Tree-Structured Multiclass Probability Estimators

A thesis
submitted in fulfilment
of the requirements for the degree
of

Doctor of Philosophy

at

The University of Waikato

by

Tim LEATHART

in the

Machine Learning Group
Department of Computer Science

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

September 4, 2019

THE UNIVERSITY OF WAIKATO

Abstract

Faculty of Computing and Mathematical Sciences
Department of Computer Science

Doctor of Philosophy

Tree-Structured Multiclass Probability Estimators

by Tim LEATHART

Nested dichotomies are used as a method of transforming a multiclass classification problem into a series of binary problems. A binary tree structure is constructed over the label space that recursively splits the set of classes into subsets, and a binary classification model learns to discriminate between the two subsets of classes at each node. Several distinct nested dichotomy structures can be built in an ensemble for superior performance. In this thesis, we introduce two new methods for constructing more accurate nested dichotomies. Random-pair selection is a subset selection method that aims to group similar classes together in a non-deterministic fashion to easily enable the construction of accurate ensembles. Multiple subset evaluation takes this, and other subset selection methods, further by evaluating several different splits and choosing the best performing one. Finally, we also discuss the calibration of the probability estimates produced by nested dichotomies. We observe that nested dichotomies systematically produce under-confident predictions, even if the binary classifiers are well calibrated, and especially when the number of classes is high. Furthermore, substantial performance gains can be made when probability calibration methods are also applied to the internal models.

Acknowledgements

First and foremost, I would like to thank my primary supervisor, Eibe. You have been an endless source of encouragement, knowledge and support. Thank you for being so available to impart wisdom, bounce ideas off of, and of course, help me to debug my distributed WEKA experiments. It has truly been a pleasure and a privilege to be your student. I also would like to thank my other supervisors, Bernhard and Geoff. I couldn't have asked for a more well-rounded supervisory panel, and all of you have helped me so much over the course of my research.

The Machine Learning group at Waikato has been a fantastic group of people to complete my postgraduate journey with. In addition, I would like to thank WAND members Brad Cowie and Richard Sanger for our interesting lunchtime and tea room conversations over the years. I wish all of you the best for the future.

This research was supported by a University of Waikato Doctoral Scholarship, and the Marsden Fund Council from Government funding administered by the Royal Society of New Zealand. I also thank NVIDIA for donating a K40c GPU to support this research.

Finally, to all of my friends and family: thank you for all of your support over the years.

Contents

List of Figures

List of Tables

List of Publications

Some of the research leading to this thesis has appeared previously in the following publications.

- Tim Leathart, Bernhard Pfahringer, Eibe Frank: **Building Ensembles of Adaptive Nested Dichotomies with Random-Pair Selection**. – *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*, September 2016, Riva del Garda, Italy.
- Tim Leathart, Eibe Frank, Bernhard Pfahringer, Geoffrey Holmes: **Ensembles of Nested Dichotomies with Multiple Subset Evaluation**. – *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, April 2019, Macau, China.
- Tim Leathart, Eibe Frank, Bernhard Pfahringer, Geoffrey Holmes: **On Calibration of Nested Dichotomies**. – *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, April 2019, Macau, China.

The following research was published during the course of the PhD, but is not included in the thesis.

- Tim Leathart, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer: **Probability Calibration Trees**. – *Asian Conference on Machine Learning*, November 2017, Seoul, South Korea.

Chapter 1

Introduction

Classification is a ubiquitous learning task in real world scenarios. *Binary* classification is the task of classifying unseen examples in some domain as one of two categories. In *multiclass* classification, the type of task considered in this thesis, a model is trained to classify unseen examples as one of three or more categories. Figure 1.1 shows a possible taxonomy of machine learning and where multiclass classification is located in that taxonomy. Examples of multiclass classification tasks in the real world include classifying images into categories (Krizhevsky and Hinton, 2009; Russakovsky et al., 2015), character and digit recognition (LeCun et al., 1998; Acharya, Pant, and Gyawali, 2015), and document classification (Bennett and Nguyen, 2009).[1] Some machine learning algorithms, like decision trees and neural networks, can handle multiclass problems directly. Other methods, such as standard support vector machines, can only be used on two-class (binary) problems. As a result, there are a number of methods for decomposing multiclass problems into a set of binary problems. We call such techniques *multiclass transformation methods* (see Section 2.1). It has also been shown that decomposing problems in this way can even be beneficial for learning algorithms that can handle the multiclass problem directly (Mayoraz and Moreira, 1997; Fürnkranz, 2002; Knerr, Personnaz,

[1] In this thesis, we use the terms 'class' and 'label' interchangeably. However, note that multiclass classification is not to be confused with multilabel classification, where each example can have more than one label.

and Dreyfus, 1992; Pimenta and Gama, 2005; Mohr, Wever, and Hüllermeier, 2018b).

As the amount of data collected online continues to grow, modern datasets utilised in machine learning tasks are quickly increasing in size. Not only do these datasets exhibit a large number of examples and features, but many also have a very high number of classes. It is not uncommon in some application areas to see datasets containing tens of thousands, or even millions, of classes (Dekel and Shamir, 2010; Agrawal et al., 2013). As such, scalable methods for handling these large label spaces are necessary for reasonable training and prediction times.

This thesis focuses on improving aspects of *nested dichotomies* (Fox, 1997), a multiclass transformation method that exhibits logarithmic complexity in the number of classes in the best case. In nested dichotomies, a binary tree structure is induced over the labels, and a set of probabilistic binary models, learned using an appropriate base learner, are created to route unseen examples through the tree (Figure 1.2). Nested dichotomies utilise the binary probability estimates to navigate the tree, which can give rise to greater predictive performance than methods that simply take hard $0/1$ classifications, usually at the cost of a larger time taken to obtain final predictions (Beygelzimer, Langford, Lifshits, et al., 2009; Dembczyński et al., 2016).

1.1 Hypotheses

The primary aim of the research presented in this thesis is to improve upon the state-of-the-art for nested dichotomies. The main hypothesis can be stated as follows:

> The performance of nested dichotomies, and ensembles of nested dichotomies, can be improved by appropriate choice of structure(s).

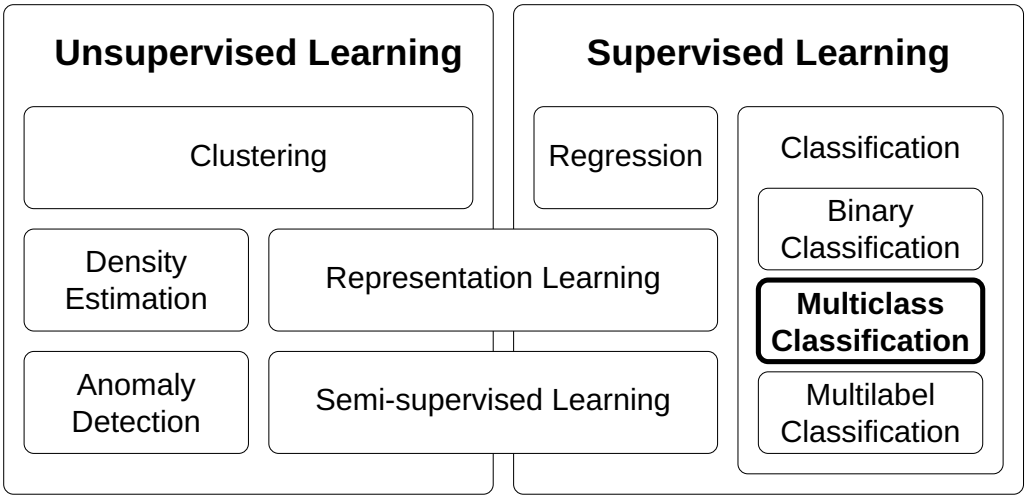


Figure 1.1: A (non-comprehensive) taxonomy of machine learning tasks. This thesis focuses on multiclass classification.

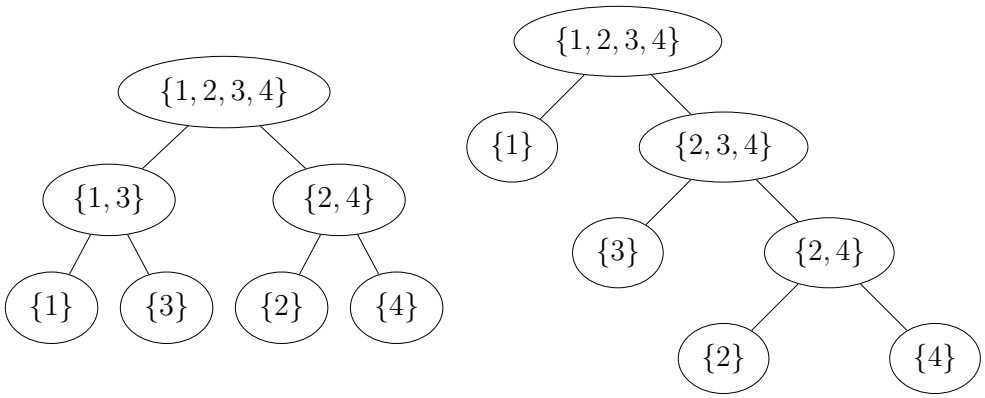


Figure 1.2: Two examples of nested dichotomies for a four class problem.

To validate this hypothesis, this thesis proposes and evaluates methods of producing nested dichotomy structures such that each internal model is learned from a comparatively simpler learning problem. Note that, as we are also concerned with ensembles of nested dichotomies, it is not enough to find the best structure[2] for a given problem. It is important that the developed methods are non-deterministic, so that a different structure is likely to be selected for each ensemble member; there is no point in forming an ensemble of predictors when all predictors are the same.

[2] i.e., the structure that gives the highest predictive accuracy.

A related, secondary hypothesis that is also considered in this thesis can be stated as follows:

> The performance of nested dichotomies can be improved by maximising the performance of each internal binary model.

This hypothesis is investigated through two avenues. Firstly, decisions on the structure of nested dichotomies are made directly by choosing internal models with greater performance. Secondly, because routing decisions are made based on probability estimates, we utilise probability calibration methods to ensure these estimates are as accurate as possible.

1.2 Contributions

The contributions of this thesis are as follows:

- A review of hierarchical multiclass transformation methods demonstrating in particular that the concept of nested dichotomies is closely related to many classification structures proposed in the machine learning literature.
- Two new methods of constructing nested dichotomies (the random-pair method and multiple subset evaluation), experimentally shown to produce nested dichotomies with high predictive performance.
- An investigation of the poor probability calibration of nested dichotomies, and empirical results showing overall predictive performance is improved by suitable application of calibration.

1.3 Mathematical Notation

In this thesis, we typically use greek symbols for parameters[3] and hyper-parameters, and Latin characters for variables that contain data, except

[3]Except for weights and bias parameters of linear models, where we follow typical conventions of using $\mathbf{w}$ or $\mathbf{W}$ and b or $\mathbf{b}$ respectively.

for sets, which are denoted by calligraphic typeface. Vectors are bold-face and lowercase (e.g. $\mathbf{c}$) while matrices are bold-face and uppercase (e.g. $\mathbf{X}$). Indexing of vectors and matrices is represented in superscript (e.g. $\mathbf{c}^{(i)}$, $\mathbf{X}^{(i,j)}$). $\mathbf{X}^{(\cdot,j)}$ refers to a column of the matrix, while $\mathbf{X}^{(i,\cdot)}$ refers to a row. Vectors are enumerated as $\mathbf{y}_i = [\mathbf{y}_i^{(1)}, \ldots, \mathbf{y}_i^{(n)}]$. Class labels are always considered to be integers. Additionally, the following conventions are used throughout the thesis for recurring items:

- $\mathbf{X}, \mathbf{y}$ represents the feature matrix and label vector of a multiclass dataset respectively,
- $\mathcal{C}$ represents the set of classes, and is of size m,
- $(\mathbf{x}_i, y_i)$ represents the feature vector and label respectively of a single instance i,
- $\mathcal{X}, \mathcal{Y}$ represent the feature and label space respectively,
- $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in (\mathcal{X} \times \mathcal{Y})\}$ represents a dataset,
- $\hat{y}_i$ represents the predicted label of a single instance i,
- $\hat{p}_i \in [0, 1]$ represents the estimated probability for a single instance i belonging to the positive class for a binary classification problem,
- $\mathbf{y}_i$ represents the true label of a single instance i in one-hot representation (i.e., a vector of length m where all values are zero except for index y_i, which is one),
- $\hat{\mathbf{p}}_i$ represents the estimated probability distribution for some instance i,
- $\mathbb{P}(\cdot)$ is the probability of $\cdot$,
- $\mathbb{E}(\cdot)$ is the expectation of $\cdot$,
- $\mathbb{I}(\cdot)$ represents the indicator function, which equals one if the condition $\cdot$ is true, and zero otherwise,

- $\mathcal{L}(\mathbf{y}, \hat{\mathbf{p}})$ represents a loss function (i.e. a function that maps the estimated probabilities and true labels to a scalar value where a lower value is indicative of more accurate predictions).

Unless stated otherwise, scalar operations written with vector arguments can be assumed to be applied elementwise e.g.

$$\log \hat{\mathbf{y}}_i = [\log \hat{\mathbf{y}}_i^{(1)}, \log \hat{\mathbf{y}}_i^{(2)}, \dots].$$

1.4 Framework for Experimental Evaluation

The experiments in this thesis generally follow a common form: individual (or ensembles of) nested dichotomies are trained with different binary base learners, and evaluated using cross-validation. Statistical significance tests are performed where possible. All of the base learners—divided into ordinary base learners and ensemble learners—and evaluation methods used in the experiments are described below.

1.4.1 Base Learners

We use three main widely-used non-ensemble base learners in this thesis: decision trees, logistic regression and naïve Bayes. Decision trees (Breiman et al., 1984; Quinlan, 1993) and logistic regression (Nelder and Wedderburn, 1972) are included because they occupy opposite ends of the bias-variance spectrum. Naïve Bayes (Lewis and Gale, 1994) is used as an example of a poorly calibrated classifier while investigating the effect of probability calibration (discussed in-depth in Section 2.3) on nested dichotomies. These methods are introduced below.

Logistic Regression

Logistic regression is a linear discriminative model that has seen widespread use across many industries. When there are two output classes, a weight vector $\mathbf{w}$ and bias b are learned and used in the following

equation to estimate the probability that an example $(\mathbf{x}, y)$, which consists of a feature vector $\mathbf{x}$ and a class label $y \in 0, 1$, belongs to class 1:

$$\mathbb{P}(y = 1|\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{w} \cdot \mathbf{x} + b) \tag{1.1}$$

$$= \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}. \tag{1.2}$$

This can be generalised to a multiclass output by learning a weights matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ instead. The weights are usually optimised through gradient-based methods like stochastic gradient descent (Kiefer, Wolfowitz, et al., 1952) and quasi-Newton methods such as L-BFGS (Byrd et al., 1995).

While often used as a standalone model, logistic regression is also commonly found as the final layer in neural networks. In this context, it is usually referred to as a *softmax* layer (Goodfellow et al., 2016).

Decision Trees

Decision trees have a rich history in machine learning (Breiman et al., 1984; Quinlan, 1993). In a decision tree, the input space is divided into regions in a hierarchical manner. In the simplest case, when the task is to perform classification with the tree, the goal is to find regions that each contain only one class of examples. Typically, these regions are defined by axis-parallel splits. The attribute to split on, and the location of the split, is usually chosen to maximise the information gain (Quinlan, 1993) or minimise the Gini impurity (Breiman et al., 1984).

Nowadays, decision trees are often used in conjunction with ensemble methods to maximise predictive performance. Random forests (Breiman, 2001) and boosted trees (Freund and Schapire, 1997; Chen and Guestrin, 2016; Ke et al., 2017; Dorogush, Ershov, and Gulin, 2018) are ubiquitous in data science applications, with gradient boosted trees in particular often winning data mining competitions on websites like Kaggle.[4] Nevertheless, it is not uncommon to see a single decision tree used in practical

[4] `https://www.kaggle.com`

applications (Kate and Nadig, 2017; López et al., 2017; Yang et al., 2018).

An advantage of using a decision tree is that, providing the model is of a reasonable size, its predictions are usually relatively easy to interpret. Given a prediction, it is straightforward to trace back along the path in the tree to see which attributes influenced the decision. As machine learning models are utilised more and more to make decisions about people, interpretability is important to ensure that models are working the way we want them to.

Naïve Bayes

Naïve Bayesian classifiers (Lewis and Gale, 1994) are simple generative models that apply Bayes' rule (Bayes, Price, and Canton, 1763) to make predictions. Naïve Bayes is named "naïve" because it assumes class-conditional statistical independence between the features. Despite its simplicity, naïve Bayes often performs competitively with decision trees and other more complex methods. A nice quality of naïve Bayes is that the maximum likelihood solution can be obtained by evaluating a closed-form expression, which can be computed in linear time. Naïve Bayes is commonly used as a simple (yet effective) baseline for text categorisation (Rennie et al., 2003).

When using a naïve Bayesian model, one needs to choose an appropriate type of probability distribution for which the data from each class is assumed to be drawn from. When dealing with continuous data, usually a Gaussian distribution is used. In the case of categorical data, a Bernoulli distribution can be used.[5] Finally, for frequency counts (like bag-of-words), a multinomial distribution is often applied.

When used with discrete data, such as frequency data and data consisting of indicator variables, naïve Bayes has an interesting relationship with logistic regression in that it forms a *generative-discriminative pair* (Ng and Jordan, 2002). In other words, naïve Bayes can be considered a way of fitting a probabilistic model that optimises the joint likelihood $\mathbb{P}(y, \mathbf{x})$

[5]Note that categorical data needs to be converted to binary indicator variables first.

while logistic regression fits the same model to optimise the conditional likelihood $\mathbb{P}(y|\mathbf{x})$.

Even though the classification accuracy of naïve Bayesian classifiers tends to be surprisingly high, the class probability estimates that are produced are known to be of poor quality because of the assumption of conditional independence (Zadrozny and Elkan, 2001b; Niculescu-Mizil and Caruana, 2005b).

1.4.2 Ensemble Methods

Several ensemble methods are used throughout the thesis, as base learners to solve the binary classification problems in nested dichotomies, by forming ensembles of decision trees: bagging, AdaBoost, and MultiBoost. Additionally, these ensemble learners are used to combine nested dichotomies into ensembles of nested dichotomies. Each of the methods used are described below.

Bagging

Bagging, shorthand for **b**ootstrap **agg**regat**ing**, is a technique for reducing the variance in a classifier's predictions by creating an ensemble (Breiman, 1996). Put simply, bagging produces a number of resampled training sets by sampling with replacement (also known as bootstrapping), and builds one classification model, such as a decision tree, from each of the resampled sets.

A convenient feature of bagging is the availability of unbiased data to test each ensemble member. Because each model is trained on a resampled training set, where each set was produced by sampling with replacement, it is almost certain that some of the examples in the original training set are not included in the resampled version, and so are not used to train the corresponding ensemble member. In practice, the so-called *out-of-bag* data for each ensemble member is approximately 37% of

the original training set.[6] As a result, performance estimates can be obtained by evaluating predictive error on this data, without the need for a validation set to avoid overly optimistic performance estimates.

Bagging is usually used for decision trees, although in theory it can be applied to any learning method. An especially common use of bagging is with randomised decision trees as the base learners, referred to in the literature as random forests (Breiman, 2001). Empirically, bagging has been shown to provide good performance gains for decision trees, neural networks and attribute-selected linear regression, but to slightly degrade the performance of very stable classifiers like k-nearest neighbours (Breiman, 1996). In bagging, a probability distribution can be obtained by averaging the probability distributions of each ensemble member, or by simply voting amongst the members.

AdaBoost

AdaBoost, shorthand for **ada**ptive **boost**ing, is a method for combining many so-called *weak learners* to form one classifier with high predictive performance (Freund and Schapire, 1997). AdaBoost is effective at reducing the bias and variance of the weak learners (Bauer and Kohavi, 1999). Intuitively, the algorithm can be described as iteratively training a set of models, with each new model attempting to correct the errors of the current set of models. Training examples that have high error are weighted more heavily in the next iteration. Final predictions are obtained through a weighted average of the predictions from the weak learners. It has been proven that as long as the weak learner is marginally better than random guessing, the performance of the final model will converge to a strong learner (Freund and Schapire, 1997).

In each iteration, the exponential loss function

$$\mathcal{L}_{exp}(y, \hat{y}) = e^{-y\hat{y}}, \tag{1.3}$$

[6] In fact, the expected proportion is $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n = \frac{1}{e} = 0.3678$ (Efron and Tibshirani, 1997).

where y is the observed class label in the training data and $\hat{y}$ is the model's prediction, is used to determine the error (and the subsequent weighting) for each example. A new weak learner k is then trained to minimise the weighted error ϵ_k. This weighted error is used to determine the weighting of the new weak learner α_k in the final prediction:

$$\alpha_k = \frac{1}{2} \ln \left(\frac{1 - \epsilon_k}{\epsilon_k} \right). \tag{1.4}$$

AdaBoost, as originally proposed, was primarily designed for binary problems, but multiclass variants have been proposed (Freund and Schapire, 1996a; Schapire and Singer, 1999; Hastie, Rosset, et al., 2009). In this thesis, we are mainly concerned with these multiclass versions as we are typically boosting nested dichotomies. Where experiments take place in WEKA, AdaBoost.M1[7] (Freund and Schapire, 1996a) is used, and where experiments are performed in Python, SAMME.R (Hastie, Rosset, et al., 2009) is used.

MultiBoost

MultiBoost is a technique for combining aspects of weighted bagging (wagging) and AdaBoost (Webb, 2000). Wagging is a modification of bagging in which the datasets are re-weighted by sampling weights from a continuous Poisson distribution rather than resampled. Combining it with AdaBoost, the bias and variance reduction of AdaBoost and variance reduction of wagging are both achieved simultaneously for higher predictive power.

Put simply, MultiBoost can be described as wagging ensembles produced by AdaBoost. In other words, AdaBoost is used as the base learner for wagging.[8] In order to have a simple interface where the number of base learners (i.e., the base learners for AdaBoost) can be specified, the

[7] AdaBoost.M1 is designed for binary classes, but it works for multi-class data if the base learner is sufficiently strong.

[8] This is not dissimilar to the approach from (Pfahringer, 2000) to win the KDD Cup in 1999.

number of AdaBoost ensembles and their respective sizes are set as the square root of the number of base learners in the overall MultiBoost ensemble by default. However, each AdaBoost ensemble can be terminated early if the error is too great or zero, resulting in one extra ensemble being added at the end to fill out the full number of base learners specified.

1.4.3 Performance Estimation and Comparison

Rigorous evaluation of newly proposed methods is important to establish their usefulness. Below, the methods used to evaluate the predictive performance of the proposed methods are described.

Cross-validation

Cross-validation is a method of using all of the available data for training and testing (Kohavi et al., 1995), and is especially useful when working on smaller datasets, or datasets with small numbers of examples for certain classes, where taking a single representative test set may not be feasible. It splits the full dataset into a number of disjoint "folds". For the sake of illustration, assume there are 10 folds. Then, nine of the folds are grouped together to form one training set, and the remaining fold is used as the test set. This process is repeated 10 times, so that each of the folds is used as the test set for one run and as part of the training set for the remaining nine runs. In the experiments presented in this thesis, stratification is used when determining the folds, which ensures that each fold has approximately the same distribution of class labels as the whole dataset.

For all experiments in this thesis, except where stated otherwise, 10 times 10-fold cross-validation is used to produce performance estimates. This means that the whole process of creating folds happens 10 times, each time with randomly selected folds, and the observed predictive performance on the test sets averaged over the 10 runs.

Establishing Statistical Significance

The corrected resampled t-test (Nadeau and Bengio, 2000) is used to compare one "baseline" method to each comparison model. Sometimes this manifests as a newly proposed method being compared to several methods from the literature, and at other times, a single baseline is compared to several new methods. The corrected resampled t-test is a modified version of the classic t-test, and was proposed to correct for an assumption of t-tests that is violated when considering runs of different machine learning algorithms on the same dataset multiple times (such as during cross-validation). In short, the standard t-test requires each pair of samples to be independent, but several pairs of performance estimates are from repeated runs on the same dataset, i.e., not independent. We use a p-value of 0.05 in our experiments.

Critical difference diagrams (Demšar, 2006) are also shown in this thesis when more than two models are being compared. These are designed to overcome issues with comparing multiple algorithms to each other. Each method is ranked according to its performance on each dataset, and the average ranks are computed. Then, a *critical difference* is computed, where each pair of algorithms that has a difference in average rank less than the critical difference is deemed statistically indistinguishable at some p-value. The critical difference is a function of the desired p-value, number of methods being compared, and the number of datasets evaluated. A p-value of 0.05 is again used for these comparisons throughout the thesis.

1.5 Thesis Structure

After discussing background and related work, the research presented in this thesis is split into two main parts, pertaining to the two hypotheses. Chapters 3 and 4 cover methods for choosing the structure of nested dichotomies in a way that non-deterministically favours simpler binary

problems without *a priori* knowledge of semantic structure or hierarchy in the label space. Chapter 5 discusses methods of maximising performance of nested dichotomies once a structure has been chosen, by applying probability calibration methods.

Chapter 2. This chapter covers necessary background and related work. Multiclass transformation methods, including hierarchical and non-hierarchical methods, and probability calibration schemes are covered.

Chapter 3. This chapter introduces a semi-random method, called the random-pair method, for producing nested dichotomies in a way that generally results in easily separable splits at each internal node. The performance of nested dichotomies produced by this method is compared to other methods for constructing nested dichotomies in individual model and ensemble settings.

Chapter 4. This chapter describes a simple extension to randomised top-down subset selection methods that can improve the overall predictive performance of nested dichotomies at the cost of a constant factor of computation. The efficacy of this method is shown for two subset selection methods in a variety of classification settings.

Chapter 5. This chapter discusses the calibration of the probability estimates produced by nested dichotomies. It is shown that for large multiclass tasks, the estimated probabilities are systematically under-confident. Furthermore, experiments are performed to investigate calibrating the internal models of nested dichotomies, in some cases yielding substantial performance improvements.

Chapter 6. In this chapter, we conclude the thesis and summarise the findings from the experiments undertaken.

Chapter 2

Background & Related Work

This chapter presents some background and related work for the research presented in this thesis. We review multiclass transformation methods—methods for transforming multiclass classification problems into a collection of (usually binary) sub-problems—in Section 2.1. These approaches are split into two groups: non-hierarchical methods (Section 2.1.1) and hierarchical methods (Section 2.1.2). Then, we discuss one of these methods, nested dichotomies, which are the focus of this thesis, in-depth in Section 2.2, covering theoretical properties and methods for construction that can be found in the literature. Finally, Section 2.3 discusses approaches to probability calibration, which can be used to calibrate the component models in nested dichotomies.

2.1 Multiclass Transformation Methods

Some models, like decision trees and neural networks, can natively handle multiclass ($m > 2$) problems, but some other models such as support vector machines can only be used on binary-class ($m = 2$) datasets. There are several existing techniques to decompose a multiclass problem into a collection of binary problems, so that binary models can be applied. Note that performing such a decomposition can even be beneficial to predictive performance when using a multiclass classifier (Mayoraz and Moreira, 1997; Fürnkranz, 2002; Knerr, Personnaz, and Dreyfus, 1992; Pimenta

and Gama, 2005; Mohr, Wever, and Hüllermeier, 2018b). In this section, we briefly describe some of the most prominent and relevant multiclass transformation methods.

2.1.1 Non-Hierarchical

The most well-known reduction methods are non-hierarchical in nature. They build K classifiers that operate independently, rather than using the output of one classifier to influence the predictions of another.

One-vs-rest

One-vs-rest is a simple method that builds a set H of m binary models $h_1 \dots h_m$ for an m-class problem (Rifkin and Klautau, 2004). The goal of each binary model h_k is to classify whether an instance belongs to class k or not. For each model h_k, the dataset is transformed such that examples belonging to class k are assigned a meta-label $\tilde{y}_{ki} = 1$, while all other examples are assigned $\tilde{y}_{ki} = 0$. The original class labels are removed. A predicted probability distribution $\hat{\mathbf{p}}_i$ can be produced by normalising the vector of output scores from each h_k

$$\hat{\mathbf{p}}_i = \frac{1}{Z}\Big[\ \mathbb{P}(\tilde{y}_{1i} = 1|h_1, \mathbf{x}_i),\ \dots,\ \mathbb{P}(\tilde{y}_{mi} = 1|h_m, \mathbf{x}_i)\ \Big] \tag{2.1}$$

where Z is the normalising constant ensuring that $\hat{\mathbf{p}}_i$ is a probability distribution. One-vs-rest scales linearly with the number of classes m.

One-vs-rest is a simple and intuitive method, but it is not without problems. One issue is that the binary problems produced by this transformation tend to have highly imbalanced training sets because all but one class is used as the set of negative examples—a problem that is exacerbated as the number of classes gets larger. Furthermore, the scale of output scores in the un-normalised vector above may differ between each model due to poor calibration (discussed further in Section 2.3).

It should be mentioned that the one-vs-rest technique is also known as the binary relevance method in multi-label classification settings (Boutell

et al., 2004). In multi-label classification, the vector of output scores is not normalised, and the potential prediction of multiple classes is desired, rather than a problem to overcome.

One-vs-one

One-vs-one (Friedman, 1996), also known as pairwise classification and round robin classification (Fürnkranz, 2002), overcomes some of the issues presented by one-vs-rest, at the cost of an increased number of models. Instead of creating one binary model for each class against all other classes, one binary model is made for each pair of classes. In total, $m(m-1)/2$ binary models are trained, meaning that the number of models required scales quadratically with the number of classes m, increasing the time required to obtain predictions and space requirements compared to one-vs-rest. However, each model is trained on a smaller subset of the training data (only two classes), resulting in a comparable training time (Fürnkranz, 2002).[1] An advantage of this method that each binary problem will be balanced if the original multiclass problem is balanced.

Obtaining probability estimates from a collection of one-vs-one models is not as straightforward as with one-vs-rest. Naïvely, one can simply take votes from each pairwise model, and normalise the resulting vector to produce a probability distribution

$$\hat{\mathbf{p}}_i = \frac{1}{Z}\Big[\sum_{h \in H_1} \mathbb{I}(\hat{y}_{ih} = 1),\ \ldots,\ \sum_{h \in H_m} \mathbb{I}(\hat{y}_{ih} = m) \Big] \tag{2.2}$$

where Z is a normalising constant and H_k denotes the set of pairwise classifiers for which one of the classes considered is k. Several methods have been proposed for computing more accurate probability estimates (Refregier and Vallet, 1991; Price et al., 1995; Hastie, Tibshirani, et al., 1998). The most widely adopted method is that proposed by Hastie, Tibshirani,

[1] In fact, the training time is reduced if the training complexity of the model is superlinear in the number of examples.

et al. (1998). It aims to find $\hat{\mathbf{p}}_i$ such that the weighted Kullback-Leibler divergence (Cover and Thomas, 2012) between $\boldsymbol{\mu}_i$ and $\mathbf{r}_i$ is minimised, where

$$\mathbf{r}_i^{(k_1,k_2)} = \mathbb{P}(y_i = k_1 | y_i \in \{k_1, k_2\}) \tag{2.3}$$

$$\boldsymbol{\mu}_i^{(k_1,k_2)} = \frac{\hat{\mathbf{p}}_i^{(k_1)}}{\hat{\mathbf{p}}_i^{(k_1)} + \hat{\mathbf{p}}_i^{(k_2)}}. \tag{2.4}$$

This technique is called pairwise coupling and can produce smoother decision boundaries than voting schemes. Some alternative formulations have been made, such as that of Wu, Lin, and Weng (2004):

$$\begin{aligned} \min_{\hat{\mathbf{p}}} \quad & \sum_{k_1,k_2 \in \mathcal{C}} (\mathbf{r}_i^{(k_1,k_2)} \hat{\mathbf{p}}_i^{(k_2)} - \mathbf{r}_i^{(k_2,k_1)} \hat{\mathbf{p}}_i^{(k_1)})^2 \\ \text{such that} \quad & \sum_{k \in \mathcal{C}} \hat{\mathbf{p}}_i^{(k)} = 1,\ \hat{\mathbf{p}}_i^{(k)} \geq 0, \forall i. \end{aligned} \tag{2.5}$$

Wu, Lin, and Weng argue that this is an improved version of the coupling approach by Refregier and Vallet (1991) and show that it is simple to optimise as it is reducible to a simple linear system.

Error-correcting Output Codes

The use of error-correcting output codes (Dietterich and Bakiri, 1995) is another well-known method for combining the results of several binary classifiers to produce a multiclass prediction. A codeword matrix $\tilde{\mathbf{Y}}$ of size $(K, |\mathcal{Y}|)$ is defined, in which each row $\tilde{\mathbf{y}}^{(y)} = \tilde{\mathbf{Y}}^{(y,\cdot)}$ is a binary vector that uniquely represents each multiclass label y. An example of such a matrix is shown in Table 2.1, where $K = 7$. Each instance $(\mathbf{x}_i, y_i)$ then has a new label vector $\tilde{\mathbf{y}}_i = \tilde{\mathbf{Y}}^{(y_i,\cdot)}$ from this mapping. A collection of K binary classifiers $H = \{h_1, \ldots, h_K\}$ is trained, where the binary labels for the training set for $h_k \in H$ are obtained from the value of $\tilde{\mathbf{Y}}^{(\cdot,k)}$ for each training instance $\mathbf{x}_i$. In other words, the labels for each binary problem h_k are taken from column $\tilde{\mathbf{Y}}^{(\cdot,k)}$. The final prediction is given as the label $\hat{y}$ corresponding to the codeword $\tilde{\mathbf{y}}$ with the lowest Hamming distance to

Table 2.1: An example of exhaustive error-correcting output codes for a 4-class problem.

y	Codeword						
1	1	1	1	1	1	1	1
2	0	0	0	0	1	1	1
3	0	0	1	1	0	0	1
4	0	1	0	1	0	1	0

the vector of binary predictions. Hamming distance d_H between two binary vectors is defined as the number of positions where corresponding bits are different:

$$d_H(\tilde{\mathbf{y}}, \hat{\mathbf{y}}_i) = \sum_{k=1}^{l} \mathbb{I}(\tilde{\mathbf{y}}^{(k)} \neq \hat{\mathbf{y}}_i^{(k)}). \tag{2.6}$$

In this manner, it is likely that the correct label is predicted, even with several of the binary classifiers making errors. If the minimum Hamming distance between any pair of binary code-words $d_H(\tilde{\mathbf{y}}^{(k_1,\cdot)}, \tilde{\mathbf{y}}^{(k_2,\cdot)}) = \delta$, the error-correcting procedure can correct at least $\left\lfloor \frac{\delta-1}{2} \right\rfloor$ single bit errors (Dietterich and Bakiri, 1995).

For small numbers of classes, exhaustive codes can be used. An example of exhaustive codes is shown in Table 2.1. However, this quickly becomes infeasible, as the number of bits (and therefore binary classifiers) K required for an m-class problem is $2^{m-1}-1$. A simple method to overcome this issue is to use random codes, which has been shown theoretically to match or exceed the performance of exhaustive codes (James and Hastie, 1998); however, in practice, the code length and number of classifiers required to achieve this performance is too expensive, and using shorter codes results in performance drops (Ghani, 2000).

Unified Approach

Allwein, Schapire, and Singer (2000) combine aspects of one-vs-one, one-vs-rest, and error correcting output codes in a unified approach, and

prove its effectiveness for margin classifiers. In this approach, a *coding matrix* $\mathbf{M} \in \{-1, 0, 1\}^{m \times K}$ is constructed. K classifiers are trained, and their predictions are compared with the values in $\mathbf{M}$ to find the closest code, much like in the standard method of error-correcting output codes. Negative and positive classes are denoted with -1 and 1 respectively while a 0 indicates a "don't care" entry.

Allwein, Schapire, and Singer argue that all three previous approaches mentioned can be considered special cases of this. In one-vs-rest, $\mathbf{M}$ is an $m \times m$ matrix with diagonal elements $+1$ and non-diagonal elements -1. One-vs-one has a coding matrix of size $m \times \binom{m}{2}$, with each column corresponding to a distinct pair of classes. Each column has $+1$ in the row for the first class, -1 in the row for the second class, and zeros for the rest. Finally, error-correcting output codes are easily defined by a coding matrix containing only non-zero elements.

The authors also experiment with other distance functions than Hamming distance. These can take the magnitude of the prediction into account, as this magnitude is considered an indication of classifier confidence:

$$d_{\mathcal{L}}(\tilde{\mathbf{y}}, \hat{\mathbf{y}}_i) = \sum_{k=1}^{l} \mathcal{L}(\tilde{\mathbf{y}}^{(k)}, \hat{\mathbf{y}}_i^{(k)}) \tag{2.7}$$

where $\tilde{\mathbf{y}}$ is a row of $\mathbf{M}$, and $\mathcal{L}(\cdot, \cdot)$ is a scalar loss function such as the exponential loss

$$\mathcal{L}_{exp}(y, \hat{y}) = e^{-y\hat{y}}. \tag{2.8}$$

In their approach, the coding matrix can be randomly initialised, or specified in a complete fashion like with error-correcting output codes. Their experiments show that for support vector machines, these coding schemes almost always perform better than one-vs-rest and that when AdaBoost (Freund and Schapire, 1997) is used to construct an ensemble classifier, there is no clear winner amongst the schemes, the best scheme

being problem dependent.

2.1.2 Hierarchical Multiclass Transformation Methods

There are also many methods in the literature for reducing a multiclass classification problem into a series of binary problems in a hierarchical fashion. Nested dichotomies, the method we focus on in this thesis, fall into this category. In this section, we briefly discuss related hierarchical approaches, before diving into nested dichotomies in-depth in Section 2.2.

Filter Trees

Beygelzimer, Langford, and Ravikumar (2007) propose a multiclass decomposition technique called filter trees, in which a tree structure is randomly generated, and the binary classifier in each internal node is trained in a bottom-up fashion. Filter tree can be considered as a single-elimination tournament on the set of labels. Starting at the parents of the leaf nodes, a binary classifier is trained to distinguish between the two classes corresponding to the leaf nodes. Then, working up to the root node, the binary classifier h_k learned at each node k is trained on data that was correctly classified by the binary classifiers in the child nodes of k. In this way, specific training examples are *filtered out* of the training sets used to produce the levels of the tree nearer the root.

Filter trees are provably consistent, i.e., if the binary models are optimal, then the filter tree is also optimal. This comes down to the way they are trained: the approach of filtering misclassified examples ensures that a greedy tree search provides consistent predictions (Beygelzimer, Langford, and Ravikumar, 2007). An attractive feature of filter trees is that it is simple to modify the training algorithm for cost-sensitive classification problems, for which consistency has also been proven.

Conditional Probability Trees

Beygelzimer, Langford, Lifshits, et al. (2009) also propose conditional probability trees, which are similar to nested dichotomies. They are built in an online fashion and discover the classes during training. When a new class is encountered, the tree structure is amended by splitting an existing leaf node. In this way, the order in which training examples arrive can have a large impact on the overall tree structure.

Choosing the leaf node to be split for a new label can present a dilemma: do we prefer a balanced tree, or a tree with greater predictive performance? Beygelzimer, Langford, Lifshits, et al. (2009) approach this issue by introducing a hyperparameter $\alpha \in [0, 1]$ that encodes this preference. $\alpha = 1$ implies that the child node chosen at some internal node is given by the side that has the fewest labels associated with it, which results in perfectly balanced trees while $\alpha = 0$ chooses the child node that the internal model favours for this example. Values of α between 0 and 1 can be chosen, representing a weighting between the two extremes.

Label Trees

Bengio, Weston, and Grangier (2010) propose a method for splitting the label space hierarchically that yields so-called label embedding trees. Binary logistic regression models at each internal node are optimised jointly to minimise the loss of the entire tree while the structure of the label tree is learned by applying spectral clustering algorithms (Ng, Jordan, and Weiss, 2002) to the symmetrised confusion matrix

$$\mathbf{A} = \frac{1}{2}(\bar{\mathbf{C}} + \bar{\mathbf{C}}^{\top}) \tag{2.9}$$

obtained from a series of one-vs-rest classifiers, which must be trained before the label tree is constructed. Bengio, Weston, and Grangier (2010) also propose a method for learning to embed the labels into a lower dimensional shared representation, where semantically similar classes have

similar label embeddings. They show that by adopting label embeddings in conjunction with a label tree structure, large speed-ups can be achieved when obtaining predictions on test data compared to one-vs-rest, and that predictive performance can be improved in some cases.

Deng et al. (2011) take this approach further by learning the structure and the binary models simultaneously. A two-step optimisation procedure is used to grow each node: first, the class partitioning is fixed and the binary classifier weights are optimised; then, the weights are fixed and the class partitioning is optimised. These two-steps are repeated several times to optimise the node. Under this scheme, overlapping subsets are allowed. Joint optimisation of tree structure and binary models allows for faster training time, as previously, a collection of one-vs-rest classifiers had to be trained to estimate the structure.

LOMTrees

Choromanska and Langford (2015) propose logarithmic-time online multiclass trees (LOMTrees) in order to achieve logarithmic training and testing time per example in an online setting. They devise a boosting algorithm for constructing decision trees with $O(m)$ nodes and $O(\log m)$ depth by generalising a theorem for boosting with decision trees in a binary setting (Kearns and Mansour, 1999) to a multiclass one (Choromanska, Choromanski, and Bojarski, 2016). The tree is built by maximising a new splitting criterion, defined as

$$J(h) = 2\sum_{k=1}^{m} \frac{\sum_{y\in\mathbf{Y}} \mathbb{I}(y=k)}{n} \; |\mathbb{P}(h(\mathbf{X})>0) - \mathbb{P}(h(\mathbf{X})>0|k)| \qquad (2.10)$$

where $h : \mathcal{X} \to \{-1, 1\}$ is some splitting function from a binary classifier, $\mathbf{X}, \mathbf{Y}$ are the set of examples arriving at the node, $\mathbb{P}(h(\mathbf{X}) > 0)$ and $\mathbb{P}(h(\mathbf{X}) > 0|k)$ are the proportion of examples $\mathbf{x}_i \in \mathbf{X}$ reaching the node for which $h(\mathbf{x}_i) > 0$, for all examples at the node and examples belonging to class k respectively. The goal of this objective function is to create splits where all elements of a class are classified in the same way by

h; typical decision tree splitting objectives such as information gain and Gini impurity are not suitable to be used in an online fashion according to Choromanska and Langford (2015).

The training procedure of LOMTrees is an online procedure, where leaf nodes of the tree can turn into a split node as more classes are observed. This process ends when the number of split nodes reaches a threshold T. However, some nodes created early in the construction process may turn out to be of low value. To combat this, Choromanska and Langford (2015) describe a node swapping algorithm that allows such split nodes to be *recycled* elsewhere in the tree.

Empirically, LOMTrees have substantially better test set performance than other logarithmic time approaches such as filter trees (Beygelzimer, Langford, and Ravikumar, 2007) while retaining the low time required for obtaining predictions per example.

Recall Trees

Daumé et al. (2017) proposed an online multiclass transformation technique that splits the class set recursively until a handful of classes remain; then, one-vs-rest is performed in this reduced set of classes (the authors refer to the overall approach as one-against-some). The aim of the so-called *recall tree* is to maximise the recall of the reduced candidate set of classes and then use a one-against-some structure with high precision to complete the classification.

A major advantage of this technique is the low memory requirement. Recall trees only take twice as much memory as one-vs-rest, compared to other online logarithmic time approaches such as LOMTrees (Choromanska and Langford, 2015) that can use 64 times as much memory. In spite of this, they usually yield higher predictive accuracy than LOMTrees; however, they are slightly more computationally expensive at test time (Daumé et al., 2017).

Hierarchical Softmax

The idea of recursively splitting the label space has also been explored in the context of natural language processing with neural networks under the name hierarchical softmax (Morin and Bengio, 2005). In the originally proposed hierarchical softmax, a tree structure over the labels is taken from the WordNet lexical database (Miller, 1995). This hierarchy is not represented as a binary tree—often, there are many children per node—so it is first transformed to a binary tree by using binary clustering techniques. This approach greatly reduces the time taken for training and obtaining predictions compared to the standard softmax generally used for classification in neural networks, but accuracy is slightly reduced compared to the traditional flat softmax.

Mnih and Hinton (2009) later proposed to learn the structure of the tree with a data-driven approach rather than using the expert knowledge from WordNet. To begin, a random structure is built and the neural network is trained. Then, a binary mixture of Gaussians model is recursively applied to the learned representation of words from the neural network to create the tree structure, before learning the parameters of the splitting functions in the new tree. Mnih and Hinton (2009) report predictive performance on par with that of flat softmax while retaining the advantage of more efficient training and predictions.

Convolutional Neural Network (CNN) Trees

Wang, Wang, and Wang (2018) propose a framework for image classification that utilises *confusion sets* to hierarchically split up the set of classes. The goal is to learn a new CNN at each node that discriminates between classes that are commonly confused by the parent CNN. The confusion sets are recovered from a variant of a confusion matrix called a softmax confusion matrix. While an entry $\bar{\mathbf{C}}^{(a,b)}$ in a standard confusion matrix is computed as the number of instances of class a that are misclassified as

class b:

$$\bar{\mathbf{C}}^{(a,b)} = \sum_{i \in \mathbf{X}_a} \mathbb{I}(y_i = b) \tag{2.11}$$

where $\mathbf{X}_a$ is the subset of training data belonging to class a, an entry $\bar{\mathbf{C}}_s^{(a,b)}$ in the softmax confusion matrix is computed as the sum of the probability estimates that instances of class a belong to class b:

$$\bar{\mathbf{C}}_s^{(a,b)} = \sum_{i \in \mathbf{X}_a} \hat{\mathbf{p}}_i^{(b)}. \tag{2.12}$$

This gives a robust estimation of the confusion classes from the training data. These classes that are commonly confused are merged together into sets by an agglomerative-clustering-style bottom-up method.

After a CNN has been trained on the full set of classes, and confusion sets have been discovered, the dataset is split and new CNNs are built to distinguish between the members of each confusion set. This process is applied recursively to produce a tree structure. Wang, Wang, and Wang (2018) showed that this can improve the overall performance of CNNs, presenting results on the ImageNet dataset.

Reduction Stumps

Mohr, Wever, and Hüllermeier (2018b) propose a related method for decomposing a multiclass problem into a series of sub-problems called reduction stumps. In a reduction stump, the set of classes is randomly split into two distinct subsets, and a binary classifier is trained to distinguish between these two subsets. Then, a multiclass classifier is trained in each of the two leaf nodes. The goal of this method is to provide a middle ground between native multiclass classifiers and nested dichotomies, focusing on automated machine learning settings (Thornton et al., 2013; Mohr, Wever, and Hüllermeier, 2018a). Note that this is not, strictly speaking, a multiclass transformation method, as it utilises multiclass classifiers in the leaf nodes.

Reduction stumps can be either *homogeneous* (the same model is used in the split node and both leaf nodes), or *heterogeneous* (up to three different models are used). In the heterogeneous case, a set of candidate classifiers is defined, and each combination of these classifiers is iterated over to find the best reduction stump for the particular problem considered. The training cost of heterogeneous reduction stumps is much greater than that of their homogeneous counterparts. The long-term goal is to use ensembles of heterogeneous reduction stumps in an automated machine learning setting, but the authors admit that a heuristic for selecting base learners is required to make this feasible.

Experimental results show that using heterogeneous reduction stumps often provides large performance gains compared to the individual base learners, homogeneous reduction stumps, and even ensembles of homogeneous reduction stumps.

Other Related Work

There are many related and similar pieces of work in the literature, which we now briefly review. Schwenker (2000) and Schwenker and Palm (2001) recursively use k-means clustering (MacQueen et al., 1967) with $k = 2$ to build a tree structure, using clustering to find similar classes that should be grouped together. Takahashi and Abe (2002) use the distances between class centroids, and a predictor that separates classes according to their Malahanobis distance, to build a tree. Lee and Oh (2003) use genetic algorithms (Mitchell, 1998) to produce the class splits, with the intention of maximising a separability measure. Vural and Dy (2004) found that choosing class splits such that that the number of examples in the left and right subtrees of each split are similar can be advantageous for datasets where the classes are not balanced. Lorena and Carvalho (2008; 2010) use a hierarchical clustering procedure to group the classes together according to their similarity. They consider class centroid distances, Fisher's discriminant ratio (Ho and Basu, 2002), and the volume of overlapping regions between classes (Souto et al., 2010).

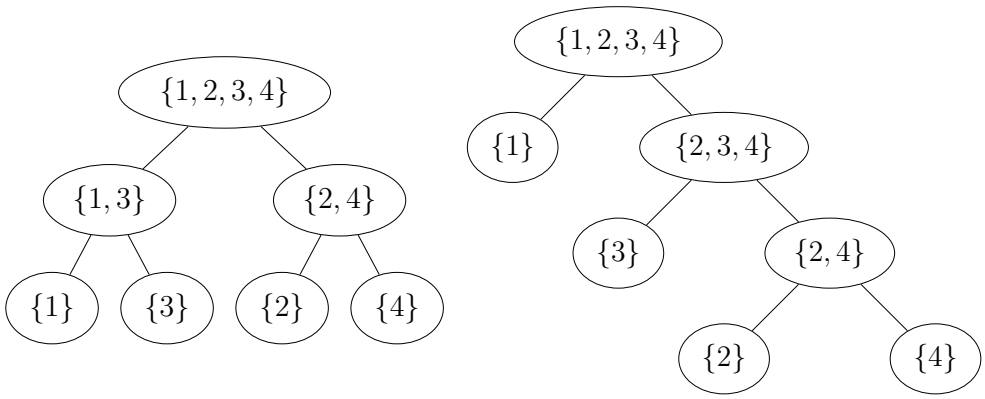


Figure 2.1: Two examples of nested dichotomies for a four class problem.

2.2 Nested Dichotomies

Nested dichotomies (Fox, 1997) are a method of decomposing a multi-class problem into a series of binary problems that is closely related to the hierarchical approaches discussed above. The main difference between nested dichotomies and the other methods discussed in Section 2.1.2 is that nested dichotomies were first considered in an ensemble setting in the machine learning literature (Frank and Kramer, 2004) while most other work in this area focuses on building a single tree. Additionally, many of these other techniques take hard splits to find a single leaf node, rather than producing a probability distribution over the class labels.

In a nested dichotomy $\mathcal{T}$, the class space is recursively split in a tree structure. At each node $k \in \mathcal{T}$, the set of classes $\mathcal{C}_k$ is split into two subsets $\mathcal{C}_l$ and $\mathcal{C}_r$ by some splitting function. A binary classifier h_k is trained at each node, using new labels $\tilde{y}_{ik}$ for the training instances

$$\tilde{y}_{ik} = \begin{cases} 0, & \text{if } y_i \in \mathcal{C}_l \\ 1, & \text{if } y_i \in \mathcal{C}_r. \end{cases} \tag{2.13}$$

Fox (1997) suggests that nested dichotomies should only be used when there is a particular reason to use a specific nested dichotomy structure, e.g., there is a hierarchical structure in the label space. However, they have

been shown to be effective classifiers for randomly constructed nested dichotomies when these are used in an ensemble setting (Frank and Kramer, 2004).

When trained in an ensemble, they have been shown to outperform one-vs-rest, and perform competitively with one-vs-all and error-correcting output codes, depending on the base learners used (Frank and Kramer, 2004; Wever, Mohr, and Hüllermeier, 2018). Originally, an ensemble was constructed by simply sampling nested dichotomies from the space of all nested dichotomies $\mathcal{S}_m$ and building the internal models with the full training dataset. However, it has been empirically shown that using other ensemble techniques like bagging (Breiman, 1996), AdaBoost (Freund and Schapire, 1997) and MultiBoost (Webb, 2000) can provide superior results (Rodríguez, García-Osorio, and Maudes, 2010).

A desirable feature of nested dichotomies is that they provide a very natural way to compute multiclass probability estimates, assuming the binary base learner can produce binary probability estimates. Because the dichotomies are nested, they are conditionally independent (Fox, 1997). This means that to obtain the multiclass probability estimate $\hat{\mathbf{p}}_i^{(c)}$ for an example i and class c, one can simply compute the product of binary probability estimates on the path $\mathcal{P}_c$ from the root node to the leaf node corresponding to c using the chain rule of probability:

$$\hat{\mathbf{p}}_i^{(c)} = \mathbb{P}(y_i = c|\mathbf{x}_i) \tag{2.14}$$

$$= \prod_{k \in \mathcal{P}_c} \big(\mathbb{I}(c \in \mathcal{C}_l)\mathbb{P}(c \in \mathcal{C}_l|\mathbf{x}_i, y_i \in \mathcal{C}_k) + \mathbb{I}(c \in \mathcal{C}_r)\mathbb{P}(c \in \mathcal{C}_r|\mathbf{x}_i, y_i \in \mathcal{C}_k)\big). \tag{2.15}$$

When the number of classes is high, computing a full probability distribution over the entire class set (Algorithm 1) may be too computationally intensive for some practical applications. A naïve approach to finding the most likely class or the top-k classes in a much quicker manner is to simply choose the branch for which the conditional probability is highest,

Algorithm 1 Nested Dichotomy Predictions (Full Distribution)
Input: Nested dichotomy node k, Dataset $\mathbf{D}$

1: **if** k is leaf node **then**
2: Return one-hot predictions of the class representing this leaf node
3: **else**
4: Get probability estimates $\hat{\mathbf{y}}_l$, $\hat{\mathbf{y}}_r$ from left and right children
5: Get probability estimates $\hat{\mathbf{y}}_k$ of $\mathbf{D}$ from internal classifier
6: Return $\hat{\mathbf{y}}_k \times \hat{\mathbf{y}}_l + (1 - \hat{\mathbf{y}}_k) \times \hat{\mathbf{y}}_r$
7: **end if**

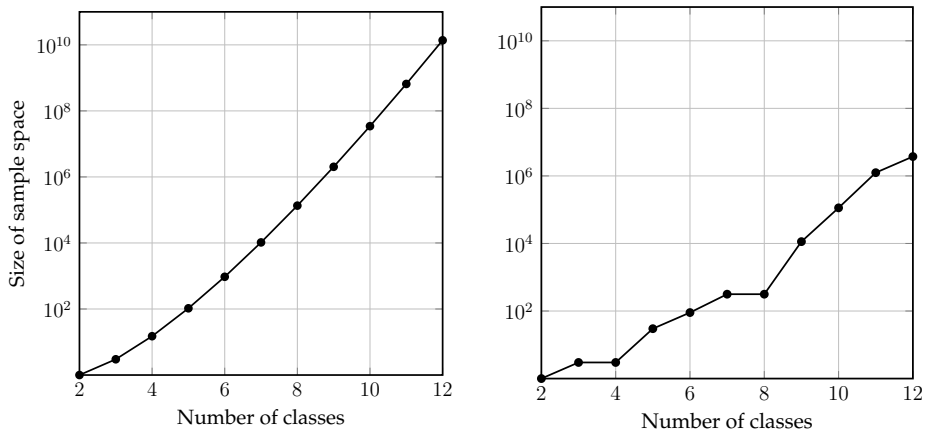
from the root to a leaf node, although this is not guaranteed to find the correct solution (Beygelzimer, Langford, and Ravikumar, 2009). Several methods have been proposed to efficiently estimate top-k classes in nested dichotomies, based on beam search (Kumar et al., 2013) and A* (Mena et al., 2015; Dembczyński et al., 2016).

2.2.1 Subset Selection Methods

Although randomly sampled nested dichotomies often work well, it can be useful to consider other methods for choosing their structure. The structure of nested dichotomies can have a significant impact on predictive performance and training time. One such class of methods is the class of *subset selection methods*, which recursively split $\mathcal{C}_k$, the set of classes present at node $k \in \mathcal{T}$, into two subsets $\mathcal{C}_{kl}, \mathcal{C}_{kr}$ in a top-down fashion. Several subset selection methods have been defined in the literature, each offering specific advantages making them more suitable for different kinds of learning problems. These advantages generally stem from removing a selection of 'bad' nested dichotomies from the sample space $\mathcal{S}_m$, the pool of possible structures to sample from for an m-class problem.

Random Selection

The simplest subset selection method is random selection. Random selection is very simple to implement and is also extremely computationally efficient. Another good quality of random selection is the very large size of



(a) Random Selection **(b)** Class-balanced Selection

Figure 2.2: Growth functions.

the sample space $\mathcal{S}_m$ for an m-class problem. A large sample space is beneficial when constructing an ensemble of nested dichotomies because it encourages a high degree of diversity inside the ensemble, which is known to improve overall predictive performance of ensembles (Kuncheva and Whitaker, 2003).[2]

In random selection, no restrictions are placed on $\mathcal{S}_m$. The number of possible nested dichotomies for an m-class problem is given by the recurrence relation

$$T(m) = (2m - 3) \times T(m - 1) \tag{2.16}$$

where $T(1) = 1$ (Frank and Kramer, 2004). We refer to the function $T(m)$, which relates the number of classes to the size of the sample space of nested dichotomies, as the *growth function* (Figure 2.2a).

Balanced Selection

The use of nested dichotomies is particularly well suited to problems with a large number of classes, due to the potentially logarithmic depth of binary decision trees. A problem with random selection is that it can produce very imbalanced trees, an issue that is more important to consider as the number of classes grows. To combat this, Dong, Frank, and Kramer

[2] Assuming, of course, that the ensemble members are also accurate.

(2005) propose to limit the sample space of nested dichotomies to only those that are balanced, i.e., by enforcing $\mathcal{C}_k$ to be split into two subsets $\mathcal{C}_l$ and $\mathcal{C}_r$ such that

$$\text{abs}(|\mathcal{C}_l| - |\mathcal{C}_r|) \leq 1. \tag{2.17}$$

The main advantage of having class-balanced nested dichotomies is the reduction in training time. Although every nested dichotomy for an m-class problem has the same number of internal nodes, and therefore $(m-1)$ internal models, balanced trees roughly halve the size of the data at each internal node,[3] meaning that the meta-classes used to train the deeper binary models usually contain fewer of the original classes. An additional benefit of class-balanced selection is that it leaves less room for prediction errors to accumulate.

The sample space for class-balanced nested dichotomies $\mathcal{S}_m^{(CB)}$ is a subset of $\mathcal{S}_m$, and it is clear that there are fewer class-balanced nested dichotomies than completely random ones, but how many are there exactly? The growth function for class-balanced nested dichotomies is given by

$$T^{(CB)}(m) = \begin{cases} \frac{1}{2}\binom{m}{m/2} T^{(CB)}(\frac{m}{2}) T^{(CB)}(\frac{m}{2}), & \text{if } m \text{ is even} \\ \binom{m}{(m+1)/2} T^{(CB)}(\frac{m+1}{2}) T^{(CB)}(\frac{m-1}{2}), & \text{if } m \text{ is odd} \end{cases} \tag{2.18}$$

where $T^{(CB)}(1) = T^{(CB)}(2) = 1$ (Dong, Frank, and Kramer, 2005). While this is substantially smaller than $T(m)$ for realistic values of m, there are still sufficiently many possibilities to ensure ensemble diversity (Figure 2.2).

Class-balanced selection works well for datasets that have relatively balanced classes, but the benefits do not apply to the same degree for highly imbalanced datasets. For example, for an imbalanced dataset, $\mathcal{C}_l$ and $\mathcal{C}_r$ may be chosen such that the large classes are put into $\mathcal{C}_l$ and the small classes are put into $\mathcal{C}_r$. This means that the internal binary models

[3] Assuming the classes are roughly balanced.

Algorithm 2 Data-Balanced Subset Selection
Input: Dataset $\mathbf{D}$, set of classes $\mathcal{C}$

Set $\mathcal{C}_1, \mathcal{C}_2 = \emptyset$, $n = |\mathbf{D}|$, $n_1, n_2 = 0$
Randomise the order of $\mathcal{C}$[4]
if $|\mathcal{C}| \neq 1$ **then**
    while ($n_1 < \lfloor \frac{n}{2} \rfloor$) and ($n_2 < \lfloor \frac{n}{2} \rfloor$) **do**
        if $|\mathcal{C}| > 1$ **then**
            Add first element $\mathcal{C}^{(1)}$ of $\mathcal{C}$ to $\mathcal{C}_1$
            Add second element $\mathcal{C}^{(2)}$ of $\mathcal{C}$ to $\mathcal{C}_2$
            Remove $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ from $\mathcal{C}$
            Set $n_1 = n_1 + |\mathbf{D}_{\mathcal{C}^{(1)}}|$, $n_2 = n_2 + |\mathbf{D}_{\mathcal{C}^{(2)}}|$
        end if
    end while
    if $n_1 \geq \lfloor \frac{n}{2} \rfloor$ **then**
        Set $\mathcal{C}_2 = \mathcal{C}_2 \cup \mathcal{C}$
    else
        Set $\mathcal{C}_1 = \mathcal{C}_1 \cup \mathcal{C}$
    end if
end if
Output subsets $\mathcal{C}_1, \mathcal{C}_2$

in the subtree pertaining to $\mathcal{C}_l$ still have the majority of the data, rather than approximately halving the data at each level of the tree. To this end, Dong, Frank, and Kramer (2005) also propose data-balanced nested dichotomies, which are constructed with the classes split such that each subset has approximately the same amount of data. This process is described in Algorithm 2.

Balanced selection was found to have little impact on the accuracy of ensembles of nested dichotomies while significantly reducing the training time (Dong, Frank, and Kramer, 2005).

Selection Based on Clustering

Random selection and balanced selection both operate under the assumption that each potential split is equally likely to be useful *a priori*. However, this is not necessarily the case, as some choices of $\mathcal{C}_{kl}$ and $\mathcal{C}_{kr}$ may be easier for the binary model at node k to classify. Intuitively, the performance of a nested dichotomy is strengthened when each of its internal models is as accurate as possible. This leads to the idea that it is beneficial

to group 'similar' classes together, to create an easier binary classification problem.

As an illustrative example, consider an image classification problem where the classes are cats, dogs, trucks, and cars. This example exhibits a natural hierarchy: cats and dogs can be grouped together as 'mammals', and trucks and cars can be grouped together as 'vehicles'. It is likely much easier for a classifier to perform a coarse classification between mammals and vehicles than a classification between subsets where these groups are broken up. This also means that the binary models deeper in the tree can focus on more fine-grained differences between similar classes.

Duarte-Villaseñor et al. (2012) propose a subset selection method based on clustering to achieve this goal. First, the classes $(c_1, c_2) \in \mathcal{C}_k$ that have the largest pairwise distance, based on some distance metric $d(\cdot, \cdot)$, are selected. Then, the remaining classes are grouped into $\mathcal{C}_l$ and $\mathcal{C}_r$ by

$$\mathcal{C}_l = \{c \in \mathcal{C}_k : d(c, c_1) < d(c, c_2)\} \tag{2.19}$$

$$\mathcal{C}_r = \{c \in \mathcal{C}_k : d(c, c_1) \geq d(c, c_2)\} \tag{2.20}$$

Several distance measures, all based on class centroids, are proposed by Duarte-Villaseñor et al. (2012). Euclidean distance between the two class centroids $\bar{c}_1$ and $\bar{c}_2$ is the simplest distance metric:

$$d(c_1, c_2) = \sqrt{\sum_{i=1}^{d} (\bar{c}_1^{(i)} - \bar{c}_2^{(i)})^2}. \tag{2.21}$$

While effective in many cases, this does not take into account the spread of the data in each class, which can lead to a sub-optimal class split. For example, two classes with distant centroids, but considerably overlapping regions, are difficult to separate. Conversely, two classes with near centroids but no overlapping due to low spread, are easy to separate. To account for these situations, a variant of (2.21) is proposed that incorporates information about the so-called 'radius' r of each class, defined as the distance from the centroid to the furthest example belonging to the

class:

$$d(c_1, c_2) = \frac{\sqrt{\sum_{i=1}^{d} (\bar{c}_1^{(i)} - \bar{c}_2^{(i)})^2}}{r_1 + r_2}. \tag{2.22}$$

Including class radius information brings improved results in some cases, but it is very sensitive to outliers—spurious examples can greatly increase the radius of a class, which may not reflect its true distribution. To combat this effect, a third distance measure is proposed that uses the average distance of each example to its respective class centroid as the radius, rather than the largest distance.

Duarte-Villaseñor et al. (2012) found that applying these techniques produces higher quality nested dichotomies on average than methods that split randomly. However, a downside to this technique is that it is deterministic, which means that it is not effective in a randomization-based ensemble. Nevertheless, ensemble diversity can be achieved when using ensemble methods that resample or re-weight the data, such as bagging and boosting.

2.2.2 Sampling Methods

Another approach to building nested dichotomies is to sample an entire nested dichotomy structure, rather than building a nested dichotomy top-down and choosing each subset by a subset selection method. We briefly review such methods in this section.

Uniform Sampling

Uniform sampling is the method used in the original publication of ensembles of nested dichotomies (Frank and Kramer, 2004). In this scheme, each possible nested dichotomy $\mathcal{T}$ in the space of nested dichotomies is equally likely to be sampled. The assumption underlying uniform sampling is that each possible nested dichotomy is just as likely to be useful without knowledge of the problem domain.

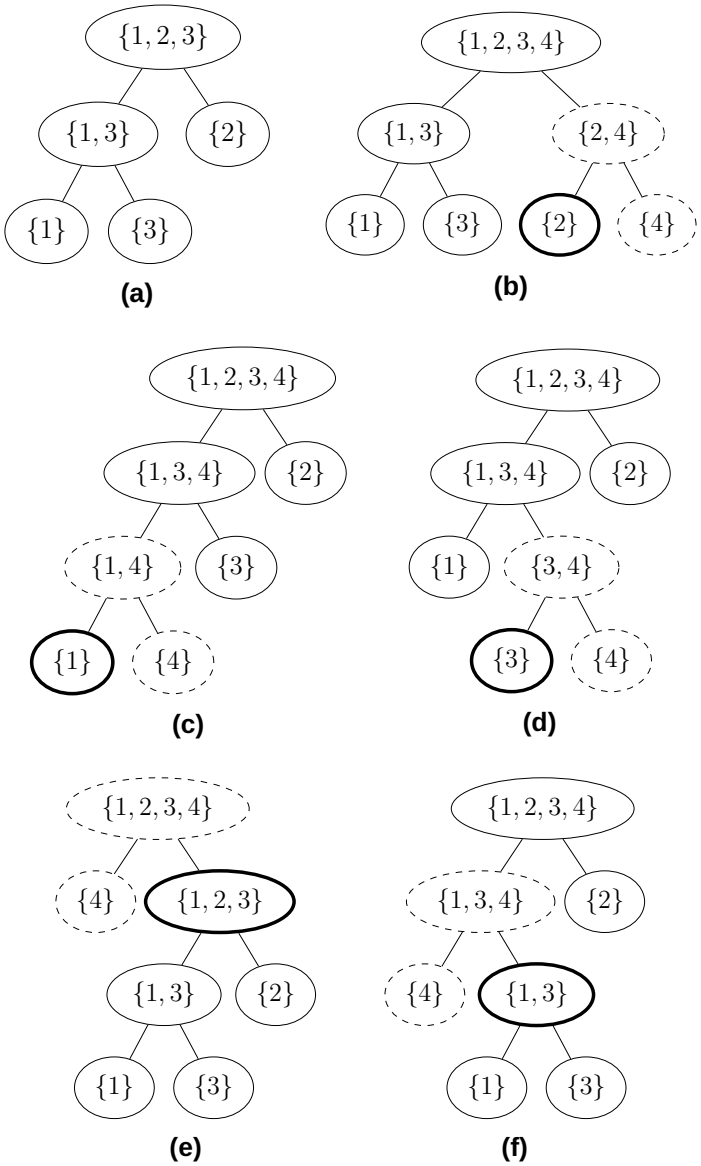


Figure 2.3: **(a)** A nested dichotomy for a three-class problem. **(b)–(f)** All distinct possibilities for resulting structures after adding the fourth class. Newly created nodes are drawn with dashed outlines while the randomly selected node to add the fourth class to is drawn with a thicker outline.

Sampling nested dichotomy structures uniformly is not the same as randomly selecting subsets in a top-down fashion (described in Section 2.2.1), so a specialised algorithm is required.[5] We now describe the procedure used by Frank and Kramer (2004) to ensure that each nested dichotomy structure is equally likely to be sampled. First, the list of classes is shuffled, and the first one is taken for the root node of the tree. Then, each remaining class is added, as a leaf node, to the tree. When a class is added, it is propagated up the tree so that it is included in all class subsets that are in the path to the newly created leaf node.

One step of building a nested dichotomy by this construction process is depicted in Figure 2.3. Figure 2.3a shows a nested dichotomy structure for a three-class problem, and we desire to add a fourth class to it such that each possible option has equal likelihood of selection. When adding a class as a leaf node to a nested dichotomy, the new leaf node is made into a sibling with an existing subtree (selected uniformly at random) and a new parent node is created. If the selected subtree trivially only consists of a single a leaf node (Figures 2.3b–2.3d), then this process can be thought of as turning it into a split node with two leaf nodes as children. When a non-trivial subtree is selected (Figures 2.3e–2.3f), then the subtree is lowered, and a new parent node created with the subtree and the new leaf node as its children. Finally, the newly added class is propagated up the tree to the root node, including it in the set $\mathcal{C}_k$ for each internal node k along the path. Because the node to add the new class to is chosen uniformly at random, each possible nested dichotomy is equally likely, which makes this algorithm sample entire structures uniformly.

Best-of-K models

Best-of-K models provides a simple method for producing higher quality nested dichotomy structures (Melnikov and Hüllermeier, 2018). Simply put, K nested dichotomies are sampled uniformly at random (using a procedure such as described above), and the one that gives the best

[5] This is not immediately obvious, and is discussed further in Section 4.1.1.

performance is selected. The error on a held-out validation set is used to determine the best nested dichotomy. Best-of-K is intended for producing a single nested dichotomy, and was not evaluated in an ensemble setting (Melnikov and Hüllermeier, 2018).

Melnikov and Hüllermeier (2018) propose a metric called the *exceedance probability* for determining the quality of a sampling scheme for nested dichotomies. The exceedance probability of some sampling scheme is the probability that a random nested dichotomy (see Section 2.2.1) has greater performance than a nested dichotomy sampled under the scheme. A sampling scheme with low exceedance probability produces better nested dichotomies on average. Bag-of-10 and bag-of-50 were shown empirically to have lower exceedance probabilities to other proposed heuristics, and to be highly competitive with state-of-the-art methods in terms of classification accuracy (Melnikov and Hüllermeier, 2018). This comes at a high training cost because K nested dichotomies must be constructed. However, the construction algorithm is easily implemented in parallel because the nested dichotomies are independent.

An interesting finding by Melnikov and Hüllermeier (2018) is that nested dichotomies that are class-balanced have a higher empirical exceedance probability than random structures—when a single structure is chosen as well as in a best-of-K setting. This is especially interesting given that Dong, Frank, and Kramer (2005) found that the average performance of class-balanced nested dichotomies is not statistically significantly different from random nested dichotomies, when trained in an ensemble setting.

Evolved Nested Dichotomies

Wever, Mohr, and Hüllermeier (2018) proposed a method for building nested dichotomies with genetic algorithms. Due to the constraints on the structure of nested dichotomies, standard approaches for tree-based genetic operations do not necessarily yield valid nested dichotomy structures. Instead, a different method is proposed for representing a nested

dichotomy structure as a string of pairwise 'distances' between classes. Ordinary genetic operations like crossover and mutation can be performed on this string, from which the nested dichotomy structure can be recovered. Fitness is determined by predictive accuracy on a held-out validation set.

An ensemble of k nested dichotomies can be produced by creating k populations of random nested dichotomies, performing a series of evolutionary runs on each population, and taking the best performing individual from each population. Each population is independent of the others, and so can be evolved in parallel. Wever, Mohr, and Hüllermeier (2018) show experimentally that ensembles of evolved nested dichotomies are competitive with state-of-the-art methods, and perform particularly well compared to other decomposition methods such as uniform sampling, balanced nested dichotomies, nested dichotomies based on clustering, when simple base models, like decision stumps and logistic regression, are used.

2.3 Probability Calibration

In a classification setting, we often desire an estimated probability distribution over the set of classes, rather than a hard classification. This is particularly pertinent when considering nested dichotomies: accurate binary probability estimates are required at the internal nodes as they are multiplied together to produce the elements of the multiclass probability distribution. While there are classifiers that can satisfy this desire, many produce probability estimates that are not well-calibrated—in other words, the estimated probability does not align well with the actual empirical probabilities observed in the data. For a set of examples that have *estimated* probability 0.8 of belonging to some class, we expect approximately 80% of these examples to *actually* belong to this class.

It has been shown that many commonly used classifiers and ensemble techniques exhibit poor probability calibration (Niculescu-Mizil and

Caruana, 2005b). A naïve Bayes model assumes that all of the features are conditionally independent given the class, which results in its probability estimates exhibiting bias towards the extreme ends of the spectrum. Random forests (Breiman, 2001), on the other hand, rarely predict extreme probability values. This is because in order for a random forest to issue probability 0 or 1, every single random tree in the forest must agree on the prediction. Given that tree construction is applied to subsamples of the data, and is generally applied with random feature subsetting, it is likely that some trees will make predictions that do not align with the majority. Boosted decision trees (Freund and Schapire, 1997; Chen and Guestrin, 2016) are amongst the most popular and effective machine learning methods in practice. However, probabilities estimated by these ensembles have been shown to exhibit bias towards the center of the probability distribution (Niculescu-Mizil and Caruana, 2005a).

Models that directly optimise the negative log-likelihood (see Section 2.3.3), like logistic regression and neural networks, tend to be well-calibrated.[6] This makes sense, as the log-loss penalises probabilities that are far from the true labels. Overfitting can occur in practice: recently it has been shown that some modern neural network architectures like residual networks (He et al., 2016) and densely-connected convolutional networks (Huang et al., 2017) are sometimes very poorly calibrated, even though the log-loss is optimised during training and test classification accuracy is high (Guo et al., 2017). As the number of layers and number of filters in convolutional networks increases, the networks become increasingly poorly calibrated. Applying batch normalisation (Ioffe and Szegedy, 2015) has a negative effect on calibration while weight decay (Vapnik, 1998) generally has the opposite effect.[7]

[6] Assuming overfitting is tackled in some manner.

[7] However, it is uncommon to apply heavy weight decay in neural networks nowadays, and many top-performing modern networks have little or no regularisation from weight decay (He et al., 2016; Simonyan and Zisserman, 2015). In fact, it has been shown that weight decay does not have an effect on regularisation when used in conjunction with batch normalisation (Laarhoven, 2017). Weight decay can also result in other strange effects depending on the optimiser used (Loshchilov and Hutter, 2019). Batch normalisation does have a regularising effect on neural networks, although exactly why

2.3.1 Why Calibrate?

In many applications that employ supervised learning to classify data into groups, the prediction that we care about is the most likely class for a given instance, rather than an accurate probability estimate associated with it. Given that probability calibration techniques usually have little effect on the classification accuracy and can even sometimes *reduce* the accuracy slightly[8], in what situations is probability calibration desirable at all?

Cost-sensitive classification. In many practical classification scenarios, there is a cost associated with misclassifications for particular classes (Elkan, 2001; Zadrozny and Elkan, 2001a; Domingos, 1999). A simple example is a model for detecting credit card fraud. It is not a big deal for the company if one of their models produces a false-positive prediction, i.e., the model predicts that a fraudulent transaction has occurred, but it turns out to be a transaction made by the card owner: this can be resolved by a simple phone call to the customer. However, if a false-negative prediction is made, i.e., a customer's details are stolen and the fraudulent transactions are not detected, then the credit card company may be liable to refund the money to the customer. In this kind of scenario with non-uniform misclassification costs, financially speaking, it is best to minimise expected cost, where the expected cost is estimated using the probability estimates obtained from the model. We call this scenario *cost-sensitive classification*. These misclassification costs can be accounted for by minimising the expected cost during classification

$$\mathbb{E}[C_i] = \sum_{c=1}^{m} C(\hat{y}_i = c|y_i)\mathbb{P}(y_i|\mathbf{x}_i) \tag{2.23}$$

and how this occurs is not fully understood by the machine learning community (Luo et al., 2019).

[8]This can be due to overfitting on the calibration set.

where $C(\hat{y}_i = c|y_i)$ is the cost of misclassifying an example $\mathbf{x}_i$ as class c when the true label is y_i. This expected cost relies on the class probabilities that are estimated by the model, so accurate probability estimates are very important to promote good generalisation performance.

Imbalanced classes. When the dataset is imbalanced, i.e., there are unequal numbers of instances for each class in the training data, the probability estimates made by some models may be biased in favour of the majority class (Zadrozny and Elkan, 2001b). This leads to poor performance, as seen with metrics like F_1 (the harmonic mean of precision and recall). A simple method for reducing the impact of dataset imbalance is to undersample the majority class during training (Akbani, Kwek, and Japkowicz, 2004). However, this also reduces the calibration of the estimated probabilities (Dal Pozzolo et al., 2015).

Model stacking. Stacking is an ensemble technique where the predictions of one or more models are used as inputs to a secondary model (Wolpert, 1992). This meta-model (typically a linear model like logistic regression) learns the relationship between the estimated probabilities from the base model (or models) and the true labels. Poorly calibrated probability estimates from the base models may be distributed inconsistently across examples, resulting in poor performance in a stacked ensemble.

Interpretability. Humans have a very natural, innate understanding of probabilities (Cosmides and Tooby, 1996). The most powerful current models, such as neural networks and boosted decision tree ensembles, are difficult to interpret and are generally considered "black boxes", so it is important that they issue accurate probabilities.

2.3.2 Calibration Methods

In this section, we give an overview of some notable probability calibration methods. Each method described is applied as a post-processing step. Common to all calibration methods is the need to train the calibration model on data that was not used to train the model producing the probability estimates, to prevent introducing unwanted bias. Cross-validation, or (more commonly for larger datasets) a held-out calibration set, can be used to achieve this. This calibration set can also be used for hyperparameter optimisation of the base model (Niculescu-Mizil and Caruana, 2005b; Guo et al., 2017).

Platt Scaling

Platt (1999) introduced a method that is now called Platt scaling for scaling the output of support vector machines in order to produce accurate probability estimates. This method fits a sigmoid function

$$\hat{p} = \sigma(\hat{y}) = \frac{1}{1 + e^{\,\alpha\hat{y}+\beta}} \tag{2.24}$$

to 'squash' the support vector machine output $\hat{y}$, which can be any real number, to a probability estimate $\hat{p} \in [0, 1]$. The parameters α and β are fitted using logistic regression to minimise the binary cross-entropy between the calibrated output $\hat{p}$ and the true label $y \in \{\tilde{y}_+, \tilde{y}_-\}$ in a validation set. Rather than using the usual values of $\tilde{y}_+ = 1$ for positive examples and $\tilde{y}_- = 0$ for negative examples, $\tilde{y}_+$ and $\tilde{y}_-$ are defined by Platt as

$$\tilde{y}_+ = \frac{N_+ + 1}{N_+ + 2}, \qquad \tilde{y}_- = \frac{1}{N_- + 2},$$

where N_+ is the number of positive examples and N_- is the number of negative examples. This follows from applying Bayes' rule to a model of out-of-sample data, with a uniform prior over the labels (Platt, 1999). Platt

scaling can only be directly used for two-class problems, but any multiclass transformation technique can be utilised for multiclass datasets.

Despite originally being proposed for use with support vector machine outputs in $\mathbb{R}$, it has been shown that Platt scaling also works well for scaling probability estimates in $[0, 1]$ from other models, such as boosted models and naïve Bayes (Niculescu-Mizil and Caruana, 2005b). Platt scaling is often applied to the log-odds (sometimes referred to as *logits*) of the probabilities rather than the probabilities themselves (e.g., (Guo et al., 2017)). This is because logistic regression assumes a linear relationship between the input data (in this case, uncalibrated probabilities or log-odds), and the log-odds of the output probabilities. The log-odds $\mathbf{z}_i$ for a multiclass probability estimate vector $\hat{\mathbf{p}}_i$ is given as

$$\mathbf{z}_i = \log\left(\frac{\hat{\mathbf{p}}_i}{1 - \hat{\mathbf{p}}_i}\right). \tag{2.25}$$

Vector Scaling & Matrix Scaling

Vector scaling and matrix scaling are proposed as multiclass extensions of Platt scaling (Guo et al., 2017). Instead of fitting only two scalar parameters α and β and computing the sigmoid only on a scalar probability value $\hat{y}_i$, a weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ are learned by fitting the function

$$\hat{\mathbf{p}}_i = \boldsymbol{\sigma}(\hat{\mathbf{y}}_i) = \frac{1}{1 + e^{\,\mathbf{W}\hat{\mathbf{y}}_i + \mathbf{b}}} \tag{2.26}$$

where $\hat{\mathbf{y}}_i$ is the full estimated probability distribution (or its corresponding log-odds representation). Matrix scaling is identical to a standard multiple logistic regression model. It is expensive for datasets with many classes because the number of parameters in the weight matrix $\mathbf{W}$ grows quadratically with the number of classes. Vector scaling is proposed to overcome this. It is a variant where $\mathbf{W}$ is restricted to be a diagonal matrix to achieve computational and memory complexity that is linear in the number of classes.

Temperature Scaling

Another calibration method, temperature scaling, pushes the predictions toward the center of the m-simplex, for an m-class problem. It has traditionally been used for knowledge distillation (Hinton, Vinyals, and Dean, 2015), but has also been considered for calibrating deep neural networks (Guo et al., 2017). Temperature scaling can be thought of as a simplification of Platt scaling, using only a single scalar parameter $\tau > 0$. Given a vector $\mathbf{z}_i$ containing the log-odds predictions for an instance from a neural network, the scaled output is given by

$$\hat{\mathbf{p}}_i = \boldsymbol{\sigma}(\mathbf{z}_i) = \frac{1}{1 + e^{\frac{\mathbf{z}_i}{\tau}}}. \tag{2.27}$$

When $\tau = 1$, the probability estimates are unchanged. As τ approaches 0, the prediction approaches a uniform distribution. An attractive feature of temperature scaling is that it does not affect the accuracy of the final prediction while calibrating the probabilities. This is because it scales each entry in $\mathbf{z}_i$ by the same amount, and therefore the largest entry in the estimated distribution is unchanged. Temperature scaling is equivalent to maximising the entropy of the output probability distribution subject to particular constraints on the log-odds (Guo et al., 2017). As in Platt scaling, τ is optimised with respect to cross-entropy between the calibrated output and the true labels for a validation set.

Histogram Binning

While the sigmoid-based transformation applied in Platt scaling and temperature scaling is suitable for calibration of the outputs of some models, others can benefit from a more general method. Histogram binning (Zadrozny and Elkan, 2001b) is a very simple non-parametric method for probability calibration of binary classification problems. The range of the uncalibrated probability estimates is split into a series of K 'bins' $B_1, \ldots, B_K$, where each bin has an assigned output probability

score θ_k. To obtain calibrated probability estimates for a given classifier output $\hat{p}_i$, one can simply return the output probability score corresponding to the bin that $\hat{p}_i$ falls into.

Typically, the bin widths are chosen to be equally spaced intervals, or such that they have equal numbers of training examples. Then, the output probability scores for each bin are chosen such that they minimise the mean squared error for each bin

$$\min_{\theta_1 \ldots \theta_K} \sum_{k=1}^{K} \sum_{i=1}^{|X|} \mathbb{I}(t_k \leq \hat{p}_i < t_{k+1})(\theta_k - y_i)^2 \tag{2.28}$$

where (t_k, t_{k+1}) are the lower and upper bin thresholds for B_k, and $t_{K+1} = 1$. It turns out that the optimal solution to (2.28) is found when each θ_k is equal to the empirical accuracy for the corresponding bin B_k, i.e., the percentage of positive examples[9] in the calibration set that land in B_k.

Isotonic Regression

Another non-parametric method for probability calibration, based on isotonic regression, has been shown to work well for a variety of classifiers (Zadrozny and Elkan, 2001; 2002). Isotonic regression is a more general method than the sigmoid-based methods (Platt, vector, matrix and temperature scaling) in that no assumptions are made about the function used to map probability estimates to calibrated probabilities other than it must be monotonically increasing.

In practice, a piecewise constant function is used (see Figure 2.4). Such a function that minimises mean squared error can be found in linear time by the pair-adjacent violators algorithm (Ayer et al., 1955), which is described in Algorithm 3. In this way, isotonic regression can be thought of as a special case of histogram binning where the bin boundaries and

[9]The names "positive" and "negative" classes refer to the classes with labels 1 and 0 in a binary classification setting.

Algorithm 3 Pair-adjacent Violators Algorithm
Input: calibration set of pairs $(\hat{y}_i, y_i)$ sorted by $\hat{y}_i$

Initialise $\mathbf{M}^{(i,i)} = y_i$, $\mathbf{W}^{(i,i)} = 1$
while $\exists i : \mathbf{M}^{(k,i-1)} \geq \mathbf{M}^{(i,l)}$ **do**
    Set $\mathbf{W}^{(k,l)} = \mathbf{W}^{(k,i-1)} + \mathbf{W}^{(i,l)}$
    Set $\mathbf{M}^{(k,l)} = (\mathbf{W}^{(k,i-1)}\mathbf{M}^{(k,i-1)} + \mathbf{W}^{(i,l)}\mathbf{M}^{(i,l)})/\mathbf{W}^{(k,l)}$
    Set $\mathbf{M}^{(k,i-1)} = \mathbf{M}^{(k,l)}$
    Set $\mathbf{M}^{(i,l)} = \mathbf{M}^{(k,l)}$
end while
Output stepwise constant function $\mathbb{P}(\hat{y}) = \mathbf{M}^{(i,j)}$ for $\hat{y}_i < \hat{y} \leq \hat{y}_j$

output estimates are optimised jointly, minimising the objective function

$$\begin{aligned} \min_{K;\, \theta_1 \ldots \theta_K;\, t_1 \ldots t_{K+1}} \quad & \sum_{k=1}^{K} \sum_{i=1}^{|X|} \mathbb{I}(t_k \leq \hat{p}_i < t_{k+1})(\theta_k - y_i)^2 \\ \text{such that} \quad & 0 = t_1 \leq \cdots \leq t_{K+1} = 1, \\ & \theta_1 \leq \cdots \leq \theta_K. \end{aligned} \tag{2.29}$$

While isotonic regression is an effective technique for probability calibration, the function produced by the pair-adjacent violators algorithm has the limitation of being piecewise constant. Several methods based on splines have been proposed to transform this function to a smoother one (Meyer, 2008; Wang and Li, 2008; Jiang et al., 2011; Zhong and Kwok, 2013), but they greatly increase the computational cost of training the calibration model.

It has been observed that isotonic regression can overfit easily on small datasets, due to its weak constraints. A proposed rule-of-thumb is to only use isotonic regression when there are more than 1,000 examples available in the training set (Niculescu-Mizil and Caruana, 2005b; Pedregosa et al., 2011).

Bayesian Binning into Quantiles

Naeini, Cooper, and Hauskrecht (2015) extend the histogram binning approach by using Bayesian model averaging across all possible binning schemes to produce a calibration model. A binning scheme for a binary

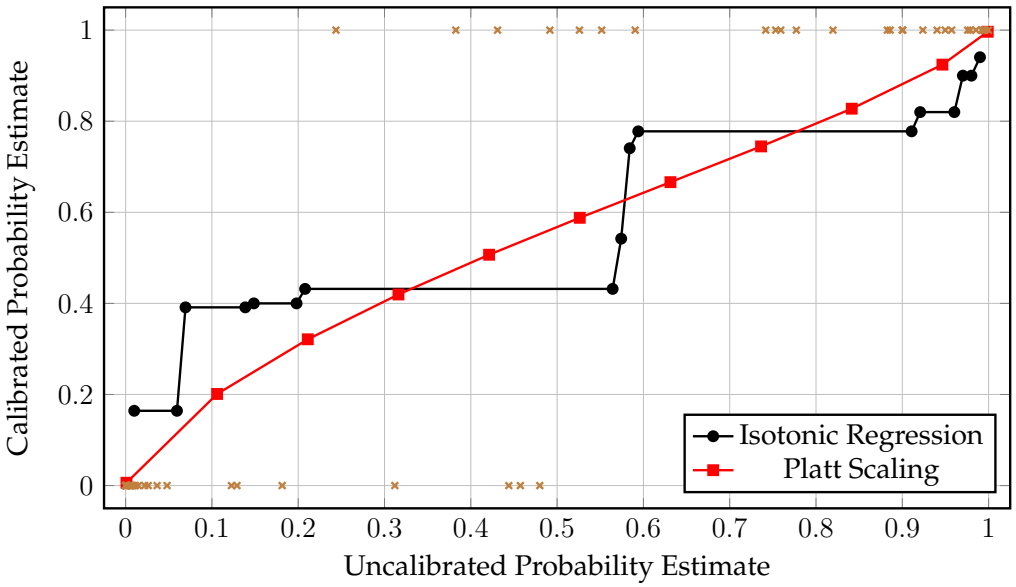


Figure 2.4: Isotonic regression and Platt scaling for probability estimates obtained from a Gaussian Naïve Bayes classifier. Brown crosses represent original, uncalibrated probability estimates on the x-axis and their true labels on the y-axis.

problem can be defined as a number of bins K and a partitioning of the probability space $[0, 1]$ into intervals such that $0 = t_1 \leq \cdots \leq t_K \leq t_{K+1} = 1$. The parameters $\theta_1, \ldots, \theta_K$ of the binning scheme are the returned calibrated probability estimates per bin. While histogram binning and isotonic regression create an individual binning scheme, Bayesian binning into quantiles considers a space $\mathcal{S}$ of potential schemes for the validation data $\mathcal{V}$:

$$\mathbb{P}(\tilde{p}_i|\hat{p}_i, \mathcal{V}) = \sum_{s \in \mathcal{S}} \mathbb{P}(\tilde{p}_i, s|\hat{p}_i, \mathcal{V}) \tag{2.30}$$

$$= \sum_{s \in \mathcal{S}} \mathbb{P}(\tilde{p}_i|s, \hat{p}_i, \mathcal{V})\mathbb{P}(s|\mathcal{V}) \tag{2.31}$$

where $\mathbb{P}(\tilde{p}_i|s, \hat{p}_i, \mathcal{V})$ is the calibrated probability using scheme s.[10] Using a uniform prior on the probability of each binning scheme, the

[10]Note that $\mathcal{S}$ is finite, because the validation set $\mathcal{V}$ has a finite number of samples.

weight $\mathbb{P}(s|\mathcal{V})$ can be derived by Bayes' rule as

$$\mathbb{P}(s|\mathcal{V}) = \frac{\mathbb{P}(\mathcal{V}|s)}{\sum_{s' \in S} \mathbb{P}(\mathcal{V}|s')}. \tag{2.32}$$

The parameters $\theta_1, \ldots, \theta_K$ associated with the bins can be thought of as the parameters of K independent binomial distributions. By imposing a beta prior on each of these parameters, a closed form expression for the marginal likelihood can be obtained. The parameters for the prior beta distribution associated with θ_k, α_k and β_k, are set as

$$\alpha_k = \frac{n'}{K} p_k, \qquad \beta_k = \frac{n'}{K}(1 - p_k)$$

where $p_k = (t_k + t_{k+1})/2$ is the midpoint of bin k, and n' is a term expressing the strength of our belief in the prior distribution.[11] The marginal likelihood can then be expressed as

$$\mathbb{P}(\mathcal{V}|s) = \prod_{k=1}^{K} \frac{\Gamma(\frac{n'}{K})}{\Gamma(n' + \frac{n'}{K})} \frac{\Gamma(n_{k+} + \alpha_k)}{\Gamma(\alpha_k)} \frac{\Gamma(n_{k-} + \beta_k)}{\Gamma(\beta_k)} \tag{2.33}$$

where $\Gamma(\cdot)$ is the gamma function

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} \mathrm{d}x \tag{2.34}$$

and n_{k+}, n_{k-} are the number of positive and negative examples in bin k respectively (Heckerman, Geiger, and Chickering, 1995). The averaged calibrated probability for any unseen, uncalibrated probability estimate can then be computed using (2.31).

Bayesian binning into quantiles has been shown to work well for obtaining calibrated probabilities. In particular, the expected calibration error and maximum calibration error (described in Section 2.3.3) are significantly lower compared to isotonic regression, Platt scaling, and histogram binning. However, it is much more computationally intensive and has

[11] Naeini, Cooper, and Hauskrecht (2015) set $n' = 2$.

been observed to take three orders of magnitude more time than these methods (Guo et al., 2017).

Probability Calibration Trees

Probability calibration trees (Leathart, Frank, et al., 2017) are a decision tree-based approach to probability calibration. Instead of calibrating the entire input space with a single calibration model, probability calibration trees attempt to split the input space into a series of regions with a decision tree and learn a calibration model for each region. Probability calibration trees are based on logistic model trees (Landwehr, Hall, and Frank, 2005; Sumner, Frank, and Hall, 2005) and learn local logistic regression models to calibrate training examples.

During the growth phase of probability calibration trees, each split node is considered a candidate leaf node, so there is a logistic-regression-based calibration model associated with every node in the tree. Instead of fitting a logistic regression model from scratch at each node, the LogitBoost algorithm (Friedman, Hastie, Tibshirani, et al., 2000), with simple linear regression models[12] as the weak learner, is used to incrementally refine logistic models that have already been learned at previous levels of the tree. Cross-validation is used to determine an appropriate number of boosting iterations. This results in an additive logistic regression model of the form

$$\hat{\mathbf{p}}^{(c)} = \mathbb{P}(y_i = c|\mathbf{x}_i) = \frac{e^{F_c(\mathbf{x}_i)}}{\sum_{j=1}^{m} e^{F_j(\mathbf{x}_i)}} \quad \text{where} \quad \sum_{j=1}^{m} F_j(\mathbf{x}_i) = 0. \tag{2.35}$$

Here, each $F_j(\mathbf{x}_i) = \sum_{k=1}^{l} f_{jk}(\mathbf{x}_i)$ where each $f_{jk}(\cdot)$ is a simple linear regression function, and l is the number of boosting iterations. This is equivalent to a multinomial logistic regression model, except that the linear models $F_c(\cdot)$ for each class c are built incrementally in an additive fashion. Note however that the maximum likelihood logistic regression model is

[12] These so-called "simple linear regression" models are based on a single attribute in the input vector. In other words, the weight vector $\mathbf{w}$ only has one non-zero entry.

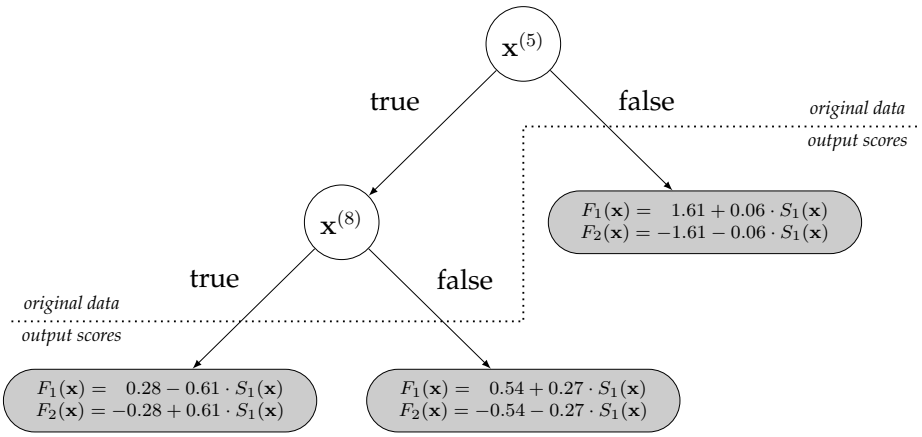


Figure 2.5: An example of a calibration tree for a simple binary dataset.

not likely to be obtained, because boosting is not run until convergence. As the logistic model tree grows, nodes that were previously leaf nodes become split nodes, and a number of child nodes are created. Each $F_c(\cdot)$ gets added to in the new children, based on the training examples that arrive in them.

An example of a calibration tree is shown in Figure 2.5, calibrating the outputs of a support vector machine with a Gaussian kernel ($C = 10, \gamma = 0.01$) on the *RDG1* dataset. *RDG1* is a small two-class dataset with 10 binary attributes, and can be generated in the WEKA software (Hall et al., 2009) using the eponymous data generator. $\mathbf{x}^{(5)}$ and $\mathbf{x}^{(8)}$ are attributes in the original data, while $S_1(\mathbf{x})$ is the output score of the support vector machine. The functions $F_j(\mathbf{x})$ compute the calibrated log-odds estimate of $\mathbf{x}$ belonging to class i, and must sum to zero. The final calibrated probabilities are computed with Equation 2.35.

Probability calibration trees have been shown to produce higher quality probability estimates than Platt scaling and isotonic regression on average, when considering root mean squared error (see Section 2.3.3), for a variety of base learners (Leathart, Frank, et al., 2017).

Robust Calibration

A common assumption made by calibration methods is that of *monotonicity*, i.e., the mapping from the original output scores to the calibrated probability is non-decreasing. Although this is a natural assumption to make for probability calibration, it makes calibration schemes that follow this assumption sensitive to outliers in the probability space, which can come about due to label noise in the data. This can be overcome by using methods from robust statistics (Rüping, 2006). Applying ideas from Rousseeuw (1984), a fraction τ of training examples with the highest and lowest absolute values can be removed from the training set before applying any standard calibration scheme. The optimal value of τ can be estimated using cross-validation, or simply using the training error (Rüping, 2006).

2.3.3 Measuring Probability Calibration

For balanced classification problems, classification accuracy is usually used as a metric for determining the quality of the model. However, this is not suitable for determining the level of probability calibration. Perfect probability calibration is defined as the case where

$$\mathbb{P}(\hat{y} = y \mid \hat{p} = p) = p, \qquad \forall p \in [0, 1], \tag{2.36}$$

and the probability is taken over the ground truth joint distribution of the features and labels. Unfortunately, this quantity is impossible to calculate in practice, as $\hat{p}$ is a continuous random variable, and we only have finitely many samples in a real dataset. Therefore, it is necessary to use a proxy metric that captures the essence of (2.36).

Negative Log-Likelihood

Negative log-likelihood (NLL), also referred to as cross-entropy (Goodfellow et al., 2016) and log-loss, is a commonly-used metric for model performance based on the quality of the estimated probabilities. It heavily penalises probability estimates that are far from the true label. Negative log-likelihood for multiclass problems is defined as

$$\text{NLL} = \mathcal{L}_{\log}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\mathbf{y}_i^{(j)}\log(\hat{\mathbf{y}}_i^{(j)})\right). \tag{2.37}$$

Note that because $\mathbf{y}_i$ is a one-hot vector, the term corresponding to the correct class in the inner summation is the only non-zero term. If the estimated distribution is equal to the one-hot representation of the label (i.e., the model assigns 100% probability to the correct class), then NLL is minimised.[13] The worst possible score for NLL is ∞, and is achieved when the estimated probability for the true class is zero. In practice, it is advisable to clip the estimates in $\hat{\mathbf{y}}_i$ to be in the range $[\epsilon, 1-\epsilon]$ for some small value of ϵ, rather than the usual probability range $[0, 1]$, to avoid numerical issues during optimisation.

For completeness, we also define the binary cross-entropy (BCE) for two class problems separately:

$$\text{BCE} = \mathcal{L}_{\log}(\hat{y}_i, y_i) = -\frac{1}{n}\sum_{i=1}^{n} y_i\log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i), \tag{2.38}$$

where $\hat{y}_i$ is a scalar probability estimate in the range $[0, 1]$. In this thesis, this representation is sometimes used for convenience of notation when discussing binary problems.

[13] However, for non-trivial learning problems, achieving zero NLL on the training set is almost always a sign of overfitting, which results in poor probability calibration on unseen examples. Also note that most calibration models have sufficiently low capacity or have regularisation applied such that, assuming the calibration set is large enough, this virtually never happens in practice.

Root Mean Squared Error

Related to the commonly used mean squared error metric for regression problems, the root mean squared error of a probability estimator (RMSE) applies a square root to a quadratic loss function in order for the final value to be on the same scale as the inputs. For multiclass classification problems, RMSE is defined as the square root of the Brier score (Brier, 1950) divided by the number of classes m:

$$\text{RMSE} = \mathcal{L}_{\text{RMSE}}(\mathbf{y}, \hat{\mathbf{p}}) = \sqrt{\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\hat{\mathbf{p}}_i^{(j)} - \mathbf{y}_i^{(j)} \right)^2}. \tag{2.39}$$

Like NLL, it also has a simplified, equivalent form for binary classification problems:

$$\text{RMSE} = \mathcal{L}_{\text{RMSE}}(y, \hat{p}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{p}_i - y_i)^2}. \tag{2.40}$$

When considering probability estimates, RMSE is zero if the estimated distribution is equal to the one-hot representation of the label, and equals $\sqrt{\frac{2}{m}}$ in the worst case scenario where one incorrect class k is assigned $\hat{\mathbf{y}}^{(k)} = 1$ and every other entry in $\hat{\mathbf{p}}$ is set to 0.[14]

Reliability Plots and Expected Calibration Error

Although RMSE and NLL are good proxy metrics for the quality of probabilities estimated by a model, there are more direct ways to measure probability calibration. Reliability diagrams provide a qualitative method to visualise model calibration (Murphy and Winkler, 1977; DeGroot and Fienberg, 1983). In reliability diagrams, the probability space is discretised into a series of bins $B_1, \ldots, B_K$. Similarly to histogram binning, the bin boundaries t_n are typically chosen such that each bin has equal numbers of instances, or equal width. Then, the *accuracy* and *confidence* of each

[14] Similarly to NLL, achieving zero RMSE is usually a sign of overfitting, and is very rare in sensible practical settings.

bin are plotted against each other. The accuracy and confidence of each bin are defined as

$$\text{acc}(B_k) = \frac{1}{|B_k|} \sum_{i \in B_k} \mathbb{I}(\hat{y}_i = y_i), \tag{2.41}$$

$$\text{conf}(B_k) = \frac{1}{|B_k|} \sum_{i \in B_k} \hat{p}_i. \tag{2.42}$$

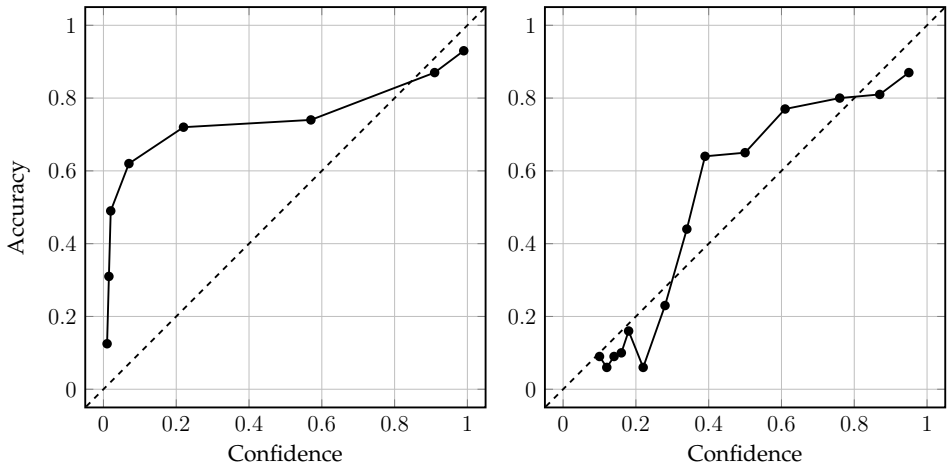
Perfectly calibrated probability estimates will result in a reliability plot where the points lie on the diagonal, which indicates the average accuracy of each bin is equal to the average confidence. This provides a simple tool to visualise miscalibration. In Figure 2.6, reliability diagrams are plotted for probability estimates from naïve Bayes, and naïve Bayes calibrated with Platt scaling (applied to log-odds) and isotonic regression. It can be seen that after applying Platt scaling and isotonic regression, the reliability curves are nearer the diagonal (shown with a dotted line), indicating better calibration.

Naeini, Cooper, and Hauskrecht (2015) build on the idea behind reliability diagrams by proposing a metric called expected calibration error (ECE), which is a weighted average of the absolute residuals in the reliability diagram. It is defined as

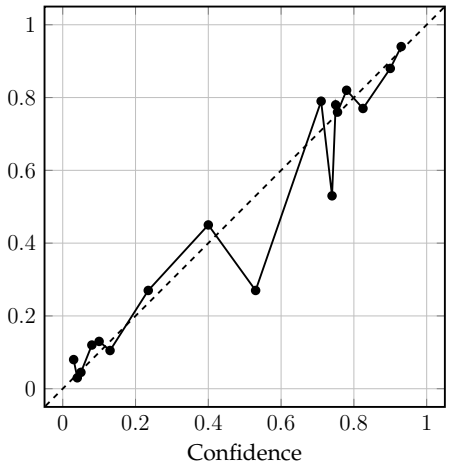
$$\text{ECE} = \sum_{k=1}^{K} \frac{|B_k|}{n} \Big|\text{acc}(B_k) - \text{conf}(B_k)\Big|. \tag{2.43}$$

In addition to considering this weighted average, it may also be of value to consider the worst-case discrepancy between confidence and accuracy for particular applications where accurate probability estimates are absolutely essential. To this end, the maximum calibration error (MCE) has also been proposed (Naeini, Cooper, and Hauskrecht, 2015) and is defined as

$$\text{MCE} = \max_{k \in 1,\ldots,K} \Big|\text{acc}(B_k) - \text{conf}(B_k)\Big|. \tag{2.44}$$



(a) Naïve Bayes **(b)** Platt Scaling



(c) Isotonic Regression

Figure 2.6: Reliability diagrams for predictions of the *credit* dataset from the UCI repository (Lichman, 2013).

ECE and MCE are both zero for a perfectly calibrated classifier.

Chapter 3

The Random-Pair Method

An adaptive, semi-random subset selection method

The structure of nested dichotomies can have a large impact on their performance. The accuracy of each individual nested dichotomy is improved when similar classes are placed in the same class subsets for as long as possible (Duarte-Villaseñor et al., 2012). However, diversity is key for ensemble performance (Kuncheva and Whitaker, 2003). This chapter describes a method for choosing structures that keeps similar classes together, while also employing an element of randomness to promote ensemble diversity.

Relevant Publications

Parts of the work presented in this chapter are included in "Building Ensembles of Nested Dichotomies with Random-Pair Selection", Leathart, Pfahringer, and Frank (2016). This chapter also includes additional examples explaining the usefulness of the method, and an investigation into using a softmax confusion matrix from (Wang, Wang, and Wang, 2018) to group the class subsets.

Software

The software used for the experiments in this chapter is available in the WEKA package manager as `nestedDichotomies`.

3.1 Motivation

Keeping similar classes together for as long as possible in the structure of a nested dichotomy is intuitively beneficial. When similar classes are grouped together, the internal binary classifiers at each node have an easier task in discriminating between the two meta-classes, and the binary classifiers that appear nearer the leaf nodes can focus on more fine-grained differences between the classes. Additionally, when the structure exhibits a natural hierarchy found in the labels, incorrect predictions are more likely to be semantically related to the correct class. This is because misclassifications in nested dichotomies are biased towards classes that share ancestors with the true class, because the majority of the probability mass is likely to reside on the path to the shared ancestor node. The benefits of keeping similar classes together when constructing hierarchical decompositions have been recognised many times in the literature (Bengio, Weston, and Grangier, 2010; Deng et al., 2011; Duarte-Villaseñor et al., 2012; Wang, Wang, and Wang, 2018). Duarte-Villaseñor et al. (2012) proposed a method based on clustering, using class centroids, to achieve this goal in nested dichotomies. This has been described in-depth in Section 2.2.1. In what follows, we identify two shortfalls of this clustering-based method that motivate a different approach:

- the deterministic nature of the construction, and
- the assumption that class centroids are necessarily useful.

3.1.1 Deterministic Subset Selection

In nested dichotomies based on clustering (Duarte-Villaseñor et al., 2012), the selection of structure is deterministic for a particular dataset. In other words, the growth function for this style of construction is

$$T^{(C)}(m) = 1. \tag{3.1}$$

This is fine for building a single nested dichotomy, but we usually wish to build an ensemble to maximise predictive accuracy. Simply applying this algorithm multiple times to the training data, as in the originally proposed randomisation-based technique for constructing ensembles of nested dichotomies, is not sufficient because all ensemble members will be identical.

Rodríguez, García-Osorio, and Maudes (2010) showed that ensemble performance can be improved in nested dichotomies by applying other ensemble methods, such as AdaBoost (Freund and Schapire, 1996b), Bagging (Breiman, 1996) and MultiBoost (Webb, 2000). Duarte-Villaseñor et al. (2012) use these methods in their experiments and indeed achieve improvements in accuracy compared to individual nested dichotomies. However, we hypothesise that further improvements can be made by encouraging stronger ensemble diversity.

3.1.2 The Usefulness of Class Centroids

Centroid-based methods assume that distance between class centroids is indicative of class similarity. While it is true that this is often the case, this assumption does not hold in certain circumstances. For example, what if a class is multimodal? When a class is comprised of several clusters, the class centroid is likely to be somewhere in-between them, in a location that is not representative of the distribution of the class. Figure 3.1 shows an example of this. Additionally, class centroids and their pairwise Euclidean distances are less useful in high-dimensional spaces. A

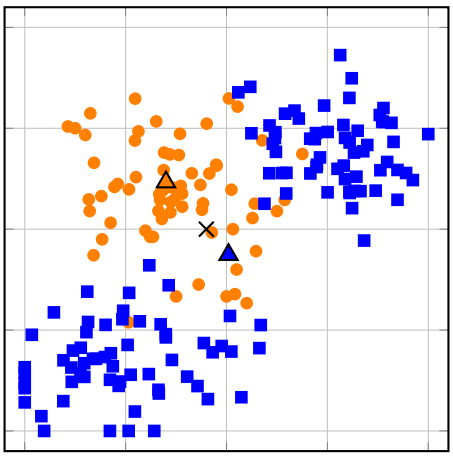


Figure 3.1: An example of a situation where centroid-based techniques fail with multimodal classes. The centroids of each class are shown as triangles. A query point (labeled $\times$) in the orange class is assigned to the blue class when using distance to the centroids for classification.

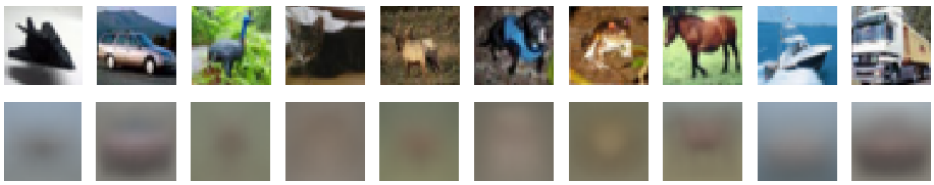


Figure 3.2: Top: A random sample of each class in *CIFAR-10*. Bottom: Visual representation of the class centroids.

well-known, unintuitive result is that in high dimensions, distances become less meaningful. In fact, the difference between the maximum and minimum distance between two points in a distribution approaches zero as the number of dimensions approaches infinity (Aggarwal, Hinneburg, and Keim, 2001).

As an illustrative example of problematic behaviour on practical data, we show the behaviour of centroid methods in *CIFAR-10*, a dataset comprised of 32 $\times$ 32 pixel natural images (Krizhevsky and Hinton, 2009). *CIFAR-10* has ten classes, including cats, dogs, cars and trucks, with 6,000 examples for each class. Figure 3.2 shows a random sample from each class, as well as the centroids of each class, as an image. Some of the classes are slightly recognisable from the centroid (e.g., horses), but it is clear that these centroids do not convey much meaningful information about the classes. This is shown further when evaluating a simple

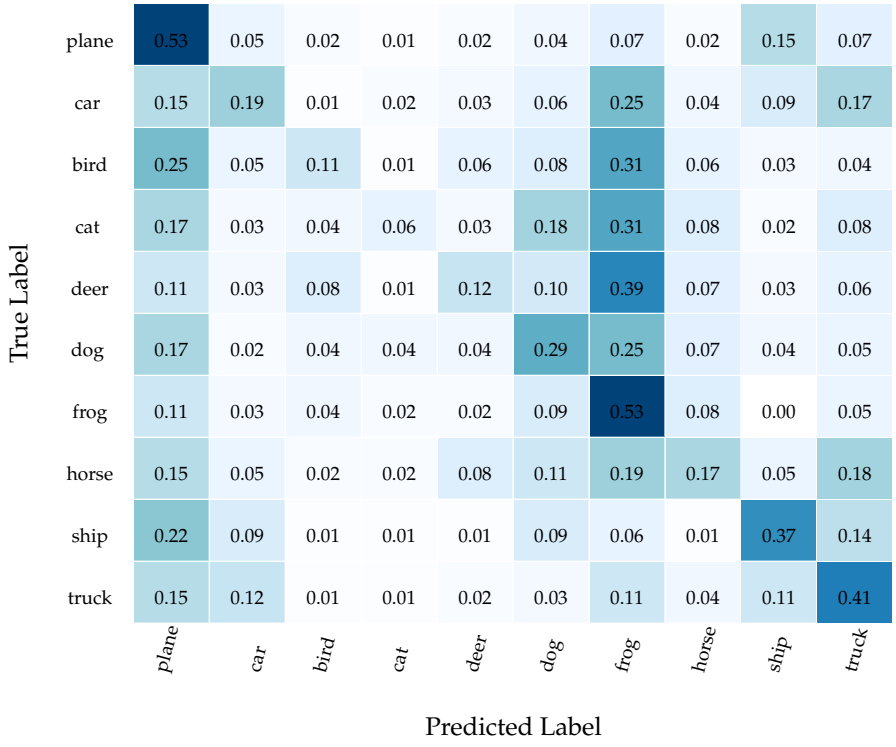


Figure 3.3: Confusion matrix for a simple centroid classifier applied to *CIFAR-10*. The darkness of the background colour of each cell indicates the proportion of examples whose predictions and true class match the row and column, normalised by row.

centroid-based classifier, which classifies test examples according to the closest centroid. Figure 3.3 shows a confusion matrix for the predictions made by such a classifier on the *CIFAR-10* test set (10,000 examples—1,000 for each class). While some classes are recognised fairly well (e.g., trucks, ships, planes and frogs), most other classes are not classified accurately. Cats, birds and deer are hardly ever correctly recognised by this simple model. Interestingly, even though the centroid of the horse class visually appears informative, horses are commonly misidentified as frogs, trucks or planes.[1]

It is clear from this example that centroid based approaches are not appropriate for some data types. This motivates the need for an approach to discover similar classes more directly.

[1]Instances were normalised to unit length, and Euclidean distance was used to determine the closest centroid.

3.2 The Random-Pair Method

The method proposed in this chapter, called the random-pair method, addresses both issues with the centroid-based methods proposed by Duarte-Villaseñor et al. (2012). In the random-pair method, the base learner used for the internal models is used directly to discover similar classes, and the approach is non-deterministic, naturally yielding diverse decompositions when building ensembles of nested dichotomies.

The random-pair method selects two classes present at a node at random, and groups the remaining classes with one of the initial random pair by using a classifier trained to distinguish the initial random pair. More formally, a random-pair of classes c_1, c_2 is chosen from $\mathcal{C}_k$, and a binary classifier h is trained using only these two classes. Then, the rest of the data corresponding to the remaining classes is classified by h. If the majority of some class c is classified as c_1 by h, then c will be grouped with c_1, and vice-versa. The procedure for splitting the class set $\mathcal{C}_k$ for some node k in a nested dichotomy is described in detail in Algorithm 4.

Once all the classes have been grouped into one of the two groups in this manner, a binary classifier is retrained on the two meta classes and used as the classifier for node k in the nested dichotomy.

This process is illustrated for an example four-class dataset in Figure 3.4. The classes are positioned such that certain choices of $\mathcal{C}_{kl}$ and $\mathcal{C}_{kr}$ will result in very poor accuracy for a linear model (Figure 3.4a). Assume the blue square and orange circle classes are selected randomly, and a linear classifier is trained (Figure 3.4b). After the binary classifier has been built, the red open circle and green triangle classes are classified by the linear model (Figure 3.4c). The majority of red open circles are classified as orange circles while all of the green triangles are classified as blue squares, so red is joined with orange and green with blue and a new linear classifier is trained (Figure 3.4d).

Assuming a suitably chosen base learner, it is simple to reason that choices of $\mathcal{C}_{kl}$ and $\mathcal{C}_{kr}$ that are not easy for the classifier to handle are rarely

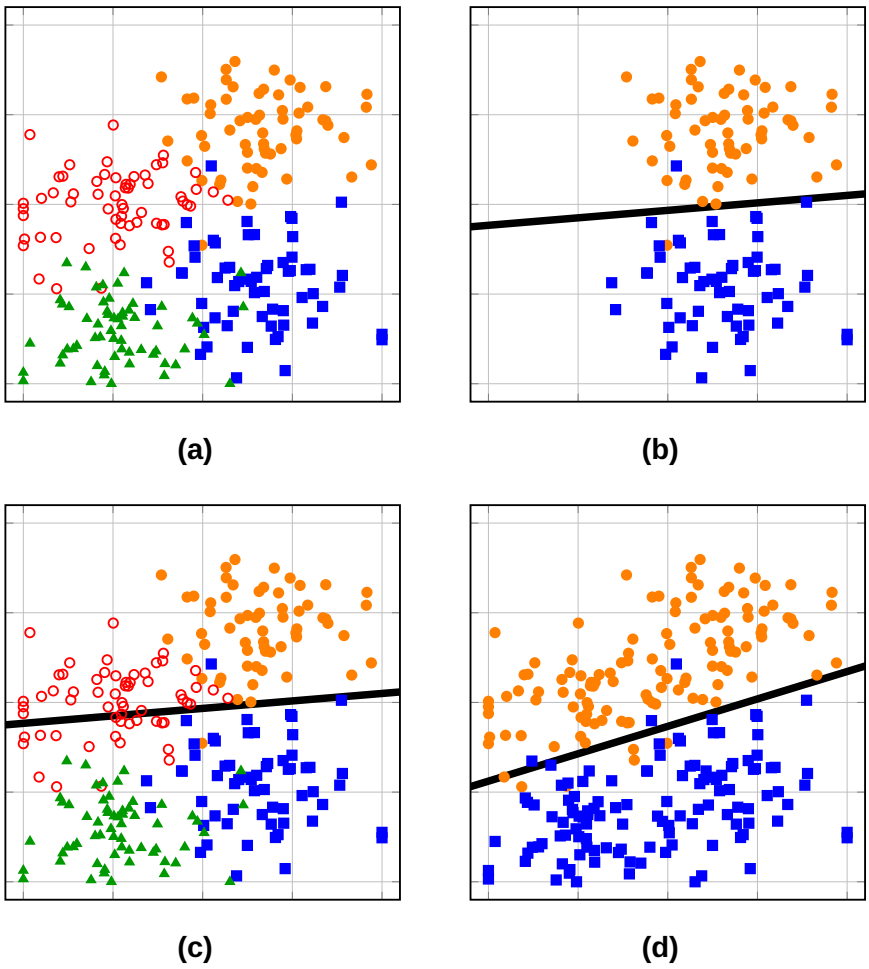


Figure 3.4: Illustration of the random-pair method on a four-class, two-dimensional problem with a linear model as the base learner.

(or never) selected under this scheme. In the example in Figure 3.4, where a linear classifier is used as the base learner, one such undesirable split is

$$\mathcal{C}_{kl} = \{\text{green, orange}\} \text{ and } \mathcal{C}_{kr} = \{\text{blue, red}\},$$

which is not linearly separable.[2] In order for this subset selection to eventuate, green and orange must not both appear in the initial random-pair (as this would mean they are in different subsets). For the same reason, blue and red must not both appear in the initial random-pair. Yet, for any random-pair such that $c_1 \in \{\text{green, orange}\}$ and $c_2 \in \{\text{blue, red}\}$, it

[2]This scenario resembles the infamous XOR learning problem (Minsky and Papert, 1969).

Algorithm 4 Random-pair method
Input: Dataset $\mathbf{D}$, classifier h

Set $\mathcal{C}$ as the set of classes present at in $\mathbf{D}$
Select $c_1, c_2 \in \mathcal{C}$ at random without replacement
Set $\mathcal{C}_l = \{c_1\}, \mathcal{C}_r = \{c_2\}$
if $|\mathcal{C}| > 2$ **then**
    Train h on $\mathbf{D}_{\{c_1,c_2\}}$
    Set $\bar{\mathbf{C}}$ as confusion matrix[3]of remaining data $\mathbf{D}_{\mathcal{C}\setminus\{c_1,c_2\}}$ classified by h
    for $c \in \setminus \{c_1, c_2\}$ **do**
        if $\bar{\mathbf{C}}^{(c,c_1)} \leq \bar{\mathbf{C}}^{(c,c_2)}$ **then**
            Add c to $\mathcal{C}_l$
        else
            Add c to $\mathcal{C}_r$
        end if
    end for
end if
Output $\mathcal{C}_l, \mathcal{C}_r$.

is clear that the worst case outlined above will not occur. For illustrative purposes, say $c_1 =$ green, and $c_2 =$ blue. Any linear classifier that would assign orange with green would also include red in the same grouping. Similarly, any linear classifier that would assign red with blue would also include orange.

This approach is superior to centroid based methods, because it uses the actual base learner employed in the nested dichotomy to determine how similar the classes are. Class similarity under a simple linear classifier is fairly well-approximated by a centroid distance measure, but for more complex models, this is not the case (as also observed by Wang, Wang, and Wang (2018)).

3.2.1 Employing a Softmax Confusion Matrix

The basic random-pair algorithm described above is based on a confusion matrix that is calculated from hard classifications. Even though the softmax confusion matrix utilised by Wang, Wang, and Wang (2018) was

[3]This 'confusion matrix' is not a typical square confusion matrix; it is of shape $|\mathcal{C}| \times 2$, as there are only two output classes for h. Alternatively, it can be thought of as a regular square confusion matrix, in which only columns corresponding to c_1 and c_2 are nonzero.

	c_1	c_2
c	50	100

(a) Partial hard confusion matrix

	c_1	c_2
c	99.0	51.0

(b) Partial softmax confusion matrix

Figure 3.5: Comparison of partial hard and softmax confusion matrices for the toy problem discussed in Section 3.2.1.

intended for direct use in multiclass learning problems (see Section 2.1.2), we can also use it to obtain a more finely-grained estimate of classifier performance in binary problems. For instance, consider a case where we are trying to assign some class c into a subset $\mathcal{C}_{kl}$ or $\mathcal{C}_{kr}$ for an internal node k. Say the predictions for instances belonging to class c are split: 50 of the examples yield very high-confidence predictions as c_1 where $\mathbb{P}(y = c_1|\mathbf{x}) \approx 1.0$, while the remaining 100 examples are assigned very low-confidence predictions as c_2 where $\mathbb{P}(y = c_2|\mathbf{x}) \approx 0.51$. A hard confusion matrix (Figure 3.5a) will show more predictions as c_2 than c_1, and so class c will be placed into $\mathcal{C}_{kr}$. Meanwhile, a softmax confusion matrix (Figure 3.5b) shows a higher percentage of the probability mass being assigned to c_1[4], and so c will be grouped accordingly.

To this end, in our experimental evaluation of the random-pair method, we also consider using binary probability estimates in this manner to assign remaining classes to the class subsets. We conjecture that utilising the softmax confusion matrix to group the classes into subsets may result in more suitable class subsets being selected, which will produce easier binary subproblems to solve. In particular, we suspect that it will have more profound effects for imbalanced datasets with some very small classes.

3.2.2 Growth Function Analysis

As previously stated, a non-deterministic subset selection algorithm is more effective for generating diverse ensembles of nested dichotomies than a deterministic one. A useful tool for analysing this is to look at

[4]Note that each prediction where $\mathbb{P}(y = c_2|\mathbf{x}) \approx 0.51$ also has $\mathbb{P}(y = c_1|\mathbf{x}) \approx 0.49$.

how the number of possible nested dichotomies grows with the number of classes. This so-called *growth function* has been discussed in the context of random and class-balanced selection in Section 2.2.

While there exists a closed form solution for the growth functions of random and class-balanced selection, the random-pair method is more complex to analyse. The growth function for an m-class problem depends on characteristics of the dataset and the base learner used at the internal nodes. An upper bound for the growth function is derived from the number of possible random-pairs that can be selected

$$T^{(RP)}(m) \leq \binom{m}{2}, \tag{3.2}$$

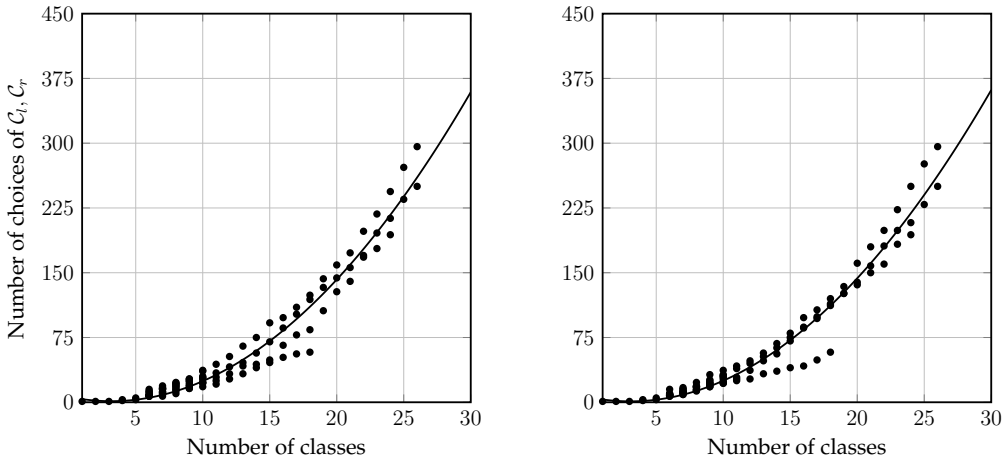
but it is likely that many random pairs will result in the same class subsets being produced by Algorithm 4.

Due to the lack of a closed form solution for $T^{(RP)}$, we empirically estimate these functions for two well-understood base learners that are at opposite ends of the bias-variance spectrum: logistic regression and C4.5 (Quinlan, 1993), as described in Section 1.4.1. We exhaustively enumerate and count the number of possible class splits for the random-pair method at each internal node of a nested dichotomy for a number of datasets (listed in Table 3.1), and plot this against the number of classes at the corresponding internal node in Figure 3.6. Fitting a second degree polynomial this data yields

$$p^{(LR)}(c) = 0.4973c^2 - 3.153c + 6.269 \tag{3.3}$$

$$p^{(DT)}(c) = 0.4970c^2 - 3.026c + 5.770 \tag{3.4}$$

Additionally, we found that the average size of the class subsets with either base learner is $\frac{2}{3}$ and $\frac{1}{3}$ of the available classes respecively, so the fitted polynomials can be used to provide a rough estimate of the growth



(a) Logistic Regression **(b)** C4.5

Figure 3.6: Estimated number of choices for $\mathcal{C}_l, \mathcal{C}_r$ for different numbers of classes.

functions for logistic regression, $T^{(RP_{LR})}(\cdot)$, and C4.5, $T^{(RP_{DT})}(\cdot)$,

$$T^{(RP_{LR})}(m) = p^{(LR)}(m)T^{(RP_{LR})}\left(\frac{m}{3}\right)T^{(RP_{LR})}\left(\frac{2m}{3}\right) \tag{3.5}$$

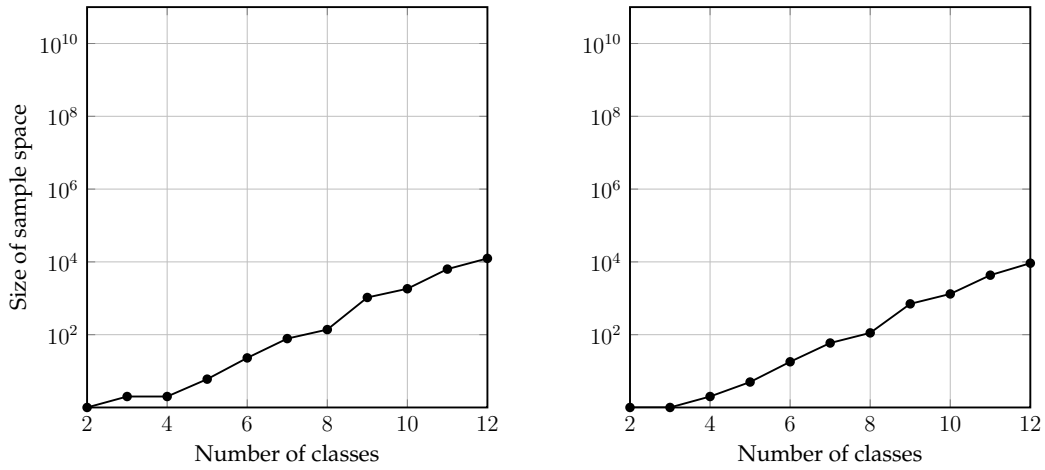
$$T^{(RP_{DT})}(m) = p^{(DT)}(m)T^{(RP_{DT})}\left(\frac{m}{3}\right)T^{(RP_{DT})}\left(\frac{2m}{3}\right) \tag{3.6}$$

where $T^{(RP_{LR})}(m) = T^{(RP_{DT})}(m) = 1$ for $m \leq 2$. These growth functions are plotted in Figure 3.7.

It is clear from the plots that although the overall number of possible random-pair nested dichotomies is lower than when random or class-balanced selection is applied (Figure 2.2a and Figure 2.2b respectively), there are sufficiently many to ensure diverse ensembles for datasets with non-trivial numbers of classes.

3.3 Experiments

The proposed approach is simple and intuitively appealing, but what ultimately matters is its effect on predictive performance. This section presents an evaluation of the random-pair selection method on 18 datasets from the UCI repository (Lichman, 2013). Table 3.1 lists and describes the datasets we used. We specifically selected datasets with at least



(a) Logistic Regression **(b)** C4.5

Figure 3.7: Estimated growth functions for nested dichotomies built with the random-pair method.

five classes, as the method should not have a large impact on datasets with few classes, because there is a relatively low number of possible nested dichotomies for low numbers of classes.

3.3.1 Experimental Setup

All experiments were conducted in WEKA (Hall et al., 2009) and performed with 10 times 10-fold cross validation. When an existing WEKA learning algorithm was applied, its default settings were used in our evaluation. We compared the proposed class subset selection method with nested dichotomies based on clustering (NDBC, (Duarte-Villaseñor et al., 2012)), class-balanced nested dichotomies (CBND, (Dong, Frank, and Kramer, 2005)), and uniform sampling (ND, (Frank and Kramer, 2004)). We did not compare against other variants of nested dichotomies such as data-balanced nested dichotomies, nested dichotomies based on clustering with radius, and nested dichotomies based on clustering with average radius because they were found to either have the same or worse performance on average in (Dong, Frank, and Kramer, 2005) and (Duarte-Villaseñor et al., 2012) respectively. Logistic regression and C4.5 were used as the base learners, as described in Section 1.4.1.

Table 3.1: The datasets used in this evaluation.

Dataset	Classes	Instances	Features
audiology	24	226	70
krkopt	18	28056	7
LED24	10	5000	25
letter	26	20000	17
mfeat-factors	10	2000	217
mfeat-fourier	10	2000	77
mfeat-karhunen	10	2000	65
mfeat-morph	10	2000	7
mfeat-pixel	10	2000	241
optdigits	10	5620	65
page-blocks	5	5473	11
pendigits	10	10992	17
segment	7	2310	20
shuttle	7	58000	10
usps	10	9298	257
vowel	11	990	14
yeast	10	1484	9
zoo	7	101	18

We present critical difference plots (Demšar, 2006) for each experimental setting. These plots compare the average ranks of each method, considering their classification accuracy. Methods whose performances are not statistically significantly different at $p = 0.05$ are linked with a black horizontal bar. We also show the full results tables in Appendix A.2. In our results tables, a bullet (•) indicates a statistically significant accuracy gain, and an open circle (◦) indicates a statistically significant accuracy reduction ($p = 0.05$) by using the random-pair method compared with another method. To establish significance, we used the corrected resampled paired t-test (Nadeau and Bengio, 2000), applied to the sets of 100 performance estimates obtained by 10 times 10-fold cross-validation.

3.3.2 Hard vs. Softmax Confusion Matrix

As previously discussed, we also experimented with the softmax confusion matrix from (Wang, Wang, and Wang, 2018) to decide the groupings

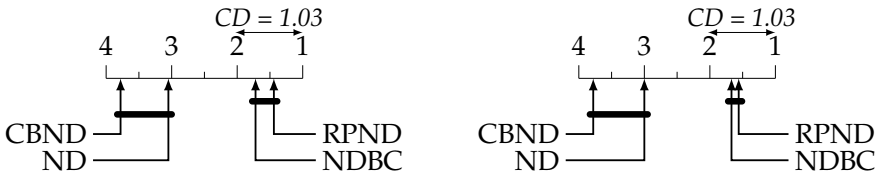
of the class subsets. We found that performance was almost identical between the two methods, even for highly imbalanced datasets like *audiology* and considering performance metrics like unweighted macro averaged F-measure that are often applied in imbalanced scenarios. In almost all cases, the resulting nested dichotomy structure is not changed. Therefore, for the remaining experiments, we use the hard confusion matrix.

3.3.3 Single Nested Dichotomy

It is expected that class subset selections with similar classes grouped together will have a larger impact in small ensembles of nested dichotomies: as ensembles grow larger, ensemble members that happen to be particularly poorly constructed will not have as great of an influence over the final predictions. To consider the extreme case of ensembles of size one, we first compare a single nested dichotomy using random-pair selection with the other class selection methods described above.

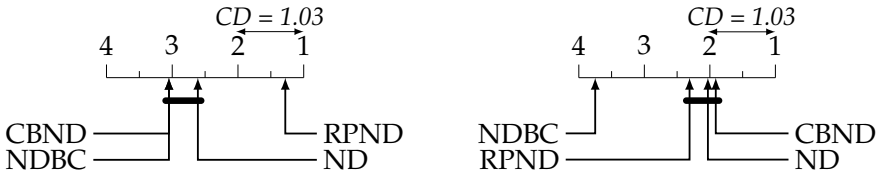
Figure 3.8 shows critical difference plots for the average ranks of each method. For both base learners, random-pair nested dichotomies are ranked first but are not significantly different from the clustering-based method at $p = 0.05$. Class-balanced and random nested dichotomies trail substantially behind. It makes intuitive sense that the data-driven approaches to construction of the hierarchy of classes should compare favourably to the fully random methods in this case.

Tables A.1 and A.2 show the full results of each method when building a single nested dichotomy. When logistic regression is used as the base learner (Table A.1), compared to more random methods (class balanced and random), the random-pair method yields a significant accuracy gain in most cases and comparable accuracy in all others. When using C4.5 as the base learner (Table A.2), our method is preferable to the fully random methods in some cases, with all other datasets showing a comparable accuracy.



(a) Logistic Regression **(b)** C4.5

Figure 3.8: Average ranks of a single nested dichotomy, considering classification accuracy, built using different selection methods.



(a) Logistic Regression **(b)** C4.5

Figure 3.9: Average ranks of ensembles of ten bagged nested dichotomies, considering classification accuracy, built using different selection methods.

In comparison to nested dichotomies based on clustering, the proposed method gives similar accuracy, with four significantly better results and four significantly worse ones across the 17 datasets. It is to be expected that the clustering-based method sometimes has better performance than the proposed method when only a single nested dichotomy is built because it selects the class split that is likely to be the most easily separable in a deterministic fashion, while the proposed method attempts to produce an easily separable class subset selection by choosing at random from a pool of suitable options.

3.3.4 Ensembles of Nested Dichotomies

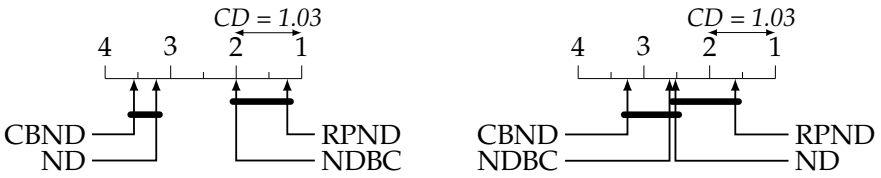
It is well known that ensembles of nested dichotomies typically outperform single nested dichotomies (Frank and Kramer, 2004; Rodríguez, García-Osorio, and Maudes, 2010). The original method for creating an ensemble of nested dichotomies was purely based on randomisation, but it was later found that better performance can be obtained by bagging and boosting nested dichotomies (Rodríguez, García-Osorio, and Maudes, 2010). For this reason, we consider three types of ensembles of

nested dichotomies in our experiments: ensembles obtained with bagging (Breiman, 1996), boosting with AdaBoost (Freund and Schapire, 1997) and boosting with MultiBoost (Webb, 2000), described in Section 1.4.2. We built ensembles of ten nested dichotomies in these experiments for each of these ensemble methods.

Bagging

Figure 3.9 shows critical difference plots for the average ranks of each method when bagging is used to create an ensemble of nested dichotomies. For logistic regression, the random-pair method is the clear winner, with all other methods performing comparably to each other. When C4.5 is used as the base learner, the random-pair method produces ensembles of nested dichotomies of similar performance to class-balanced and random sampling, while the clustering-based method yields poor ensembles.

Tables A.3 and A.4 show the full results of using bagging to construct an ensemble of nested dichotomies for each method and for both base learners. When logistic regression is used as a base learner, the proposed method outperforms all other methods in many cases. When C4.5 is used as the base learner, the proposed method compares favourably with the clustering-based method and achieves comparable accuracy to the random methods. The proposed method is better in a bagging scenario than NDBC because of the first problem highlighted in Section 3.1, *i.e.*, using the furthest centroids to select a class split results in a deterministic class split. Evidently, with bagged datasets, this method of class subset selection is too stable to be utilized effectively. The proposed method, on the other hand, is sufficiently unstable to be useful in a bagged ensemble.



(a) Logistic Regression **(b)** C4.5

Figure 3.10: Average ranks of ensembles of ten nested dichotomies boosted with AdaBoost, considering classification accuracy, built using different selection methods.

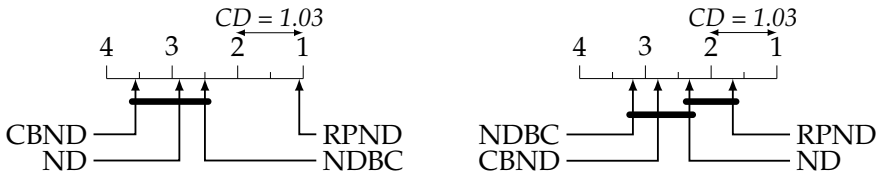
AdaBoost

Figure 3.10 shows critical difference plots for the average ranks of each method when AdaBoost is used to create an ensemble of nested dichotomies. The random-pair method is ranked first when either logistic regression or C4.5 is used as the base learner, although it is accompanied by nested dichotomies based on clustering and random nested dichotomies in the top-ranked groups respectively.

Tables A.5 and A.6 show the full results of using AdaBoost to build an ensemble of nested dichotomies for each method and for both base learners. When comparing with the random methods, we observe a similar result to bagged ensembles. When using logistic regression, we see a significant improvement in accuracy in many cases, and when C4.5 is used, we typically see comparable results, with a small number of significant accuracy gains. When comparing with the clustering-based method, we see a small improvement for the vast majority of datasets, but these differences are almost never individually significant. In one instance (*krkopt* with C4.5 as the base learner), we achieve a significant accuracy gain using the proposed method.

MultiBoost

Figure 3.11 shows critical difference plots for the average ranks of each method when MultiBoost is used to create an ensemble of nested dichotomies. Again, the random-pair method is ranked first when either



(a) Logistic Regression **(b)** C4.5

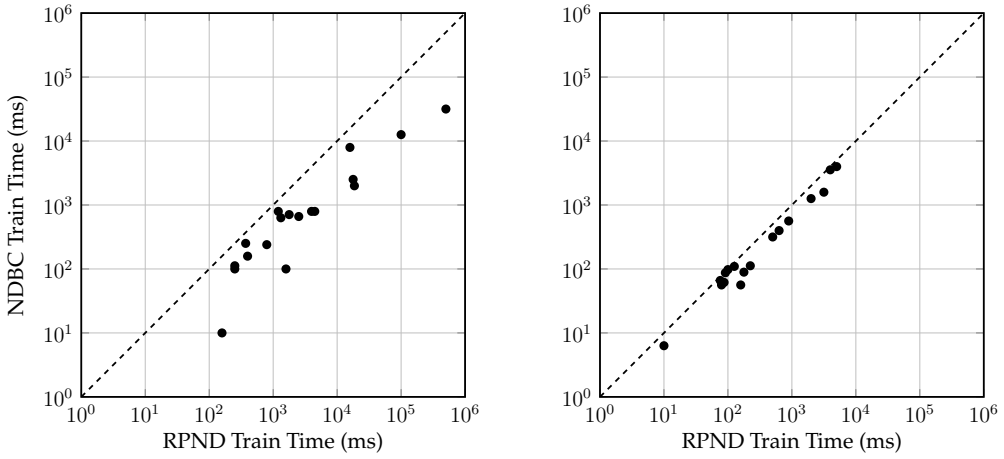
Figure 3.11: Average ranks of ensembles of ten nested dichotomies boosted with MultiBoost, considering classification accuracy, built using different selection methods.

logistic regression or C4.5 are used as base learners, performing especially well for logistic regression where its average rank is almost exactly one, indicating it gives the best classification accuracy in nearly all cases. When C4.5 is used, it does not perform significantly differently to random nested dichotomies at $p = 0.05$.

Tables A.7 and A.8 show the full results of using MultiBoost to build an ensemble of nested dichotomies for each method and for both base learners. Compared to the random methods, we see similar results to the other ensemble methods: with logistic regression as the base learner we obtain many significant improvements, and using C4.5 as the base learner typically produces comparable results, with few significant improvements. In comparison to the clustering-based method, we see many small (although statistically insignificant) improvements across both base learners, with some significant gains in accuracy on some datasets.

3.3.5 Training Time

Figure 3.12 shows the training time in milliseconds for training a single nested dichotomy with random-pair selection and with the clustering-based method, with logistic regression and C4.5 as the base learners, for each of the datasets used in this evaluation. As can be seen from the plots, there is a computational cost for using the random-pair method over the clustering-based method, which is to be expected as there is an additional classifier trained and tested at each split node of the tree. However, the



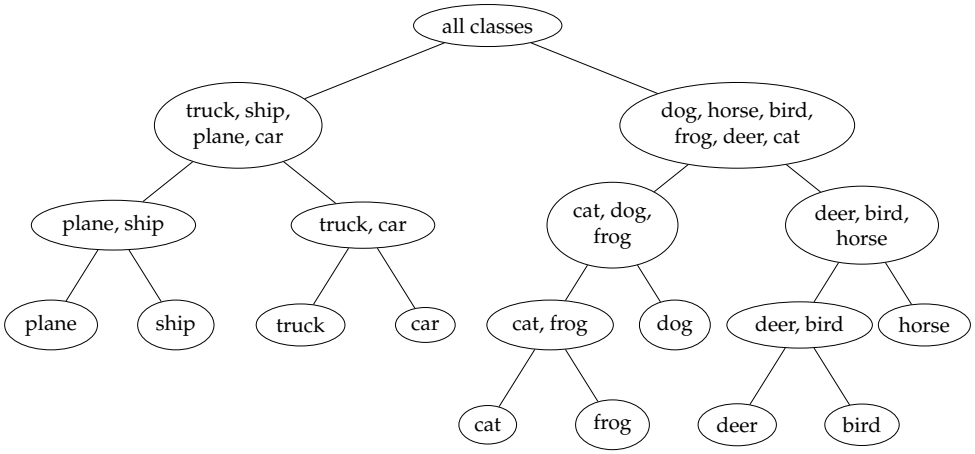
(a) Logistic Regression **(b)** C4.5

Figure 3.12: Log-log plots of training time (ms) for a single nested dichotomy.

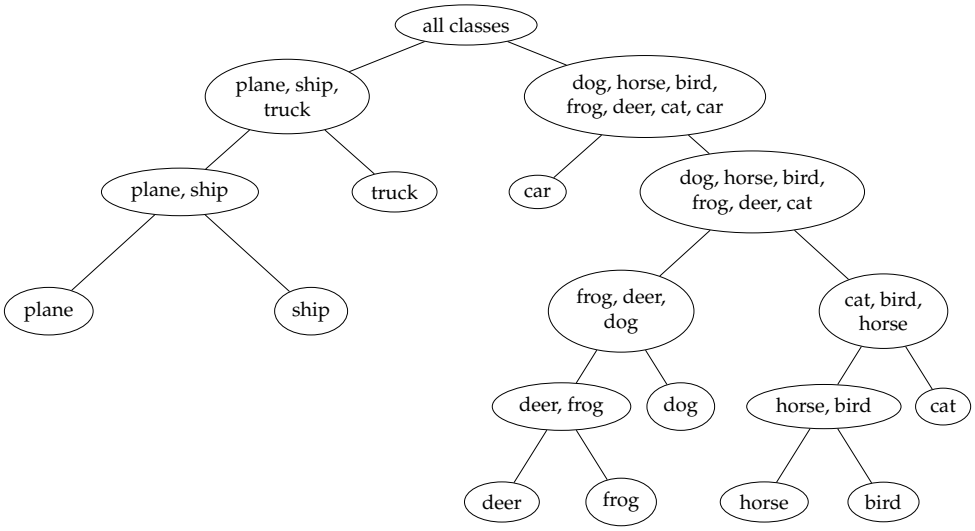
gradient of both plots is approximately one, which indicates that the proposed method does not add additional computational complexity to the problem. As logistic regression typically takes longer to converge than a decision tree takes to train in WEKA, the runtime is comparatively worse for logistic regression.

3.3.6 Case Study: *CIFAR-10*

To test how well our method adapts to other base learners, we trained nested dichotomies with convolutional networks (CNNs) as the base learners to classify the *CIFAR-10* dataset (Krizhevsky and Hinton, 2009). CNNs learn features from the data automatically, and perform well on high dimensional, highly correlated data such as images (Goodfellow et al., 2016). We implemented nested dichotomies in Python, and used Lasagne (Dieleman et al., 2015), a wrapper for Theano (Bastien et al., 2012; Bergstra et al., 2010), to construct the CNN base learners. The CNN that we used as the base learner is relatively simple: it has two convolutional layers with 32 3×3 filters each, one 3×3 maxpool layer with 2×2 stride after each convolutional layer, and one fully-connected layer of 128 units before a softmax layer.



(a) Random-pair selection.



(b) Centroid-based selection.

Figure 3.13: Nested dichotomies trained on *CIFAR-10*.

As discussed in Section 3.1, the centroids for datasets like *CIFAR-10* are not very descriptive, and as such, we expect clustering-based nested dichotomies with convolutional networks as the base learner to produce class splits that are not as well-founded as those built with random-pair selection. We present a visualisation of the nested dichotomy structures produced by both random-pair selection and the clustering approach for *CIFAR-10* in Figure 3.13. It can be seen that both methods produce a reasonable dichotomy structure, but there are some cases in which the random-pair method results in more intuitive splits. For example, the root node of the random-pair nested dichotomy splits the full set of classes

into the two natural subsets present (vehicles and animals), whereas the clustering-based tree omits the 'car' class from the left-hand subset. Two pairs of similar classes in the animal subset—'deer' and 'horse', and 'cat' and 'dog'—are kept together until near the leaves in the random-pair nested dichotomy, but are split relatively early in the clustering-based one. Of course, the quality of the nested dichotomy under random-pair selection is dependent on the initial pair of classes that is selected. If two classes that are similar to each other are selected to be the initial random-pair, the tree can end up with splits that make less intuitive sense.

3.3.7 Comparison to Recent Work

The majority of the research presented in this chapter was undertaken in early 2016, and published later in the same year (Leathart, Pfahringer, and Frank, 2016). There is more recent work on nested dichotomy construction by other authors, in which two new methods are compared to the random-pair method: best-of-K models (Melnikov and Hüllermeier, 2018) and a genetic algorithm-based approach called evolved nested dichotomies (Wever, Mohr, and Hüllermeier, 2018). These methods were described in Section 2.2.2. We briefly summarise and discuss their experimental performance to show how the random-pair method compares to this recent work and include critical discussion of the methods and their evaluation.

Best-of-K Models

Melnikov and Hüllermeier (2018) build nested dichotomies by training K independent nested dichotomies, and choose the one with the best performance on a held-out validation set. They consider both uniformly sampled trees (BoK) and balanced trees (BBoK). In their paper, they claim that this method is superior to the random-pair method when learning a

Table 3.2: Sign test comparing wins (W), draws (D) and losses (L) for random-pair nested dichotomies against best-of-K considering exceedance probability, when $K = 10$ and $K = 50$. Results are taken from Table 3 of (Melnikov and Hüllermeier, 2018).

	Best-of-10				Best-of-50			
	W	D	L	p-value	W	D	L	p-value
CART	13	0	14	0.5	10	1	16	0.1635
Logistic Regression	10	2	15	0.2122	7	4	16	**0.0466**
Decision Stump	12	1	14	0.4225	13	0	14	0.5

single nested dichotomy for prediction, showing experimental results for $K = 10$ and $K = 50$.[5]

Melnikov and Hüllermeier (2018) define and focus on a metric called *exceedance probability*, which is defined for a nested dichotomy construction method as the probability that a nested dichotomy produced by the method will perform worse than a randomly sampled one. A lower exceedance probability implies better nested dichotomies than uniform sampling, on average. It is interesting that despite the authors claims regarding the superiority of their method, when considering exceedance probabilities,[6] for two of three base learners investigated, there is actually insufficient evidence that taking the best of K uniformly sampled nested dichotomies (when $K = 10$ or $K = 50$) outperforms nested dichotomies constructed with the random-pair method. Table 3.2 shows a sign test for the exceedance probability of both methods. None of the comparisons are statistically significantly different at $p = 0.05$, except when $K = 50$ and logistic regression is used as the base learner. The same significance results are obtained when performing a sign test for classification accuracy (Table 3.3).

[5]They also propose a method based on agglomerative clustering (ACND), which yielded similar results to nested dichotomies based on clustering (NDBC) proposed by Duarte-Villaseñor et al. (2012).

[6]Stated in Table 3 of (Melnikov and Hüllermeier, 2018).

Table 3.3: Sign test comparing wins (W), draws (D) and losses (L) for random-pair nested dichotomies against best-of-K, considering classification accuracy, where $K = 10$ and $K = 50$. The values are based on estimated classification accuracies taken from Tables 5,7 and 9 of (Melnikov and Hüllermeier, 2018).

	Best-of-10				Best-of-50			
	W	D	L	p-value	W	D	L	p-value
CART	13	0	14	0.5	11	0	16	0.2210
Logistic Regression	10	0	17	0.1239	8	0	19	**0.0261**
Decision Stump	13	0	14	0.5	13	0	14	0.5

We also show critical difference plots ($p = 0.05$) for the results presented in Tables 5, 7 and 9 of (Melnikov and Hüllermeier, 2018) in Figure 3.14. Note that this research only considered individual nested dichotomies, so these results are not directly comparable to the results presented earlier in this chapter for ensemble methods. Melnikov and Hüllermeier's results are consistent with ours in that the performance of individual nested dichotomies constructed with the random-pair method (RPND) is not statistically significantly different from the clustering-based approach (NDBC) for any of the base learners. All of their additionally proposed methods (Bo10, Bo50, BBo10, BBo50 and ACND) perform equally well at $p = 0.05$ for decision trees and decision stumps. However, even when logistic regression is used as the base learner, the performance of their best models (Bo10, Bo50) is not statistically significantly superior to the random-pair method according to this significance test.

It is important to note that the computational overhead of building and evaluating K nested dichotomies in order to use the best-of-K method to build a single nested dichotomy is much greater than that of the random-pair method. Like the random-pair method, best-of-K scales linearly in the number of classes in terms of the number of base learners to be trained, but best-of-K has a much higher constant factor. When the random-pair method is used, $2(m - 1)$ base learners must be built, half of them only being trained on the data corresponding to the initially chosen

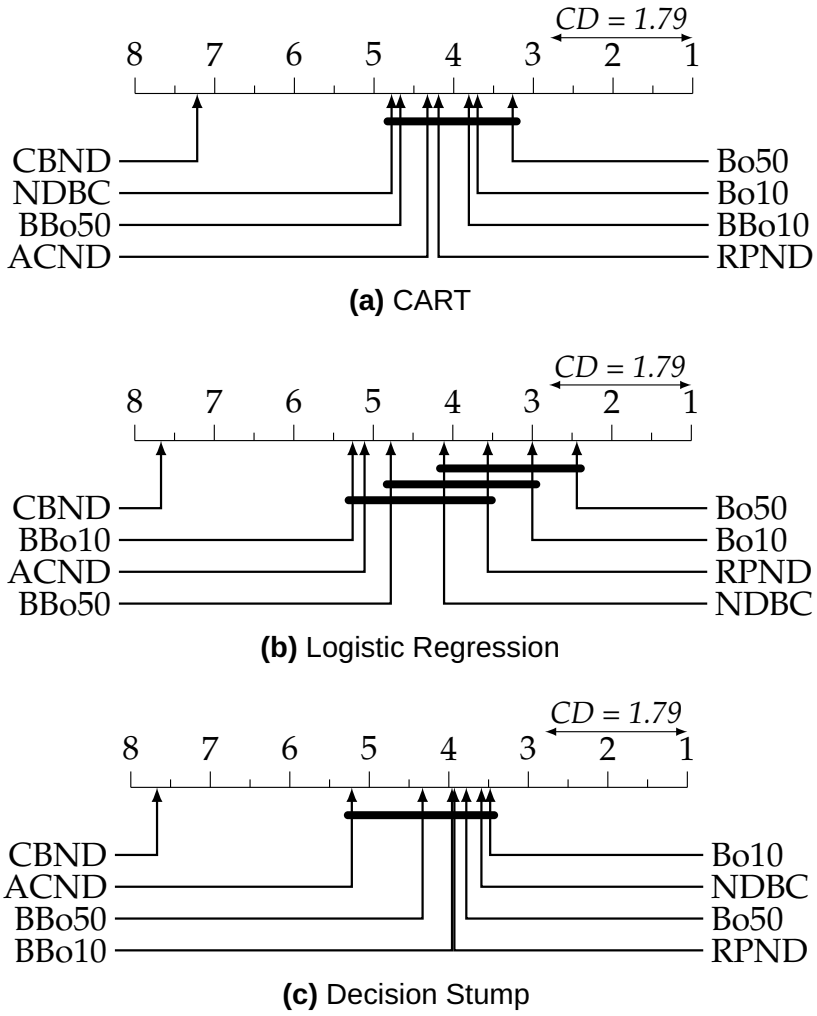


Figure 3.14: Average ranks of individual nested dichotomies, considering classification accuracy, built using different selection methods from (Melnikov and Hüllermeier, 2018).

random-pair, and the other half on the full data arriving at their corresponding internal nodes. On the other hand, best-of-K requires $K(m-1)$ internal models to be trained—each model being trained on the full data arriving at the internal node. Note that, as Melnikov and Hüllermeier (2018) mention, this constant factor can be mitigated by carrying out training in a parallel manner because each of the nested dichotomies is trained independently.

Given that the expected performance between the random-pair and best-of-K methods for almost all cases is very similar, with only minor improvements attained in some cases using best-of-K, one may argue that

Table 3.4: Sign test comparing statistically significant wins (W), draws (D) and losses (L) for random-pair nested dichotomies against the genetic algorithm-based method, considering classification accuracy of a bagged ensemble. Results are taken from Tables 2, 3 and 4 of (Wever, Mohr, and Hüllermeier, 2018).

	W	D	L	p-value
C4.5	1	27	7	**0.0352**
Logistic Regression	3	23	9	0.0730
Decision Stump	5	24	6	0.3016

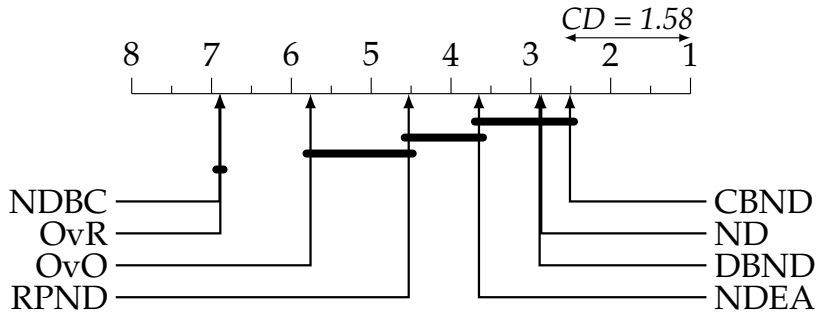
the random-pair method is a more attractive option for real world practitioners of data science. It appears to provide a better trade-off between predictive performance, runtime, and ease of implementation.

Evolved Nested Dichotomies

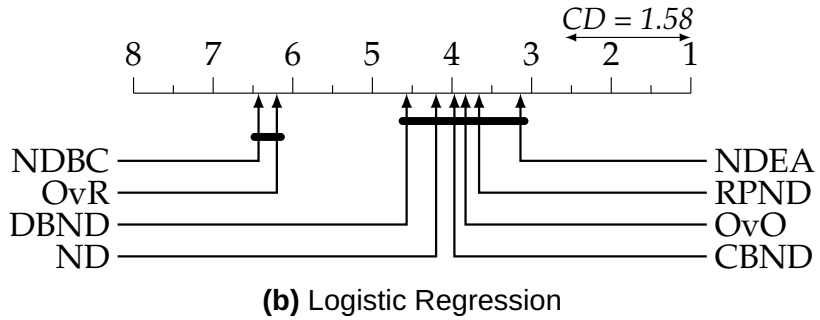
Wever, Mohr, and Hüllermeier (2018) use genetic algorithms to build individual nested dichotomies, as discussed in Section 2.2.2. A population of candidate nested dichotomies is evolved, and the best tree (based on performance on a held-out validation set) is selected. In order to produce an ensemble of K nested dichotomies, K independent populations can be evolved, taking the best performing tree for each population as above. Wever, Mohr, and Hüllermeier (2018) present experimental results obtained by applying this approach in conjunction with bagging, claiming to have a slight edge over nested dichotomies constructed with the random-pair method, especially for simple base learners.[7]

Table 3.4 shows sign tests comparing the random-pair method to evolved nested dichotomies when employed in a bagged ensemble, based on the results presented in Tables 2, 3 and 4 of (Wever, Mohr, and Hüllermeier, 2018). Wever, Mohr, and Hüllermeier (2018) report the statistical significance of their experimental results, so we use statistically significant wins and losses for our sign test, with each non-significant result

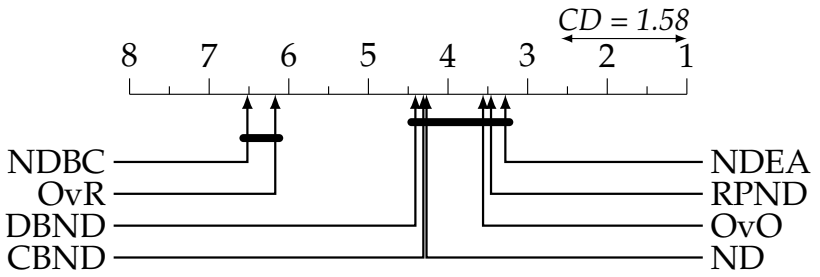
[7] One-vs-one (OvO), one-vs-rest (OvR) and data-balanced nested dichotomies (DBND) are also included in their experiments (Wever, Mohr, and Hüllermeier, 2018). These methods were described in Chapter 2.



(a) C4.5



(b) Logistic Regression



(c) Decision Stump

Figure 3.15: Average ranks of ensembles of ten bagged nested dichotomies, considering classification accuracy, built using different selection methods from (Wever, Mohr, and Hüllermeier, 2018).

counting as a draw. Of the three base learners, namely logistic regression, C4.5 and decision stumps, C4.5 is the only base learner for which we obtain a significant result at $p = 0.05$.

We also present critical difference plots in Figure 3.15 for the results presented in Tables 2, 3 and 4 of (Wever, Mohr, and Hüllermeier, 2018). Under this testing framework, the random-pair method appears to perform just as well as evolved nested dichotomies (NDEA) at $p = 0.05$ for each base learner, but it should be noted that the genetic-algorithm-based

approach always has a slightly higher average rank. When logistic regression and decision stumps are used as base learners, the random-pair method appears in the top-ranked group, corroborating the results from the sign tests in Table 3.4.

One comment made by Wever, Mohr, and Hüllermeier (2018) is that when C4.5 is used as the base learner for bagged nested dichotomies, some datasets yielded significant differences that were inconsistent with the results in (Leathart, Pfahringer, and Frank, 2016). Specifically, some test cases from their experiments show their method significantly outperforming the random-pair method while exhibiting significant degradation compared to uniform sampling. Referring to bagged ensembles of nested dichotomies with C4.5 base learners, Wever, Mohr, and Hüllermeier (2018) state

> "Considering RPND and ND, although NDEA is performing clearly better than RPND, it performs significantly worse than ND on several datasets, most probably due to the effect of overfitting the respective data. This observation is somewhat surprising, as it contradicts the results reported in [12],[8] where no such significant degradations of RPND have been mentioned."

The only datasets exhibiting this in their experiments that also appear in ours are *mfeat-karnuhen* and *pendigits*. In both cases, we report uniform sampling to have a higher average accuracy than the random-pair method so there is apparently no contradiction, although the differences are not statistically significant in our experiments, ostensibly due to the different significance test employed (Mann-Whitney U-test (Mann and Whitney, 1947) vs corrected resampled paired t-test (Nadeau and Bengio, 2000)). An assumption made by the Mann-Whitney U-test is that the samples from the different populations are independent. When considering the performance of learning algorithms, this is not the case as each pair

[8] Referring to (Leathart, Pfahringer, and Frank, 2016).

of samples contains performance estimates from the same dataset. The *corrected* resampled t-test used to establish statistical significance in our experiments accounts for this dependence.

Similarly to the best-of-K approach, building an ensemble of evolved nested dichotomies incurs a significant computational overhead, the magnitude of which is not discussed in-depth by Wever, Mohr, and Hüllermeier (2018). Each population is independent, so they can be evolved in parallel. Nevertheless, even training one nested dichotomy in this fashion is substantially slower than training an ensemble using the random-pair method. In the experiments performed by Wever, Mohr, and Hüllermeier (2018), they use population sizes of 16, and stop evolving after a maximum of 200 generations. The evolution is stopped early if the best performing nested dichotomy structure does not change after 15 generations.[9] This means that a minimum of $16 \times 15 = 240$ and a maximum of $16 \times 200 = 3200$ nested dichotomies must be trained and evaluated on the validation set.[10]

While not without merit, we again argue that the genetic-algorithm-based approach is a less attractive option than the random-pair method for practitioners of data science in real-world settings. The expected predictive performance of this method is usually not significantly different to the random-pair method, and implementing the training mechanism is much more difficult for evolved nested dichotomies. Moreover, the computational cost of training is many orders of magnitude higher, even when each candidate population is evolved in parallel.

[9] A timeout condition for early stopping is also included by Wever, Mohr, and Hüllermeier, which we ignore for the purposes of this discussion.

[10] This number could potentially be reduced by caching the trained base learners from previous generations *à la* (Dong, Frank, and Kramer, 2005). Wever, Mohr, and Hüllermeier do not mention whether this is being performed in their implementation.

3.4 Conclusion

In this chapter, we have proposed a semi-random method of class subset selection in ensembles of nested dichotomies, where the class selection is directly based on the ability of the base learner to separate classes. The method non-deterministically produces easily separable class splits, which not only improves the accuracy over random methods for a single classifier, but also for ensembles of nested dichotomies. The method outperforms previously published non-random methods when nested dichotomies are used in a bagged ensemble and an ensemble boosted with MultiBoost, and otherwise gives comparable results. For every set of experiments in our results, random-pair nested dichotomies either have the best average rank, or are statistically indistinguishable at $p = 0.05$ from the top-ranked method. The method also compares competitively for most base learners to some recently published alternative approaches for selecting nested dichotomies (Melnikov and Hüllermeier, 2018; Wever, Mohr, and Hüllermeier, 2018) while being substantially more efficient to train.

Chapter 4

Nested Dichotomies with Multiple Subset Evaluation

Going further with randomised subset selection methods

There exist several methods for selecting the structure of a nested dichotomy, each with various advantages, some of which were evaluated in detail in the last chapter. This chapter describes a simple method for improving the performance of nested dichotomies for any randomised subset selection method while retaining the specific advantages of the method in question.

Relevant Publications

Parts of the work presented in this chapter are included in "Ensembles of Nested Dichotomies with Multiple Subset Evaluation", Leathart, Frank, et al. (2019a). This chapter also includes additional discussion of sampling uniform nested dichotomies, and the differences between it and random subset selection.

Software

The software used for the experiments in this chapter is available in the WEKA package manager in the `nestedDichotomies` package.

4.1 Building Nested Dichotomies

For non-trivial multi-class problems, the space of potential nested dichotomies is very large. An ensemble classifier can be formed by choosing suitable decompositions from this space. More advanced methods for building nested dichotomies restrict the space to sample from with the aim of excluding less desirable structures by some metric have been reviewed in Section 2.2.1. These methods can be split into three groups: top-down subset selection methods, bottom-up methods, and full structure sampling methods.

In the original formulation of ensembles of nested dichotomies, decompositions are sampled with uniform probability (Frank and Kramer, 2004). Several top-down subset selection methods have since been proposed, including class-balanced selection (Dong, Frank, and Kramer, 2005), selection based on clustering (Duarte-Villaseñor et al., 2012), and random-pair selection (see Chapter 3). Recently, additional sampling methods have been proposed, such as using bag-of-K models (Melnikov and Hüllermeier, 2018) and genetic algorithms (Wever, Mohr, and Hüllermeier, 2018). Melnikov and Hüllermeier (2018) also experimented with bottom-up agglomerative clustering methods but found the performance to be inferior to bag-of-K and random-pair selection.

4.1.1 Uniformly Sampling Nested Dichotomies: An Interlude

In this chapter, we describe a simple method for improving nested dichotomies that are constructed with top-down subset selection techniques. We refer to the case where, at each split node k, the left and

right subsets $\mathcal{C}_{kl}$ and $\mathcal{C}_{kr}$ are sampled uniformly from the set of partitions of $\mathcal{C}_k$ of size two as 'random subset selection' in this chapter. Note that (perhaps counter-intuitively) this is not identical to the case that nested dichotomy structures are uniformly sampled. Consider the case where a nested dichotomy is being constructed for a four class problem where $\mathcal{C} = \{a, b, c, d\}$. By adopting a top-down approach, there are seven equally-likely choices for the first split:

$$\big(\{a, b\}, \{c, d\}\big),\ \big(\{a, c\}, \{b, d\}\big),\ \big(\{a, d\}, \{b, c\}\big),$$
$$\big(\{a\}, \{b, c, d\}\big),\ \big(\{b\}, \{a, c, d\}\big),\ \big(\{c\}, \{a, b, d\}\big) \text{ and } \big(\{d\}, \{a, b, c\}\big).$$

For the three balanced options, the tree is already completely specified, i.e., there is only one option for both of the subtrees because there are only two classes available for each one and empty class subsets are not allowed. However, for the remaining four non-balanced options there are three possible subtrees for each of the right subsets. Therefore, the balanced options in this case are three times as likely as the imbalanced options. This is in contrast to uniform sampling, where all final distinct structures are equally likely.

In general, the less "balanced" a tree is, the less likely it is to be sampled under a random subset selection scheme. $T(m)$ grows factorially with m[1] (Frank and Kramer, 2004), so

$$T\left(\frac{m}{2}\right)T\left(\frac{m}{2}\right) \ll T(1)T(m-1). \tag{4.1}$$

In other words, there are fewer possible subtrees for balanced splits than there are for imbalanced ones, making them more likely to be selected. This inequality becomes greater for problems with more classes, although one can view this as a positive feature, as more balanced trees are generally faster to train (Dong, Frank, and Kramer, 2005), and result in faster

[1]To be more precise, it actually grows by the *double factorial* ($m!!$), defined for even m as the product of all even numbers less than or equal to m but greater than or equal to two, and for odd m as the product of all odd numbers less than or equal to m but greater than or equal to one.

Algorithm 5 Naïve method for sampling class subsets randomly.
Input: Set of classes $\mathcal{C}$

1: Shuffle $\mathcal{C}$
2: Uniformly sample an integer s in interval $[1, |\mathcal{C}|)$
3: Set $\mathcal{C}_l$ as $[\mathcal{C}^{(1)} \dots \mathcal{C}^{(s)}]$
4: Set $\mathcal{C}_r$ as $[\mathcal{C}^{(s+1)} \dots \mathcal{C}^{(|\mathcal{C}|)}]$
5: Output class subsets $\mathcal{C}_l, \mathcal{C}_r$

predictions if a heuristic tree search algorithm is utilised (Dembczyński et al., 2016).

Uniformly Sampling Class Subsets

Interestingly, uniformly selecting two class subsets from $\mathcal{C}_k$ at some node k is also not as straightforward as one would hope. A naïve, yet efficient approach to selecting two subsets is described in Algorithm 5. In short, the set of classes is shuffled, and a split point is chosen at random. Again, perhaps counter-intuitively, this does not result in uniform sampling. Balanced class subsets are less likely to be sampled than imbalanced ones.

In order to achieve true uniform sampling of class subsets from $\mathcal{C}_k$, one can create a Boolean mask $\mathbf{m}$ where the 1 elements indicate classes that belong to $\mathcal{C}_{kl}$ and 0 elements indicate those that belong to $\mathcal{C}_{kr}$. Each element $m \in \mathbf{m}$ must be drawn independently from a Bernoulli distribution with $p = 0.5$. However, the whole mask is subject to the constraint that not all elements are equal, as this would result in one of the class subsets being empty. Such a mask can easily be created by uniformly sampling an integer in the interval $[1, 2^{|\mathcal{C}_k|})$, and converting the integer to a bit string. By excluding the values 0 and $2^{|\mathcal{C}_k|}$ from the interval we can ensure that subsets contain at least one element while keeping the expected value of each element of $\mathbf{m}$ equal to 0.5.

4.2 Multiple Subset Evaluation

In existing class subset selection methods, at each internal node k, a single class split $(\mathcal{C}_{kl}, \mathcal{C}_{kr})$ of $\mathcal{C}_k$ is considered, produced by some splitting

function

$$S(\mathcal{C}_k) : \mathbb{N}^n \rightarrow \mathbb{N}^a \times \mathbb{N}^b \tag{4.2}$$

where $a + b = n$. This chapter's approach for improving the predictive power of nested dichotomies is a simple extension. We propose to, at each internal node i, consider λ subsets $\{(\mathcal{C}_{kl}, \mathcal{C}_{kr})_1 \ldots (\mathcal{C}_{kl}, \mathcal{C}_{kr})_\lambda\}$ and choose the split for which the corresponding model has the lowest training RMSE. RMSE is chosen over other measures such as classification accuracy because it is smoother and a very sensitive indicator of generalisation performance. Previously proposed methods with single subset selection can be considered a special case of this method where $\lambda = 1$.

Although conceptually simple, this method has several attractive qualities, which are now discussed.

Predictive Performance. By choosing the best of a series of models at each internal node, the overall performance is likely to improve, assuming the size of the sample space of nested dichotomies is not hindered to the point where ensemble diversity begins to suffer.

Generality. Multiple subset evaluation is widely applicable. If a subset selection method S has some level of randomness, then multiple subset evaluation can be used to improve the performance. One nice feature is that advantages pertaining to S are retained. For example, if class-balanced selection is chosen due to a learning problem with a very high number of classes, we can boost the predictive performance of the ensemble while keeping each nested dichotomy in the ensemble balanced.

Simplicity. Implementing multiple subset evaluation is very simple. Furthermore, the computational cost for evaluating multiple subsets of classes scales linearly in the size of the tuneable hyperparameter λ, making the tradeoff between predictive performance and training

time easy to navigate. Additionally, multiple subset evaluation has no effect on prediction times.

Higher values of λ give diminishing returns on predictive performance, so a value that is suitable for the computational budget should be chosen. When training an ensemble of nested dichotomies, it may be desirable to adopt a *class threshold*, where single subset selection will be used if fewer than a certain number of classes is present at an internal node. This reduces the probability that the same subtrees will appear in many ensemble members, and therefore reduces ensemble diversity. In the lower levels of the tree, the number of possible binary problems is relatively low (Figure 4.1).

A downside of using a measure like training RMSE is that it has the potential to overfit to the training data, especially with more powerful base learners such as decision trees. For this reason, we recommend only using this method with simple base learners like logistic regression.

4.2.1 Effect on Growth Functions

Performance of an ensemble of nested dichotomies relies on the size of the sample space of nested dichotomies, given an n-class problem, to be relatively large. Multiple subset evaluation removes the $\lambda - 1$ class splits that correspond to the worst-performing binary models at each internal node i from being able to be used in the tree. The effect of multiple subset evaluation on the growth function is non-deterministic for random selection, as the sizes of $\mathcal{C}_{kl}$ and $\mathcal{C}_{kr}$ affect the values of the growth function for the subtrees that are children of i. The upper bound occurs when all worst-performing splits isolate a single class, and the lower bound is given when all worst-performing splits are class-balanced, given that the number of classes present and value chosen for λ allow for this number of worst performing splits. Class-balanced selection, on the other hand, is affected deterministically as the size of $\mathcal{C}_{kl}$ and $\mathcal{C}_{kr}$ are the same for the same number of classes.

Growth functions for values of $\lambda \in \{1, 3, 5, 7\}$, for random, class-balanced and random-pair selection methods (for logistic regression), are plotted in Figure 4.1. The growth curves for random and class-balanced selection were generated using brute-force computational enumeration while the effect on random-pair selection is computed by adapting the estimated growth function obtained in Section 3.2.2:

$$T_{\lambda}^{(RP_{LR})}(m) = p^{(LR)}(m) T_{\lambda}^{(RP_{LR})}\left(\frac{m}{3} - (\lambda - 1)\right) T_{\lambda}^{(RP_{LR})}\left(\frac{2m}{3} - (\lambda - 1)\right). \tag{4.3}$$

4.2.2 Analysis of error

In this section, we provide a theoretical analysis showing that performance of each internal binary model is likely to be improved by adopting multiple subset evaluation. We also show empirically that the estimates of performance improvements are accurate, even when the assumptions are violated.

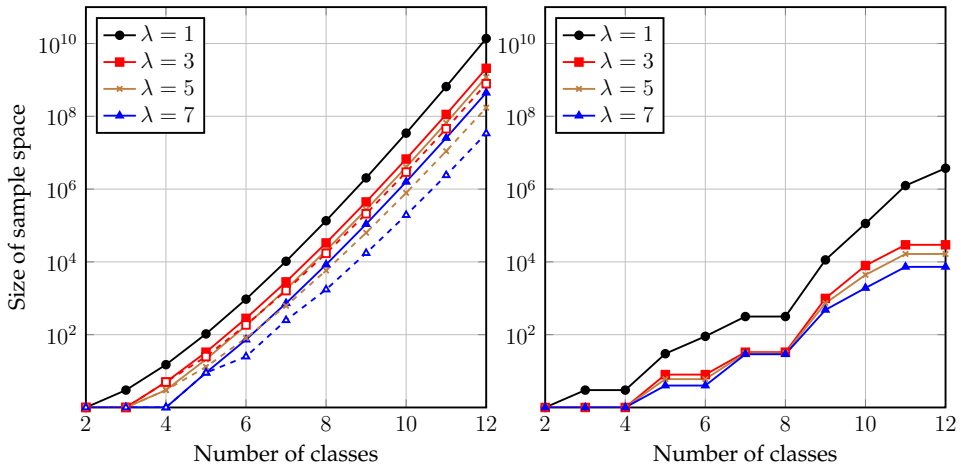
Let E be a random variable representing the training RMSE for some classifier for a given pair of class subsets $\mathcal{C}_{i1}$ and $\mathcal{C}_{i2}$, and assume $E \sim N(\mu, \sigma^2)$ for a given dataset under some class subset selection scheme.[2] For a given set of λ selections of subsets $\mathcal{S} = \{(\mathcal{C}_{i1}, \mathcal{C}_{i2})_1, \ldots, (\mathcal{C}_{i1}, \mathcal{C}_{i2})_\lambda\}$ and corresponding training RMSEs $\mathcal{E} = \{E_1, \ldots, E_\lambda\}$, let $\hat{E}_\lambda = min(\mathcal{E})$. There is no closed form expression for the expected value of $\hat{E}_\lambda$, the minimum of a set of normally distributed random variables, but an approximation is given by

$$\mathbb{E}[\hat{E}_\lambda] \approx \mu + \sigma\Phi^{-1}\left(\frac{1-\alpha}{\lambda - 2\alpha + 1}\right) \tag{4.4}$$

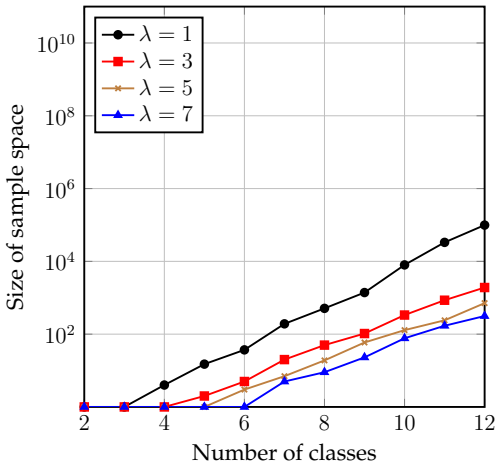
where $\Phi^{-1}(x)$ is the inverse normal cumulative distribution function (Royston, 1982), and the *compromise value* α is the suggested value for λ given by Harter (1961).[3]

[2] It is later shown that the normality assumption is not restrictive in practice.

[3] Appropriate values for α for a given λ can be found in Table 3 of (Harter, 1961).



(a) Random Selection **(b)** Class-balanced Selection



(c) Random-Pair Selection

Figure 4.1: Growth functions for random selection, class-balanced selection, and random-pair selection with multiple subset evaluation and $\lambda \in \{1, 3, 5, 7\}$. For random selection, solid lines indicate the upper bound, and dotted lines indicate the lower bound.

Figure 4.2 illustrates how this expected value changes when increasing values of λ from 1 to 4. The first two rows show the distribution of E and estimated $\mathbb{E}[\hat{E}_\lambda]$ on the UCI dataset *mfeat-fourier*, for a logistic regression model trained on 1,000 random splits of the class set $\mathcal{C}$. These rows show the training and testing RMSE respectively, using 90% of the data for training and the rest for testing. Note that as λ increases, the distribution of the train and test error shifts to lower values and the variance decreases.

This reduction in error affects each binary model in the tree structure,

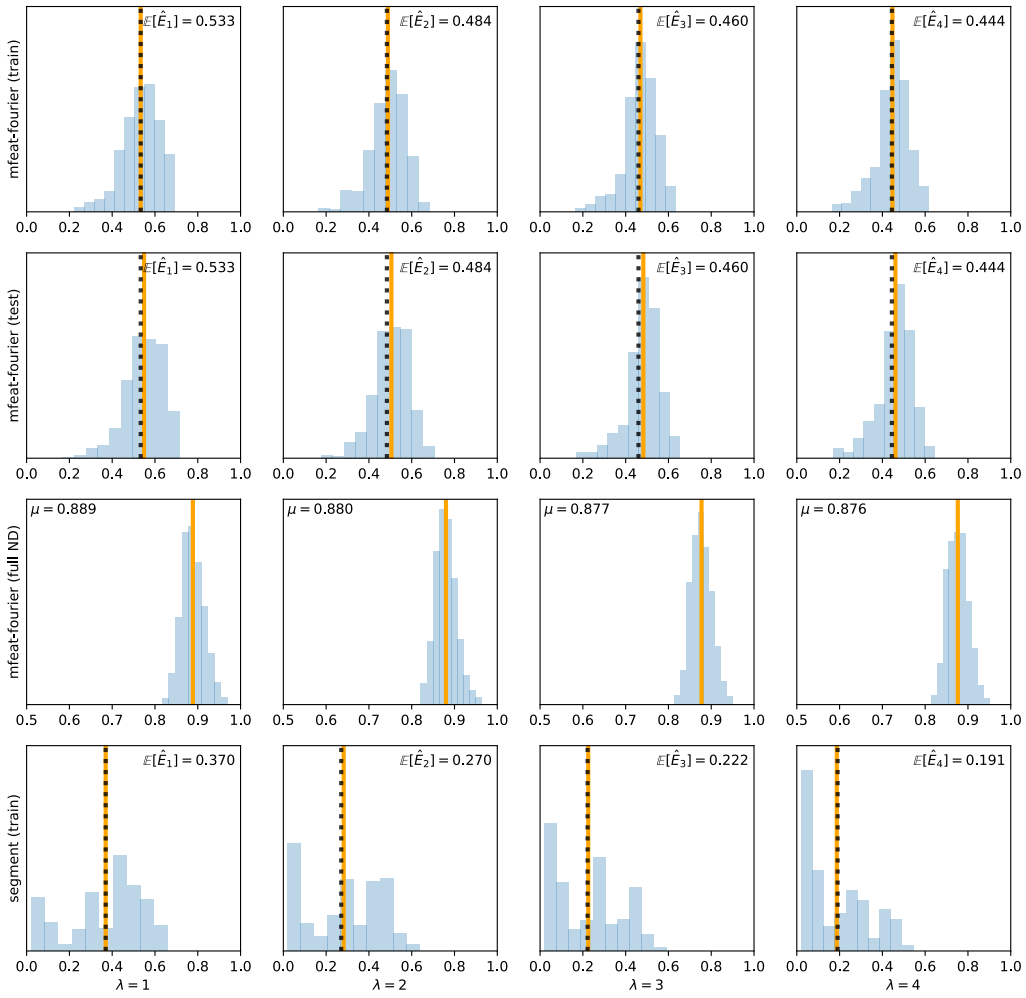


Figure 4.2: Empirical distribution of RMSE of logistic regression trained on random binary class splits, for values of λ from one to four. The shaded region indicates empirical histogram, the orange vertical line shows the empirical mean, and the black dotted vertical line is expected value, estimated from (4.4). Top two rows: train and test RMSE of logistic regression trained on random binary class splits of *mfeat-fourier* UCI dataset. For the test data, the approximated value of $\mathbb{E}[E_\lambda]$ is estimated from the mean and standard deviation of the train error. Third row: train RMSE of a nested dichotomy built with random splits and multiple-subset evaluation, trained on *mfeat-fourier* for different values of λ. Bottom row: train RMSE of logistic regression trained on random binary class splits of *segment* data.

so the effects accumulate when constructing a nested dichotomy. The third row shows the distribution of training RMSE of 1,000 nested dichotomies trained with multiple subset evaluation on *mfeat-fourier*, using logistic regression as the base learner, considering increasing values of λ. As expected, a reduction in error with diminishing returns is seen.

In order to show an example of how the estimate from (4.4) behaves when the error is not normally distributed, the distribution of E for logistic regression trained on the *segment* UCI data is plotted in the bottom row. This assumption is commonly violated in real datasets, as the distribution is often skewed towards zero error. As with the other examples, 1,000 different random choices for $\mathcal{C}_1$ and $\mathcal{C}_2$ were used to generate the histogram. Although the distribution in this case is not very well modelled by a Gaussian, the approximation of $\mathbb{E}[\hat{E}_\lambda]$ from (4.4) still closely matches the empirical mean. This shows that even when the normality assumption is violated, performance gains of the same degree can be expected. This example is not cherry picked; the same behaviour was observed on the entire collection of datasets used in this study. Similar plots for the training RMSE of the remaining datasets are given in Figures B.1 and B.2.

4.3 Experiments

We now consider generalisation performance of nested dichotomies and ensembles of nested dichotomies training with multiple subset evaluation. All experiments were conducted in WEKA 3.9 (Hall et al., 2009), and performed with 10 times stratified 10-fold cross validation. We use class-balanced nested dichotomies and nested dichotomies built with random-pair selection, with logistic regression as the base learner. The subset selection methods were chosen as they are the most prevalent non-deterministic subset selection methods. Logistic regression was chosen as the base learner due to its relative stability compared to other methods like decision trees. For both splitting methods, we compare values of $\lambda \in \{1, 3, 5, 7\}$ in a single nested dichotomy structure, as well as

in ensemble settings with bagging (Breiman, 1996) and AdaBoost (Freund and Schapire, 1996b). The default settings in WEKA were used for the `Logistic` classifier as well as for the `Bagging` and `AdaBoostM1` meta-classifiers. We evaluate performance on a collection of 15 commonly used datasets from the UCI repository (Lichman, 2013). Datasets used in our experiments and their characteristics are listed in Table 4.1.

Table 4.1: The datasets used in our experiments.

Dataset	Classes	Instances	Features
audiology	24	226	70
krkopt	18	28056	7
LED24	10	5000	25
letter	26	20000	17
mfeat-factors	10	2000	217
mfeat-fourier	10	2000	77
mfeat-karhunen	10	2000	65
mfeat-morph	10	2000	7
mfeat-pixel	10	2000	241
MNIST	10	70000	784
optdigits	10	5620	65
page-blocks	5	5473	11
pendigits	10	10992	17
segment	7	2310	20
usps	10	9298	257
vowel	11	990	14
yeast	10	1484	9

As in the previous chapter, we provide critical difference plots (Demšar, 2006) to summarise the results of the experiments. These plots present average ranks of models trained with differing values of λ. Models producing results that are not significantly different from each other at the 0.05 significance level are connected with a horizontal black bar. Full results tables showing RMSE for each experimental run, including significance tests, are shown in Tables 4.2–4.7.

4.3.1 Obtaining Probability Estimates from AdaBoost.M1

AdaBoost.M1 only produces hard classifications, as described by Freund and Schapire (1996a), but probability estimates are required in order to compute and compare RMSE. In WEKA, these probability estimates are obtained by

$$\mathbb{P}(y = c|\mathbf{x}) = \frac{e^{F_c(\mathbf{x}_i)}}{\sum_{j=0}^{m} e^{F_j(\mathbf{x})}}. \tag{4.5}$$

Here, each $F_j(\mathbf{x})$ is the sum of weights of the weak learners that predicted j:

$$F_j(\mathbf{x}) = \sum_{k=0}^{K} \mathbb{I}(\hat{y}_k = j)\alpha_k \tag{4.6}$$

where there are K weak learners each giving a prediction $\hat{y}_k$ and weighted by α_k. This is a multiclass generalisation of a result given by Friedman, Hastie, Tibshirani, et al. (2000), where AdaBoost is interpreted as an additive logistic regression model.

4.3.2 Individual Nested Dichotomies

Restricting the sample space of nested dichotomies through multiple subset evaluation is expected to have a greater performance impact on smaller ensembles than larger ones. This is because in a larger ensemble, a poorly performing ensemble member does not have a large impact on the overall performance. On the other hand, in a small ensemble, one poorly performing ensemble member can degrade ensemble performance significantly. In the extreme case, where a single nested dichotomy is trained, there is no need for ensemble diversity, so a technique for improving the predictive performance of an individual nested dichotomy should be effective. Therefore, we first compare the performance of single nested dichotomies for different values of λ.

Figure 4.3 shows critical difference plots for both subset selection methods. Class-balanced selection shows a clear trend that increasing λ

improves the RMSE, with the average rank for $\lambda = 1$ being exactly 4. For random-pair selection, choosing $\lambda = 3$ is shown to be statistically indistinguishable from $\lambda = 1$ while higher values of λ give superior results on average.

Tables 4.2 and 4.3 show the full results of our experiments with individual nested dichotomies, and compares the RMSE for $\lambda \in \{3, 5, 7\}$ to that of $\lambda = 1$. For class-balanced selection, there is a universal reduction in RMSE for all datasets as λ increases, with the majority of these reductions being statistically significant compared to $\lambda = 1$. The top performing model is $\lambda = 7$ in all cases except for *audiology* and *yeast*. There are no datasets for which predictive performance is degraded by adopting multiple subset evaluation. In the case of random-pair selection, the results are less homogeneous, but there are still many datasets with statistically significant reductions in RMSE. *Mfeat-fourier* and *mfeat-karhunen* see an increase in RMSE, but these are not statistically significant at $p = 0.05$.

4.3.3 Ensembles of Nested Dichotomies

Typically, nested dichotomies are utilised in an ensemble, so we investigate the predictive performance of ensembles of ten nested dichotomies with multiple subset evaluation, with bagging and AdaBoost employed as the ensemble methods.

Class Threshold

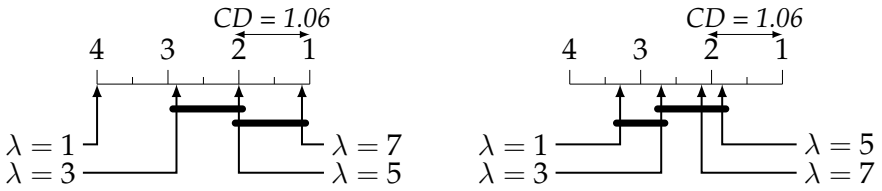
The number of binary problems is reduced when multiple subset evaluation is applied, which can have a negative effect on ensemble diversity, potentially reducing predictive performance. To investigate this, we built ensembles of nested dichotomies with multiple subset evaluation by introducing a *class threshold*, the number of classes present at a node required to perform multiple subset evaluation, and varying its value from one to seven. We plot the test RMSE, relative to having a class threshold of one, averaged over all the datasets for both ensemble methods in

Table 4.2: RMSE of individual class-balanced nested dichotomies for the range of values of λ.

Dataset	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$
audiology	0.1541 ± 0.03	**0.1462 ± 0.03**	0.1503 ± 0.02	0.1492 ± 0.03
krkopt	0.2128 ± 0.00	0.2123 ± 0.00	0.2117 ± 0.00 •	**0.2116 ± 0.00** •
led24	0.2200 ± 0.01	0.2101 ± 0.01 •	0.2066 ± 0.01 •	**0.2050 ± 0.01** •
letter	0.1603 ± 0.00	0.1534 ± 0.00 •	0.1500 ± 0.00 •	**0.1482 ± 0.00** •
mfeat-factors	0.1219 ± 0.02	0.1109 ± 0.02	0.1094 ± 0.02	**0.1066 ± 0.02**
mfeat-fourier	0.1997 ± 0.01	0.1967 ± 0.01	0.1960 ± 0.01	**0.1957 ± 0.01**
mfeat-karhunen	0.1543 ± 0.02	0.1397 ± 0.02 •	0.1388 ± 0.02 •	**0.1366 ± 0.02** •
mfeat-morph	0.2204 ± 0.02	0.2041 ± 0.01 •	0.2004 ± 0.01 •	**0.1956 ± 0.01** •
mfeat-pixel	0.1531 ± 0.02	0.1368 ± 0.02 •	0.1309 ± 0.02 •	**0.1291 ± 0.02** •
MNIST	0.1540 ± 0.01	0.1422 ± 0.01 •	0.1377 ± 0.01 •	**0.1358 ± 0.01** •
optdigits	0.1354 ± 0.02	0.1224 ± 0.01 •	0.1169 ± 0.01 •	**0.1148 ± 0.01** •
page-blocks	0.1210 ± 0.01	0.1130 ± 0.01 •	0.1120 ± 0.01 •	**0.1109 ± 0.01** •
pendigits	0.1622 ± 0.02	0.1405 ± 0.02 •	0.1339 ± 0.01 •	**0.1298 ± 0.01** •
segment	0.1599 ± 0.03	0.1354 ± 0.02 •	0.1245 ± 0.02 •	**0.1183 ± 0.02** •
usps	0.1407 ± 0.01	0.1275 ± 0.01 •	0.1249 ± 0.01 •	**0.1232 ± 0.01** •
vowel	0.2382 ± 0.01	0.2205 ± 0.02 •	0.2101 ± 0.02 •	**0.2045 ± 0.02** •
yeast	0.2401 ± 0.01	0.2392 ± 0.01	**0.2378 ± 0.01**	0.2378 ± 0.01

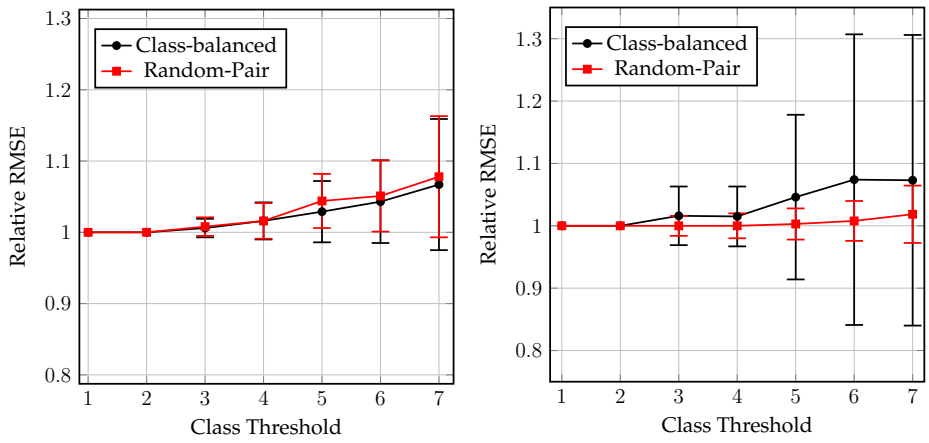
Table 4.3: RMSE of individual random-pair nested dichotomies for the range of values of λ.

Dataset	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$
audiology	0.1365 ± 0.02	0.1369 ± 0.03	**0.1320 ± 0.02**	0.1356 ± 0.02
krkopt	0.2094 ± 0.00	0.2092 ± 0.00	0.2090 ± 0.00 •	**0.2090 ± 0.00**
led24	0.2000 ± 0.01	0.1981 ± 0.01	0.1977 ± 0.01 •	**0.1973 ± 0.01** •
letter	0.1327 ± 0.00	0.1272 ± 0.00 •	0.1257 ± 0.00 •	**0.1246 ± 0.00** •
mfeat-factors	0.0923 ± 0.02	0.0860 ± 0.02	0.0855 ± 0.02	**0.0839 ± 0.01**
mfeat-fourier	**0.1936 ± 0.01**	0.1963 ± 0.01	0.1991 ± 0.01	0.2031 ± 0.01
mfeat-karhunen	**0.1333 ± 0.02**	0.1404 ± 0.02	0.1425 ± 0.02	0.1422 ± 0.01
mfeat-morph	0.1858 ± 0.01	0.1837 ± 0.01	**0.1827 ± 0.01**	0.1828 ± 0.01
mfeat-pixel	0.1274 ± 0.02	0.1178 ± 0.02	0.1183 ± 0.02	**0.1168 ± 0.02**
MNIST	0.1285 ± 0.01	0.1192 ± 0.00 •	0.1165 ± 0.00 •	**0.1159 ± 0.00** •
optdigits	0.1080 ± 0.01	0.1005 ± 0.01	**0.0986 ± 0.01**	0.0990 ± 0.01
page-blocks	0.1070 ± 0.01	0.1037 ± 0.01	**0.1033 ± 0.01**	0.1033 ± 0.01
pendigits	0.1204 ± 0.01	0.1036 ± 0.01 •	0.0993 ± 0.01 •	**0.0963 ± 0.01** •
segment	0.1109 ± 0.02	0.1017 ± 0.01	**0.0996 ± 0.01**	0.0999 ± 0.01
usps	0.1152 ± 0.01	0.1101 ± 0.01	**0.1089 ± 0.01**	0.1090 ± 0.00
vowel	0.1600 ± 0.02	0.1540 ± 0.02	0.1531 ± 0.02	**0.1527 ± 0.02**
yeast	0.2354 ± 0.01	0.2347 ± 0.01	0.2348 ± 0.01	**0.2348 ± 0.01**



(a) Class-balanced selection **(b)** Random-pair selection

Figure 4.3: Average ranks for individual nested dichotomies.



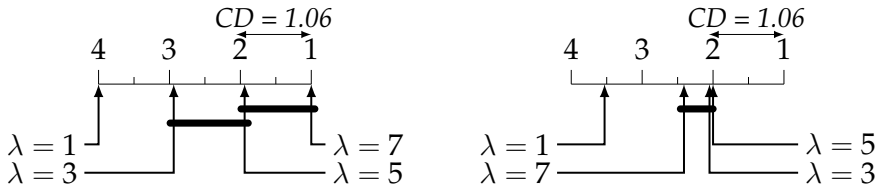
(a) Bagging **(b)** AdaBoost

Figure 4.4: Effect of changing the class threshold on RMSE for ensembles of nested dichotomies.

Figure 4.4. Interestingly, the RMSE increases monotonically in both cases, showing that the potentially reduced ensemble diversity does not have a negative effect on the RMSE for ensembles of this size. Therefore, we use a class threshold of one in our subsequent experiments. However, note that increasing the class threshold has a positive effect on training time, and the average degradation in performance is small, so it may still be useful to apply it in practice.

Number of Subsets

We now investigate the effect of λ when using bagging and boosting. Figure 4.5 shows critical difference plots for bagging. Both subset selection methods improve when utilising multiple subset selection. When class-balanced selection is used, as was observed for single nested dichotomies, the average ranks across all datasets closely correspond to the integer values, showing that increasing the number of subsets evaluated consistently



(a) Class-balanced selection **(b)** Random-pair selection

Figure 4.5: Average ranks for bagged ten nested dichotomies.

improves performance. For random-pair selection, a more constrained subset selection method, each value of $\lambda > 1$ is statistically indistinguishable from one another and superior to the single subset case.

Tables 4.4 and 4.5 show the full results of our experiments with bagged ensembles of nested dichotomies. When multiple subset evaluation is employed, there is a statistically significant RMSE reduction in almost all datasets when class-balanced selection is used, and several datasets also see significant improvements when random-pair selection is used. One dataset (*mfeat-fourier*) results in significantly greater RMSE, presumably due to overfitting.

The critical difference plot in Figure 4.6a shows that nested dichotomies in conjunction with AdaBoost are significantly improved by increasing the number of subsets sufficiently when class-balanced nested dichotomies are used. This can also be seen in Table 4.6, which shows that for most datasets, the best results are obtained by using a value of λ greater than one. Results are less consistent for random-pair selection, reflected in the critical differences plot (Figure 4.6b), which shows single subset evaluation statistically being indistinguishable from multiple subset selection for all values of λ, with $\lambda = 7$ performing markedly worse on average. Similar conclusions are reached from the full results in Table 4.7, with few statistically significant results in either direction. As RMSE is based on probability estimates, this may be in part due to poor probability calibration, which is known to affect boosted ensembles (Niculescu-Mizil and Caruana, 2005b).

Table 4.4: RMSE of an ensemble of 10 bagged class-balanced nested dichotomies for the range of values of λ.

Dataset	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$
audiology	0.1151 ± 0.01	0.1137 ± 0.01	0.1126 ± 0.01	**0.1120 ± 0.01**
krkopt	0.2112 ± 0.00	0.2104 ± 0.00 •	0.2101 ± 0.00 •	**0.2101 ± 0.00** •
led24	0.2070 ± 0.00	0.1999 ± 0.00 •	0.1983 ± 0.00 •	**0.1977 ± 0.00** •
letter	0.1489 ± 0.00	0.1400 ± 0.00 •	0.1366 ± 0.00 •	**0.1345 ± 0.00** •
mfeat-factors	0.0763 ± 0.01	0.0707 ± 0.01 •	0.0694 ± 0.01 •	**0.0684 ± 0.01** •
mfeat-fourier	0.1663 ± 0.01	0.1619 ± 0.01 •	0.1607 ± 0.01 •	**0.1598 ± 0.01** •
mfeat-karhunen	0.1098 ± 0.01	0.0981 ± 0.01 •	0.0947 ± 0.01 •	**0.0936 ± 0.01** •
mfeat-morph	0.2034 ± 0.01	0.1911 ± 0.01 •	0.1875 ± 0.01 •	**0.1859 ± 0.01** •
mfeat-pixel	0.0980 ± 0.01	0.0910 ± 0.01 •	0.0889 ± 0.01 •	**0.0883 ± 0.01** •
optdigits	0.0974 ± 0.01	0.0850 ± 0.00 •	0.0818 ± 0.00 •	**0.0803 ± 0.00** •
page-blocks	0.1114 ± 0.01	0.1067 ± 0.01 •	0.1055 ± 0.01 •	**0.1054 ± 0.01** •
pendigits	0.1259 ± 0.01	0.1060 ± 0.00 •	0.1002 ± 0.00 •	**0.0970 ± 0.00** •
segment	0.1283 ± 0.01	0.1090 ± 0.01 •	0.1027 ± 0.01 •	**0.1007 ± 0.01** •
usps	0.1070 ± 0.00	0.0984 ± 0.00 •	0.0957 ± 0.00 •	**0.0947 ± 0.00** •
vowel	0.2102 ± 0.01	0.1851 ± 0.01 •	0.1734 ± 0.01 •	**0.1655 ± 0.01** •
yeast	0.2361 ± 0.00	0.2355 ± 0.01	0.2348 ± 0.01 •	**0.2346 ± 0.01** •

Table 4.5: RMSE of an ensemble of 10 bagged random-pair nested dichotomies for the range of values of λ.

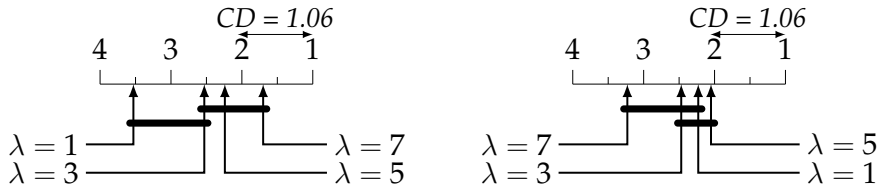
Dataset	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$
audiology	0.1082 ± 0.01	**0.1073 ± 0.01**	0.1083 ± 0.02	0.1083 ± 0.02
krkopt	0.2088 ± 0.00	0.2088 ± 0.00	0.2088 ± 0.00	**0.2088 ± 0.00**
led24	0.1963 ± 0.01	0.1959 ± 0.01	0.1959 ± 0.01	**0.1958 ± 0.01**
letter	0.1208 ± 0.00	0.1166 ± 0.00 •	0.1155 ± 0.00 •	**0.1151 ± 0.00** •
mfeat-factors	0.0629 ± 0.01	0.0594 ± 0.01 •	0.0584 ± 0.01 •	**0.0583 ± 0.01** •
mfeat-fourier	**0.1596 ± 0.01**	0.1612 ± 0.01	0.1628 ± 0.01 ◦	0.1635 ± 0.01 ◦
mfeat-karhunen	0.0924 ± 0.01	0.0924 ± 0.01	**0.0920 ± 0.01**	0.0923 ± 0.01
mfeat-morph	0.1817 ± 0.01	0.1815 ± 0.01	**0.1814 ± 0.01**	0.1815 ± 0.01
mfeat-pixel	0.0913 ± 0.01	0.0883 ± 0.01	**0.0876 ± 0.01** •	0.0876 ± 0.01 •
optdigits	0.0762 ± 0.00	**0.0717 ± 0.01** •	0.0726 ± 0.01 •	0.0731 ± 0.01 •
page-blocks	0.1023 ± 0.01	**0.1006 ± 0.01** •	0.1008 ± 0.01	0.1008 ± 0.01
pendigits	0.0938 ± 0.00	0.0838 ± 0.01 •	0.0807 ± 0.00 •	**0.0786 ± 0.00** •
segment	0.0984 ± 0.01	0.0934 ± 0.01 •	**0.0928 ± 0.01** •	0.0931 ± 0.01 •
usps	0.0920 ± 0.00	**0.0910 ± 0.00**	0.0919 ± 0.00	0.0921 ± 0.00
vowel	0.1294 ± 0.01	**0.1263 ± 0.01**	0.1265 ± 0.01	0.1270 ± 0.01
yeast	0.2336 ± 0.01	**0.2336 ± 0.01**	0.2336 ± 0.01	0.2337 ± 0.01

Table 4.6: RMSE of an ensemble of 10 boosted class-balanced nested dichotomies for the range of values of λ.

Dataset	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$
audiology	0.1152 ± 0.02	0.1144 ± 0.02	$\mathbf{0.1138 \pm 0.02}$	0.1146 ± 0.02
krkopt	0.2130 ± 0.00	0.2123 ± 0.00	0.2121 ± 0.00 •	$\mathbf{0.2119 \pm 0.00}$ •
led24	0.2617 ± 0.01	0.2588 ± 0.00	0.2581 ± 0.00	$\mathbf{0.2580 \pm 0.00}$ •
letter	$\mathbf{0.1619 \pm 0.01}$	0.1830 ± 0.01 ○	0.1872 ± 0.01 ○	0.1862 ± 0.00 ○
mfeat-factors	0.0698 ± 0.02	0.0646 ± 0.02	0.0650 ± 0.02	$\mathbf{0.0637 \pm 0.02}$
mfeat-fourier	0.1714 ± 0.01	0.1718 ± 0.01	$\mathbf{0.1712 \pm 0.01}$	0.1716 ± 0.01
mfeat-karhunen	0.0997 ± 0.01	0.0943 ± 0.01	0.0955 ± 0.02	$\mathbf{0.0934 \pm 0.01}$
mfeat-morph	0.2558 ± 0.02	0.2488 ± 0.01	0.2464 ± 0.01	$\mathbf{0.2463 \pm 0.01}$
mfeat-pixel	0.1003 ± 0.01	0.0963 ± 0.01	0.0958 ± 0.01	$\mathbf{0.0948 \pm 0.01}$
optdigits	0.0765 ± 0.01	0.0720 ± 0.01	0.0728 ± 0.01	$\mathbf{0.0714 \pm 0.01}$
page-blocks	0.1208 ± 0.01	0.1179 ± 0.01	$\mathbf{0.1175 \pm 0.01}$	0.1185 ± 0.01
pendigits	0.1015 ± 0.01	$\mathbf{0.0891 \pm 0.01}$ •	0.0894 ± 0.01 •	0.0894 ± 0.01 •
segment	0.1146 ± 0.02	$\mathbf{0.1115 \pm 0.02}$	0.1134 ± 0.01	0.1124 ± 0.02
usps	0.1028 ± 0.01	0.0994 ± 0.01	0.0990 ± 0.01 •	$\mathbf{0.0985 \pm 0.01}$ •
vowel	0.2559 ± 0.02	0.1921 ± 0.04 •	0.1509 ± 0.02 •	$\mathbf{0.1359 \pm 0.02}$ •
yeast	$\mathbf{0.2819 \pm 0.02}$	0.2871 ± 0.01	0.2830 ± 0.02	0.2864 ± 0.02

Table 4.7: RMSE of an ensemble of 10 boosted random-pair nested dichotomies for the range of values of λ.

Dataset	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$
audiology	0.1127 ± 0.02	0.1137 ± 0.02	0.1144 ± 0.02	$\mathbf{0.1117 \pm 0.02}$
krkopt	0.2095 ± 0.00	0.2094 ± 0.00	0.2092 ± 0.00	$\mathbf{0.2092 \pm 0.00}$
led24	0.2567 ± 0.00	0.2566 ± 0.00	$\mathbf{0.2565 \pm 0.00}$	0.2567 ± 0.00
letter	0.1784 ± 0.00	0.1766 ± 0.00 •	0.1764 ± 0.00 •	$\mathbf{0.1764 \pm 0.00}$ •
mfeat-factors	0.0618 ± 0.02	$\mathbf{0.0613 \pm 0.01}$	0.0618 ± 0.02	0.0632 ± 0.01
mfeat-fourier	$\mathbf{0.1723 \pm 0.01}$	0.1759 ± 0.01	0.1737 ± 0.01	0.1773 ± 0.01
mfeat-karhunen	$\mathbf{0.0937 \pm 0.01}$	0.0964 ± 0.01	0.0947 ± 0.01	0.0954 ± 0.01
mfeat-morph	0.2412 ± 0.00	$\mathbf{0.2410 \pm 0.00}$	0.2418 ± 0.00	0.2418 ± 0.00
mfeat-pixel	$\mathbf{0.0936 \pm 0.01}$	0.0939 ± 0.01	0.0938 ± 0.01	0.0940 ± 0.01
optdigits	$\mathbf{0.0714 \pm 0.01}$	0.0718 ± 0.01	0.0726 ± 0.01	0.0740 ± 0.01
page-blocks	0.1163 ± 0.01	0.1147 ± 0.01	$\mathbf{0.1143 \pm 0.01}$	0.1145 ± 0.01
pendigits	0.0895 ± 0.01	0.0893 ± 0.01	$\mathbf{0.0892 \pm 0.01}$	0.0897 ± 0.01
segment	0.1069 ± 0.01	0.1064 ± 0.02	$\mathbf{0.1062 \pm 0.01}$	0.1070 ± 0.02
usps	$\mathbf{0.1005 \pm 0.01}$	0.1015 ± 0.01	0.1025 ± 0.01	0.1035 ± 0.01
vowel	$\mathbf{0.1161 \pm 0.02}$	0.1204 ± 0.02	0.1240 ± 0.02	0.1260 ± 0.02 ○
yeast	0.2874 ± 0.01	0.2900 ± 0.01	$\mathbf{0.2874 \pm 0.01}$	0.2908 ± 0.00



(a) Class-balanced selection **(b)** Random-pair selection

Figure 4.6: Average ranks for ensembles of ten nested dichotomies, ensembled with AdaBoost.

4.3.4 Limitations

In the experiments in this chapter, we focus on the RMSE metric, which is a measure for the quality of the probability estimates. While not without merit because it directly measures the quality of the probability estimates produced, the usual metric of interest when considering nested dichotomies is classification accuracy. We originally considered classification accuracy in our experiments with multiple subset selection, but unfortunately we did not observe any statistically significant improvements in this setting.

4.4 Conclusion

Multiple subset selection in nested dichotomies can improve predictive performance while retaining the particular advantages of the subset selection method employed. We present an analysis of the effect of multiple subset selection on expected RMSE and show empirically that adopting the proposed technique can improve predictive performance, at the cost of a constant factor in training time.

The results of the experiments suggest that for class-balanced selection, performance can be consistently improved significantly by utilising multiple subset evaluation. For random-pair selection, $\lambda = 3$ yields the best trade-off between predictive performance and training time, but when AdaBoost is used to form an ensemble of nested dichotomies, multiple subset evaluation is not generally beneficial.

Chapter 5

Calibrating Nested Dichotomies

Minimising the accumulation of errors

Nested dichotomies provide a natural way to compute multiclass probability estimates from the binary classifiers they contain – the probability estimate for class c is obtained by multiplying the binary probability estimates along the path to the leaf node corresponding to c. This means that errors in binary probability estimates from poorly calibrated internal models can accumulate over the tree, resulting in poor predictive performance. Furthermore, this chapter shows that even when well-calibrated base learners are used, the entire nested dichotomy can exhibit poor calibration. In this chapter, we discuss calibration strategies for nested dichotomies and show that overall predictive performance can be improved substantially through the use of such techniques.

Relevant Publications

Parts of the work presented in this chapter are published in "On Calibration of Nested Dichotomies", Leathart, Frank, et al. (2019b). This chapter also includes nested dichotomies constructed with the random-pair

method and considers matrix scaling as an additional baseline and external calibration method. Different strategies of building multiclass reliability diagrams are also discussed.

Software

The software used for the experiments in this chapter is publically available at `http://github.com/timleathart/pynd`.

5.1 Motivation

Nested dichotomies, unlike some other hierarchical binary decomposition techniques (e.g., see Section 2.1.2), utilise probabilistic classifiers at the internal nodes. This gives rise to several advantages, such as higher accuracy (Beygelzimer, Langford, and Ravikumar, 2009) and efficient top-k predictions (Dembczyński et al., 2016). The use of probabilistic binary classifiers at internal nodes also provides a natural way to compute multiclass probability estimates from the binary probability estimates in the internal models: for a given instance $(\mathbf{x}, y)$, the probability estimate $\hat{p}^{(c)}$ that it belongs to class c from a nested dichotomy $\mathcal{T}$ is computed from the probability chain rule

$$\begin{aligned}\hat{p}^{(c)} &= p(y = c|\mathbf{x}) \\ &= \prod_{k \in \mathcal{T}} \mathbb{I}\{c \in \mathcal{C}_{kl}\} p(c \in \mathcal{C}_{kl}|\mathbf{x}, y \in \mathcal{C}_k) + \mathbb{I}\{c \in \mathcal{C}_{kr}\} p(c \in \mathcal{C}_{kr}|\mathbf{x}, y \in \mathcal{C}_k),\end{aligned} \tag{5.1}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function (Fox, 1997). Despite these advantages, the calculation in (5.1) can cause problems with probability calibration (see Section 2.3 for an introduction to probability calibration). If the binary models utilised are not well-calibrated, such as naïve Bayesian classifiers, then the probability estimates used in each term of the product will not be reliable, leading to bad multiclass predictions. Even when

internal models are well-calibrated, such as logistic regression, perfect calibration is typically not achievable in practice. This leads to small errors that can accumulate over the tree structure. For larger trees that are constructed for datasets with many classes, the average path length from root to leaf is large, which exacerbates these issues.

In order to combat these two sources of performance degradation, we propose to use probability calibration strategies. Specifically, we investigate calibrating the internal models of nested dichotomies when they are poorly calibrated, and also consider calibrating the entire nested dichotomy structure for both well-calibrated and poorly calibrated internal base classifiers. We refer to these as *internal calibration* and *external calibration* respectively. We hypothesise that the overall calibration, as well as classification accuracy, can be improved through the use of these techniques.

5.1.1 Internal Calibration

It seems reasonable that improving the calibration of internal models will result in superior quality probability estimates for the nested dichotomy, but is it possible to theoretically quantify this improvement? It turns out that reducing the binary negative log-likelihood (NLL) of any internal model by some amount δ strictly reduces the multiclass NLL of the nested dichotomy. Moreover, depending on the depth of the internal model being calibrated, the reduction in multiclass NLL can be as high as δ.

Proposition 1. *The NLL of an instance under a nested dichotomy is equal to the sum of NLLs of the instance under the binary models on the path from the root node to the leaf node.*

Proof. The NLL of an instance is given by

$$\text{NLL} = -\mathbf{y}_i \log \hat{\mathbf{p}}_i = -\log \hat{\mathbf{p}}_i^{(c)} \tag{5.2}$$

where $\hat{\mathbf{p}}_i^{(c)}$ is the probability estimate for the true class c. Let $\mathcal{P}_c$ be the set of internal nodes on the path from the root to the leaf corresponding to class c. Then, $\hat{\mathbf{p}}_i^{(c)}$ can be expressed as

$$\hat{\mathbf{p}}_i^{(c)} = \prod_{k \in \mathcal{P}_c} \tilde{y}_{ik}\hat{p}_{ik} + (1 - \tilde{y}_{ik})(1 - \hat{p}_{ik}) \tag{5.3}$$

where $\hat{p}_{ik} \in [0, 1]$ is the scalar prediction and $\tilde{y}_{ik} \in \{0, 1\}$ is the meta-label for instance i for the binary model at node k respectively. Because $\tilde{y}_{ik} \in \{0, 1\}$, it is equivalent to write

$$\hat{\mathbf{p}}_i^{(c)} = \prod_{k \in \mathcal{P}_c} \hat{p}_{ik}^{\tilde{y}_{ik}}(1 - \hat{p}_{ik})^{(1-\tilde{y}_{ik})}. \tag{5.4}$$

Plugging this into (5.2) yields

$$\begin{aligned} \text{NLL} &= -\log \prod_{k \in \mathcal{P}_c} \hat{p}_{ik}^{\tilde{y}_{ik}}(1 - \hat{p}_{ik})^{(1-\tilde{y}_{ik})} && (5.5) \\ &= -\sum_{k \in \mathcal{P}_c} \log \left(\hat{p}_{ik}^{\tilde{y}_{ik}}(1 - \hat{p}_{ik})^{(1-\tilde{y}_{ik})} \right) && (5.6) \\ &= -\sum_{k \in \mathcal{P}_c} \tilde{y}_{ik} \log \hat{p}_{ik} + (1 - \tilde{y}_{ik}) \log(1 - \hat{p}_{ik}), && (5.7) \end{aligned}$$

the sum of NLLs from the binary models $k \in \mathcal{P}_c$. □

It directly follows that reducing the binary NLL for the model at internal node k by some amount δ results in a reduction of the multiclass NLL by δ for each class corresponding to the leaf nodes that are descendants of node k. This means that a calibration resulting in a binary NLL reduction of δ for some internal node k reduces the multiclass NLL by $\delta(n_k/n)$, where n_k is the number of examples that belong to classes whose corresponding leaf nodes are descendants of k.

In our empirical investigation of the effect of internal calibration exhibited in what follows, we consider Platt scaling and isotonic regression as the internal calibration methods, rather than the more expensive and

expressive calibration methods mentioned in Section 2.3. The main reason for this is computational efficiency. Nested dichotomies have $m - 1$ internal models, where m is the number of classes, so the use of overly expressive calibration models can greatly increase training time. For example, Guo et al. (2017) found that Bayesian binning into quantiles (see Section 2.3.2) takes roughly two orders of magnitude more time than isotonic regression.

5.1.2 External Calibration

As well as calibrating each internal model, we also consider external calibration of the entire nested dichotomy. Even models like logistic regression are usually not perfectly calibrated in practice. We hypothesise that these minor miscalibrations accumulate as the nested dichotomy gets deeper, leading to more serious miscalibration in the overall nested dichotomy, which can be rectified by an external calibration model. Naturally, this effect is greater for problems with more classes, because the paths to leaf nodes will be longer.

As an illustrative investigation into the effect of nested dichotomy depth on their calibration, we built a nested dichotomy with logistic regression base learners for the ALOI dataset. Figure 5.1 shows reliability plots for this nested dichotomy that has been "cut-off" at incrementally increasing depths. A test example is considered to be classified correctly at depth d if its actual class is in the subset of classes $\mathcal{C}_k$ of the node k with highest probability amongst the nodes with maximum depth d. Limited to a depth of one, the nested dichotomy is simply a single binary logistic regression model, which exhibits good calibration. However, as the depth cut-off limit increases, it is clear that the nested dichotomy becomes increasingly *under-confident*, i.e., bins that have high accuracy often have low confidence (Figure 5.1, top two rows). This corresponds to the reliability curve sitting above the diagonal line. The ECE increases approximately linearly with the depth of the tree.

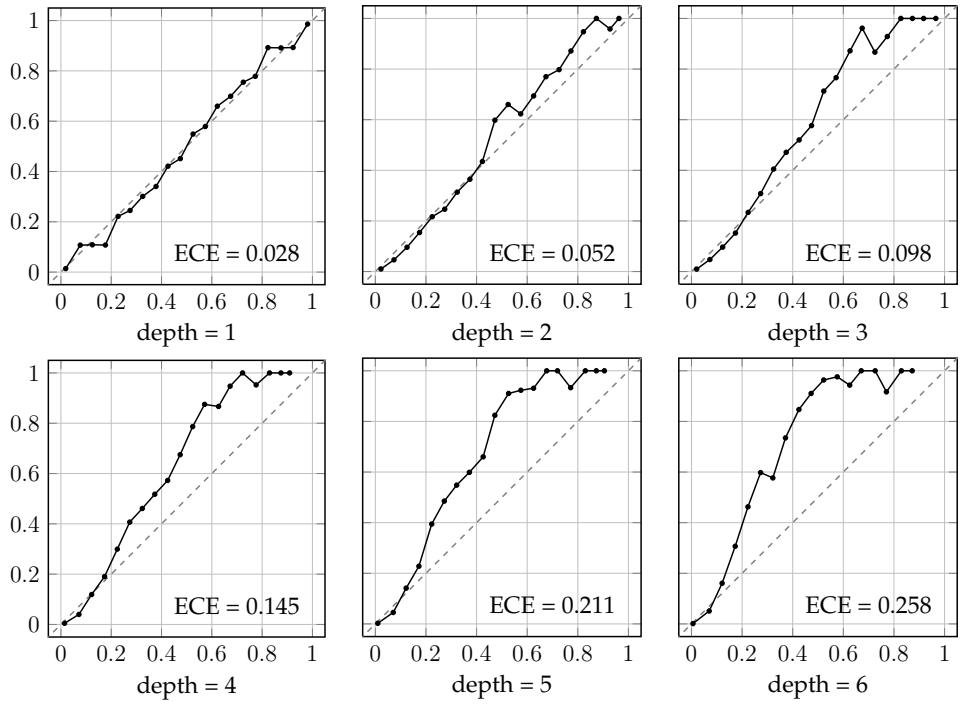


Figure 5.1: Reliability plots for a nested dichotomy with no external calibration, cut off at increasing depth. The nested dichotomy has logistic regression base learners, which individually, are well-calibrated. As the depth increases, the nested dichotomy becomes increasingly under-confident because of the effect of multiplying probabilities together.

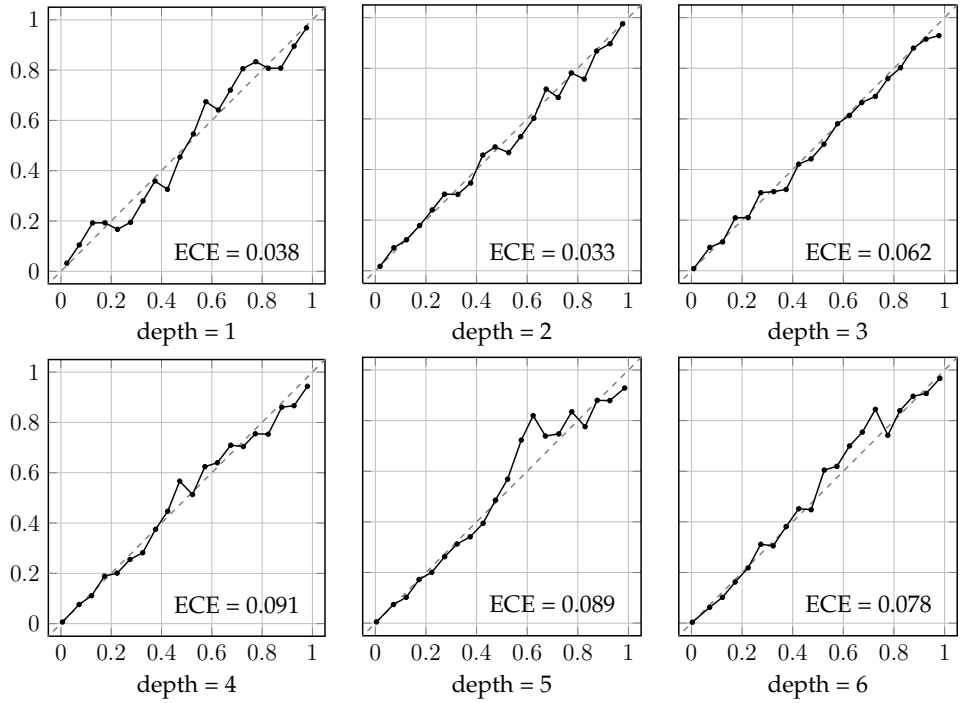


Figure 5.2: Reliability plots for the same nested dichotomy as Figure 5.1, but with external calibration applied.

This is adequately and efficiently compensated for by applying an external calibration model, such as vector scaling (Figure 5.1, bottom two rows). Vector scaling exhibits low complexity in the number of classes—only two parameters per class need to be estimated in the calibration model—making it suitable for problems with many classes typically handled by nested dichotomies. For externally calibrated nested dichotomies, the ECE initially increases linearly with the depth of the tree (although for $d > 1$, the ECE values are much lower than their uncalibrated counterparts). However, at $d = 5$, the ECE levels off and even begins to decrease slightly.

5.2 Multiclass Calibration Plots and ECE: An Interlude

In this thesis, we present reliability diagrams and ECE for several multiclass datasets. However, it is not entirely straightforward to reason about how these should be constructed. Reliability diagrams are described for binary problems in Section 2.3.3. For binary problems, the probability range $[0, 1]$ is split into K bins, and examples are assigned to these bins based on the estimated probability that they belong to the positive class. The x-coordinate of each point in the plot is the mid-point of the bin while the corresponding y-coordinate is the percentage of examples in that bin that belong to the positive class.

For multiclass problems there are two approaches: either we consider the entire estimated probability distribution, or we only consider the estimated confidence of the correct class. When considering the full distribution, each element of the estimated probability distribution is assigned to one of the bins. For the example shown in Table 5.1, and assuming we choose $K = 10$ equal-width bins, $\hat{\mathbf{p}}_i^{(1)}$—$\hat{\mathbf{p}}_i^{(3)}$, $\hat{\mathbf{p}}_i^{(6)}$—$\hat{\mathbf{p}}_i^{(10)}$ are assigned to the first bin, $\hat{\mathbf{p}}_i^{(5)}$ is assigned to the second bin, and $\hat{\mathbf{p}}_i^{(4)}$ is assigned to the

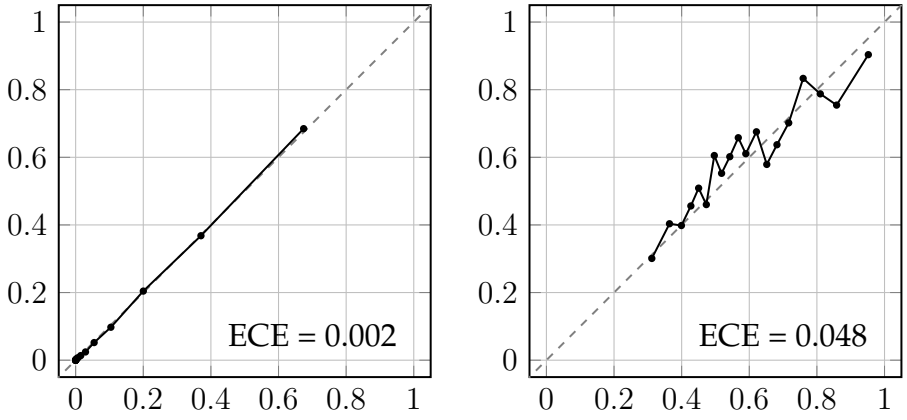
Table 5.1: An example estimated probability distribution for some instance i for a 10-class problem. The true class $y_i = 4$.

$\hat{\mathbf{p}}_i^{(1)}$	$\hat{\mathbf{p}}_i^{(2)}$	$\hat{\mathbf{p}}_i^{(3)}$	$\hat{\mathbf{p}}_i^{(4)}$	$\hat{\mathbf{p}}_i^{(5)}$	$\hat{\mathbf{p}}_i^{(6)}$	$\hat{\mathbf{p}}_i^{(7)}$	$\hat{\mathbf{p}}_i^{(8)}$	$\hat{\mathbf{p}}_i^{(9)}$	$\hat{\mathbf{p}}_i^{(10)}$
0.01	0.04	0.03	0.68	0.10	0.02	0.01	0.05	0.01	0.05

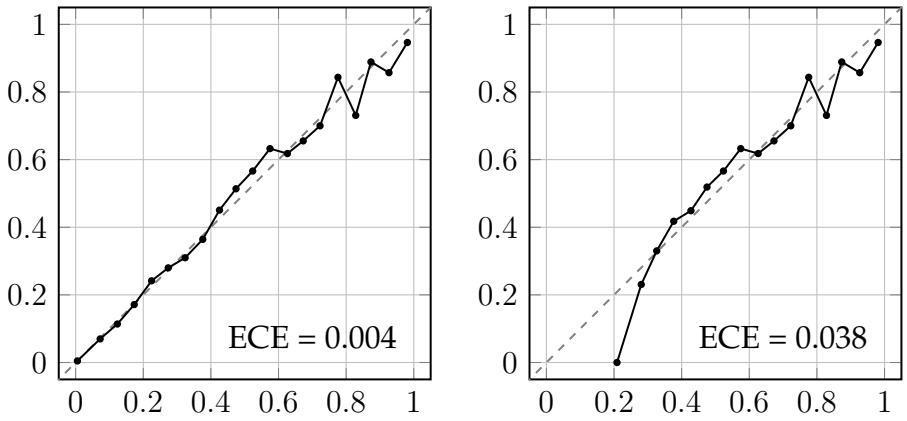
seventh bin. This approach is analogous to one-vs-rest in that each example contributes m terms to the sums used to calculate the accuracy and confidence for the bins. A drawback of this method is that as the number of classes grows, so does the number of elements with very low values in the distribution. Applying equal-frequency binning results in all bins having very low confidence while applying equal-width binning results in the first bin containing a very high number of samples, skewing the ECE calculations.

Alternatively, one can only consider the estimated probability of the true class in order to have a single estimate per example. While overcoming the problems faced by using the full distribution, there are still issues with this formulation when the number of classes is relatively low. By only taking the most likely class, the lowest possible estimated probability value is $\frac{1}{m}$. Equal-frequency binning is necessary for low numbers of classes, as lower-valued equal-width bins will all be empty, giving an ineffective visualisation of the calibration.

Both of these approaches with equal-frequency and equal-width binning are compared in Figure 5.3. The diagrams depict the calibration of predictions made by a nested dichotomy with logistic regression base learners, after being externally calibrated, for an artificial dataset with 16 classes. We use $K = 20$ bins in these visualisations. We argue that of these four options, equal-width binning in conjunction with the full estimated probability distributions generates reliability diagrams with the best representation of model calibration for multiclass problems, especially when the number of classes is low ($m \ll 100$). However, the ECE is too optimistic, due to the first bins containing the lion's share of the examples.



(a) Equal-frequency, full distribution **(b)** Equal-frequency, true class only



(c) Equal-width, full distribution **(d)** Equal-width, true class only

Figure 5.3: A comparison of different strategies for reliability diagrams and ECE a multiclass problems.

Therefore, in order to have a value for ECE that reflects the reliability diagram, we compromise by taking a standard, unweighted average over the bins:

$$\mathrm{ECE}_{\mathrm{MC}} = \frac{1}{K}\sum_{k=1}^{K}\Big|\mathrm{acc}(B_k) - \mathrm{conf}(B_k)\Big| \tag{5.8}$$

where the accuracy and confidence of each bin is defined in (2.41) and (2.42) respectively. ECE and reliability diagrams for multiclass datasets have been previously presented in the literature (Naeini, Cooper, and Hauskrecht, 2015; Guo et al., 2017; Lee, Lee, et al., 2018; Heo et al., 2018) but without discussion of respective advantages and disadvantages. Guo et al. (2017) uses the estimated probability of the true class with

equal-width bins. In their paper, reliability diagrams are only shown for CIFAR-100, which has sufficiently many classes to mostly avoid the issues discussed above. However, even with 100 classes, it is often the case that the first bin is empty in their reliability diagrams.

5.3 Experiments

In this section, we present experimental results obtained by calibrating nested dichotomies with different base classifiers on a series of datasets. The datasets we used in our experiments are listed in Table 5.2, and were chosen to span a range of numbers of classes.

In order to obtain performance estimates, we performed 10 times stratified 10-fold cross-validation for the datasets from the UCI repository while adopting the standard train/test splits for the larger datasets with a larger number of classes ($m > 50$).[1] The number of instances stated in Table 5.2 for the larger datasets are split into number of training and test instances. Standard deviations are given in parentheses, and the best result per row appears in bold face. The original *ODP* dataset contains 105,000 classes—we took the subset of the most frequent 5,000 classes to create *ODP-5K* for the purposes of this investigation. We also reduce the dimensionality to 1,000 when evaluating the performance on boosted trees, by using a Gaussian random projection (Bingham and Mannila, 2001).

5.3.1 Implementation Details

We implemented vector scaling (Guo et al., 2017) and nested dichotomies in Python 3.4, and used the implementations of the base learners, isotonic regression and Platt scaling available in `scikit-learn` (Pedregosa et al., 2011). To calibrate the base learners, we used the

[1]Note that in each fold and run of 10 times 10-fold cross-validation, a different random nested dichotomy structure is constructed. In the case of the larger datasets with defined train/test splits, the average of 10 randomly constructed nested dichotomies is reported instead.

Table 5.2: Datasets used in our experiments.

Name	Instances	Features	Classes
optdigits[1]	5,620	64	10
krkopt[1]	28,056	7	18
micromass[1]	571	1,301	20
letter[1]	20,000	16	26
devanagari[1]	92,000	1,000	46
RCV1[2]	15,564/518,571	47,236	53
sector[2]	6,412/3,207	55,197	105
ALOI[2]	97,200/10,800	128	1,000
ILSVR2010[3]	1,111,406/150,000	1,000	1,000
ODP-5K[4]	361,488/180,744	422,712	5,000

[1] UCI Repository (Lichman, 2013)
[2] LIBSVM Repository (Chang and Lin, 2011)
[3] ImageNet (Russakovsky et al., 2015)
[4] ODP (Bennett and Nguyen, 2009)

`CalibratedClassifierCV` wrapper from `scikit-learn`. Our nested dichotomy implementation initially determines the structure of the tree, before learning all of the base learners at the split nodes in parallel using the `multiprocessing`[2] and `joblib`[3] packages in Python.

In order to report NLL, it was necessary to produce full probability distributions over the large label spaces used in these experiments. The naïve approach of recursively exploring the tree (Algorithm 1) has memory complexity of $O(nm^2)$, which can quickly overwhelm systems even with substantial amounts of memory. To combat this, we devised a simple algorithm for obtaining predictions for a batch of test examples with memory complexity $O(nm)$, presented in Algorithm 6. In short, the binary predictions from each internal node are accumulated linearly, eliminating the need for recursion and storage of several nodes' predictions at once. Even though this is slightly more computationally complex than Algorithm 1, it turns out to be much faster in practice, presumably due to the accumulated prediction matrix being stored in the CPU cache. Additionally, if lines 4 and 5 can be completed atomically, then this algorithm

[2] `https://docs.python.org/3.4/library/multiprocessing.html`
[3] `https://pypi.org/project/joblib/`

Algorithm 6 Efficient Nested Dichotomy Predictions (Full Distribution)
Input: Nested dichotomy $\mathcal{T}$, Dataset $\mathbf{D}$

1: Initialise predictions $\hat{\mathbf{p}}$ as matrix of ones, shaped (n, m)
2: **for** $k \in \mathcal{T}$ **do**
3: Get binary predictions $\hat{\mathbf{p}}_k$ from internal classifier for $\mathbf{D}$
4: Multiply columns of $\hat{\mathbf{p}}$ corresponding to classes in $\mathcal{C}_{kl}$ by $\hat{\mathbf{p}}_k$
5: Multiply columns of $\hat{\mathbf{p}}$ corresponding to classes in $\mathcal{C}_{kr}$ by $(1 - \hat{\mathbf{p}}_k)$
6: **end for**
7: **return** $\hat{\mathbf{p}}$

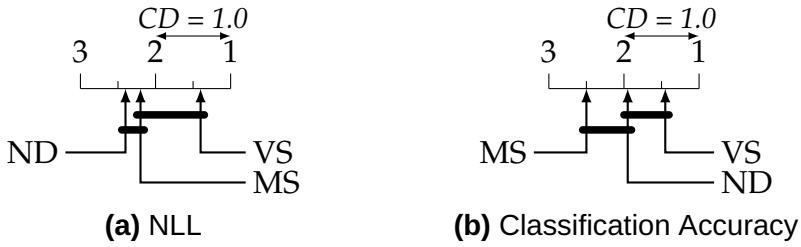
is embarrassingly parallel, as the predictions from the internal nodes can be evaluated in any order.

5.3.2 Well-Calibrated Base Learners

As shown in Figure 5.1, overall calibration of nested dichotomies can degrade as the depth of the tree increases, even if the base learners are well-calibrated. To further investigate the effects of external calibration on predictive performance when well-calibrated base learners are used, experiments were performed to determine the extent to which the classification accuracy and NLL are affected as well.

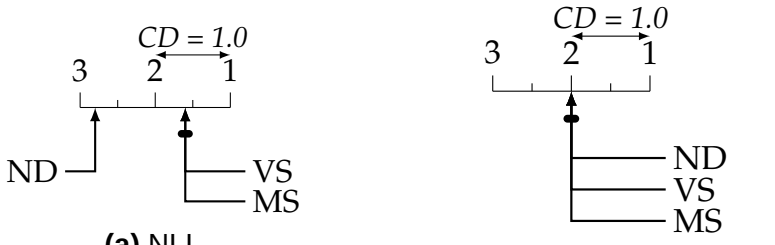
Tables 5.3 and 5.4 show the NLL and classification accuracy respectively of nested dichotomies with logistic regression, before and after external calibration is applied. Logistic regression models are known to be well-calibrated (Niculescu-Mizil and Caruana, 2005b). Vector scaling (VS) and matrix scaling (MS) (Guo et al., 2017) are used as the external calibration models, and compared to the uncalibrated baseline (ND). Figures 5.4a and 5.4b show the average ranks of these methods, considering NLL and classification accuracy respectively. A 90% stratified sample is used to build the nested dichotomy including the base classifiers and the remaining 10% is used train the external calibration model.

It is clear from Figure 5.4a that external calibration with vector scaling is beneficial for random nested dichotomies with logistic regression base learners. Vector scaling (VS) is the clear winner amongst the methods when considering NLL, and is not detrimental to classification accuracy



(a) NLL **(b)** Classification Accuracy

Figure 5.4: Average ranks of external calibration strategies for random nested dichotomies with logistic regression base learners.



(a) NLL **(b)** Classification Accuracy

Figure 5.5: Average ranks of external calibration strategies for random-pair nested dichotomies with logistic regression base learners.

(Figure 5.4b). Figure 5.4b shows that matrix scaling (MS) on average reduces the classification accuracy compared to nested dichotomies with no external calibration, and achieves comparable NLL, indicating that matrix scaling is not a useful approach for calibrating nested dichotomies with logistic regression base learners.

The results are similar for nested dichotomies built with random-pair selection, summarised in Figure 5.5. Vector scaling and matrix scaling both improve upon the baseline to an equal degree when considering NLL, but all methods have equal average ranks for classification accuracy. The full results in Tables 5.3 and 5.4 show similar behaviour between random nested dichotomies and those constructed with random-pair selection, with improvements to NLL almost always exhibited when applying external calibration and no clear pattern when considering classification accuracy.

Table 5.3: NLL of nested dichotomies with logistic regression, before and after external calibration is applied.

	Dataset	ND	VS	MS
Random	optdigits	0.302 ± 0.07	$\mathbf{0.301 \pm 0.08}$	0.855 ± 0.10
	krkopt	$\mathbf{2.426 \pm 0.04}$	2.530 ± 0.13	2.440 ± 0.19
	micromass	5.918 ± 1.83	$\mathbf{1.878 \pm 0.51}$	2.119 ± 0.10
	letter	1.502 ± 0.06	$\mathbf{1.435 \pm 0.08}$	2.600 ± 0.20
	devanagari	2.430 ± 0.11	$\mathbf{2.028 \pm 0.05}$	3.491 ± 0.05
	RCV1	1.004 ± 0.02	$\mathbf{0.584 \pm 0.01}$	0.589 ± 0.02
	sector	2.858 ± 0.01	$\mathbf{1.248 \pm 0.02}$	1.533 ± 0.03
	ALOI	3.604 ± 0.02	3.050 ± 0.03	$\mathbf{1.376 \pm 0.02}$
	ILSVR2010	6.442 ± 0.01	5.780 ± 0.01	$\mathbf{5.533 \pm 0.01}$
	ODP-5K	5.792 ± 0.01	$\mathbf{4.967 \pm 0.01}$	6.740 ± 0.00
Random Pair	optdigits	0.764 ± 0.03	0.221 ± 0.03	$\mathbf{0.218 \pm 0.03}$
	micromass	1.513 ± 0.13	2.180 ± 0.31	$\mathbf{1.501 \pm 0.19}$
	krkopt	2.263 ± 0.01	$\mathbf{1.584 \pm 0.01}$	1.597 ± 0.01
	letter	3.102 ± 0.10	2.354 ± 0.25	$\mathbf{1.128 \pm 0.10}$
	devnagari	2.758 ± 0.03	1.794 ± 0.04	$\mathbf{1.470 \pm 0.02}$
	RCV1	2.389 ± 0.01	$\mathbf{0.487 \pm 0.01}$	0.548 ± 0.03
	sector	4.501 ± 0.01	$\mathbf{1.212 \pm 0.01}$	1.487 ± 0.04
	ALOI	6.028 ± 0.01	$\mathbf{1.301 \pm 0.03}$	1.328 ± 0.03
	ILSVR2010	6.474 ± 0.01	5.710 ± 0.01	$\mathbf{5.533 \pm 0.01}$
	ODP-5K	4.911 ± 0.02	$\mathbf{4.808 \pm 0.02}$	6.846 ± 0.00

5.3.3 Poorly Calibrated Base Learners

Our experiments with poorly calibrated base learners are much more extensive, as we additionally evaluate several internal calibration strategies. Specifically, the following schemes are considered:

ND Baseline nested dichotomy with no additional calibration.

VS Nested dichotomy calibrated externally with vector scaling.

MS Nested dichotomy calibrated externally with matrix scaling.

iPS Nested dichotomy with Platt scaling applied to internal models.

VSiPS Nested dichotomy with Platt scaling applied to internal models, calibrated externally with vector scaling.

Table 5.4: Classification accuracy of nested dichotomies with logistic regression, before and after external calibration is applied.

	Dataset	ND	VS	MS
Random	optdigits	0.905 ± 0.02	$\mathbf{0.906 \pm 0.03}$	0.904 ± 0.03
	krkopt	$\mathbf{0.333 \pm 0.02}$	0.231 ± 0.10	0.216 ± 0.07
	micromass	$\mathbf{0.804 \pm 0.06}$	0.772 ± 0.05	0.725 ± 0.07
	letter	0.512 ± 0.03	$\mathbf{0.536 \pm 0.03}$	0.303 ± 0.07
	devanagari	0.428 ± 0.02	$\mathbf{0.428 \pm 0.02}$	0.134 ± 0.06
	RCV1	0.814 ± 0.01	$\mathbf{0.855 \pm 0.00}$	0.848 ± 0.00
	sector	0.848 ± 0.01	$\mathbf{0.867 \pm 0.00}$	0.778 ± 0.01
	ALOI	0.274 ± 0.01	0.331 ± 0.01	$\mathbf{0.778 \pm 0.00}$
	ILSVR2010	$\mathbf{0.063 \pm 0.00}$	0.053 ± 0.00	0.084 ± 0.00
	ODP-5K	0.189 ± 0.00	$\mathbf{0.228 \pm 0.00}$	0.073 ± 0.00
Random Pair	optdigits	$\mathbf{0.942 \pm 0.02}$	0.936 ± 0.01	0.940 ± 0.01
	micromass	$\mathbf{0.805 \pm 0.06}$	0.725 ± 0.05	0.691 ± 0.04
	krkopt	0.413 ± 0.01	$\mathbf{0.416 \pm 0.00}$	0.411 ± 0.00
	letter	0.182 ± 0.07	0.306 ± 0.07	$\mathbf{0.690 \pm 0.04}$
	devnagari	0.558 ± 0.02	0.553 ± 0.02	$\mathbf{0.610 \pm 0.01}$
	RCV1	0.810 ± 0.01	$\mathbf{0.871 \pm 0.00}$	0.857 ± 0.00
	sector	0.314 ± 0.01	$\mathbf{0.865 \pm 0.00}$	0.780 ± 0.01
	ALOI	0.685 ± 0.01	0.704 ± 0.00	$\mathbf{0.760 \pm 0.00}$
	ILSVR2010	$\mathbf{0.095 \pm 0.00}$	0.083 ± 0.00	0.084 ± 0.00
	ODP-5K	$\mathbf{0.259 \pm 0.00}$	0.257 ± 0.00	0.046 ± 0.00

MSiPS Nested dichotomy with Platt scaling applied to internal models, calibrated externally with matrix scaling.

iIR Nested dichotomy with isotonic regression applied to internal models.

VSiIR Nested dichotomy with isotonic regression applied to internal models, calibrated externally with vector scaling.

MSiIR Nested dichotomy with isotonic regression applied to internal models, calibrated externally with matrix scaling.

As previously, vector scaling and matrix scaling are used as the external calibration methods due to their relative efficiency and simple implementation. Three-fold cross validation is used to produce the training

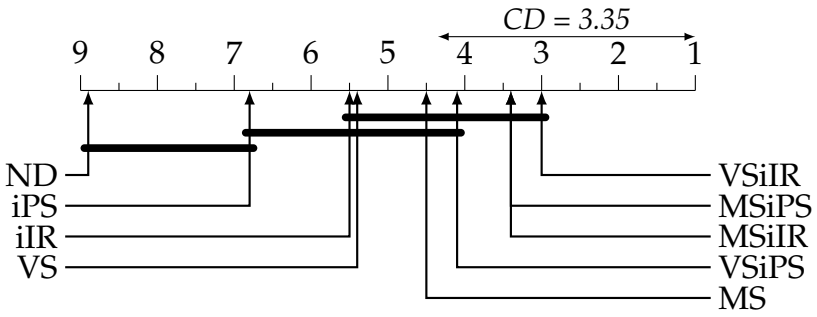
data for the internal calibration models, rather than splitting the training data. This is to ensure that each internal calibration model has a reasonable number of data points to train on, given that internal nodes near the leaves often have few training data available. We used the `CalibratedClassifierCV` wrapper in `scikit-learn` to achieve this in our experiments. In this implementation, the training data is split into three folds, and three base learners are trained—one on each combination of two of the three folds. Three calibration models are trained on the predictions made by the base learner on the remaining fold not used to train that particular base learner. At prediction time, the estimates from the three calibration models are averaged to produce the final calibrated binary probability estimates.

When external calibration is performed, 10% of the data is held out to train the external calibration model. Note that this means 10% less data is available to train the nested dichotomy and, where applicable, perform internal calibration.

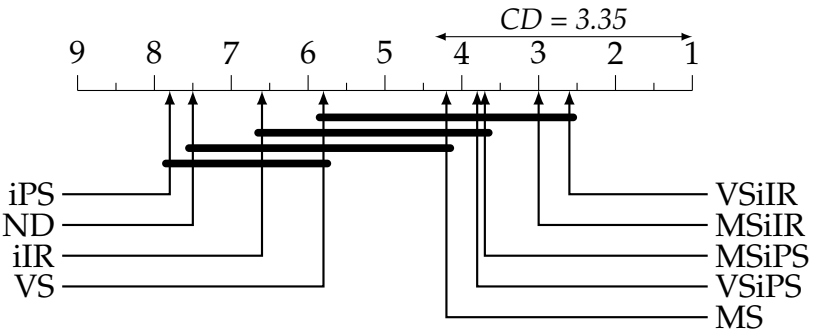
Naïve Bayes

Naïve Bayes is known to provide poor probability estimates, and is usually recommended to be calibrated with isotonic regression (Zadrozny and Elkan, 2001b). Naïve Bayes tends to push the estimated probabilities towards the extreme ends of the spectrum (Niculescu-Mizil and Caruana, 2005b). In our experiments, Gaussian naïve Bayes is applied for *optdigits*, *micromass*, *krkopt*, *letter* and *devanagari*, and multinomial naïve Bayes is used for *RCV1*, *sector*, *ALOI*, *ILSVR2010* and *ODP-5K* as they have sparse features. In other words, the data is assumed to be drawn from a d-dimensional Gaussian or multinomial distribution respectively.[4]

[4]Some of the text data has been scaled in such a way that it is not strictly multinomial with integer counts, by applying methods such as the commonly used term frequency-inverse document frequency technique (Luhn, 1957; Spärck Jones, 1972). Note that multinomial naïve Bayes has been shown to still work well in this scenario (Rennie et al., 2003).



(a) Random Nested Dichotomy

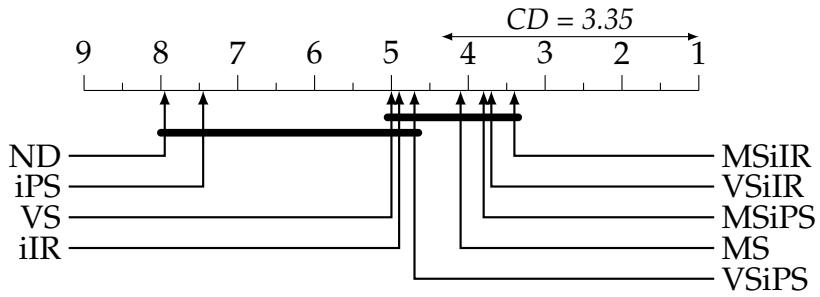


(b) Random-Pair Nested Dichotomy

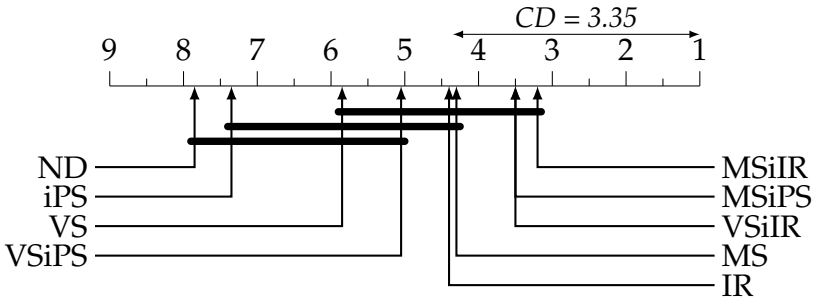
Figure 5.6: Average ranks of different internal calibration methods for nested dichotomies with naïve Bayesian base learners, considering NLL.

Figures 5.6a and 5.7a show the average ranks of each internal calibration strategy for random nested dichotomies with naïve Bayesian base learners. In both cases, performing no calibration (ND) is ranked last, and external calibration methods in conjunction with internal isotonic regression (VSiIR and MSiIR) are ranked highly. It is no surprise that performing internal isotonic regression works well here, as it has been shown before that this method is suitable for calibration of naïve Bayesian models (Niculescu-Mizil and Caruana, 2005a).

Figures 5.6b and 5.7b show the same comparison for nested dichotomies constructed with random-pair selection and naïve Bayesian base learners. Again, the best-performing calibration techniques here are external calibration methods with internal isotonic regression.



(a) Random Nested Dichotomy



(b) Random-Pair Nested Dichotomy

Figure 5.7: Average ranks of different internal calibration methods for nested dichotomies with naïve Bayesian base learners, considering classification accuracy.

Tables 5.5 and 5.6 show the full results of internal calibration strategies of nested dichotomies with naïve Bayesian base learners. By looking deeper than aggregated results such as average ranks, some interesting patterns emerge. One intriguing observation is that for random nested dichotomies, the text datasets (*RCV1*, *sector* and *ODP-5K*) see the best performance when applying external calibration only. Interestingly, approaches with matrix scaling often provide good results for nested dichotomies, in some cases providing substantially better predictive performance. For example, *ALOI* with random nested dichotomies sees the best classification accuracy when external matrix scaling is applied with internal isotonic regression, for which the accuracy jumps to 73% while the best approach not involving matrix scaling reaches only 16.6%. *ILSVR2010* and *devanagari* have the best classification accuracy when external matrix scaling is used in conjunction with internal isotonic regression, which is especially interesting due to the comments made by Guo et al. (2017) regarding matrix scaling with ImageNet, where they found that it does not

help. In general, external vector or matrix scaling with internal isotonic regression (VSiIR and MSiIR) provide reasonably good results in terms of NLL and classification accuracy for all datasets.

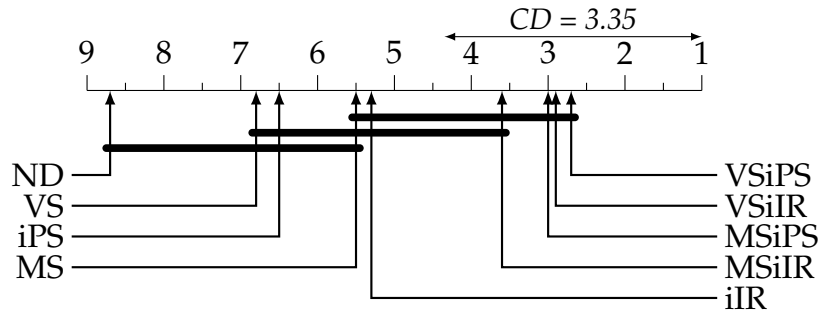
Many of the above trends carry over to nested dichotomies constructed with random-pair selection. As seen in Figures 5.6b and 5.7b, applying internal isotonic regression with vector scaling and matrix scaling gives the best NLL and classification accuracy respectively. As expected, the baseline accuracy of random-pair nested dichotomies is often superior to random nested dichotomies. Similarly, the best performing model after calibration is usually a calibrated nested dichotomy built with random-pair selection.

Micromass is a noteworthy dataset in that it sees reduced classification accuracy when external calibration is applied for both of the nested dichotomy construction methods. This is likely due to having relatively fewer samples per class, leading to overfitting when training the external calibration model on a 10% sample.

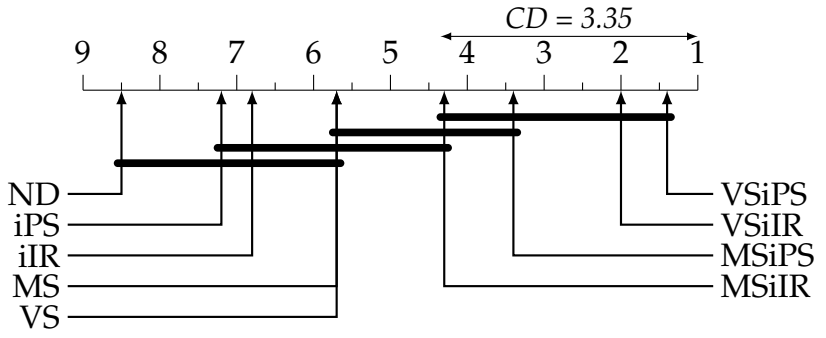
Boosted Decision Trees

Boosted decision trees have also been shown to produce poor probability estimates. As a margin-based classifier, boosted ensembles are usually recommended to be calibrated using Platt scaling (Niculescu-Mizil and Caruana, 2005a). An ensemble of 50 decision trees created with the SAMME.R variant of AdaBoost (Hastie, Rosset, et al., 2009) is used in our experiments, limiting the depth of the trees to three.

Figures 5.8a and 5.9a show the average ranks of each internal calibration strategy for random nested dichotomies with boosted decision tree base learners. Like with naïve Bayes, no calibration (ND) is ranked last for both metrics, but vector scaling with internal Platt scaling (VSiPS) is ranked the best on average—again, this is not surprising due to Platt scaling's affinity with margin-based classifiers and the previous results for



(a) Random Nested Dichotomy



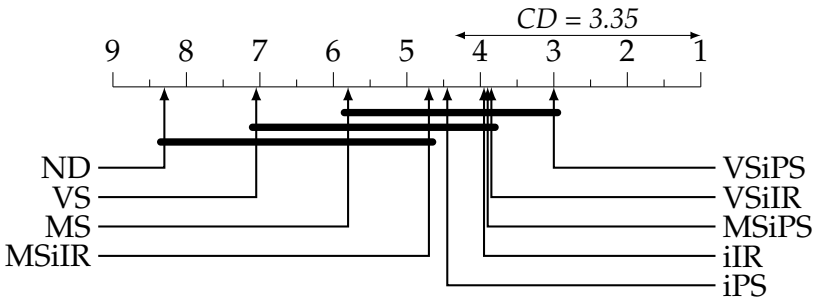
(b) Random-Pair Nested Dichotomy

Figure 5.8: Average ranks of different internal calibration methods for nested dichotomies with boosted decision tree base learners, considering NLL.

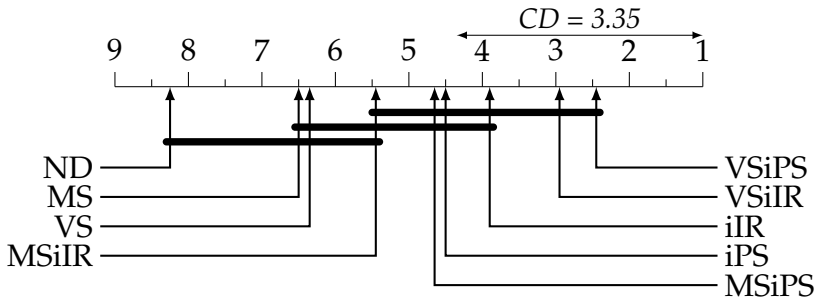
vector scaling. An interesting observation is that matrix scaling with internal Platt scaling (MSiPS) is extremely similarly ranked on average to vector scaling with internal Platt scaling for classification accuracy.

Figures 5.8b and 5.9b show the same comparisons for nested dichotomies built with random-pair selection and with boosted decision tree base learners. The calibration methods applied to random-pair nested dichotomies produce similar rankings to random nested dichotomies when considering NLL and classification accuracy.

Analysing the full results in Tables 5.7 and 5.8 reveals some deeper insight. Considering NLL in Table 5.7, the best method for random nested dichotomies is most commonly vector scaling with internal Platt scaling (VSiPS), but when considering classification accuracy, matrix scaling with internal Platt scaling (MSiPS) is the top-ranked method for just as many datasets as the corresponding vector scaling method. Random-pair



(a) Random Nested Dichotomy



(b) Random-Pair Nested Dichotomy

Figure 5.9: Average ranks of different internal calibration methods for nested dichotomies with boosted decision tree base learners, considering classification accuracy.

nested dichotomies see similar trends when considering NLL, but in classification accuracy, internal Platt scaling without external scaling (iPS) is ranked first for many datasets. Like with naïve Bayes, *micromass* sees a reduction in classification accuracy when external calibration is applied.

Overall, it is particularly noteworthy that matrix scaling approaches give good results in many cases. For example, in random nested dichotomies, the highest classification accuracy for *ALOI* with matrix scaling is 84.0% while the best result with vector scaling is only 74.3%.

5.4 Conclusion

In this chapter, we have demonstrated theoretically and empirically that probability calibration techniques can have a large positive effect on the predictive performance of nested dichotomies. The experiments show that if a base learner with typically poor calibration is employed, then overall performance can often be enhanced substantially by calibrating

Table 5.5: NLL of nested dichotomies with naïve Bayesian base classifiers.

	Dataset	ND	VS	MS	iPS	VSiPS	MSiPS	iIR	VSiIR	MSiIR
Random	optdigits	4.252 ± 0.92	0.841 ± 0.13	1.563 ± 0.28	0.852 ± 0.11	0.807 ± 0.09	0.905 ± 0.10	0.714 ± 0.12	**0.642 ± 0.09**	0.663 ± 0.07
	krkopt	2.654 ± 0.09	2.098 ± 0.05	1.906 ± 0.20	2.713 ± 0.04	1.976 ± 0.03	**1.335 ± 0.12**	2.609 ± 0.03	1.843 ± 0.04	2.042 ± 0.20
	micromass	8.668 ± 1.72	1.624 ± 0.42	1.822 ± 0.03	0.926 ± 0.08	0.752 ± 0.12	1.799 ± 0.02	0.766 ± 0.12	**0.712 ± 0.15**	1.760 ± 0.02
	letter	2.338 ± 0.08	2.155 ± 0.08	**1.061 ± 0.04**	2.165 ± 0.06	2.068 ± 0.07	1.208 ± 0.04	2.055 ± 0.07	1.953 ± 0.06	1.308 ± 0.03
	devanagari	13.14 ± 0.59	3.310 ± 0.16	2.660 ± 0.09	2.986 ± 0.05	2.602 ± 0.02	1.890 ± 0.03	2.754 ± 0.07	2.444 ± 0.05	**1.619 ± 0.04**
	RCV1	1.690 ± 0.19	0.866 ± 0.01	**0.704 ± 0.01**	1.145 ± 0.07	0.947 ± 0.03	0.731 ± 0.03	0.998 ± 0.04	0.914 ± 0.02	0.803 ± 0.03
	sector	3.795 ± 0.36	1.408 ± 0.09	**1.208 ± 0.03**	2.075 ± 0.20	1.518 ± 0.10	1.769 ± 0.08	1.913 ± 0.21	1.774 ± 0.09	3.242 ± 0.15
	ALOI	32.98 ± 0.43	6.841 ± 0.02	2.189 ± 0.02	5.533 ± 0.02	4.333 ± 0.03	1.941 ± 0.05	4.859 ± 0.03	4.137 ± 0.01	**1.256 ± 0.02**
	ILSVR2010	32.39 ± 0.20	6.819 ± 0.00	6.274 ± 0.04	6.162 ± 0.00	6.125 ± 0.01	5.809 ± 0.00	6.176 ± 0.00	6.116 ± 0.00	**5.786 ± 0.00**
	ODP-5K	8.498 ± 0.31	**5.161 ± 0.05**	6.320 ± 0.01	6.103 ± 0.00	5.518 ± 0.02	5.906 ± 0.00	6.051 ± 0.11	5.364 ± 0.01	5.702 ± 0.00
Random Pair	optdigits	1.655 ± 0.29	1.238 ± 0.29	1.103 ± 0.20	1.437 ± 0.14	0.967 ± 0.19	0.830 ± 0.08	1.065 ± 0.06	**0.644 ± 0.08**	0.696 ± 0.08
	krkopt	2.762 ± 0.08	2.096 ± 0.19	2.117 ± 0.44	2.635 ± 0.03	1.027 ± 0.16	1.326 ± 0.13	2.501 ± 0.03	**1.006 ± 0.16**	2.260 ± 0.39
	micromass	**1.550 ± 0.12**	2.141 ± 0.06	1.888 ± 0.06	1.893 ± 0.09	1.929 ± 0.05	1.808 ± 0.04	1.765 ± 0.10	1.745 ± 0.02	1.734 ± 0.02
	letter	2.311 ± 0.04	1.565 ± 0.03	**1.191 ± 0.04**	2.410 ± 0.05	1.562 ± 0.05	1.211 ± 0.04	2.322 ± 0.05	1.435 ± 0.06	1.287 ± 0.04
	devanagari	3.346 ± 0.05	3.063 ± 0.08	2.671 ± 0.06	3.658 ± 0.03	2.451 ± 0.02	1.896 ± 0.02	3.453 ± 0.05	2.225 ± 0.06	**1.654 ± 0.03**
	RCV1	2.631 ± 0.03	1.112 ± 0.04	**0.642 ± 0.02**	2.417 ± 0.09	1.095 ± 0.06	0.838 ± 0.04	2.316 ± 0.06	0.945 ± 0.05	1.099 ± 0.04
	sector	4.427 ± 0.06	1.792 ± 0.11	**1.339 ± 0.03**	4.023 ± 0.17	1.883 ± 0.08	2.037 ± 0.14	3.868 ± 0.19	1.951 ± 0.17	3.680 ± 0.37
	ALOI	6.701 ± 0.01	2.128 ± 0.07	2.303 ± 0.03	6.605 ± 0.02	2.109 ± 0.07	1.659 ± 0.01	6.487 ± 0.05	2.123 ± 0.07	**1.276 ± 0.01**
	ILSVR2010	6.846 ± 0.00	6.780 ± 0.06	6.506 ± 0.01	6.907 ± 0.00	5.987 ± 0.05	9.010 ± 0.03	6.904 ± 0.00	5.870 ± 0.02	**5.728 ± 0.04**
	ODP-5K	15.001 ± 0.12	7.336 ± 0.09	6.775 ± 0.04	6.606 ± 0.08	5.818 ± 0.06	6.149 ± 0.08	6.620 ± 0.04	**5.229 ± 0.02**	5.868 ± 0.02

Table 5.6: Accuracy of nested dichotomies with naïve Bayes.

	Dataset	ND	VS	MS	iPS	VSiPS	MSiPS	iIR	VSiIR	MSiIR
Random	optdigits	0.719 ± 0.05	0.749 ± 0.04	0.479 ± 0.13	0.719 ± 0.05	0.735 ± 0.04	0.694 ± 0.05	0.774 ± 0.04	0.795 ± 0.04	**0.809 ± 0.03**
	krkopt	0.227 ± 0.03	0.260 ± 0.02	**0.610 ± 0.06**	0.191 ± 0.03	0.276 ± 0.02	0.581 ± 0.05	0.254 ± 0.02	0.328 ± 0.02	0.439 ± 0.04
	micromass	0.749 ± 0.05	0.724 ± 0.05	0.339 ± 0.01	0.770 ± 0.05	0.762 ± 0.05	0.340 ± 0.01	**0.772 ± 0.05**	0.756 ± 0.05	0.363 ± 0.01
	letter	0.329 ± 0.02	0.364 ± 0.03	**0.711 ± 0.01**	0.318 ± 0.03	0.365 ± 0.03	0.659 ± 0.02	0.376 ± 0.03	0.412 ± 0.03	0.636 ± 0.01
	devanagari	0.202 ± 0.02	0.224 ± 0.04	0.296 ± 0.02	0.167 ± 0.02	0.269 ± 0.01	0.476 ± 0.01	0.265 ± 0.04	0.340 ± 0.01	**0.559 ± 0.01**
	RCV1	0.644 ± 0.04	0.781 ± 0.00	**0.816 ± 0.00**	0.691 ± 0.03	0.756 ± 0.00	0.794 ± 0.01	0.734 ± 0.01	0.765 ± 0.01	0.778 ± 0.01
	sector	0.337 ± 0.07	0.772 ± 0.01	**0.797 ± 0.01**	0.633 ± 0.04	0.737 ± 0.03	0.699 ± 0.02	0.692 ± 0.04	0.690 ± 0.01	0.532 ± 0.01
	ALOI	0.024 ± 0.00	0.029 ± 0.00	0.639 ± 0.00	0.019 ± 0.00	0.124 ± 0.00	0.670 ± 0.01	0.094 ± 0.00	0.166 ± 0.01	**0.730 ± 0.00**
	ILSVR2010	0.009 ± 0.00	0.015 ± 0.00	0.023 ± 0.02	0.014 ± 0.00	0.019 ± 0.00	0.048 ± 0.02	0.021 ± 0.00	0.026 ± 0.00	**0.049 ± 0.01**
	ODP-5K	0.043 ± 0.01	**0.210 ± 0.00**	0.054 ± 0.00	0.091 ± 0.00	0.161 ± 0.00	0.102 ± 0.00	0.135 ± 0.01	0.180 ± 0.00	0.114 ± 0.00
Random Pair	optdigits	0.551 ± 0.12	0.648 ± 0.10	0.712 ± 0.07	0.622 ± 0.10	0.671 ± 0.09	0.742 ± 0.04	0.816 ± 0.03	0.810 ± 0.03	**0.831 ± 0.03**
	krkopt	0.200 ± 0.03	0.691 ± 0.05	0.620 ± 0.07	0.235 ± 0.02	**0.736 ± 0.05**	0.600 ± 0.05	0.288 ± 0.02	0.734 ± 0.04	0.506 ± 0.07
	micromass	0.789 ± 0.05	0.255 ± 0.02	0.331 ± 0.02	0.782 ± 0.06	0.292 ± 0.03	0.334 ± 0.02	**0.795 ± 0.05**	0.372 ± 0.01	0.380 ± 0.00
	letter	0.526 ± 0.02	0.549 ± 0.01	**0.712 ± 0.01**	0.525 ± 0.02	0.550 ± 0.02	0.685 ± 0.01	0.570 ± 0.02	0.588 ± 0.02	0.676 ± 0.01
	devnagari	0.301 ± 0.02	0.312 ± 0.02	0.337 ± 0.02	0.312 ± 0.02	0.343 ± 0.01	0.497 ± 0.01	0.402 ± 0.02	0.412 ± 0.02	**0.576 ± 0.01**
	RCV1	0.568 ± 0.02	0.728 ± 0.01	**0.831 ± 0.00**	0.716 ± 0.04	0.728 ± 0.01	0.799 ± 0.01	0.739 ± 0.02	0.761 ± 0.02	0.785 ± 0.01
	sector	0.187 ± 0.05	0.700 ± 0.02	**0.786 ± 0.01**	0.518 ± 0.08	0.677 ± 0.01	0.672 ± 0.02	0.558 ± 0.06	0.657 ± 0.02	0.496 ± 0.02
	ALOI	0.187 ± 0.05	0.544 ± 0.02	0.603 ± 0.01	0.518 ± 0.08	0.508 ± 0.02	0.688 ± 0.00	0.558 ± 0.06	0.530 ± 0.01	**0.730 ± 0.00**
	ILSVR2010	0.029 ± 0.00	0.029 ± 0.00	0.027 ± 0.00	0.001 ± 0.00	0.036 ± 0.00	0.045 ± 0.00	0.008 ± 0.00	0.049 ± 0.00	**0.055 ± 0.00**
	ODP-5K	0.018 ± 0.00	0.046 ± 0.00	0.030 ± 0.00	0.043 ± 0.00	0.107 ± 0.00	0.067 ± 0.00	0.123 ± 0.00	**0.196 ± 0.00**	0.100 ± 0.00

Table 5.7: NLL of nested dichotomies with boosted tree base classifiers.

	Dataset	ND	VS	MS	iPS	VSiPS	MSiPS	iIR	VSiIR	MSiIR
Random	optdigits	3.861 ± 0.57	0.634 ± 0.07	0.220 ± 0.04	0.403 ± 0.04	0.298 ± 0.04	0.102 ± 0.02	0.390 ± 0.03	0.303 ± 0.04	$\mathbf{0.145 \pm 0.02}$
	krkopt	2.751 ± 0.07	1.600 ± 0.09	1.404 ± 0.24	2.628 ± 0.07	0.950 ± 0.05	0.943 ± 0.09	2.472 ± 0.07	$\mathbf{0.939 \pm 0.06}$	1.695 ± 0.26
	micromass	10.01 ± 2.05	2.518 ± 0.52	1.339 ± 0.06	1.263 ± 0.11	1.005 ± 0.14	0.907 ± 0.04	1.235 ± 0.27	0.959 ± 0.19	$\mathbf{0.904 \pm 0.03}$
	letter	4.869 ± 0.27	0.924 ± 0.04	0.691 ± 0.04	0.563 ± 0.02	0.446 ± 0.03	$\mathbf{0.352 \pm 0.01}$	0.557 ± 0.02	0.449 ± 0.03	0.426 ± 0.02
	devanagari	3.424 ± 0.28	1.030 ± 0.04	0.984 ± 0.03	2.266 ± 0.17	0.710 ± 0.02	$\mathbf{0.584 \pm 0.01}$	1.977 ± 0.12	0.735 ± 0.02	0.678 ± 0.02
	RCV1	1.964 ± 0.02	1.029 ± 0.00	0.983 ± 0.03	0.932 ± 0.01	0.716 ± 0.01	$\mathbf{0.704 \pm 0.01}$	0.862 ± 0.00	0.745 ± 0.01	0.843 ± 0.02
	sector	3.631 ± 0.20	2.910 ± 0.11	5.327 ± 0.11	2.672 ± 0.03	$\mathbf{2.036 \pm 0.03}$	3.406 ± 0.07	2.597 ± 0.05	2.208 ± 0.07	6.366 ± 0.26
	ALOI	4.443 ± 0.26	2.512 ± 0.05	0.927 ± 0.04	4.889 ± 0.03	1.056 ± 0.02	1.066 ± 0.02	4.284 ± 0.04	1.173 ± 0.03	$\mathbf{0.842 \pm 0.01}$
	ILSVR2010	6.553 ± 0.10	5.863 ± 0.00	6.047 ± 0.02	5.643 ± 0.00	5.219 ± 0.00	5.410 ± 0.00	5.452 ± 0.00	$\mathbf{5.201 \pm 0.00}$	5.330 ± 0.00
	ODP-5K	7.733 ± 0.04	7.192 ± 0.00	7.098 ± 0.00	7.120 ± 0.00	6.603 ± 0.00	7.379 ± 0.00	6.981 ± 0.00	$\mathbf{6.576 \pm 0.00}$	7.182 ± 0.00
Random Pair	optdigits	1.029 ± 0.06	0.199 ± 0.05	0.209 ± 0.04	0.712 ± 0.01	$\mathbf{0.094 \pm 0.01}$	0.099 ± 0.02	0.703 ± 0.01	0.118 ± 0.02	0.157 ± 0.02
	krkopt	2.762 ± 0.05	1.937 ± 0.43	2.318 ± 0.54	2.454 ± 0.07	$\mathbf{0.847 \pm 0.08}$	0.979 ± 0.15	2.217 ± 0.06	0.888 ± 0.09	1.476 ± 0.45
	micromass	1.460 ± 0.09	1.782 ± 0.07	1.392 ± 0.03	1.711 ± 0.06	0.818 ± 0.06	0.795 ± 0.03	1.604 ± 0.07	$\mathbf{0.725 \pm 0.06}$	0.746 ± 0.03
	letter	2.593 ± 0.05	0.567 ± 0.03	0.611 ± 0.04	1.518 ± 0.02	$\mathbf{0.247 \pm 0.02}$	0.336 ± 0.02	1.467 ± 0.01	0.276 ± 0.02	0.469 ± 0.04
	devnagari	3.751 ± 0.03	1.077 ± 0.05	0.879 ± 0.02	2.868 ± 0.12	$\mathbf{0.563 \pm 0.02}$	0.566 ± 0.02	2.787 ± 0.10	0.565 ± 0.03	0.608 ± 0.02
	RCV1	3.591 ± 0.06	0.964 ± 0.05	0.986 ± 0.06	2.270 ± 0.01	$\mathbf{0.627 \pm 0.02}$	0.661 ± 0.01	2.224 ± 0.03	0.647 ± 0.01	0.756 ± 0.01
	sector	4.456 ± 0.03	2.673 ± 0.04	4.729 ± 0.09	4.278 ± 0.04	$\mathbf{1.893 \pm 0.05}$	3.137 ± 0.08	4.055 ± 0.04	2.053 ± 0.07	6.128 ± 0.22
	ALOI	6.818 ± 0.01	2.349 ± 0.00	0.822 ± 0.05	6.609 ± 0.04	$\mathbf{0.668 \pm 0.00}$	1.105 ± 0.01	6.411 ± 0.02	0.753 ± 0.04	0.970 ± 0.01
	ILSVR2010	6.907 ± 0.00	6.380 ± 0.01	6.325 ± 0.01	6.902 ± 0.00	5.291 ± 0.03	5.376 ± 0.02	6.898 ± 0.00	$\mathbf{5.106 \pm 0.01}$	5.462 ± 0.01
	ODP-5K	11.882 ± 0.12	7.630 ± 0.01	7.562 ± 0.04	6.542 ± 0.04	$\mathbf{6.405 \pm 0.02}$	7.376 ± 0.02	7.761 ± 0.01	6.579 ± 0.04	7.375 ± 0.05

Table 5.8: Accuracy of nested dichotomies with boosted trees.

	Dataset	ND	VS	MS	iPS	VSiPS	MSiPS	iIR	VSiIR	MSiIR
Random	optdigits	0.888 ± 0.02	0.883 ± 0.02	0.941 ± 0.01	0.922 ± 0.01	0.917 ± 0.01	**0.970 ± 0.01**	0.921 ± 0.01	0.915 ± 0.01	0.958 ± 0.01
	krkopt	0.173 ± 0.06	0.377 ± 0.03	0.636 ± 0.05	0.246 ± 0.06	0.637 ± 0.02	**0.738 ± 0.04**	0.379 ± 0.05	0.636 ± 0.02	0.517 ± 0.04
	micromass	0.710 ± 0.06	0.650 ± 0.06	0.478 ± 0.02	**0.741 ± 0.06**	0.729 ± 0.06	0.652 ± 0.02	0.737 ± 0.06	0.727 ± 0.05	0.654 ± 0.01
	letter	0.859 ± 0.01	0.851 ± 0.01	0.794 ± 0.01	0.888 ± 0.01	0.883 ± 0.01	**0.896 ± 0.00**	0.890 ± 0.01	0.884 ± 0.01	0.874 ± 0.01
	devanagari	0.103 ± 0.06	0.710 ± 0.01	0.721 ± 0.01	0.488 ± 0.11	0.793 ± 0.01	**0.829 ± 0.00**	0.636 ± 0.04	0.783 ± 0.01	0.803 ± 0.01
	RCV1	0.681 ± 0.06	0.748 ± 0.01	0.730 ± 0.01	0.810 ± 0.00	**0.814 ± 0.01**	0.804 ± 0.00	0.810 ± 0.00	0.807 ± 0.00	0.777 ± 0.00
	sector	0.174 ± 0.08	0.409 ± 0.02	0.269 ± 0.01	**0.603 ± 0.01**	0.576 ± 0.01	0.339 ± 0.01	0.578 ± 0.01	0.552 ± 0.01	0.217 ± 0.01
	ALOI	0.071 ± 0.03	0.451 ± 0.01	0.819 ± 0.01	0.161 ± 0.01	0.743 ± 0.01	0.808 ± 0.01	0.368 ± 0.01	0.723 ± 0.00	**0.840 ± 0.00**
	ILSVR2010	0.019 ± 0.00	0.040 ± 0.00	0.040 ± 0.00	0.093 ± 0.00	**0.102 ± 0.00**	0.078 ± 0.00	0.097 ± 0.00	0.100 ± 0.00	0.080 ± 0.00
	ODP-5K	0.032 ± 0.00	0.038 ± 0.00	0.056 ± 0.00	0.055 ± 0.00	0.068 ± 0.00	0.037 ± 0.00	0.062 ± 0.00	**0.072 ± 0.00**	0.043 ± 0.00
Random Pair	optdigits	0.964 ± 0.01	0.965 ± 0.01	0.957 ± 0.01	**0.980 ± 0.01**	0.974 ± 0.00	0.974 ± 0.00	0.978 ± 0.01	0.971 ± 0.00	0.965 ± 0.00
	krkopt	0.179 ± 0.05	0.729 ± 0.04	0.677 ± 0.06	0.359 ± 0.05	**0.770 ± 0.04**	0.708 ± 0.05	0.505 ± 0.06	0.756 ± 0.04	0.649 ± 0.06
	micromass	0.819 ± 0.04	0.310 ± 0.02	0.494 ± 0.01	**0.824 ± 0.05**	0.703 ± 0.02	0.712 ± 0.01	0.819 ± 0.05	0.728 ± 0.02	0.724 ± 0.01
	letter	0.581 ± 0.09	0.843 ± 0.01	0.837 ± 0.01	**0.937 ± 0.01**	0.927 ± 0.01	0.910 ± 0.00	0.935 ± 0.01	0.921 ± 0.00	0.894 ± 0.01
	devnagari	0.131 ± 0.04	0.706 ± 0.02	0.796 ± 0.01	0.807 ± 0.02	0.841 ± 0.01	**0.859 ± 0.01**	0.821 ± 0.02	0.840 ± 0.01	0.850 ± 0.00
	RCV1	0.160 ± 0.08	0.768 ± 0.01	0.739 ± 0.01	0.826 ± 0.00	**0.833 ± 0.01**	0.821 ± 0.00	0.831 ± 0.00	0.831 ± 0.00	0.797 ± 0.00
	sector	0.194 ± 0.09	0.454 ± 0.02	0.306 ± 0.01	0.511 ± 0.02	**0.608 ± 0.01**	0.376 ± 0.01	0.590 ± 0.01	0.579 ± 0.01	0.253 ± 0.01
	ALOI	0.137 ± 0.04	0.549 ± 0.02	0.834 ± 0.01	0.501 ± 0.04	**0.837 ± 0.01**	0.793 ± 0.00	0.675 ± 0.01	0.830 ± 0.01	0.825 ± 0.00
	ILSVR2010	0.001 ± 0.00	0.024 ± 0.00	0.029 ± 0.00	0.010 ± 0.00	0.094 ± 0.00	0.072 ± 0.00	0.019 ± 0.00	**0.107 ± 0.00**	0.071 ± 0.00
	ODP-5K	0.003 ± 0.00	0.052 ± 0.00	0.038 ± 0.00	0.072 ± 0.00	0.087 ± 0.00	0.025 ± 0.00	0.075 ± 0.00	**0.088 ± 0.00**	0.030 ± 0.00

each internal model. It has also been shown that nested dichotomies systematically produce under-confident probability estimates, even if the base learners are well-calibrated. This can be mitigated by applying calibration models to the multiclass output of the nested dichotomy. Furthermore, applying both internal and external calibration usually yields the best results for nested dichotomies, by applying an internal calibration model well-suited to the base learner employed and using vector or matrix scaling for external calibration. External calibration has been shown to be essential in order to achieve good NLL. In contrast, for classification accuracy, there are some cases where it is best to not calibrate externally.

Chapter 6

Conclusion

Datasets with large label spaces are increasingly common in real world applications, and efficient approaches to classification are needed in order to handle these effectively. This thesis describes a number of methods for improving the predictive performance of nested dichotomies, a method of classification that can enable logarithmic scaling in the number of classes. The random-pair method and multiple subset evaluation were developed in order to make each internal binary model as accurate as possible while maintaining ensemble diversity, which in turn improves the accuracy of entire nested dichotomies and the ensembles that they form. A number of calibration techniques and strategies were also investigated in relation to nested dichotomies, shown to substantially improve their classification accuracy and negative log likelihood (NLL) in many cases.

6.1 Empirical Results

The results of the experiments undertaken as part of this thesis are summarised below.

6.1.1 Random-Pair Method

The random-pair method (Chapter 3, (Leathart, Pfahringer, and Frank, 2016)) has been shown to be a good general-purpose method for constructing nested dichotomies. At the time of its initial publication in 2016, our experiments showed that it generally outperforms the other available methods (Frank and Kramer, 2004; Dong, Frank, and Kramer, 2005; Duarte-Villaseñor et al., 2012) for logistic regression and C4.5 base learners. This thesis has also shown that it is a highly competitive method compared to newly proposed techniques for constructing nested dichotomies (Wever, Mohr, and Hüllermeier, 2018; Melnikov and Hüllermeier, 2018) while taking substantially less time to train.

The use of softmax confusion matrices (Wang, Wang, and Wang, 2018)—a soft extension of the traditional hard confusion matrix that considers the entire estimated class probability distribution—was investigated for grouping subsets of classes, although no performance gains were observed.

The random-pair method works better for logistic regression base learners than for C4.5. It is especially effective compared to other methods when using bagging (Breiman, 1996) and MultiBoost (Webb, 2000) to produce an ensemble.

6.1.2 Multiple Subset Evaluation

Multiple subset evaluation (Chapter 4, (Leathart, Frank, et al., 2019a)) was proposed as a method to improve randomised subset selection methods, such as class-balanced selection (Dong, Frank, and Kramer, 2005) and the random-pair method (Leathart, Pfahringer, and Frank, 2016). The experimental results presented in the chapter show that greater predictive performance is achieved through utilising multiple subset evaluation when logistic regression is used as the base learner.

6.1.3 Calibration of Nested Dichotomies

Predictions from nested dichotomies are obtained by multiplying binary probability estimates together. Our experiments show that calibrating the binary models inside nested dichotomies results in greater overall accuracy. However, even when the internal models are well-calibrated, nested dichotomies are shown to be systematically under-confident in their predictions, especially for problems with many classes. An investigation into these effects showed that often, accuracy and negative log-likelihood can be substantially improved by applying multiclass calibration techniques to the probability estimates produced by nested dichotomies, and binary calibration techniques to the internal models.

When naïve Bayes base learners were used, strategies using isotonic regression to calibrate the internal models tended to perform best; for boosted decision trees, strategies involving Platt scaling of the internal models generally had the best performance. Interestingly, the text-based datasets saw the best performance when multinomial naïve Bayes was chosen as the base learner and external calibration was applied without internal calibration.

6.2 Revisiting the Hypotheses

This section restates the hypotheses from Chapter 1, with discussion in reference to the research presented in Chapters 3–5.

6.2.1 First Hypothesis

The first hypothesis was:

> The performance of nested dichotomies, and ensembles of nested dichotomies, can be improved by appropriate choice of structure(s).

This hypothesis was investigated through the random-pair method and multiple subset evaluation (Chapters 3 and 4), each being methods of selecting high-performing nested dichotomies. Both of these methods are non-deterministic, meaning they can be applied several times in succession in order to create diverse ensembles.

Random-pair selection was shown to improve the classification accuracy significantly in many cases, with no statistically significant degradations. Multiple subset evaluation was shown to improve upon this method (as well as class-balanced selection), often providing significantly improved probability estimates in terms of root mean squared error (RMSE). Individual nested dichotomies and ensembles were tested, thus the hypothesis is confirmed.

6.2.2 Second Hypothesis

The second hypothesis was:

> The performance of nested dichotomies can be improved by maximising the performance of each internal binary model.

This hypothesis was investigated through multiple subset evaluation and internal calibration of nested dichotomies (Chapters 4 and 5). Multiple subset evaluation performs several different splits at each node and picks the one with the best training performance, while internal calibration attempts to improve the quality of the probability estimates at each internal binary model.

As stated earlier, multiple subset evaluation generally leads to better predictive performance in terms of RMSE. Performing internal calibration usually improves the performance if the internal binary models are poorly calibrated; however one must be careful to choose an appropriate calibrator for the internal models. In rare cases, performing internal calibration degrades predictive performance, but usually NLL is decreased and accuracy improved. These results, stemming from two very different

methods of improving the performance of each internal model, support the second hypothesis.

6.3 Future Work

In this section, we identify areas of future research for the work presented in each chapter. In the authors opinion, the most promising areas are combining aspects of the random-pair method and multiple subset evaluation with newer methods for constructing nested dichotomies—Best-of-K models (Melnikov and Hüllermeier, 2018) and the genetic algorithm-based approach (Wever, Mohr, and Hüllermeier, 2018)—and the investigation of matrix scaling as a general calibration method.

6.3.1 Random-Pair Method

Future work in this area could include combining aspects of recent works (Melnikov and Hüllermeier, 2018; Wever, Mohr, and Hüllermeier, 2018) and the random-pair method. Both of these methods rely on sets of uniformly sampling nested dichotomies—either to create the K models from which the best one is selected, or to generate base populations for genetic algorithms to evolve. The quality of the sampled nested dichotomies in these sets could be improved by replacing the uniformly sampled nested dichotomies with those generated by the random-pair method. In this manner, it is possible that the same improved results could be obtained more quickly, or superior predictive performance could be achieved.

Although the investigation into the usage of softmax confusion matrices revealed that they usually produce identical nested dichotomy structures to those where traditional confusion matrices were employed, calibrating the initial model's probabilities before producing a softmax confusion matrix may allow more nuanced decisions about the structure of the nested dichotomy to be made.

6.3.2 Multiple Subset Evaluation

Multiple subset evaluation is a simple idea, which leads to numerous avenues of future research.

Optimising Other Metrics

It is unlikely that training RMSE of the internal models will be a reliable indicator when selecting splits based on more complex models such as decision trees or random forests, so other metrics may be needed. Possible candidates include entropy-based measures, such as information gain, which directly promote class purity amongst the child nodes. As a single incorrect routing decision along the path to the leaf nodes results in an incorrect prediction, node purity is especially important in hierarchical models like nested dichotomies. Such metrics could also be computed in a held-out validation set. In the case that bagging is used to create an ensemble, the out-of-bag samples could be utilised for this purpose. Note that classification accuracy was not considered as a metric because it is less sensitive to the accuracy of the models' probability estimates, but it could potentially be appropriate for some learning algorithms.

Also, it may be beneficial to choose subsets such that maximum ensemble diversity is achieved, possibly through information theoretic measures such as variation of information (Meilă, 2003). Existing metaheuristic approaches to constructing individual nested dichotomies like genetic algorithms (Wever, Mohr, and Hüllermeier, 2018) could also be adapted to optimise ensembles of nested dichotomies in this way.

Random-Pair Optimisation

While we can get a good estimate of the performance of a particular partition of class subsets for some internal node of a nested dichotomy through the train RMSE, we still have to train λ separate models for each node. Depending on the data size, this may be prohibitively expensive. In order to combat this, heuristics for determining the quality of particular random

pairs of classes chosen with the random-pair method could be developed. Hopefully, the suitability of a particular class split can be adequately estimated through properties of the data belonging to the initial random pair of classes, before the base model is re-trained on the entire data present at the node. This way, the λ models are only trained on the two classes selected for the random pair.

The main desirable property of the two-class subsets is easy separability for the base classifier. To this end, some type of *separability* score of the predictions made for each of the remaining classes after the random pair is selected and the initial model is trained could be investigated. A simple way to measure this is given by

$$\text{separability} = s_k = \frac{1}{m}\sum_{c=1}^{m}\frac{1}{n_c}\left|\bar{\mathbf{C}}_k^{(c,c_1)} - \bar{\mathbf{C}}_k^{(c,c_2)}\right| \tag{6.1}$$

where $\bar{\mathbf{C}}_k$ is the non-normalised confusion matrix of the remaining classes for the initial classifier at node k, (c_1, c_2) is the random pair, and n_c is the number of instances of class c. In essence, this separability score measures how well the initial classifier, trained on only the random pair of classes (c_1, c_2), can separate the remaining classes. In the worst-case scenario, every remaining class is exactly split between the two random pairs, in which case the separability score is equal to zero. On the other hand, if each remaining class is perfectly split into c_1 or c_2, a separability score of one is recovered. The assumption underlying this avenue of research is that a more separable initial split leads to a more separable split once the classifier is trained on the full data.

Applying Multiple Subset Evaluation to Newer Methods

Much like the random-pair method, multiple subset evaluation could be combined with aspects of recent works (Melnikov and Hüllermeier, 2018; Wever, Mohr, and Hüllermeier, 2018). Simply replacing the step of sampling random nested dichotomies with other methods, and incorporating multiple subset evaluation, could be beneficial.

Further Investigation into use for Hierarchy Induction

As shown in the case study on *CIFAR-10* in Section 3.3.6, the random-pair method is able to uncover semantic hierarchies present in the data. An issue with applying the random-pair method directly for this task is that if the initial random pair chosen at a node contains two classes from the same semantic group, they are forced to be split up. An example of this can be seen in Figure 3.13, where in the lower levels of the tree, 'cat' and 'dog' are split while 'cat' and 'frog' are placed in the same group. This could potentially be addressed in a heuristic manner with multiple subset evaluation. We conjecture that, with an appropriately chosen base classifier, the above example would produce higher RMSE than the natural grouping of $\{\text{cat}, \text{dog}\}$ and $\{\text{frog}\}$, and thus would be less likely to be selected under multiple subset evaluation.

6.3.3 Calibration of Nested Dichotomies

In this research, we only considered random nested dichotomies and those generated with the random-pairs method. There is no apparent reason that the calibration techniques described would not positively affect the predictive performance of nested dichotomies that are constructed through other means, but experimental verification of this would be beneficial. It would also be interesting to apply these techniques to individual members of ensembles of nested dichotomies.

Matrix Scaling as a General Calibration Method

Matrix scaling has been shown to sometimes be a useful method for calibration of nested dichotomies, even though in the past it has empirically performed much worse than other methods when calibrating neural networks (Guo et al., 2017). Guo et al. go so far as to assert

> "Matrix scaling performs poorly on datasets with hundreds of classes ... and fails to converge on the 1000-class ImageNet

> dataset. This is expected, since the number of parameters scales quadratically with the number of classes. Any calibration model with tens of thousands (or more) parameters will overfit to a small calibration set, even when applying regularisation.";

a statement that our results seemingly contradict. Further investigation into the general applicability of matrix scaling for calibration of other models is warranted. Note that Guo et al. (2017) observed poor results for expected calibration error (ECE), not NLL or classification accuracy, the metrics reported in our experiments. However, if the source of the poor performance is overfitting as they claim, this will be apparent in NLL as well, as it is the metric being directly optimised in the calibration process.

Internal calibration has some obvious areas for future research that may reduce the training time or improve the predictive performance. While outside the scope of this thesis, we describe these identified areas below.

Selective Model Calibration

The number of descendant leaf nodes for an internal node k in a balanced tree equals 2^l, where l is the length of the paths to the leaf nodes from k. Therefore, it is more important to ensure the models nearer the root node are well calibrated than the models nearer leaf nodes. However, calibration of entire layers of a nested dichotomy should have an effect that is independent of the particular layer being calibrated. Note that even though each layer has twice as many internal models as the layer before it, each internal model at the lower layer is trained on approximately half the amount of data as the models in the previous layer. Calibration models typically scale linearly with the number of examples (Guo et al., 2017), so the time taken to train calibration models for each layer of a nested dichotomy should be comparable.

It may be the case that reductions in training time can be achieved by only calibrating the worst models in the tree, without losing a great deal of predictive accuracy. Two options should be investigated:

Layer-wise Calibration. As discussed above, each layer should have roughly equal training cost. It would be interesting to investigate whether layers nearer the leaves, or nearer the root, are comparatively poorly- or well-calibrated in a systematic fashion.

Selective Internal Model Calibration. Rather than performing calibration on every internal model, it would also be interesting to experiment with only calibrating the worst binary models in the tree. Determining the best candidates for calibration can be done through measuring NLL or ECE with a held-out validation set. In the case that external calibration with a held-out calibration set is used, the subset of this set that pertains to the particular internal model in question can be re-used for this purpose.

Adaptive Selection of Internal Calibration Models

Some calibration models are better suited to particular classifiers—for example, naïve Bayes and decision trees have been shown to be calibrated more effectively with isotonic regression than Platt scaling (Zadrozny and Elkan, 2001b; Niculescu-Mizil and Caruana, 2005b). This leads to the idea that isotonic regression should be the calibration model of choice when internally calibrating a nested dichotomy with these base learners. However, due to its more flexible nature, isotonic regression is known to overfit more easily on small calibration samples compared to Platt scaling (Niculescu-Mizil and Caruana, 2005b). Even in the case that a dataset has many training examples, some of the binary models built as part of a nested dichotomy can have very small training sets, particularly nearer the leaf nodes. For this reason, it may be beneficial to adaptively select a calibration model for each problem based on the amount of training data

present. Niculescu-Mizil and Caruana (2005b) suggest as a general rule of thumb that isotonic regression be saved for the case that the calibration set has 1,000 or more examples, and for naïve Bayes specifically, Platt scaling gives superior results where the calibration set contains less than about 250 examples. These rules of thumb could be leveraged to reduce overfitting on smaller binary problems.

Appendix A

Results for the Random-Pair Method

A.1 Dataset Information

The datasets used in these experiments and their characteristics are listed in Table 3.1.

A.2 Full Results Tables

Full results tables for each of the experiments described in Section 3.3 are available below. The best performing algorithm for each dataset is bolded, while the bullets (•) and open circles (○) denote statistically significant wins and losses respectively when comparing random-pair nested dichotomies to the others. Statistical significance is determined with a corrected paired t-test (Nadeau and Bengio, 2000).

Table A.1: Accuracy of a single nested dichotomy with logistic regression base learners.

Dataset	RPND	NDBC	CBND	ND
audiology	**75.73±7.91**	72.56±8.15	68.71±9.37 •	71.48±9.06
krkopt	**33.30±1.02**	33.19±0.76	28.25±1.47 •	28.85±1.63 •
LED24	**72.94±2.36**	72.77±2.16	67.32±4.08 •	70.53±3.37 •
letter	64.91±3.17	**72.20±0.93** ○	48.06±3.23 •	52.96±4.30 •
mfeat-factors	95.12±1.86	**96.62±1.16** ○	91.93±2.31 •	92.92±2.15 •
mfeat-fourier	**76.71±2.99**	75.22±2.79	72.83±3.37 •	74.20±3.39
mfeat-karhunen	89.55±2.41	**90.81±1.94**	85.09±3.60 •	86.31±2.90 •
mfeat-morph	**72.48±2.82**	70.43±2.73 •	61.23±8.25 •	65.77±5.63 •
mfeat-pixel	71.16±9.98	**88.67±2.51** ○	61.25±9.25	47.44±9.15 •
optdigits	**92.34±2.00**	91.99±1.08	87.67±3.10 •	90.72±2.84
page-blocks	**96.30±0.71**	95.76±0.76 •	95.48±0.81 •	95.61±0.91 •
pendigits	**90.04±2.72**	87.98±0.96 •	82.05±4.41 •	87.01±4.15
segment	**93.82±2.46**	88.71±1.79 •	87.23±4.33 •	89.11±3.84 •
shuttle	96.64±0.44	**96.86±0.20**	92.23±6.80	91.77±6.98 •
usps	87.47±1.47	**87.64±1.06**	84.70±2.26 •	85.83±1.97 •
vowel	80.37±5.91	**81.05±3.70**	47.70±8.15 •	53.05±9.11 •
yeast	58.26±3.93	**59.01±3.52**	56.27±3.78	56.14±3.96
zoo	**92.16±8.45**	87.67±9.30	88.31±9.30	88.76±9.27

Table A.2: Accuracy of a single nested dichotomy with C4.5 base learners.

Dataset	RPND	NDBC	CBND	ND
audiology	**76.35±7.39**	74.93±7.28	73.86±8.17	73.89±7.98
krkopt	**69.37±2.07**	69.28±0.98	64.60±1.70 •	65.44±2.48 •
LED24	72.87±2.06	**72.92±1.92**	72.01±2.27	72.21±2.11
letter	86.41±0.87	**86.54±0.85**	85.25±0.90 •	86.05±0.87
mfeat-factors	88.62±2.38	**88.74±2.01**	86.67±2.41	87.40±2.38
mfeat-fourier	**74.43±3.12**	74.02±2.72	72.88±2.93	73.12±2.95
mfeat-karhunen	81.79±2.85	**82.41±2.74**	79.91±3.15	80.62±3.52
mfeat-morph	72.38±2.36	**72.45±2.28**	71.77±2.32	71.92±2.36
mfeat-pixel	**82.29±2.92**	81.50±2.65	77.23±3.77 •	79.40±3.65
optdigits	**90.95±1.31**	90.69±1.19	89.32±1.54 •	90.14±1.53
page-blocks	**97.11±0.65**	97.03±0.67	97.01±0.67	97.04±0.63
pendigits	**95.96±0.62**	95.83±0.61	95.55±0.68	95.74±0.66
segment	96.29±1.32	**96.64±1.22**	95.98±1.39	95.90±1.37
shuttle	99.97±0.02	**99.98±0.02**	99.97±0.02	99.97±0.03
usps	88.30±1.21	**89.49±0.93** ○	86.06±1.48 •	86.70±1.35 •
vowel	**78.64±3.94**	76.45±4.53	75.95±4.43	75.62±4.81
yeast	57.43±3.42	**57.73±3.79**	56.52±3.64	56.87±3.66
zoo	**91.64±8.51**	88.13±8.92	90.69±7.88	90.74±8.17

Table A.3: Accuracy of an ensemble of 10 bagged nested dichotomies with logistic regression base learners.

Dataset	RPND	NDBC	CBND	ND
audiology	81.79±7.56	81.25±7.25	80.32±7.69	**82.35±7.57**
krkopt	**33.77±0.78**	33.29±0.77 •	31.73±0.98 •	31.99±0.94 •
LED24	**73.56±1.90**	73.42±2.01	73.50±1.94	73.49±1.85
letter	**78.65±0.94**	76.16±0.96 •	73.76±1.24 •	74.51±1.27 •
mfeat-factors	**98.11±1.02**	97.39±1.10 •	97.72±1.09	97.94±1.01
mfeat-fourier	**83.08±2.18**	80.03±2.25 •	82.16±2.66	82.14±2.39
mfeat-karhunen	**95.66±1.54**	93.67±1.75 •	94.88±1.56	94.89±1.57
mfeat-morph	**73.71±2.79**	72.33±2.87	73.19±2.94	73.55±2.45
mfeat-pixel	**94.70±1.95**	93.15±1.49 •	90.96±2.51 •	83.65±4.01 •
optdigits	**97.15±0.68**	93.56±0.93 •	96.50±0.83 •	96.83±0.68
page-blocks	**96.46±0.68**	96.14±0.66 •	95.92±0.72 •	96.11±0.68 •
pendigits	**95.93±0.80**	88.90±1.08 •	94.61±1.00 •	95.12±0.88 •
segment	**95.37±1.61**	89.26±1.95 •	94.03±1.96 •	94.15±1.73 •
shuttle	96.74±0.24	**96.86±0.21**	94.94±1.52 •	94.86±1.39 •
usps	**93.83±0.69**	92.02±0.91 •	93.59±0.70	93.32±0.73 •
vowel	**89.76±3.04**	85.72±3.49 •	77.52±4.90 •	78.30±4.61 •
yeast	58.86±3.85	**59.18±3.84**	58.91±3.64	58.92±3.62
zoo	**94.87±6.03**	91.62±8.33	93.36±7.16	93.20±7.37

Table A.4: Accuracy of an ensemble of 10 bagged nested dichotomies with C4.5 base learners.

Dataset	RPND	NDBC	CBND	ND
audiology	79.52±6.98	80.33±6.11	**80.65±7.29**	79.30±7.30
krkopt	**76.31±0.97**	73.93±0.90 •	74.20±1.00 •	74.82±1.00 •
LED24	73.19±1.86	73.12±1.82	73.10±1.90	**73.23±1.92**
letter	93.71±0.54	92.73±0.66 •	93.92±0.50	**94.07±0.49**
mfeat-factors	95.15±1.43	93.37±1.76 •	**95.80±1.40**	95.44±1.52
mfeat-fourier	81.02±2.58	78.79±2.64 •	**81.26±3.02**	80.97±2.53
mfeat-karhunen	92.60±1.84	90.27±2.11 •	92.86±1.69	**93.01±1.58**
mfeat-morph	73.27±2.65	72.78±2.72	72.97±2.84	**73.37±2.55**
mfeat-pixel	**92.77±1.68**	87.01±2.47 •	92.24±1.82	92.65±1.79
optdigits	**97.07±0.75**	95.34±0.90 •	97.04±0.72	97.00±0.72
page-blocks	97.37±0.63	97.29±0.62	**97.39±0.59**	97.36±0.63
pendigits	98.57±0.39	97.67±0.46 •	**98.68±0.35**	98.64±0.38
segment	97.43±1.09	97.52±1.11	**97.54±1.14**	97.53±0.88
shuttle	99.97±0.02	99.97±0.02	**99.98±0.02**	99.98±0.02
usps	**94.63±0.59**	93.85±0.72 •	94.52±0.59	94.61±0.70
vowel	87.99±3.51	85.82±3.73	**89.15±3.46**	88.26±3.25
yeast	59.74±3.53	59.55±3.38	**59.93±3.54**	59.72±3.79
zoo	93.99±6.68	91.70±7.77	93.57±6.81	**94.36±6.17**

Table A.5: Accuracy of an ensemble of 10 nested dichotomies boosted with AdaBoost with logistic regression as the base learner.

Dataset	RPND	NDBC	CBND	ND
audiology	**82.51±8.26**	80.31±6.92	79.87±7.49	80.78±7.50
krkopt	**33.04±1.07**	32.81±0.77	28.24±1.47 •	28.66±1.44 •
LED24	72.32±2.21	**72.93±1.99**	69.17±2.77 •	70.44±2.72 •
letter	70.19±3.01	**71.44±1.49**	47.42±3.29 •	55.16±5.35 •
mfeat-factors	**97.75±1.03**	97.66±0.99	97.11±1.25	97.52±1.17
mfeat-fourier	**80.84±2.48**	79.96±2.52	80.22±2.51	80.18±2.75
mfeat-karhunen	**94.87±1.66**	94.42±1.61	93.60±1.64 •	94.01±1.58
mfeat-morph	**72.51±3.02**	71.02±3.10	66.89±6.86 •	69.43±5.48
mfeat-pixel	**94.15±1.81**	93.87±1.59	91.16±2.39 •	86.21±3.48 •
optdigits	**96.89±0.74**	96.84±0.77	96.27±0.74 •	96.38±0.87
page-blocks	**96.22±0.75**	95.93±0.75	95.43±0.84 •	95.77±0.91
pendigits	**94.99±0.87**	94.83±0.77	93.87±1.29 •	93.67±1.03 •
segment	**94.70±1.56**	94.66±1.48	93.84±1.93	93.91±1.77
shuttle	96.68±0.42	**96.86±0.26**	96.50±1.57	96.40±2.18
usps	**92.03±0.88**	91.83±0.86	91.91±0.91	91.66±0.85
vowel	**89.84±3.30**	89.74±3.10	48.45±9.68 •	58.93±9.42 •
yeast	58.17±3.96	**58.39±3.62**	56.90±4.05	56.56±3.66
zoo	**95.75±5.65**	94.96±6.33	94.38±7.44	94.77±6.19

Table A.6: Accuracy of an ensemble of 10 nested dichotomies boosted with AdaBoost with C4.5 as the base learner.

Dataset	RPND	NDBC	CBND	ND
audiology	**83.64±7.37**	83.29±6.68	82.63±6.87	82.58±7.36
krkopt	**81.01±0.78**	79.37±0.80 •	77.25±0.95 •	78.36±1.04 •
LED24	**69.59±2.13**	69.49±2.11	69.04±1.95	69.42±1.78
letter	94.58±0.49	94.37±0.48	94.30±0.49	**94.60±0.55**
mfeat-factors	**95.75±1.36**	95.31±1.48	95.49±1.38	95.62±1.37
mfeat-fourier	80.43±2.74	79.54±2.60	80.12±2.49	**80.74±2.47**
mfeat-karhunen	**93.20±1.80**	92.67±1.83	92.96±1.76	92.85±1.84
mfeat-morph	70.48±3.10	70.45±3.19	70.13±2.84	**70.50±2.45**
mfeat-pixel	**93.76±1.53**	93.27±1.80	92.48±1.80 •	93.01±1.83
optdigits	**97.31±0.72**	97.23±0.70	97.25±0.68	97.20±0.70
page-blocks	97.05±0.62	97.05±0.66	**97.11±0.64**	97.11±0.66
pendigits	**98.95±0.30**	98.89±0.33	98.91±0.30	98.93±0.28
segment	98.23±0.84	**98.24±0.84**	98.09±0.86	98.09±0.94
shuttle	**99.99±0.01**	99.99±0.01	99.99±0.01	99.99±0.01
usps	94.85±0.64	**94.86±0.64**	94.41±0.72	94.59±0.66
vowel	**91.95±2.71**	90.73±3.00	91.28±2.82	91.30±2.78
yeast	57.39±3.76	**57.42±4.02**	56.93±3.27	57.25±4.19
zoo	95.45±6.19	**95.53±6.39**	95.15±6.21	95.36±6.13

Table A.7: Accuracy of an ensemble of 10 nested dichotomies boosted with MultiBoost with logistic regression as the base learner.

Dataset	RPND	NDBC	CBND	ND
audiology	**81.83±7.25**	80.05±7.20	78.90± 7.51	79.53± 7.73
krkopt	**33.04±1.07**	32.81±0.77	28.24± 1.47 •	28.66± 1.44 •
LED24	**73.36±1.91**	73.31±2.15	72.01± 2.67	72.75± 2.38
letter	**76.04±2.64**	75.36±1.03	47.42± 3.29 •	55.86± 6.25 •
mfeat-factors	**97.84±1.07**	97.70±1.09	97.40± 1.31	97.53± 1.17
mfeat-fourier	**81.79±2.29**	80.22±2.28 •	80.28± 2.42	80.77± 2.43
mfeat-karhunen	**95.15±1.53**	94.70±1.57	93.80± 1.67 •	94.16± 1.68
mfeat-morph	**73.20±2.96**	72.33±2.64	67.61± 7.06 •	70.40± 5.81
mfeat-pixel	**94.37±1.48**	94.16±1.30	91.89± 2.71 •	86.37± 4.74 •
optdigits	**97.08±0.67**	96.10±0.79 •	96.25± 0.78 •	96.47± 0.81 •
page-blocks	**96.47±0.73**	96.09±0.72	96.01± 0.68 •	96.20± 0.74
pendigits	**96.07±0.68**	94.24±1.33 •	94.17± 1.04 •	94.76± 0.93 •
segment	**95.64±1.47**	94.10±1.95 •	94.14± 1.94 •	94.36± 1.63 •
shuttle	96.77±0.29	**96.87±0.24**	96.63± 1.53	96.65± 1.59
usps	**93.12±0.78**	92.45±0.84 •	92.62± 0.83	92.57± 0.84
vowel	**89.31±3.08**	87.52±3.03	48.92±11.26 •	60.91±12.38 •
yeast	**58.83±3.90**	58.60±3.93	57.13± 4.03	57.03± 3.88
zoo	**95.35±6.21**	94.65±6.79	94.46± 7.35	94.07± 7.02

Table A.8: Accuracy of an ensemble of 10 nested dichotomies boosted with MultiBoost with C4.5 as the base learner.

Dataset	RPND	NDBC	CBND	ND
audiology	81.18±7.30	**82.14±7.39**	81.25±7.48	80.32±7.37
krkopt	**76.83±0.96**	75.05±0.84 •	73.54±1.03 •	74.58±1.14 •
LED24	**72.10±1.87**	71.90±1.99	71.78±1.89	71.96±1.99
letter	93.93±0.58	93.65±0.53	93.78±0.55	**93.98±0.46**
mfeat-factors	**95.48±1.40**	94.82±1.45	95.32±1.46	95.14±1.48
mfeat-fourier	80.52±2.59	79.54±2.36	80.32±2.82	**80.64±2.90**
mfeat-karhunen	**92.80±1.83**	91.82±1.91	92.49±1.69	92.52±1.91
mfeat-morph	71.52±2.81	71.26±2.85	71.34±3.05	**71.68±2.80**
mfeat-pixel	**93.10±1.71**	91.15±1.86 •	91.75±1.67 •	92.40±1.90
optdigits	**97.10±0.65**	96.80±0.75	96.91±0.73	97.00±0.69
page-blocks	97.33±0.63	97.24±0.63	**97.34±0.64**	97.29±0.66
pendigits	98.74±0.32	98.69±0.35	**98.78±0.33**	98.75±0.28
segment	97.87±0.94	**98.06±0.94**	97.79±0.95	97.87±0.99
shuttle	**99.99±0.01**	99.99±0.02	99.99±0.02	99.99±0.01
usps	**94.67±0.65**	94.48±0.64	94.25±0.58	94.33±0.71
vowel	**88.98±2.91**	88.33±3.61	88.79±3.18	88.34±3.56
yeast	**58.99±3.57**	58.91±3.56	58.53±3.63	58.35±3.92
zoo	95.35±6.20	94.17±7.34	94.26±6.48	**95.66±6.11**

Appendix B

Results for Multiple Subset Evaluation

B.1 Distribution of Train RMSE

Figures B.1 and B.2 show the empirical distributions of the RMSE of a logistic regression model trained on 1,000 random class splits for each dataset listed in Table 4.1 (except for `mfeat-fourier` and `segment`, which are shown in Fig. 4.2). For each plot, the orange line shows the empirical mean, and the black dotted line shows the value estimated by (4.4). Note that even though many of the datasets are not normally distributed, the estimation provided by (4.4) still closely matches the empirical mean in almost all cases.

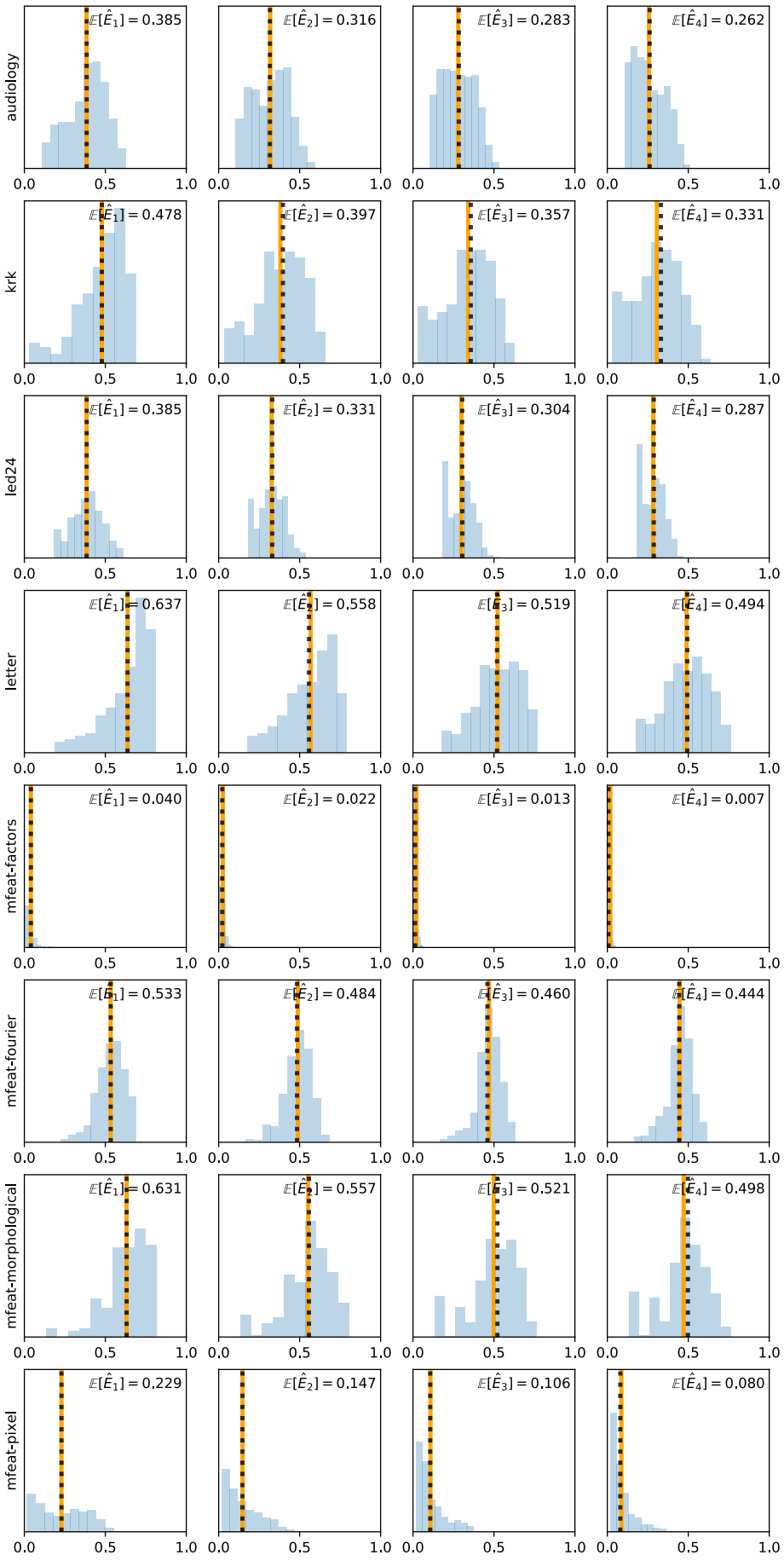


Figure B.1: Distribution of train RMSE of random class splits.

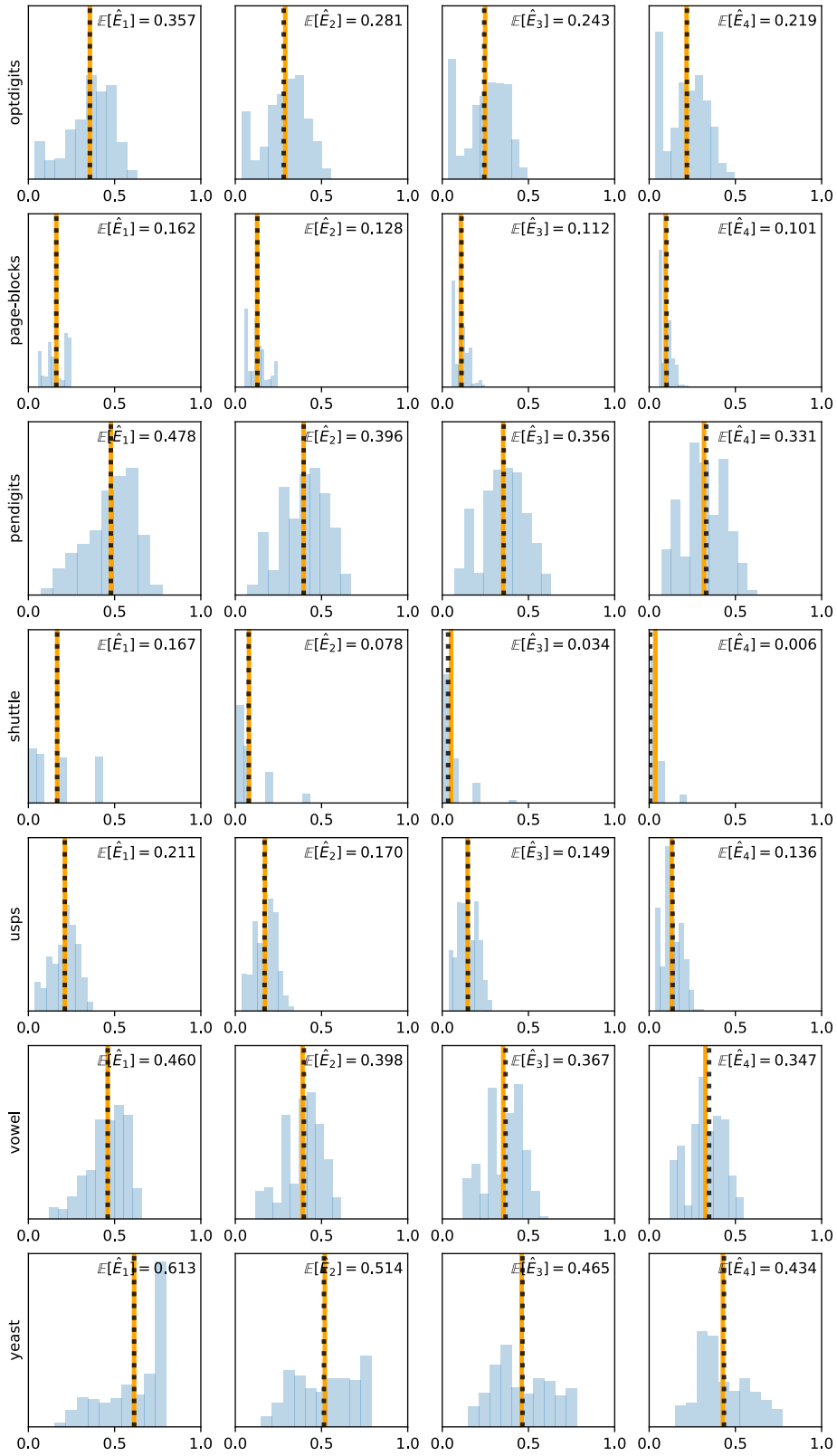


Figure B.2: Distribution of train RMSE of random class splits (continued).

Bibliography

Shailesh Acharya, Ashok Kumar Pant, and Prashnna Kumar Gyawali (2015). "Deep learning based large scale handwritten Devanagari character recognition". In: *Proceedings of the International Conference on Software, Knowledge, Information Management and Applications*. IEEE, pp. 1–6.

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim (2001). "On the surprising behavior of distance metrics in high dimensional space". In: *Proceedings of the International Conference on Database Theory*. Springer, pp. 420–434.

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma (2013). "Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages". In: *Proceedings of the International Conference on the World Wide Web*. ACM, pp. 13–24.

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz (2004). "Applying support vector machines to imbalanced datasets". In: *Proceedings of the European Conference on Machine Learning*. Springer, pp. 39–50.

Erin L Allwein, Robert E Schapire, and Yoram Singer (2000). "Reducing multiclass to binary: A unifying approach for margin classifiers". In: *Journal of Machine Learning Research* 1.Dec, pp. 113–141.

Miriam Ayer, H Daniel Brunk, George M Ewing, William T Reid, and Edward Silverman (1955). "An empirical distribution function for sampling with incomplete information". In: *The Annals of Mathematical Statistics*, pp. 641–647.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio

(2012). "Theano: new features and speed improvements". In: *Proceedings of the Deep Learning and Unsupervised Feature Learning Workshop, Neural Information Processing Systems*.

Eric Bauer and Ron Kohavi (1999). "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants". In: *Machine Learning* 36.1-2, pp. 105–139.

Thomas Bayes, Richard Price, and John Canton (1763). "An essay towards solving a problem in the doctrine of chances". In:

Samy Bengio, Jason Weston, and David Grangier (2010). "Label embedding trees for large multi-class tasks". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 163–171.

Paul N Bennett and Nam Nguyen (2009). "Refined experts: improving classification in large taxonomies". In: *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 11–18.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio (2010). "Theano: a CPU and GPU Math Expression Compiler". In: *Proceedings of the Python for Scientific Computing Conference*.

Alina Beygelzimer, John Langford, Yuri Lifshits, Gregory Sorkin, and Alex Strehl (2009). "Conditional probability tree estimation analysis and algorithms". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 51–58.

Alina Beygelzimer, John Langford, and Pradeep Ravikumar (2007). "Multiclass classification with filter trees". In: *Preprint, June* 2.

— (2009). "Error-correcting tournaments". In: *Proceedings of the International Conference on Algorithmic Learning Theory*. Springer, pp. 247–262.

Ella Bingham and Heikki Mannila (2001). "Random projection in dimensionality reduction: applications to image and text data". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 245–250.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown (2004). "Learning multi-label scene classification". In: *Pattern Recognition* 37.9, pp. 1757–1771.

Leo Breiman (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140.

— (2001). "Random forests". In: *Machine Learning* 45.1, pp. 5–32.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone (1984). "Classification and regression trees". In:

Glenn W Brier (1950). "Verification of forecasts expressed in terms of probability". In: *Monthly Weather Review* 78.1, pp. 1–3.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu (1995). "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5, pp. 1190–1208.

Chih-Chung Chang and Chih-Jen Lin (2011). "LIBSVM: a library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2.3, p. 27.

Tianqi Chen and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. ACM, pp. 785–794.

Anna E Choromanska and John Langford (2015). "Logarithmic time online multiclass prediction". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 55–63.

Anna Choromanska, Krzysztof Choromanski, and Mariusz Bojarski (2016). "On the boosting ability of top-down decision tree learning algorithm for multiclass classification". In: *Computing Research Repository*.

Leda Cosmides and John Tooby (1996). "Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty". In: *Cognition* 58.1, pp. 1–73.

Thomas M Cover and Joy A Thomas (2012). *Elements of information theory*. John Wiley & Sons.

Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi (2015). "Calibrating probability with undersampling for unbalanced classification". In: *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, pp. 159–166.

Hal Daumé III, Nikos Karampatziakis, John Langford, and Paul Mineiro (2017). "Logarithmic Time One-Against-Some". In: *Proceedings of the International Conference on Machine Learning*. Vol. 70. PMLR, pp. 923–932.

Morris H DeGroot and Stephen E Fienberg (1983). "The comparison and evaluation of forecasters". In: *The Statistician*, pp. 12–22.

Ofer Dekel and Ohad Shamir (2010). "Multiclass-multilabel classification with more classes than examples". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 137–144.

Krzysztof Dembczyński, Wojciech Kotłowski, Willem Waegeman, Róbert Busa-Fekete, and Eyke Hüllermeier (2016). "Consistency of probabilistic classifier trees". In: *Proceedings of the Joint European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer, pp. 511–526.

Janez Demšar (2006). "Statistical comparisons of classifiers over multiple data sets". In: *Journal of Machine Learning Research* 7.Jan, pp. 1–30.

Jia Deng, Sanjeev Satheesh, Alexander C Berg, and Fei-Fei Li (2011). "Fast and balanced: Efficient label tree learning for large scale object recognition". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 567–575.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, diogo149, Brian McFee, Hendrik Weideman, takacsg84, peterderivaz, Jon, instagibbs, Dr. Kashif Rasul, CongLiu, Britefury, and Jonas Degrave (2015). *Lasagne: First release.* DOI: 10.5281/zenodo.27878.

Thomas G. Dietterich and Ghulum Bakiri (1995). "Solving multiclass learning problems via error-correcting output codes". In: *Journal of Artificial Intelligence Research*, pp. 263–286.

Pedro Domingos (1999). "MetaCost: A general method for making classifiers cost-sensitive". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 155–164.

Lin Dong, Eibe Frank, and Stefan Kramer (2005). "Ensembles of balanced nested dichotomies for multi-class problems". In: *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*. Springer, pp. 84–95.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin (2018). "CatBoost: gradient boosting with categorical features support". In: *arXiv preprint arXiv:1810.11363*.

Miriam Mónica Duarte-Villaseñor, Jesús Ariel Carrasco-Ochoa, José Francisco Martínez-Trinidad, and Marisol Flores-Garrido (2012). "Nested Dichotomies Based on Clustering". In: *Proceedings of the Iberoamerican Congress on Pattern Recognition*. Springer, pp. 162–169.

Bradley Efron and Robert Tibshirani (1997). "Improvements on cross-validation: the 632+ bootstrap method". In: *Journal of the American Statistical Association* 92.438, pp. 548–560.

Charles Elkan (2001). "The foundations of cost-sensitive learning". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. Vol. 17. 1. Lawrence Erlbaum Associates Ltd, pp. 973–978.

John Fox (1997). *Applied Regression Analysis, Linear Models, and Related Methods*. Sage.

Eibe Frank and Stefan Kramer (2004). "Ensembles of nested dichotomies for multi-class problems". In: *Proceedings of the International Conference on Machine Learning*. ACM, p. 39.

Yoav Freund and Robert E Schapire (1996a). "Experiments with a new boosting algorithm". In: *Proceedings of the International Conference on Machine Learning*. Vol. 96, pp. 148–156.

Yoav Freund and Robert E Schapire (1996b). "Game theory, on-line prediction and boosting". In: *Proceedings of the Conference on Computational Learning Theory*, pp. 325–332.

— (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of Computer and System Sciences* 55.1, pp. 119–139.

Jerome H Friedman (1996). "Another approach to polychotomous classification". In: *Technical Report, Statistics Department, Stanford University*.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The Annals of Statistics* 28.2, pp. 337–407.

Johannes Fürnkranz (2002). "Round robin classification". In: *Journal of Machine Learning Research* 2.Mar, pp. 721–747.

Rayid Ghani (2000). "Using error-correcting codes for text classification". In: *Proceedings of the International Conference on Machine Learning*, pp. 303–310.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio (2016). *Deep learning*. Vol. 1. MIT Press Cambridge.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger (2017). "On Calibration of Modern Neural Networks". In: *Proceedings of the International Conference on Machine Learning*. PMLR, pp. 1321–1330.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten (2009). "The WEKA data mining software: an update". In: *ACM SIGKDD Explorations Newsletter* 11.1, pp. 10–18.

H Leon Harter (1961). "Expected values of normal order statistics". In: *Biometrika* 48.1/2, pp. 151–165.

Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou (2009). "Multi-class adaboost". In: *Statistics and its Interface* 2.3, pp. 349–360.

Trevor Hastie, Robert Tibshirani, et al. (1998). "Classification by pairwise coupling". In: *The Annals of Statistics* 26.2, pp. 451–471.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

David Heckerman, Dan Geiger, and David M Chickering (1995). "Learning Bayesian networks: The combination of knowledge and statistical data". In: *Machine Learning* 20.3, pp. 197–243.

Jay Heo, Hae Beom Lee, Saehoon Kim, Juho Lee, Kwang Joon Kim, Eunho Yang, and Sung Ju Hwang (2018). "Uncertainty-Aware Attention for Reliable Interpretation and Prediction". In: *Proceedings of Advances in Neural Information Processing Systems*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean (2015). "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531*.

Tin Kam Ho and Mitra Basu (2002). "Complexity measures of supervised classification problems". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 3, pp. 289–300.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten (2017). "Densely connected convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2, p. 3.

Sergey Ioffe and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the International Conference on Machine Learning*. PMLR, pp. 448–456.

Gareth James and Trevor Hastie (1998). "The error coding method and PICTs". In: *Journal of Computational and Graphical Statistics* 7.3, pp. 377–387.

Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado (2011). "Smooth isotonic regression: A new method to calibrate predictive models". In: *AMIA Summits on Translational Science Proceedings* 2011, p. 16.

Rohit J Kate and Ramya Nadig (2017). "Stage-specific predictive models for breast cancer survivability". In: *International journal of medical informatics* 97, pp. 304–311.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu (2017). "LightGBM: A highly efficient gradient boosting decision tree". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 3146–3154.

Michael Kearns and Yishay Mansour (1999). "On the boosting ability of top–down decision tree learning algorithms". In: *Journal of Computer and System Sciences* 58.1, pp. 109–128.

Jack Kiefer, Jacob Wolfowitz, et al. (1952). "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* 23.3, pp. 462–466.

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus (1992). "Handwritten digit recognition by neural networks with single-layer training". In: *IEEE Transactions on Neural Networks* 3.6, pp. 962–968.

Ron Kohavi et al. (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. Vol. 14. 2, pp. 1137–1145.

Alex Krizhevsky and Geoffrey Hinton (2009). *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer.

Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan (2013). "Beam search algorithms for multilabel learning". In: *Machine Learning* 92.1, pp. 65–89.

Ludmila I Kuncheva and Christopher J Whitaker (2003). "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy". In: *Machine Learning* 51.2, pp. 181–207.

Twan van Laarhoven (2017). "L_2 regularization versus batch and weight normalization". In: *arXiv preprint arXiv:1706.05350*.

Niels Landwehr, Mark Hall, and Eibe Frank (2005). "Logistic model trees". In: *Machine Learning* 59.1-2, pp. 161–205.

Tim Leathart, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes (2017). "Probability Calibration Trees". In: *Proceedings of the Asian Conference on Machine Learning*. PMLR, pp. 145–160.

— (2019a). "Ensembles of Nested Dichotomies with Multiple Subset Selection". In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer.

— (2019b). "On Calibration of Nested Dichotomies". In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer.

Tim Leathart, Bernhard Pfahringer, and Eibe Frank (2016). "Building ensembles of adaptive nested dichotomies with random-pair selection". In: *Proceedings of the Joint European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer, pp. 179–194.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *IEEE* 86.11, pp. 2278–2324.

Jin-Seon Lee and Il-Seok Oh (2003). "Binary classification trees for multiclass classification problems". In: *Proceedings of the International Conference on Document Analysis and Recognition*. IEEE, p. 770.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin (2018). "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *Proceedings of the International Conference on Learning Representations*.

David D Lewis and William A Gale (1994). "A sequential algorithm for training text classifiers". In: *Proceedings of the International SIGIR Conference on Research and Development in Information Retrieval*. Springer, pp. 3–12.

Moshe Lichman (2013). *UCI machine learning repository*.

Yosvany López, Abdollah Dehzangi, Sunil Pranit Lal, Ghazaleh Taherzadeh, Jacob Michaelson, Abdul Sattar, Tatsuhiko Tsunoda, and Alok Sharma (2017). "SucStruct: prediction of succinylated lysine

residues by using structural properties of amino acids". In: *Analytical biochemistry* 527, pp. 24–32.

Ana C Lorena and André CPLF de Carvalho (2008). "Hierarchical decomposition of multiclass problems". In: *Neural Network World* 18.5, p. 407.

— (2010). "Building binary-tree-based multiclass classifiers using separability measures". In: *Neurocomputing* 73.16-18, pp. 2837–2845.

Ilya Loshchilov and Frank Hutter (2019). "Decoupled Weight Decay Regularization". In: *Proceedings of the International Conference on Learning Representations*.

Hans Peter Luhn (1957). "A statistical approach to mechanized encoding and searching of literary information". In: *IBM Journal of Research and Development* 1.4, pp. 309–317.

Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng (2019). "Towards Understanding Regularization in Batch Normalization". In: *Proceedings of the International Conference on Learning Representations*.

James MacQueen et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 14, pp. 281–297.

Henry B Mann and Donald R Whitney (1947). "On a test of whether one of two random variables is stochastically larger than the other". In: *The Annals of Mathematical Statistics*, pp. 50–60.

Eddy Mayoraz and Miguel Moreira (1997). "On the decomposition of polychotomies into dichotomies". In: *Proceedings of the International Conference on Machine Learning*. EPFL-CONF-82398. Morgan Kaufmann.

Marina Meilă (2003). "Comparing clusterings by the variation of information". In: *Proceedings of the Workshop on Learning Theory and Kernel Machines, Computational Learning Theory*. Springer, pp. 173–187.

Vitalik Melnikov and Eyke Hüllermeier (2018). "On the effectiveness of heuristics for learning nested dichotomies: an empirical analysis". In: *Machine Learning*, pp. 1–24.

Deiner Mena, Elena Montañés, José Ramón Quevedo, and Juan José Del Coz (2015). "Using A^* for inference in probabilistic classifier chains". In: *Proceedings of the International Joint Conference on Artificial Intelligence*.

Mary C Meyer (2008). "Inference using shape-restricted regression splines". In: *The Annals of Applied Statistics* 2.3, pp. 1013–1033.

George A Miller (1995). "WordNet: A Lexical Database for English". In: 38.11, pp. 39–41.

Marvin Minsky and Seymour A Papert (1969). *Perceptrons: An introduction to computational geometry*. MIT press.

Melanie Mitchell (1998). *An introduction to genetic algorithms*. MIT press.

Andriy Mnih and Geoffrey E Hinton (2009). "A scalable hierarchical distributed language model". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 1081–1088.

Felix Mohr, Marcel Wever, and Eyke Hüllermeier (2018a). "ML-Plan: Automated machine learning via hierarchical planning". In: *Machine Learning* 107.8-10, pp. 1495–1515.

— (2018b). "Reduction Stumps for Multi-class Classification". In: *Proceedings of the International Symposium on Intelligent Data Analysis*. Springer, pp. 225–237.

Frederic Morin and Yoshua Bengio (2005). "Hierarchical Probabilistic Neural Network Language Model". In: *Proceedings of the International Workshop on Artificial Intelligence and Statistics*. Vol. 5, pp. 246–252.

Allan H Murphy and Robert L Winkler (1977). "Reliability of subjective probability forecasts of precipitation and temperature". In: *Applied Statistics*, pp. 41–47.

Claude Nadeau and Yoshua Bengio (2000). "Inference for the generalization error". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 307–313.

Mahdi Naeini, Gregory Cooper, and Milos Hauskrecht (2015). "Obtaining Well Calibrated Probabilities Using Bayesian Binning." In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2901–2907.

John Ashworth Nelder and Robert WM Wedderburn (1972). "Generalized linear models". In: *Journal of the Royal Statistical Society: Series A (General)* 135.3, pp. 370–384.

Andrew Y Ng and Michael I Jordan (2002). "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 841–848.

Andrew Y Ng, Michael I Jordan, and Yair Weiss (2002). "On spectral clustering: analysis and an algorithm". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 849–856.

Alexandru Niculescu-Mizil and Rich Caruana (2005a). "Obtaining Calibrated Probabilities from Boosting." In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, p. 413.

— (2005b). "Predicting good probabilities with supervised learning". In: *Proceedings of the International Conference on Machine Learning*. ACM, pp. 625–632.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. (2011). "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12.Oct, pp. 2825–2830.

Bernhard Pfahringer (2000). "Winning the KDD99 classification cup: bagged boosting". In: *ACM SIGKDD Explorations Newsletter* 1.2, pp. 65–66.

Edgar Pimenta and Joao Gama (2005). "A study on error correcting output codes". In: *Proceedings of the Portuguese Conference on Artificial Intelligence*. IEEE, pp. 218–223.

John Platt (1999). "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". In: *Advances in Large Margin Classifiers* 10.3, pp. 61–74.

David Price, Stefan Knerr, Léon Personnaz, and Gérard Dreyfus (1995). "Pairwise neural network classifiers with probabilistic outputs". In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 1109–1116.

John R Quinlan (1993). *C4.5: Programs for machine learning*. Elsevier.

Philippe Refregier and François Vallet (1991). "Probabilistic approach for multiclass classification with neural networks". In: *Artificial Neural Networks*. Elsevier, pp. 1003–1006.

Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger (2003). "Tackling the poor assumptions of naïve bayes text classifiers". In: *Proceedings of the International Conference on Machine Learning*. AAAI, pp. 616–623.

Ryan Rifkin and Aldebaro Klautau (2004). "In defense of one-vs-all classification". In: *Journal of Machine Learning Research* 5, pp. 101–141.

Juan J Rodríguez, César García-Osorio, and Jesús Maudes (2010). "Forests of nested dichotomies". In: *Pattern Recognition Letters* 31.2, pp. 125–132.

Peter J Rousseeuw (1984). "Least median of squares regression". In: *Journal of the American Statistical Association* 79.388, pp. 871–880.

JP Royston (1982). "Algorithm AS 177: Expected normal order statistics (exact and approximate)". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31.2, pp. 161–165.

Stefan Rüping (2006). "Robust probabilistic calibration". In: *Proceedings of the European Conference on Machine Learning*. Springer, pp. 743–750.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Robert E Schapire and Yoram Singer (1999). "Improved boosting algorithms using confidence-rated predictions". In: *Machine Learning* 37.3, pp. 297–336.

Friedhelm Schwenker (2000). "Hierarchical support vector machines for multi-class pattern recognition". In: *Proceedings of the International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*. Vol. 2. IEEE, pp. 561–565.

Friedhelm Schwenker and Günther Palm (2001). "Tree-structured support vector machines for multi-class pattern recognition". In: *Proceedings of the International Workshop on Multiple Classifier Systems*. Springer, pp. 409–417.

Karen Simonyan and Andrew Zisserman (2015). "Very deep convolutional networks for large-scale image recognition". In:

Marcilio CP de Souto, Ana C Lorena, Newton Spolaôr, and Ivan G Costa (2010). "Complexity measures of supervised classifications tasks: a case study for cancer gene expression data". In: *Proceedings of the International Joint Conference on Neural Networks*. IEEE, pp. 1–7.

Karen Spärck Jones (1972). "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of Documentation* 28.1, pp. 11–21.

Marc Sumner, Eibe Frank, and Mark Hall (2005). "Speeding up logistic model tree induction". In: *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*. Springer, pp. 675–683.

Fumitake Takahashi and Shigeo Abe (2002). "Decision-tree-based multiclass support vector machines". In: *Proceedings of the International Conference on Neural Information Processing*. Vol. 3. IEEE, pp. 1418–1422.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown (2013). "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 847–855.

Vladimir Vapnik (1998). *Statistical learning theory*. Vol. 3. Wiley, New York.

Volkan Vural and Jennifer G Dy (2004). "A hierarchical method for multi-class support vector machines". In: *Proceedings of the International Conference on Machine Learning*. ACM, p. 105.

Xiao Wang and Feng Li (2008). "Isotonic smoothing spline regression". In: *Journal of Computational and Graphical Statistics* 17.1, pp. 21–37.

Zhenhua Wang, Xingxing Wang, and Gang Wang (2018). "Learning fine-grained features via a CNN tree for large-scale classification". In: *Neurocomputing* 275, pp. 1231–1240.

Geoffrey I Webb (2000). "Multiboosting: A technique for combining boosting and wagging". In: *Machine Learning* 40.2, pp. 159–196.

Marcel Wever, Felix Mohr, and Eyke Hüllermeier (2018). "Ensembles of evolved nested dichotomies for classification". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 561–568.

David H Wolpert (1992). "Stacked generalization". In: *Neural Networks* 5.2, pp. 241–259.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng (2004). "Probability estimates for multi-class classification by pairwise coupling". In: *Journal of Machine Learning Research* 5.Aug, pp. 975–1005.

Limin Yang, Suming Jin, Patrick Danielson, Collin Homer, Leila Gass, Stacie M Bender, Adam Case, Catherine Costello, Jon Dewitz, Joyce Fry, et al. (2018). "A new generation of the United States National Land Cover Database: Requirements, research priorities, design, and implementation strategies". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 146, pp. 108–123.

Bianca Zadrozny and Charles Elkan (2001a). "Learning and making decisions when costs and probabilities are both unknown". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 204–213.

Bianca Zadrozny and Charles Elkan (2001b). "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In:

Proceedings of the International Conference on Machine Learning. Vol. 1. Citeseer, pp. 609–616.

— (2002). "Transforming classifier scores into accurate multiclass probability estimates". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 694–699.

Wenliang Zhong and James T Kwok (2013). "Accurate Probability Calibration for Multiple Classifiers." In: *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1939–1945.

They don't think it be like it is, but it do.

Oscar Gamble, 1975