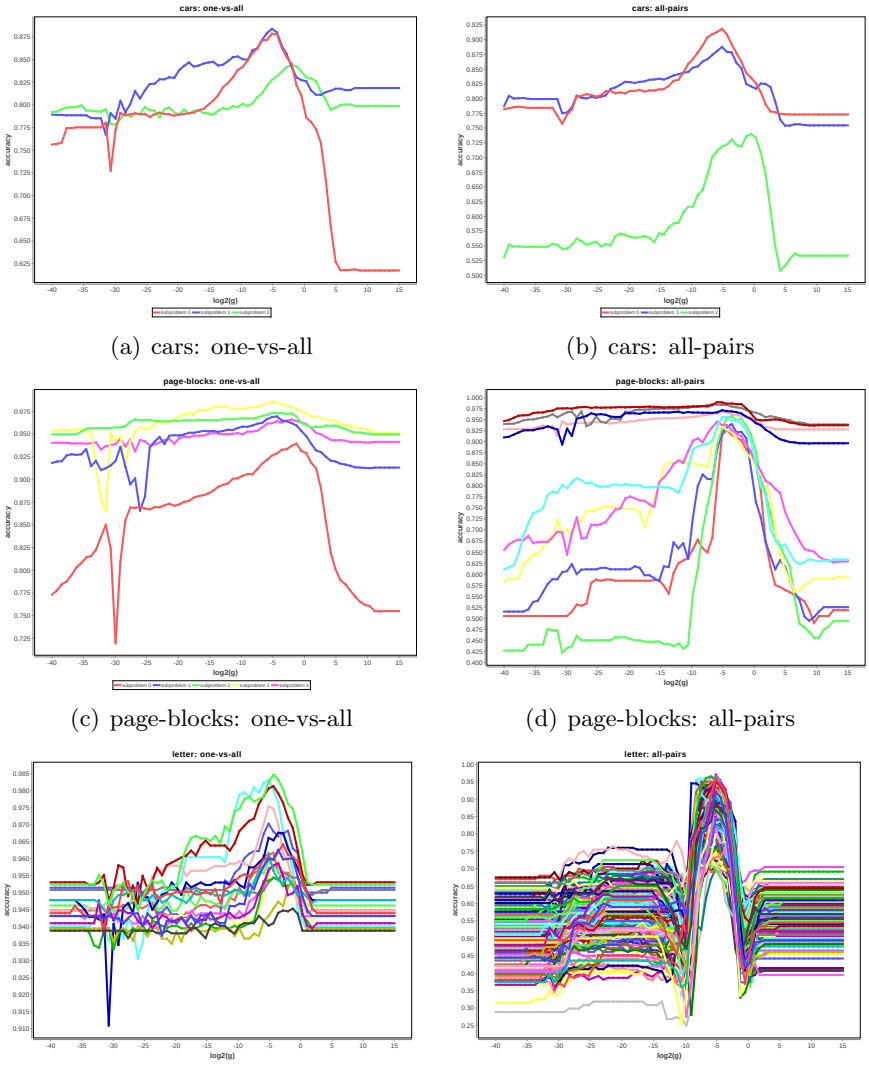


Figure 4.5: Independent model selection curves for the 6 all-pairs subproblems in the *vehicle* data set.

4.3.1.2 Illustration of Model Selection Curves

In this section, we show the model selection curves for the *cars*, *page-blocks* and *letter* data sets under the one-vs-all and all-pairs reduction methods, which represent many of the salient features in the 20 data sets. Note that the curves tend to peak near the same regions, and that often, subproblems have similar shapes.



(a) cars: one-vs-all

(b) cars: all-pairs

(c) page-blocks: one-vs-all

(d) page-blocks: all-pairs

(e) letter: one-vs-all

(f) letter: all-pairs

Figure 4.6: Examples of subproblem performance as a function of γ for the *cars*, *page-blocks* and *letter* data sets.

4.3.1.3 Aggregate Results

In order to quantify the pertinent differences between subproblems, we compute the average binary subproblem loss at the hyperparameter value selected by shared-hyperparameter optimization. Specifically, we identify the hyperparameter value γ_s selected by the shared-hyperparameter method, then for each of the subproblems, we compute the difference between optimal binary accuracy of the subproblem attained at the optimal value of gamma γ_o. The difference is $d = \bar{a}(\gamma_o) - a(\gamma_s)$, which quantifies the degree to which the subproblems have similar peaks.

The average subproblem losses incurred by shared-hyperparameter optimization for the one-vs-all reduction are indicated in Figure 4.7. All differences are less than 0.80%, with an average of 0.30%. The datasets *halloffame* and *vehicle* attain the optimal values for all subproblems. These results validate our hypothesis because they show that at a particular hyperparameter value, the subproblems do not differ significantly from their optimal value. Furthermore, this result indicates that the individual subproblems attain very good values at the hyperparameter selected by shared-hyperparameter optimization.

For the all-pairs reduction (see Figure 4.8), the average loss is 4.24%, significantly larger than the average loss for the one-vs-all reduction. The largest loss values for the all-pairs reduction are 36.4% for *letter* and 29.4% for *dj30-1985-2003*. Note that the larger loss values occur at large number of classes. In this case, subproblems deviate from each other so significantly that they suggest that independent-optimization should outperform shared-hyperparameter optimization, contrary to the results obtained in Section 4.2. In Section 4.3.5, we show that independent optimization is, in fact, more appropriate for the Hamming decoding techniques (including all-pairs) once subsampling problems are ruled out.

The similarity of subproblems does not guarantee that shared-hyperparameter op-

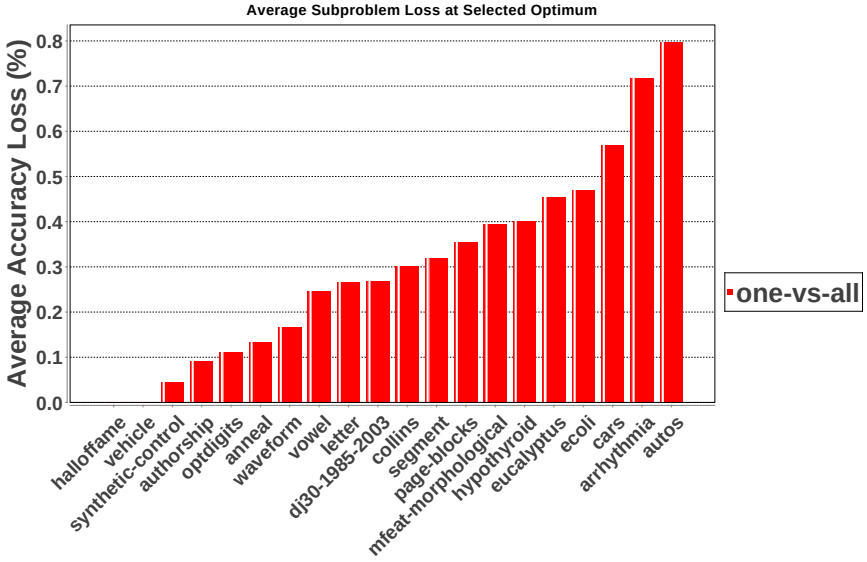


Figure 4.7: Average subproblem accuracy loss at the value selected by shared-hyperparameter optimization for the one-vs-all reduction.

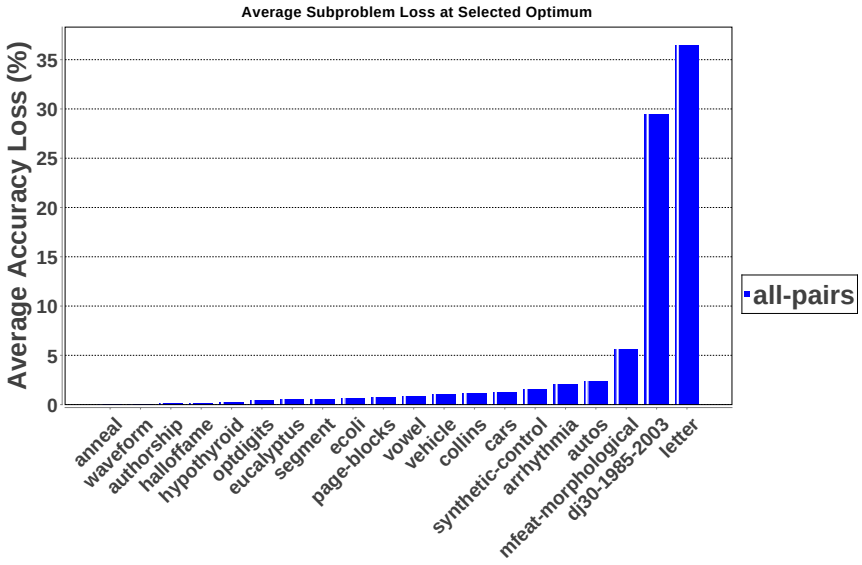


Figure 4.8: Average subproblem accuracy loss at the value selected by shared-hyperparameter optimization for the all-pairs reduction.

timization will outperform independent optimization; independent optimization should be able to determine the same hyperparameter sets for each subproblem. However, an advantage of shared-hyperparameter optimization in the case of similar subproblems is that the effective amount of validation data is the combination of validation data for

all subproblems, whereas in independent optimization, each model selection must be performed with a significantly smaller amount of validation data. This efficient re-use of validation data may explain the benefits of of shared-hyperparameter optimization over independent optimization[2] ; further experimental studies would be necessary in order to isolate this effect and validate this hypothesis.

4.3.2 Differing Subproblems Favor Independent Optimization

In this section, we construct synthetic problems to show that independent optimization is more effective than shared-hyperparameter optimization in problems in which subproblem decision boundaries differ in shape. In particular, we investigate the behavior of both model selection techniques in two synthetic data sets that are designed to have different optima for each subproblem with respect to model selection. Two of the parameters that govern performance as a function of hyperparameter sets are (a) noise and (b) shape of the decision boundary. We first experiment with Gaussian data sets and varying degrees of noise between each data set, which contains only linear decision boundaries. In the next section, we experiment with datasets in which some subproblem decision boundaries are linear and others are nonlinear. For both synthetic data sets, we use 300 training and 150 test points over 10 random resamplings as in Section 4.2.

Linear Decision Boundaries with Varying Noise In our first synthetic data experiment, we define a three-class problem in which each intra-class decision boundary is linear, and there are varying degrees of noise between each class. To implement this, we sample data points from each class from a Gaussian, each with the identity covariance matrix. The centroids of each Gaussian are $c_1 = (0, 0), c_2 = (0, 1), c_3 = (0, 3)$ so that the intra-class distances are $1, 2, 3$. Qualitatively, these settings correspond to a small, medium and large amount of noise between each pair of classes.

[2] Thanks to Shumin Wu for identifying this explanation.

In the one-vs-all reduction, some subproblem classes correspond to the union of two classes, and again we have varying degrees of noise between each subproblem. In the c_1 vs c_{23} subproblem, the amount of noise between class c_1 and c_{23} is determined by the closest class in c_{23}, namely c_2. Therefore, in this case, there are interclass distances of 1 (between class c_1 and c_{23}) and 2 (between class c_3 and c_{12}). The subproblem corresponding to c_2 vs c_{13} has class c_2 between the larger composite class c_{13}. Therefore, this configuration of Gaussian clusters offers varying degrees of noise between each subproblem for both all-pairs and one-vs-all reductions. The datasets are shown in Figure 4.9.

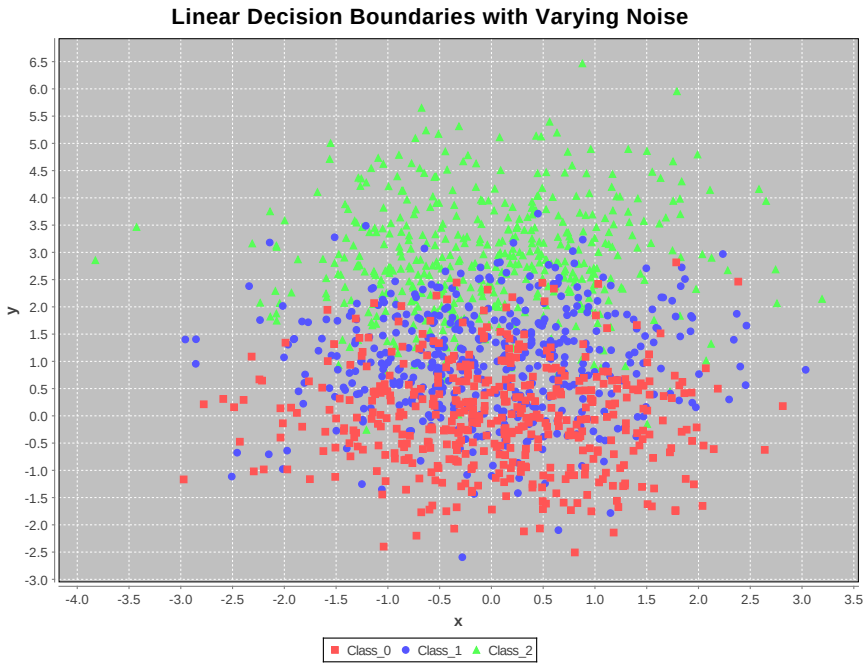


Figure 4.9: Synthetic datasets generated by Gaussian distributions with varying degrees of noise.

Results The results are indicated in Table 4.3. For this synthetic data set, shared-hyperparameter model selection outperforms independent optimization in all three reductions. The improved accuracy of shared-hyperparameter optimization is not very large, less than 1.1% in all cases. To understand these results, the model selection

plots are shown in Figures 4.10 (for the one-vs-all reduction) and 4.11 (for the all-pairs reduction). Note that in both cases, despite having a different amount of noise in each subproblem, the optima are not significantly separated, and that choosing the optimal point with respect to the *multiclass* curve doesn't incur much loss on any of the subproblems. Therefore, we conclude that shared-hyperparameter optimization is sufficient for this synthetic problem because the subproblems are structurally similar with respect to model selection.

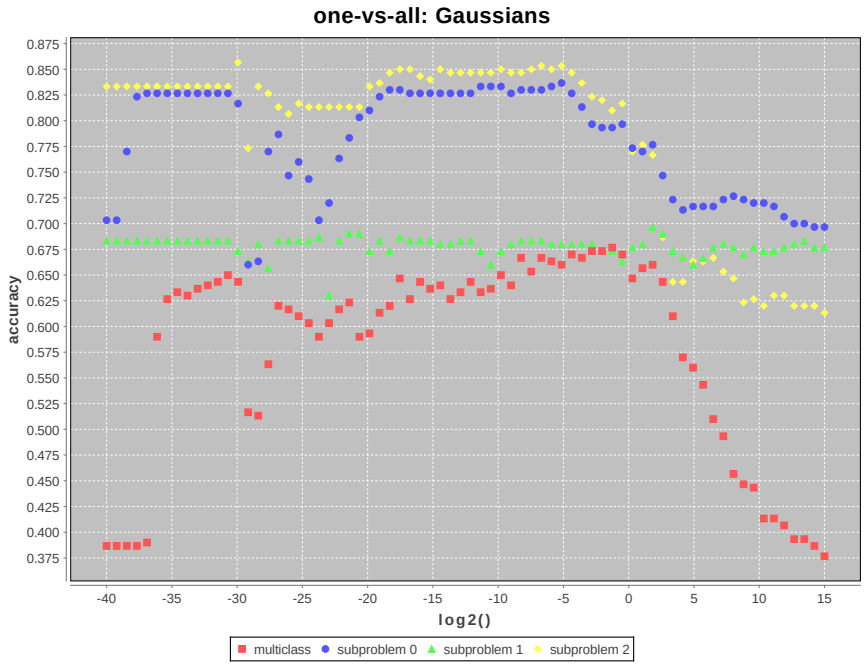


Figure 4.10: Model selection curves for Gaussian synthetic data sets under the one-vs-all reduction

Table 4.3: Accuracy results for linear decision boundaries (in %), for synthetic data sets as described in paragraph 4.3.2. Standard error over 10 random samplings is indicated in parentheses.

reduction	shared	independent
one-vs-all	**66.7** (1.3)	66.1 (1.3)
one-vs-all-hamming	**58.2** (2.5)	58.1 (1.9)
all-pairs	**67.6** (1.3)	66.5 (1.9)

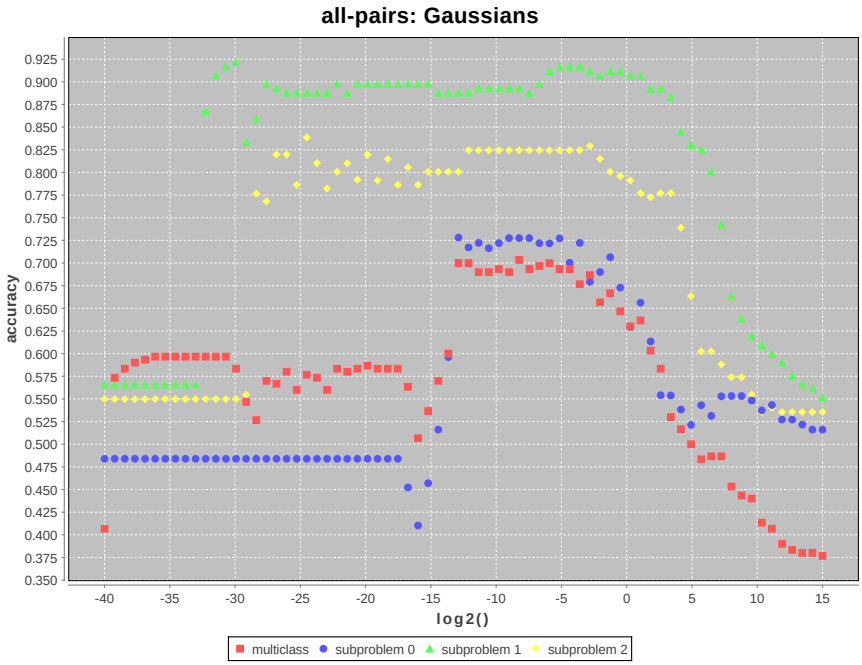


Figure 4.11: Model selection curves for Gaussian synthetic data sets under the all-pairs reduction

Linear and Nonlinear Decision Boundaries In our second synthetic experiment, we introduce a nonlinear boundary between classes c_1 and c_2. In particular, for classes c_1 and c_2, a point is drawn uniformly from the unit circle, and the point is assigned to class c_1 if $y > 0.35sin(6.4\pi x)$; otherwise it is assigned to class c_2. This distribution yields a sinusoidal decision boundary with a frequency of $f = 3.2$ (about 7 periods across the unit circle). A third class c_3 is drawn from a Gaussian distribution with an a covariance matrix of $0.1I$, where I is the 2×2 identity matrix, and a centroid located at $x = 0, y = 1$, so that it overlaps significantly with class c_2 and less with class c_1.

Results The results are indicated in Table 4.3. For this synthetic data set, independent optimization outperforms shared-hyperparameter model selection in all three reductions. The improved accuracy of independent optimization is 1% for each of the one-vs-all reductions, and 1.8% for the all-pairs reduction. To understand these

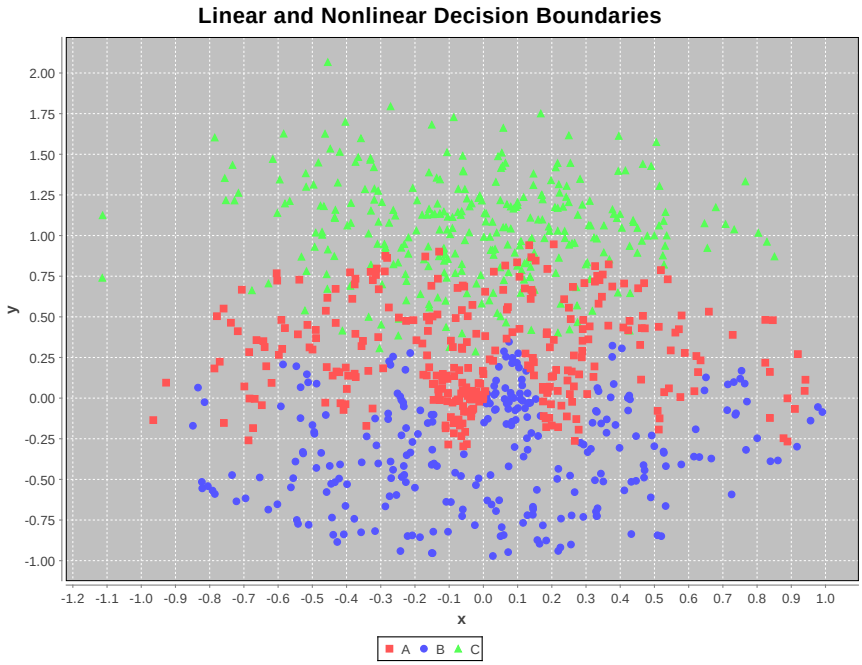


Figure 4.12: Synthetic datasets with sinusoidal and linear decision boundaries.

results, the model selection plots are shown in figures 4.13 (for the one-vs-all reduction) and 4.14 (for the all-pairs reduction). This problem was constructed to have differing optimal hyperparameters for each subproblem. For both the one-vs-all and all-pairs reductions, the optimal hyperparameter on one subproblem gives more than 2% error on at least one other subproblem. Therefore, since the subproblems are significantly different with respect to model selection, independent optimization is essential for this problem.

Table 4.4: Accuracy results for mixed linear and nonlinear decision boundaries (in %), described in paragraph 4.3.2. Standard error over 10 random samplings is indicated in parentheses.

reduction	shared	independent
one-vs-all	82.4 (0.6)	**83.5** (0.9)
one-vs-all-hamming	78.5 (1.3)	**79.5** (1.3)
all-pairs	82.4 (1.3)	**84.2** (0.9)

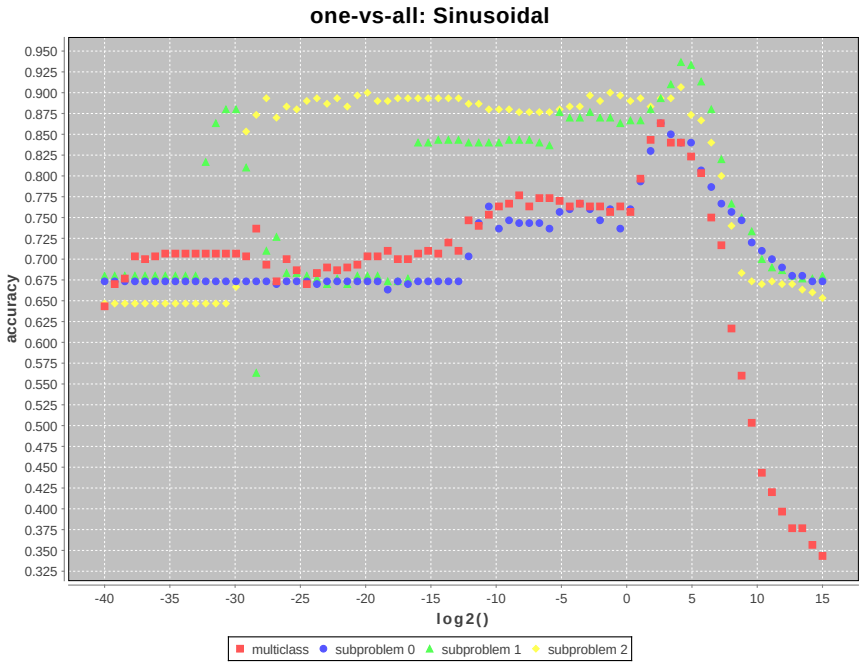


Figure 4.13: Model selection curves for Sinusoidal synthetic data sets.

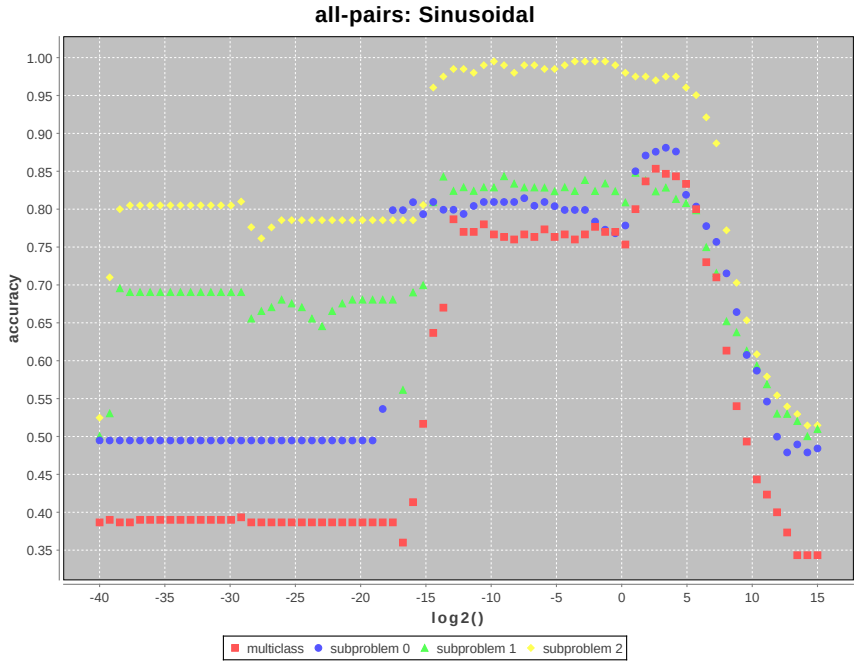


Figure 4.14: Model selection curves for Sinusoidal synthetic data sets.

Summary The results of the synthetic experiments indicate that when subproblems have significantly different shape with respect to tuning of hyperparameters

(as in the mixed linear/nonlinear example above), that it is essential to allow each subproblem to select its own optimal hyperparameters. On the other hand, when the subproblems have similar shape, or, more specifically, have optima in similar regions of the hyperparameter space, then shared-hyperparameter optimization is more effective. Therefore, since shared-hyperparameter optimization outperformed independent optimization in the real-world data sets, discussed in Section 4.2, we infer that this set of real world data sets entailed subproblems that have similar structure with respect to hyperparameter selection.

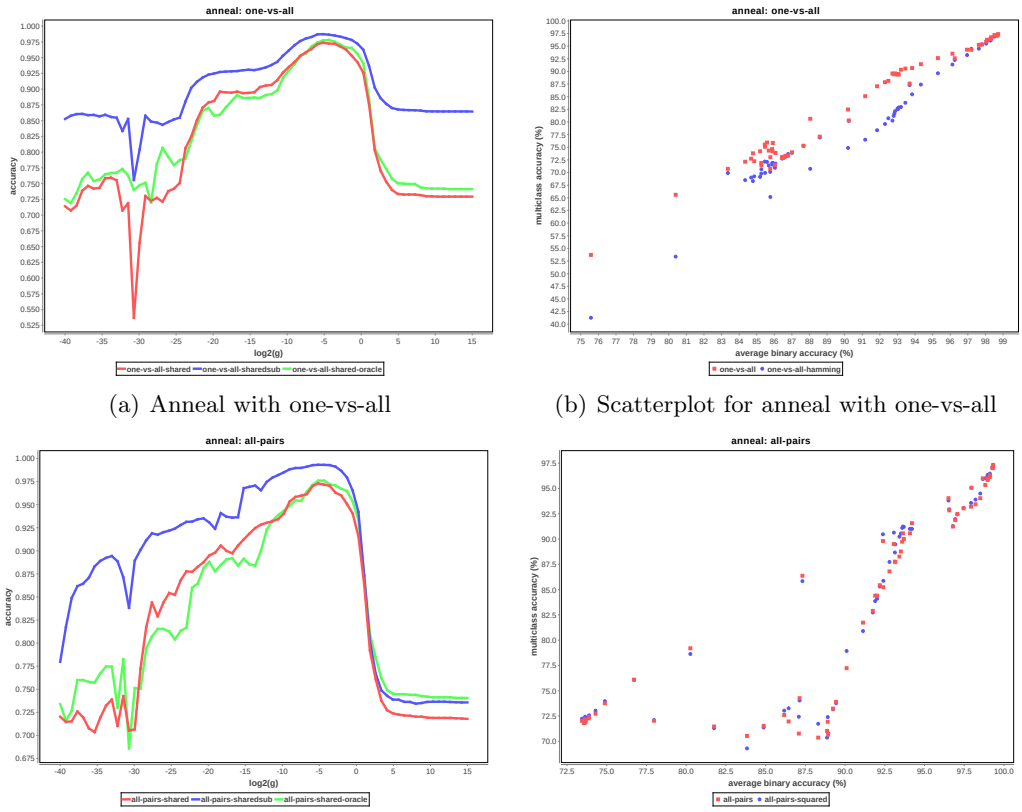
4.3.3 Average Binary and Multiclass Accuracy are Highly Correlated

In this section, we show that there is a strong correlation between binary and multiclass accuracy. This correlation has several ramification for model selection in multiclass classification with binary classifiers. First, it means that optimizing with respect to the binary average is a reasonable way to optimize the multiclass accuracy. Second, it means that more effective binary classifiers (possibly obtained through independent optimization) exhibit higher multiclass classification. First, we present an illustrative example in Section 4.3.3.1, then generalize results over all 20 data sets in Section 4.3.3.2.

4.3.3.1 Case Study: Anneal

Figure 4.15(a) indicates the accuracy as a function of the hyperparameter γ for the curves. The curve labeled *one-vs-all-shared* indicates the model selection performance for the one-vs-all method, while the curve labeled *one-vs-all-sharedsub* indicates the average binary accuracy for all subproblems as a function of γ. Note that these curves are qualitatively very highly correlated. The fact that the *one-vs-all-sharedsub* is several percent above the other curve indicates that the subproblems are much easier than the original multiclass problem. Third, the curve labeled *one-vs-all-shared-oracle* indicates

the value of the model as determined by hold-out test data (instead of on validation data as in the previous two curves). The similarity between the oracle curve and the *one-vs-all-shared* curve indicates that model selection on validation data is choosing models that are indeed effective on the test set.



(a) Anneal with one-vs-all

(b) Scatterplot for anneal with one-vs-all

(c) Anneal with all-pairs

(d) Scatterplot for anneal with all-pairs

Figure 4.15: Correlation between average binary accuracy and multiclass accuracy for the dataset *anneal*

4.3.3.2 Aggregate Results

To quantify the linearity of the relationship between the average binary accuracy and multiclass accuracy, we fit a linear model to the multiclass vs. average-binary scatter plots for each data set and compute the R^2 statistic. A R^2 value of 1.0 indicates that the data points are collinear (note that this result does not guarantee that the generating model from which the points were sampled is linear). Figure 4.16 indicates

the R^2 statistics for linear fits for each of the variables for the one-vs-all reduction. In this case, the average R^2 statistic is 0.791. For all-pairs (see Figure 4.17), the average R^2 statistic is 0.910. These results indicate that there is a significant correlation between the average binary accuracy and multiclass accuracy.

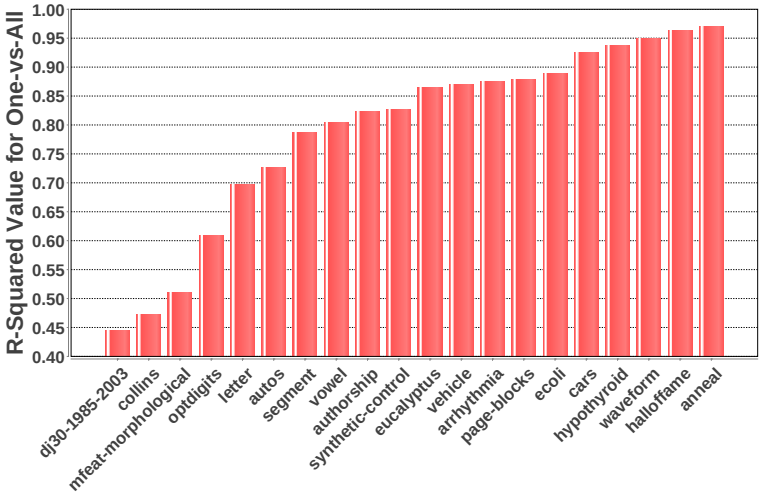


Figure 4.16: R^2 values indicating goodness-of-fit of a linear relationship between the average binary accuracy and the multiclass accuracy for the one-vs-all reduction.

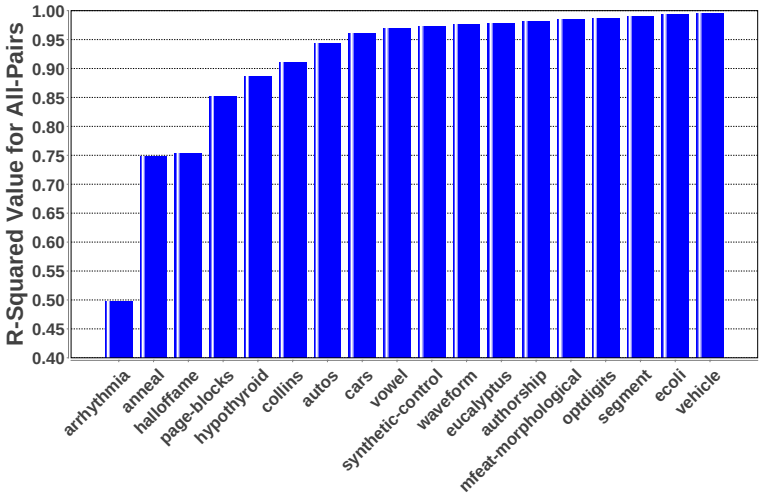


Figure 4.17: R^2 values indicating goodness-of-fit of a linear relationship between the average binary accuracy and the multiclass accuracy for the all-pairs reduction. The R^2 computation yielded NaN for the datasets *letter* and *dj30-1985-2003*, so those results are omitted.

4.3.4 Model Selection Effective on Non-Target Metric

One possible explanation for the advantage of shared-hyperparameter optimization over independent optimization is that shared-hyperparameter model selection is performed on the target metric while independent optimization is performed on a set of different (though related) metrics. In order to identify whether this mechanism explains the results obtained in Section 4.2, we implemented a new classification algorithm that operates by constraining hyperparameters to be shared, but selects the hyperparameter set by optimizing average binary subproblem accuracy rather than the true target metric of multiclass accuracy. We refer to this new technique as *shared-sub* because it constrains hyperparameters to be shared, but selects them based on *sub*problem accuracy instead of multiclass accuracy.

For all combinations of one-vs-all, all-pairs x Hamming, squared error, there are no statistically significant differences between shared-sub and shared-hyperparameters based on multiclass accuracy. The statistical test is pairwise Wilcoxon signed-ranks test at a $p \leq 0.05$ level. The Wilcoxon signed ranks test indicate difference for one-vs-all at $p \leq 0.6507$, for all-pairs at $p \leq 0.1042$, for one-vs-all-hamming at $p \leq 0.5678$ and for all-pairs-squared at $p \leq 0.1231$. (A three-way Holm test which also includes the "independent hyperparameters" method also indicates no statistically significant differences between the two methods). Shared w/subproblem-selection also beats independent optimization everywhere that shared w/multiclass-selection does, except for one-vs-all-hamming, in which case it ties with independent optimization.

The above results indicate that sharing hyperparameters provides regularization and protection against choosing suboptimal binary classifiers, even though selection is not performed on the true multiclass metric.

4.3.5 Oracle Selection Rules Out Sampling Problems

One possible explanation for the tendency for shared-hyperparameter model selection to outperform independent optimization is related to to the bias-variance tradeoff. In independent optimization, more parameters must be fit during model selection. In particular, in shared-hyperparameter model selection, only m hyperparameters must be determined, where m is the number of hyperparameters used by the learning algorithm (e.g. $m = 2$ for support vector machines with a Gaussian kernel). In independent optimization, $m \times L$ hyperparameters must be determined, where L is the number of subproblems. With the increased flexibility of the model, overfitting becomes a significant problem, and, in the absence of a sufficiently large validation set, incorrect hyperparameter values may be selected for many of the subproblems. In order to rule out the possibility of subsampling problems in determining the model-selection parameters for independent optimization, we introduce an oracle, which performs model selection not on validation data but rather on actual test data. While this technique cannot be used in practice, we experiment with it here in order to determine whether the advantage of shared-hyperparameter model selection is due to selection of hyperparameter sets that, while optimal with respect to the validation set, are suboptimal with respect to the actual test dataset. Note that the test data must be reduced into binary subproblems in order to use it as validation data on the binary subproblems in independent optimization.

The results for oracle selection are indicated in Table 4.5. Even when an oracle is used to select optimal models for both shared and independent methods, shared hyperparameters still outperforms independent optimization for the one-vs-all method. Independent optimization with oracle selection outperforms shared optimization for all-pairs and one-vs-all-hamming, and there is no statistically significant differences between the methods for all-pairs-squared.

	one-vs-all	all-pairs	one-vs-all-hamming	all-pairs-squared
accuracy	**shared** 0.0071	**indep** 5.72×10^{-6}	**indep** 4.77×10^{-5}	indep 0.3955

Table 4.5: Winning strategy for each combination of reduction and metric, when using the oracle selection method. Statistically significant wins (at $p \leq 0.05$) are highlighted. P-values from the Wilcoxon signed-ranks test are indicated after the winning strategy.

Another explanation for the accuracy of shared-hyperparameter optimization is that requiring re-use of the same hyperparameter set for each subproblem may provide some level of regularization for the problem; that is, it may provide essential smoothing that is omitted by independent-optimization.

4.4 Supplemental Results

In addition to the main results discussed in Section 4.2, we also discuss two supplementary results: a comparison of all methods in this study (Section 4.4.1) and discussion of performance under a probability calibration metric (Section 4.4.2).

4.4.1 One-vs-All vs. All-Pairs

Several studies have previously compared the all-pairs reduction to one-vs-all [85]. We also compared 7 of the algorithms (omitting one-vs-all-hamming-independent, a poorly-performing algorithm, so that familywise-error statistics would be computationally feasible). The results are indicated in Table 4.6, and the average ranks are depicted in Figure 4.18. The highest ranking algorithm is all-pairs using shared-hyperparameter optimization, but it is not statistically significantly different from all-pairs with squared-error decoding, one-vs-all with shared-hyperparameter optimization or one-vs-all with independent optimization.

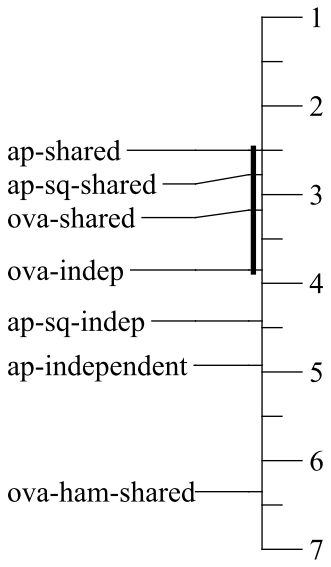


Figure 4.18: Average ranks of the 7 algorithms under study; algorithms not statistically significantly different from the top-scoring algorithm are connected to it with a vertical line.

4.4.2 Brier Metric

In order to evaluate the calibration of each method, we use the Brier metric [16], also known as the mean squared error (MSE) metric, in which $b(\boldsymbol{x}) = \frac{1}{d}\sum_j (t_j(\boldsymbol{x}) - \hat{p}_j(\boldsymbol{x}))^2$, where $\boldsymbol{x}$ is the input vector, t_j is the target probability for the the jth class, $\hat{p}_j$ is the probability estimated by the classification method and d is the dimensionality of the input vector. We take $t_j(\boldsymbol{x})$ to be 1 if the label belongs to the jth class and 0 otherwise. This metric is used in other related work such as Wu et al. [103] and Zadrozny [106]. In the results presented below, we use the rectified Brier score $r(\boldsymbol{x}) = (1 - b(\boldsymbol{x})) \times 100$ so that the results are structurally similar to percent accuracy; namely that higher is better and that the values range between 0% and 100%. For all four combinations of {one-vs-all, all-pairs} $\times$ {squared, Hamming}, there is no statistically significant difference between shared hyperparameter optimization and independent optimization on the Brier metric (see Table 4.2). One possible explanation for this result is that a significantly different metric is used for model selection than for model evaluation.

Future studies could experiment with using a binary Brier metric for purposes of model selection to see whether performance is improved.

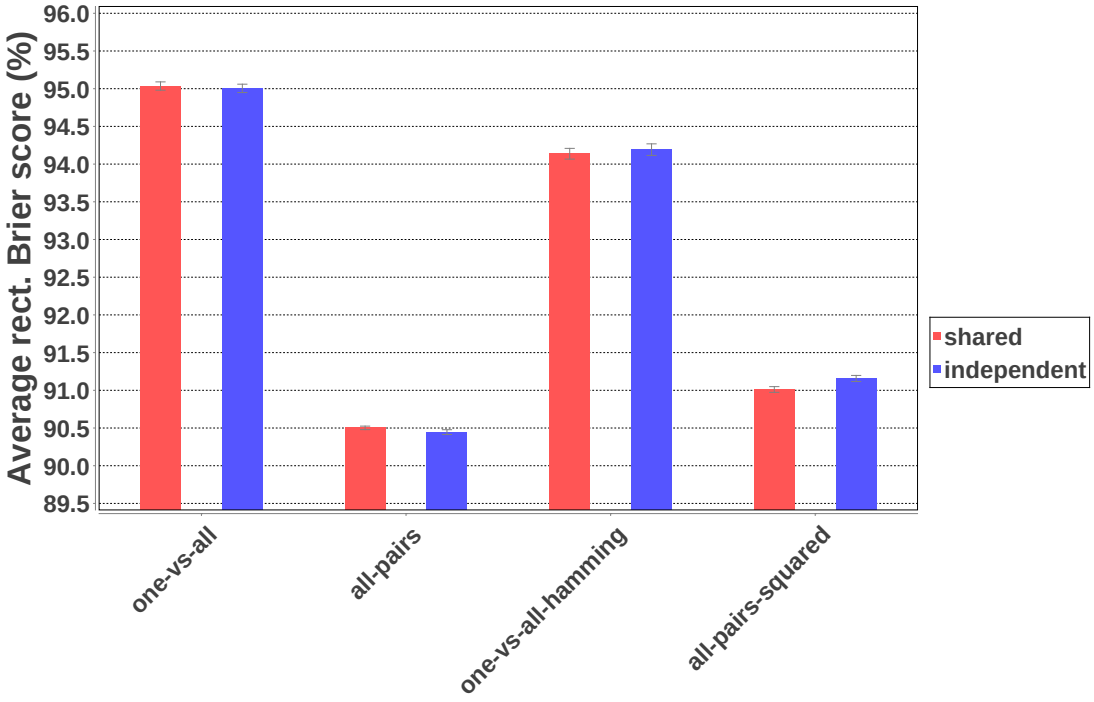


Figure 4.19: Average rectified Brier scores comparing independent to shared hyperparameter selection for each reduction.

4.5 Conclusion

In this chapter, we focused on the issue of model selection in solving multiclass classification problems by combining complementary binary classifiers. In particular, we showed that it is more effective to constrain hyperparameters to be shared across subproblems rather than independently optimizing each subproblem. This result is counterintuitive since it suggests that it is better to accept suboptimal binary models in order to improve the overall model. We performed several control studies to isolate the mechanism behind the effectiveness of shared-hyperparameter optimization, and showed that shared-hyperparameter maintains its superiority even when (a) the average binary accuracy is used as the metric instead of the true multiclass target metric and (b) the

oracle is used to ensure that suboptimal models are not selected due to low validation set sizes. We showed that binary subproblems in multiclass classification problems often have similar structure, and that this is an explanation for the superiority of shared-hyperparameter optimization. Conversely, we constructed a synthetic data set with differing optima for each subproblem and showed that independent optimization is more effective in this case. As a supplementary result, we also showed all-pairs with voting to rank higher than other algorithms, but not statistically significantly higher than one-vs-all in a 7-way comparison.

4.5.1 Future Work

This work could be extended to cover random coding matrices (with a variety of loss functions). Another interesting avenue of research would be to study (possibly heterogeneous) model combination in each of the subproblems, as in Caruana et al. [17] and see if the same results we obtained for model selection also apply to the case of model combination (i.e. averaging binary models vs. averaging multiclass models). Also, this research focused on the accuracy metric (0-1 loss) and Brier score metric. Future work could investigate custom or domain-specific multiclass loss functions, including those that do not necessarily have a corresponding analogous loss function for the associated binary subproblems. Additional studies could determine whether there is any advantage in standardizing each subproblem independently, instead of standardizing only the multiclass problem as done in our studies.

4.6 Appendix

4.6.1 Shared vs. Independent Optimization - Results

This section provides the results comparing shared-hyperparameter optimization to independent optimization on each data set. Tables 4.7-4.10 show the accuracy values

for one-vs-all and all-pairs under the squared-error and hamming decodings. Tables 4.11-4.14 show the rectified Brier scores.

Table 4.6: Results for 7 methods under the accuracy metric

dataset	ova-shared	ova-indep	ova-ham-shared	ap-shared	ap-independent	ap-sq-shared	ap-sq-indep
anneal	**97.47** (97.47)	97.33 (97.33)	97.20 (97.20)	97.27 (97.27)	97.13 (97.13)	97.20 (97.20)	97.27 (97.27)
arrhythmia	72.02 (72.02)	72.56 (72.56)	66.51 (66.51)	72.09 (72.09)	**73.80** (73.80)	71.09 (71.09)	73.41 (73.41)
authorship	99.33 (99.33)	99.40 (99.40)	98.80 (98.80)	99.33 (99.33)	99.20 (99.20)	**99.47** (99.47)	99.07 (99.07)
autos	70.74 (70.74)	71.18 (71.18)	64.85 (64.85)	**73.68** (73.68)	67.79 (67.79)	70.74 (70.74)	68.82 (68.82)
cars	78.53 (78.53)	78.38 (78.38)	**79.19** (79.19)	77.65 (77.65)	77.06 (77.06)	78.24 (78.24)	76.69 (76.69)
collins	42.73 (42.73)	40.47 (40.47)	26.27 (26.27)	41.87 (41.87)	43.20 (43.20)	**44.87** (44.87)	44.33 (44.33)
dj30-1985-2003	**22.84** (22.84)	18.36 (18.36)	17.31 (17.31)	19.40 (19.40)	16.12 (16.12)	19.70 (19.70)	18.96 (18.96)
ecoli	**86.41** (86.41)	86.31 (86.31)	84.27 (84.27)	86.12 (86.12)	83.01 (83.01)	85.92 (85.92)	83.30 (83.30)
eucalyptus	54.40 (54.40)	53.13 (53.13)	39.40 (39.40)	55.93 (55.93)	56.80 (56.80)	57.53 (57.53)	**58.33** (58.33)
halloffame	90.27 (90.27)	**90.67** (90.67)	89.80 (89.80)	90.07 (90.07)	90.27 (90.27)	90.13 (90.13)	90.07 (90.07)
hypothyroid	92.13 (92.13)	93.07 (93.07)	90.87 (90.87)	**94.00** (94.00)	91.60 (91.60)	93.80 (93.80)	91.27 (91.27)
letter	**55.59** (55.59)	44.56 (44.56)	39.71 (39.71)	55.29 (55.29)	36.91 (36.91)	53.53 (53.53)	45.00 (45.00)
mfeat-morphological	74.40 (74.40)	72.53 (72.53)	65.47 (65.47)	**75.80** (75.80)	73.27 (73.27)	75.40 (75.40)	72.33 (72.33)
optdigits	93.80 (93.80)	93.33 (93.33)	89.47 (89.47)	**94.07** (94.07)	90.47 (90.47)	93.80 (93.80)	92.60 (92.60)
page-blocks	89.60 (89.60)	89.33 (89.33)	89.33 (89.33)	**91.00** (91.00)	90.07 (90.07)	89.93 (89.93)	89.93 (89.93)
segment	91.53 (91.53)	90.93 (90.93)	85.93 (85.93)	**92.07** (92.07)	89.20 (89.20)	91.80 (91.80)	90.13 (90.13)
synthetic-control	98.13 (98.13)	98.13 (98.13)	98.00 (98.00)	**98.33** (98.33)	91.73 (91.73)	**98.33** (98.33)	98.07 (98.07)
vehicle	79.53 (79.53)	80.33 (80.33)	72.27 (72.27)	**80.40** (80.40)	79.13 (79.13)	80.20 (80.20)	80.13 (80.13)
vowel	80.13 (80.13)	78.13 (78.13)	74.20 (74.20)	84.60 (84.60)	78.47 (78.47)	**84.67** (84.67)	79.13 (79.13)
waveform	81.47 (81.47)	81.00 (81.00)	79.73 (79.73)	**83.93** (83.93)	83.07 (83.07)	83.87 (83.87)	83.00 (83.00)
average	77.55	76.46	72.43	**78.14**	75.41	78.01	76.59
average rank	3.2	3.8	6.4	**2.5**	4.9	2.8	4.4

Table 4.7: Average accuracy over 10 random splits for shared and independent model selection strategies with the one-vs-all reduction, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	one-vs-all-shared	one-vs-all-independent
anneal	**97.47** (0.98)	97.33 (0.94)
arrhythmia	72.02 (3.42)	**72.56** (2.26)
authorship	99.33 (0.54)	**99.40** (0.58)
autos	70.74 (4.13)	**71.18** (4.05)
cars	**78.53** (1.86)	78.38 (2.84)
collins	**42.73** (3.88)	40.47 (4.44)
dj30-1985-2003	**22.84** (3.59)	18.36 (5.59)
ecoli	**86.41** (4.09)	86.31 (4.61)
eucalyptus	**54.40** (5.41)	53.13 (2.41)
halloffame	90.27 (2.04)	**90.67** (2.63)
hypothyroid	92.13 (1.98)	**93.07** (3.00)
letter	**55.59** (8.85)	44.56 (10.35)
mfeat-morphological	**74.40** (3.08)	72.53 (4.58)
optdigits	**93.80** (1.51)	93.33 (1.83)
page-blocks	**89.60** (2.18)	89.33 (1.96)
segment	**91.53** (1.89)	90.93 (1.78)
synthetic-control	**98.13** (1.25)	**98.13** (0.93)
vehicle	79.53 (2.97)	**80.33** (3.15)
vowel	**80.13** (1.91)	78.13 (2.59)
waveform	**81.47** (4.25)	81.00 (3.97)
average	**77.55**	76.46
average rank	**1.3**	1.7

Table 4.8: Average accuracy over 10 random splits for shared and independent model selection strategies with the all-pairs reduction, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	all-pairs-shared	all-pairs-independent
anneal	**97.27** (1.19)	97.13 (1.18)
arrhythmia	72.09 (3.47)	**73.80** (2.22)
authorship	**99.33** (0.54)	99.20 (0.98)
autos	**73.68** (4.35)	67.79 (4.41)
cars	**77.65** (2.90)	77.06 (3.04)
collins	41.87 (3.74)	**43.20** (3.47)
dj30-1985-2003	**19.40** (4.61)	16.12 (4.03)
ecoli	**86.12** (4.69)	83.01 (5.22)
eucalyptus	55.93 (4.63)	**56.80** (5.09)
halloffame	90.07 (1.87)	**90.27** (1.99)
hypothyroid	**94.00** (1.44)	91.60 (2.71)
letter	**55.29** (9.12)	36.91 (8.92)
mfeat-morphological	**75.80** (2.65)	73.27 (4.12)
optdigits	**94.07** (1.42)	90.47 (4.28)
page-blocks	**91.00** (1.70)	90.07 (2.40)
segment	**92.07** (1.87)	89.20 (3.04)
synthetic-control	**98.33** (0.85)	91.73 (12.50)
vehicle	**80.40** (3.18)	79.13 (2.83)
vowel	**84.60** (2.82)	78.47 (3.02)
waveform	**83.93** (2.07)	83.07 (2.74)
average	**78.14**	75.41
average rank	**1.2**	1.8

Table 4.9: Average accuracy over 10 random splits for shared and independent model selection strategies with the one-vs-all reduction with Hamming decoding, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	one-vs-all-hamming-shared	one-vs-all-hamming-independent
anneal	**97.20** (1.21)	96.60 (1.35)
arrhythmia	**66.51** (7.95)	65.04 (6.02)
authorship	**98.80** (0.61)	98.47 (1.44)
autos	**64.85** (4.24)	62.35 (6.97)
cars	**79.19** (4.23)	77.50 (4.78)
collins	**26.27** (3.83)	25.87 (4.16)
dj30-1985-2003	**17.31** (5.23)	15.82 (5.46)
ecoli	84.27 (4.53)	**84.76** (4.60)
eucalyptus	**39.40** (4.63)	38.93 (5.10)
halloffame	**89.80** (1.99)	88.47 (3.28)
hypothyroid	90.87 (1.69)	**91.87** (2.75)
letter	**39.71** (7.20)	34.12 (7.13)
mfeat-morphological	**65.47** (3.09)	65.20 (3.20)
optdigits	**89.47** (2.55)	88.67 (3.16)
page-blocks	**89.33** (1.44)	88.73 (2.10)
segment	**85.93** (2.91)	84.53 (3.74)
synthetic-control	**98.00** (1.30)	97.13 (0.95)
vehicle	**72.27** (4.31)	71.53 (3.81)
vowel	**74.20** (3.33)	73.00 (5.19)
waveform	79.73 (3.57)	**80.87** (3.72)
average	**72.43**	71.47
average rank	**1.2**	1.8

Table 4.10: Average accuracy over 10 random splits for shared and independent model selection strategies with the all-pairs-squared reduction, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	all-pairs-squared-shared	all-pairs-squared-independent
anneal	97.20 (1.25)	**97.27** (1.02)
arrhythmia	71.09 (2.92)	**73.41** (2.31)
authorship	**99.47** (0.61)	99.07 (0.90)
autos	**70.74** (3.63)	68.82 (3.59)
cars	**78.24** (2.67)	76.69 (2.86)
collins	**44.87** (3.24)	44.33 (3.07)
dj30-1985-2003	**19.70** (4.03)	18.96 (5.41)
ecoli	**85.92** (4.42)	83.30 (5.09)
eucalyptus	57.53 (5.92)	**58.33** (4.73)
halloffame	**90.13** (1.88)	90.07 (2.14)
hypothyroid	**93.80** (1.44)	91.27 (2.77)
letter	**53.53** (8.94)	45.00 (8.00)
mfeat-morphological	**75.40** (3.88)	72.33 (3.71)
optdigits	**93.80** (1.22)	92.60 (2.28)
page-blocks	**89.93** (1.68)	**89.93** (2.16)
segment	**91.80** (1.75)	90.13 (2.99)
synthetic-control	**98.33** (0.85)	98.07 (1.27)
vehicle	**80.20** (2.69)	80.13 (2.47)
vowel	**84.67** (2.70)	79.13 (3.61)
waveform	**83.87** (2.24)	83.00 (2.87)
average	**78.01**	76.59
average rank	**1.2**	1.8

Table 4.11: Average rectified Brier score over 10 random splits for shared and independent model selection strategies with the one-vs-all reduction, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	one-vs-all-shared	one-vs-all-independent
anneal	**98.95** (0.36)	98.90 (0.29)
arrhythmia	91.44 (1.13)	**91.62** (0.63)
authorship	99.48 (0.21)	**99.62** (0.21)
autos	91.43 (1.09)	**91.79** (0.89)
cars	**89.23** (0.86)	88.83 (1.26)
collins	**93.55** (0.21)	93.29 (0.34)
dj30-1985-2003	**95.59** (0.16)	95.52 (0.32)
ecoli	94.58 (1.64)	**94.60** (1.54)
eucalyptus	**88.34** (0.75)	88.10 (0.58)
halloffame	**94.73** (1.14)	94.66 (1.21)
hypothyroid	95.92 (0.83)	**96.05** (1.35)
letter	**96.58** (0.31)	96.13 (0.51)
mfeat-morphological	**96.10** (0.41)	95.96 (0.47)
optdigits	**98.97** (0.22)	98.90 (0.24)
page-blocks	**96.91** (0.56)	96.91 (0.59)
segment	**98.04** (0.35)	97.95 (0.32)
synthetic-control	99.33 (0.64)	**99.46** (0.14)
vehicle	92.89 (0.82)	**92.97** (0.75)
vowel	97.26 (0.24)	**97.32** (0.25)
waveform	91.40 (1.28)	**91.51** (1.43)
average	**95.04**	95.00
average rank	**1.4**	1.6

Table 4.12: Average rectified Brier score over 10 random splits for shared and independent model selection strategies with the all-pairs reduction, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	all-pairs-shared	all-pairs-independent
anneal	**90.07** (0.07)	90.06 (0.32)
arrhythmia	**88.31** (0.69)	88.04 (0.74)
authorship	90.24 (0.06)	**90.29** (0.16)
autos	**88.08** (0.40)	87.95 (0.30)
cars	**86.24** (0.84)	85.92 (0.94)
collins	92.62 (0.05)	**92.66** (0.04)
dj30-1985-2003	**95.36** (0.02)	95.33 (0.04)
ecoli	88.94 (0.55)	**89.18** (1.18)
eucalyptus	**87.28** (0.40)	87.19 (0.52)
halloffame	89.91 (0.68)	**89.99** (0.49)
hypothyroid	**91.16** (0.35)	90.60 (0.62)
letter	**95.12** (0.05)	95.05 (0.04)
mfeat-morphological	92.49 (0.06)	**92.56** (0.15)
optdigits	92.57 (0.02)	**92.60** (0.05)
page-blocks	89.59 (0.24)	**89.94** (0.64)
segment	**90.81** (0.10)	90.74 (0.07)
synthetic-control	**90.35** (0.02)	90.30 (0.27)
vehicle	**88.88** (0.47)	88.65 (0.39)
vowel	92.98 (0.03)	**93.02** (0.07)
waveform	**89.03** (0.47)	88.84 (0.61)
average	**90.50**	90.45
average rank	**1.4**	1.6

Table 4.13: Average rectified Brier score over 10 random splits for shared and independent model selection strategies with the one-vs-all reduction with Hamming decoding, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	one-vs-all-hamming-shared	one-vs-all-hamming-independent
anneal	98.76 (0.52)	**98.80** (0.41)
arrhythmia	89.80 (0.88)	**89.93** (0.91)
authorship	**99.73** (0.14)	99.63 (0.27)
autos	90.30 (1.48)	**90.57** (1.06)
cars	**88.42** (1.17)	88.30 (1.81)
collins	92.10 (0.32)	**92.19** (0.40)
dj30-1985-2003	**95.45** (0.34)	95.18 (0.77)
ecoli	92.88 (2.30)	**93.34** (2.25)
eucalyptus	85.30 (1.14)	**85.88** (1.29)
halloffame	93.94 (1.57)	**94.21** (1.24)
hypothyroid	95.13 (0.60)	**95.73** (1.29)
letter	**96.10** (0.42)	95.80 (0.62)
mfeat-morphological	95.04 (0.51)	**95.14** (0.57)
optdigits	**98.74** (0.34)	98.64 (0.41)
page-blocks	96.43 (0.51)	**96.58** (0.74)
segment	**97.66** (0.45)	97.54 (0.50)
synthetic-control	**99.50** (0.34)	99.39 (0.22)
vehicle	**90.98** (1.44)	90.78 (0.90)
vowel	**97.07** (0.28)	96.99 (0.49)
waveform	**89.41** (2.33)	89.22 (2.27)
average	94.14	**94.19**
average rank	**1.5**	**1.5**

Table 4.14: Average rectified Brier score over 10 random splits for shared and independent model selection strategies with the all-pairs-squared reduction, with the standard deviation indicated in parentheses. The winner for each data set is indicated in bold.

dataset	all-pairs-squared-shared	all-pairs-squared-independent
anneal	91.26 (0.69)	**92.02** (1.23)
arrhythmia	**89.67** (0.92)	89.20 (0.52)
authorship	91.42 (0.34)	**92.29** (0.64)
autos	**88.45** (0.63)	88.44 (0.55)
cars	87.01 (1.15)	**87.91** (1.27)
collins	92.63 (0.06)	**92.66** (0.04)
dj30-1985-2003	**95.36** (0.03)	95.34 (0.03)
ecoli	89.82 (0.59)	**90.57** (0.71)
eucalyptus	**87.36** (0.35)	87.29 (0.29)
halloffame	**92.60** (1.74)	92.24 (1.14)
hypothyroid	**91.31** (0.74)	90.81 (0.94)
letter	**95.07** (0.04)	95.06 (0.03)
mfeat-morphological	92.45 (0.12)	**92.71** (0.13)
optdigits	92.59 (0.06)	**92.65** (0.06)
page-blocks	90.02 (0.59)	**90.86** (0.80)
segment	90.75 (0.06)	**90.77** (0.16)
synthetic-control	**90.88** (0.21)	90.80 (0.25)
vehicle	**89.30** (0.29)	89.24 (0.35)
vowel	92.90 (0.04)	**93.05** (0.07)
waveform	**89.37** (0.46)	89.23 (0.44)
average	91.01	**91.16**
average rank	**1.5**	**1.5**

Chapter 5

Probabilistic Pairwise Classification

Chapter Abstract

Pairwise classification is a technique for solving multiclass classification problems by constructing a classifier to discriminate between each pair of classes. Early pairwise classification techniques combined discrete votes from each pairwise classifier to produce a multiclass classification [38]. Subsequent work has shown the advantage of combining probabilistic predictions (instead of votes) to produce a probability distribution over classes (instead of a discrete classification) [51, 106, 103]. Pairwise classification methods have been criticized because each pairwise classifier is trained on only two of the classes but makes predictions for instances from any class [51, 23]. In this chapter, we propose a new pairwise classification technique that addresses this problem by weighting each pairwise prediction with an estimated probability that the instance belongs to the pair. The technique is based on the Theorem of Total Probability, and relies on only the assumption that each instance is assigned exactly one label. Furthermore, our method is conceptually simpler and easier to implement than other methods that incorporate and produce probabilities in pairwise classification. Experimental studies on real world data sets indicate that our proposed technique performs better than voted pairwise classification [38] and the pairwise coupling methods from Hastie and Tibshirani [51] and Wu et al. [103], at the cost of increased computational demands.

5.1 Introduction

Multiclass classification is an important machine learning problem, encompassing domains such as handwritten text recognition, heartbeat arrhythmia monitoring, protein structure prediction [73], and many others. However, several supervised machine learning techniques such as support vector machines [10] and AdaBoost [35] are designed for solving binary classification problems. While many modifications and extensions have been proposed for adapting these methods to multiclass classification problems, another prominent line of research instead focuses on reducing the multiclass problem to a set of binary problems. The most general framework for solving multiclass classification problems with binary classifiers is the loss-based decoding framework [2], which is flexible in how the multiclass problem is coded into binary classification problems and how the binary classifier predictions are decoded as a multiclass prediction. Loss-based decoding generalizes a widely-used technique called one-vs-all, which creates one binary classification problem for each class to discriminate the class from the union of the remaining classes. In this chapter, we focus on a related reduction technique called pairwise classification (or all-pairs), which creates a binary classifier to discriminate between each pair of classes.

5.1.1 Pairwise Classification

In pairwise classification (also known as all-pairs, all-vs-all (or AVA) [85], round-robin classification [40] and 1-against-1 (or 11) [105]), a k-class classification problem is reduced to $\frac{k(k-1)}{2}$ subproblems, one for each pair of classes. For example, Figure 5.1 indicates a $k = 3$ class problem with the decision boundary between classes A and C. At prediction time, each binary classifier votes for one class, and the class with the most votes is selected as the multiclass prediction, with ties broken randomly [38]. This original formulation, which we refer to as voted pairwise classification (VPC), ignores

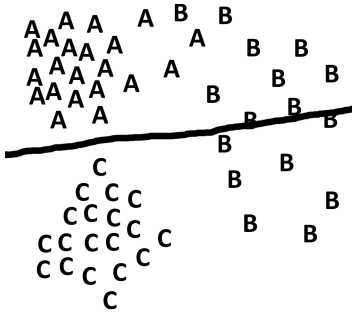


Figure 5.1: Illustration of an A-C decision boundary in a 2D, 3-class example of the all-pairs reduction.

confidence predictions from each binary classifier, instead combining a discrete vote from each classifier, and produces a multiclass prediction rather than a probability distribution over classes. Despite these drawbacks, voted pairwise classification has several advantages. Even though pairwise classification creates a large number of subproblems (quadratic in the number of classes), each subproblem contains only a small fraction of the instances in the multiclass classification problem. Furthermore, since these subproblems involve only two classes, they can be significantly simpler than the original k-class problem. That is, the decision boundary between two classes can be less complex than a decision boundary discriminating between three or more classes. Another important feature of pairwise classification is consistency.

5.1.1.1 Consistency

A desirable property of any reduction scheme is consistency, which guarantees optimality of the multiclass classifier if each of the binary classifiers is optimal [5]. When pairwise classification was introduced in 1996, Friedman proved it to be consistent by showing that Bayes optimal binary classifiers combine to produce a Bayes optimal multiclass classifier. The Bayes optimal decision is given by

$$\hat{y}(\boldsymbol{x}) = \arg\max_{i=1..k} p(y = c_i | \omega \in L, \boldsymbol{x}) \tag{5.1}$$

where $L = \{c_1, c_2, ..., c_k\}$ is the set of possible labels, y is the true label, $\boldsymbol{x}$ is the input feature vector and $\hat{y}$ is the predicted label. This is equivalent to

$$\hat{y}(\boldsymbol{x}) = \arg\max_{i=1..k} \sum_{j=1..k} 1(\frac{p_i}{p_i + p_k} > \frac{p_k}{p_i + p_k}) \tag{5.2}$$

where $1(x) = 1$ if x is true and 0 otherwise. This reduces to

$$\hat{y}(\boldsymbol{x}) = \arg\max_{i=1..k} \sum_{j=1..k} 1(p(y = c_i | y \in \{c_i, c_j\}, \boldsymbol{x}) > p(y = c_j | y \in \{c_i, c_j\}, \boldsymbol{x})) \tag{5.3}$$

Therefore, given reliable values for $p(y = c_i | y \in \{c_i, c_j\}, \boldsymbol{x})$, binary reduction under the all-pairs reduction yields the Bayes optimal decision. This analysis assumes $p(y = c_i | y \in \{c_i, c_k\}, \boldsymbol{x})$ can be determined exactly for each subproblem, whereas in practice, it is difficult or impossible to accurately obtain this probability distribution given a finite sample size. However, even with finite sample sizes this analysis shows the all-pairs reduction to be consistent and motivates solving each subproblem independently and as accurately as possible. Friedman argues that the all-pairs reduction works according to a bias-variance tradeoff, increasing variance slightly at the cost of significantly reduced bias [38].

Friedman presents experimental results using nearest-neighbor classifiers and decision tree (CART) methods with axis-oriented splits and linear combination splits on synthetic data sets generated from Gaussian mixtures. For the nearest-neighbor algorithm, Friedman identified that all-pairs is more accurate than one-vs-all and suggests that this is due to the ability to tune the regularization hyperparameter (number of neighbors) separately in each subproblem. He also observed that the all-pairs reduction outperforms the one-vs-all reduction for CART methods with linear combination, and suggested that is because the one-vs-all problems are significantly more difficult than the two-class problems in the all-pairs reduction.

5.1.1.2 Pairwise Coupling Schemes

While Friedman's pairwise classification method produces a class prediction, several authors have proposed algorithms for obtaining multiclass probabilities from pairwise probability predictions. These methods are often called pairwise coupling methods because the predicted pairwise probabilities are coupled together to produce the multiclass classification. In this paper, we use the terms pairwise coupling and pairwise classification interchangeably. Pairwise coupling methods use discriminative binary classifier to estimate the pairwise probabilities $\mu_{ij} = p(y = c_i | y = c_i \text{ or } c_j, \boldsymbol{x})$ (note that $\mu_{ij} = \frac{p_i}{p_i + p_j}$) and differ in how the μ_{ij} are used to estimate the multiclass probability estimates $\boldsymbol{p} = \{p_1, p_2, ..., p_k\} = f(\hat{\mu}_{ij}(\boldsymbol{x}))$.

Hastie & Tibshirani In 1996, Hastie & Tibshirani proposed a pairwise coupling algorithm that works by tuning the k-dimensional multiclass probability estimate $\boldsymbol{p} = \{p_1, p_2, ..., p_k\}$ in order to minimize the Kullback-Liebler divergence between the obtained pairwise estimates and the true pairwise probability values [51]. With the true probability estimates defined as $\mu_{ij} = p(y = c_i | y \in \{c_i, c_j\}, \boldsymbol{x})$, a discriminative binary classification algorithm is then used to obtain pairwise probability estimates $\hat{\mu}_{ij} \approx \mu_{ij}$. An initial guess vector $\boldsymbol{p}$ is selected (with all elements $p_i \geq 0.0$ for $i = 1..k$, and normalized so that $\sum_{i=1}^{k} p_i = 1$) and used to compute a new iteration of values for μ_{ij}. Then by minimizing the Kullback-Liebler divergence between $\hat{\mu}_{ij}$ and μ_{ij} in this iteration, we obtain an updated $\boldsymbol{p}$ vector. This algorithm is proved to converge to the minimum of the KL divergence between $\hat{\mu}_{ij}$ and μ_{ij}. For more details, see Hastie & Tibshirani [51], or the description in Wu et al. [103].

Wu, Lin & Weng In 2004, Wu et al. [103] proposed the following procedure for pairwise coupling to produce probability estimates as an improvement over the method proposed by Hastie & Tibshirani. Again defining $\mu_{ij} = p(y = c_i | y = c_i \text{ or } c_j, \boldsymbol{x})$, discriminative binary classification algorithms are used to obtain pairwise estimates

$\hat{\mu}_{ij} \approx \mu_{ij}$. Using the identity

$$\mu_{ij} = \frac{p_i}{p_i + p_j}$$

we have also the identity

$$\frac{\mu_{ij}}{\mu_{ji}} = \frac{p_i}{p_j}$$

The multiclass probability vector is then approximated as the solution to the constrained optimization problem

$$\min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j \neq i} (\hat{\mu}_{ji} p_i - \hat{\mu}_{ij} p_j)^2$$

subject to

$$\sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i$$

This optimization problem reduces to a linear system, which is also solvable with an iterative process, and is proved to be convergent. Wu et al. show this pairwise classification method to be superior to the method proposed by Hastie & Tibshirani and voted pairwise classification over a variety of data sets [103].

5.1.1.3 A Problem with Pairwise Classification

Pairwise classification techniques have been criticized because each pairwise classifier is trained on only two of the k classes, but makes predictions for instances of any of the k classes. Evaluation of classifiers on a different distribution than the training distribution can be problematic because it introduces unnecessary bias into the predictions [51, 23]. Consider, for example, a 4-class problem with classes {A,B,C,D}. Assume without loss of generality that we wish to classify a test point whose true label is A. At prediction time, all 6 classifiers are evaluated; however, the predictions from the BC, CD and BD classifiers will be unreliable since they were not trained on distributions containing instances with class A. If there is bias, such as D instances being more easily mistaken for B than for C, multiclass classification errors can occur. Hastie and Tibshirani show a simulation result that indicates this bias to be a real problem, but

they comment that other (non-pairwise) approaches may not fare any better. Furthermore, they show that when this bias is present that the probabilistic predictions of the multiclass classifier are more evenly distributed, indicating reduced confidence in the predicted class.

5.2 Probabilistic Pairwise Classification

5.2.1 Proposed Method

In this section, we introduce a new pairwise coupling technique called probabilistic pairwise classification (PPC) that weights each pairwise probability with a probability that the instance belongs to the pair. This approach has the the following features: it incorporates predicted probabilities from the base classifiers (rather than discrete votes), it produces a posterior probability (rather than a discrete classification), it compensates for the problem that pairwise classifiers make predictions for instances with different class labels than those used during training, and it has the property of consistency. It is also conceptually simple, theoretically motivated and easy to implement. First, we derive the probabilistic pairwise classification rule. Given N mutually exclusive and exhaustive events $a_1, ...a_N$ (i.e. with $\sum_{i=1}^{N} p(a_i) = 1$), the Theorem of Total Probability is

$$p(b|\boldsymbol{x}) = \sum_{i=1}^{N} p(b|\boldsymbol{x}, a_i)p(a_i) \tag{5.4}$$

Letting $L = \{c_1..c_k\}$ be the set of labels in a k-class classification problem, and substituting into Equation (5.4) the values $N = 2$, $a_1 = c_i \cup c_j$, $a_2 = L - c_i - c_j$ and $b = c_i$, we have

$$p(c_i|L, \boldsymbol{x}) = p(c_i|c_i \cup c_j, \boldsymbol{x})p(c_i \cup c_j|L, \boldsymbol{x}) + p(c_i|L - c_i - c_j, \boldsymbol{x})p(L - c_i - c_j|L, \boldsymbol{x}) \tag{5.5}$$

since $p(c_i \cup c_j|L, \boldsymbol{x}) + p(L - c_i - c_j|\boldsymbol{x}) = 1$. However, $p(c_i|L - c_i - c_j, \boldsymbol{x}) = 0$ since no instance can be labeled both c_i and any label from the set $L - c_i - c_j$. Therefore,

Equation (5.5) reduces to

$$p(c_i|\boldsymbol{x}) = p(c_i|c_i \cup c_j, \boldsymbol{x})p(c_i \cup c_j|L, \boldsymbol{x}) \tag{5.6}$$

In practice, we do not obtain true probabilities from the trained classifiers (whether density-based or discriminative), but rather probability estimates $\hat{p}$. Therefore, we substitute the predicted probabilities and make the prediction $\hat{p}(c_i|\boldsymbol{x}) = \hat{p}(c_i|c_i \cup c_j, \boldsymbol{x})\hat{p}(c_i \cup c_j|L, \boldsymbol{x})$. Here, $\hat{p}(c_i|c_i \cup c_j, \boldsymbol{x})$ is the estimated probability that the given instance $\boldsymbol{x}$ belongs to class c_i given that it belongs to either class c_i or c_j and is produced by a discriminative binary classifier trained using points from only classes c_i and c_j, where c_i is the positive indicator class and c_j is the negative indicator class. Fürnkranz provides theory and evidence that suggests that these pairwise discrimination problems are much simpler than corresponding one-vs-all discrimination problems [40]. The more difficult problem is estimating $\hat{p}(c_i \cup c_j|L, \boldsymbol{x})$, the probability that an instance belongs to either class c_i or c_j, selecting from all labels L. Depending on the properties of the underlying distributions, predicting $\hat{p}(c_i \cup c_j|L, \boldsymbol{x})$ may be more difficult than predicting the posterior probability $p(c_i|L, \boldsymbol{x})$ itself. Because the predictions are based on estimated values for probabilities instead of true probabilities, we average over all possibilities for $j \neq i$ (rather than arbitrarily selecting a single value for j in Equation (5.6)), giving

$$\hat{p}(c_i|L, \boldsymbol{x}) = \frac{1}{k-1} \sum_{j \neq i} \hat{p}(c_i|c_i \cup c_j, \boldsymbol{x})\hat{p}(c_i \cup c_j|L, \boldsymbol{x}) \tag{5.7}$$

Note that the average is over $k-1$ terms since the $i = j$ case can't be used for discrimination. The motivation for this averaging is that the individual classifiers may make noisy predictions and that averaging may be able to decrease the multiclass predictive error, as long as the classifiers are not systematically biased in the same way. In practice, the values $\hat{p}(c_i|L, \boldsymbol{x})$ should also be normalized so that $\sum_i \hat{p}(c_i|L, \boldsymbol{x}) = 1$.

Probabilistic pairwise classification can be used with any classification algorithm capable of performing binary classification (including multiclass classifiers); the classification algorithm used with PPC (or any pairwise coupling method) is known as the base

classifier. In our experimental studies, we also investigate treating multiclass classifiers as binary classifiers, including decision trees, nearest-neighbor classifiers and random forests.

Other decompositions are possible, for example:

$$p(a|\boldsymbol{x}) = p(a|ab, \boldsymbol{x})p(ab|abc, \boldsymbol{x})p(abc|L, \boldsymbol{x})$$

Such decompositions may be valuable in problems for which domain-specific knowledge can be used to construct or simplify the intermediate models. However, in our studies, we assume that no domain specific knowledge is available, and focus solely on the reduction to pairwise and pair-vs-rest terms.

Also note that the assumption $p(c_i|L - c_i - c_j, \boldsymbol{x}) = 0$ means this technique is unsuitable for multi-label classification problems (in which each instance may be assigned more than one class label).

5.2.2 Computational Demand

Probabilistic pairwise classification is more computationally demanding than either voted pairwise classification or one-vs-all. In one-vs-all, there are k problems, each with N data points. In the voted all-pairs reduction or other pairwise coupling schemes, there are $k(k-1)/2$ classifiers, each with approximately $2/N$ training data points (for a balanced problem). In probabilistic pairwise classification, there are $k(k-1)$ classifiers, 2 for each of the $k(k-1)/2$ pairs, one for discriminating between elements of the pair and one for discriminating between the pair and the rest of the classes. The pairwise classifiers are each trained on approximately $2/N$ data points as above, but the pair-vs-rest classifiers are each trained on the entire training set. These values are summarized in Table 5.1. Therefore, probabilistic pairwise classification is more computationally demanding than both one-vs-all and pairwise classification. For applications that have a large number of training instances or classes, probabilistic pairwise classi-

Table 5.1: Computational complexity of one-vs-all (OVA), pairwise coupling (PC) and probabilistic pairwise classification (PPC)

	OVA	PC	PPC
subproblems	k	k(k-1)/2	k(k-1)
instances per subproblem	N	$2N/k$	N (half) + $2N/k$ (other half)
computational complexity/SVM	$O(kN^3)$	$O(k^{-1}N^3)$	$O(k^2N^3)$

fication may be inappropriate; however, in situations in which accuracy is the primary goal, probabilistic pairwise classification has the potential to increase accuracy. It is important to note that all of the aforementioned methods are fully parallelizable. For offline applications in which training time is the bottleneck, the main difference between PPC and other pairwise coupling methods is the training of the (k*(k-1))/2 pair-vs-rest classifiers, which each train on all N data points instead of just 2N/k data points.

For a base classifier with a known computational complexity, the time complexity of the various methods can be computed. For instance, the binary SVM has a computational complexity for training time of $O(N^3)$, where N is the number of training data points. For one-vs-all, each of k subproblems must train on all N points, so the computational complexity is $O(kN^3)$. For balanced pairwise coupling methods, for which the number of points per subproblem is approximately $N_s = 2N/k$, the computational complexity is $O(k^2N_s^3) = O(k^{-1}N^3)$. For PPC, the computational complexity of each subproblem is $O(k^2N^3) + O(k^{-1}N^3) = O(k^2N^3)$, since each pairwise subproblem must train on all N data points. These computational complexities are summaries in the last row of Table 5.1.

5.2.3 Discriminative Classifiers

It can be argued that it is inappropriate to use discriminative classifiers to provide estimates for $p(c_i|c_i \cup c_j)$ and that proper density models should be used instead; this

argument applies equally well to one-vs-all and other reductions. Furthermore, if any desired probability $p(c_i|L)$ could be easily and accurately estimated, then the one-vs-all reduction would be a simple and straightforward approach for predicting probabilities on the multiclass problem. However, we hypothesize that inducing pairwise probabilities should be easier than inducing one-against-all class probabilities since all-pairs subproblems tend to be simpler than one-vs-all subproblems (and, in turn, simpler than other ECOC encodings). Therefore probabilistic pairwise classification should have a higher probability of high accuracy. Evidence of the simplicity of all-pairs subproblems compared to one-vs-all subproblems is given in Fürnkranz [40].

5.2.4 Relationship to Previous Methods

Wu et al. show that Hastie and Tibshirani's pairwise coupling method can be derived as an approximation to the identity

$$p_i = \sum_{j:j\neq i} (\frac{p_i + p_j}{k-1})\mu_{ij} \tag{5.8}$$

with $\mu_{ij} = p(y = c_i|y \in \{c_i, c_j\}, \boldsymbol{x})$ where $p_i + p_j$ is assumed to be $2/k$ and μ_{ij} are taken as the pairwise predictions $\hat{\mu}_{ij}$ [103]. PPC is also based on this identity, but instead of assuming that $p_i + p_j = 2/k$, PPC trains a discriminative classifier to explicitly estimate $p_i + p_j = \hat{p}(c_i \cup c_j|L, \boldsymbol{x})$.

5.3 Methodology

In this section, we describe experimental results comparing probabilistic pairwise classification to other pairwise classification methods, and to classifiers that are capable of making multiclass predictions directly, including decision trees, nearest-neighbor algorithms and random forests. The experimental results indicate that under a wide variety of conditions, probabilistic pairwise classification either outperforms or ties other methods.

5.3.1 Data Sets

Experiments are performed over 20 publicly available datasets. The datasets, number of classes, number of features and number of instances are indicated in Table 5.2. We formalize our data set selection decision procedure below to rule out bias in data set selection. We downloaded collections of preprocessed datasets in Weka's ARFF format from `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`, omitting any collection that consisted solely of artificial, ordinal or regression datasets, and filtered them, requiring that (a) the number of classes must be three or more, so that it is a multiclass classification problem instead of a binary classification problem, (b) the number of numeric attributes is between 5 and 500 ensuring there are sufficiently many but not too many attributes and (c) the number of instances is 200 or more, so that each class will have sufficiently many instances in each binary training set. Uniquely identifying attributes that completely specify the identity of an instance (such as ID or index attributes) are discarded, specifically: the *counter* attribute in *collins*, the *BookID* attribute in *authorship* and the *ID* attribute in *dj30-1985-2003*. Classes with less than 20 instances are deleted, along with corresponding instances. The free parameters in the above rules were hand-tuned until 20 datasets were selected to facilitate statistical analysis. After deletion of classes, any duplicate instances (based on attribute values, not class values) are deleted. Data set selection was performed before evaluation of algorithms in order to avoid bias.

Stratified subsampling is used to reduce the total number of instances for large problems in order to reduce computational demands, while maintaining a distribution over class labels commensurate with the original sample. For data sets with $N \geq 450$ instances, random draws are sampled with $N_{tr} = 300$ training points and $N_{ts} = 150$ test points. For data sets with $N < 450$, random draws are taken with 2/3 of the instances used for training and the remaining approximate 1/3 points for testing. Missing values

are filled in with the mean of non-missing values for each attribute. Datasets from similar domains are discarded in order to improve tests for statistical significance of results, as prescribed by Demšar [24]. In particular, *pendigits* was discarded because of its similarity to *optdigits*, only one of the *mfeat-* series was selected, and *anneal.ORIG*, *heart-h* and *cars-with-names* were filtered out due to similarity with other data sets. For data sets with $k > 20$ (*letter* and *dj30-1985-2003*), 1/3 of the classes are removed to decrease computational demands, and further stratified subsampling removes 1/3 of the instances.

Table 5.2 indicates the datasets used in our experiments, and their relevant properties. The column labeled *entropy* refers to the normalized entropy (in base 2) of the class distribution, $e = \frac{-1}{log_2 k} \sum_{i=1}^{k} p_i log_2(p_i)$, where p_i is the proportion of instances with the class label c_i and k is the number of classes. For instance, the entropy is 1 for a class with an even distribution of class labels $p_1 = ... = p_k = 1/k$ and 0 for a distribution that has only instances with one label, i.e. $p_i = 1, p_{j \neq i} = 0$. To summarize, the number of classes varies from 3 to 20, with entropy varying between 0.4819 and 0.9976. The smallest training sample size (after subsampling) is 133, and the number of attributes ranges from 6 to 254.

5.3.2 Multiclass and Pairwise Classification Methods

We compare probabilistic pairwise classification to four related methods:

5.3.2.1 Multiclass Classifiers

For some of our experiments, we use a base classifier that is itself capable of performing multiclass classification. For example, we use decision trees, random forests and k-nearest neighbor algorithms.

Table 5.2: Properties of the 20 data sets used in our experimental studies.

dataset	classes	entropy	numeric	nominal	train	test	sampled-from
anneal	4	0.6282	6	23	300	150	878
arrhythmia	5	0.7311	198	56	257	129	386
authorship	4	0.9363	70	0	300	150	841
autos	5	0.9328	15	10	134	68	202
cars	3	0.8693	6	1	270	136	406
collins	11	0.9543	19	0	300	150	451
dj30-1985-2003	20	0.9936	6	0	133	67	138123
ecoli	4	0.9008	6	0	204	103	307
eucalyptus	5	0.9725	14	5	300	150	736
halloffame	3	0.5010	15	1	300	150	1340
hypothyroid	3	0.4819	6	20	300	150	3707
letter	18	0.9920	16	0	136	68	18668
mfeat-morphological	10	0.9911	6	0	300	150	1888
optdigits	10	0.9971	58	0	300	150	5620
page-blocks	5	0.5945	10	0	300	150	5393
segment	7	0.9927	18	0	300	150	2086
synthetic-control	6	0.9976	60	0	300	150	600
vehicle	4	0.9923	18	0	300	150	846
vowel	11	0.9910	10	3	300	150	990
waveform	3	0.9942	40	0	300	150	5000

5.3.2.2 Voted Pairwise Classification (VPC)

This is the method proposed by Friedman in which each base classifier makes a discrete vote for the prediction and the classifier makes a discrete classification rather than a posterior probability prediction [38].

5.3.2.3 Hastie-Tibshirani (HT)

This is the pairwise coupling scheme proposed in [51] and implemented in Weka's class MultiClassClassifier in Weka version 3.7.0. This method is described in Section 5.1.1.2.

5.3.2.4 Wu-Lin-Weng (WLW)

WLW is the pairwise coupling scheme proposed in [103] and implemented in LibSVM version 2.9. We perform the same preprocessing as implemented in LibSVM's

method *svm_predict_probability*, namely clamping probability predictions to be between 10^{-7} and $1 - 10^{-7}$, and calling the method *multiclass_probability*, which is described in Section 5.1.1.2. While it would be possible to use the entire LibSVM implementation to obtain predictions for WLW-SVM-121, we merely use the *multiclass_probability* method in order to reduce the number of differences between the different multiclass and pairwise methods, ensuring that (a) there are no other pre- or post-processing steps included in one framework but not the other and (b) the pairwise classifiers are identical between methods. In contrast, we use the Weka framework for HT because it is compatible with any base classifier, whereas WLW full implementation in LibSVM is only compatible with SVM-121. Preliminary studies showed a statistically significant difference between the full LibSVM implementation of WLW-SVM-121 and the implementation in our framework, but we used our framework implementation in order to make a more straightforward comparison between the methods.

5.3.3 Base Classifiers

We experiment with a variety of base classifiers, which are the classifiers used with the various reduction schemes or used directly as multiclass classifiers. The first three algorithms provide direct support for multiclass classification, so we are able to compare multiclass classification algorithms to the pairwise coupling methods, complementing results given in Fürnkranz [39] and Wu et al. [103]. We use the following classifiers:

5.3.3.1 Decision Tree (J48)

We use the decision tree classifier as implemented in Weka's J48 (in Weka 3.7.0), which is a reimplementation of Quinlan's C4.5 decision trees [83]. We use Weka's default parameter settings: a pruned decision tree that allows multiway splits on nominal attributes.

5.3.3.2 K-Nearest Neighbor (KNN)

We use the K-nearest neighbor algorithm (see [1]) as implemented in Weka's IBk class, and in our experiments, we fix $k = 1$ and use no distance weighting, so that Euclidean distance is used for numerical attributes and Hamming distance is used for nominal attributes.

5.3.3.3 Random Forests (RF-100)

We use the random forest algorithm as implemented in Weka's RandomForest class. Following Breiman [15], we set the number of trees in each forest to be $L = 100$ and set the number of features in the random inputs scheme to be $(log_2(d) + 1)$, where d is the number of attributes. The size of the trees is unconstrained. In Section 5.5.2, we investigate the performance as a function of the number of trees.

5.3.3.4 Support Vector Machines (SVM-121)

We use the SVM algorithm as implemented in LibSVM [103], using the Gaussian kernel. To search over the hyperparameters $\{c, \gamma\}$, we perform a search over the coarse 7×7 grid of $\{-5, -1.666, 1.666, 5.0, 8.333, 11.666, 15.0\}^2$ to first determine a value for the cost hyperparameter c. Then a separate search is performed over a finer grid of 72 samples (ranging from -40 to 15) at the previously determined c-value to obtain the value for γ. We used this sampling scheme since there was much more sensitivity to the γ hyperparameter than to the c hyperparameter, and so that our scheme would take a total of 121 samples, as done in many other grid searches, such as LibSVM [103]. Platt scaling is used to fit a sigmoid to each of the SVM models to improve probability estimates [82]. Additionally, since we are using LibSVM with probability estimates enabled, a pairwise implementation of WLW is used to improve the pairwise predictions before determining the full multiclass probability distributions by using the methods identified above. This implementation therefore has a double-layer of the WLW

method, once it estimate the binary probabilities and again to estimate the multiclass probabilities.

5.4 Results

The experiments are performed over the following combinations of components: 4 base classifiers × 2 metrics × 5 multiclass/pairwise coupling methods × 20 data sets. For a given pairwise classification method METHOD and base classifier BASE, we describe the combination as METHOD-BASE. For example, probabilistic pairwise classification with the J48 decision tree base classifier is named PPC-J48. When a method is used directly as a multiclass classifier, we use the prefix MULTI, such as MULTI-J48. To improve readability, aggregate behavior over all data sets is reported here in the main text, while performance on individual data sets is discussed in the Appendix (Section 5.7). We first report on results for the accuracy metric (Section 5.4.1), then discuss performance under probability predictions (Section 5.4.2).

In order to identify statistically significant differences between algorithms, we use the nonparametric statistical tests recommended by Demšar [24] and refined by García et al. [42]. In particular, we use the Holm test for comparing one algorithm against many others; this is a nonparametric algorithm that controls for the family-wise error rate, and yields adjusted p-values. To perform these computations, we use the software provided by García et al. [42].

5.4.1 Accuracy

The accuracy metric counts the number of correctly classified instances, ignoring probabilistic predictions: $a = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}(\boldsymbol{x}) = y(\boldsymbol{x}))$, where $\hat{y}(\boldsymbol{x}) = \text{argmax}_i\, p_i(\boldsymbol{x})$ is the predicted class, $\boldsymbol{x}$ is the input attribute vector, and $y(\boldsymbol{x})$ is the true class. Tables and figures indicating performance on individual data sets are located in the Appendix.

Figure 5.2 indicates the behavior of each base classifier and reduction technique

on the accuracy metric, averaged over all data sets. The first category (multiclass) indicates using the specified classifier directly as a multiclass classifier rather than using a pairwise coupling scheme. For the base classifiers J48, RF-100 and SVM-121, PPC yields the highest average accuracy.

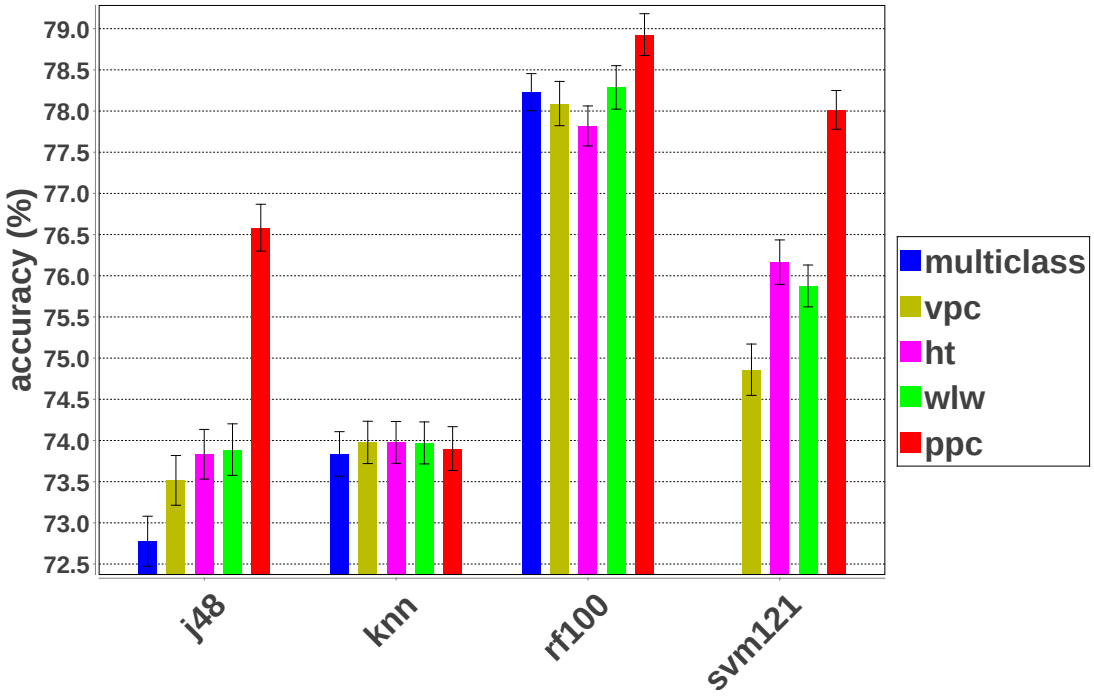


Figure 5.2: Accuracy averaged over all 20 data sets for all combinations of base classifier and reduction method, with one standard error indicated.

5.4.2 Predicting Probabilities

In this section, we test the ability of probabilistic pairwise classification to predict well-calibrated probabilities. In order to evaluate the calibration of each method, we use the Brier metric [16], also known as the mean squared error (MSE) metric, in which $b(\boldsymbol{x}) = \frac{1}{d}\sum_j (t_j(\boldsymbol{x}) - \hat{p}_j(\boldsymbol{x}))^2$, where $\boldsymbol{x}$ is the input vector, t_j is the target probability for the the jth class, $\hat{p}_j$ is the probability estimated by the classification method and d is the dimensionality of the input vector. We take $t_j(\boldsymbol{x})$ to be 1 if the label belongs to the jth class and 0 otherwise. This metric is used in other related work such as Wu et al.

[103] and Zadrozny [106]. In the results presented below, we use the rectified Brier score $r(\boldsymbol{x}) = (1 - b(\boldsymbol{x})) \times 100$ so that the results are structurally similar to percent accuracy; namely that higher is better and that the values range between 0% and 100%. Again, tables and figures indicating performance on each data set are located in the Appendix.

Figure 5.3 shows the average performance of each base classifier and reduction technique under the Brier metric, with a standard deviation for the PPC methods also indicated. The first category *multiclass* indicates using the specified classifier directly as a multiclass classifier. Again PPC has the highest average performance for the base classifiers J48, RF-100 and SVM-121.

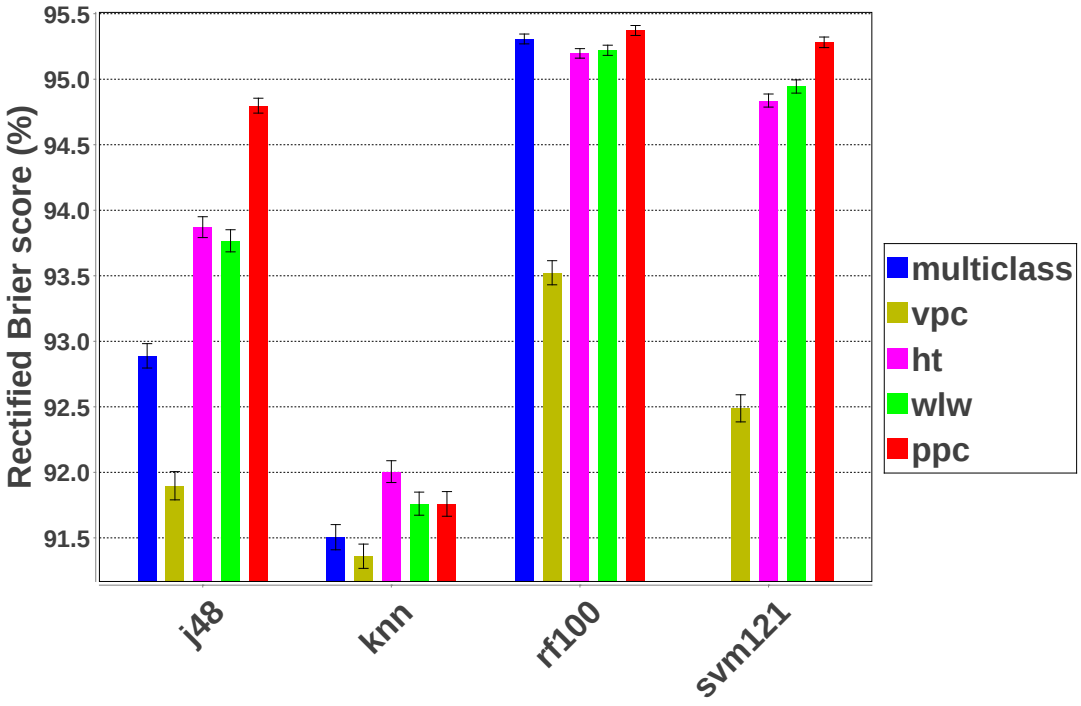


Figure 5.3: Rectified Brier score averaged over all 20 data sets for all combinations of base classifier and reduction method, with one standard error indicated.

5.4.3 Discussion

Figure 5.4 plots the average rank of each algorithm over all 20 data sets. Depiction of average rank (as opposed to average accuracy or average rectified Brier score) is

valuable since (a) it doesn't assume that errors across different data sets are commensurate and (b) it doesn't allow excellent performance on a single problem to obscure mediocre performance on several other data sets. This visualization technique and related statistical tests are described in Demšar [24]. A vertical bar connects the top algorithm to any other algorithm that is not statistically significantly different from it under the Holm test [42] at a $p \leq 0.05$ level of significance. Note that in all of these instances PPC is either the highest ranking method (6 out of the 8 runs), or not statistically significantly different from the top ranking method. Therefore, we conclude that for a variety of base classifiers, metrics and data sets, that PPC yields statistically significantly better performance than other pairwise coupling methods and multiclass classification using the base classifier.

We noted that the predictions and performance were not preserved under permutation of the subproblem assignments. Specifically, while only $k(k-1)/2$ pairwise and $k(k-1)/2$ pair-vs-rest terms need to be computed, we found that modeling $\hat{\mu}_{ij}$ and computing $\hat{\mu}_{ji} = 1 - \hat{\mu}_{ij}$ for $i = 1..k$ and $j = 1..i-1$ provided slightly different results than modeling $\hat{\mu}_{ji}$ and computing $\hat{\mu}_{ij} = 1 - \hat{\mu}_{ji}$. This suggests an asymmetry in the behavior of the base classification algorithms. Future work could investigate whether it is valuable to estimate and utilize both $\hat{\mu}_{ij}$ and $\hat{\mu}_{ji}$ independently in order to achieve a smoother, more accurate response.

5.5 Additional Results

In order to understand the behavior of probabilistic pairwise classification, we analyze the learning curves (Section 5.5.1), look at the accuracy as a function of more accurate base classifiers (Section 5.5.2), perform a series of supplemental experiments on synthetic data sets (Section 5.5.3), study performance as a function of number of classes (Section 5.5.4), identify dependence on data set entropy (Section 5.5.5) and test performance degradation under related simplified models (Section 5.5.6).

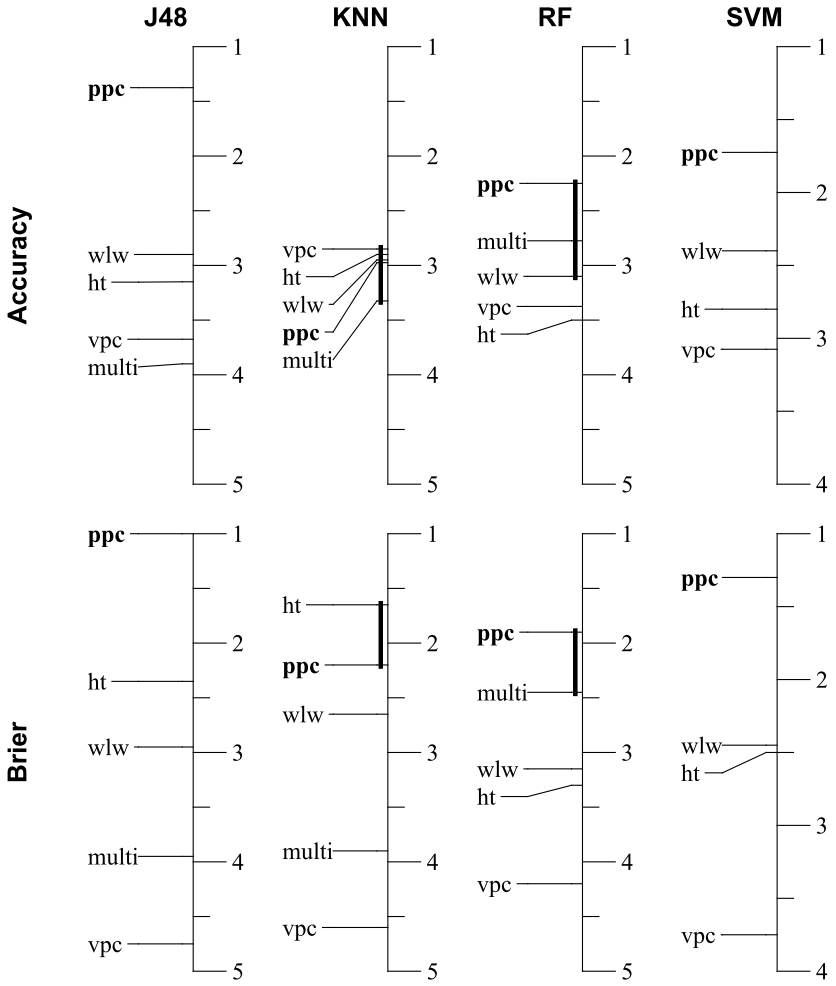


Figure 5.4: Graphical depiction of the rank of each algorithm as averaged over all 20 data sets, shown for the accuracy metric (top row) and for the Brier metric (bottom row). A vertical bar connects the top algorithm to any other algorithms(s) that are not statistically significantly different from it, if any.

5.5.1 Learning Curves

In this section, we investigate the predictive accuracy as a function of the amount of training data—the so-called learning curves. Since the statistical comparisons performed in Section 5.4 use a single value for the amount of training data, it is valuable to look at the dependence on the amount of training data in order to determine, for example, whether the relative effectiveness of each method changes with varying training sample sizes. For this experiment, we take the 10 largest data sets from the 20 data sets

used in Section 5.4, again with a 2/3 training and 1/3 training proportion. We focus on using the random forest classifier with 100 trees (RF-100), since it was demonstrated to be competitive compared to the other base classifiers, and less computationally expensive than the model-selected SVM-121. The average results over all 10 data sets are indicated in Figure 5.5; behavior on individual data sets is not qualitatively different than the composite behavior, so plots for individual data sets are omitted.

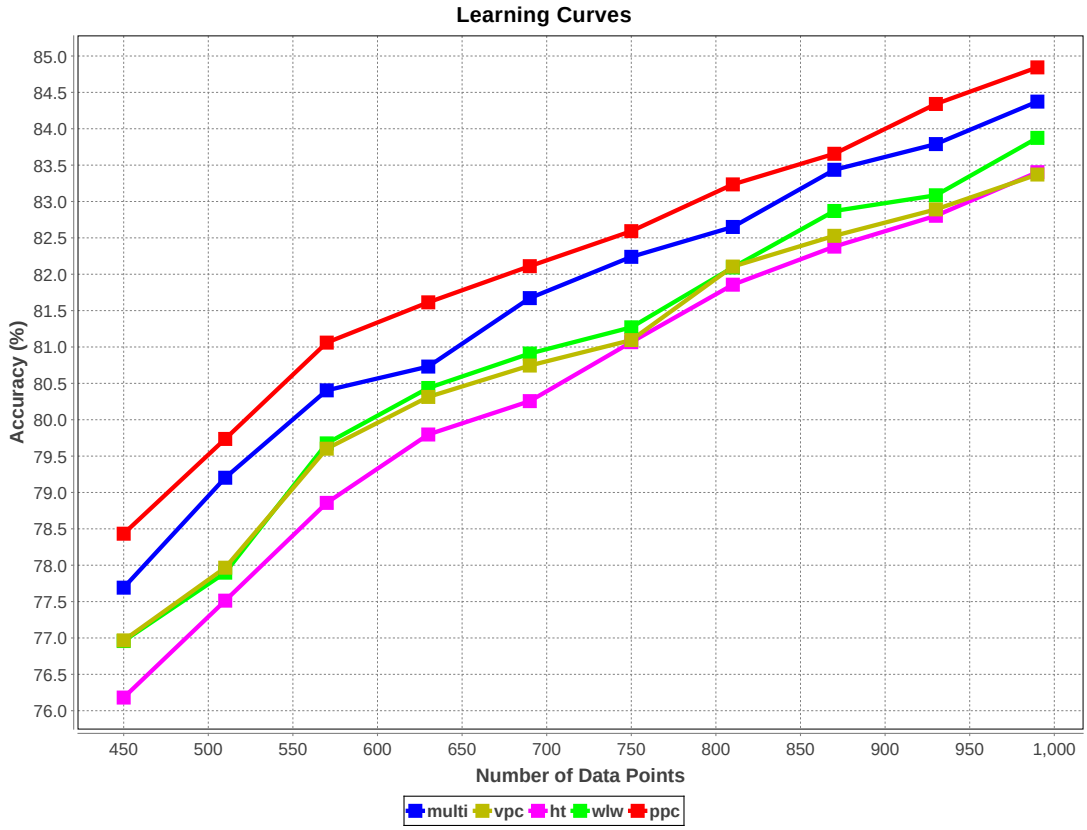


Figure 5.5: Accuracy as a function of the sample size (2/3 of which used for training), averaged over the 10 largest data sets described in Table 5.2

Note that the relative performance of all 5 methods does not vary significantly as the amount of training data is doubled; only voted pairwise classification (VPC) shows an average change in rank, decreasing from the accuracy of WLW at low sample sizes to the accuracy of HT at larger samples sizes. The fact that the ranks are predominantly stable indicates that statistical comparisons in Section 5.4 apply to a larger ranger of

training sample sizes. Also note that the amount of training data is crucial in determining the performance of the learning algorithm—for instance, the worst method with 570 data points (380 training points) outperforms the best method with 450 data points (300 training points). Informally, the addition of 20% more data points is more valuable than switching from the least to the most effective combination rule.

5.5.2 Varying the Base Classifier Accuracy

Many studies assume that the benefits of a reduction scheme with a poor or untuned base classification algorithm will generalize to more accurate base classifiers [27, 64, 2]. As pointed out by Rifkin and Klautau [85], it is essential to study the reduction with well-tuned base classifiers, since we are interested in understanding the behavior in the regime with the best predictive power. In this section, we investigate the accuracy of the various methods as the performance of the base classifiers is varied, by using random forests as the base classifier and increasing the number of trees. The accuracy of random forest classifiers tends to increase monotonically with the number of trees [15]. We use the following numbers of trees: $\{10, 50, 100, 200, 500, 1000\}$. Figure 5.6 depicts the average accuracy of each of the 5 methods as a function of the base-10 log of the number of classes.

While the average over 20 data sets indicates the advantage of PPC over the other methods for varying numbers of trees, the variability between the methods decreases as the number of trees increases. At 10 trees, the difference between the best and worst performing methods is about 2.25%, while at 1000 trees, the difference between the best and worst performing methods is only about 1.25%. While this average behavior is indicated in Figure 5.6, the behavior on several individual data sets is indicated in Figure 5.5.2, and varies dramatically across data sets.

For instance, in the *anneal* dataset, the accuracy of all methods basically level out around 200 trees, without too much noise. In the *dj30-1985-2003* data set, the

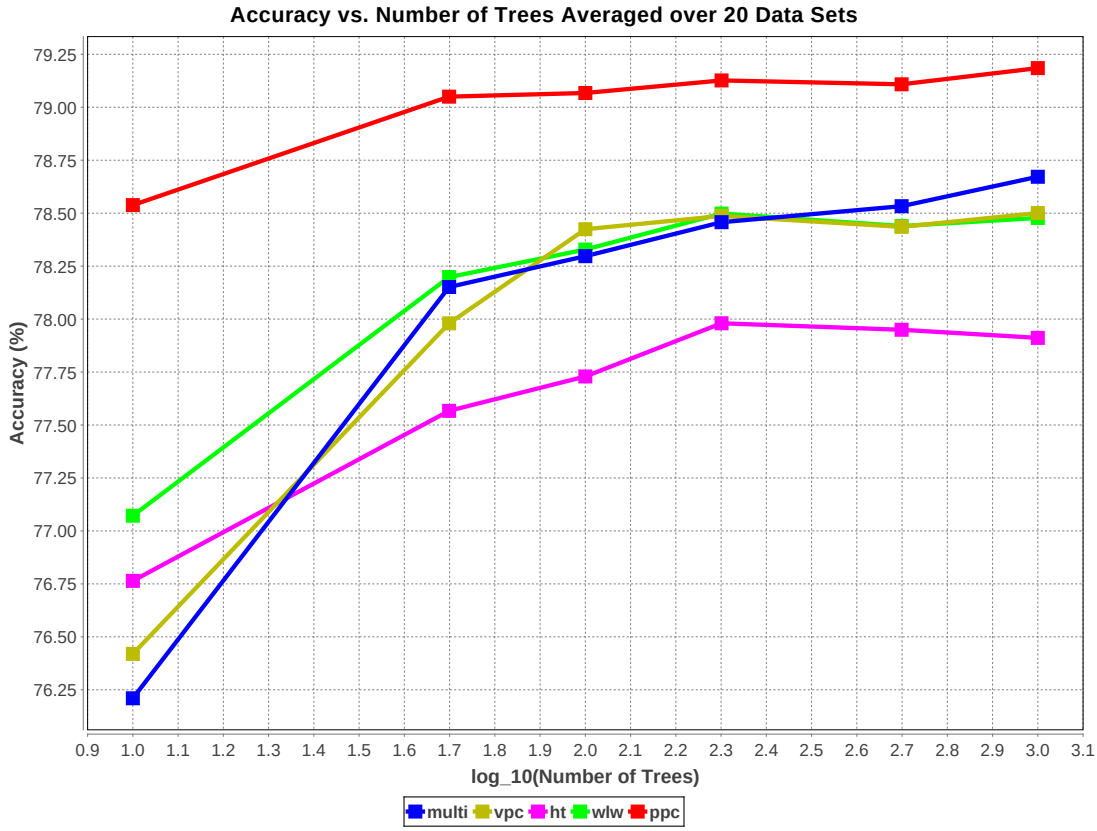
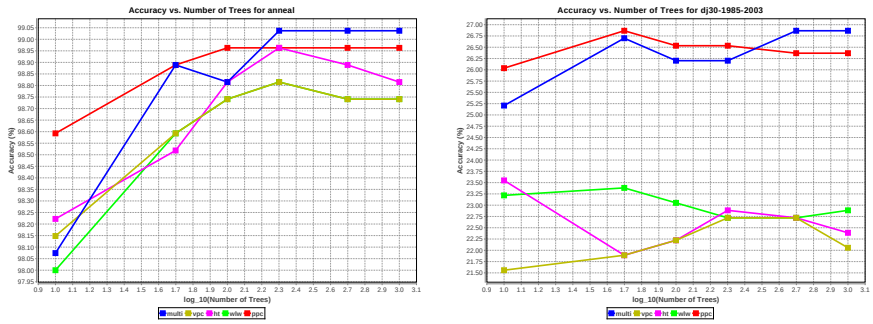


Figure 5.6: Accuracy as a function of the (log_{10} of the) number of trees in the random forest base classifier, averaged over all 20 data sets described in Table 5.2

methods PPC and MULTI have a similar performance, significantly higher than the other methods. In the *eucalyptus* data set, the multiclass classifier doesn't improve performance as the number of trees is increased. Broadly speaking, increasing the number of trees is effective in all methods, and while the curves for VPC, HT and WLW tend to level out around 200 trees (with the performance of HT slightly decreasing), the multiclass method and PPC still attain increased accuracy with 500 or 1000 trees.

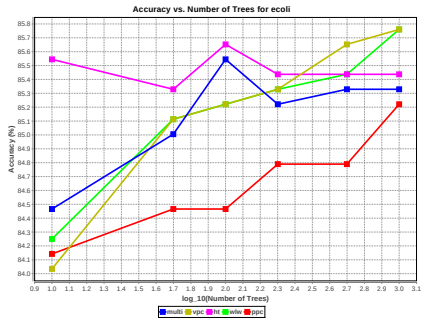
5.5.3 Synthetic Data Sets

One of the primary differences between the direct multiclass methods (such as decision trees and nearest-neighbor methods) and pairwise classification methods is that the direct multiclass methods operate on all classes simultaneously, while the pairwise

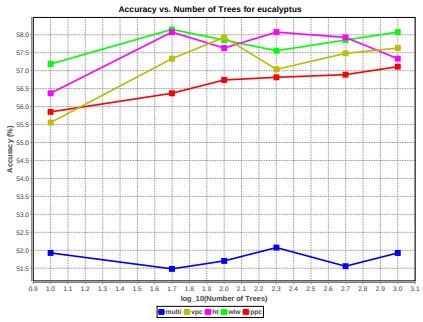


(a) anneal

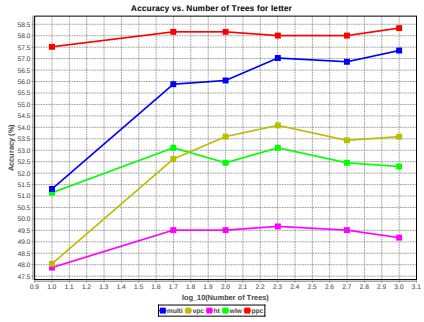
(b) dj30-1985-2003



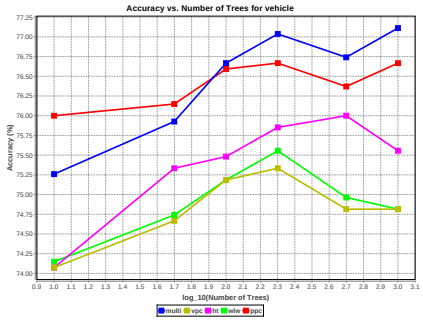
(c) ecoli



(d) eucalyptus



(e) letter



(f) vehicle

Figure 5.7: Examples of accuracy as a function of the number of trees in the random forest base classifier for 6 of the data sets.

classification methods are restricted to using one pair of classes at a time. In a PPC prediction (see Equation (5.6)), the first term $p(c_i|c_i \cup c_j, \boldsymbol{x})$ is restricted to using pairwise discriminations, and is multiplied by a pair-vs-rest weight $p(c_i \cup c_j|L, \boldsymbol{x})$. In this section, we hypothesize that PPC will be less effective than a direct multiclass method for a problem in which it is valuable to operate on all classes simultaneously. Specifically, we construct a 4-class dataset in which each class is generated by a Gaussian distribution centered in one of the four quadrants. By varying the covariance matrices

of the Gaussian distributions, we are able to change the amount of noise in each problem. The motivation for the structure of this synthetic data set is that the decision tree algorithm can see all four classes simultaneously, and therefore has the potential to use information about classes A and B to inform the $C - D$ decision boundary. Ideally, it would use exactly the same decision boundaries for $A - B$ as for $C - D$. Since the PPC algorithm never sees more than two classes at a time, PPC will not be able to obtain this same benefit. As for the data sets discussed in Section 5.3, 300 training points and 150 test points are used.

5.5.3.1 Noiseless Synthetic Data Set

Figure 5.8 indicates the 4-class synthetic problem described above, and the results of the MULTI-J48 classifier and PPC classifier using MULTI-J48 base classifiers is indicated in Table 5.4. The results for this experiment are averaged over 100 random draws from the underlying generative distribution; more runs are possible for this synthetic study since the decision-tree based algorithms are computationally inexpensive. Even though this is a simple learning problem, note that the the accuracies are not exactly 100%; this result is due to the fact that the convex hull of training points is responsible for inducing the decision boundary. Since 1/3 of the data is removed for testing purposes, the decision tree obtains suboptimal decision boundaries. For this data set, the decision tree outperforms probabilistic pairwise classification by 0.527%. The two-tailed paired t-test indicates a statistically significant difference at the $p \leq 0.05$ level; the actual p-value is $p \leq 3.15 \times 10^{-11}$. Therefore we have verified our hypothesis that a multiclass classification method can outperform PPC on a problem for which significant benefits can be obtained by operating on all subproblems simultaneously.

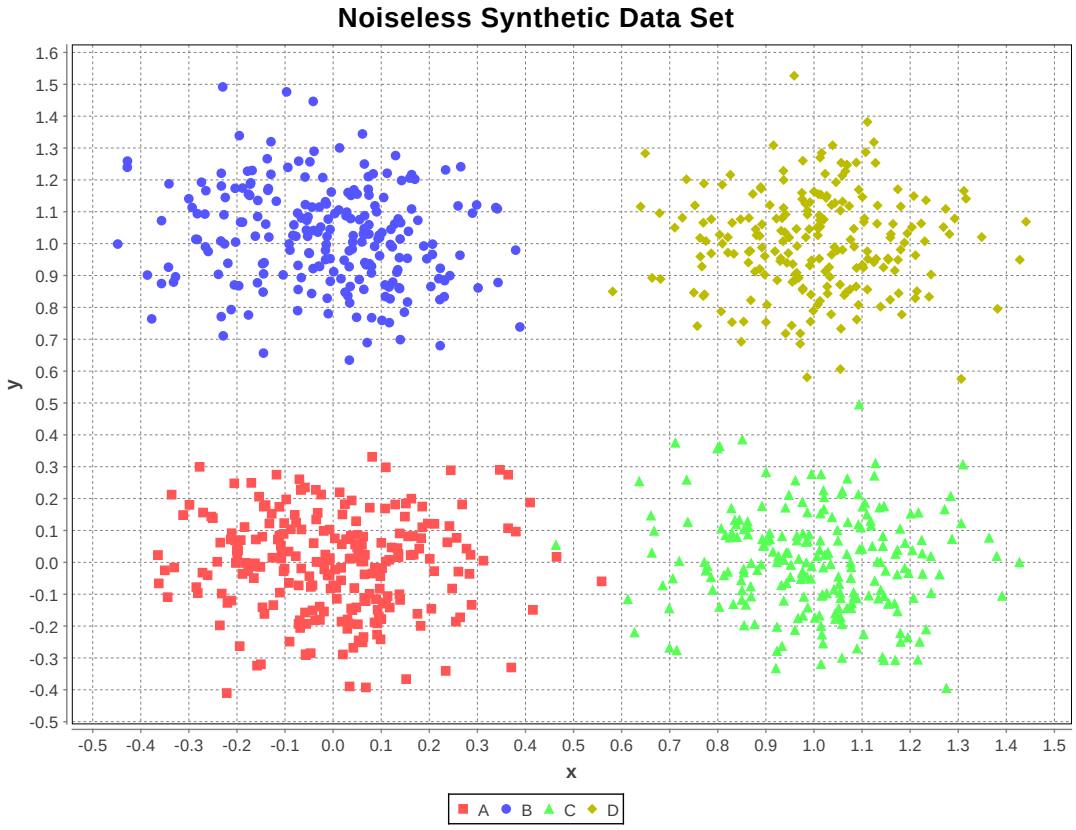


Figure 5.8: Noiseless 4-class synthetic data set

5.5.3.2 Noisy Synthetic Data Set

We also repeated the above experiment with an increased amount of noise to see whether the benefits of the multiclass decision tree will be retained. Figure 5.9 indicates the 4-class problem with a significant amount of noise. Again, the results for this experiment are averaged over 100 random draws from the underlying generative distribution. The results from using decision trees (MULTI-J48) and probabilistic pairwise classification with decision tree base classifiers (ppc-j48) are indicated in Table **??**. The increased noise is responsible for the significantly decreased accuracy for both algorithms, compared to the synthetic dataset presented in Section 5.5.3.1. In this case, PPC outperforms MULTI-J48 by 1.48%; that is, on this sample data set, it is slightly more accurate to reduce the dataset into $\binom{4}{2} = 6$ subproblems than to solve the 4-class

Table 5.3: Accuracy results (%) for the comparatively noiseless synthetic data set. The standard error over 100 random samplings is indicated in parentheses.

multi-j48	ppc-j48
99.2 (0.08)	98.7 (0.10)

problem using directly using a multiclass decision tree. The two-tailed paired t-test indicates a statistically significant difference at the $p \leq 0.05$ level; the actual p-value is $p \leq 2.14 \times 10^{-9}$. This result is contradictory to the result in Section 5.5.3.1, even though the structure of the decision boundaries remains unchanged. It is not entirely clear why multiclass decision trees do not maintain the same advantage in the noisy synthetic problem as in the noiseless synthetic problem. One possible explanation for the advantage of PPC is that the estimation and average of 6 subproblems performs a valuable smoothing that is lacking in the direct multiclass method.

5.5.3.3 Summary

In this section, we hypothesized that a multiclass problem in which many subproblems share decision boundaries would favor operating on all classes simultaneously (and thus a direct multiclass method) rather than limiting decision regions to pairwise classifications. In Section 5.5.3.1, we constructed a synthetic data set that validated this hypothesis. However, in a synthetic data set with identical decision boundaries and only increased noise, PPC is surprisingly more effective. We suggested that PPC may have an advantage in the noisy situation due to its averaging over a large number

Table 5.4: Accuracy results (%) for the noisy synthetic data set. The standard error over 100 random samplings is indicated in parentheses.

multi-j48	ppc-j48
84.5 (0.34)	**86.0** (0.31)

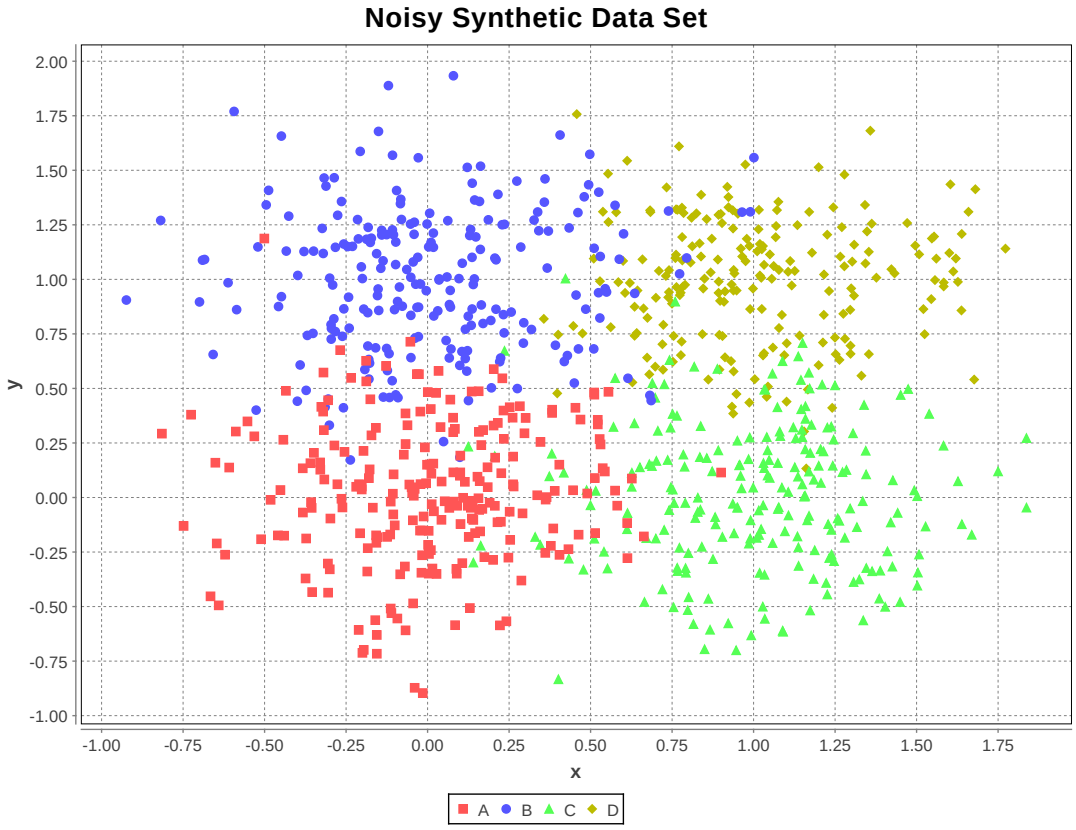


Figure 5.9: Noisy 4-class data set

of estimates. Further experiments would be necessary in order to validate this new hypothesis.

5.5.4 Performance vs. Number of Classes

For the J48 base classifier and accuracy metric, PPC exhibits excellent performance gains for the data sets *letter*, *optdigits*, *collins* and *vowel*, which have 18, 10, 11, and 11 classes, respectively. Since this larger performance gain occurs on four of the six data sets with the largest number of classes, it suggests that PPC may have a bigger advantage on data sets with more classes. First, we evaluate this hypothesis under the 20 benchmark data sets described in Section 5.3.1, then evaluate this hypothesis on new data sets in which we incrementally increase the number of classes. In this section, we

restrict our focus to using random forest as the base classifier, since it had competitive performance on the benchmark data sets, and since it is computationally less expensive than the model-selected support vector machine algorithm. We begin by quantifying the relative gain over a multiclass algorithm as a function of the number of classes for the benchmark data sets from Section 5.4.

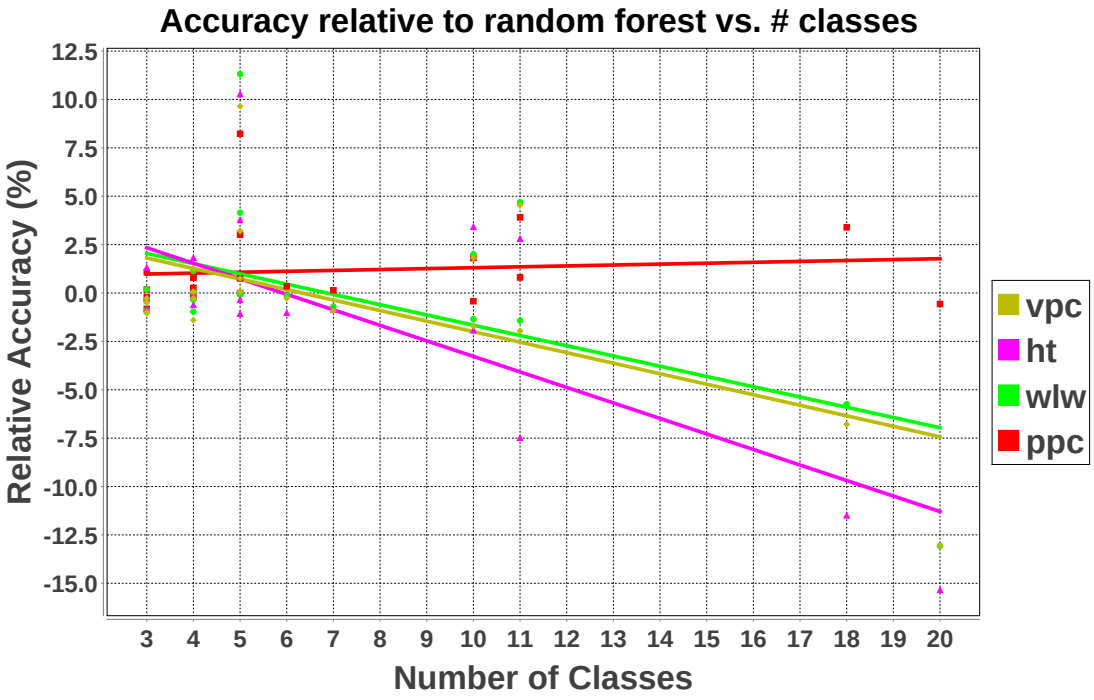


Figure 5.10: The accuracy relative to a random forest with 100 trees as a function of the number of classes in the data set for voted pairwise classification (VPC), Hastie-Tibshirani's method (HT), Wu-Lin-Weng's method (WLW), and probabilistic pairwise classification (PPC). There is one data point for each of the 20 benchmark data sets (see Section 5.3.1) and for each of the methods.

Figure 5.10 indicates the performance relative to random forest as a function of the number of classes. For instance, a relative accuracy of 0% indicates (at that number of classes) that random forests and the pairwise classification method have identical performance. A positive slope indicates that the algorithm improves its benefit over random forests as the number of classes increases; a negative slope indicates that random forests is more effective at a higher number of classes. For voted pairwise classification

(VPC), Hastie-Tibshirani's method (HT) and Wu-Lin-Weng's method (WLW), there is a negative correlation between the number of classes and the relative accuracy (with multiclass random forests as the baseline). The results comparing the multiclass classifier to VPC, HT and WLW are consistent with the observation in Wu et al. [103] that multiclass classification is more effective than the pairwise classification schemes as the number of classes increases. In contrast, for probabilistic pairwise classification (PPC), there is an average gain of about 1% over the range of 17 classes, indicating that it is approximately as effective as the multiclass classifier as the number of classes is increased. Qualitatively similar results are obtained for the Brier metric (omitted for brevity).

Since the benchmark data sets were used to construct the hypothesis, they cannot be used to validate the hypothesis; instead, we experiment with a set of 9 new multiclass classification problems. These problems are formed using regression data sets whose numeric target attribute is discretized into different classes. For instance, a regression problem with outputs varying uniformly between 0 and 1 is transformed into a 3-class classification problem by taking class 1 to be instances with output between 0 and $\frac{1}{3}$, and so on. This technique used for conversion of regression into classification problems was proposed in Frank and Hall [34] and used in Fürnkranz [41]. We used this method due to its prominence in the related literature and because it provides a straightforward way to gradually tune the number of classes in the corresponding multiclass classification problems.

To obtain the data sets for this study, we started with the collection of 37 regression data sets available from the Weka website (obtained from various sources) and filtered out data sets that had less than 300 instances. The remaining data sets are indicated in Table 5.5. As in Section 5.3, we use 2/3 of the points for training and 1/3 of the points for testing. We stop increasing the number of classes when the number of instances per class drops below 10 (for training and testing).

Table 5.5: Regression data sets converted to classification problems with varying numbers of classes.

dataset	numeric	nominal	sampled-from
autoMpg	4	3	398
cholesterol	6	7	303
cleveland	6	7	303
housing	12	1	506
meta	19	2	528
pbc	10	8	418
quake	3	0	2178
sensory	0	11	576
strike	5	1	625

Results are indicated in Figure 5.11. The vertical axis indicates the accuracy relative to the random forest algorithm; a value of 0% indicates that the pairwise classification algorithm had equivalent performance to the random forest classifier. A positive slope indicates increased advantage over the random forest at a higher number of classes. On the *meta* dataset, there is a negative correlation between number of classes and relative accuracy. In the *housing* data set, the relative accuracy is independent of the number of classes. For the other 6 data sets, there are varying degrees of improvement. The *cholesterol* data set attains the largest benefit; in this case, PPC increases relative improvement over the random forest algorithm by about 9.5% as the number of classes is raised from 2 to 13. While there are exceptions, the results on discretized data sets support the hypothesis that PPC confers more benefits at a higher number of classes. One possible explanation for this behavior could be a bagging phenomenon, as we discuss in Section 5.6.1.2.

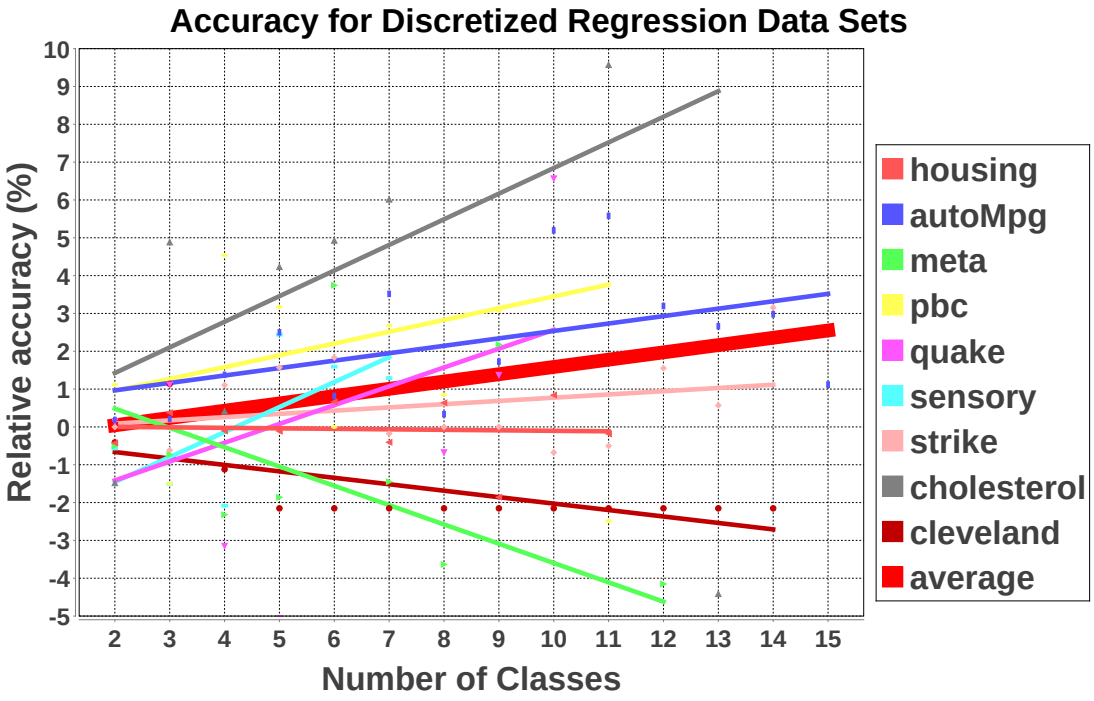


Figure 5.11: The accuracy relative to a random forest with 100 trees as a function of the number of classes for regression data sets that have been discretized with varying number of classes. The average over the 9 data sets is indicated by the wide red line series.

5.5.5 Performance vs. Entropy

In this section, we investigate the related issue of performance as a function of the normalized entropy of the class distribution; note that the probabilities of class membership (the terms p_i) are not equivalent to class priors which determine the normalized entropy. Recall that normalized class entropy (defined in Section 5.3.1) varies from 0 to 1, with a value of 0 meaning that all instances share the same class and a value of 1 meaning that the class distribution is uniform. In order to test the behavior of PPC and other algorithms under varying class distributions, we plot the accuracy as a function of normalized class entropy. The results on the accuracy metric are indicated in Figure 5.12, with each combination method compared to the direct multiclass method RF-100.

There are many data sets of varying difficulty that all have nearly unity normalized entropy, so the results are very noisy. Over the domain of normalized entropies from 0.5 to 1.0, the largest difference is between PPC and HT, with an average difference of approximately 3% near a normalized entropy of unity for the linear fits. These small and noisy differences do not indicate a significantly different performance for the various methods as a function of normalized entropy; further investigation would be necessary in order to isolate an effect of the class distributions, possibly using a discretization scheme such as the one used in Section 5.5.4. In contrast, note that in Hastie & Tibshirani's assumption that $p_i + p_j \approx 2/k$, the probability values p_i and p_j are true probabilities of class membership rather than priors over class distributions. Wu et al. show that Hastie & Tibshirani's method indeed performs more poorly as the probabilities of class membership are skewed [103].

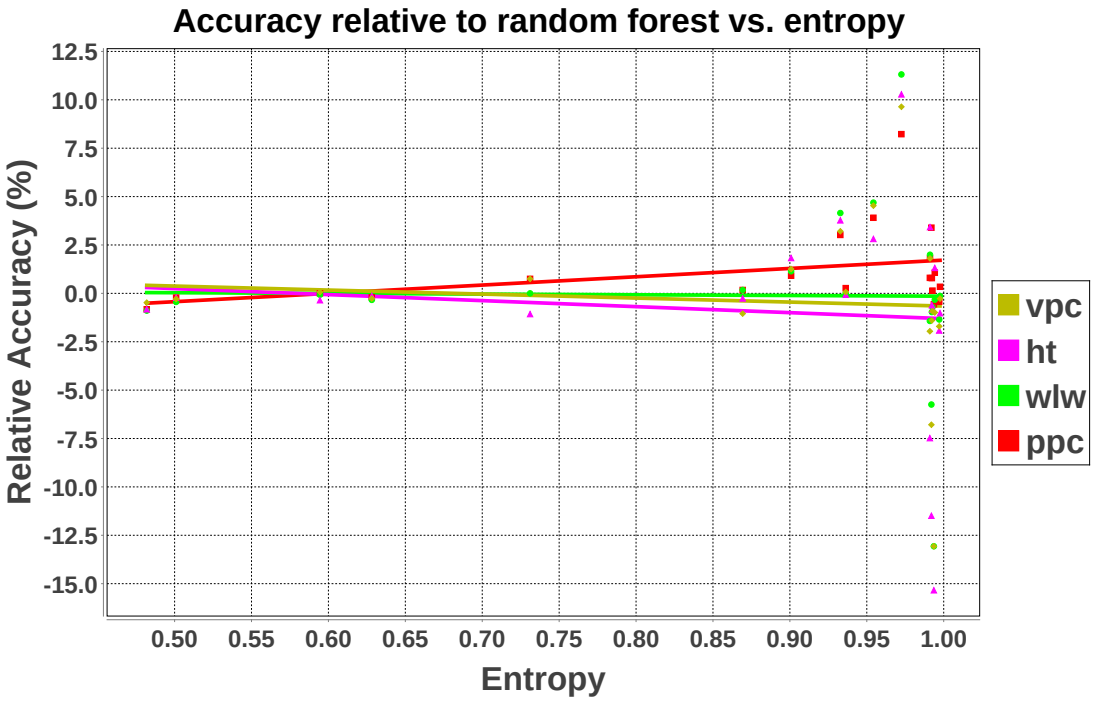


Figure 5.12: The entropy relative to a random forest with 100 trees as a function of the number of classes in the data set for voted pairwise classification (VPC), Hastie-Tibshirani's method (HT), Wu-Lin-Weng's method (WLW), and probabilistic pairwise classification (PPC)

5.5.6 Degrading Performance by Omitting Terms

Probabilistic pairwise classification makes each probabilistic prediction based on the product of two terms (see Equation (5.6)):

$$\hat{p}(c_i|L,\boldsymbol{x}) = \frac{1}{k-1}\sum_{j\neq i}\hat{p}(c_i|c_i\cup c_j,\boldsymbol{x})\hat{p}(c_i\cup c_j|L,\boldsymbol{x}) \tag{5.9}$$

By setting the terms to be the normalized constants $\hat{p}(c_i|c_i\cup c_j,\boldsymbol{x}) = \frac{2}{k(k-1)}$ and/or $\hat{p}(c_i\cup c_j|L,\boldsymbol{x}) = \frac{1}{k}$, we can better understand the relative importance of these terms in producing the overall multiclass probability estimates. Table 5.6 shows the average accuracy scores for the MULTI-J48 classifier, with statistically significant differences indicated in Table 5.7. We use the Holm test to identify statistically significant differences

between every pair of methods, and find that of the 6 scheme $\times$ scheme comparisons, the only pair that doesn't exhibit a statistically significantly different behavior under the accuracy metric is the comparison of the no-weight degradation to the no-pair reduction. Therefore, we conclude that both terms are equally important in making the pairwise classification. Note that there are special cases in which the degradations actually perform more accurately than the PPC itself, such as for the *anneal* dataset. Specifically, assuming a uniform distribution over the $\hat{p}(c_i|c_i \cup c_j, \boldsymbol{x})$ term or the $\hat{p}(c_i \cup c_j|L, \boldsymbol{x})$ term produces approximately 0.2% performance benefit for the *anneal* data set. Furthermore, removal of both the weight and pairwise terms still yields relatively large accuracy on some of the data sets (*cars*: 64%, *halloffame*: 85.3%, *hypothyroid*: 85.3%). These accuracies seem to be commensurate with the entropies of the data sets: (*cars*: 0.869, *halloffame*: 0.501, *hypothyroid*: 0.482).

Table 5.6: Accuracy scores for the J48 base classifier with various degradations, each cell is averaged over 10 random samplings.

dataset	ppc-j48	no-weight	no-pair	no-pair-and-no-weight
anneal	0.982	0.984	0.984	0.059333
arrhythmia	0.777519	0.727907	0.770543	0.063566
authorship	0.939333	0.916	0.852667	0.212667
autos	0.705882	0.691176	0.7	0.119118
cars	0.822059	0.808824	0.803676	0.641912
collins	0.404	0.367333	0.406	0.097333
dj30-1985-2003	0.231343	0.220896	0.222388	0.049254
ecoli	0.849515	0.853398	0.840777	0.120388
eucalyptus	0.600667	0.591333	0.586667	0.140667
halloffame	0.89	0.885333	0.874667	0.853333
hypothyroid	0.984667	0.984	0.98	0.852667
letter	0.561765	0.485294	0.542647	0.052941
mfeat-morphological	0.730667	0.716667	0.730667	0.072667
optdigits	0.899333	0.826	0.904	0.098
page-blocks	0.932667	0.915333	0.93	0.060667
segment	0.94	0.911333	0.94	0.139333
synthetic-control	0.937333	0.887333	0.931333	0.17
vehicle	0.725333	0.698667	0.674667	0.248
vowel	0.693333	0.616667	0.688667	0.094
waveform	0.709333	0.7	0.672667	0.346
average	0.765837	0.739375	0.751802	0.224592

Table 5.7: Adjusted p-values under the specified degradations for the accuracies indicated in Table 5.6.

hypothesis	p_{Holm}
ppc-j48 vs .ppc-j48-no-pair-and-no-weight	2.25×10^{-10}
ppc-j48-no-pair vs .ppc-j48-no-pair-and-no-weight	6.87×10^{-5}
ppc-j48-no-weight vs .ppc-j48-no-pair-and-no-weight	7.49×10^{-4}
ppc-j48 vs .ppc-j48-no-weight	0.012
ppc-j48 vs .ppc-j48-no-pair	0.04693
ppc-j48-no-weight vs .ppc-j48-no-pair	0.540291

5.6 Conclusion

In this chapter, we introduced a new multiclass classification algorithm called probabilistic pairwise classification (PPC). The derivation of the method is based on the Theorem of Total Probability, and the method reduces a k-class classification problem into $\frac{k(k-1)}{2}$ pairwise classification problems and $\frac{k(k-1)}{2}$ pair-vs-rest problems. Because PPC transforms multiclass problems into a set of binary problems, it can be combined with any binary or multiclass base classifier. Like related pairwise coupling methods [51, 103], PPC incorporates probabilistic predictions (rather than discrete votes) and produces a probability estimate (rather than a discrete classification). Our experimental results over 20 data sets, 4 base classifiers and 2 metrics show that PPC outranks related methods or is not statistically significantly different from the highest ranking method. There is some variability across methods; for instance, for the k-nearest neighbor base classifier and the accuracy metric, no method is statistically significantly different from the highest average ranking method, which is voted pairwise classification. Under the Brier metric, PPC ranked first on all 20 data sets. In order to understand the tradeoffs of PPC versus direct multiclass classification, we constructed synthetic data sets that were meant to favor a multiclass decision tree MULTI-J48 instead of PPC-J48. We showed that under some circumstances it is more valuable to perform a direct multiclass classification, but this result was sensitive to the amount of noise in the problem. By

discretizing regression data sets into classification problems with varying numbers of classes, we showed that the advantage of PPC over a multiclass random forest tends to increase with the number of classes, while voted pairwise classification and the methods from Hastie-Tibshirani and Wu-Lin-Weng exhibited poorer relative performance as the number of classes increased. Our results also indicate that the choice of base classifier has a large impact on the effectiveness of each method; for instance, k-nearest neighbor as a base classifier was not competitive with decision trees, random forests or support-vector machine base classifiers on either the accuracy or the Brier metric. Furthermore, we showed the value of increased strength of base classifiers and of increased amount of training data. PPC exhibits excellent performance an a variety of problems and metrics, but may be contraindicated in problems for which it is prohibitively computationally expensive to train $\frac{k(k-1)}{2}$ pair-vs-rest classifiers on the entire training sample.

5.6.1 Future Work

5.6.1.1 Faster Pair-Vs-Rest Classifiers

The bottleneck in training a PPC classifier is the training of the pair-vs-rest classifiers. Unlike the pairwise subproblems, the pair-vs-rest subproblems incorporate all data points (with the pair as the positive indicator class and the other classes as the negative indicator class). When using an expensive classification algorithm, such as support vector machines with a model selection scheme, this algorithm can take superlinearly longer to execute because of the increased number of training points. One possible way to alleviate this problem would be to use a more efficient (while less accurate) classification algorithm for the pair-vs-rest predictions, while still using a more expensive algorithm to handle the smaller pairwise problems. For instance, while using a model-selected SVM for the pairwise classifications, a decision tree might be used to make the pair-vs-rest classifications. There is no inherent reason that each

term needs to be produced by the same classification algorithm, and it is plausible that for some problems, different classification algorithms would be suitable for the pairwise discrimination than for the pair-vs-rest discrimination. Experimental studies with several different settings for pairwise and pair-vs-rest classifiers could be used to evaluate the performance of this mixed-base-classifier technique.

5.6.1.2 Relationship to Bagging

Bagging (short for bootstrap aggregating) is a classifier combination method that aims to improve predictive accuracy by producing many classifier based on resamplings (bootstrap samples) of the training set and averaging or voting their predictions [11]. Bagging reduces variance and prevents overfitting, but only provides an advantage for unstable classifiers. The fact that probabilistic pairwise classification exhibits a higher benefit for J48 (an unstable algorithm) than for KNN (a stable algorithm) suggests that the benefit is due to a bagging-like phenomenon. Bagging operates by resampling over data points in the training set by drawing bootstrap samples (samples with the same size as the original training set, with replacement.) In PPC, resamplings is done over classes rather than training points; however, this resampling may result in a similar benefit that models are able to complement each other to increase the multiclass classification accuracy. One argument against this explanation is that the relative benefit of PPC over the base multiclass classifier does not always show a positive correlation. If class-bagging were the explanation for the advantage of PPC, then the benefit should correspond to the number of pairwise classifiers, which varies as the square of the number of classes. Some preliminary results indicate that as the number of classes is increased, the relative benefit of PPC increases (see Section 5.5.4). Future work could identify whether class-bagging is responsible for the benefit in PPC, and could additionally investigate whether instance-bagging on subproblems combined with PPC can lead to additional performance improvements, augmenting studies such as Fürnkranz [40].

5.7 Appendix

This section reports the performance of each base classifier under the accuracy metric (see Section 5.7.1) and the Brier metric (see Section 5.7.2). Composite results (aggregated over data sets) are reported in Section 5.4. The following sections break down the results first by metric, then by base classifier, with results given for each multiclass classification method (multiclass classifier or pairwise classification method) and dataset. In each of the tables, the standard deviation is indicated in parentheses. The averages are computed over 10 random resamples, or 5 resamples for the SVM-121 methods.

5.7.1 Accuracy Metric

5.7.1.1 Decision Tree (J48)

Table 5.8 indicates the accuracy of the various pairwise classification methods while using the decision tree MULTI-J48 as the base classifier. Note that PPC has only four losses over the 20 data sets, which is statistically significant at $p \leq 0.05$ under the Holm test. Figure 5.13 shows the relative gain in accuracy over a multiclass J48 decision tree. Note that the PPC has a higher gain than VPC, HT and WLW for many data sets, and that for many data sets, PPC has a positive gain over MULTI-J48 while VPC, HT and WLW have a negative gain.

5.7.1.2 Nearest Neighbor (KNN)

Table 5.9 indicates the accuracy of the various pairwise classification methods while using the K-nearest neighbor algorithm as the base classifier. Figure 5.14 shows the relative gain in accuracy over a multiclass KNN classifier.

Table 5.8: Results for j48 under the accuracy metric

dataset	multi-j48	vpc-j48	ht-j48	wlw-j48	ppc-j48
anneal	98.20 (1.04)	**98.40** (0.84)	98.33 (0.85)	**98.40** (0.78)	98.20 (0.71)
arrhythmia	70.47 (4.01)	71.24 (3.36)	72.95 (2.65)	72.40 (2.11)	**77.75** (2.61)
authorship	91.93 (1.35)	91.40 (2.10)	91.40 (2.19)	91.00 (2.25)	**93.93** (1.62)
autos	**70.88** (7.78)	68.24 (6.66)	68.68 (6.97)	68.38 (6.33)	70.59 (6.69)
cars	80.37 (2.84)	80.51 (2.95)	80.88 (2.96)	80.81 (3.26)	**82.21** (2.98)
collins	35.00 (4.36)	36.80 (2.59)	37.20 (3.77)	37.20 (3.90)	**40.40** (4.61)
dj30-1985-2003	**25.37** (6.05)	20.60 (6.04)	21.94 (6.05)	21.04 (7.75)	23.13 (6.18)
ecoli	84.37 (2.61)	84.37 (4.45)	85.15 (3.69)	**85.24** (3.98)	84.95 (4.25)
eucalyptus	57.27 (4.12)	57.73 (4.90)	58.80 (4.72)	59.13 (4.73)	**60.07** (2.84)
halloffame	87.93 (2.71)	88.60 (3.68)	88.27 (4.08)	88.60 (3.60)	**89.00** (3.92)
hypothyroid	98.07 (1.76)	98.27 (1.55)	98.00 (1.51)	98.40 (1.58)	**98.47** (1.48)
letter	42.21 (7.17)	48.82 (7.65)	48.53 (7.27)	49.71 (8.74)	**56.18** (7.81)
mfeat-morphological	69.93 (4.55)	71.93 (3.89)	71.67 (3.75)	71.73 (3.99)	**73.07** (3.68)
optdigits	72.40 (3.46)	81.80 (3.08)	82.60 (3.56)	83.13 (3.34)	**89.93** (3.15)
page-blocks	92.20 (1.48)	91.67 (2.85)	90.93 (2.61)	91.87 (2.91)	**93.27** (1.79)
segment	92.47 (2.35)	90.93 (2.92)	91.20 (2.66)	91.60 (2.52)	**94.00** (2.15)
synthetic-control	86.53 (2.70)	87.67 (3.90)	88.27 (3.42)	88.07 (3.52)	**93.73** (2.54)
vehicle	70.87 (4.44)	70.53 (4.00)	70.13 (4.20)	69.93 (4.01)	**72.53** (3.22)
vowel	60.00 (5.25)	60.80 (2.51)	61.27 (3.42)	61.13 (3.27)	**69.33** (3.27)
waveform	69.07 (2.90)	70.00 (2.55)	70.47 (2.06)	70.00 (2.53)	**70.93** (2.74)
average	72.78	73.52	73.83	73.89	**76.58**
average rank	3.9	3.7	3.2	2.9	**1.4**

Table 5.9: Results for knn under the accuracy metric

dataset	multi-knn	vpc-knn	ht-knn	wlw-knn	ppc-knn
anneal	**96.33** (1.38)	**96.33** (1.38)	**96.33** (1.38)	**96.33** (1.38)	**96.33** (1.38)
arrhythmia	**60.16** (4.09)	59.61 (4.01)	59.61 (3.98)	59.61 (3.98)	**60.16** (4.09)
authorship	**99.20** (1.03)	99.13 (0.89)	99.13 (0.89)	99.13 (0.89)	**99.20** (1.03)
autos	**73.82** (4.54)	73.68 (4.62)	73.68 (4.62)	73.68 (4.62)	**73.82** (4.54)
cars	75.00 (3.10)	**75.44** (2.84)	**75.44** (2.84)	**75.44** (2.84)	75.29 (2.62)
collins	35.00 (3.45)	35.80 (2.79)	**36.07** (2.42)	35.87 (2.51)	35.00 (3.45)
dj30-1985-2003	28.51 (4.95)	**29.70** (5.05)	**29.70** (5.05)	**29.70** (5.05)	28.51 (4.95)
ecoli	**81.75** (3.23)	81.36 (3.39)	81.36 (3.39)	81.36 (3.39)	**81.75** (3.23)
eucalyptus	**49.33** (3.62)	49.00 (3.59)	49.00 (3.59)	49.00 (3.59)	**49.33** (3.62)
halloffame	87.13 (2.81)	87.33 (2.55)	87.33 (2.55)	87.33 (2.55)	**87.47** (2.66)
hypothyroid	83.80 (3.76)	**85.07** (3.33)	84.93 (3.42)	84.93 (3.42)	84.40 (3.37)
letter	**53.97** (7.50)	52.94 (7.87)	52.94 (7.56)	53.09 (7.42)	**53.97** (7.50)
mfeat-morphological	68.07 (2.52)	**68.33** (2.02)	**68.33** (2.02)	**68.33** (2.02)	68.07 (2.52)
optdigits	**94.87** (2.65)	94.73 (2.14)	94.73 (2.14)	94.73 (2.14)	**94.87** (2.65)
page-blocks	89.40 (1.84)	**90.20** (2.16)	**90.20** (2.04)	**90.20** (2.04)	89.40 (1.84)
segment	**91.00** (2.76)	90.60 (2.77)	90.60 (2.77)	90.53 (2.70)	**91.00** (2.76)
synthetic-control	95.27 (1.73)	**95.53** (1.63)	**95.53** (1.63)	**95.53** (1.63)	95.27 (1.73)
vehicle	69.13 (4.41)	**69.40** (2.73)	69.27 (2.96)	69.27 (3.15)	69.20 (4.46)
vowel	75.40 (3.09)	**75.67** (3.08)	**75.67** (3.08)	**75.67** (3.08)	75.40 (3.09)
waveform	69.60 (2.40)	**69.67** (2.78)	**69.67** (2.78)	**69.67** (2.78)	69.60 (2.40)
average	73.84	**73.98**	**73.98**	73.97	73.90
average rank	3.3	**2.8**	2.9	3	3

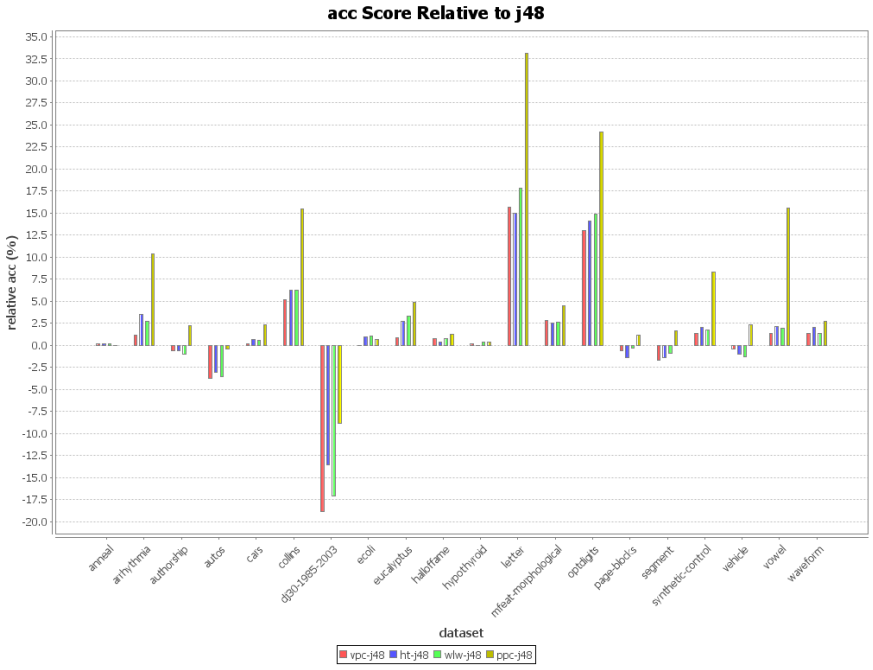


Figure 5.13: Relative accuracy for decision trees under accuracy metric.

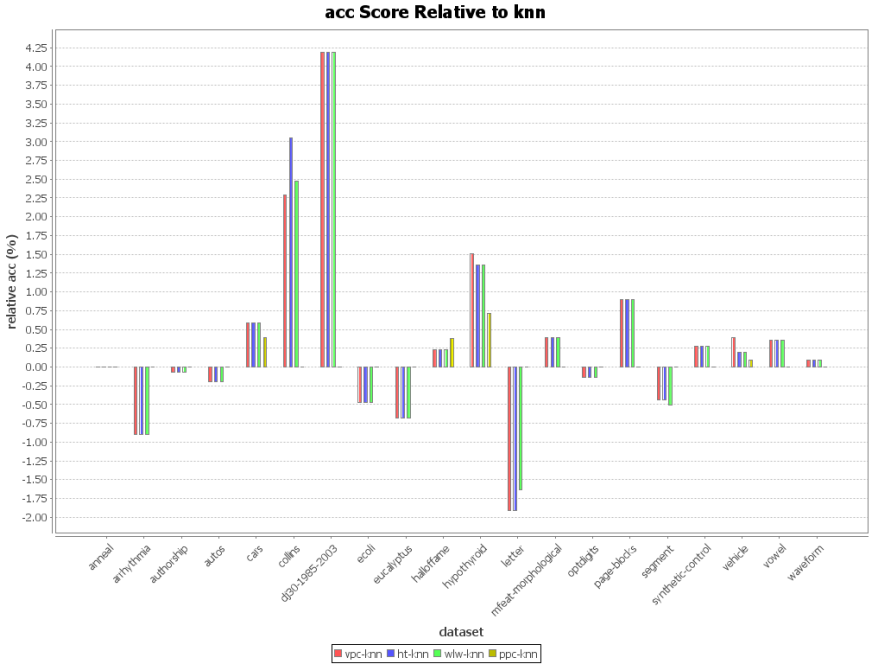


Figure 5.14: Relative accuracy for k-nearest neighbor under accuracy metric.

5.7.1.3 Random Forests (RF-100)

Table 5.10 indicates the accuracy of the various pairwise classification methods while using the random forest algorithm as the base classifier. Figure 5.15 shows the

relative gain in accuracy over a multiclass random forest classifier.

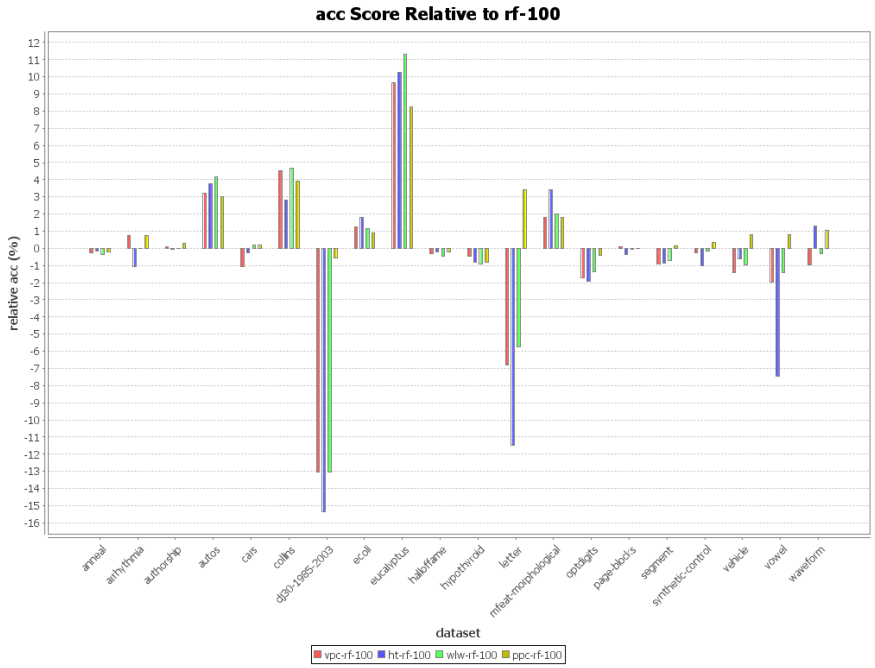


Figure 5.15: Relative accuracy for Random Forest under accuracy metric.

5.7.1.4 Support Vector Machines (SVM-121)

Table 5.11 indicates the accuracy of the various pairwise classification methods while using the support vector machine as the base classifier.

5.7.2 Predicting Probabilities

5.7.2.1 Decision Tree

Table 5.12 indicates the rectified Brier score of the various pairwise classification methods while using the decision tree J48 as the base classifier. Figure 5.16 shows the relative gain in rectified Brier score over a multiclass J48 decision tree.

Table 5.10: Results for rf100 under the accuracy metric

dataset	multi-rf-100	vpc-rf-100	ht-rf-100	wlw-rf-100	ppc-rf-100
anneal	**99.20** (0.61)	98.93 (0.90)	99.07 (0.72)	98.87 (1.04)	99.00 (0.72)
arrhythmia	72.40 (3.00)	**72.95** (2.67)	71.63 (2.61)	72.40 (3.19)	**72.95** (2.79)
authorship	98.73 (0.86)	98.80 (1.08)	98.67 (1.04)	98.73 (1.11)	**99.00** (0.85)
autos	77.94 (3.40)	80.44 (3.54)	80.88 (4.16)	**81.18** (3.38)	80.29 (4.50)
cars	84.26 (2.82)	83.38 (4.94)	84.04 (3.95)	**84.41** (4.67)	**84.41** (3.95)
collins	42.67 (2.01)	44.60 (3.41)	43.87 (4.21)	**44.67** (3.27)	44.33 (2.69)
dj30-1985-2003	**26.27** (5.64)	22.84 (5.54)	22.24 (5.00)	22.84 (4.45)	26.12 (5.51)
ecoli	85.05 (3.72)	86.12 (3.69)	**86.60** (3.33)	86.02 (3.21)	85.83 (3.46)
eucalyptus	51.87 (3.74)	56.87 (4.21)	57.20 (3.11)	**57.73** (4.73)	56.13 (3.76)
halloffame	**90.73** (2.56)	90.47 (1.91)	90.53 (1.91)	90.33 (1.87)	90.53 (1.91)
hypothyroid	**97.27** (1.42)	96.80 (1.69)	96.47 (1.81)	96.40 (2.14)	96.47 (2.01)
letter	56.32 (4.86)	52.50 (9.51)	49.85 (7.83)	53.09 (8.89)	**58.24** (8.15)
mfeat-morphological	70.13 (2.99)	71.40 (2.77)	**72.53** (2.72)	71.53 (2.67)	71.40 (3.72)
optdigits	**93.80** (2.18)	92.20 (2.79)	92.00 (2.83)	92.53 (2.79)	93.40 (2.30)
page-blocks	93.87 (2.53)	**93.93** (2.46)	93.53 (2.22)	93.80 (2.22)	93.87 (2.20)
segment	94.27 (1.78)	93.40 (2.50)	93.47 (2.28)	93.60 (2.58)	**94.40** (2.02)
synthetic-control	97.87 (0.88)	97.60 (0.90)	96.87 (0.95)	97.73 (1.05)	**98.20** (0.89)
vehicle	76.07 (2.16)	75.00 (2.87)	75.60 (1.55)	75.33 (3.16)	**76.67** (2.85)
vowel	74.87 (3.64)	73.40 (4.43)	69.27 (3.99)	73.80 (4.03)	**75.47** (4.31)
waveform	81.00 (3.30)	80.20 (2.69)	**82.07** (2.05)	80.73 (2.91)	81.87 (2.31)
average	78.23	78.09	77.82	78.29	**78.93**
average rank	2.8	3.4	3.5	3.1	**2.2**

Table 5.11: Results for svm121 under the accuracy metric

dataset	vpc-svm-121	ht-svm-121	wlw-svm-121	ppc-svm-121
anneal	97.47 (0.30)	**97.60** (0.76)	97.33 (0.82)	97.33 (0.67)
arrhythmia	**73.33** (3.03)	72.71 (2.87)	73.02 (3.17)	72.71 (3.07)
authorship	99.47 (0.87)	99.33 (0.82)	99.47 (0.56)	**99.73** (0.60)
autos	67.06 (2.23)	**69.12** (4.99)	67.06 (2.46)	68.82 (1.61)
cars	**77.50** (2.18)	76.18 (2.63)	**77.50** (3.47)	**77.50** (2.88)
collins	40.27 (3.58)	**44.93** (4.34)	42.80 (3.57)	44.40 (4.96)
dj30-1985-2003	14.63 (5.21)	19.40 (3.66)	16.72 (3.72)	**29.25** (5.44)
ecoli	82.33 (5.29)	81.75 (4.47)	82.33 (5.25)	**83.50** (2.91)
eucalyptus	52.80 (2.80)	**56.00** (2.79)	53.73 (1.92)	55.20 (2.64)
halloffame	90.93 (1.92)	90.53 (2.38)	**91.07** (2.93)	90.93 (3.08)
hypothyroid	**90.80** (3.84)	90.27 (3.70)	90.40 (3.61)	90.40 (2.89)
letter	37.94 (9.32)	45.88 (9.38)	39.71 (8.25)	**56.76** (8.74)
mfeat-morphological	72.67 (4.85)	70.93 (4.89)	72.93 (3.93)	**74.27** (2.43)
optdigits	90.27 (5.97)	92.80 (2.18)	92.80 (2.60)	**93.73** (1.67)
page-blocks	89.20 (2.38)	90.80 (2.23)	89.33 (2.36)	**91.60** (2.48)
segment	90.93 (1.86)	89.20 (3.21)	**92.00** (1.94)	92.00 (1.76)
synthetic-control	90.53 (10.00)	97.47 (1.73)	**98.13** (1.10)	98.00 (1.05)
vehicle	79.33 (1.41)	79.33 (1.33)	79.73 (2.34)	**80.00** (2.26)
vowel	77.33 (4.03)	76.40 (2.97)	78.93 (3.39)	**80.93** (1.74)
waveform	82.40 (3.67)	82.67 (3.43)	82.53 (3.57)	**83.20** (3.51)
average	74.86	76.17	75.88	**78.01**
average rank	3.1	2.8	2.4	**1.7**

Table 5.12: Results for j48 under the rectified Brier metric

dataset	multi-j48	vpc-j48	ht-j48	wlw-j48	ppc-j48
anneal	99.15 (0.48)	99.20 (0.42)	99.20 (0.40)	99.23 (0.40)	**99.30** (0.35)
arrhythmia	89.43 (1.45)	88.50 (1.34)	91.01 (0.90)	90.95 (0.87)	**92.85** (0.69)
authorship	96.20 (0.79)	95.70 (1.05)	96.23 (1.16)	96.08 (1.15)	**97.23** (0.45)
autos	90.47 (2.46)	87.29 (2.66)	90.51 (1.37)	90.25 (1.80)	**91.89** (0.94)
cars	88.74 (1.88)	87.01 (1.96)	90.14 (1.87)	89.79 (2.04)	**91.03** (1.54)
collins	89.49 (0.76)	88.51 (0.47)	92.29 (0.45)	91.86 (0.49)	**93.34** (0.27)
dj30-1985-2003	94.14 (0.52)	92.06 (0.60)	95.57 (0.28)	95.51 (0.32)	**95.66** (0.17)
ecoli	93.49 (1.39)	92.18 (2.22)	93.92 (1.46)	93.82 (1.49)	**94.35** (1.19)
eucalyptus	87.21 (1.12)	83.09 (1.96)	89.17 (1.01)	88.79 (1.17)	**89.77** (0.47)
halloffame	92.79 (1.84)	92.40 (2.46)	93.29 (2.55)	93.58 (2.43)	**94.31** (1.60)
hypothyroid	98.76 (1.08)	98.84 (1.03)	98.68 (0.95)	98.91 (0.98)	**98.94** (0.98)
letter	94.45 (0.68)	94.31 (0.85)	96.27 (0.32)	96.18 (0.38)	**96.52** (0.28)
mfeat-morphological	95.36 (0.76)	94.39 (0.78)	95.98 (0.45)	95.89 (0.52)	**96.19** (0.38)
optdigits	95.04 (0.56)	96.36 (0.62)	97.37 (0.36)	97.18 (0.36)	**97.96** (0.25)
page-blocks	97.24 (0.47)	96.67 (1.14)	96.88 (0.77)	97.13 (0.84)	**97.83** (0.51)
segment	97.94 (0.59)	97.41 (0.83)	98.03 (0.62)	97.97 (0.67)	**98.53** (0.34)
synthetic-control	95.61 (0.89)	95.89 (1.30)	96.63 (0.97)	96.54 (1.01)	**98.05** (0.37)
vehicle	87.62 (1.74)	85.27 (2.00)	89.39 (1.21)	89.18 (1.27)	**90.72** (0.98)
vowel	93.89 (0.79)	92.87 (0.46)	95.07 (0.33)	94.85 (0.34)	**95.81** (0.27)
waveform	80.78 (2.09)	80.00 (1.70)	81.78 (1.65)	81.65 (1.71)	**85.68** (1.60)
average	92.89	91.90	93.87	93.77	**94.80**
average rank	4	4.8	2.4	3	**1**

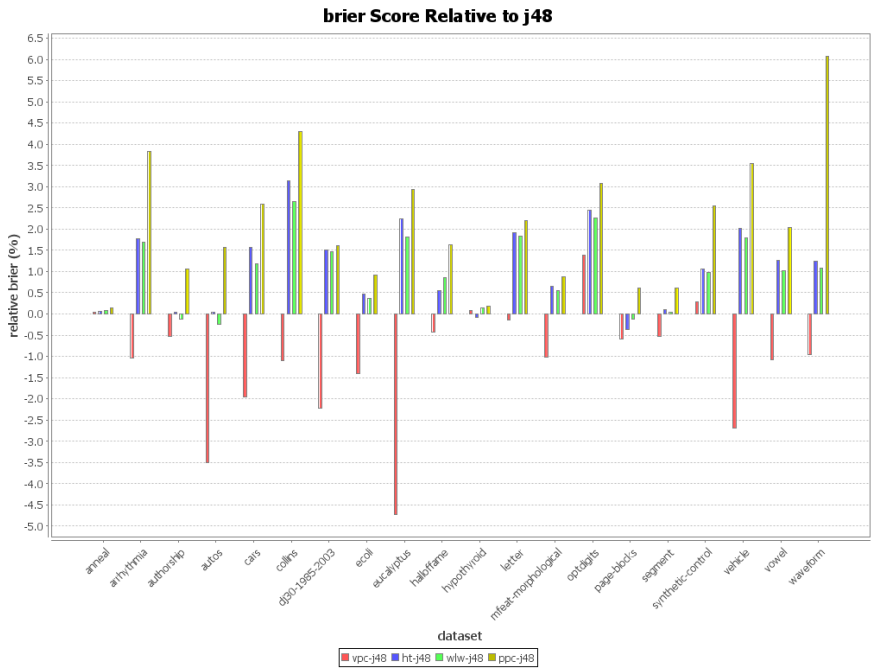


Figure 5.16: Relative Brier score for decision trees.

5.7.2.2 Nearest Neighbor

Table 5.13 indicates the rectified Brier score of the various pairwise classification methods while using the K-nearest neighbor algorithm as the base classifier. Figure 5.17 shows the relative gain in rectified Brier score over a multiclass KNN classifier.

5.7.2.3 Random Forests

Table 5.14 indicates the rectified Brier score of the various pairwise classification methods while using the random forest algorithm as the base classifier. Figure 5.18 shows the relative gain in rectified Brier score over a multiclass random forest classifier.

5.7.2.4 Support Vector Machines

Table 5.15 indicates the rectified Brier score of the various pairwise classification methods while using the support vector machine as the base classifier.

Table 5.13: Results for knn under the rectified Brier metric

dataset	multi-knn	vpc-knn	ht-knn	wlw-knn	ppc-knn
anneal	98.19 (0.68)	98.17 (0.69)	**98.21** (0.67)	98.20 (0.68)	98.19 (0.68)
arrhythmia	84.36 (1.60)	83.84 (1.60)	84.54 (1.48)	84.33 (1.48)	**84.79** (1.55)
authorship	99.60 (0.51)	99.57 (0.45)	99.56 (0.43)	99.57 (0.44)	**99.64** (0.44)
autos	89.89 (1.75)	89.47 (1.85)	**90.45** (1.60)	90.02 (1.72)	89.95 (1.76)
cars	83.51 (2.04)	83.63 (1.89)	**83.96** (1.85)	83.85 (1.87)	83.90 (1.91)
collins	88.59 (0.60)	88.33 (0.51)	**90.57** (0.38)	89.27 (0.46)	88.96 (0.60)
dj30-1985-2003	93.70 (0.43)	92.97 (0.50)	**95.43** (0.15)	94.97 (0.29)	93.81 (0.44)
ecoli	91.04 (1.58)	90.68 (1.69)	**91.29** (1.63)	90.99 (1.73)	91.27 (1.56)
eucalyptus	80.06 (1.43)	79.60 (1.43)	**80.73** (1.39)	80.17 (1.44)	80.66 (1.66)
halloffame	91.50 (1.86)	91.56 (1.70)	91.76 (1.65)	91.72 (1.66)	**91.81** (1.78)
hypothyroid	89.30 (2.48)	90.04 (2.22)	**90.13** (2.24)	90.12 (2.23)	89.97 (2.24)
letter	95.41 (0.74)	94.77 (0.87)	**95.96** (0.26)	95.95 (0.54)	95.49 (0.71)
mfeat-morphological	93.81 (0.49)	93.67 (0.40)	**94.78** (0.31)	94.05 (0.37)	93.87 (0.48)
optdigits	99.00 (0.51)	98.95 (0.43)	98.84 (0.33)	98.98 (0.40)	**99.01** (0.50)
page-blocks	95.83 (0.73)	96.08 (0.86)	**96.27** (0.77)	96.22 (0.79)	96.09 (0.74)
segment	97.48 (0.77)	97.31 (0.79)	97.51 (0.70)	97.42 (0.76)	**97.54** (0.77)
synthetic-control	98.45 (0.57)	98.51 (0.54)	**98.57** (0.49)	98.55 (0.53)	98.53 (0.56)
vehicle	84.77 (2.18)	84.70 (1.36)	85.28 (1.38)	84.99 (1.40)	**85.41** (1.99)
vowel	95.68 (0.54)	95.58 (0.56)	**95.99** (0.45)	95.82 (0.52)	95.72 (0.52)
waveform	79.93 (1.58)	79.78 (1.85)	80.27 (1.80)	80.06 (1.83)	**80.57** (1.72)
average	91.51	91.36	**92.01**	91.76	91.76
average rank	3.9	4.6	**1.6**	2.6	2.2

Table 5.14: Results for rf100 under the rectified Brier metric

dataset	multi-rf-100	vpc-rf-100	ht-rf-100	wlw-rf-100	ppc-rf-100
anneal	99.35 (0.11)	**99.47** (0.45)	99.33 (0.13)	99.32 (0.16)	99.36 (0.13)
arrhythmia	92.24 (0.57)	89.18 (1.07)	91.99 (0.48)	92.11 (0.51)	**92.31** (0.52)
authorship	98.15 (0.15)	**99.40** (0.54)	97.91 (0.21)	97.91 (0.20)	98.08 (0.18)
autos	93.80 (0.47)	92.18 (1.42)	93.78 (0.48)	93.83 (0.46)	**93.92** (0.49)
cars	**92.61** (0.94)	88.92 (3.29)	92.47 (1.07)	92.26 (1.27)	92.47 (1.13)
collins	93.61 (0.13)	89.93 (0.62)	93.61 (0.15)	93.66 (0.16)	**93.70** (0.16)
dj30-1985-2003	95.35 (0.42)	92.28 (0.55)	95.58 (0.26)	**95.61** (0.28)	95.59 (0.31)
ecoli	94.54 (1.31)	93.06 (1.85)	94.68 (1.32)	94.65 (1.26)	**94.71** (1.29)
eucalyptus	87.94 (0.61)	82.75 (1.68)	89.03 (0.54)	**89.17** (0.59)	88.86 (0.56)
halloffame	95.52 (0.77)	93.64 (1.28)	**95.56** (0.75)	95.52 (0.80)	95.53 (0.77)
hypothyroid	**98.55** (0.44)	97.87 (1.12)	98.24 (0.55)	98.32 (0.60)	98.33 (0.62)
letter	**96.62** (0.23)	94.72 (1.06)	95.99 (0.19)	96.08 (0.19)	96.60 (0.23)
mfeat-morphological	95.97 (0.39)	94.28 (0.55)	**96.09** (0.38)	96.03 (0.38)	96.09 (0.38)
optdigits	97.79 (0.16)	**98.44** (0.56)	96.98 (0.17)	97.02 (0.19)	97.69 (0.18)
page-blocks	**98.14** (0.54)	97.57 (0.99)	98.04 (0.45)	98.09 (0.51)	98.12 (0.48)
segment	**98.63** (0.24)	98.11 (0.72)	98.45 (0.34)	98.48 (0.36)	98.62 (0.29)
synthetic-control	98.58 (0.19)	**99.20** (0.30)	98.41 (0.14)	98.51 (0.14)	98.62 (0.15)
vehicle	92.16 (0.46)	87.50 (1.43)	91.95 (0.48)	91.93 (0.54)	**92.23** (0.50)
vowel	96.23 (0.33)	95.16 (0.80)	95.36 (0.23)	95.67 (0.22)	**96.28** (0.25)
waveform	90.36 (0.57)	86.80 (1.79)	**90.48** (0.40)	90.24 (0.50)	90.34 (0.50)
average	95.31	93.52	95.20	95.22	**95.37**
average rank	2.4	4.2	3.3	3.2	**1.9**

Table 5.15: Results for svm121 under the rectified Brier metric

dataset	vpc-svm-121	ht-svm-121	wlw-svm-121	ppc-svm-121
anneal	98.73 (0.15)	98.86 (0.19)	98.83 (0.16)	**98.93** (0.14)
arrhythmia	89.33 (1.21)	**92.00** (0.77)	91.92 (0.71)	91.86 (0.64)
authorship	**99.73** (0.43)	99.24 (0.36)	99.24 (0.46)	99.55 (0.24)
autos	86.82 (0.89)	91.09 (0.95)	91.14 (0.44)	**91.95** (0.38)
cars	85.00 (1.45)	90.61 (0.49)	90.79 (0.74)	**91.00** (0.78)
collins	89.14 (0.65)	93.67 (0.17)	93.65 (0.19)	**93.67** (0.14)
dj30-1985-2003	91.46 (0.52)	95.46 (0.03)	95.41 (0.06)	**95.64** (0.07)
ecoli	91.17 (2.65)	93.66 (1.31)	93.84 (1.37)	**93.85** (1.13)
eucalyptus	81.12 (1.12)	**88.71** (0.35)	88.62 (0.20)	88.60 (0.22)
halloffame	93.96 (1.28)	95.10 (1.03)	95.28 (1.03)	**95.43** (1.10)
hypothyroid	93.87 (2.56)	94.92 (1.70)	95.35 (2.10)	**96.01** (0.97)
letter	93.10 (1.04)	95.52 (0.12)	95.46 (0.16)	**96.44** (0.32)
mfeat-morphological	94.53 (0.97)	95.83 (0.37)	95.93 (0.25)	**96.29** (0.42)
optdigits	98.05 (1.19)	97.90 (0.40)	98.05 (0.63)	**98.91** (0.27)
page-blocks	95.68 (0.95)	97.04 (0.72)	96.93 (0.72)	**97.38** (0.62)
segment	97.41 (0.53)	97.43 (0.43)	97.67 (0.47)	**98.03** (0.33)
synthetic-control	96.84 (3.33)	98.24 (0.93)	98.68 (0.45)	**99.39** (0.14)
vehicle	89.67 (0.71)	93.19 (0.24)	93.41 (0.40)	**93.43** (0.44)
vowel	95.88 (0.73)	96.47 (0.33)	96.93 (0.31)	**97.50** (0.26)
waveform	88.27 (2.45)	**91.79** (1.09)	91.76 (1.26)	91.76 (1.06)
average	92.49	94.84	94.94	**95.28**
average rank	3.8	2.5	2.4	**1.3**

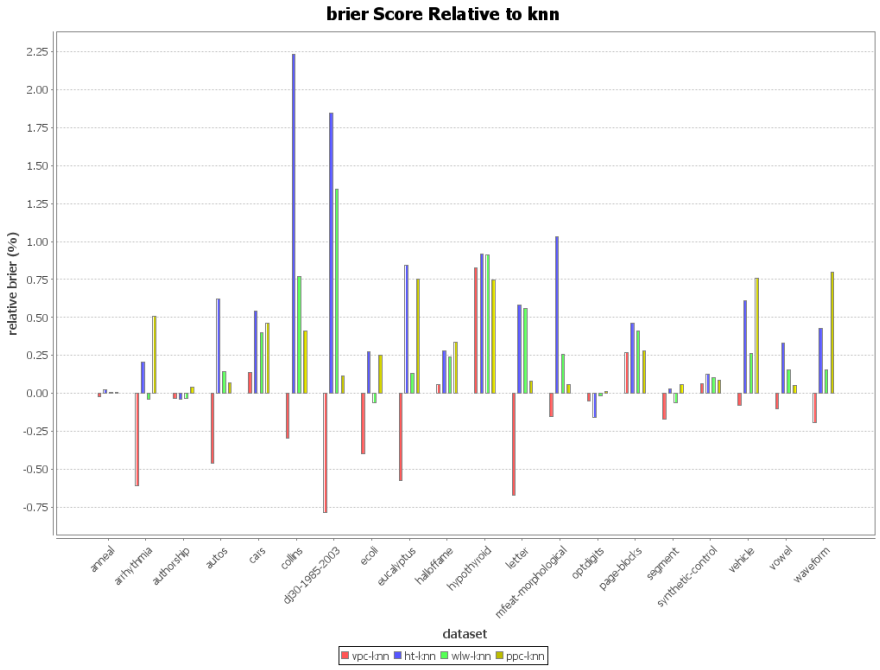


Figure 5.17: Relative Brier score for KNN.

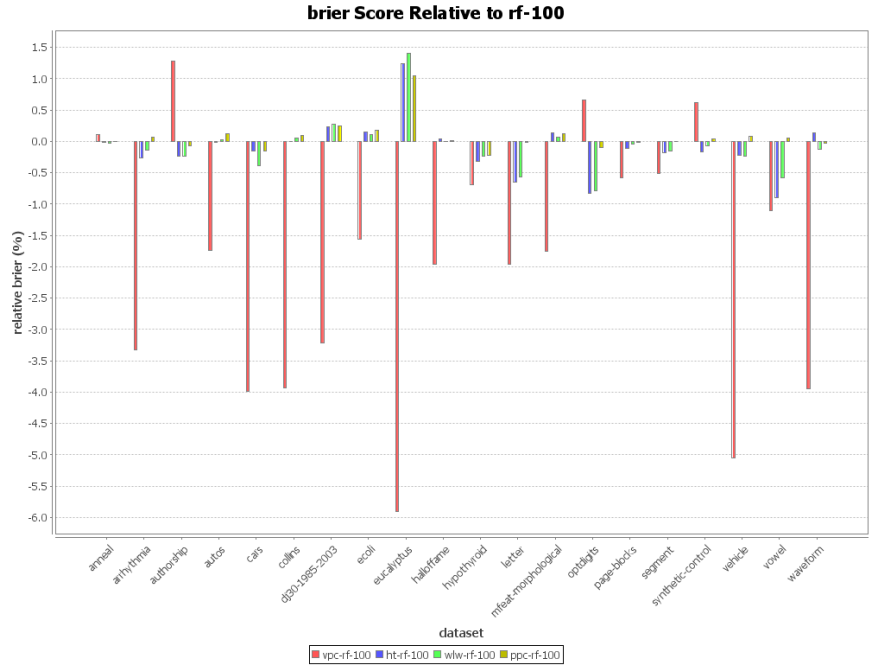


Figure 5.18: Relative Brier score for RF-100.

Chapter 6

Conclusion

6.1 Conclusions

In this thesis, we investigated several previously unexplored aspects of multiclass classification, focusing in particular on the related issues of model combination and model selection, and how to perform model selection for models that are combined. We experimented with previously unexplored algorithms in commensurate model combination and complementary model combination. Specifically, we showed that linear combinations of multiclass classifiers benefit significantly from ridge regularization (Chapter 3), and that applying one weight per prediction rather than one weight per classifier allows classifiers to focus on subproblems. We also showed that when multiclass classification problems are reduced to binary classification problems, it is often more effective to perform shared-hyperparameter optimization than to optimize each subproblem independently (Chapter 4). Shared-hyperparameter optimization is more effective because it provides regularization, reducing the probability of choosing poor binary models, and because subproblems typically share similar structure. Finally, we proposed an algorithm called probabilistic pairwise classification (PPC) that overcomes a well-known flaw in other pairwise classification approaches, and is also conceptually simple and easy to implement. Evaluation of PPC on real-world data sets indicates that it is superior not only to other commonly used pairwise classification approaches, but can also be used to improve the classification performance of multiclass classifiers.

The tradeoff is that PPC is computationally more expensive, but this cost is mitigated somewhat by the fact that it is easily parallelizable.

6.2 Future Work

In regularized linear combinations of multiclass classifiers (Chapter 3), we used the single-point maximum-likelihood estimates implicit in the ridge and lasso regularizers. An interesting extension of this work would be to examine the full Bayesian solutions corresponding to the Gaussian prior over weights (corresponding to ridge regularization) Laplacian prior over weights (corresponding to lasso regularization). Other work could study additional regularization by selecting a single regularization hyperparameter for use in all subproblems or constraining the weights to be non-negative for each subproblem.

Most of this work has focused on the accuracy metric (Chapters 3 - 5), with some studies under the Brier metric (Chapter 5). Future work should investigate how our results on model selection and regularization transfer to other domain-specific metrics.

One of the issues with model selection in subproblems induced by multiclass classification is the relationship between the binary metric and the multiclass metric. Other studies have shown that a mismatch between the training metric and the target metric is problematic [17]. Further studies could investigate the relationship between the training and target metrics in the case of multiclass to binary reduction methods.

While this thesis focused on the complementary issues of model selection and model combination, and how to perform model selection for models that are combined, we have only used the shared-hyperparameters technique for the PPC algorithm. Another valuable line of research would be to see whether PPC attains a higher benefit from independent optimization than that attained by one-vs-all or other pairwise classification techniques. Since the Theorem of Total Probability is the foundation for PPC, being able to more accurately estimate each term in the total probability should lead

to more accurate multiclass predictions. However, further studies would need to be performed to identify whether PPC requires shared-hyperparameters for the same reasons as one-vs-all and other pairwise classification techniques, namely that smoothing over subproblems provides a necessary regularization and that subproblems are often similar. One aspect of PPC that may make it more amenable to improved accuracy by independent optimization is that it already has a level of smoothing (see Equation (5.7)), which may alleviate the need for shared hyperparameter constraints.

Bibliography

[1] David W. Aha and Dennis Kibler. Instance-based learning algorithms. In Machine Learning, pages 37–66, 1991.

[2] Erin Allwein, Robert Schapire, Yoram Singer, and Pack Kaelbling. Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research, 1:113–141, 2000.

[3] Ethem Alpaydin. Introduction to Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2004.

[4] Kenneth J. Arrow. Social choice and individual values. Yale University Press, 1951.

[5] Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, and Bianca Zadrozny. Error limiting reductions between classification tasks. In ICML '05: Proceedings of the 22nd International Conference on Machine learning, pages 49–56, New York, NY, USA, 2005. ACM.

[6] Alina Beygelzimer, John Langford, and Pradeep Ravikumar. Error-correcting tournaments. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, ALT, volume 5809 of Lecture Notes in Computer Science, pages 247–262. Springer, 2009.

[7] Alina Beygelzimer, John Langford, and Bianca Zadrozny. Weighted one-against-all. In Manuela M. Veloso and Subbarao Kambhampati, editors, AAAI, pages 720–725. AAAI Press / The MIT Press, 2005.

[8] A. L. Blum and R. L. Rivest. Training a 3-node neural network is np-complete. In S. J. Hanson, W. Remmele, and R. L. Rivest, editors, Machine Learning: From Theory to Applications, pages 9–28. Springer, Berlin, Heidelberg, 1993.

[9] Kai bo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In Proceedings of the Sixth International Workshop on Multiple Classifier Systems, pages 278–285, 2005.

[10] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In COLT '92: Proceedings of the fifth annual workshop on computational learning theory, pages 144–152, New York, NY, USA, 1992. ACM.

[11] Leo Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.

[12] Leo Breiman. Stacked regressions. Machine Learning, 24(1):49–64, 1996.

[13] Leo Breiman. Using adaptive bagging to debias regressions, 1999.

[14] Leo Breiman. Random forests. Machine Learning, 45(1):5–32, 2001.

[15] Leo Breiman. Some infinity theory for predictor ensembles, 2001.

[16] G. W. Brier. Verification of forecasts expressed in terms of probability. Monthly Weather Review, 75, 1950.

[17] Rich Caruana, Art Munson, and Alexandru Niculescu-Mizil. Getting the most out of ensemble selection. In ICDM '06: Proceedings of the Sixth International Conference on Data Mining, pages 828–833, Washington, DC, USA, 2006. IEEE Computer Society.

[18] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning, page 18, New York, NY, USA, 2004. ACM.

[19] Philip K. Chan and Salvatore J. Stolfo. On the accuracy of meta-learning for scalable data mining. Journal of Intelligent Information Systems, 8(1):5–28, 1997.

[20] Robert T. Clemen and Robert L. Winkler. Limits for the precision and value of information from dependent sources. Operations Research, 3(2):427442, 1985.

[21] Marquis de Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. Imprimerie Royale, Paris, 1785.

[22] Koby Crammer and Yoram Singer. Improved output coding for classification using continuous relaxation. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, NIPS, pages 437–443. MIT Press, 2000.

[23] Florin Cutzu. Polychotomous classification with pairwise classifiers: A new voting principle. In Windeatt and Roli [99], pages 115–124.

[24] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7:1–30, 2006.

[25] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation, 10(7):1895–1923, 1998.

[26] Thomas G. Dietterich. Ensemble methods in machine learning. Lecture Notes in Computer Science, 1857:1–15, 2000.

[27] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 2:263–286, 1995.

[28] Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. In Proc. 17th International Conference on Machine Learning, pages 223–230. Morgan Kaufmann, San Francisco, CA, 2000.

[29] Kaibo Duan, S. Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. Multi-category classification by soft-max combination of binary classifiers. In Windeatt and Roli [99], pages 125–134.

[30] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, November 2000.

[31] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? Machine Learning, 54(3):255–273, 2004.

[32] Bradley Efron, Trevor Hastie, Lain Johnstone, and Robert Tibshirani. Least angle regression. Annals of Statistics, 32:407–499, 2004.

[33] Bruno Feres de Souza, Andre C. P. L. F. de Carvalho, Rodrigo Calvo, and Renato Porfirio Ishii. Multiclass svm model selection using particle swarm optimization. In HIS '06: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems, page 31, Washington, DC, USA, 2006. IEEE Computer Society.

[34] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In EMCL '01: Proceedings of the 12th European Conference on Machine Learning, pages 145–156, London, UK, 2001. Springer-Verlag.

[35] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In International Conference on Machine Learning, pages 148–156, 1996.

[36] J. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. Technical report, Stanford, 2008.

[37] J. Friedman and B. Popescu. Importance sampled learning ensembles, 2003.

[38] Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[39] Johannes Fürnkranz. Pairwise classification as an ensemble technique. In ECML '02: Proceedings of the 13th European Conference on Machine Learning, pages 97–110, London, UK, 2002. Springer-Verlag.

[40] Johannes Fürnkranz. Round robin classification. Journal of Machine Learning Research, 2:721–747, 2002.

[41] Johannes Fürnkranz. Round robin ensembles. Intell. Data Anal., 7(5):385–403, 2003.

[42] Salvador García and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research, 9:2677–2694, December 2008.

[43] Ashutosh Garg and Vladimir Pavlovic. Bayesian networks as ensemble of classifiers. In ICPR, Quebec City, Quebec, 2002.

[44] Christian Genest and Kevin J. McConway. Allocating the weights in the linear opinion pool. Journal of Forecasting, 9(53-73), 1990.

[45] Christian Genest and Mark J. Schervish. Modeling expert judgments for bayesian updating. The Annals of Statistics, 13:1198–1212, 1985.

[46] Zoubin Ghahramani and Hyun-Chul Kim. Bayesian classifier combination. Gatsby Technical Report, 2003.

[47] Peter D. Grnwald. The Minimum Description Length Principle. Number 0262072815 in MIT Press Books. The MIT Press, December 2007.

[48] L. K. Hansen and P. Salamon. Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell., 12(10):993–1001, 1990.

[49] Sherif Hashem. Effects of collinearity on combining neural networks. Connection Science, 8(3):315–336, 1996.

[50] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical learning. Springer, 2003.

[51] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In The Annals of Statistics, pages 507–513. MIT Press, 1996.

[52] Tin Kam Ho. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell., 20(8):832–844, 1998.

[53] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989.

[54] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines, 2002.

[55] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on, 13(2):415–425, 2002.

[56] Eyke Hüllermeier and Stijn Vanderlooy. Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. Pattern Recogn., 43(1):128–142, 2010.

[57] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. Information Processing Letters, 5(1):15–17, 1976.

[58] Robert A. Jacobs. Methods for combining experts' probability assessments. Neural Computation, 7(5):867–888, 1995.

[59] G. James and T. Hastie. Generalizations of the bias/variance decomposition for prediction error. Technical report.

[60] Joseph M. Kahn. A generative bayesian model for aggregating experts' probabilities. In AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence, pages 301–308, Arlington, Virginia, United States, 2004. AUAI Press.

[61] Ron Kohavi and George H. John. Wrappers for feature subset selection. Artificial Intelligence, 97(1-2):273–324, 1997.

[62] Ludmila I. Kuncheva. Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, 2004.

[63] L. Lam and C.Y. Suen. Application of majority voting to pattern recognition: An analysis of the behavior and performance. IEEE Trans. Systems Man Cybernet, 27(5):553–567, 1997.

[64] John Langford and Alina Beygelzimer. Sensitive error correcting output codes. In Peter Auer and Ron Meir, editors, COLT, volume 3559 of Lecture Notes in Computer Science, pages 158–172. Springer, 2005.

[65] Gilles Lebrun, Olivier Lezoray, Christophe Charrier, and Hubert Cardot. An ea multi-model selection for svm multiclass schemes. In Francisco Sandoval Hernández, Alberto Prieto, Joan Cabestany, and Manuel Graña, editors, IWANN, volume 4507 of Lecture Notes in Computer Science, pages 260–267. Springer, 2007.

[66] Martina Liepert. Topological fields chunking for german with svm's: Optimizing svm-parameters with ga's. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2003), Borovets, Bulgaria, 2003.

[67] Chih-Wei Hsu Chih-Jen Lin. A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on, 13:415–425, 2002.

[68] D. V. Lindley. Reconciliation of discrete probability distributions. Bayesian Statistics 2, pages 375–390, 1985.

[69] Y. Liu and X. Yao. Ensemble learning via negative correlation. Neural Networks, 12(10):1399–1404, 1999.

[70] Ana Carolina Lorena and André C. P. L. F. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. Neurocomput., 71(16-18):3326–3334, 2008.

[71] K. J. McConway. Marginalization and linear opinion pools. Journal of the American Statistical Association, 76(374):410–414, 1981.

[72] P. Melville and R. J. Mooney. Constructing diverse classifier ensembles using artificial training examples. In Eighteenth International Joint Conference on Artificial Intelligence, pages 505–510, 2003.

[73] Iain Melvin, Eugene Ie, Jason Weston, William Stafford Noble, and Christina Leslie. Multi-class protein classification using adaptive codes. Journal of Machine Learning Research, 8:1557–1581, 2007.

[74] Thomas P. Minka. Bayesian model averaging is not model combination.

[75] Tom M. Mitchell. Machine Learning. McGraw-Hill, New York, 1997.

[76] Peter Morris. Bayesian Expert Resolution. PhD thesis, Stanford University, 1971.

[77] Peter Morris. An axiomatic approach to expert resolution. Management Sciences, 29(1):24–32, 1979.

[78] D. W. Opitz and J. W. Shavlik. Actively searching for an effective neural-network ensemble. Connection Science, 8(3/4):337–353, 1996.

[79] David M. Pennock, Pedrito Maynard-Reid II, C. Lee Giles, and Eric Horvitz. A normative examination of ensemble learning algorithms. In Proc. 17th International Conference on Machine Learning, pages 735–742. Morgan Kaufmann, San Francisco, CA, 2000.

[80] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, Neural Networks for Speech and Image Processing, pages 126–142. Chapman-Hall, 1993.

[81] John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In Advances in Neural Information Processing Systems, pages 547–553. MIT Press, 2000.

[82] John C. Platt and John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Advances in Large Margin Classifiers, pages 61–74. MIT Press, 1999.

[83] J. Ross Quinlan. C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann, 1 edition, January 1993.

[84] Samuel Robert Reid and Gregory Z. Grudic. Regularized linear models in stacked generalization. In Jon Atli Benediktsson, Josef Kittler, and Fabio Roli, editors, MCS, volume 5519 of Lecture Notes in Computer Science, pages 112–121. Springer, 2009.

[85] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. Journal of Machine Learning Research, 5:101–141, 2004.

[86] Fabio Roli, Giorgio Giacinto, and Gianni Vernazza. Methods for designing multiple classifier systems. In MCS '01: Proceedings of the Second International Workshop on Multiple Classifier Systems, pages 78–87, London, UK, 2001. Springer-Verlag.

[87] C. Schaffer. Cross-validation, stacking, and bi-level stacking: Meta-methods for classification learning. In P. Cheeseman & R. W. Oldford (Eds.), Selecting models from data: Artificial intelligence and statistics IV, New York, NY, 1994. Springer-Verlag.

[88] Cullen Schaffer. Technical note: Selecting a classification method by cross-validation. Machine Learning, 13(1):135–143, 1993.

[89] R. Schapire. The boosting approach to machine learning: An overview, 2001.

[90] Alexander K. Seewald. How to make stacking better and faster while also taking care of an unknown weakness. In ICML, pages 554–561, 2002.

[91] Alexander K. Seewald. Towards a theoretical framework for ensemble classification. In IJCAI, pages 1443–1444, 2003.

[92] L. Shapley and B Grofman. Optimizing group judgmental accuracy in the presence of interdependencies. Public Choice, 1984.

[93] Amanda J. C. Sharkey, Noel E. Sharkey, Uwe Gerecke, and G. O. Chandroth. The "test and select" approach to ensemble combination. Lecture Notes in Computer Science, 1857, 2000.

[94] Gero Szepannek, Bernd Bischl, and Claus Weihs. On the combination of locally optimal pairwise classifiers. In MLDM '07: Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition, pages 104–116, Berlin, Heidelberg, 2007. Springer-Verlag.

[95] Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. Journal of Artificial Intelligence Research, 10:271–289, 1999.

[96] Joaquín Torres-Sospedra, Carlos Hernández-Espinosa, and Mercedes Fernández-Redondo. Combining mf networks: A comparison among statistical methods and stacked generalization. In Friedhelm Schwenker and Simone Marinai, editors, ANNPR, volume 4087 of Lecture Notes in Computer Science, pages 210–220. Springer, 2006.

[97] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. Connection Science, 8(3-4):385–403, 1996.

[98] N. Ueda and R. Nakano. Generalization error of ensemble estimators. Proc. IEEE Int'l Conf. Neural Networks, pages 90–95, 1996.

[99] Terry Windeatt and Fabio Roli, editors. Multiple Classifier Systems, 4th International Workshop, MCS 2003, Guilford, UK, June 11-13, 2003, Proceedings, volume 2709 of Lecture Notes in Computer Science. Springer, 2003.

[100] R L Winkler. Expert resolution. Manage. Sci., 32(3):298–303, 1986.

[101] David H. Wolpert. Stacked generalization. Neural Networks, 5(2):241–259, 1992.

[102] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1):67–82, April 1997.

[103] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research, 5:975–1005, 2004.

[104] P. Xu and A. K. Chan. An efficient algorithm on multi-class support vector machine model selection. In International Joint Conference on Neural Networks, 2003.

[105] Naoto Yukinawa, Shigeyuki Oba, Kikuya Kato, and Shin Ishii. Optimal aggregation of binary classifiers for multiclass cancer diagnosis using gene expression profiles. IEEE/ACM Trans. Comput. Biol. Bioinformatics, 6(2):333–343, 2009.

[106] B. Zadrozny. Reducing multiclass to binary by coupling probability estimates. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, NIPS, pages 1041–1048. MIT Press, 2001.

[107] Yi Zhang, Samuel Burer, and W. Nick Street. Ensemble pruning via semi-definite programming. Journal of Machine Learning Research, 7:1315–1338, 2006.

[108] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. Artificial Intelligence, 2002, 137(1-2): 239-263, 137(1-2):239–263, 2002.

[109] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B, 67:301–320, 2005.

Appendix A

Software and Supporting Material

Companion software, data sets, errata and other supporting material are available at `http://spot.colorado.edu/~reids/thesis/`.

Appendix B

Data Set Descriptions

(1) *anneal* (Annealing Data Set): Predict class for a steel annealing problem, given percentage of carbon, hardness, strength, thickness, width, length. This dataset is conspicuously missing important metadata, such as its origin and the meaning of the classes to be predicted; however, it is available from the UCI repository and appears in many experimental studies for machine learning algorithms. For more details, please see `http://archive.ics.uci.edu/ml/datasets/Annealing`.

(2) *arrhythmia* (Cardiac Arrhythmia Database): Predict whether a cardiac arrythmia exists, and if so, which class it belongs to (of 15 classes), given patient age, gender, height, heart rate and temporal characteristics of the cardiac waveform

(3) *authorship* (from Analysis of Categorical Data): Predict whether an author was {Austen, London, Milton, Shakespeare} based on word counts of 68 words such as {a, all, also, an, and}.

(4) *autos* (1985 Auto Imports Database): Predict the *symboling* or risk factor (from -3 to 3) based on features such as engine size, length, weight, stroke, bore, price, etc.

(5) *cars*: Predict whether a car is American, European or Japanese based on MPG, number of cylinders, engine displacement in cubic inches, horsepower, vehicle

weight, time to accelerate from 0 to 60 mph, etc.

(6) *collins*: Predict the genre of a text from the Brown corpus using features such as FirstPerson, InnerThinking, ThinkPositive, Reasoning, TimeInterval, etc.

(7) *dj30-1985-2003*: Predict the identity of a stock, given attributes such as its opening and closing values on a given date.

(8) *ecoli*: Predict localization sites in gram-negative bacteria given McGeoch's method for signal sequence recognition, von Heijne's method for signal sequence recognition, score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins, etc.

(9) *eucalyptus* (from agridatasets): Predict which eucalyptus seedlots are best for soil conservation in a seasonally dry hill country, out of {none,low,average,good,best}. The attributes include features such as {altitude, rainfall, frosts, year of planting, species, seedlot, height and stem, crown and branch form}. Note that this class could be modeled as ordinal, but in our experiments is treated as nominal.

(10) *halloffame* (from Analyzing Categorical Data, p. 418): Predict whether a baseball player was {not inducted into the Hall of Fame, elected into the Hall of Fame by the Baseball Writers' Association of America, selected by a vote of the Veterans Committee if not elected by the members of the BBWAA}

(11) *hypothyroid*: Predict presence, absence or severity of thyroid disease in patients, given measurements such as age, TSH, T3, TT4, T4U, FTI, TBG

(12) *letter*: Predict letter (A-Z) generated by one of 20 fonts and distorted, given features extracted from a pixellar representation such as statistical moments and edge counts.

(13) *mfeat-morphological*: Predict the handwritten numeral (0-9) based on Fourier

coefficients of character shapes, profile correlations, Karhunen-Love coefficients, pixel averages, Zernike moments and morphological features. The instances were constructed from Dutch utility maps.

(14) *optdigits*: Predict the handwritten numeral (0-9) based on pixel counts over windows on an 8x8 grid.

(15) *page-blocks*: Predict whether a page block is (text, horizontal line, picture, vertical line or graphic) given its height, length, area, eccentricity, percentage of black pixels, etc.

(16) *segment*: Predict whether an image segment is (brickface, sky, foliage, cement, window, path, grass) based on 19 input attributes such as column, row, number of lines passing through the region, contrast of horizontally adjacent pixels in the region, average intensity, average red, green and blue values, etc.

(17) *synthetic-control*: Predict which of the following trends a timeseries takes: (Cyclic,Decreasing-trend,Downward-shift,Increasing-trend,Normal,Upward-shift) given 60 timeseries samples.

(18) *vehicle*: Predict whether a vehicle silhouette is (opel,saab,bus,van) based on extracted features such as compactness, circularity, distance-circularity, radius ratio, kurtosis about major and minor axes.

(19) *vowel*: Predict the audio form of a vowel (hid, hId, hEd, hAd, hYd, had, hOd, hod, hUd, hud, hed) given 9 extracted features.

(20) *waveform*: Predict which of three wave types a waveform belongs to given 21 noisy attributes.