Statistical Learning Approaches to Some Classification Problems

by

Hyukjun Gweon

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Statistics

Waterloo, Ontario, Canada, 2017

Examining Committee Membership

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

External Examiner:	Jilles Vreeken Professor (Saarland University)
Supervisor(s):	Matthias Schonlau Professor Stefan Steiner Professor
Internal Member:	Mu Zhu Professor
Internal Member:	Ryan Browne Assistant Professor
Internal-External Member:	Mark Crowley Assistant Professor (Electrical and Computer Engineering)

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

Classification is essential in statistical learning. This thesis deals with three topics in classification: multi-label classification, nonparametric multi-class classification and a special type of text categorization called occupation coding. For each topic, novel approaches are proposed with the goal of high predictive performance. This is empirically demonstrated for each method.

In multi-label classification, observations may be associated with multiple classes or labels simultaneously. Generally, correlations exist between labels and taking into account the label correlations is important during the classification process. This thesis proposes an approach based on the nearest neighbor principle that considers neighbors both in the feature ($\mathbf{x}$) and the label ($\mathbf{y}$) space. The proposed method chooses the labelset of a training observation that minimizes a weighted function of the distances in feature and label space. By selecting an entire labelset as the prediction, the method implicitly considers label correlations.

In multi-class classification, the well-known k-nearest neighbors method is especially desirable when the response surface exhibits highly local behavior. A novel approach is presented that makes a prediction based on the k^{th} nearest neighbor from each class. The method not only provides estimates for class posterior probabilities but also converges to the Bayes classifier as the size of the training data increases. Further, the method is extended using the idea of an ensemble.

Occupation coding is an important multi-class text categorization problem. Since fully automated classification is challenging, researchers focus more on partially automated coding. Three approaches based on underlying statistical learning methods are proposed to improve the classification accuracy of the underlying statistical learning methods.

Acknowledgements

I would like to thank my supervisors Dr. Matthias Schonlau and Dr. Stefan Steiner. This thesis would not have been completed without their support and guidance. Matthias has always been encouraging my research. His insightful advice and ideas about the research and scientific writing have been invaluable. Stefan has also been very supportive and provided insightful discussions and suggestions about the research. I am lucky to have them as my mentors.

I also thank my thesis committees, Dr. Mu Zhu, Dr. Ryan Browne, Dr. Mark Crowley, and Dr. Jilles Vreeken for serving my committee members.

I am deeply grateful to my parents, brothers and Shu Li, for their remarkable and unconditional support. I am extremely lucky to have them in my life.

My PhD research has received financial support from many sources, including the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ontario Graduate Scholarship (OGS), and teaching and research assistantships from the Department of Statistics and Actuarial Science. I am grateful for having this funding during my PhD program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the advent of the information age, data are everywhere and statistical problems have exploded both in size and complexity. Learning from data is essential for statistical researchers. As the amount of data grows, classifying data manually has become infeasible. At the same time, automated classification using statistical learning algorithms has been an essential part of modern statistics.

In traditional classification problems, each observation is associated with a single class label. Such a problem is called either binary classification when there are only two classes or multi-class (or multinomial) classification when there are more than two classes. (For the rest of this thesis, we refer single-label classification as multi-class classification for convenience.) On the other hand, in multi-label classification each observation may belong to multiple labels simultaneously. Compared with multi-class classification, learning from multi-label data has recently received increasing attention from machine learning researchers (Madjarov et al., 2012; Tsoumakas et al., 2010).

There are many approaches to classification. The k nearest neighbor (kNN) method

(Fix and Hodges, 1951) is one of the most popular approaches (Wu et al., 2008). The principle of kNN is simple; for the prediction of a new observation, kNN identifies nearest training observations based on the feature information and makes a prediction using the labels/classes of the nearest observations. Despite its simplicity, kNN performs well in many classification problems. This thesis proposes various approaches to different classification problems that extend the nearest neighbor principle.

This thesis contributes novel approaches to both multi-class and multi-label classifications in statistical learning. Evaluated on a suite of data sets, the proposed approaches compare favorably to state-of-the-art methods on one or multiple criteria. Specifically, this thesis covers three different topics in supervised classification: (a) multi-label classification, (b) nonparametric multi-class classification, and (c) an important application of multi-class classification called occupation coding. Although many of the proposed approaches are based on the nearest neighbor principle, the approaches are all different. For multi-label classification, we propose a method that finds the nearest neighbor using both the feature and label spaces. For nonparametric multi-class problems, we propose a method that predicts using nearest neighbors conditional on each class. For occupation coding, we propose a modified nearest neighbor approach, as well as two other statistical learning methods, to improve the quality of automated coding.

Significant parts of the work presented in this thesis are published or have been submitted for publication:

- H. Gweon, M. Schonlau, S. H. Steiner. "Nearest Labelset Using Double Distances for Multi-label Classification". Submitted to ECML-PKDD.
- H. Gweon, M. Schonlau, S. H. Steiner. "The k Conditional Nearest Neighbor Algorithm for Classification". Submitted to ICML.

- H. Gweon, M. Schonlau, L. Kaczmirek, M. Blohm, S. Steiner. "Three Methods for Occupation Coding Based on Statistical Learning". *Journal of Official Statistics*. Volume 33, Issue 1, Pages 101–122, 2017.

Chapters 2, 3 and 4 are self-contained and each corresponds to one of the papers listed above. This thesis is organized as follows.

In Chapter 2, we study multi-label classification. Predicting each label independently has been criticized for not exploiting any correlation between labels. We propose a novel approach, called Nearest Labelset using Double Distances ($NLDD$). The proposed method is based on the distances between new and training observations both in the feature and in the label spaces. The predicted labelset is the labelset of a training observation that minimizes a weighted sum of the two distances. The weights are estimated from binomial regression of the number of misclassified labels on the two distance variables. The weights are estimated by maximum likelihood. $NLDD$ only considers labelsets observed in the training data, thus implicitly taking into account label dependencies. Experiments on benchmark multi-label data sets show that $NLDD$ on average outperforms other well-known approaches.

In Chapter 3, we study nearest neighbor-based nonparametric approaches to multi-class classification. We introduce a novel nonparametric method, called k conditional nearest neighbor ($kCNN$), based on nearest neighbors conditional on each class: $kCNN$ calculates the distance between a new observation and the k^{th} nearest neighbor from each class, estimates posterior probabilities of class memberships using the distances, and assigns the observation to the class with the largest posterior. We prove that the $kCNN$ approach converges to the Bayes classifier as the size of the training data increases. Further, we extend the proposed approach to an ensemble method. Experiments on benchmark data

sets show that both $kCNN$ and the ensemble version of $kCNN$ on average outperform other methods including the traditional k nearest neighbor method.

In Chapter 4, we study occupation coding, an important application of multi-class classification. Occupation coding is a type of text categorization where the features are text answers from survey respondents and the response variable is occupation codes, where an occupation code refers to a hierarchically structured numeric code that identifies a unique job title as well as its parent groups. Usually occupation coding is partially automated; some answers need to be coded manually. The goal is to determine the fraction of the observations that can be coded automatically with high coding quality. We introduce three methods for automatic coding: combining separate models for the detailed occupation codes and for aggregate occupation codes, a hybrid method that combines a duplicate-based approach with a statistical learning method, and a modified nearest neighbor approach. Using data from the German General Social Survey (ALLBUS), we show that the methods improve the automated coding accuracy of the underlying statistical learning methods. Also, we show that the proposed methods allow us to code a larger fraction of the observations automatically for any given target accuracy. Further, we find defining duplicates based on ngram variables is preferable to one based on exact string matches.

In Chapter 5, we summarize the thesis and discuss future work.

Chapter 2

NLDD: a new algorithm for Multi-label classification

2.1 Introduction

In multi-label classification, an observation can belong to multiple labels at the same time. This is different from multi-class or binary classification, where an observation can only be associated with a single label. For example, a newspaper article talking about electronic books may be labelled with multiple topics such as business, arts and technology simultaneously. Multi-label classification has been applied in many areas of application including text (Schapire and Singer, 2000; Godbole and Sarawagi, 2004), image (Boutell et al., 2004; Zhang and Zhou, 2007), music (Li and Ogihara, 2003; Trohidis et al., 2008) and functional genomics (Elisseeff and Weston, 2001; Struyf et al., 2005; Blockeel et al., 2006). A labelset for an observation is the set of all labels that are associated with that observation. A multi-label classification problem is a special case of multi-target classification (also

known as multi-dimensional or multi-objective classification), where each label can take multiple values rather than binary (Bielza et al., 2011).

Approaches for solving multi-label classification problems may be categorized into either problem transformation methods or algorithm adaptation methods (Tsoumakas and Katakis, 2007). Problem transformation methods transform a multi-label problem into one or more single-label problems. For the single-label classification problems, binary or multi-class classifiers are used. The results are combined and transformed back into a multi-label representation. Algorithm adaptation methods, on the other hand, modify specific learning algorithms directly for multi-label problems. Tsoumakas et al. (2010), Madjarov et al. (2012) and Zhang and Zhou (2014) give overviews over multi-label algorithms and evaluation metrics. Also, Madjarov et al. (2012) conduct an extensive experimental comparison of various multi-label learning methods.

In this chapter, we propose a new problem transformation approach that applies the nearest neighbor method based on the shortest distance in the feature space. However, because we have multiple labels, we additionally consider the shortest (Euclidean) distance in the label space where the input of the test observation in the label space consists of probability outputs obtained by independent binary classifiers. We then find the labelset that minimizes the expected label misclassification rate as a function of both distances, feature space and label space, exploiting high-order interdependencies between labels. The nonlinear function is estimated using maximum likelihood.

The effectiveness of the proposed approach is evaluated with nine multi-label data sets. Our experiments show that the proposed method achieves the lowest average rank on 0/1 *loss* and *multi-label accuracy*, and the second lowest on *Hammming loss* and *F-measure*, compared with eight other commonly used algorithms.

2.2 Approaches for Multi-label Classification

In this section, we briefly review the multi-label approaches that are existing competitors to our proposed approach.

The most common approach, Binary Relevance (BR) (Zhang and Zhou, 2005; Tsoumakas and Katakis, 2007), transforms a multi-label problem into separate binary problems. That is, using training data, BR constructs a binary classifier for each label independently. Suppose there are L possible labels. For a test observation, the prediction set of labels is obtained simply by combining the individual binary results. In other words, the predicted labelset is the union of the results predicted from the binary models. This approach requires one binary model for each label. The method has been adapted in many domains including text (Gonçalves and Quaresma, 2003), music (Li and Ogihara, 2003) and images (Boutell et al., 2004). One drawback of the basic binary approach is that it does not account for any correlation that may exist between labels, because the labels are modelled independently. Taking correlations into account is often critical for good prediction in multi-label problems (Godbole and Sarawagi, 2004; Ji et al., 2008).

Subset-Mapping ($SMBR$) (Schapire and Singer, 1999; Read et al., 2011) is a method related to BR. For a new observation, a vector of labels is obtained by the binary outputs of BR and the final prediction is made by the training labelset with the shortest $Hamming$ distance to the prediction set. $SMBR$ makes predictions by selecting labelsets observed in the training data. Once a labelset is obtained by BR, the latter process can be considered a nearest neighbor approach in the label space with $Hamming$ distance as the distance metric.

An extension of binary relevance is Classifier Chain (CC) (Read et al., 2011). CC fits labels sequentially using binary classifiers. Labels already predicted are included as fea-

tures in subsequent classifiers until all labels have been fit. Including previous predictions as features "chains" the classifiers together and also takes into account potential label correlations. However, the order of the labels in a chain affects the predictive performances. Read et al. (2011) also introduced the ensemble of classifier chains (ECC), where multiple CC are built with re-sampled training sets. The order of the labels in each CC is randomly chosen. The prediction label of an ECC is obtained by the majority vote of the CC models.

Label Powerset learning (LP) transforms a multi-label classification into a multi-class problem (Tsoumakas and Katakis, 2007). In other words, LP treats each labelset as a single label. The transformed problem requires a single classifier. Although LP captures correlations between labels, the number of classes in the transformed problem increases exponentially with the number of original labels. LP learning can only choose observed labelsets for predictions (Tsoumakas and Katakis, 2007; Read et al., 2008).

The random k-labelsets method, ($RAKEL$) (Tsoumakas and Vlahavas, 2007), is a variation on the LP approach. In a multi-label problem with L different labels, $RAKEL$ employs m multi-class models each of which considers $k(\leq L)$ randomly chosen labels, rather than the entire labelset. For a test observation, the prediction labelset is obtained by the majority vote of the results based on the m models. $RAKEL$ overcomes the problem that the number of multinomial classes increases exponentially as a function of the number of labels. It also considers interdependencies between labels by using multi-class models with subsets of the labels.

A hierarchy of multi-label classifiers ($HOMER$) (Tsoumakas et al., 2008) constructs a tree-shaped hierarchy by partitioning the labels recursively into smaller disjoint subsets (i.e. nodes) using a balanced clustering algorithm. Tsoumakas et al. (2008) proposed a balanced k means algorithm that extends the k means algorithm with an additional

constraint on the size of each cluster. Using the balanced algorithm places similar labels together into the same subsets. After that, $HOMER$ constructs a multi-label classifier for the labelsets in each node. For the prediction of a new observation, $HOMER$ follows a top-down recursive process from the root. A classifier on a non-root node is called only if the prediction of its parent node is positive. The final labelset is determined by the positive leaves (i.e. labels) whose parent nodes are all positive.

A popular learning algorithm based on the k Nearest Neighbours (kNN) approach is $MLKNN$ (Zhang and Zhou, 2007). Like other kNN-based methods, $MLKNN$ identifies the k nearest training observations in the feature space for a test observation. Then, for each label, $MLKNN$ estimates the prior probability and probability for the number of neighbours associated with the label. Using Bayes theorem, $MLKNN$ calculates the posterior probability from which a prediction is made.

The Conditional Bernoulli Mixtures (CBM) (Li et al., 2016) approach transforms a multi-label problem into a mixture of binary and multi-class problems. CBM divides the feature space into K regions and learns a multi-class classifier for the regional components as well as binary classifiers in each region. The posterior probability for a labelset is obtained by mixing the multi-class and multiple binary classifiers. The model parameters are estimated using the Expectation Maximization algorithm.

Multi-target classification approaches may also be used for multi-label classification. A number of multi-target learning methods use the predictive clustering tree (PCT) (Blockeel et al., 1998) as the base classifier (Kocev et al., 2007; Madjarov et al., 2016). A competitive approach is random forest of predictive clustering trees (RF-PCT) (Kocev et al., 2007). RF-PCT is a tree-based ensemble method using PCTs as base classifiers. Different PCTs are constructed by using different bootstrap training data (Breiman, 1996) and a random subset of the features during learning. It has been shown in Madjarov et al. (2012) that

RF-PCT is competitive for multi-label classification.

2.3 The Nearest Labelset Using Double Distances Approach

2.3.1 Hypercube View of a Multi-label Problem

In multi-label classification, we are given a set of possible output labels $\mathcal{L} = \{1, 2, ..., L\}$. Each observation with a feature vector $\mathbf{x} \in \mathbb{R}^p$ is associated with a subset of these labels. Equivalently, the subset can be described as $\mathbf{y} = (y^{(1)}, y^{(2)}, ..., y^{(L)})$, where $y^{(i)} = 1$ if label i is associated with the observation and $y^{(i)} = 0$ otherwise. A multi-label training data set is described as $T = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, ..., N\}$.

Any labelset $\mathbf{y}$ can be described as a vertex in the L-dimensional unit hypercube (Tai and Lin, 2012). Each component $y^{(i)}$ of $\mathbf{y}$ represents an axis of the hypercube. As an example, Figure 2.1 illustrates the label space of a multi-label problem with three labels ($y^{(1)}$, $y^{(2)}$, $y^{(3)}$).

Assume that the presence or absence of each label is modeled independently with a probabilistic classifier. For a new observation, the classifiers provide the probabilities, $p^{(1)}$, ..., $p^{(L)}$, that the corresponding labels are associated with the observation. Using the probability outputs, we may obtain a L-dimensional vector $\hat{\mathbf{p}} = (p^{(1)}, p^{(2)}, ..., p^{(L)})$. Every element of $\hat{\mathbf{p}}$ has a value from 0 to 1 and the vector $\hat{\mathbf{p}}$ is an inner point in the hypercube (see Figure 2.1). Given $\hat{\mathbf{p}}$ the prediction task is completed by assigning the inner point to a vertex of the cube.

For the new observation, we may calculate the Euclidean distance, $D_{\mathbf{y}_i}$, between $\hat{\mathbf{p}}$ and

each $\mathbf{y}_i$ (i.e. the labelset of the i^{th} training observation). In Figure 2.1, three training observations $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$ and the corresponding distances are shown. A small distance $D_{\mathbf{y}_i}$ indicates that $\mathbf{y}_i$ is likely to be the labelset for the new observation.

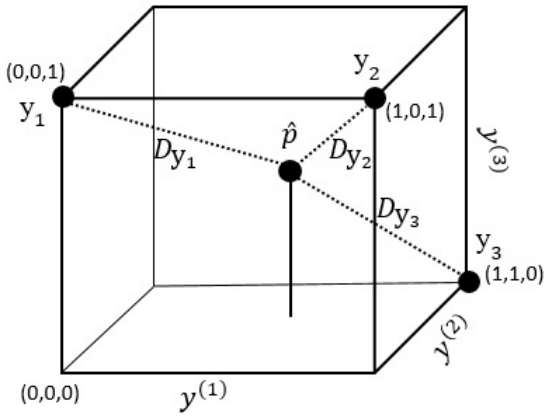


Figure 2.1: An illustration of the label space when $L = 3$. Each vertex represents a labelset. The inner point represents a fitted vector of an observation. $D_{\mathbf{y}_i}$ represents the distance between $\hat{\mathbf{p}}$ and $\mathbf{y}_i$.

2.3.2 Nearest Labelset Using Double Distances ($NLDD$)

In addition to computing the distance in the label space, $D_{\mathbf{y}_i}$ (as shown in Figure 2.1), we may also obtain the (Euclidean) distance in the feature space, denoted by $D_{\mathbf{x}_i}$. Note $D_{\mathbf{y}_i}$, $D_{\mathbf{x}_i} \geq 0$. The proposed method, $NLDD$, uses both $D_{\mathbf{x}}$ and $D_{\mathbf{y}}$ as predictors to find a training labelset that minimizes the expected loss. For each test observation, we define loss as the number of misclassified labels out of L labels. The expected loss is then $L\theta$ where $\theta = g(D_{\mathbf{x}}, D_{\mathbf{y}})$ represents the probability of misclassifying each label. The predicted

labelset, $\hat{\mathbf{y}}^*$, is the labelset observed in the training data that minimizes the expected loss:

$$\hat{\mathbf{y}}^* = \underset{\mathbf{y} \in T}{\operatorname{argmin}}\ L\, g(D_{\mathbf{x}}, D_{\mathbf{y}}) \tag{2.1}$$

The loss follows a binomial distribution with L and a parameter θ. We model $\theta = g(D_{\mathbf{x}}, D_{\mathbf{y}})$ as follows:

$$\log\left(\frac{\theta}{1-\theta}\right) = \beta_0 + \beta_1 D_{\mathbf{x}} + \beta_2 D_{\mathbf{y}} \tag{2.2}$$

where β_0, β_1 and β_2 are the model parameters. Larger values for β_1 and β_2 imply that θ becomes more sensitive to the distances in the feature and label spaces, respectively. The misclassification probability decreases as $D_{\mathbf{x}}$ and $D_{\mathbf{y}}$ approach zero.

A test observation with $D_{\mathbf{x}} = D_{\mathbf{y}} = 0$ has a duplicate observation in the training data (i.e. with identical features). In this case, the predicted probabilities for the test observation are either 0 or 1 and match the labels of the duplicate training observation. For such a "double"-duplicate observation (i.e. $D_{\mathbf{x}} = D_{\mathbf{y}} = 0$), the probability of misclassification is $1/(1 + e^{-\beta_0}) > 0$. As expected, the uncertainty in classifying a test observation with a "double-duplicate" training observation is greater than zero.

The model in (2.2) implies $g(D_{\mathbf{x}}, D_{\mathbf{y}}) = 1/(1 + e^{-(\beta_0 + \beta_1 D_{\mathbf{x}} + \beta_2 D_{\mathbf{y}})})$. Because $\log\left(\frac{\theta}{1-\theta}\right)$ is a monotone transformation of θ and L is a constant, the minimization problem in (2.1) is equivalent to

$$\hat{\mathbf{y}}^* = \underset{\mathbf{y} \in T}{\operatorname{argmin}}\ \beta_1 D_{\mathbf{x}} + \beta_2 D_{\mathbf{y}} \tag{2.3}$$

That is, $NLDD$ predicts by choosing the labelset of the training observation that minimizes the weighted sum of the distances. The prediction does not change if the argument $\beta_1 D_{\mathbf{x}} + \beta_2 D_{\mathbf{y}}$ is multiplied by a constant. Therefore, only the relative weight β_2/β_1 matters for the minimization. For prediction, the only remaining issue is how to estimate the weights.

2.3.3 Estimating the Relative Weights of the Two Distances

We need to estimate the parameters β_0, β_1 and β_2. This requires computing $D_{\mathbf{y}}$, but of course the outcomes in the test data are not known. We therefore split the training data, T, equally into two data sets, T_1 and T_2. T_2 is used for validation. Using T_1, we next fit a binary classifier to each of the L labels separately and obtain the labelset predictions (i.e. probability outcomes) for the observations in T_2. We then create a set of $(D_{\mathbf{x}}, D_{\mathbf{y}})$ by pairing observations in T_1 with those in T_2. Note that matching any single observation in T_2 to those in T_1 results in $N/2$ distance pairs. Most of the pairs are uninformative because the distance in either the feature space or the label space is very large. Moreover, since T_2 contains $N/2$ observations, the number of possible pairs is potentially large ($N^2/4$). Therefore, to reduce computational complexity, for each observation we only identify two pairs: the pair with the smallest distance in $\mathbf{x}$ and the pair with the smallest distance in $\mathbf{y}$. Note that more than two distance pairs may be used. Using two pairs is the smallest set. In case of ties in one distance, the pair with the smallest value in the other distance is chosen. More formally we identify the first pair m_{i_1} by

$$m_{i_1} = \underset{(D_x, D_y) \in W_{ix}}{\operatorname{argmin}} D_y$$

where W_{ix} is the set of pairs that are tied; i.e. that each corresponds to the minimum distance in D_x. Similarly, the second pair m_{i_2} is found by

$$m_{i_2} = \underset{(D_x, D_y) \in W_{iy}}{\operatorname{argmin}} D_x.$$

where W_{iy} is the set of labels that are tied with the minimal distance in D_y. Figure 2.2 illustrates an example of how to identify m_{i_1} and m_{i_2} for $N = 20$. Our goal was to identify the observation with the smallest distance in $\mathbf{x}$ and the observation with the smallest distance in $\mathbf{y}$. Note that m_{i_1} and m_{i_2} may be the same observation If we find a

single observation that minimizes both distances, we use just that observation. (A possible duplication of that observation is unlikely to make any difference in practice).

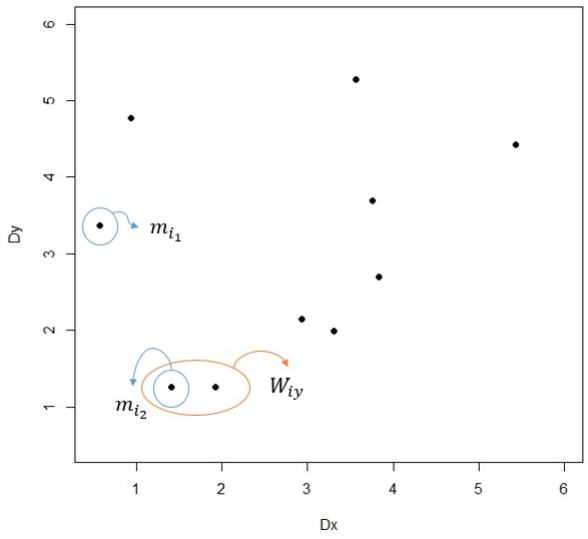


Figure 2.2: An illustration of how to identify m_{i_1} and m_{i_2} for $N = 20$. T_1 and T_2 contain 10 observations each. The 10 points in the scatter plot were obtained by calculating D_x and D_y between an observation in T_2 and the 10 observations in T_1. In this example two points have the lowest distance in D_y and are candidates for m_{i_2}. Among the candidates, the point with the lowest D_x is chosen.

The two pairs corresponding to the i^{th} observation in T_2 are denoted as the set $S_i = \{m_{i_1}, m_{i_2}\}$, and their union for all observations is denoted as $S = \bigcup_{i=1}^{N/2} S_i$. The binomial regression specified in (2.2) is performed on the observations in S and maximum likelihood estimators of the parameters are obtained. Algorithm 1 outlines the training procedure.

For the classification of a new observation, we first obtain $\hat{\mathbf{p}}$ using the probabilistic classifiers fitted to the training data T. $D_{\mathbf{x}_j}$ and $D_{\mathbf{y}_j}$ are obtained by matching the observation with the j^{th} training observation. Using the $MLEs$ $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\beta}_2$, we calculate $\hat{\theta}_j = \frac{e^{\hat{f}_j}}{1+e^{\hat{f}_j}}$

Algorithm 1 The training process of $NLDD$

Input: training data T, number of labels L

Output: probabilistic classifiers $h^{(i)}$, binomial regression g

Split T into T_1 and T_2

for $i = 1$ **to** L **do**

train probabilistic classifier $h^{(i)}$ based on T

train probabilistic classifier $h_*^{(i)}$ based on T_1

end for

$S, W \leftarrow \emptyset$

for each observation in T_2 **do**

obtain $\hat{\mathbf{p}} = (h_*^{(1)}(\mathbf{x}), ..., h_*^{(L)}(\mathbf{x}))$

for each observation in T_1 **do**

compute $D_{\mathbf{x}}$ and $D_{\mathbf{y}}$

$W \leftarrow W \cup (D_{\mathbf{x}}, D_{\mathbf{y}})$

end for

find $m_1, m_2 \in W$

update $S \leftarrow S \cup \{m_1, m_2\}$

end for

Fit $\log\left(\frac{\theta}{1-\theta}\right) = \beta_0 + \beta_1 D_{\mathbf{x}} + \beta_2 D_{\mathbf{y}}$ to S

Obtain $g : S \rightarrow \hat{\theta} = \frac{e^{\hat{f}}}{1+e^{\hat{f}}}$ where $\hat{f} = \hat{\beta}_0 + \hat{\beta}_1 D_{\mathbf{x}} + \hat{\beta}_2 D_{\mathbf{y}}$

where $\hat{f}_j = \hat{\beta}_0 + \hat{\beta}_1 D_{\mathbf{x}_j} + \hat{\beta}_2 D_{\mathbf{y}_j}$. The final prediction of the new observation is obtained by

$$\hat{\mathbf{y}} = \underset{\mathbf{y}_j \in T}{\operatorname{argmin}} \ \hat{E}(loss) = \underset{\mathbf{y}_j \in T}{\operatorname{argmin}} \ \hat{\theta}_j.$$

The second equality holds because $\hat{E}(loss) = L\hat{\theta}$ and L is a constant. As in LP, $NLDD$ chooses a training labelset as the predicted vector. Algorithm 2 outlines the classification procedure.

Algorithm 2 The classification process of $NLDD$

Input: new observation $\mathbf{x}$, binomial model g, probabilistic classifiers $h^{(i)}$, training data T of size N

Output: multi-label classification vector $\hat{\mathbf{y}}$

for $j = 1$ to N **do**

compute $\hat{\mathbf{p}} = (h^{(1)}(\mathbf{x}), ..., h^{(L)}(\mathbf{x}))$

compute $D_{\mathbf{x}_j}$ and $D_{\mathbf{y}_j}$

obtain $\hat{\theta}_j \leftarrow g(D_{\mathbf{x}_j}, D_{\mathbf{y}_j})$

end for

return $\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y}_j \in T}{\operatorname{argmin}} \ \hat{\theta}_j$

The training time of $NLDD$ is $O(L(f(d,N) + f(d,N/2) + g(d,N/2)) + N^2(d+L) + Nlog(k))$ where $O(f(d,N))$ is the complexity of each binary classifier with d features and N training observations, $O(g(d,N/2))$ is the complexity for predicting each label for T_2, $N^2(d+L)$ is the complexity for obtaining the distance pairs for the regression and $O(Nlog(k))$ is the complexity for fitting a binomial regression with k-digit precision of the parameters. T_1 and T_2 have $N/2$ observations respectively. $O(Lf(d,N/2))$ is the complexity for fitting binary classifiers using T_1 and obtaining the probability results for T_2 takes $O(Lg(d,N/2))$. For each observation of T_2, there are $N/2$ numbers of distance

pairs. This has complexity $O((N/2)(d+L))$. Since there are $N/2$ observations, overall it takes $O((N/2)(N/2)(d+L))$ or $O(N^2(d+L))$ when omitting the constant. Among the $N/2$ pairs for each observation of T_2, we only identify at most 2 pairs. This implies $N/2 \leq s \leq N$ where s is the number of elements in S. Each iteration of the Newton-Raphson method has a complexity of $O(N)$. For k-digit precision complexity $O(logk)$ is required (Ypma, 1995). Combined, the complexity for estimating the parameters with k-digit precision is $O(Nlog(k))$. In practice, however, this term is dominated by $N^2(d+L)$ as we can set $k << N$.

2.4 Experimental Evaluation

In this section we compare six algorithms for multi-label classification on nine data sets in terms of *Hamming loss*, 0/1 *loss*, *multi-label accuracy* and *F-measure*. We next introduce the data sets and the evaluation measures and then present the results of our experiments.

2.4.1 Data Sets

We evaluated the proposed approach using nine commonly used multi-label data sets from different domains. Table 3.1 shows basic statistics for each data set including its domain, numbers of labels and features. In the text data sets, all features are categorical (i.e. binary). The last column "lcard", short for label cardinality, represents the average number of labels associated with an observation. The data sets are ordered by $(|L| \cdot |X| \cdot |E|)$.

The *emotions* data set (Trohidis et al., 2008) consists of pieces of music with rhythmic and timbre features. Each observation is associated with up to 6 emotion labels such as

"sad-lonely", "amazed-surprised" and "happy-pleased". The *scene* data set (Boutell et al., 2004) consists of images with 294 visual features. Each image is associated with up to 6 labels including "mountain", "urban" and "beach". The *yeast* data set (Elisseeff and Weston, 2001) contains 2417 yeast genes in the Yeast Saccharomyces Cerevisiae. Each gene is represented by 103 features and is associated with a subset of 14 functional labels. The *medical* data set consists of documents that describe patient symptom histories. The data were made available in the Medical Natural language Processing Challenge in 2007. Each document is associated with a set of 45 disease codes. The *slashdot* data set consists of 3782 text observations with 22 labels obtained from Slashdot.org. The *enron* data set (Klimt and Yang, 2004) contains 1702 email messages from the Enron corporation employees. The emails were categorized into 53 labels. The *ohsumed* data set (Hersh et al., 1994) is a collection of medical research articles from MEDLINE database. We used the same data set as in Read et al. (2011) that contains 13929 observations and 23 labels. The *tmc*2007 data set (Srivastava and Zane-Ulman, 2005) contains 28596 aviation safety reports associated with up to 22 labels. Following Tsoumakas et al. (2011), we used a reduced version of the data set with 500 features. The *bibtex* data set (Katakis et al., 2008) consists of 7395 bibtex entries for automated tag suggestion. The entries were classified into 159 labels. All data sets are available online at: http://mulan.sourceforge.net/datasets-mlc.html and http://meka.sourceforge.net/#datasets.

2.4.2 Evaluation Metrics for Multi-label Classification

Multi-label classifiers can be evaluated with various loss functions. Here, four of the most popular criteria are used: *Hamming loss*, 0/1 *loss*, *multi-label accuracy* and *F-measure*. These criteria are defined in the following paragraphs.

name	domain	labels ($\|L\|$)	features ($\|X\|$)	examples ($\|E\|$)	lcards
emotions	music	6	72	593	1.87
scene	image	6	294	2407	1.07
yeast	biology	14	103	2417	4.24
medical	text	45	1449	978	1.25
slashdot	text	22	1079	3782	1.18
enron	text	53	1001	1702	3.37
ohsumed	text	23	1002	13929	1.66
tmc2007	text	22	500	28596	2.16
bibtex	text	159	1836	7395	2.40

Table 2.1: Multi-label data sets and their associated characteristics. Label cardinality (lcards) is the average number of labels associated with an observation

Let L be the number of labels in a multi-label problem. For a particular test observation, let $\mathbf{y} = (y^{(1)}, ..., y^{(L)})$ be the labelset where $y^{(j)} = 1$ if the j^{th} label is associated with the observation and 0 otherwise. Let $\hat{\mathbf{y}} = (\hat{y}^{(1)}, ..., \hat{y}^{(L)})$ be the predicted values obtained by any machine learning method. *Hamming loss* refers to the percentage of incorrect labels. The *Hamming loss* for the observation is

$$Hamming\ loss = 1 - \frac{1}{L}\sum_{j=1}^{L} \mathbb{1}\{y^{(j)} = \hat{y}^{(j)}\}$$

where $\mathbb{1}$ is the indicator function. Despite its simplicity, the *Hamming loss* may be less discriminative than other metrics. In practice, an observation is usually associated with a small subset of labels. As the elements of the L-dimensional label vector are mostly zero, even the empty set (i.e. zero vector) prediction may lead to a decent *Hamming loss*.

The 0/1 *loss* is 0 if all predicted labels match the true labels and 1 otherwise. Hence,

$$0/1 \; loss = 1 - \mathbb{1}\{\mathbf{y} = \hat{\mathbf{y}}\}.$$

Compared to other evaluation metrics, 0/1 *loss* is strict as all the L labels must match to the true ones simultaneously.

The *multi-label accuracy* (Godbole and Sarawagi, 2004) (also known as the *Jaccard* index) is defined as the number of labels counted in the intersection of the predicted and true labelsets divided by the number of labels counted in the union of the labelsets. That is,

$$Multi\text{-}label \; accuracy = \frac{|\mathbf{y} \cap \hat{\mathbf{y}}|}{|\mathbf{y} \cup \hat{\mathbf{y}}|}.$$

The *multi-label accuracy* measures the similarity between the true and predicted labelsets.

The *F-measure* is the harmonic mean of precision and recall. The *F-measure* is defined as

$$F\text{-}measure = \frac{2|\mathbf{y} \cap \hat{\mathbf{y}}|}{|\mathbf{y}| + |\hat{\mathbf{y}}|}.$$

The metrics above were defined for a single observation. On each metric, the overall value for an entire test data set is obtained by averaging out the individual values.

2.4.3 Experimental Setup

We compared our proposed method against BR, $SMBR$, ECC, $MLKNN$, $RAKEL$ and CBM. To train multi-label classifiers, the parameters recommended by the authors were used. In the case of $MLKNN$, we set the number of neighbors and the smoothing parameter to 10 and 1 respectively. For $RAKEL$, we set the number of separate models to $2L$ and the size of each sub-labelset to 3. For ECC, the number of CC models for each

ensemble was set to 10. For $HOMER$, the number of clusters was set to 3 as used in Liu et al. (2015). On the larger data sets (*ohsumed*, *tmc*2007 and *bibtex*), we fit ECC using reduced training data sets (75% of the observations and 50% of the features) as suggested in Read et al. (2011). On the same data sets, we ran $NLDD$ using 70% of the training data to reduce redundancy in learning.

For $NLDD$, BR, $SMBR$, ECC, $RAKEL$ and $HOMER$, support vector machines (SVM) (Vapnik, 2000) were chosen as a base classifier using unscaled variables with a linear kernel and tuning parameter $C = 1$. The SVM scores were converted into probabilities using Platt's method (Platt, 2000). The analysis was conducted in R (R Core Team, 2014) using the $e1071$ package (Meyer et al., 2014) and *utiml* (Rivolli, 2016) packages. For the data sets with less than 5,000 observations 10-fold cross validations (CV) were performed. On the larger data sets, we used 75/25 train/test splits. For fitting binomial regression models, we divided the training data sets at random into two parts of equal sizes.

For $RF\text{-}PCT$, we used the $Clus$[1] system. In the pre-pruning strategy of PCT, the significance level for the F-test was automatically chosen from {0.001, 0.005, 0.01, 0.05, 0.1, 0.125} using a reserved prune-set.

For implementing CBM we used a Java program[2] developed by the authors. The default settings (e.g. logistic regression and 10 iterations for the EM algorithm) were used on non-large data sets. For the large data sets *tmc*2007 and *bibtex*, the number of iterations was set to 5 and random feature reduction was applied as suggested by the developers. On each data set we used train/test split available at their website (https://github.com/cheng-li/pyramid).

To test the hypothesis that all classifiers perform equally, we used the Friedman test

[1] http://clus.sourceforge.net

[2] https://github.com/cheng-li/pyramid

as recommended by Demšar Demšar (2006). We then compared $NLDD$ with each of the other methods using Wilcoxon signed-rank tests. We adjusted p-values for multiple testing using Hochberg's method Hochberg (1988).

In $NLDD$, when calculating distances in the feature spaces we used the standardized features so that no particular features dominated distances. For a numerical feature variable x, the standardized variable z is obtained by $z = (x - \bar{x})/\text{sd}(x)$ where $\bar{x}$ and $\text{sd}(x)$ are the mean and standard deviation of x in the training data.

2.4.4 Results

Tables 2.2 to 2.5 summarize the results in terms of *Hamming loss*, 0/1 *loss*, *multi-label accuracy* and *F-measure*, respectively. We also ranked the algorithms for each metric.

According to the Friedman tests, the classifiers are not all equal ($p < 0.05$). The post-hoc analysis - adjusted for multiple testing - showed that $NLDD$ performed significantly better than $SMBR$ on all metrics, significantly better than BR, $RAKEL$ and $MLKNN$ on all but *Hamming loss*, significantly better than $HOMER$ on *Hamming loss* and 0/1 *loss*, and significantly better than ECC and $RF\text{-}PCT$ on 0/1 *loss*. On any evaluation metric, no method performed statistically significantly better than $NLDD$.

$NLDD$ achieved highest average ranks on 0/1 *loss* and *multi-label accuracy*, while ECC and $RF\text{-}PCT$ achieved the highest average ranks on the *F-measure* and *Hamming loss*, respectively. On both *F-measure* and *Hamming loss*, $NLDD$ achieved the second lowest (i.e. best) average ranks. CBM achieved the second lowest average rank on 0/1 *loss* and *multi-label accuracy*. The performance of CBM on the 0/1 *loss* was very variable achieving the highest rank on five out of nine data sets and the second worst on two data sets.

Table 2.6 shows the running time in seconds of the methods. On the non-large data sets, the relative differences of running time between $NLDD$ and BR tended to increase with the size of the data sets. On two of the large data sets, *ohsumend* and *tmc*2007, $NLDD$ required less time than BR as we only used 70% of the training data.

Data	BR	$SMBR$	$\boldsymbol{NLDD}$	ECC	$RAKEL$	$HOMER$	$RF{-}PCT$	$MLKNN$	CBM
emotions	0.1964(4)	0.1995(5)	0.1901(2)	0.2010(6)	0.1952(3)	0.2113(7)	0.1883(1)	0.2646(8)	0.3366(9)
scene	0.1042(7)	0.1298(9)	0.0948(5)	0.0939(4)	0.0895(2)	0.1087(8)	0.0882(1)	0.0903(3)	0.0953(6)
yeast	0.1990(5)	0.2048(6)	0.1902(1)	0.2056(7)	0.1964(4)	0.2544(9)	0.1916(2)	0.1952(3)	0.2130(8)
medical	0.0096(3)	0.0111(6)	0.0097(4)	0.0091(2)	0.0097(5)	0.0135(8)	0.0120(7)	0.0153(9)	0.0086(1)
slashdot	0.0467(5)	0.0541(8)	0.0452(4)	0.0473(6)	0.0439(2)	0.0552(9)	0.0444(3)	0.0518(7)	0.0436(1)
enron	0.0578(9)	0.0563(8)	0.0550(5)	0.0528(3)	0.0552(6)	0.0553(7)	0.0456(1)	0.0526(2)	0.0531(4)
ohsumed	0.0670(5)	0.0717(7)	0.0630(3)	0.0737(8)	0.0605(2)	0.0794(9)	0.0565(1)	0.0697(6)	0.0638(4)
tmc2007	0.0583(2)	0.0587(3)	0.0595(5)	0.0633(6)	0.0588(4)	0.0646(7)	0.0534(1)	0.0706(9)	0.0699(8)
bibtex	0.0158(8)	0.0151(7)	0.0134(1)	0.0147(6)	0.0150(5)	0.0205(9)	0.0135(2)	0.0139(4)	0.0138(3)
av. ranks	5.3	6.6	3.4	5.2	3.7	8.1	2.1	5.7	4.8

Table 2.2: *Hamming loss* (lower is better) averaged over 10 cross validations (with ranks in parentheses). The data sets are ordered as in Table 3.1. The results from the Wilcoxon test on whether or not any two results are statistically significant from one another are summarized at the bottom of the table.

We next look at the performance of $NLDD$ by whether or not the true labelsets were observed in the training data. A labelset has been observed if the exact labelset can be found in the training data and unobserved otherwise. Since $NLDD$ makes a prediction by choosing a training labelset, a predicted labelset can only be partially correct on an unobserved labelset. Table 2.7 compares the evaluation results of BR and $NLDD$ on two separate subsets of the test set of the *bibtex* data. The bibtex data were chosen because the data set contains by far the largest percentage of unobserved labelsets (33%) among the

Data	*BR*	*SMBR*	***NLDD***	*ECC*	*RAKEL*	*HOMER*	*RF−PCT*	*MLKNN*	*CBM*
emotions	0.7181(7)	0.7080(5)	0.6900(3)	0.7100(6)	0.6793(2)	0.6949(4)	0.6623(1)	0.8850(9)	0.7980(8)
scene	0.4674(9)	0.4242(7)	0.3190(1)	0.3511(3)	0.3640(4)	0.3769(6)	0.4362(8)	0.3702(5)	0.3211(2)
yeast	0.8940(8)	0.8180(6)	0.7484(1)	0.7977(3)	0.8130(4)	0.9768(9)	0.8212(7)	0.8179(5)	0.7514(2)
medical	0.3191(6)	0.3068(4)	0.2792(2)	0.3017(3)	0.3191(5)	0.3212(7)	0.3916(8)	0.4940(7)	0.2263(1)
slashdot	0.6452(7)	0.6253(5)	0.5232(2)	0.6000(4)	0.6277(6)	0.5970(3)	0.7967(8)	0.9386(9)	0.5127(1)
enron	0.9065(8)	0.8765(4)	0.8657(2)	0.8788(5)	0.9000(6)	0.9060(7)	0.8707(3)	0.9588(9)	0.8300(1)
ohsumed	0.7990(7)	0.7872(6)	0.7462(2)	0.8193(8)	0.7742(4)	0.7759(5)	0.7682(3)	0.9495(9)	0.7338(1)
tmc2007	0.7063(5)	0.7043(4)	0.7030(3)	0.7316(7)	0.7026(2)	0.7299(6)	0.6452(1)	0.7732(9)	0.7360(8)
bibtex	0.8504(6)	0.8201(3)	0.8081(2)	0.8391(4)	0.8413(5)	0.8994(7)	0.9134(8)	0.9441(9)	0.7815(1)
av. ranks	6.8	4.9	2.0	4.8	4.3	6.1	5.2	8.1	2.8

Table 2.3: 0/1 *loss* (lower is better) averaged over 10 cross validations (with ranks in parentheses). The loss is 0 if a predicted labelset matches the true labelset exactly and 1 otherwise. The results from the Wilcoxon test on whether or not any two results are statistically significant from one another are summarized at the bottom of the table.

data sets investigated. The test data set was split into subsets A and B; if the labelset of a test observation was an observed labelset, the observation was assigned to A; otherwise the observation was assigned to B. For all of the four metrics, $NLDD$ outperformed BR even though 33% of the labelsets in the test data were unobserved labelsets.

We next look at the three regression parameters the proposed method ($NLDD$) estimated (equation 2.2) for each data set in more detail. Table 2.8 displays the MLE of the parameters of the binomial model in each data set. In all data sets, the estimates of β_1 and β_2 were all positive. The positive slopes imply that the expected loss (or, equivalently the probability of misclassification for each label) decreases as $D_{\mathbf{x}}$ or $D_{\mathbf{y}}$ decreases.

From the values of $\hat{\beta}_0$ we may infer how low the expected loss is when either $D_{\mathbf{x}}$ or $D_{\mathbf{y}}$ is 0. For example, $\hat{\beta}_0 = -3.5023$ in the *scene* data set. If $D_{\mathbf{x}} = 0$ and $D_{\mathbf{y}} = 0$,

Data	*BR*	*SMBR*	***NLDD***	*ECC*	*RAKEL*	*HOMER*	*RF−PCT*	*MLKNN*	*CBM*
emotions	0.5248(7)	0.5467(6)	0.5624(2)	0.5587(3)	0.5548(4)	0.5787(1)	0.5523(5)	0.3253(9)	0.4033(8)
scene	0.6357(8)	0.6512(7)	0.7422(1)	0.6985(4)	0.6990(3)	0.6919(5)	0.5873(9)	0.6900(6)	0.7178(2)
yeast	0.4992(8)	0.5092(7)	0.5461(1)	0.5428(2)	0.5194(4)	0.4306(9)	0.5154(5)	0.5103(6)	0.5216(3)
medical	0.7655(6)	0.7676(5)	0.7991(2)	0.7934(3)	0.7643(7)	0.7694(4)	0.6747(9)	0.5787(8)	0.8167(1)
slashdot	0.4517(7)	0.4687(5)	0.5354(2)	0.5067(3)	0.4577(6)	0.4950(4)	0.2159(8)	0.0694(9)	0.5495(1)
enron	0.3974(8)	0.4226(5)	0.4122(6)	0.4708(1)	0.4088(7)	0.4273(4)	0.4527(2)	0.3175(9)	0.4297(3)
ohsumed	0.3848(7)	0.3968(5)	0.4105(4)	0.4316(2)	0.3940(6)	0.4220(3)	0.3409(8)	0.0798(9)	0.4918(1)
tmc2007	0.5750(5)	0.5784(4)	0.5692(6)	0.5670(7)	0.5710(3)	0.5738(2)	0.6074(1)	0.4719(9)	0.5186(8)
bibtex	0.3259(6)	0.3387(3)	0.3492(2)	0.3321(4)	0.3335(5)	0.2556(7)	0.1588(8)	0.1281(9)	0.3761(1)
av. ranks	6.7	4.9	2.9	3.3	5.1	4.7	6.0	8.3	3.1

Table 2.4: *Multi-label accuracy* (higher is better) averaged over 10 cross validations (with ranks in parentheses). The results from the Wilcoxon test on whether or not any two results are statistically significant from one another are summarized at the bottom of the table.

$\hat{p} = 0.0292$ because $\log \frac{\hat{p}}{1-\hat{p}} = -3.5023$. Hence $\hat{E}(loss) = L\hat{p} = 6 \cdot 0.0292 = 0.1752$. This is the expected number of mismatched labels for choosing a training labelset whose distances to the new observation are zero in both feature and label spaces. The results suggest the expected loss would be very small when classifying a new observation that had a duplicate in the training data ($D_{\mathbf{x}} = 0$) and whose labels are predicted with probability 1 and the predicted labelset was observed in the training data ($D_{\mathbf{y}} = 0$).

2.4.5 How *NLDD* Works Compared With *BR* Using the *yeast* Data

In this section, we illustrate how *NLDD* can outperform *BR* using the *yeast* data set. The *yeast* data set contains 14 distinct labels (i.e. $L = 14$). The binomial regression model for the expected misclassification rate based on the training data was obtained as

Data	*BR*	*SMBR*	***NLDD***	*ECC*	*RAKEL*	*HOMER*	*RF−PCT*	*MLKNN*	*CBM*
emotions	0.6033(7)	0.6291(5)	0.6446(3)	0.6477(2)	0.6316(4)	0.6699(1)	0.6283(6)	0.3989(9)	0.4723(8)
scene	0.6245(8)	0.6429(7)	0.7358(1)	0.7150(4)	0.6922(5)	0.7155(3)	0.5952(9)	0.6833(6)	0.7307(2)
yeast	0.6094(8)	0.6159(5)	0.6438(2)	0.6465(1)	0.6249(3)	0.5615(9)	0.6215(4)	0.6140(7)	0.6154(6)
medical	0.7945(6)	0.7957(5)	0.8268(2)	0.8257(3)	0.7928(7)	0.8005(4)	0.6966(8)	0.6030(9)	0.8310(1)
slashdot	0.5027(6)	0.5163(5)	0.5619(2)	0.5612(3)	0.5021(7)	0.5279(4)	0.2201(8)	0.0733(9)	0.5673(1)
enron	0.5119(8)	0.5299(4)	0.5200(7)	0.5852(1)	0.5224(5)	0.5459(3)	0.5619(2)	0.4259(9)	0.5220(6)
ohsumed	0.4529(7)	0.4546(6)	0.4758(4)	0.5238(1)	0.4550(5)	0.4973(2)	0.3813(8)	0.0910(9)	0.4942(3)
tmc2007	0.6662(4)	0.6703(3)	0.6552(7)	0.6635(5)	0.6596(6)	0.6722(2)	0.6875(1)	0.5561(9)	0.6013(8)
bibtex	0.3966(5)	0.3929(6)	0.4130(2)	0.4055(3)	0.4023(4)	0.3231(7)	0.1904(8)	0.1601(9)	0.4372(1)
av. ranks	6.6	5.1	3.3	2.6	5.1	3.8	6.0	8.4	4.0

Table 2.5: *F-measure* (higher is better) averaged over 10 cross validations (with ranks in parentheses). The results from the Wilcoxon test on whether or not any two results are statistically significant from one another are summarized at the bottom of the table.

$\log\left(\frac{\hat{\theta}}{1-\hat{\theta}}\right) = -3.9 + 0.12D_{\mathbf{x}} + 0.92D_{\mathbf{y}}$. For example, one of the test observations had true labelset (0,0,1,1,0,...,0,1,1,0). The BR approach predicted the labelset (0,0,1,0,...,0,1,1,0) failing to predict $y^{(4)}$ correctly. On the other hand, $NLDD$ chose the correct labelset (0,0,1,1,0,...,0,1,1,0), since the selected training observation had both small $D_{\mathbf{x}}$ and small $D_{\mathbf{y}}$. The labelset predicted by BR was not observed in the training data, meaning that $NLDD$ would not consider the labelset for a prediction. If only $D_{\mathbf{y}}$ was used without $D_{\mathbf{x}}$, another incorrect labelset (0,0,0,0,0,...,0,1,1,0) would be chosen. If only $D_{\mathbf{x}}$ was used without $D_{\mathbf{y}}$, another incorrect labelset (1,1,1,1,0,...,0,1,1,0) would be chosen.

Now consider another example where the labelset chosen by BR is observed in the training data. Specifically, a test observation had true labelset (1,1,0,0,0,1,1,1,0...,0,1,1,0). The BR approach predicted the labelset (1,1,0,...,0,1,1,0) failing to predict $y^{(6)}, y^{(7)}$ and $y^{(8)}$ correctly. $NLDD$ chose the correct labelset, since the chosen observation had both

Data	*BR*	*SMBR*	***NLDD***	*ECC*	*RAKEL*	*HOMER*	*RF-PCT*	*MLKNN*	*CBM*
emotions	19	19	27	40	21	14	9	4	23
scene	37	38	88	104	57	34	45	112	195
yeast	59	61	96	141	90	43	177	59	530
medical	43	44	101	312	73	17	54	93	1809
slashdot	52	57	428	280	104	40	66	1023	2540
enron	126	127	248	572	265	119	128	201	16232
ohsumed	22834	22987	12152	15799	37872	28784	799	10641	7588
tmc2007	21376	22145	16253	10023	23252	22340	1400	27394	38912
bibtex	2337	2466	2762	3574	5017	1220	2356	6280	48834

Table 2.6: Running times (seconds) on benchmark multi-label data sets

small $D_{\mathbf{y}}$ ($= 1.39$) and $D_{\mathbf{x}}$ ($= 4.24$) resulting in $\hat{\theta} = 0.103$. Because it was observed, $NLDD$ would choose the same labelset as BR if only $D_{\mathbf{y}}$ was used without $D_{\mathbf{x}}$. Despite the small $D_{\mathbf{y}}$ ($= 0.96$), this labelset was not chosen because the corresponding observation had a large $D_{\mathbf{x}}$ ($= 10.26$) resulting in $\hat{\theta} = 0.135$.

2.4.6 Scaling Up $NLDD$

As seen in Section 2.3.2, the time complexity of $NLDD$ is dependent on the size of the training data (N). In particular, the term $O(N^2(d+L))$ makes the complexity of $NLDD$ quadratic in N. For larger data sets the running time could be reduced by running the algorithm on a fraction of the N observations, but performance may be affected. This is investigated next.

Figure 2.3 illustrates the running time and the corresponding performance of $NLDD$ as a function of the percentage of N. For the result, we used the *tmc*2007 data with 75/25 train/test splits. After splitting, we randomly chose 10% - 100% of the training data and

	Subset A		Subset B		Total ($A \cup B$)	
	BR	$NLDD$	BR	$NLDD$	BR	$NLDD$
Hamming loss	0.0113	0.0091	0.0250	0.0224	0.0158	0.0134
0/1 *loss*	0.7804	0.7163	0.9958	1.0000	0.8504	0.8084
Multi-label accuracy	0.3807	0.4273	0.2118	0.1870	0.3259	0.3492
F-measure	0.4402	0.4785	0.3065	0.3058	0.3966	0.4130

Table 2.7: Evaluation results on the bibtex data set by whether or not the labelset was observed (Subset A) or unobserved (Subset B) in the training data. Subset A contains 67% of the test observations and subset B contains 33%. For *Hamming loss* and 0/1 *loss*, lower is better. For *Multi-label accuracy* and *F-measure*, higher is better.

ran $NLDD$ with the reduced data. As before, we used SVM with a linear kernel as the base classifier.

The result shows that $NLDD$ can obtain similar predictive performances for considerably less time. The running time increased quadratically as a function of N while the improvement of the performance of $NLDD$ appeared to converge. Using 60% of the training data, $NLDD$ achieved almost the same performance in the number of mismatched labels as using the full training data. Similar results were obtained on other large data sets.

2.5 Discussion

For the sample data sets selected, $NLDD$ achieved the lowest average ranks on 0/1 *loss* and *multi-label accuracy*. $NLDD$ performed significantly better than $SMBR$ on all of the

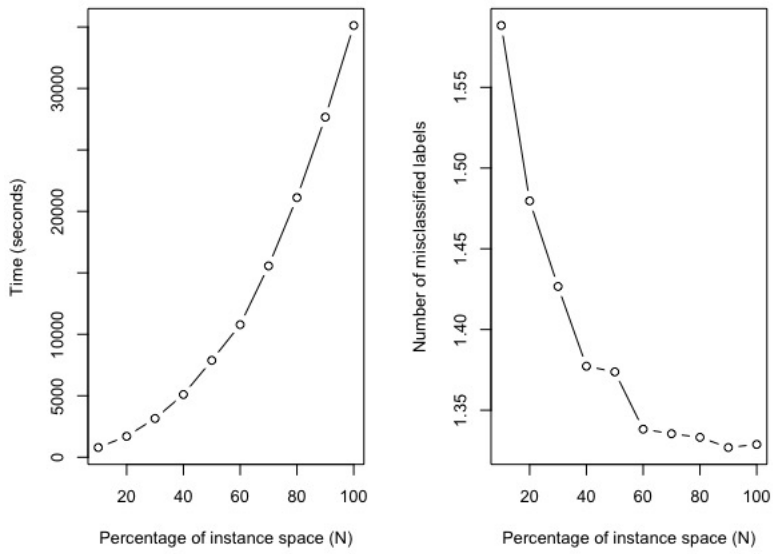


Figure 2.3: Running time (left) and the average number of mismatched labels (right) as a function of the percentage of the observation space for $NLDD$

Data	$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\beta}_2$
emotions	-2.6353	0.0321	1.0912
scene	-3.5023	0.0134	1.8269
yeast	-3.9053	0.1409	0.8546
medical	-5.5296	0.1089	1.6933
slashdot	-4.2503	0.1204	1.3925
enron	-3.8827	0.0316	0.7755
bibtex	-4.8436	0.0093	0.7264
ohsumed	-3.1341	0.0022	0.9855
tmc2007	-3.6862	0.0370	1.1056

Table 2.8: The maximum likelihood estimates of the parameters of equation 2.2 averaged over 10 cross validations

four metrics. *NLDD* also significantly outperformed *BR*, *RAKEL* and *MLKNN* on all but *Hamming loss*, *HOMER* on *Hamming loss* and 0/1 *loss*, and *ECC* and *RF-PCT* on 0/1 *loss*. *NLDD* achieved lower average ranks than *CBM* on all of the four metrics (not statistically significant).

Like *BR*, *NLDD* uses outputs of independent binary classifiers. Using the distances in the feature and label spaces in binomial regression, *NLDD* can make more accurate predictions than *BR*. *NLDD* was also significantly superior to *SMBR*, which is similar to *NLDD* in the sense that it makes predictions by choosing training labelsets using binary classifiers. *SMBR* is based on the label space only, while *NLDD* uses the distances in the feature space as well.

Like *LP*, the proposed method treats each training labelset as a different class of a

single-label problem in the prediction stage. Using a training labelset as a predicted vector, the proposed approach takes potentially high order label correlations into account.

In fitting the binomial regression, $NLDD$ restricts the fit of the binomial model to distance pairs with low distances in the feature and label spaces. This dramatically reduces the size of the data used for regression fitting. In the *yeast* data set, the training data T contained 2178 observations. Since we equally divided the training data into T_1 and T_2, each of them contained 1089 observations. Hence the number of possible observations available for fitting is $1089 * 1089 = 1,185,921$. On the other hand, $NLDD$ used only $2,018$ observations which is less than 0.2% of all observations.

$NLDD$ has higher time than BR. The relative differences of running time between $NLDD$ and BR depended on the size of the training data (N). The number of labels and features had less impact on the differences, as the complexity of $NLDD$ is linear in them. For the larger data sets, we reduced the running time of $NLDD$ by using a subset (70%) of the training data. The results of *ohsumed* and *tmc*2007 data sets show that $NLDD$ with reduced data can perform fast compared to not only BR but also the other methods on large data problems.

Because $NLDD$ makes a prediction by choosing a training labelset, the prediction label vector is confined to a labelset appearing in the training data. If a new observation has a true labelset unobserved in the training data, there will be at least one incorrect predicted label. Even so, $NLDD$ beat the other methods on average. How frequently an unobserved labelset occurs depends on the data set. For most data sets, less than 5% of the test data contained labelsets not observed in the training data. In other words, most of the labelsets of the test observations could be found in the training data. However, for the bibtex data set about 33% of the test data contained unobserved labelsets. As seen in Table 2.7, when the true labelsets of the test observations were not observed in the training data (subset

B), BR performed slightly better than $NLDD$ in terms of 0/1 *loss*, *multi-label accuracy* and *F-measure*. On the other hand, when the true labelsets of the test observations were observed in the training data (subset A), $NLDD$ outperformed BR on all of the metrics. Combined, $NLDD$ achieved higher performances than BR on the entire test data. For the *bibtex* data set, $NLDD$ performed the best on *Hamming loss* and the second best on the other three metrics.

$NLDD$ uses binomial regression to estimate the parameters. This setup assumes that the observations in S are independent. While it turned out that this assumption worked well in practice, dependencies may arise between the two pairs of a given S_i. If required this dependency could be modeled using, for example, generalized estimating equations (Liang and Zeger, 1986). We examined results from a GEE model on the selected data using an exchangeable correlation structure. The estimates were almost the same and the prediction results were unchanged. The analogous results are not shown.

For prediction, the minimization in (2.3) only requires the estimates of the coefficients β_1 and β_2 which determine the tradeoff between $D_{\mathbf{x}}$ and $D_{\mathbf{y}}$. The estimate of β_0 is not needed. However, estimating β_0 allows us to estimate the probability of a misclassification of a label for an observation, $\hat{\theta}$. Such an assessment of uncertainty of the prediction can be useful. For example, one might only want to classify observations where the probability of misclassification is below a certain threshold value.

$NLDD$ uses a linear model for binomial regression specified in 2.2. To investigate how the performance of $NLDD$ changes in nonlinear models, we also considered a model: $\log\left(\frac{\theta}{1-\theta}\right) = \beta_0 + D_{\mathbf{x}}^{\beta_1} \cdot D_{\mathbf{y}}^{\beta_2}$ in which the distances are combined in a multiplicative way. The difference of prediction results obtained by the linear and multiplicative models was small. The analogous results are not shown.

While SVM was employed as the base classifier, other algorithms could be chosen provided the classifier can estimate posterior probabilities rather than just scores. Better predictions from the binary classifiers will make distances in the label space more useful and hence lead to a better performance.

Chapter 3

$kCNN$: a new algorithm for classification based on conditional nearest neighbors

3.1 Introduction

This chapter concerns nearest neighbor-based nonparametric approaches to multi-class classification. Nonparametric classifiers are often used when it is difficult to make assumptions about the class distribution for the problem. The k-nearest neighbor (kNN) approach (Fix and Hodges, 1951) is one of the most popular nonparametric approaches (Wu et al., 2008). For an input $\mathbf{x}$, the kNN algorithm identifies k objects in the training data that are closest to $\mathbf{x}$ in a predefined metric and makes a prediction by majority vote from the classes of the k objects. Although the kNN method is simple and does not require a priori knowledge about the class distributions, kNN has been successfully applied in many prob-

lems such as character recognition (Belongie et al., 2002), image processing (Mensink et al., 2013) and bioinformatics (Raymer et al., 2003; Maji, 2011). A number of experiments on different classification problems have demonstrated its competitive performance (Ripley, 2007). Moreover, it has been shown that the error rate (or misclassification rate) of kNN converges to the optimal Bayes error rate when the number of training observations N and the number of neighbors k increase and $k/N \to 0$ (Cover and Hart, 1967). Approaches to improving the kNN method include weighted kNN (Dudani, 1976; Gou et al., 2012), condensed nearest neighbor (Gowda and Krishna, 1979), rank nearest neighbor (Bagui et al., 2003), clustered kNN (Yong et al., 2009) and prototype based nearest neighbor (Garcia et al., 2012). A detailed survey of the literature about kNN can be found in (Bhatia and Vandana, 2010).

A successful extension of the kNN method is the local mean based k nearest neighbor approach ($LMkNN$) (Mitani and Hamamoto, 2006). The $LMkNN$ method (Mitani and Hamamoto, 2006) calculates the local mean vector for each class and uses them as the class prototypes for prediction. Let $\mathbf{x}_{w|i}$ be the w^{th} nearest neighbor of class c_i $(i = 1, ..., L)$ where L is the number of classes. Given a fixed k, the $LMkNN$ computes $\bar{\mathbf{x}}_i$ the local mean vector for class c_i as

$$\bar{\mathbf{x}}_i^{(k)} = \frac{1}{k} \sum_{w=1}^{k} \mathbf{x}_{w|i}. \tag{3.1}$$

For prediction, $LMkNN$ chooses class $\hat{c}$ if

$$\hat{c} = \underset{i}{\operatorname{argmin}} \, |\mathbf{x} - \bar{\mathbf{x}}_i^{(k)}| \tag{3.2}$$

That is, the distance between $\mathbf{x}$ and each local mean is calculated and the class corresponding to the smallest distance is assigned to $\mathbf{x}$. Empirical evidence suggests that

compared to kNN, $LMkNN$ is robust to outliers when the training data are small (Mitani and Hamamoto, 2006). The idea of $LMkNN$ has been applied to many other methods such as pseudo nearest neighbor (Zeng et al., 2009), local mean-based pseudo k-nearest neighbor (Gou et al., 2014), group-based classification (Samsudin and Bradley, 2010), discriminant analysis (Yang et al., 2011). Recently, an extension of $LMkNN$, the multi-local means-based k-harmonic nearest neighbor ($MLM\text{-}kHNN$) (Pan et al., 2017), was introduced. Instead of a single local mean vector, $MLM\text{-}kHNN$ combines multiple local mean vectors using the harmonic mean metric. For each class c_i, $MLM\text{-}kHNN$ calculates k local mean vectors $\bar{\mathbf{x}}_i^{(1)}, \bar{\mathbf{x}}_i^{(2)}, ..., \bar{\mathbf{x}}_i^{(k)}$. The harmonic mean distance of the local mean vectors is obtained by

$$HMD(\mathbf{x}, \bar{\mathbf{x}}_i^{(k)}) = \frac{k}{\sum_{w=1}^{k} \frac{1}{|\mathbf{x} - \bar{\mathbf{x}}_i^{(w)}|}} \tag{3.3}$$

For prediction, $MLM\text{-}kHNN$ chooses class $\hat{c}$ if

$$\hat{c} = \underset{i}{\operatorname{argmin}}\ HMD(\mathbf{x}, \bar{\mathbf{x}}_i^{(k)}). \tag{3.4}$$

Unlike $LMkNN$, $MLM\text{-}kHNN$ computes k different local mean vectors in each class. $MLM\text{-}kHNN$ calculates their harmonic mean distance to $\mathbf{x}$ and assigns the class with the minimum distance. An experimental study showed that $MLM\text{-}kHNN$ achieves high classification accuracy and is less sensitive to the parameter k compared to other kNN-based methods.

In this chapter, we propose a new nonparametric classifier, k conditional nearest neighbor ($kCNN$), based on nearest neighbors conditional on each class. For any positive integer k, the proposed method estimates posterior probabilities using the k^{th} nearest neighbor in each class. We show that classification based on those posteriors is approximately Bayes optimal for a two-class problem. Furthermore, we demonstrate that the classification approach converges in probability to the Bayes classifier as the size of the training data

increases. We also introduce an ensemble of $kCNN$ that combines $kCNN$ classifiers with different values for k. Our experiments on benchmark data sets show that the proposed methods perform on average better than kNN, $LMkNN$ and MLM-$kHNN$ in terms of the error rate.

3.2 Methods

3.2.1 The k Conditional Nearest Neighbor Algorithm

In multi-class classification, an observation with a feature vector $\mathbf{x} \in \mathbb{R}^p$ is associated with one of the possible classes $c_1, ..., c_L$. We assume a set of training data containing N classified observations. For any $\mathbf{x}$ and a given k, we denote by $\mathbf{x}_{k|i}$ the k^{th} nearest neighbor of class c_i $(i = 1, ..., L)$. Let $d(\mathbf{x}, \mathbf{x}_{k|i}) = |\mathbf{x} - \mathbf{x}_{k|i}|$ be the (Euclidean) distance between $\mathbf{x}$ and $\mathbf{x}_{k|i}$. Figure 3.1 illustrates an example that points out the distance between $\mathbf{x}$ and the second nearest neighbor (i.e. $k = 2$) of each class.

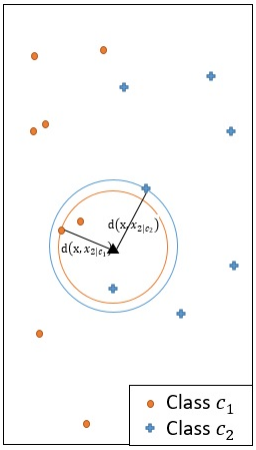


Figure 3.1: An illustrative example of $d(\mathbf{x}, \mathbf{x}_{k|i})$, i=1,2, when $k = 2$

Consider a hypersphere with radius $d(\mathbf{x}, \mathbf{x}_{k|i})$ centered at $\mathbf{x}$. By the definition of $\mathbf{x}_{k|i}$, the hypersphere contains k observations of class c_i. We may approximate the local conditional density $p(\mathbf{x}|c_i)$ as

$$\hat{p}(\mathbf{x}|c_i) = \frac{k}{N_i V_i} \tag{3.5}$$

where V_i is the volume of the hypersphere with radius $d(\mathbf{x}, \mathbf{x}_{k|i})$ centered at $\mathbf{x}$ and N_i represents the number of observations classified as class c_i. This approximation was also introduced in (Fukunaga and Hostetler, 1975). The approximation assumes that $p(\mathbf{x}|c_i)$ is nearly constant within the hypersphere of volume V_i. Using the prior $\hat{p}(c_i) \approx \frac{N_i}{N}$ where $N = \sum_{i=1}^{L} N_i$ and Bayes theorem, the approximate posterior may be obtained as

$$\hat{p}_k(c_i|\mathbf{x}) = \frac{\hat{p}(c_i)\hat{p}(\mathbf{x}|c_i)}{\hat{p}(\mathbf{x})} = \frac{1}{\hat{p}(\mathbf{x})}\frac{k}{NV_i}. \tag{3.6}$$

Because $\sum_{i=1}^{L} \hat{p}(c_i|\mathbf{x}) = 1$, we have $\hat{p}(x) = \sum_{i=1}^{L} \frac{k}{NV_i}$. Then, $\hat{p}_k(c_i|\mathbf{x})$ may be obtained as

$$\hat{p}_k(c_i|\mathbf{x}) = \frac{\frac{k}{NV_i}}{\sum_{j=1}^{L} \frac{k}{NV_j}} = \frac{d(\mathbf{x}, \mathbf{x}_{k|i})^{-p}}{\sum_{j=1}^{L} d(\mathbf{x}, \mathbf{x}_{k|j})^{-p}}$$

since $V_i \propto d(\mathbf{x}, \mathbf{x}_{k|i})^p$. The class with the shortest distance among the L distances has the highest posterior.

Smoothing parameters can improve predictive accuracy (e.g., LaPlace smoothing for Naive Bayes algorithm (Mitchell, 1997, Chapter 6.9). We introduce an optional tuning parameter, r, as follows:

$$\hat{p}_k(c_i|\mathbf{x}) = \frac{d(\mathbf{x}, \mathbf{x}_{k|i})^{-p/r}}{\sum_{j=1}^{L} d(\mathbf{x}, \mathbf{x}_{k|j})^{-p/r}} \tag{3.7}$$

where $r \geq 1$ controls the influence of the dimension of the feature space p. As r increases, each posterior converges to $1/L$. That is, increasing r smoothes the posterior estimates.

The k conditional nearest neighbor ($kCNN$) approach classifies $\mathbf{x}$ into the class with the largest estimated posterior probability. That is, class $\hat{c}$ is assigned to $\mathbf{x}$ if

$$\hat{c} = \underset{i}{\text{argmax}}\ \hat{p}_k(c_i|\mathbf{x}).$$

The proposed classifier is equivalent to kNN when $k = 1$. We summarize the $kCNN$ classifier in Algorithm 3.

Algorithm 3 The k conditional nearest neighbor algorithm

Input: A training data set D, an observation vector $\mathbf{x}$ with dimension p, a positive integer k, parameter r, a distance metric d

for $i = 1$ **to** L **do**

(a) From D, select $\mathbf{x}_{k|i}$, the k^{th} nearest neighbor of $\mathbf{x}$ for class c_i

(b) Calculate $d(\mathbf{x}, \mathbf{x}_{k|i})$, the distance between $\mathbf{x}$ and $\mathbf{x}_{k|i}$

end for

for $i = 1$ **to** L **do**

Obtain $\hat{p}_k(c_i|\mathbf{x}) \leftarrow \frac{d(\mathbf{x},\mathbf{x}_{k|i})^{-p/r}}{\sum_{j=1}^{L} d(\mathbf{x},\mathbf{x}_{k|j})^{-p/r}}$

end for

Classify $\mathbf{x}$ into $\hat{c}$ if $\hat{c} = \underset{i}{\text{argmax}}\ \hat{p}_k(c_i|\mathbf{x})$

Note that r does not affect the classification. However, we will show in Section 3.3 that the tuning parameter affects the classification of the ensemble of $kCNN$, which is presented in Section 3.2.4.

Figure 3.2 illustrates an example of a two-class classification problem. For a given k, the method calculates the distance between $\mathbf{x}$ and the k^{th} nearest neighbor of each class. When $k = 1$ and $k = 3$, class c_2 has a larger posterior probability than c_1 as the corresponding distance is shorter. When $k = 2$, however, the posterior for class c_1 is greater.

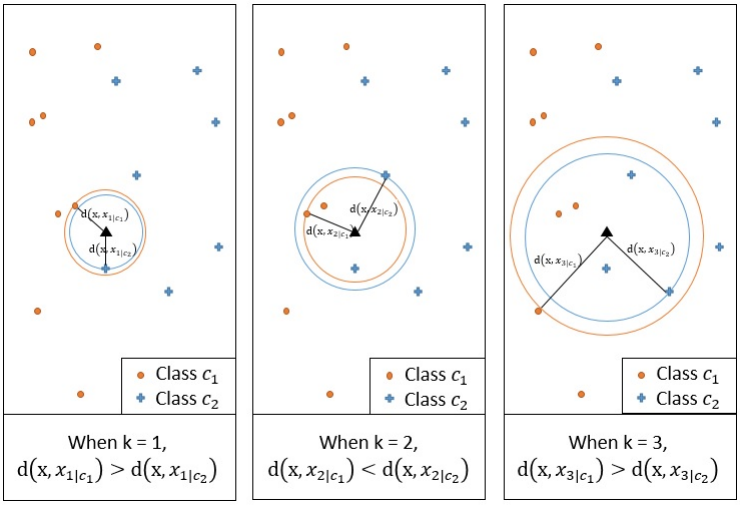


Figure 3.2: Comparing posteriors using distances between $\mathbf{x}$ and conditional nearest neighbors. Class c_2 has higher posteriors than class c_1 when $k = 1$ and 3 while class c_1 has a higher posterior when $k = 2$.

3.2.2 Convergence of $kCNN$

Theorem (convergence of $kCNN$): Consider a two-class problem with c_1 and c_2 where $p(c_1) > 0$ and $p(c_2) > 0$. Assume that $p(\mathbf{x}|c_i)$ $(i = 1, 2)$ is continuous on $\mathbb{R}^p$. If the following conditions (a) $k \to \infty$, and (b) $\frac{k}{\min_i N_i} \to 0$ are satisfied, then for any $\mathbf{x}$ where $p(\mathbf{x}) > 0$, $kCNN$ with $r = 1$ converges in probability to the Bayes classifier.

Proof: Since $kCNN$ makes predictions by approximate posteriors in (3.6), it is sufficient to show that $\hat{p}_k(c_i|\mathbf{x})$ converges in probability to the true posterior.

We first consider the convergence of the prior estimate $\hat{p}(c_i) = N_i/N$. Let $c^{(j)}$ be the class of the j^{th} training observation. The prior estimate may be described as $\hat{p}(c_i) = \frac{1}{N}\sum_{j=1}^{N} I(c^{(j)} = c_i)$ where I is the indicator function. Hence, by the weak law of large

numbers, $\hat{p}(c_i) \xrightarrow{p}$[1] $p(c_i)$.

We next show that the approximation $\hat{p}(\mathbf{x}|c_i)$ in equation (3.5) converges in probability to the true conditional density function. Let $f_N(\mathbf{x}) = \frac{k}{NV}$ be an estimate of the density function $f(\mathbf{x})$ where V is the volume of the hypersphere centered at $\mathbf{x}$ containing k training observations. Loftsgaarden and Quesenberry (1965) showed that $f_N(\mathbf{x})$ converges in probability to $f(\mathbf{x})$ if $k \to \infty$ and $\frac{k}{N} \to 0$ as N increases. We may apply this result to the convergence of the conditional density functions. By the second condition, both $\frac{k}{N_1}$ and $\frac{k}{N_2}$ converge to zero. Hence, $\hat{p}(\mathbf{x}|c_i)$ converges in probability to the true conditional density function $p(\mathbf{x}|c_i)$.

Since $\hat{p}(c_i) \xrightarrow{p} p(c_i)$ and $\hat{p}(\mathbf{x}|c_i) \xrightarrow{p} p(\mathbf{x}|c_i)$, $\hat{p}(\mathbf{x}) = \sum_{i=1}^{2} \hat{p}(c_i)\hat{p}(\mathbf{x}|c_i) \xrightarrow{p} \sum_{i=1}^{2} p(c_i)p(\mathbf{x}|c_i) = p(\mathbf{x})$. Hence, the approximate posterior in (3.6) converges in probability to the true posterior. This implies that $kCNN$ converges in probability to the Bayes classifier.

3.2.3 Time complexity of $kCNN$

The time complexity of kNN is $O(Np + Nk)$ (Zuo et al., 2008) ($O(Np)$ for computing distances and $O(Nk)$ for finding the k nearest neighbors and completing the classification). In the classification stage, $kCNN$ (a) calculates the distances between the test observation to all training observations from each class, (b) identifies the k^{th} nearest neighbor from each class, and (c) calculates posterior estimates by comparing the L distances and assigns the test observation to the class with the highest posterior estimate. Step (a) requires $O(N_1p+...+N_Lp) = O(Np)$ multiplications. Step (b) requires $O(N_1k+...+N_Lk) = O(Nk)$ comparisons. Step (c) requires $O(L)$ sum and comparison operations. Therefore, the time complexity for $kCNN$ is $O(Np + Nk + L)$. In practice the $O(L)$ component is dominated

[1] $A \xrightarrow{p} B$ means A converges in probability to B.

by the other components, since L is usually much smaller than N. That is, the difference of the complexities between kNN and $kCNN$ is small.

3.2.4 Ensemble of $kCNN$

The illustrative example in Figure 3.2 shows that the classification is affected by the choice of k. Therefore, we propose an ensemble version of $kCNN$ that combines the multiple $kCNN$ algorithms with different values of k. Ensembles are well known as a method for improving predictive performance (Wu et al., 2008; Rokach, 2010). The ensemble of k conditional nearest neighbor ($EkCNN$) method makes a prediction based on the averaged posteriors for different values of k. These values are now indexed by w: $w = 1, ..., k$. In the ensemble $EkCNN$, k represents the number of ensemble members. Suppose that posterior probability $\hat{p}_w(c_i|\mathbf{x})$ is estimated by (3.7) for each $w = 1, ..., k$. For a new observation $\mathbf{x}$ the predicted class $\hat{c}$ is determined by

$$\hat{c} = \underset{c_i}{\operatorname{argmax}}\ \hat{p}(c_i|\mathbf{x}) = \underset{c_i}{\operatorname{argmax}}\ \frac{1}{k}\sum_{w=1}^{k}\hat{p}_w(c_i|\mathbf{x}).$$

That is, $EkCNN$ assigns $\mathbf{x}$ to the class with the highest average posterior estimate. Using multiple values of k makes the prediction less reliant on single k.

The complexity of $EkCNN$ may be obtained analogously to steps (a)-(c) in Section 3.2.3. The complexities of $EkCNN$ required in step (a) and (b) are the same as those of $kCNN$. In step (c), $EkCNN$ requires $O(kL)$ sum and comparison operations. Hence, the complexity of $EkCNN$ is $O(Np + Nk + kL)$.

3.3 Experimental evaluation

3.3.1 Data sets

We evaluated the proposed approaches using real benchmark data sets available at the UCI machine learning repository (Lichman, 2013). Table 3.1 shows basic statistics of each data set including its numbers of classes and features. All data sets are available online at: https://archive.ics.uci.edu/ml/datasets.html. The data sets are ordered by the number of observations.

3.3.2 Experimental setup

We compared $kCNN$ and $EkCNN$ against kNN, $LMkNN$ and $MLM\text{-}kHNN$. For $EkCNN$, we used $r = p$ where p is the number of features of the data set. For $kCNN$ and $EkCNN$, we added $\epsilon = 10^{-7}$ to each distance in equation (3.7) to avoid the division of zero when the distance is zero.

The analysis was conducted in R (R Core Team, 2014). For assessing the performance of the classifiers, we used 10-fold cross validation for each data. In the experiments, we varied the size of the neighborhood k from 1 to 15. For each method, the optimal value of k has to be determined based on the training data only. To that end, each training fold of the cross-validation (i.e., 90% of the data) was split into two random parts: internal training data (2/3) and internal validation data (1/3). The optimal k was the value that minimized classification error on the internal validation set.

We applied the Wilcoxon signed-rank test (Wilcoxon, 1945; Demšar, 2006) to carry out the pairwise comparisons of the methods over multiple data sets because unlike the t–test

name	features	classes	observations
Wine	13	3	178
Parkins	22	2	195
Cancer	24	2	198
Sonar	60	2	208
Seeds	7	3	210
Haberman	3	2	306
Ecoli	7	8	336
Blood	4	2	748
Diabetes	8	2	768
Vehicle	18	4	846
German	24	2	1000
Yeast	8	10	1484
Image	19	7	2310
Wave	21	2	5000
Magic	10	2	19020

Table 3.1: 15 benchmark data sets and their associated characteristics

it does not make a distributional assumption. Also, the Wilcoxon test is more robust to outliers than the t–test (Demšar, 2006). The Wilcoxon test results report whether or not any two methods were ranked differently across data sets. Each test was one-sided at a significance level of 0.05.

3.3.3 Results

Table 3.2 summarizes the error rate (or misclassification rate) of each approach on each data set under the optimized value of k. Note that different approaches may achieve the lowest error rate at different values of k. $EkCNN$ performed best on 8 out of the 15 data sets and $kCNN$ performed best on 3 data sets. $EkCNN$ achieved the lowest (i.e. best) average rank and $kCNN$ the second lowest average rank. In the cases where $kCNN$ performed the best, $EkCNN$ was the second best method. According to the Wilcoxon test, $EkCNN$ had a significantly lower (i.e. better) rank than kNN, $LMkNN$ and $kCNN$ with p-values 0.0005, 0.0042 and 0.0035 respectively. There was marginal evidence that $EkCNN$ had a lower average rank than $MLM\text{-}kHNN$ (p-value = 0.0535). Also, $kCNN$ performed significantly better than kNN and $LMkNN$ with p-values 0.002 and 0.013 respectively.

Equation (3.7) contains a tuning parameter r. As mentioned above, increasing r smoothes posterior estimates. For the results of $EkCNN$ presented in Table 3.2, we chose $r = p$ for all data sets. While not shown here, using $r = p$ resulted in lower or equal error rates compared with using $r = 1$ on 14 out of 15 data sets. Specifying $r = p$ reduced the error rate up to 6% relative to the error rate for $r = 1$.

For the *ecoli* and *yeast* data sets, some classes contained only a few observations. As k increased, k became larger than the number of training observations for those sparse classes. Considering that the k^{th} nearest neighbors of those classes were not available,

$kCNN$ assigned the zero posterior probability to those classes. The posterior probabilities for the other classes were estimated using the k^{th} nearest neighbors of the available classes.

Figure 3.3 and 3.6 illustrate the error rate of each method on each data set varying with k. Note that the different error rates of the methods at $k = 1$ on data sets *Haberman* and *Blood* were due to ties. Compared to kNN, $EkCNN$ tends to outperform throughout the whole range of k. Also, $kCNN$-3 outperformed $kCNN$-2 for most data sets and choices of k.

3.3.4 Illustrating the choice of r and ϵ on the sonar data set

In this section, we investigated the impact of r on error rate for the *sonar* data set. The *sonar* data set has the largest number of features ($p = 60$) among the 15 data sets presented in Table 3.1.

Figure 3.7 shows that the error rate varied little for small values of k. For this data set, larger values of r are consistently preferable to smaller values. Note that error rates for $r = 60$ were almost identical to those for $r = 100$.

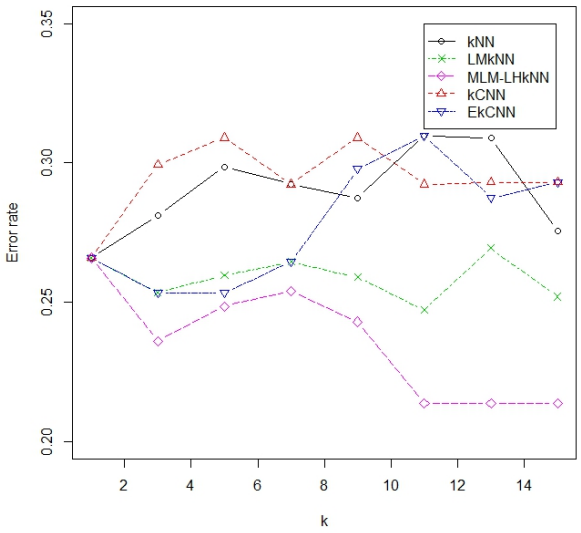
In our experiments, we added a tiny value $\epsilon = 10^{-7}$ to each distance to avoid the division of zero. Figure 3.8 shows that the estimate is not sensitive to the exact value of ϵ when ϵ is small. The error rate effectively does not change whether ϵ is very small (e.g. 0.01) or tiny (e.g. 0.0000001).

3.4 Exploring properties of $kCNN$ via simulation

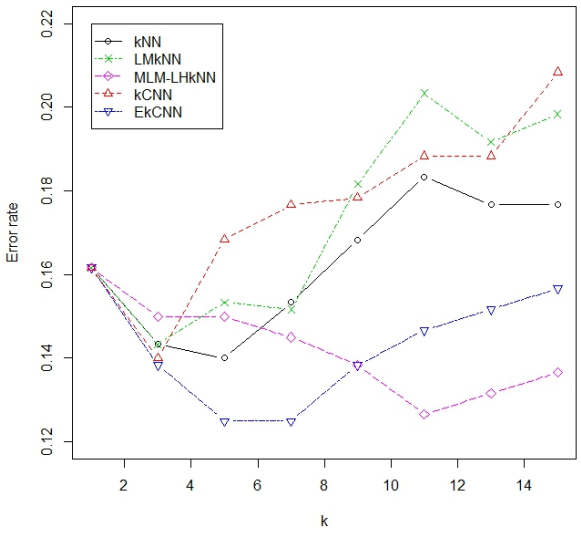
In the following subsections, we investigate $kCNN$'s decision boundary and posterior probability using simulation. Further, we also discuss where $kCNN$ beats kNN for posterior

	kNN	*LMkNN*	*MLM-kHNN*	*kCNN*	*EkCNN*
Wine	0.2871	0.2819	**0.2361**	0.2770	0.2534
Parkins	0.1783	0.1983	0.1833	0.1783	**0.1710**
Cancer	0.2782	0.3006	0.2927	0.2524	**0.2410**
Sonar	0.1815	0.1820	**0.1534**	0.1767	0.1666
Seeds	0.1500	0.0952	0.1000	0.1000	**0.0901**
Haberman	0.2769	0.3305	0.3388	**0.2572**	0.2604
Ecoli	0.1365	0.1482	0.1335	0.1394	**0.1305**
Blood	0.2438	0.2433	0.3208	0.2432	**0.2207**
Vehicle	0.3666	**0.3028**	0.3087	0.3643	0.3560
Diabetes	0.2643	0.2629	0.2759	0.2616	**0.2560**
German	0.3200	0.3200	0.3120	**0.3020**	0.3100
Yeast	0.4143	0.4219	0.4191	0.4029	**0.3812**
Image	0.0346	0.0337	**0.0316**	0.0346	0.0346
Wave	0.1590	0.1522	0.1606	**0.1478**	0.1520
Magic	0.1856	0.1962	0.1859	0.1854	**0.1780**
Average	0.2321	0.2313	0.2302	0.2223	**0.2143**
Ranking	3.8	3.8	3.3	2.5	**1.7**

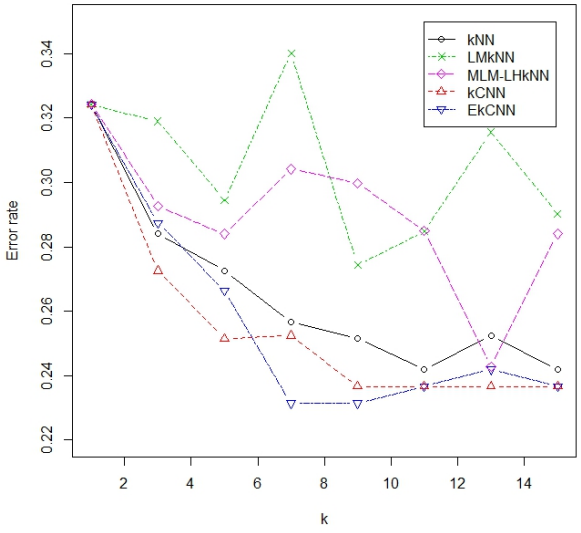
Table 3.2: The lowest error rates of each method on benchmark data. "Ranking" refers to the average ranking score of each method over the fifteen data sets. Lower is better.



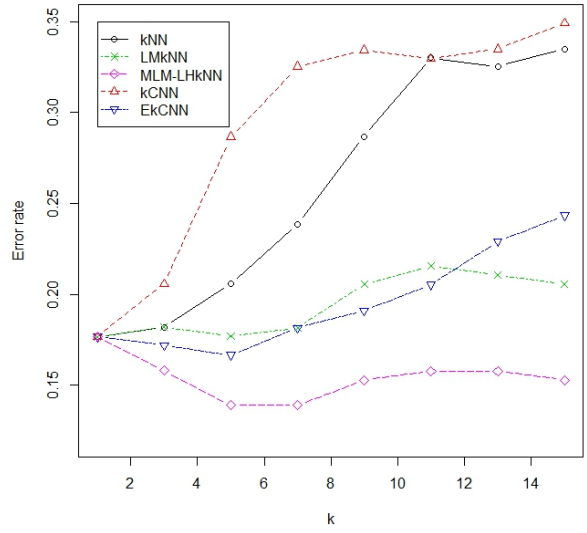
(a) Wine



(b) Parkins

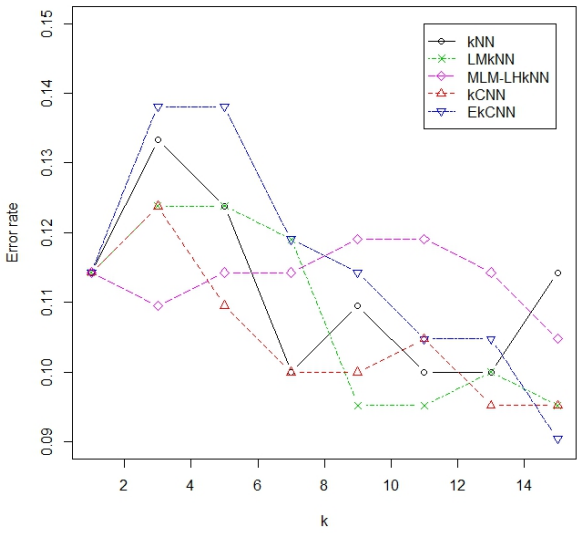


(c) Cancer

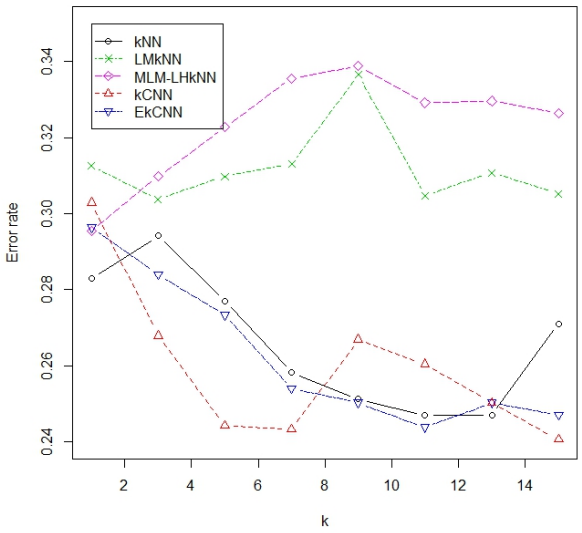


(d) Sonar

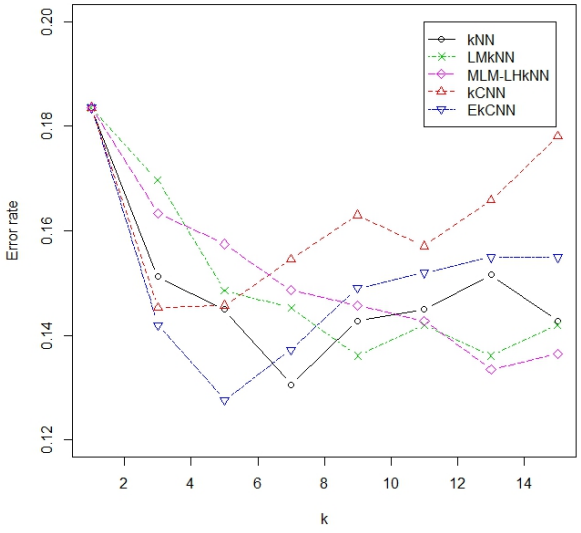
Figure 3.3: Error rates averaged over 10 cross validations for the *wine*, *parkins*, *cancer* and *sonar* data sets.



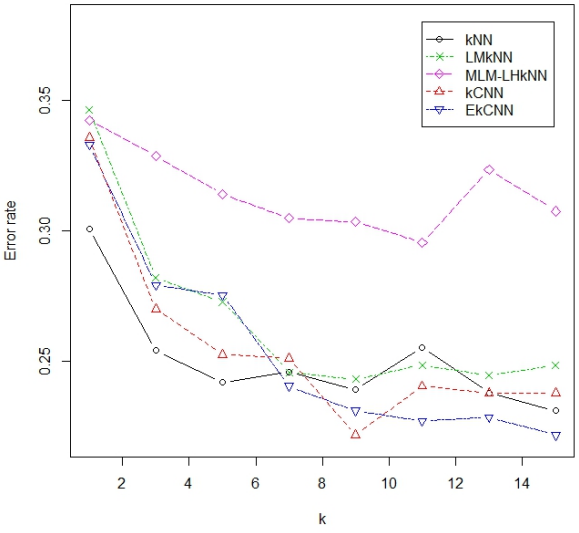
(a) Seeds



(b) Haberman

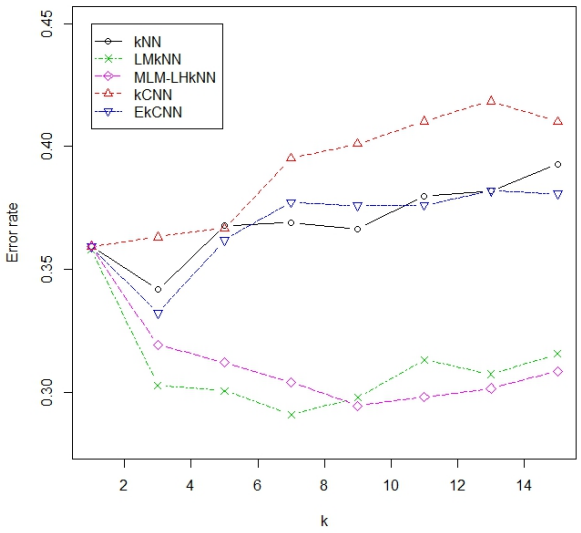


(c) Ecoli

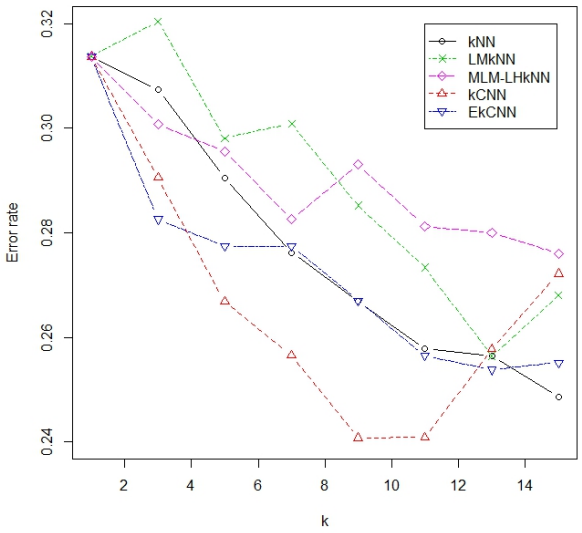


(d) Blood

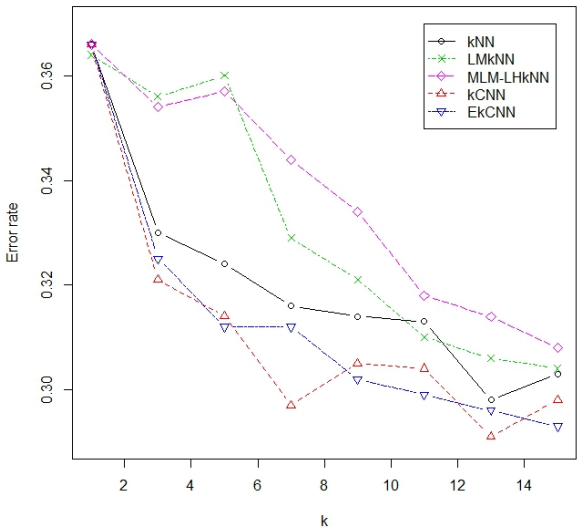
Figure 3.4: Error rates averaged over 10 cross validations for the *seeds*, *haberman*, *ecoli* and *blood* data sets.



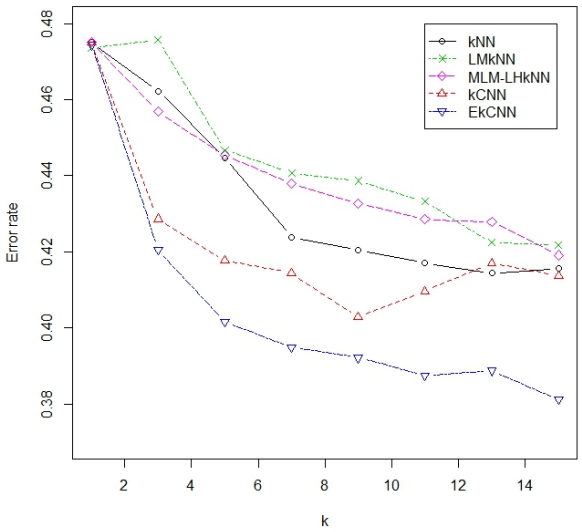
(a) Vehicle



(b) Diabetes

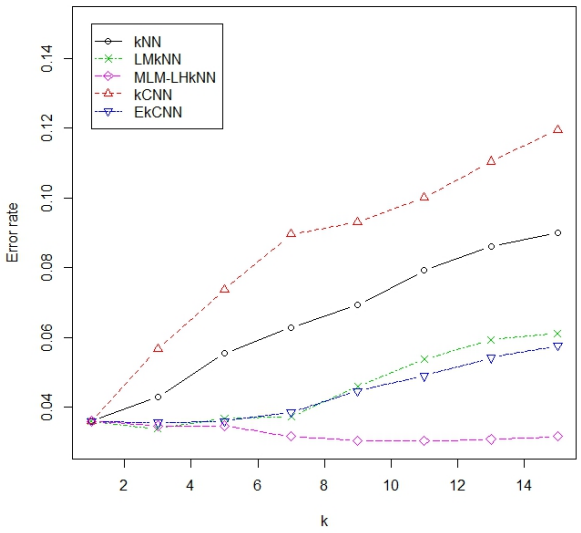


(c) German

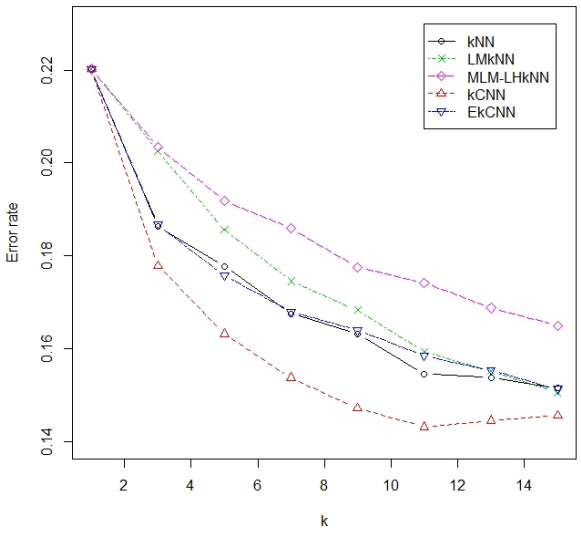


(d) Yeast

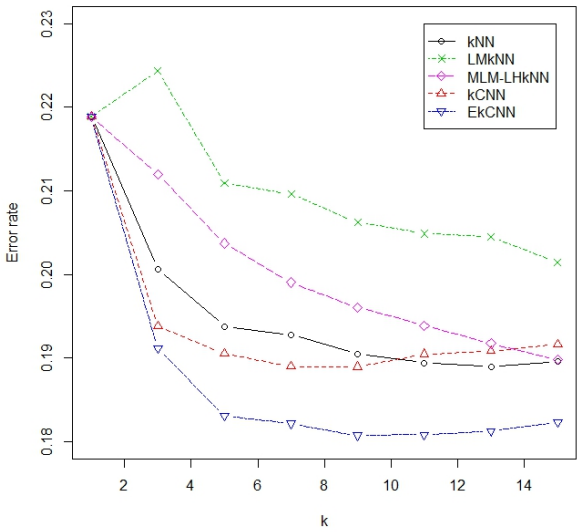
Figure 3.5: Error rates averaged over 10 cross validations for the *vehicle*, *diabetes*, *german* and *yeast* data sets.



(a) Image



(b) Wave



(c) Magic

Figure 3.6: Error rates averaged over 10 cross validations for the *image*, *wave* and *magic* data sets.

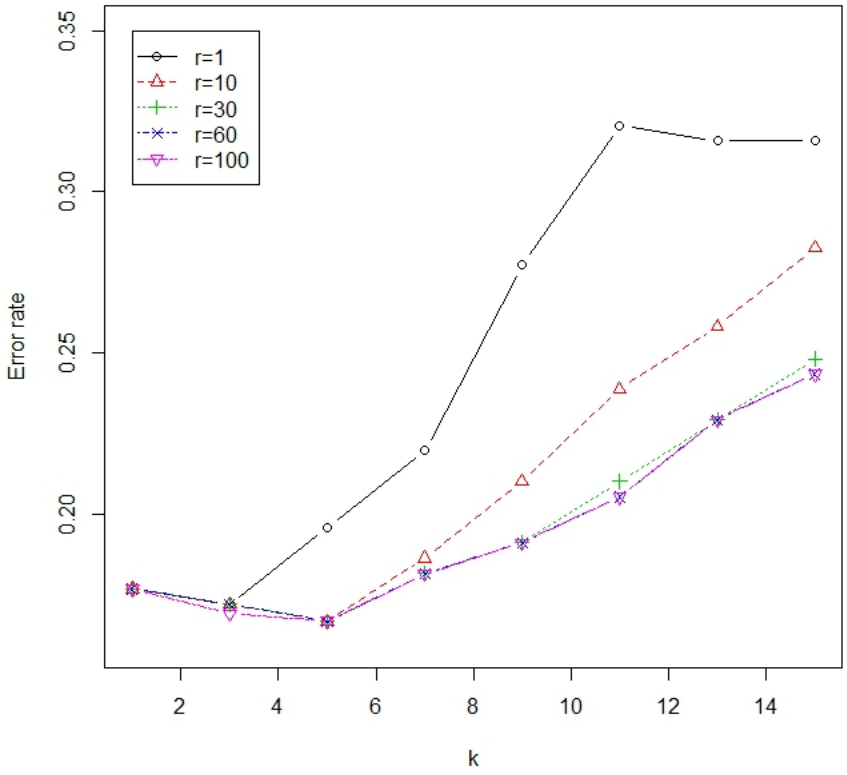


Figure 3.7: Impact of the tuning parameter r on error rates using the *sonar* data set.

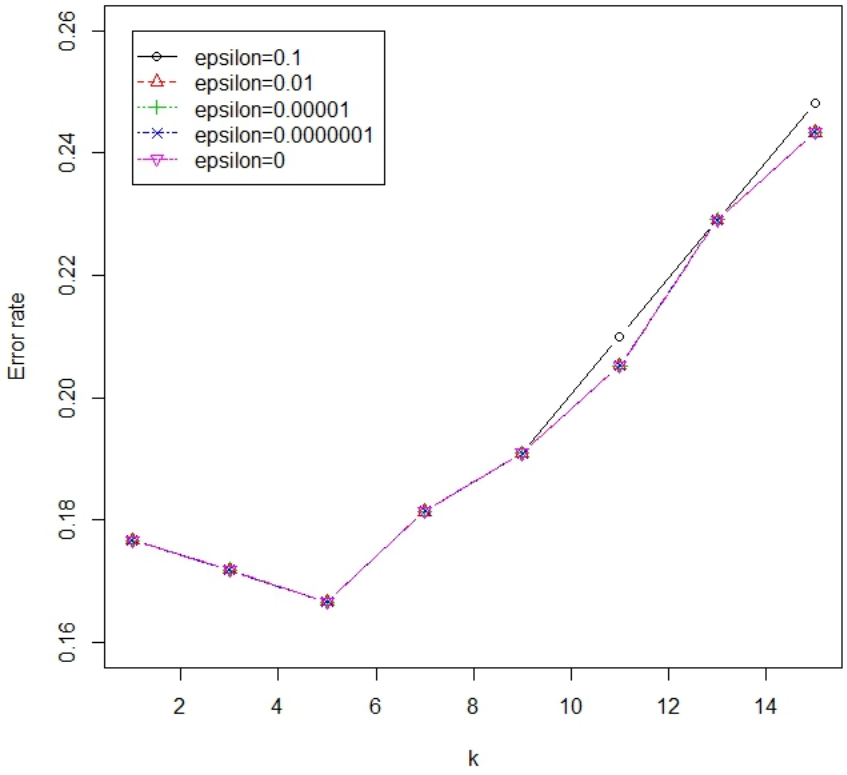


Figure 3.8: Impact of ϵ on error rates using the *sonar* data set.

estimation.

3.4.1 Decision boundary of $kCNN$ and $EkCNN$ with varying k

This section illustrates that the decision boundary is more smooth as k increases for both $kCNN$ and $EkCNN$. We used a simulated data set from Friedman et al. (2001). The classification problem contains two classes and two real value features.

Figure 3.9 shows the decision boundary of $kCNN$ with different k (solid curve) and the optimal Bayes decision boundary (dashed red curve). Increasing k resulted in smoother decision boundaries. However, when k is too large (e.g., $k = 30$ in this example), the decision boundary was overly smooth.

Analogously, Figure 3.10 shows the decision boundary of $EkCNN$ at $r = 2$ and different values k. Similar to $kCNN$, the decision boundary was smoothed as k increased. However, the magnitude of the changes was relatively less variable. For example, the decision boundaries of $EkCNN$ at $k = 10$ and $k = 30$ were similar, while those of $kCNN$ were quite different.

3.4.2 Comparison of the posterior probability distribution of kNN and $kCNN$

Rather than considering classification, this section compares $kCNN$ with kNN in terms of posterior probabilities. Probabilities are of interest, for example, when evaluating the entropy criterion. Using the same data set as in Section 3.4.1, we plot the full posterior probability contours of kNN and $kCNN$ in Figure 3.11. We set $r = p = 2$ for $kCNN$. For $k = 1$, as expected, the posteriors estimated by kNN was always either 0 or 1. By

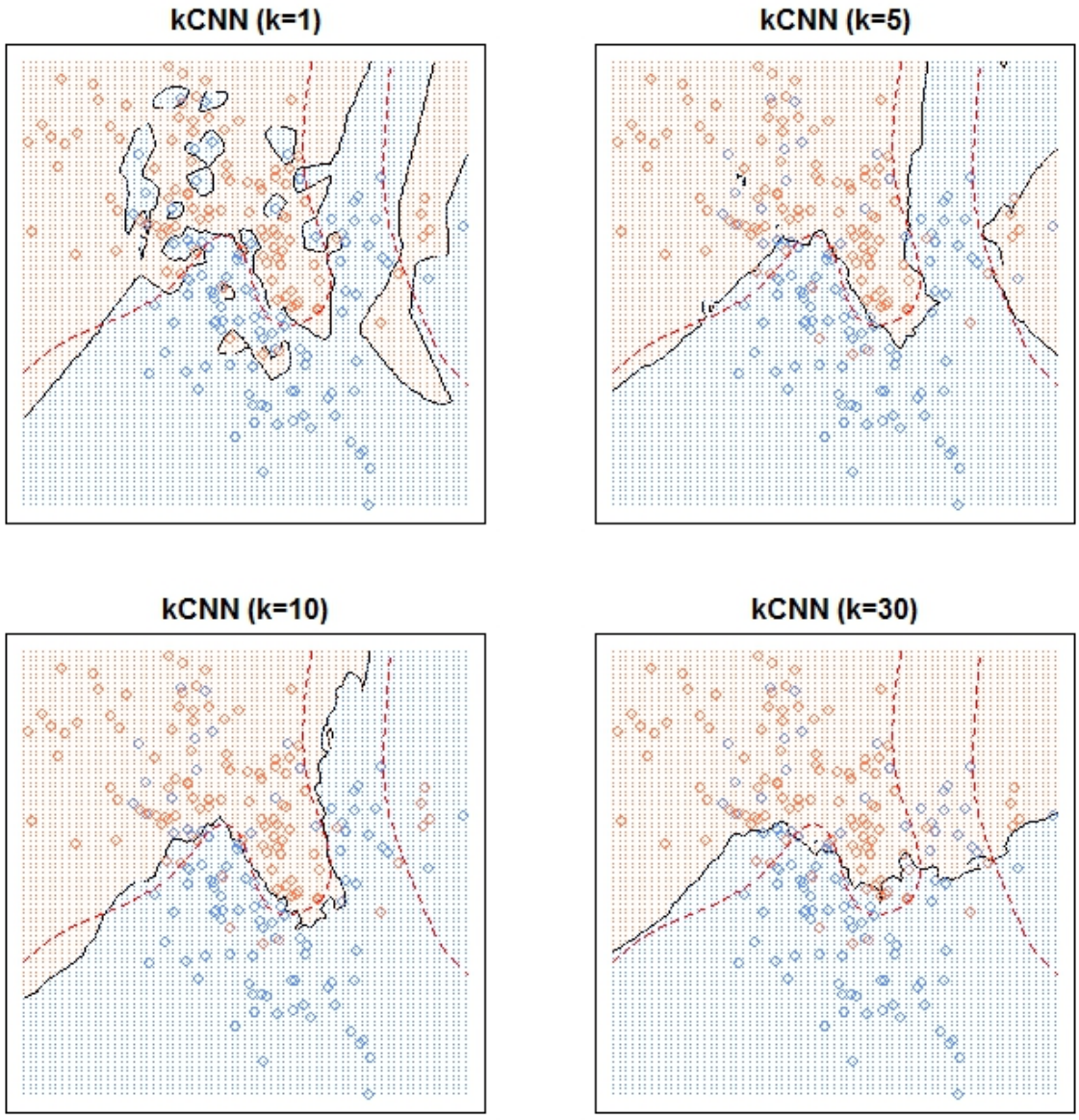


Figure 3.9: $kCNN$ on the simulated data with different choices of k. The broken red curve is the Bayes decision boundary.

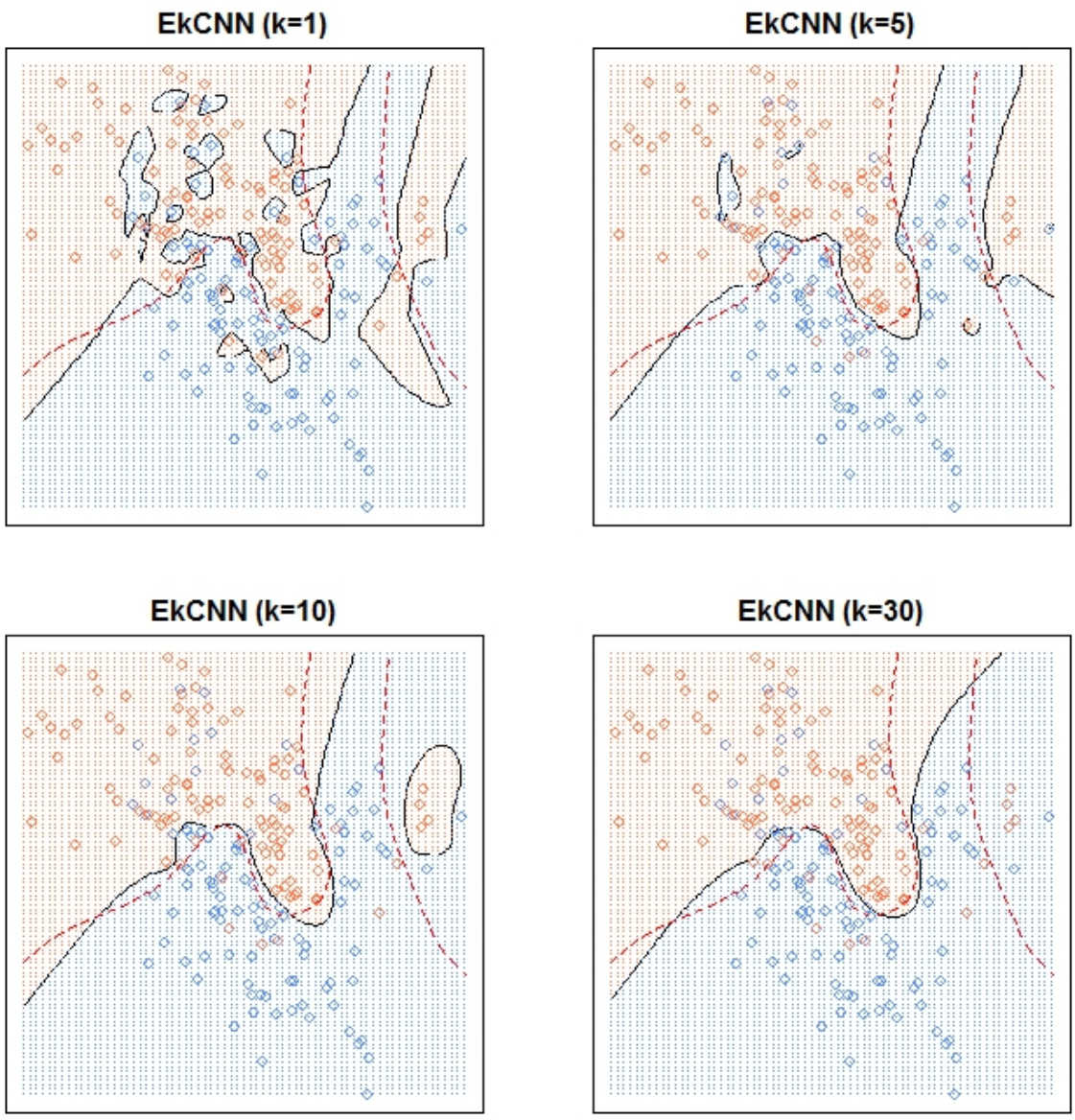


Figure 3.10: *EkCNN* on the simulated data with different choices of k. The broken red curve is the Bayes decision boundary.

contrast, $kCNN$ provided less extreme posterior results even at $k = 1$. The posterior probabilities changed more gradually.

When $k = 3$, posterior probabilities from kNN jumped between four possible values (0, 1/3, 2/3, 1), whereas those from $kCNN$ were much smoother. The result shows that unlike kNN, $kCNN$ can produce smooth posterior probability fields even at small values of k.

3.4.3 Under what circumstances does $kCNN$ beat kNN for posterior estimation?

$kCNN$ may be useful when the true posterior distribution has a full range of probabilities rather than near dichotomous probabilities (close to 0 or 1). This occurs when the distributions of the classes substantially overlap. When the distribution of each class is well separated, for any data point the classification probabilities will be (near) 1 for one class and (near) 0 for the other classes. Otherwise, when the distributions overlap, the classification probabilities will be less extreme.

We conducted a small simulation to illustrate that $kCNN$ is preferable to kNN when k is small and the distributions of the classes overlap. Assume that instances from each class are independently distributed following a multivariate normal distribution. Denote by μ_i the mean vector and by $\sum_i$ the covariance matrix of class c_i. The parameters were given as

$$\mu_1 = (0, 0, ..., 0), \ \sum\nolimits_1 = I_p$$
$$\mu_2 = (\frac{s}{\sqrt{p}}, ..., \frac{s}{\sqrt{p}}), \ \sum\nolimits_2 = I_p$$

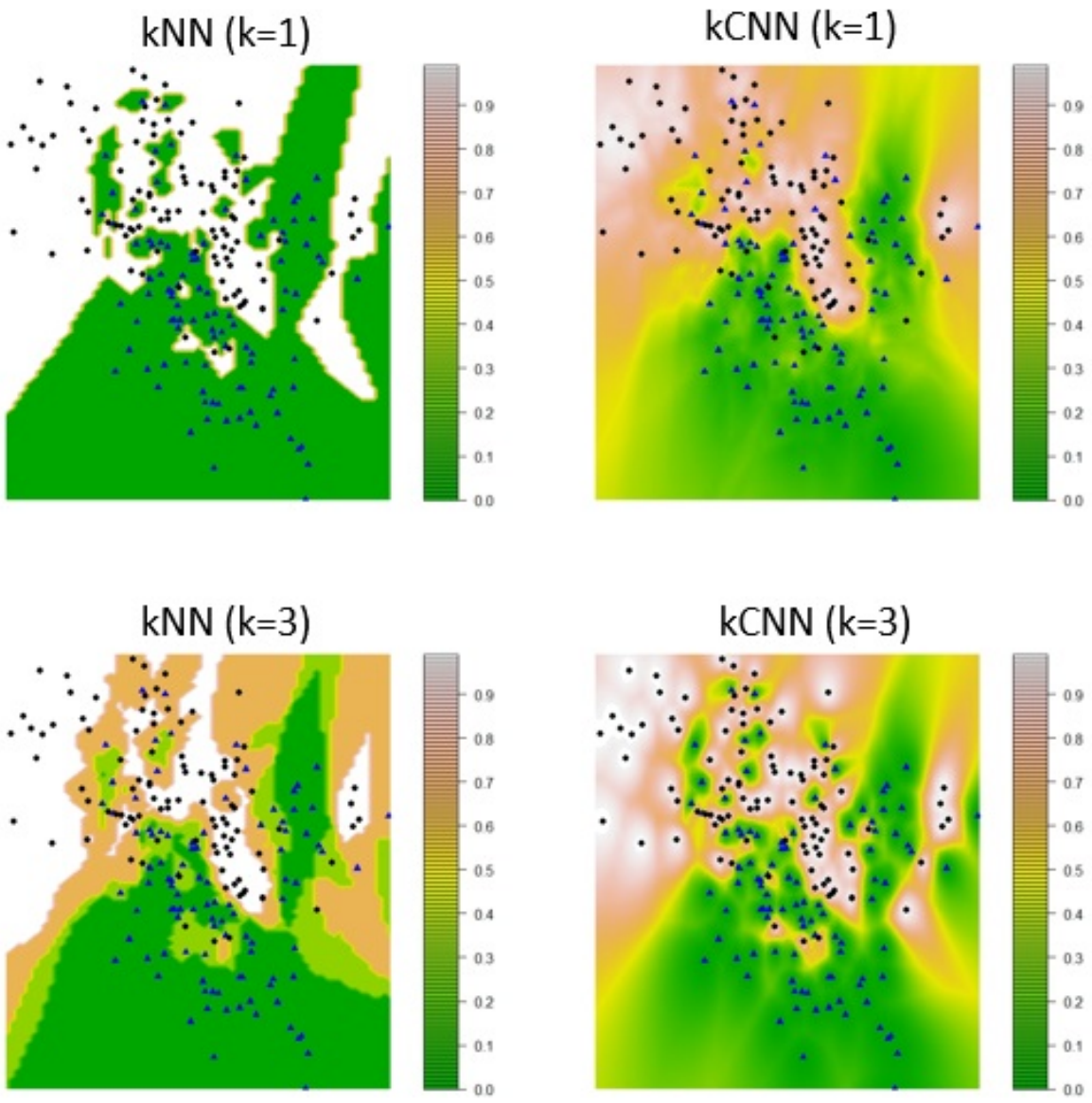


Figure 3.11: Contour plots of posterior probabilities of kNN and $kCNN$ for $k = 1$ and $k = 3$

where I_p is the p dimensional identity matrix. Note that s is the Euclidean distance between the two means. Therefore, s controls the degree of overlap between the distributions of the two classes.

In order to obtain less variable results, we used 10 independent replicates for each parameter setting. The final outputs were obtained by averaging the results. We used 100 training and 1000 test instances and the equal prior setting for the classes. Like Wu et al. (2004), we evaluated the posterior estimates based on mean squared error (MSE). The MSE for the test data is obtained as

$$MSE = \frac{1}{1000}\frac{1}{2}\sum_{j=1}^{1000}\sum_{i=1}^{2}\left(\hat{p}(c_i|\mathbf{x_j}) - p(c_i|\mathbf{x_j})\right)^2$$

where $\mathbf{x}_j$ represents the j^{th} test instance.

Table 3.3 shows the MSE for each method as a function of s and k when $p = 2$. The $kCNN$ method beat kNN for small values of s. Small values of s mean that the mean vectors are close to each other, and hence there is more overlap between the two conditional densities. The difference in performance between the two methods decreased as s or k increased.

Next, we considered the effect of feature dimension p on each method. Table 3.4 shows the MSE for each method as a function of p and k when $s = 0.1$. Throughout the range of p, $kCNN$ outperformed kNN. As p increased the MSE for $kCNN$ was less affected by the choice of k.

3.5 Discussion

For the 15 benchmark data sets, $EkCNN$ had the lowest and $kCNN$ the second lowest error rate (or, equivalently, accuracy). In terms of statistical significance, $EkCNN$ per-

Table 3.3: MSE as a function of k and s for kNN and $kCNN$. 100 training instances and $p = 2$ were used. The results were the averages of 10 replicates.

	k=1		k=5		k=10		k=20	
s	kNN	$kCNN$	kNN	$kCNN$	kNN	$kCNN$	kNN	$kCNN$
0.1	0.504	**0.074**	0.115	**0.017**	0.065	**0.011**	0.038	**0.006**
0.5	0.483	**0.080**	0.094	**0.022**	0.046	**0.019**	0.025	**0.016**
1	0.449	**0.113**	0.082	**0.054**	**0.042**	0.053	**0.028**	0.058
1.5	0.308	**0.104**	**0.056**	0.064	**0.024**	0.073	**0.016**	0.085
2	0.211	**0.096**	**0.045**	0.082	**0.024**	0.094	**0.016**	0.113

formed significantly better than kNN, $LMkNN$ and $kCNN$ on error rate. For the same data sets, $kCNN$ performed significantly better than kNN and $LMkNN$.

The ensemble method $EkCNN$ performed better than $kCNN$. For each k, $kCNN$ uses a single posterior estimate for each class, whereas $EkCNN$ combines multiple posterior estimates. This more differentiated estimate for posteriors may be the reason for the greater classification accuracy.

We have shown that $kCNN$ is asymptotically Bayes optimal for $r = 1$. It is interesting that for the ensemble version of $kCNN$, $r = p$ is clearly better for large p. While surprising, this is not contradictory since the Bayes optimality only applies asymptotically and only for $kCNN$ and not for the ensemble version $EkCNN$.

While the tuning parameter r does not affect classification for $kCNN$, r does affect classification for $EkCNN$. For the empirical results presented in Table 3.2, we chose $r = p$ for all data sets. We also noted that in 14 of the 15 data sets $r = p$ leads to a lower or equal error rate as compared to $r = 1$. Rather than just tuning the parameter k, it would

Table 3.4: MSE as a function of k and p for kNN and $kCNN$. 100 training instances and $s = 0.1$ were used. The results were the averages of 10 replicates.

	k=1		k=5		k=10		k=20	
p	kNN	$kCNN$	kNN	$kCNN$	kNN	$kCNN$	kNN	$kCNN$
2	0.502	**0.070**	0.122	**0.014**	0.054	**0.006**	0.022	**0.004**
5	0.499	**0.017**	0.100	**0.003**	0.048	**0.002**	0.021	**0.002**
10	0.503	**0.007**	0.112	**0.003**	0.058	**0.002**	0.027	**0.002**
30	0.500	**0.002**	0.102	**0.002**	0.053	**0.001**	0.026	**0.001**
50	0.494	**0.002**	0.103	**0.001**	0.049	**0.001**	0.023	**0.001**

be possible to simultaneously tune k and r. While this may further improve the error rates of $EkCNN$, the improvement, if any, comes at additional computational cost and is not expected to be appreciably large. For example, for the *sonar* data set - the data set with the largest number of features - we have also demonstrated in Section 3.3.4 that no improvement was obtained when $r > p$.

The simulation study in Section 3.4 showed that the decision boundary obtained by $kCNN$ can be smoothed by increasing k. Although this aspect seems similar to that of kNN, the reasons for smoothed decision boundaries are different. As k increases, kNN considers more observations for classification and thus the classification is less affected by noise or outliers. By contrast, $kCNN$ always uses the same number of observations (the number of classes) to make a prediction regardless of k. The $kCNN$ approach ignores the first k-1 nearest neighbors from each class and this makes the decision boundary less local.

Since $EkCNN$ is a combination of multiple $kCNN$ classifiers, its decision boundary is also a combined result of multiple decision boundaries from $kCNN$. Because the decision

boundary obtained by $kCNN$ is smoothed as k increases, that obtained by $EkCNN$ is also smoothed. However, the smoothing occurs more gradually, since the decision boundary obtained at k is always combined with the k-1 less smooth decision boundaries. This implies that $EkCNN$ is more robust against underfitting than $kCNN$ that may occur at large k. The decision boundaries shown in Section 3.4.1 confirmed this.

An advantage of $kCNN$ over kNN, especially when k is low, is that $kCNN$ can estimate more fine-grained probability scores than kNN, even at low values of k. For kNN, a class probability for a new observation is estimated as the fraction of observations classified as that class. By contrast, $kCNN$ estimates the posteriors based on distances and thus gives more fine-grained probability scores. The probability contour plots in Section 3.4.2 confirmed this.

The simulation in Section 3.4.3 suggests that the greater the overlap among the posterior distribution of each class, the more likely that $kCNN$ beats kNN in terms of the MSE. In most applications class distributions overlap, which partially explains why in the experiment in Section 3.3.3 $kCNN$ performed better than kNN in many cases.

Chapter 4

Improving Automated Occupation coding

4.1 Introduction

Classifying a respondent's occupation is essential in official statistics and social science research. It enables the international comparison of the official statistics on occupation and work and is the starting point for numerous status scales or prestige measures. It is a "foundation of much, if not most research on social stratification" (Ganzeboom and Treiman, 2003) and social inequality. Because occupation is a risk factor in many diseases, classifying occupations is an important first step for epidemiological analyses, industrial hygiene, and other biomedical sciences.

There are quite a few different classification schemes but all have hundreds of occupation codes and the codes are always nested in hierarchies. For example, the International Standard Classification of Occupations 1988 (ISCO-88) (Elias, 1997) is a classification of

four nested levels characterized by four digits. The first digit distinguishes nine major groups and an undifferentiated tenth major group for the Armed Forces. There are 28 sub-major groups (2-digit combinations), 116 minor groups (3-digit combinations) and 390 unit groups (4-digit combinations). Table 4.1 gives coding for sub-major group 71, extraction and building trades workers.

To ascertain a survey respondent's occupation, typically an open-ended question is asked (Belloni et al., 2014). Alternative ways to find a respondent's occupation include the use of search trees in web surveys (Tijdens, 2014, 2015), but open-ended questions are most common. The main example in this paper is the bi-annual ALLBUS survey (ALLBUS, 2015) conducted by GESIS, a German social science institute. The ALLBUS survey uses open-ended questions to ask about occupation (Scholz and Wasmer, 2009). Using multiple choice questions to elicit 4-digit occupation codes is not sensible because there are too many codes, and more importantly, respondents often would not know how to classify themselves because occupation coding rules are complex (International Labour Office, 1990; Geis, 2011; Elias, 1997; Belloni et al., 2014).

Traditionally, assigning an occupation code to each answer text has been conducted manually by human coders. Manual coding is time-consuming and expensive, requiring professional knowledge. Occupation coding is also difficult: there are hundreds of pre-defined occupation codes and even more occupation titles. For example, the ISCO-88 classification contains 390 four-digit occupation codes. Another difficulty is that coding even by professional coders may be inconsistent. The coding quality of a record depends on the length of the occupation description as well as the difficulty of the words in the record (Conrad et al., 2016).

In an attempt to partially automate coding, researchers have implemented various rule-

71	Extraction and building trades workers		
	711	Miners, shotfirers, stone cutters and carvers	
		7111	Miners and quarry workers
		7112	Shotfirers and blasters
		7113	Stone splitters, cutters and carvers
	712	Building frame and related trades workers	
		7121	Builders
		7122	Bricklayers and stonemasons
		7123	Concrete placers, concrete finishers and related workers
		7124	Carpenters and joiners
		7129	Building frame and related trades workers not elsewhere classified
	713	Building finishers and related trades workers	
		7131	Roofers
		7132	Floor layers and tile setters
		7133	Plasterers
		7134	Insulation workers
		7135	Glaziers
		7136	Plumbers and pipe fitters
		7137	Building and related electricians
		7139	Building finishers and related trade workers not elsewhere classified
	714	Painters, building structure cleaners and related trades workers	
		7141	Painters and related workers
		7143	Building structure cleaners

Table 4.1: ISCO-88 Sub-Major Group 71:Extraction and building trades workers

based coding schemes. For example, if the text answer contained a word matching an entry in a pre-defined dictionary, then the corresponding code in the dictionary was assigned. More recently, statistical learning or machine learning approaches (Statistical learning and machine learning are synonymous for the purpose of this paper. For brevity we just use the phrase "statistical learning" for the remainder of the paper.) have been employed: a model is trained on manually coded training data and is then used to predict the most probable code for new data. This approach is favored, for example, by the Australian Bureau of Statistics (Clarke and Brooker, 2011). Autocoders based on statistical learning have also been developed in the U.S.A (Day, 2014) and in Germany (Bethmann et al., 2014).

Although the automated methods reduce costs for occupation coding, fully automated coding remains challenging. With partial automatic coding easy-to-code answers are coded automatically, and-hard-to-code answer are coded manually. A measure of confidence – a numerical score – is used to distinguish between easy-to-code and hard-to-code text answer (Scholtus et al., 2014). For example, the CASCOT system proposes to code manually when a score for the coding quality drops below a modifiable threshold (Jones and Elias, 2004).

In this chapter we consider three new techniques for improving automated coding: a) combining two statistical learning models for different levels of aggregation, b) combining a duplicate-based approach with a statistical learning one, and c) a modified nearest neighbor approach. We evaluate the proposed approaches with data from the 2006 German ALLBUS survey coded by GESIS based on ISCO-88 codes.

4.2 Automated occupation coding

This section gives an overview of how to evaluate the performance in automated occupation coding as well as two commonly used approaches: rule-based approaches and approaches

based on statistical learning. The new approaches we introduce in this chapter are mostly based in statistical learning.

4.2.1 Production rate and accuracy

When some answer texts are coded automatically and some manually, a score or a probability is needed to distinguish between hard-to-code and easy-to-code answers. All new records with scores above a threshold are coded automatically; all others are coded manually. The threshold is set according to the desired combination of accuracy and production rate. The production rate is the proportion of observations that can be coded automatically. For a given production rate, accuracy is the proportion of codes that are coded correctly. Note there is a tradeoff between accuracy and production rate. High accuracy can be achieved for a small number of easy-to-code records. However, as the production rate increases and more difficult answers are included, accuracy tends to decrease. The tradeoff relationship was illustrated in (Chen et al., 1993).

4.2.2 Preprocessing

Before automated coding begins, text is often preprocessed. There is no standardized way of preprocessing but there are a range of options such as lower or upper casing all letters, removing duplicate blank spaces, automatically correcting spelling errors, removing very common words (so-called stopwords), and, less common in occupation coding but common in text mining, reducing words to their grammatical root (stemming). Preprocessing is an attempt to reduce the noise in the data.

4.2.3 Rule-based occupation coding

If the text answer meets a prespecified logical condition (e.g., presence of a certain word) a specific code is assigned. Such "if-then" statements are called rules. Rules are written by experts or can be based on previous data analysis. Rules can be combined using boolean logic. Any one rule based coding scheme consists of hundreds of rules leading to large dictionaries or look-up tables. Schierholz (2014) reports that this approach rarely codes more than 50% of records accurately. A variation on rule-based methods is to assign a score in favor of a category. If a text answer matches a rule, evidence can accumulate for multiple codes. In the end the text answer is classified into the occupation code with the highest score. One of the earliest references to rule-based coding is O'Reagan (1972).

Rule based systems are implemented in many institutions: the Washington State Department of Health (Ossiander and Milham, 2006), the 1970 U.S. Population and Housing Census (Knaus, 1987), the 1991 census data for Croatia and Bosnia-Herzegovina (Kalpic, 1994), the AIOCS system at the U.S. Census Bureau (Appel and Hellerman, 1983; Chen et al., 1993). Statistics Canada further developed the AIOCS system and created the G-Code (old name ACTR) software (Wenzowski, 1988; Tourigny and Moloney, 1995), which was also used for Italian census data (Ferrillo et al., 2008). The University of Warwick has a popular tool for automatic categorization called CASCOT (Jones and Elias, 2004), which also has been adapted to the Dutch language (Belloni et al., 2014).

4.2.4 Occupation coding based on statistical learning

Statistical models learn from already classified training data. Such methods can be used not only for occupation coding but also for general classification problems. Once the model has been trained, other observations can be classified automatically.

To build a model, text is first converted to numerical data. The standard text mining approach is to create a variable for each word that occurs in any of the answer texts. These so-called unigram variables or 1-grams either record the frequency of the word occurring in an answer text or simply the presence or absence of the word from the given answer text (Weiss et al., 2010; Joachims, 1998). There are many different variations of this text mining approach, adding variables for the presence or absence of multi-word sequences (ngram variables), removing highly used words (stopwords) because they are probably not useful, and stemming words to their grammatical root. The large number of variables are modeled with black-box statistical learning algorithms, such as support vector machines (SVM) (Vapnik, 2000). The model may incorporate additional variables if available.

Different learning algorithms have been used for occupation coding. The Australian Bureau of Statistics (ABS) employed fully automatic categorization using support vector machines to code data from the 2006 Australian Census (Clarke and Brooker, 2011). The ABS uses the Australian and New Zealand Standard Classification of Occupation (ANZSCO) scheme. To our knowledge this system is still in use by the ABS.

The American Community Survey (ACS) uses a variation on text mining (Thompson et al., 2012). Variables created from the text include one-word and two-word sequences (called "wordbits") as well as the full text. To limit the number of variables for analysis, a rareness threshold of 30 is used (i.e. the text has to occur at least 30 times before it is used as a variable). To further limit the number of variables for analysis, the corresponding text has to be "associated with a single industry/occupation code at least 50% of the time". The remaining variables, as well as variables like age and gender, are fed into a logistic regression. The code with the highest probability obtained by the logistic regression is assigned to a new record.

Some authors have investigated a nearest neighbor strategy, which assigns the code of

the answer in the training data that most closely resembles the answer in question. Different similarity metrics have been employed to measure nearness or resemblance between two answers. The PACE system employed the k nearest neighbor method with weighted feature metrics and reported accuracy 0.86 at production rate 0.57 for the U.S. Census Bureau data (Creecy et al., 1992). Jung et al. (2008) used cosine similarity but found this did not work well, possibly because they were working in Korean, a language quite different from languages with roots in Latin. Russ et al. (2014) used the nearest neighbor approach with a Jaccard similarity measure for classifying text answers into the Standard Occupational Classification (SOC) scheme. Coding by the nearest neighbour approach was considered correct if it agreed with one or both of the codes provided by the two human coders. The accuracy, i.e. the proportion of correctly classified observations, for fully automated coding was 0.51 at the 6-digit level and 0.64 at the 3-digit level.

The ALWA survey at the German Institute for Employment Research (IAB) used the 5-digit German national classification KldB 2010 (Schierholz, 2014). The approach presented in Schierholz (2014) used the full pre-processed verbatim answer text rather than the text mining approach using ngram variables. Preprocessing included converting special German characters into regular ones, stripping leading and trailing spaces. Using verbatim answers (rather than ngrams) drastically reduced the number of variables for learning. Schierholz (2014) then experimented with various methods including Naive Bayes and a gradient boosting model (Friedman, 2001). The experiment concluded that boosting and the Bayesian approaches performed similarly when high accuracy was desired.

4.3 Three methods for automated occupation coding

We first explain the duplicate method, a simple automated coding approach based on duplicate training observations. Next, we propose three new methods for automated occupation coding. The first of these methods, combining statistical learning models at different levels of aggregation, is later also incorporated with the second method resulting in two versions of the second method. For statistical learning models, any method that outputs probabilities can be used. In Section 4.4, we chose Support Vector Machines (Vapnik, 2000) for our application.

For each method the predicted occupation code is the code that has the highest score.

4.3.1 The duplicate method with the ngram based definition of duplicates

An exact-string duplicate refers to two strings that are identical. Simple string preprocessing might improve performance and leads to what we call a preprocessed-string duplicate. Preprocessing the string might consist, for example, of lower-casing all letters and removing leading and trailing blanks. For example "Apotheker" (pharmacist), "apotheker" and " apotheker" would be considered duplicates after preprocessing.

We introduce a different definition of duplicates based on ngram variables: An ngram-duplicate refers to a training observation with a text answer that has the same ngram representation (i.e. the same values for the variables created from the text). This is slightly different than an observation with the identical text answer. For example, the answer "Verwaltungsangestellte im Krankenhaus" (administrator in the hospital) and "Verwaltungsangestellte in einem Krankenhaus" (administrator in a hospital) are not identical

texts. However, since “in”, “im” and “einem” are stopwords and stopwords are removed, these two strings contain the same unigrams (“Verwaltungsangestellte”, “Krankenhaus”).

Suppose that there exist some duplicates of a new input record $\mathbf{x}$. Let $m_i(\mathbf{x})$ be the number of training duplicates having code c_i $(i = 1, 2, ..., L)$. We estimate the probability $p_d(c_i|\mathbf{x})$ based on the relative frequency of the training duplicates having code c_i:

$$\hat{p}_d(c_i|\mathbf{x}) = \begin{cases} \dfrac{m_i(\mathbf{x})}{M(\mathbf{x})} & \text{if } M(\mathbf{x}) > 0 \\ \dfrac{1}{L} & \text{otherwise} \end{cases}$$

where $M(\mathbf{x}) = \sum_{i=1}^{L} m_i(\mathbf{x})$ is the number of duplicates of $\mathbf{x}$ found in the training data. If no duplicate is found, the method assigns equal probability to each class. The code with the highest probability is chosen as the predicted code. The duplicate method leads to high accuracy for duplicates but not to 100% accuracy because coders try to resolve ambiguous situations with additional undocumented information or because of human error.

4.3.2 Combining models from different levels of aggregation

As seen in Table 4.1, occupation codes have a hierarchical structure. The ISCO-88 occupation codes consist of 4-digit numbers. For example, the code 7131 (roofers) is part of the minor group 713 (Building finishers and related trades workers). Three digit group codes aggregate related occupations. We propose to apply statistical learning separately to the 4-digit unit occupation codes and to the 3-digit groups and to combine probabilities as explained in the next paragraph. The motivation is as follows: Given the large number of occupation codes the number of observations at the 4-digit level can be sparse. The number of observations will be relatively less sparse at the 3-digit level. If classification from a 4-digit classifier results in a near tie of occupation codes with different minor groups

(different 3rd digit), the evidence from the 3-digit classifier may sway the classification to the correct 4-digit code.

Suppose that code c_i $(i = 1, ..., L)$ belongs to a 3-digit minor group m_j $(j = 1, ..., l)$ where L and l are the numbers of the 4-digit and 3-digit group codes respectively. Denote the probabilities from the statistical learning model for 3-digits and 4-digits as $\hat{p}_{3digit}(m_j|\mathbf{x})$ and $\hat{p}_{4digit}(c_i|\mathbf{x})$ for a record $\mathbf{x}$, respectively. We average the two probabilities:

$$\hat{p}_{3/4digit}(c_i|\mathbf{x}) = \frac{\hat{p}_{3digit}(m_j|\mathbf{x}) + \hat{p}_{4digit}(c_i|\mathbf{x})}{2} \quad . \tag{4.1}$$

This averaging approach will also break ties at the four digit level, unless the tied codes have the same three digit code. A recent review of hierarchical classification methods in general (Silla and Freitas, 2011), does not contain the proposed method. However, the proposed method may be viewed as a member of the local-classifier-per-level approaches as it fits a classifier for each of 3-digit and 4-digit levels independently.

4.3.3 A hybrid approach: Combining duplicate and statistical learning approaches

The proposed hybrid approach combines the approach based on duplicates in the training data with a statistical learning approach.

Let $\hat{p}_s(c_i|x)$ be the estimated probability obtained by a statistical learning approach. For the hybrid approach we define a combined score $\theta(c_i|\mathbf{x})$ as

$$\theta(c_i|\mathbf{x}) = \frac{M(\mathbf{x})}{M(\mathbf{x})+1} \cdot \hat{p}_d(c_i|\mathbf{x}) + \frac{1}{M(\mathbf{x})+1} \cdot \hat{p}_s(c_i|\mathbf{x}) \tag{4.2}$$

If there are no duplicates the score equals the probability from the statistical learning approach $\hat{p}_s(c_i|\mathbf{x})$. When there are duplicates, coding by the duplicate method is desirable

as it leads to high accuracy. Hence in the hybrid approach the statistical learning algorithm only influences the prediction when there is a tie among different duplicate codes. Equation (4.2) assigns the statistical learner a weight equivalent to that of a single duplicate and the single duplicate is downweighted by the probability $\hat{p}_s(c_i|\mathbf{x}) < 1$.

When the production rate is less than 100%, the easier-to-learn new records are categorized automatically. The statistical learning algorithms also influences this prioritization of new records. When two new records each have the same number of duplicates and if $\hat{p}_d(c_i|\mathbf{x})$ is the same in each case, the record with the larger $\hat{p}_s(c_i|\mathbf{x})$ is assigned a greater $\theta(c_i|\mathbf{x})$ and therefore is prioritized for lower production rates.

We call this approach "hybrid-4digit" when $p_s(c_i|\mathbf{x})$ in equation (4.2) is estimated using the statistical learning model for 4-digit occupation codes, $\hat{p}_{4digit}(c_i|\mathbf{x})$. Section 4.3.2 defined $\hat{p}_{3/4digit}(c_i|x)$ in equation (4.1) which combined two statistical learning models from different levels of aggregation. This idea can also be applied here. We call this approach "hybrid-3/4digit" when $p_s(c_i|\mathbf{x})$ in equation (4.2) is estimated using $\hat{p}_{3/4digit}(c_i|x)$.

4.3.4 A Modified Nearest Neighbor Approach

The nearest neighbour approach (NN) (Fix and Hodges, 1951) is another method employed in the occupation coding. NN classification finds a new record's nearest neighbor in the training data and assigns the occupation code of that nearest neighbor to the new record also. There can be multiple nearest neighbors (Yu, 2003). NN can be viewed as a generalization of the duplicate approach: Duplicates are nearest neighbors with a distance of zero. To define "near", a measure of distance, or, equivalently, a measure of similarity is needed. For text classification cosine similarity is widely used (Knaus, 1987; Iezzi et al., 2014; Maitra and Ramler, 2010). Cosine similarity between two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined

as

$$cosine(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|} = \frac{\sum u_i v_i}{\sqrt{\sum u_i^2}\sqrt{\sum v_i^2}} \quad . \tag{4.3}$$

where $\mathbf{u}$ and $\mathbf{v}$ are vector representations of presence or absence of ngrams in the text. Similarity ranges from 0 to 1 depending on the degree of the similarity between two records. Similarity is 0 if two records have no common words and 1 if the two records are identical (in the sense of having the same ngram representation). When duplicates exist, the NN method predicts the code of records with similarity 1, which is equivalent to the duplicate method.

As before, we may want to only code easy-to-code text answers and leave difficult ones for manual coding. Hence, we propose to use a score that assigns a higher value to NN predictions that are believed to be more accurate. Given a new text input $\mathbf{x}$, denote $K(\mathbf{x})$ the number of nearest neighbors in the training data and $s(\mathbf{x})$ the similarity of the nearest neighbors. (Often $K(\mathbf{x}) > 1$ when multiple observations are the nearest neighbors.) Suppose that $k_i(\mathbf{x})$ out of the $K(\mathbf{x})$ records have code c_i $(i = 1, ..., L)$. As in the duplicate method, we estimate the probability for code c_i in the NN approach by $\hat{p}_{nn}(c_i|\mathbf{x}) = k_i(\mathbf{x})/K(\mathbf{x})$. We define the score for the text answer as

$$\gamma(c_i|\mathbf{x}) = \hat{p}_{nn}(c_i|\mathbf{x}) \; s(\mathbf{x}) \; \left(\frac{K(\mathbf{x})}{K(\mathbf{x}) + 0.1}\right). \tag{4.4}$$

The predicted code depends only on $\hat{p}_{nn}(c_i|\mathbf{x})$ because $K(\mathbf{x})$ and $s(\mathbf{x})$ are constant for any given answer text. The role of $s(\mathbf{x})$ and $K(\mathbf{x})/(K(\mathbf{x}) + 0.1)$ is to order observations such that easier-to-classify-answers have a higher score.

The multiplier $s(\mathbf{x})$ makes sense: greater similarity of a new text and its nearest neighbor leads to more accurate classifications. The last term in equation (4.4) can be motivated as follows: All else being equal, classification based on a larger number of nearest neighbors

will likely be more accurate than that based on fewer nearest neighbors. The multiplier $K(\mathbf{x})/(K(\mathbf{x})+0.1)$ equals 0.91 when $K(\mathbf{x})=1$ and converges to 1 as $K(\mathbf{x})$ increases. Reflecting lesser importance, this multiplier can at most reduce the score by about 10% whereas both $\hat{p}_{nn}(c_i|\mathbf{x})$ and s can drive the score to zero. We will show below that this works empirically, however, we readily admit this is not the only multiplier that achieves this goal and the choice of 0.1 is arbitrary. Using a larger constant extends the range of the multiplier component and hence makes the score more sensitive to $K(\mathbf{x})$. (This is not desirable as the other two multipliers are more important.)

For example, the text answer of a new record was "Heizungs und Lüftungsbauer, Drucker". The text consisted of three (stemmed) unigram variables: "heizung" (heating), "lüftungsbau" (ventilation construction) and "druck" (printer). No duplicates existed but 4 records in the training data contained one of the three words. Table 4.2 shows that 3 out of the 4 training records had the answer "Drucker" ("druck" in the stemmed ngram representation) with code 8251 and the other had "Lüftungsbauer" ("lüftungsbau" in the stemmed ngram representation) with code 7136. Based on equation (4.3) the similarity between the test answer and any of the training records in Table 4.2 was $\frac{1}{\sqrt{3}\sqrt{1}}=0.5774$. So the multiplier in equation (4.4) is $K(\mathbf{x})/(K(\mathbf{x})+0.1)=4/4.1=0.9756$. However, $\hat{p}_{nn}(c_i=8251|\mathbf{x})=3/4$ and $\hat{p}_{nn}(c_i=7136|\mathbf{x})=1/4$. The difference of the γ scores of the two codes was due to the different probability estimates. In this example, the test answer was assigned code 8251 because it had the largest score ($\gamma=0.4225$).

4.4 Occupation coding for the ALLBUS survey

We first describe the ALLBUS data (Section 4.4.1) and then show the importance of our definition of duplicates (Section 4.4.2). Next, we compare the proposed automatic coding

Record	(Nonzero) ngram variables			Occ. code	$\hat{p}_{nn}(c_i\|\mathbf{x})$	$s(\mathbf{x})$	$\frac{K(\mathbf{x})}{K(\mathbf{x})+0.1}$	$\gamma(c_i\|\mathbf{x})$
	heizung	lüftungsbau	druck					
Training 1	0	0	1					
Training 2	0	0	1	8251	0.75	0.5774	0.9756	0.4225
Training 3	0	0	1					
Training 4	0	1	0	7136	0.25	0.5774	0.9756	0.1408
Test answer	1	1	1	$\hat{c}_i = 8251$				

Table 4.2: Illustration of calculating $\gamma(c_i|\mathbf{x})$. The unigram variables contain 1 if the word is present in the record and 0 otherwise.

methods on the ALLBUS data (Sections 4.4.3 and 4.4.4). We conclude with a simulation to explore the influence of duplicates and noise variables in Section 4.4.5.

4.4.1 Problem and Data

The German General Social Survey (ALLBUS) conducts repeated cross-sectional surveys of the adult German population living in private households, with an oversampling of the residents of Eastern Germany. ALLBUS has been conducted every two years since 1980; initially covering West Germany and expanding to East Germany since German reunification in 1990 (ALLBUS, 2015; Koch and Wasmer, 2004). The main topics concern attitudes, behavior and social structure.

The targeted net sample size is usually $3,500$. Since 1994, the samples are drawn in two stages. In the first stage, about 160 communities (primary sampling units) are selected. In the second stage, addresses of individuals are randomly selected from the lists of residents for every community. Every two years a fresh probability sample is drawn from the German register. ALLBUS surveys are conducted face-to-face.

ALLBUS interviewers asked about occupation multiple times: current occupation of respondent, last occupation of respondent (if not employed), occupation of spouse (if married), occupation of partner (if not married but with partner), occupation of father, occupation of mother. In the ALLBUS survey the interviewer asks the following questions which are recommended by official statistics in Germany (Statistisches Bundesamt, 2010): "What work do you do in your main job? Please describe your work precisely. Does this job, this work have a special name?" (Scholz and Wasmer, 2009). Interviewers were free to combine the answers, and were not asked to write one answer after another. The occupation questions for partners/spouses/parents are analogous with the same format. The answers were pooled to form a single dataset. Prior to the open-ended questions for all occupations, respondents were also asked: "Please classify your occupational status according to this list." The list contains 32 occupation statuses in 12 categories. We refer to this below as (self-recorded) occupation status.

The ISCO-88 coding of the text answers was done by GESIS in a two-step procedure. First, automatic coding was attempted with the in-house software *textpack* (Geis and Hoffmeyer-Zlotnik, 2000; Züll, 2014). Such automatically coded answers were verified by a professional coder afterwards. All remaining responses were coded in a second step manually according to an extensive coding manual (Geis, 2011). The in-house software used a dictionary with about $4,500$ predefined combinations of ISCO codes. Because the dictionary mostly contains duplicates from previous surveys, *textpack* implements the duplicate approach with additional hand-crafted rules (however, the coder may also override some codes in light of occupational status, education or other information).

For each word or phrase listed in the dictionary, textpack searches for exact matches in the data and outputs the associated code. Such rules were applied one at a time (and the rule order may affect the result). If a rule was matched exactly, a response was coded. If

none of the rules applied, it was manually coded by professional coders. Typically, *textpack* coded about 50% of the responses. GESIS used self-reported occupation status only if text was unclear or ambiguous. In the 2006 survey, the $9,137$ observations were coded into 399 distinct unit occupation codes and 140 minor group codes.

To apply the proposed methods we encoded text answers into unigram variables (Schonlau and Guenther, 2016). All such variables were indicator variables specifying the presence or absence of the corresponding word. We applied stemming using a German Porter stemmer (Snowball, 2015) and removed German "stopwords" as well as punctuation marks. The removal of stopwords and stemming reduced the number of ngram variables. As is standard practice, we also created a variable that counted the number of words contained in the answer. All in all, $4,232$ indicator variables were created in addition to the number-of-words variable. In addition to the text response the survey also contains self-reported occupation status which was also included among the independent variables.

For a statistical learning approach we use support vector machines (SVM) (Vapnik, 2000) with a linear kernel, which has been shown to work well in text categorization (Joachims, 1998). The linear kernel requires only a single tuning parameter, C, that controls the trade off between the training error and model complexity. In this data set the choice of C had little influence on prediction accuracy and we used $C = 1$ throughout the study. As is common, the SVM scores were converted into probabilities using Platt's method (Platt, 2000) which performs a regularized logistic regression of class membership on the SVM score.

We evaluate the approaches using 10-fold cross validation (CV). This means, we randomly divide the data into 10 equal sized parts. We use the first 9 parts to train the model, and the last part to test the model. Accuracy is only evaluated on the test data. In turn, we use each of the 10 parts as test data and average the results. As a consequence, the size

of the training data is therefore 90% of the data or 8,223 observations. For the purpose of evaluating prediction accuracy we assume that the original codes assigned by GESIS and the professional coders are correct.

The analysis was carried out in *R* (R Core Team, 2014), and package *e*1071 (Meyer et al., 2014) is used for the construction of the SVM models.

Most open-ended answers were short. 66.5% of the answers consisted of a single word. The median length was 1 word; the average length 1.8 words and the maximum length 17 words. About 60% of the data consisted of (ngram-based) duplicate observations. Among duplicate observations, the median number of duplicates was 3 with a higher average (6.8) due to some very frequent duplicates (maximum = 221 duplicates). The text with the most duplicates was "Landwirt" (farmer).

4.4.2 ngram vs. string-based definition of duplicates

The purpose of this section is to demonstrate that the ngram-based method of duplicate is preferable to the string-based ones. Here we explore how much the definition of duplicate mattered for the two best performing methods, NN-3 and hybrid-3/4digit, which are explained later. We compared the ngram-based method with original string (without any processing) and preprocessed string methods. Preprocessed strings refer to lower casing and stripping off leading and trailing spaces of the original strings. As described in Section 4.4.1, ngram variables were obtained after stemming and removing stopwords and punctuation marks.

The percentage of duplicates is 52.6% for the identical-string-duplicates, 56.7% for the preprocessed-string-duplicates, and 60.0% for the ngram-duplicates. However, the quality of the duplicates did not degrade: identical-string-duplicates (preprocessed-string-

duplicates, ngram-duplicates) had identical occupation codes 91.9% (91.6%, 92.0%) of the time. The remaining 8% represent coders' attempt to recode otherwise unambiguous text in light of occupational status or education. For example, a pharmacist with lower occupational status might be reclassified as pharmaceutical assistant. Of course, misclassification errors are possible, too.

Figure 4.1 shows the tradeoff between accuracy and production rate for the three definitions of duplicates for hybrid-3/4digit (left panel) and NN-3 (right panel). The use of the ngram definition of duplicates improved accuracy in both methods for moderate and high production rates. With full automation, the accuracy increased from 0.54 (without preprocessed) to 0.65 for the hybrid-3/4digit method and from 0.47 (without preprocessed) to 0.65 for the NN-3 method. Preprocessed-string-duplicates fare somewhat better than unprocessed strings, but the success of the ngram-based definition clearly goes far beyond string preprocessing.

4.4.3 Accuracy of the nearest neighbor method

We first investigated the coding performance of the modified NN method. The score in equation (4.4) has three components. To demonstrate that all three components are helpful, we evaluate both the proposed overall score (NN-3) as well as reduced score missing one (NN-2) or two components (NN-1) with corresponding scores γ_1, γ_2 and γ_3:

$$\begin{aligned}
&\text{(NN-1)} \qquad \gamma_1 &&= \max_i \hat{p}_{nn}(c_i|\mathbf{x}) \\
&\text{(NN-2)} \qquad \gamma_2 &&= \max_i \hat{p}_{nn}(c_i|\mathbf{x})\ s(\mathbf{x}) \\
&\text{(NN-3)} \qquad \gamma_3 &&= \max_i \hat{p}_{nn}(c_i|\mathbf{x})\ s(\mathbf{x})\ \left(\tfrac{K(\mathbf{x})}{K(\mathbf{x})+0.1}\right)
\end{aligned}$$

Figure 4.2 shows the accuracies of each approach as a function of the production rate. (These were average accuracies from the 10-fold cross validation mentioned earlier). Answer

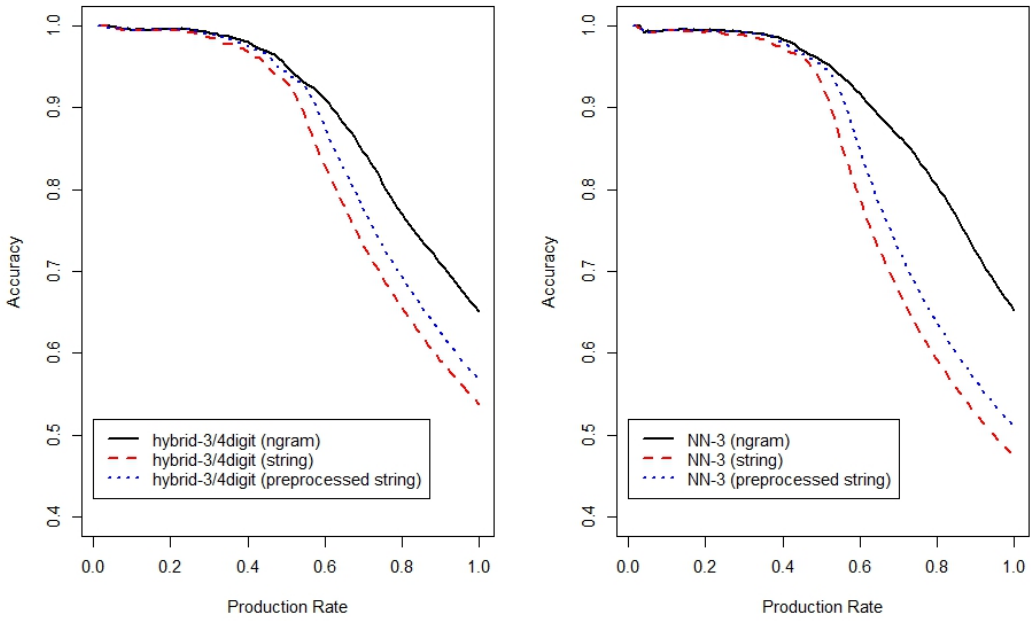


Figure 4.1: Accuracy for a given production rate for two approaches based on three different definitions of duplicates "ngram", "string" and "preprocessed string". The left panel shows the results of hybrid-3/4digit and the right panel shows those of NN-3. The "ngram" definition of duplicates is far superior.

texts with higher scores were coded first; a production rate of, say, 10% refers to coding 10% of the answer texts with the highest scores automatically. When the production rate equals 100%, the accuracy is the same for all the approaches because the second and third terms in equation (4.4) do not affect which code is assigned, but rather used to prioritize more similar observations and observations with multiple nearest neighbors by assigning them a higher score. Prioritizing affects the accuracy at production rates of less than 100% (because observations with the highest score are chosen first). The improvement from NN-1 to NN-2 showed that similarity s was helpful for finding easier-to-classify-answers. Likewise, the accuracy differences between NN-2 and NN-3 showed that the term $\frac{K(\mathbf{x})}{K(\mathbf{x})+0.1}$ improved the performance at low to medium production rates.

Having established that NN-3 is preferable to NN-1 and NN-2, we next compare NN-3 with all other approaches.

4.4.4 Comparison of methods

Here we compare the accuracy as a function of production rate for the proposed methods (hybrid-4digit, hybrid-3/4digit, and NN-3) as well as some default methods (duplicate method, svm-4digit, svm-3/4digit). The duplicate method refers to assigning the code of ngram duplicates (or a random code if no duplicates exist), svm-4digit refers to an SVM model based on 4-digit occupation codes. svm-3/4 digit refers to an SVM model based on averaged probability from separate models for 3-digit and 4-digit occupation codes as described in equation (4.1). For all methods, a production rate of x% refers to the x% of the data that have the highest score (or probability).

Figure 4.3 shows the accuracy as a function of the production rate for the different methods. For all methods, there were trade-offs between the accuracy and the production

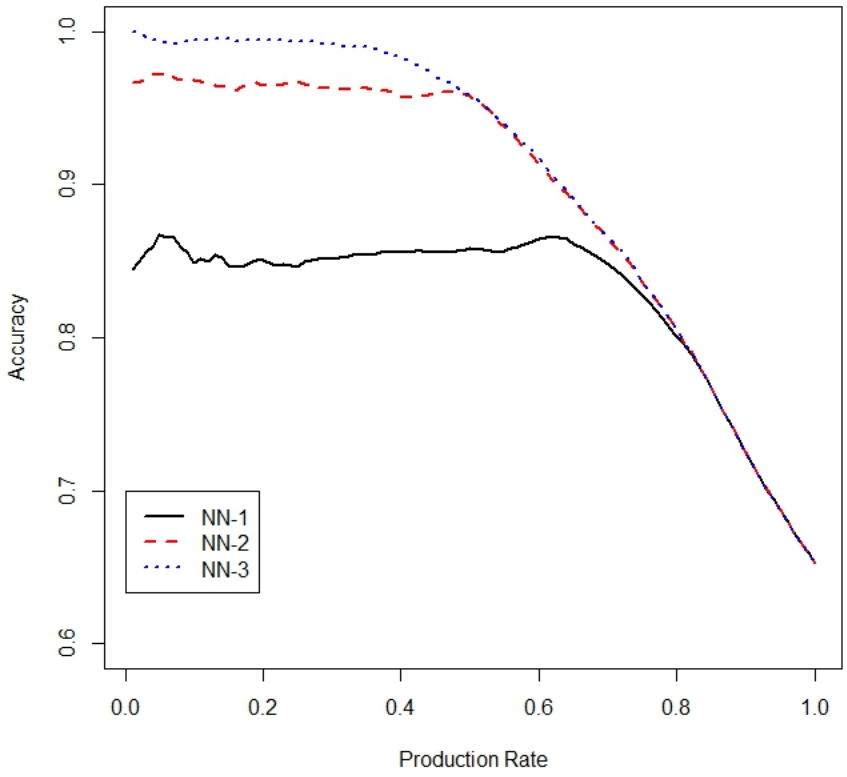


Figure 4.2: Accuracy of three variations on the nearest neighbor approach as a function of production rates. NN-1, NN-2 and NN-3 refer to scores using $\gamma_1 = \hat{p}_{nn}(c_i|\mathbf{x})$, $\gamma_2 = \hat{p}_{nn}(c_i|\mathbf{x})s$ and $\gamma_3 = \hat{p}_{nn}(c_i|\mathbf{x})s\left(\frac{K(\mathbf{x})}{K(\mathbf{x})+0.1}\right)$, respectively.

rate. The modified nearest neighbor method, NN-3, performs equal to or slightly better than the next best method, hybrid-3/4digit. NN-3, hybrid-4digit and hybrid-3/4digit uniformly beat the duplicate method and both SVM methods.

A production rate of 100% corresponds to classifying all answers automatically. At full automation, NN-3 and hybrid-3/4digit perform equally well. At full automation, svm-3/4digit has an accuracy of 59%, the duplicate method has an accuracy of 53%, and the hybrid-3/4digit method increases the accuracy to 65%.

Figure 4.3 also shows the duplicate accuracy stayed at around 95% up to a production rate of about 0.55. About 55% of the test data in any given cross validation were duplicates and thus duplicates were used for coding. However, when no duplicates exist in the training data the duplicate approach assigned equal probabilities to all codes, resulting in the random code assignment and accuracy near zero. The accuracy started decreasing at production rate around 0.55 from which no additional records of some CV test samples could be classified by the method. From production rate 0.60, all of the CV test datasets had no duplicates and the method performed poorly. NN-3, hybrid-4digit and hybrid-3/4digit beat the duplicate method even for production ranges where duplicates are available.

Combining the 4-digit unit and 3-digit minor code methods (svm-3/4digit) was uniformly superior to using the unit code method only (svm-4digit). For example, for fully automated coding, the accuracy for svm-3/4digit was 0.59 as compared to 0.52 for svm-4digit. The hybrid approaches performed very similarly up to about a production rate of 60%. After that, the hybrid-3/4digit performs a little better than hybrid-4digit. When duplicates were available for hybrid-3/4digit, the predicted codes mostly agreed (83%) with those predicted by the duplicate method.

The performances of hybrid-3/4digit and the NN-3 were similar for fully-automated

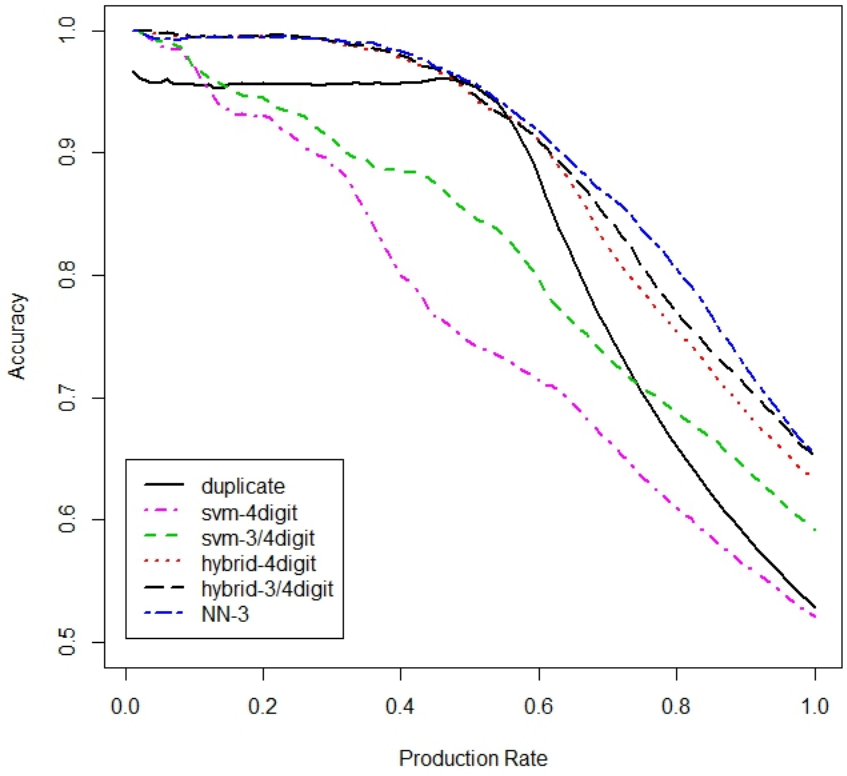


Figure 4.3: Comparison of different methods for occupation coding. Methods include statistical learning (svm-4digit), statistical learning from two models at different levels of aggregation (svm-3/4digit), and two hybrid methods combining duplicate-predictions with svm-4digit and svm-3/4digit, respectively.

coding as well as at low-medium production rates. NN-3 appeared to slightly outperform hybrid-3/4digit at medium-high production rates.

The curves in Figure 4.3 help us decide which texts should be classified automatically and which manually. For example, if the client decides that 80% accuracy is required, then Figure 4.3 suggests that 76% of the data can be classified automatically with the hybrid method and 81% with the NN-3 method. Relative to applying the duplicate based approach, this increases production from about 58% to 76% or 81%.

4.4.5 Simulation

The purpose of this section is to explore to what extent the methods are robust to possible idiosyncrasies of the data. We considered two possible concerns with our example data: 1) The data contain a large percentage (50%) of duplicates. 2) The text answers are unusually clean and contain fewer superfluous words than usual.

For the first case, in the context of occupation coding a large number of duplicates is very common. (Duplicates here refers to ngram duplicates). To simulate a data set with fewer duplicates, a random subset of duplicate records was removed so that in the reduced data only about 10% duplicates of the test records had duplicates. The reduced dataset contained 4,722 observations.

As expected, Figure 4.4 shows that the accuracy (for a given production rate) for all methods decreased for this much harder problem. The relative performance of the methods is very similar with one notable exception: Previously, both NN-3 and hybrid3/4-digit performed similarly. Now, NN-3 clearly outperforms the hybrid-3/4digit method. The NN-3 method remains superior to NN-1 and NN-2 analogous to Figure 4.2 (The analogous Figure is not shown).

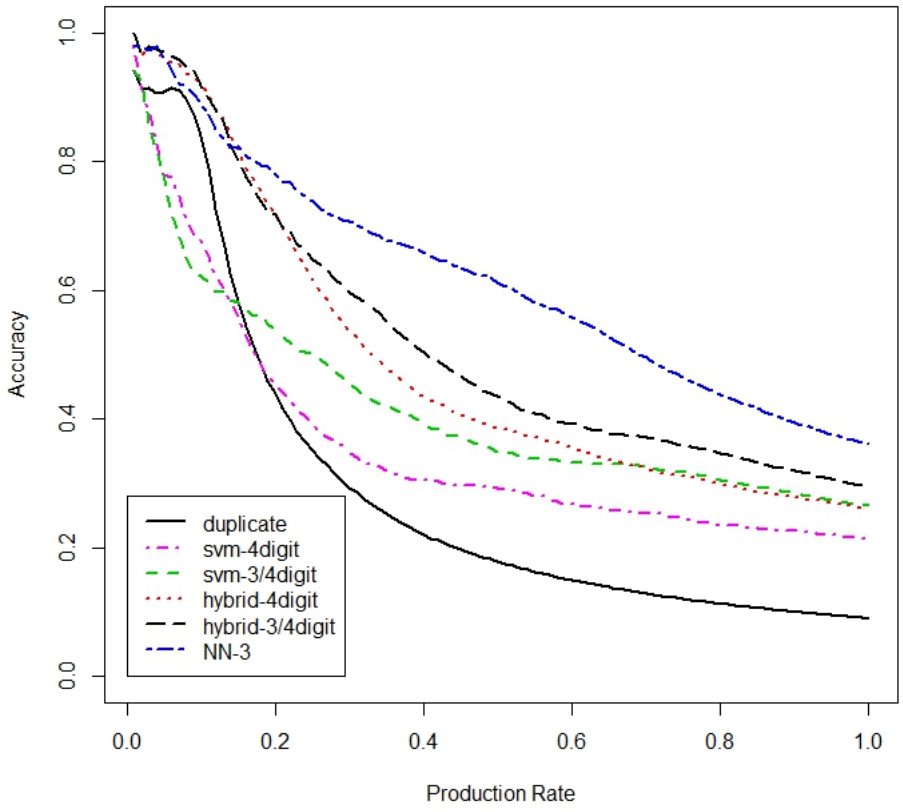


Figure 4.4: Comparison of the same methods as in Figure 4.3 on a reduced dataset containing only 10% duplicates.

For the second case, less clean text answers would have resulted into additional words that are not related to the occupation code. Such additional words translate into indicator variables (presence or absence of the word) in the data. There are typically many of such variables, each with a low probability. We added 100 independent "noise" indicator variables to the data. Each variable followed a Bernoulli distribution with probability of success of 0.01.

The results are shown in Figure 4.5. Adding the noise variables decreased the number of duplicates. Hence the accuracy of the duplicate method started decreasing at production rate around 0.2 instead of around 0.55. The results lead to roughly the same conclusions as we obtained from Figures 4.3 and 4.4. NN-3 and hybrid-3/4digit were comparable with NN-3 having a slight edge at lower production rates.

4.5 Discussion

We have investigated several novel approaches for automated occupation coding for any desired production rate. The two best performing methods, the modified nearest neighbor method (NN-3) and a hybrid method (hybrid-3/4digit) substantially improve the accuracy compared to both statistical learning (SVM in the example) by itself and the duplicate method at any production rate in the ALLBUS data. As the percentage of duplicates decreases, a simulation shows that NN-3 gains a relative advantage over the hybrid method.

Either accuracy or production rate can be set to a target rate which determines the second measure. For example, targeting 80% accuracy for the automated coding, the hybrid-3/4digit and NN-3 approaches could categorize 76% and 81% of the data automatically, while the numbers obtained by the SVM and duplicate methods individually were 60% and 66%, respectively. If production rate is fixed at 80%, the hybrid-3/4digit

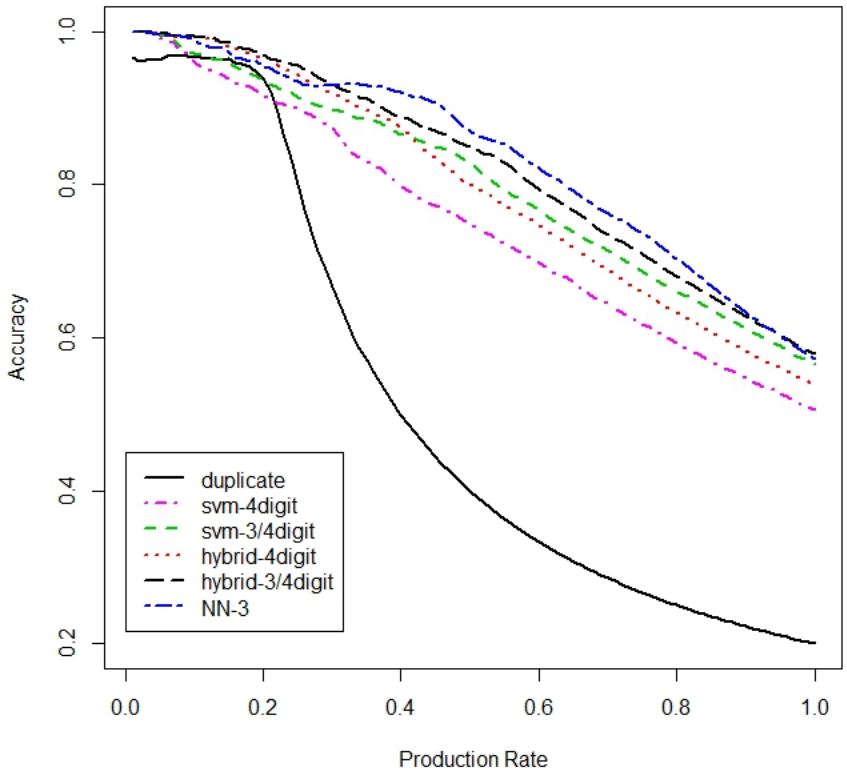


Figure 4.5: Comparison of the same methods as in Figure 4.3 with 100 noise variables added to the data.

and NN-3 could achieve accuracy of 77% and 81%, while the SVM and duplicate approaches reported accuracy of 69% and 66%. Note that accuracy for each category may differ from the overall accuracy. Categories that contain more hard-to-code answers than others achieve lower accuracies.

In addition, we have learned: 1) Even at low production rates when duplicates exist, NN-3 and hybrid achieve a higher accuracy than the duplicate method. 2) Using the duplicate method where duplicates exist and using statistical learning otherwise is not the best strategy (Figure 4.3 shows the proposed methods beat the duplicate method where duplicates exist.). We instead recommend the hybrid method that integrates the two approaches. 3) Combining aggregate and detailed learners improves accuracy for some learning algorithms. For example, where svm-4digit and svm-3/4digit disagree in the ALLBUS data, svm-3/4digit is 87% of the time correct.

Why do the NN-3 and hybrid methods beat SVM and the duplicate approach? Because a duplicate is also a nearest neighbor, both methods rely on nearest neighbors. Nearest neighbor algorithms are effective when prediction is highly local and little can be gained from observations farther away. This may explain why NN-3 and hybrid methods beat SVM, one of best statistical learning algorithms in existence. Both proposed methods beat the duplicate approach because a) they both can distinguish between easier-to-code and harder-to-code duplicates leading to higher accuracies at lower production rates, b) the hybrid- 3/4 method can break ties among duplicates and c) the duplicate approach performs poorly when no duplicates exist.

The NN-3 approach can be computationally expensive when the training dataset is very large. The hybrid method requires finding duplicates but finding duplicates is much less expensive because it does not require a sorting step.

We have combined the aggregate method with the hybrid method leading to better results. The modified nearest neighbor method could also be combined with the idea of aggregating different level scores. However, the resulting method showed almost the same performance as NN-3.

We now comment on the importance of some data analysis choices. First, duplicates were defined as having the same ngram representation rather than being identical strings. This increased the number of duplicates and substantially improved accuracy at moderate and high production levels. Second, self-reported occupation status (STIB) was used as a covariate for statistical learning. We found including STIB made little difference. Third, we supported German language stemming, but it turned out this had almost no effect. Because the text was written by interviewers (rather than respondents) our data were relatively clean with many one-word answers. Stemming is likely more important with messier data.

We next comment on possible limitations arising from idiosyncrasies of the ALLBUS data set. The proposed methods are not limited to ISCO-88 coding scheme. One of the methods relies on a hierarchical coding scheme, but all occupation codes are hierarchical. We have analysed 9137 observations. While this data set is probably larger than most data sets analysed in statistical journals, at national statistics agencies far larger data sets arise with sometimes millions of observations. The proposed methodology is not limited to a specific data size, but it is unclear whether the performance of the proposed methodology relative to the alternative algorithms would be equally impressive with millions of observations. We have pooled self-recorded occupations and occupations from partners, spouses and parents. We investigated whether this distorted result somehow. Specifically, we reduced the data set to one occupation question per respondent. We found this did not meaningfully affect the results.

For the hybrid method we used SVM as the statistical learning method of choice. While SVM is one of best performing methods available, other statistical learning methods could be chosen provided that they output a probability (or a score that can be transformed into a pseudo-probability) rather than just a classification. Naturally, better predictions from the statistical learning method will tend to improve the hybrid method also, particularly when there are no duplicates.

All proposed approaches rely on training data. For statistical learning, the size of the training data needs to be large relative to the number of occupation codes. In the ALLBUS data the size of the training data (implied by cross validation) was $8,226$. Relative to the 399 occupation codes, this is an average of 20.6 observations per code. More training data will tend to increase the number of duplicates.

Cross validation deals with unseen data but does not take into account time trends. To the extent that language use changes from year to year, any classifier would slowly degrade over time.

In summary, we proposed new approaches to automated occupation coding that lead to vastly improved coding accuracy at both high and low production rates in our example data. While not conclusive, this bodes well for other occupation data sets.

Chapter 5

Summary and Future Work

5.1 Summary of the Thesis

This thesis addressed three different topics in statistical learning: (a) general multi-label classification, (b) nonparametric multi-class classification and (c) automated occupation coding. The major contributions of this thesis are different novel classification approaches in order to achieve higher prediction performance compared with other methods.

In Chapter 2, we have presented $NLDD$ based on probabilistic binary classifiers. The proposed method chooses a training labelset with the minimum expected loss, where the expected loss is a function of two variables: the distances in feature and label spaces. The parameters are estimated by maximum likelihood. $NLDD$ relies on labelsets observed in the training data and is unable to predict previously unobserved labelsets. $NLDD$ outperformed other methods on the selected data sets where most test data sets contained 5% unobserved labelsets. While the method still outperforms the other methods with 33% of unobserved labelsets on the *bibtex* data, the method might not fare as well when the

percentage of unobserved labelsets is substantially greater.

In Chapter 3, we have proposed a new nonparametric classification method, $kCNN$, using the k^{th} nearest neighbor from each class. We have demonstrated that $kCNN$ is an approximation of the Bayes classifier. Moreover, we showed that $kCNN$ converges in probability to the Bayes classifier as the number of training instances increase. We also considered an ensemble type of $kCNN$ ($EkCNN$). Our experimental results on 15 benchmark data sets showed that $EkCNN$ was significantly superior to kNN, $kCNN$ and $LMkNN$ in terms of the error rate.

In Chapter 4, we have investigated several novel approaches for automated occupation coding for any desired production rate. The proposed approaches to automated occupation coding lead to vastly improved coding accuracy at both high and low production rates in our example data. While not conclusive, this bodes well for other occupation data sets.

5.2 Future Work

5.2.1 A weighted $EkCNN$ classifier

In this thesis we have shown that the $EkCNN$ nonparametric classifier is simple and flexible, and can outperform kNN. The $EkCNN$ classifier still has significant potential for further improvements and applications. Many of the directions for improving $EkNN$ may also be applied for improving the $kCNN$ method. One way is to give different weights for different k. In the context of kNN, a number of weighted kNN approaches have been introduced (Dudani, 1976; Keller et al., 1985; Gou et al., 2012). Many of the methods give large weights to closer neighbors and small weights to farther ones. Similar to this idea, we may consider a weighting method that gives large weights to $kCNN$ classifiers

with small k and small weights to $kCNN$ with large k. On the other hand, the results of $kCNN$ in Chapter 3 show that the performance of $kCNN$ is dependent on k. Based on the performances of $kCNN$ with different k, we may also consider a weighting method that gives large weights to more accurate $kCNN$.

5.2.2 Multi-label and other types of classification

The $NLDD$ classifier may still be improved in terms of its run time and prediction accuracy. The directions for future research include examining feature selection, label thresholding, and prototype labelset approaches. The $NLDD$ method may also be used in other types of practical classification problems, including hierarchical (possibly multi-label) classification where the class sets are organized into a class hierarchy. Different algorithms have been developed for multi-label classification, yet few of them have been extended to hierarchical classification contexts. I propose to extend our multi-label classifier to hierarchical classification problems. The idea is to transform a hierarchical problem into a multi-label problem by treating the ancestor (or intermediate) classes as additional labels.

References

ALLBUS (2015). http://www.gesis.org/allbus (accessed October 10, 2016).

Appel, M. V. and Hellerman, E. (1983). Census bureau experiments with automated industry and occupation coding. In *Proceedings of the American Statistical Association, Section on Survey Research Methods*, pages 32–40.

Bagui, S. C., Bagui, S., Pal, K., and Pal, N. R. (2003). Breast cancer detection using rank nearest neighbor classification rules. *Pattern Recognition*, 36(1):25–34. DOI: https://doi.org/10.1016/S0031-3203(02)00044-4.

Belloni, M., Brugiavini, A., Meschi, E., and Tijdens, K. (2014). *Measurement Error in Occupational Coding: an Analysis on SHARE Data*. Ca' Foscari University of Venice, Department of Economics, Working Paper 24. DOI: http://dx.doi.org/10.2139/ssrn.2539080.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522. DOI: https://doi.org/10.1109/34.993558.

Bethmann, A., Schierholz, M., Wenzig, K., and Zielonka, M. (2014). Automatic coding of occupations. In *Proceedings of Statistics Canada Symposium 2014*. http:

//www.statcan.gc.ca/sites/default/files/media/14291-eng.pdf (accessed October 10, 2016).

Bhatia, N. and Vandana (2010). Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8(2):302–305.

Bielza, C., Li, G., and Larrañaga, P. (2011). Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705 – 727. DOI: https://doi.org/10.1016/j.ijar.2011.01.007.

Blockeel, H., Raedt, L., and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, Madison, Wisconsin. Morgan Kaufmann.

Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., and Clare, A. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *Proceedings of the 10th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 18–29, Berlin, Germany. Springer. DOI: https://doi.org/10.1007/11871637_7.

Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771. DOI: https://doi.org/10.1016/j.patcog.2004.03.009.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140. DOI: https://doi.org/10.1007/BF00058655.

Chen, B.-C., Creecy, R. H., and Appel, M. V. (1993). Error control of automated industry and occupation coding. *Journal of Official Statistics*, 9(4):729–745. http://www.jos.nu/Articles/abstract.asp?article=94729 (accessed October 10, 2016).

Clarke, F. R. and Brooker, S. (2011). Use of machine learning for automated survey coding. In *Proceedings of the 58th ISI World Statistics Congress*, Dublin, Ireland.

Conrad, F. G., Couper, M. P., and Sakshaug, J. W. (2016). Classifying open-ended reports: Factors affecting the reliability of occupation codes. *Journal of Official Statistics*, 32(1):75–92. DOI: http://dx.doi.org/10.1515/JOS-2016-0003.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27. DOI: https://doi.org/10.1109/TIT.1967.1053964.

Creecy, R. H., Masand, B. M., Smith, S. J., and Waltz, D. L. (1992). Trading mips and memory for knowledge engineering. *Communications of the ACM*, 35(8):48–64. DOI: http://dx.doi.org/10.1145/135226.135228.

Day, J. (2014). Using an autocoder to code industry and occupation in the american community survey. Presentation for the Federal Economic Statistics Advisory Committee Meeting. http://www2.census.gov/adrm/fesac/2014-06-13_day.pdf (accessed October 10, 2016).

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

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327. DOI: https://doi.org/10.1109/TSMC.1976.5408784.

Elias, P. (1997). Occupational classification (ISCO-88): Concepts, methods, reliability, validity and cross-national comparability. *OECD Labour Market and Social Policy*

Occasional Papers 20, OECD Publishing. https://ideas.repec.org/p/oec/elsaaa/20-en.html (accessed October 10, 2016).

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14*, pages 681–687, Vancouver, Canada. MIT Press.

Ferrillo, A., Macchia, S., and Vicari, P. (2008). Different quality tests on the automatic coding procedure for the economic activities descriptions. In *Proceedings of the European Conference on Quality in Official Statistics - Q2008*. Rome, Italy.

Fix, E. and Hodges, J. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medivine, Randolph Field, Texas. Project 21-49-004, Rept. 4, Contract AF41(128)-31, February 1951.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*, volume 1. Springer, Berlin.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29:1189–1232. http://www.jstor.org/stable/2699986 (accessed October 10, 2016).

Fukunaga, K. and Hostetler, L. (1975). K-nearest-neighbor Bayes-risk estimation. *IEEE Transactions on Information Theory*, 21(3):285–293. DOI: https://doi.org/10.1109/TIT.1975.1055373.

Ganzeboom, H. B. and Treiman, D. J. (2003). Three internationally standardised measures for comparative research on occupational status. In Hoffmeyer-Zlotnik, J. H. P. and Wolf, C., editors, *Advances in Cross-National Comparison: A European Working Book*

for Demographic and Socio-Economic Variables, pages 159–193. DOI: http://dx.doi.org/10.1007/978-1-4419-9186-7_9.

Garcia, S., Derrac, J., Cano, J., and Herrera, F. (2012). Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):417–435. DOI: https://doi.org/10.1109/TPAMI.2011.142.

Geis, A. (2011). Handbuch für die Berufsvercodung. Technical report, GESIS, Mannheim, Germany. http://www.gesis.org/fileadmin/upload/dienstleistung/tools_standards/handbuch_der_berufscodierung_110304.pdf (accessed October 10, 2016).

Geis, A. J. and Hoffmeyer-Zlotnik, J. H. (2000). Stand der Berufsvercodung. *ZUMA Nachrichten*, 24:103–128.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Proceedings of the 8th Pacific-Asia conference on knowledge discovery and data mining*, pages 22–30, Sydney, Australia. Springer. DOI: https://doi.org/10.1007/978-3-540-24775-3_5.

Gonçalves, T. and Quaresma, P. (2003). A preliminary approach to the multilabel classification problem of Portuguese juridical documents. In *Proceedings of the 11th Portuguese Conference on Artificial Intelligence*, pages 435–444, Beja, Portugal. Springer. DOI: https://doi.org/10.1007/978-3-540-24580-3_50.

Gou, J., Du, L., Zhang, Y., and Xiong, T. (2012). A new distance-weighted k-nearest neighbor classifier. *Journal of Information and Computational Science*, 9(6):1429–1436.

Gou, J., Zhan, Y., Rao, Y., Shen, X., Wang, X., and He, W. (2014). Improved pseudo nearest neighbor classification. *Knowledge-Based Systems*, 70:361–375. DOI: https://doi.org/10.1016/j.knosys.2014.07.020.

Gowda, K. and Krishna, G. (1979). The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (corresp.). *IEEE Transactions on Information Theory*, 25(4):488–490. DOI: https://doi.org/10.1109/TIT.1979.1056066.

Hersh, W., Buckley, C., Leone, T. J., and Hickam, D. (1994). OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, Dublin, Ireland. Springer-Verlag. DOI: https://doi.org/10.1007/978-1-4471-2099-5_20.

Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802. DOI: https://doi.org/10.1093/biomet/75.4.800.

Iezzi, D. F., Lori, M., Lorenzini, F., Nicosia, M., and Stoppiello, S. (2014). An application of text mining technique for the census of nonprofit institutions. In Crescenzi, F. and Mignani, S., editors, *Statistical Methods and Applications from a Historical Perspective*, pages 143–152. Springer. DOI: http://dx.doi.org/10.1007/978-3-319-05552-7_13.

International Labour Office (1990). *International Standard Classification of Occupations, ISCO-88*. International Labour Office. http://www.ilo.org/public/libdoc/ilo/1990/90B09_411_engl.pdf (accessed October 10, 2016).

Ji, S., Tang, L., Yu, S., and Ye, J. (2008). Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, pages 381–389, Las Vegas, Nevada. ACM. DOI: https://doi.org/10.1145/1401890.1401939.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, volume 1398, pages 137–142, Chemnitz, Germany. DOI: http://dx.doi.org/10.1007/BFb0026683.

Jones, R. and Elias, P. (2004). CASCOT: Computer-assisted structured coding tool. Technical report, Institute for Employment Research. Coventry: University of Warwick. http://www2.warwick.ac.uk/fac/soc/ier/publications/software/cascot/ (accessed October 10, 2016).

Jung, Y., Yoo, J., Myaeng, S.-H., and Han, D.-C. (2008). A web-based automated system for industry and occupation coding. In Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., and Wang, X., editors, *Web Information Systems Engineering - WISE 2008*, volume 5175, pages 443–457. Springer. DOI: http://dx.doi.org/10.1007/978-3-540-85481-4_33.

Kalpic, D. (1994). Automated coding of census data. *Journal of Official Statistics*, 10(4):449–463.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD Discovery Challenge*, Antwerp, Belgium. Springer.

Keller, J. M., Gray, M. R., and Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(4):580–585. DOI: https://doi.org/10.1109/TSMC.1985.6313426.

Klimt, B. and Yang, Y. (2004). The enron corpus: a new dataset for email classification research. In *Proceedings of the 15th European Conference on Machine Learning*, pages 217–226, Pisa, Italy. Springer. DOI: https://doi.org/10.1007/978-3-540-30115-8_22.

Knaus, R. (1987). Methods and problems in coding natural language survey data. *Journal of Official Statistics*, 3(1):45–67.

Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *Proceedings of the 18th European Conference on Machine Learning*, pages 624–631, Warsaw, Poland. Springer. DOI: https://doi.org/10.1007/978-3-540-74958-5_61.

Koch, A. and Wasmer, M. (2004). Der allbus als instrument zur untersuchung sozialen wandels: Eine zwischenbilanz nach 20 jahren. In Schmitt-Beck, R., Wasmer, M., and Koch, A., editors, *Sozialer und Politischer Wandel in Deutschland*. VS Verlag für Sozialwissenschaften.

Li, C., Wang, B., Pavlu, V., and Aslam, J. A. (2016). Conditional Bernoulli mixtures for multi-label classification. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2482–2491, New York City, NY.

Li, T. and Ogihara, M. (2003). Detecting emotion in music. In *Proceedings of the 4th International Symposium on Music Information Retrieval*, pages 239–240, Baltimore, Maryland.

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22. DOI: https://doi.org/10.2307/2336267.

Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Liu, F., Zhang, X., Ye, Y., Zhao, Y., and Li, Y. (2015). MLRF: multi-label classification through random forest with label-set partition. In *International Conference on Intelligent Computing*, pages 407–418, Fuzhou, China. Springer. DOI: https://doi.org/10.1007/978-3-319-22053-6_44.

Loftsgaarden, D. O. and Quesenberry, C. P. (1965). A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051.

Madjarov, G., Gjorgjevikj, D., Dimitrovski, I., and Džeroski, S. (2016). The use of data-derived label hierarchies in multi-label classification. *Journal of Intelligent Information Systems*, 47(1):57–90. DOI: https://doi.org/10.1007/s10844-016-0405-8.

Madjarov, G., Kocev, D., Gjorgjevikj, D., and Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104. DOI: https://doi.org/10.1016/j.patcog.2012.03.004.

Maitra, R. and Ramler, I. P. (2010). A k-mean-directions algorithm for fast clustering of data on the sphere. *Journal of Computational and Graphical Statistics*, 19(2):377–396. DOI: http://dx.doi.org/10.1198/jcgs.2009.08155.

Maji, P. (2011). Fuzzy-rough supervised attribute clustering algorithm and classification of microarray data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):222–233. DOI: https://doi.org/10.1109/TSMCB.2010.2050684.

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637. DOI: https://doi.org/10.1109/TPAMI.2013.83.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2014). *e1071: misc functions of the department of statistics, TU Wien*. http://CRAN.R-project.org/package=e1071, R package version 1.6.7.

Mitani, Y. and Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27(10):1151–1159. DOI: https://doi.org/10.1016/j.patrec.2005.12.016.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

O'Reagan, R. T. (1972). Computer-assigned codes from verbal responses. *Communications of the ACM*, 15(6):455–459. http://dx.doi.org/10.1145/361405.361419.

Ossiander, E. M. and Milham, S. (2006). A computer system for coding occupation. *American Journal of Industrial Medicine*, 49(10):854–857. DOI: http://dx.doi.org/10.1002/ajim.20355.

Pan, Z., Wang, Y., and Ku, W. (2017). A new k-harmonic nearest neighbor classifier based on the multi-local means. *Expert Systems with Applications*, 67:115–125. DOI: https://doi.org/10.1016/j.eswa.2016.09.031.

Platt, J. (2000). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A., Bartlett, P., Schoelkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

R Core Team (2014). *R: a language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

Raymer, M. L., Doom, T. E., Kuhn, L. A., and Punch, W. F. (2003). Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm.

IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 33(5):802–813. DOI: https://doi.org/10.1109/TSMCB.2003.816922.

Read, J., Pfahringer, B., and Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 995–1000, Pisa, Italy. DOI: https://doi.org/10.1109/ICDM.2008.74.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359. DOI: https://doi.org/10.1007/s10994-011-5256-5.

Ripley, B. (2007). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Rivolli, A. (2016). *utiml: Utilities for Multi-Label Learning*. R package version 0.1.2.9001.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39. DOI: https://doi.org/10.1007/s10462-009-9124-7.

Russ, D., Ho, K.-Y., Johnson, C., and Friesen, M. (2014). Computer-based coding of occupation codes for epidemiological analyses. In *Proceedings of the 27th IEEE International Symposium on Computer-Based Medical Systems*, pages 347–350, New York, USA. DOI: http://dx.doi.org/10.1109/CBMS.2014.79.

Samsudin, N. A. and Bradley, A. P. (2010). Nearest neighbour group-based classification. *Pattern Recognition*, 43(10):3458–3467. DOI: https://doi.org/10.1016/j.patcog.2010.05.010.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated

predictions. *Machine Learning*, 37(3):297–336. DOI: https://doi.org/10.1023/A:1007614523901.

Schapire, R. E. and Singer, Y. (2000). BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2):135–168. DOI: https://doi.org/10.1023/A:1007649029923.

Schierholz, M. (2014). Automating survey coding for occupation. Master's thesis, Ludwig-Maximilians-Universität Munich. https://epub.ub.uni-muenchen.de/21444/index.html (accessed October 10, 2016).

Scholtus, S., van de Laar, R., and Willenborg, L. (2014). The memobust handbook on methodology for modern business statistics.

Scholz, E. and Wasmer, M. (2009). German general social survey 2006. English translation of the German "ALLBUS"- questionnaire. Technical report, GESIS, Mannheim, Germany. http://nbn-resolving.de/urn:nbn:de:0168-ssoar-207035 (accessed October 10, 2016).

Schonlau, M. and Guenther, N. (2016). Text mining using n-grams. *Social Science Research Network*. Doi: http://dx.doi.org/10.2139/ssrn.2759033.

Silla, C. N. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72. DOI: http://dx.doi.org/10.1007/s10618-010-0175-9.

Snowball (2015). German stemming algorithm. http://snowball.tartarus.org/algorithms/german/stemmer.html (accessed October 10, 2016).

Srivastava, A. and Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. In *Proceedings of the 2005 IEEE Aerospace Conference*, pages 3853–3862, Big Sky, Montana. IEEE. DOI: https://doi.org/10.1109/AERO.2005.1559692.

Statistisches Bundesamt (2010). Demographische standards. Technical report, Wiesbaden, Germany. https://www.destatis.de/DE/Methoden/StatistikWissenschaftBand17.html (accessed October 10, 2016).

Struyf, J., Džeroski, S., Blockeel, H., and Clare, A. (2005). Hierarchical multi-classification with predictive clustering trees in functional genomics. In *Progress in Artificial Intelligence: 12th Portuguese Conference on Artificial Intelligence*, pages 272–283, Covilha, Portugal. Springer. DOI: https://doi.org/10.1007/11595014_27.

Tai, F. and Lin, H.-T. (2012). Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542.

Thompson, M., Kornbau, M. E., and Vesely, J. (2012). Creating an automated industry and occupation coding process for the american community survey. Unpublished. http://ftp.census.gov/adrm/fesac/2014-06-13_thompson_kornbau_vesely.pdf (accessed October 10, 2016).

Tijdens, K. (2014). Dropout rates and response times of an occupation search tree in a web survey. *Journal of Official Statistics*, 30(1):23–43. DOI: http://dx.doi.org/10.2478/jos-2014-0002.

Tijdens, K. (2015). Self-identification of occupation in web surveys: Requirements for search trees and look-up tables. *Survey Methods: Insights from the Field (SMIF)*. DOI: http://dx.doi.org/10.13094/SMIF-2015-00008.

Tourigny, J. and Moloney, J. (1995). The 1991 Canadian census of population experience with automated coding. In United Nations Statistical Commission on Statistical Data Editing.

Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 325–330, Philadelphia, Pennsylvania.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: an overview. *International Journal of Data Warehousing and Mining*, 3:1–13.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of the ECML/PKDD Workshop on Mining Multidimensional Data*, pages 30–44, Antwerp, Belgium. Springer.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). Mining multi-label data. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer, Boston. DOI: https://doi.org/10.1007/978-0-387-09823-4_34.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2011). Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089. DOI: https://doi.org/10.1109/TKDE.2010.164.

Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: an ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning*, pages 406–417, Berlin, Germany. Springer. DOI: https://doi.org/10.1007/978-3-540-74958-5_38.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory*. 2nd edition. Springer.

Weiss, S. M., Indurkhya, N., Zhang, T., and Damerau, F. (2010). *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, Heidelberg.

Wenzowski, M. (1988). ACTR - a generalised automated coding system. *Survey Methodology*, 14(2):299–308.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83. DOI: https://doi.org/10.2307/3001968.

Wu, T.-F., Lin, C.-J., and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005.

Wu, X., Kumar, V., Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Philip, S., and Zhou, Z. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37. DOI: https://doi.org/10.1007/s10115-007-0114-2.

Yang, J., Zhang, L., Yang, J.-Y., and Zhang, D. (2011). From classifiers to discriminators: A nearest neighbor rule induced discriminant analysis. *Pattern Recognition*, 44(7):1387–1402. DOI: https://doi.org/10.1016/j.patcog.2011.01.009.

Yong, Z., Youwen, L., and Shixiong, X. (2009). An improved knn text classification algorithm based on clustering. *Journal of Computers*, 4(3):230–237.

Ypma, T. J. (1995). Historical development of the Newton-Raphson method. *SIAM Review*, 37(4):531–551. DOI: https://doi.org/10.1137/1037125.

Yu, C. (2003). *High-Dimensional Indexing: Transformational Approaches to High-Dimensional Range and Similarity Searches*, volume 2341. Springer. DOI: http://dx.doi.org/10.1007/3-540-45770-4.

Zeng, Y., Yang, Y., and Zhao, L. (2009). Pseudo nearest neighbor rule for pattern classification. *Expert Systems with Applications*, 36(2):3587–3595. DOI: https://doi.org/10.1016/j.eswa.2008.02.003.

Zhang, M.-L. and Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *Proceedings of the 1st IEEE International Conference on Granular Computing*, volume 2, pages 718–721, Beijing, China. DOI: https://doi.org/10.1109/GRC.2005.1547385.

Zhang, M.-L. and Zhou, Z.-H. (2007). ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038 – 2048. DOI: https://doi.org/10.1016/j.patcog.2006.12.019.

Zhang, M. L. and Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837. DOI: https://doi.org/10.1109/TKDE.2013.39.

Züll, C. (2014). Berufscodierung. Technical report, GESIS - Leibniz Institut für Sozialwissenschaften (SDM Survey Guidelines). Mannheim. DOI: http://dx.doi.org/10.15465/sdm-sg_019.

Zuo, W., Zhang, D., and Wang, K. (2008). On kernel difference-weighted k-nearest neighbor classification. *Pattern Analysis and Applications*, 11(3-4):247–257. DOI: https://doi.org/10.1007/s10044-007-0100-z.