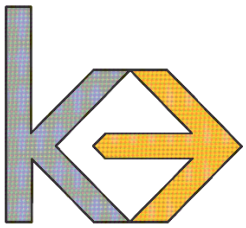
The effect of artificial subclass induction on classification performance

Studienarbeit

Nathan Valenti

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Technische Universität Darmstadt

Fachbereich Informatik

Fachgebiet Knowledge Engineering

Prof. Dr. Johannes Fürnkranz

Studienarbeit zu dem Thema:

The effect of artificial subclass induction on classification performance

Bearbeitet von: Nathan Valenti

Matr.-Nr.: 2459015

Studiengang: Wirtschaftsinformatik (M.Sc.)

Eingereicht am: 19.06.2019

Contents

List of Figures

List of Tables

1 Introduction

In a multi-class classification setting, it is common practice to artificially decompose the data into a series of binary classification problems. Common methods are pairwise classification (Hastie and Tibshirani, 1998), one-vs-all (Rifkin and Klautau, 2004), error-correcting output codes (Dietterich and Bakiri, 1994) and nested dichotomies (Frank and Kramer, 2004). The opposite, decomposing the given classes into artificial subclasses, is less researched. Hoffmann et al. (2001) show improved classification performance with increasing number of classes on simulated datasets, but report mixed results when experimenting with real datasets. Additionally Chen et al. (2018) show improved classification performance for CNNs when training on a dataset using a very fine label structure compared to a coarse one. Luo (2008) performs experiments using a multi-layer perceptron with a pre-labeled subclass structure. The author searches through the subclass structure for the best granularity. Increased performance is reported for the inclusion of well-separated subclasses but performance deterioration for entangled ones.

As the previous work deals with either simulated datasets or an existing subclass structure, the question remains open whether classification performance can be enhanced by the introduction of artificial subclasses into the dataset. This is, in theory, especially promising as the subclasses can be chosen to fulfill certain criteria, e.g. to be well-separated. To address this question three different subclass induction techniques are used. These are paired with three base classifiers (naive Bayes, C4.5 and random forest) and tested on 15 datasets. The classification is performed using the WEKA software (Hall et al., 2009).

The remainder of the paper is organized as follows: Section 2 describes the classifiers used in this paper. Section 3 describes the data enhancing techniques in more detail. Section 4 gives an overview of the data used and the experimental setup. The results are presented in section 5. The conclusion is presented in section 6.

2 Basic concepts

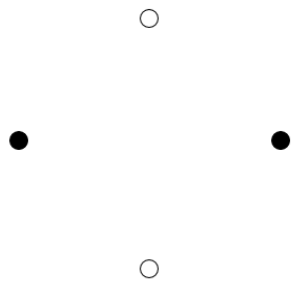
This section provides the theoretical foundations for the remainder of the paper. The VC dimension as a measurement for classifier expressiveness is described first, followed by a brief overview over the classifiers used in this paper.

2.1 VC dimension

The VC (Vapnik–Chervonenkis) dimension is a measure introduced by Vapnik and Chervonenkis (1971) for the expressive capability of a classifier. A high VC dimension indicates a high expressiveness. A classifier is said to *shatter* a set of vectors S with cardinality d if there exists a configuration of the classifier that can perfectly separate all possible 2^d 2-class problems. The VC dimension of a classifier is now the maximum number d for which a set of cardinality d exists which can be shattered by the classifier. The VC dimension can be infinite (Vapnik, 1999).

A straight line in a two-dimensional space, for instance, has a VC dimension of 3 as it can shatter any 3 points, but not 4 (see Figure 1). In general, a linear classifier in d dimensions has a VC dimension of $d + 1$. However, there are simple concepts that have infinite VC dimension such as the sine function (Goldberg and Jerrum, 1995).

Figure 1: Four points that cannot be separated by a line



A linear classifier cannot separate the empty from the full points and there is no other set of four points which can be shattered by a line.

2.2 Classifiers

2.2.1 Naive Bayes

Naive Bayes is a simple probabilistic classifier based on Bayes theorem. For each data sample x it predicts the class i with maximum posterior probability $P(i|x)$. Using Bayes theorem, this

can be reformulated into $P(i|x) \propto P(i)P(x|i)$ with $P(i)$ being the prior probability and $P(x|i)$ the likelihood (Hand and Yu, 2001). The likelihood can then be fitted using maximum likelihood while the prior can be initialized as the relative class frequency (Murphy, 2012). If x is a d-dimensional vector the algorithm takes the (naive) assumption that all variables are independent so $P(x|i)$ can be factorized into $P(x|i) = P(x_1|i)P(x_2|i)...P(x_d|i)$. A common choice for the functional form of $P(x_j|i)$ is a normal distribution for numeric variables. For categorical variables a multinomial distribution can be used (Hand and Yu, 2001).

Naive Bayes has a quadratic decision boundary in the 2-class setting using a normal distribution, which becomes piecewise quadratic with multiple classes[1]. In a d-dimensional space it therefore has a VC dimension[2] of $2d + \frac{d(d-1)}{2} + 1$. Using d subclasses per original class the VC dimension of the classifier can be increased by at least d times[3]. Using subclasses together with naive Bayes thus substantially increases the expressiveness of the classifier.

2.2.2 C4.5

C4.5 is a decision tree algorithm introduced by Quinlan (1986). It offers support for missing values, handling of categorical as well as numerical values and pruning with heuristic formulas. The splits are chosen based on minimal entropy (Quinlan, 1986).

C4.5 splits on axis values and therefore learns piecewise rectangular concepts. As a decision tree algorithm it has theoretically infinite VC dimension[4]. It therefore does not increase its expressiveness by using subclasses.

2.2.3 Random forest

A random forest is an ensemble method which focuses on combining the output of multiple classifiers (Breiman, 2001). If the individual classifiers are sufficiently independent, their individual errors will partially cancel out resulting in an overall better and more robust model (Breiman, 1996). A random forest takes the majority vote of several decision tree classifiers to compute the output of the combined classifier. To ensure the necessary heterogeneity of the trees the training of the individual classifiers occurs on a random subset of input variables as well as a random subset of data samples. These restrictions force the individual trees to grow different splits and to focus on different aspects of the data (Breiman, 2001).

Random forests have theoretically infinite VC dimension[5] as do decision trees and therefore do not increase their expressiveness by using subclasses either.

[1] *Proof* see appendix section 7.1
[2] *Proof* see appendix section 7.2
[3] *Proof* see appendix section 7.3
[4] *Proof* see appendix section 7.4
[5] *Proof* see appendix section 7.5

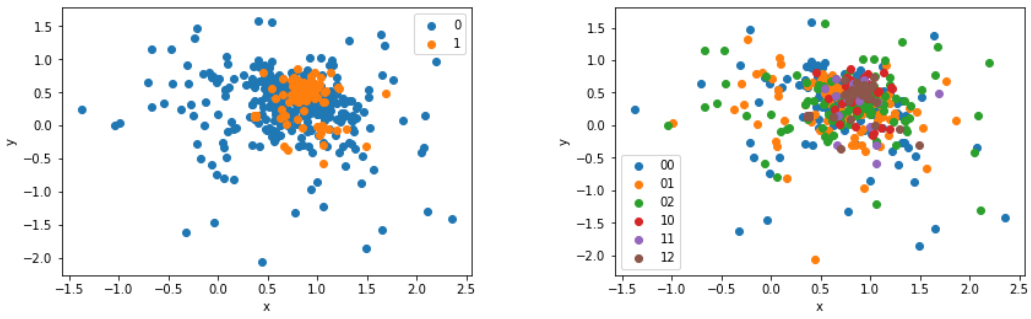
3 Subclass methods

This section presents the methods to artificially induce subclasses into the dataset. They will be visualized on a generated dataset. The subclass methods follow two motivations: First, they can increase the expressiveness of certain classifiers such as naive Bayes (see section 2.2.1). Secondly, they can provide guidance to the classifier during training by grouping data instances together in a meaningful way.

3.1 Random subclasses

Each instance is randomly assigned to a fixed number of subclasses. The instances are equally distributed across all subclasses. This method offers the easiest subclass split and offers no additional guidance to the classifier, as all subclasses follow (in theory) the same distribution with respect to the input features. It is visualized in Figure 2.

Figure 2: Visualization of random subclasses (using 3 subclasses)

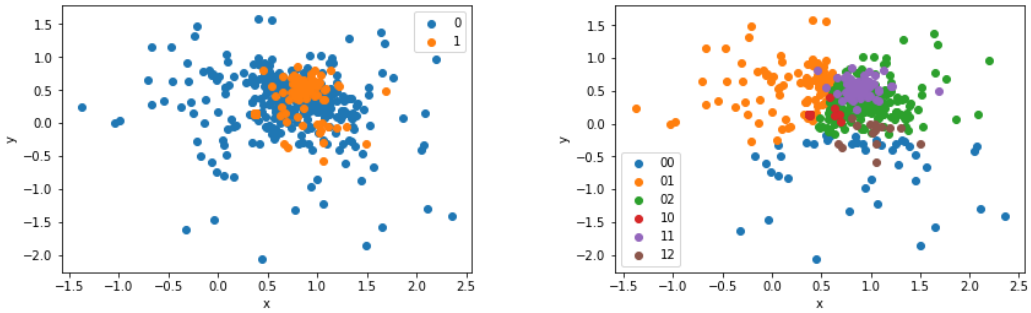


This figure shows the effect of random subclasses on a generated 2-class dataset. Left the original dataset can be seen, right the enhanced one. The first character of the class labels of the enhanced dataset corresponds to the original label.

3.2 Subclasses through clustering

Each class is split into a fixed number of subclasses through clustering. The WEKA clusterer SimpleKMeans is used as clustering algorithm using standard hyperparameters. This method is supposed to generate subclasses which are well-separated in the input space. It is visualized in Figure 3.

Figure 3: Visualization of clustered subclasses (using 3 subclasses)



This figure shows the effect of clustering subclasses on a generated 2-class dataset. Left the original dataset can be seen, right the enhanced one. The first character of the class labels of the enhanced dataset corresponds to the original label.

3.3 Error subclasses

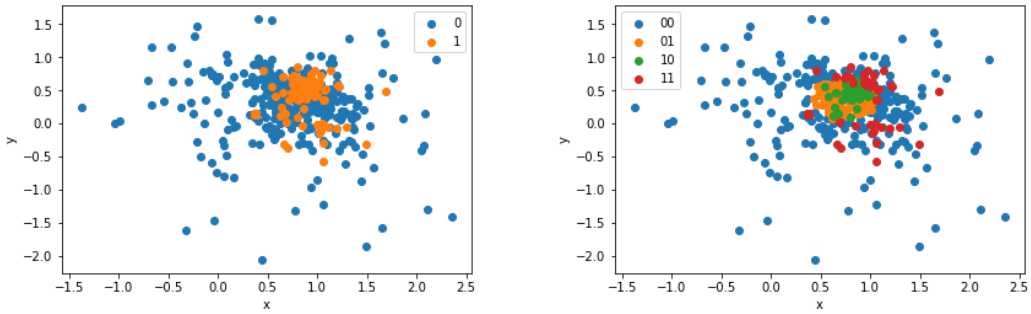
This method uses the most information of all three approaches. While the random subclasses do not use any information, the clustering approach uses the input space to determine the subclasses. This method additionally makes use of the classifier. It works as follows:

The training dataset is evaluated using the original classifier, which results in the following confusion matrix:

A is right and predicted	A is predicted but B is right
B predicted but A is right	B right and predicted

Each entry is assigned a new class, so the original 2-class problem is transformed into a 4-class problem. The evaluation is done using 3-fold cross-validation. Figure 4 visualizes the effect of this split.

Figure 4: Visualization of error subclasses (using naive Bayes as base classifier)



This figure shows the effect of error subclasses on a generated 2-class dataset. Left the original dataset can be seen, right the enhanced one. The first character of the class labels of the enhanced dataset corresponds to the original label.

4 Experimental setting

This section explains the experimental setting. The datasets and the evaluation procedure is presented together with rationales for the setup chosen.

4.1 Datasets

For the experiments 15 2-class datasets have been taken from the UCI-repository (Dua and Graff, 2017). 2-class problems are chosen as they should, in theory, be able to benefit more from subclasses than a multi-class problem as fewer initial classes are present. Additionally they offer a direct relation to the concept of the VC dimension (see section 2.1). The datasets are shown in Table 1 together with some base descriptive statistics.

Table 1: Descriptive statistics

Name	Number of attributes	Number of instances	Share of majority class
breast-cancer	10	286	0.70
breast-w	10	699	0.66
colic	23	368	0.63
credit-a	16	690	0.56
credit-g	21	1000	0.70
diabetes	9	768	0.65
heart-statlog	14	270	0.56
hepatitis	20	155	0.79
ionosphere	35	351	0.64
kr-vs-kp	37	3196	0.52
labor	17	57	0.65
mushroom	23	8124	0.52
sick	30	3772	0.94
sonar	61	208	0.53
vote	17	435	0.61

4.2 Experiments

The three base classifiers introduced in section 2 are used: Naive Bayes, C4.5 and random forest. These algorithms have been chosen to use strong classifiers on one hand, but on the other hand also have some diversity. As consequence, they might react differently to the addition of different subclasses. These differences include naive Bayes being a statistical algorithm with finite VC dimension while C4.5 and random forest are symbolic algorithms. In the same way naive Bayes has finite VC dimension, which can be increased using subclasses, while C4.5 and random forest both have theoretically infinite VC dimension. Random forest is furthermore an ensemble

method. Additionally, all three algorithms can natively handle multiple classes.

In all cases the WEKA implementations (using WEKA 3.8) are used with standard hyperparameters. For random subclasses as well as the clustering approach 2, 5, 10 and 20 subclasses per original class have been tested. The evaluation is done by performing 100 times 2-fold cross-validation.

5 Results and discussion

This section presents the results of the experiments and their discussion. Section 5.1 shows the results of the experiments presented in section 4. Section 5.2 then presents additional experiments based on the MNIST dataset. Theoretical explanations of the results are then provided in section 5.3.

5.1 Evaluation on 2-class problems

5.1.1 Naive Bayes

Tables 2, 3 and 4 show the results for naive Bayes using the different subclass methods. As can be seen, the results are mixed. The random subclasses achieve several significant outperformances over the base classifier, but they also significantly underperform on other datasets. This is interesting insofar, as the subclasses are purely random. Similar results can be observed for clustering and error subclasses. Although all subclass methods differ in their specific performances, the overall tendencies are similar. This will be examined more closely in section 5.3.1.

5.1.2 C4.5

The results for C4.5 can be found in Tables 5, 6 and 7. The subclass algorithms generally underperform the standard C4.5 tree. There are small exceptions to the general underperformance of the subclass methods (such as the *sonar* dataset for clustering subclasses) but the overall tendency is clear. The reasons can be found in the differences of C4.5 to naive Bayes, which will be explained in section 5.3.2.

5.1.3 Random forest

The results for random forest can be found in Tables 8, 9 and 10. Random forest behaves very similar to C4.5 which is not surprising as the algorithms are strongly related. The random subclasses only provide a significant higher performance on the *breast-w* dataset (and *diabetes*), but the magnitudes are small. The same goes for the error subclasses. The clustering method additionally achieves a strong outperformance on the *sonar* dataset. These are exceptions, however, as the the subclass methods generally underperform the base classifier.

Table 2: Accuracies of naive Bayes using random subclasses

	nb random 2	nb random 5	nb random 10	nb random 20	nb base
breast-cancer	0.725	0.730⊕	0.731⊕	0.733⊕	0.723
breast-w	0.961⊕	0.964⊕	0.966⊕	0.966⊕	0.960
colic	0.786	0.787	0.777⊖	0.764⊖	0.788
credit-a	0.782⊕	0.792⊕	0.803⊕	0.809⊕	0.777
credit-g	0.742⊖	0.740⊖	0.735⊖	0.729⊖	0.745
diabetes	0.752	0.754⊕	0.754⊕	0.746⊖	0.752
heart-statlog	0.840	0.838	0.828⊖	0.814⊖	0.840
hepatitis	0.844⊕	0.848⊕	0.829⊖	0.816⊖	0.839
ionosphere	0.851⊕	0.893⊕	0.913⊕	0.920⊕	0.827
kr-vs-kp	0.872⊖	0.871⊖	0.869⊖	0.868⊖	0.873
labor	0.887⊖	0.866⊖	0.829⊖	0.799⊖	0.899
mushroom	0.945⊖	0.939⊖	0.936⊖	0.934⊖	0.951
sick	0.922⊖	0.926	0.941⊕	0.957⊕	0.925
sonar	0.695⊕	0.728⊕	0.747⊕	0.719⊕	0.682
vote	0.901	0.903⊕	0.904⊕	0.906⊕	0.901

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 3: Accuracies of naive Bayes using clustering subclasses

	nb cluster 2	nb cluster 5	nb cluster 10	nb cluster 20	nb base
breast-cancer	0.718⊖	0.727⊕	0.732⊕	0.737⊕	0.723
breast-w	0.958⊖	0.946⊖	0.943⊖	0.940⊖	0.960
colic	0.798⊕	0.791	0.780⊖	0.762⊖	0.788
credit-a	0.781⊕	0.789⊕	0.798⊕	0.798⊕	0.777
credit-g	0.730⊖	0.717⊖	0.713⊖	0.714⊖	0.745
diabetes	0.724⊖	0.700⊖	0.696⊖	0.690⊖	0.752
heart-statlog	0.814⊖	0.788⊖	0.769⊖	0.765⊖	0.840
hepatitis	0.842	0.845⊕	0.821⊖	0.794⊖	0.839
ionosphere	0.897⊕	0.898⊕	0.870⊕	0.835⊕	0.827
kr-vs-kp	0.825⊖	0.847⊖	0.877⊕	0.900⊕	0.873
labor	0.875⊖	0.824⊖	0.766⊖	0.855⊖	0.899
mushroom	0.981⊕	0.988⊕	0.994⊕	0.997⊕	0.951
sick	0.936⊕	0.948⊕	0.955⊕	0.959⊕	0.925
sonar	0.721⊕	0.789⊕	0.767⊕	0.695⊕	0.682
vote	0.934⊕	0.948⊕	0.945⊕	0.940⊕	0.901

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 4: Accuracies of naive Bayes using error subclasses

	nb error	nb base
breast-cancer	0.714⊖	0.723
breast-w	0.948⊖	0.960
colic	0.799⊕	0.788
credit-a	0.809⊕	0.777
credit-g	0.736⊖	0.745
diabetes	0.743⊖	0.752
heart-statlog	0.813⊖	0.840
hepatitis	0.833⊖	0.839
ionosphere	0.869⊕	0.827
kr-vs-kp	0.904⊕	0.873
labor	0.885⊖	0.899
mushroom	0.984⊕	0.951
sick	0.958⊕	0.925
sonar	0.764⊕	0.682
vote	0.955⊕	0.901

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 5: Accuracies of C4.5 using random subclasses

	tree random 2	tree random 5	tree random 10	tree random 20	tree base
breast-cancer	0.686⊖	0.676⊖	0.671⊖	0.668⊖	0.710
breast-w	0.939⊖	0.933⊖	0.931⊖	0.929⊖	0.945
colic	0.837⊖	0.834⊖	0.830⊖	0.831⊖	0.848
credit-a	0.831⊖	0.821⊖	0.817⊖	0.818⊖	0.850
credit-g	0.693⊖	0.690⊖	0.684⊖	0.677⊖	0.712
diabetes	0.704⊖	0.698⊖	0.698⊖	0.695⊖	0.730
heart-statlog	0.759⊖	0.746⊖	0.749⊖	0.751⊖	0.773
hepatitis	0.777⊖	0.775⊖	0.772⊖	0.765⊖	0.795
ionosphere	0.893⊕	0.886	0.872⊖	0.855⊖	0.887
kr-vs-kp	0.989⊖	0.985⊖	0.979⊖	0.972⊖	0.990
labor	0.791	0.793	0.767⊖	0.772	0.785
mushroom	1	1 ⊖	1 ⊖	1 ⊖	1
sick	0.983⊖	0.980⊖	0.980⊖	0.979⊖	0.984
sonar	0.682⊖	0.679⊖	0.669⊖	0.659⊖	0.700
vote	0.952	0.949⊖	0.948⊖	0.950⊖	0.953

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 6: Accuracies of C4.5 using clustering subclasses

	tree cluster 2	tree cluster 5	tree cluster 10	tree cluster 20	tree base
breast-cancer	0.697⊖	0.685⊖	0.686⊖	0.678⊖	0.710
breast-w	0.942⊖	0.938⊖	0.936⊖	0.936⊖	0.945
colic	0.835⊖	0.833⊖	0.830⊖	0.828⊖	0.848
credit-a	0.841⊖	0.839⊖	0.831⊖	0.821⊖	0.850
credit-g	0.681⊖	0.670⊖	0.666⊖	0.664⊖	0.712
diabetes	0.703⊖	0.685⊖	0.678⊖	0.675⊖	0.730
heart-statlog	0.746⊖	0.743⊖	0.736⊖	0.735⊖	0.773
hepatitis	0.792	0.783⊖	0.775⊖	0.757⊖	0.795
ionosphere	0.871⊖	0.839⊖	0.821⊖	0.811⊖	0.887
kr-vs-kp	0.981⊖	0.965⊖	0.958⊖	0.955⊖	0.990
labor	0.791	0.766⊖	0.770⊖	0.720⊖	0.785
mushroom	1	1 ⊖	1	1 ⊖	1
sick	0.981⊖	0.977⊖	0.975⊖	0.972⊖	0.984
sonar	0.711⊕	0.711⊕	0.696	0.674⊖	0.700
vote	0.951⊖	0.950⊖	0.947⊖	0.938⊖	0.953

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 7: Accuracies of C4.5 using error subclasses

	tree error	tree base
breast-cancer	0.706	0.710
breast-w	0.944	0.945
colic	0.843⊖	0.848
credit-a	0.846⊖	0.850
credit-g	0.706⊖	0.712
diabetes	0.717⊖	0.730
heart-statlog	0.763⊖	0.773
hepatitis	0.788⊖	0.795
ionosphere	0.887	0.887
kr-vs-kp	0.988⊖	0.990
labor	0.788	0.785
mushroom	1	1
sick	0.983⊖	0.984
sonar	0.683⊖	0.700
vote	0.951⊖	0.953

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 8: Accuracies of random forest using random subclasses

	rf random 2	rf random 5	rf random 10	rf random 20	rf base
breast-cancer	0.679⊖	0.675⊖	0.674⊖	0.675⊖	0.692
breast-w	0.965	0.965⊕	0.967⊕	0.967⊕	0.964
colic	0.846⊖	0.843⊖	0.829⊖	0.818⊖	0.848
credit-a	0.857⊖	0.850⊖	0.847⊖	0.842⊖	0.860
credit-g	0.739⊖	0.732⊖	0.730⊖	0.729⊖	0.748
diabetes	0.757	0.759⊕	0.754	0.753⊖	0.757
heart-statlog	0.815	0.816	0.814	0.814	0.816
hepatitis	0.830	0.827⊖	0.822⊖	0.816⊖	0.832
ionosphere	0.931	0.925⊖	0.912⊖	0.890⊖	0.933
kr-vs-kp	0.985⊖	0.983⊖	0.981⊖	0.978⊖	0.986
labor	0.860	0.853	0.842⊖	0.850	0.857
mushroom	1	1	1	1	1
sick	0.980	0.979⊖	0.977⊖	0.974⊖	0.980
sonar	0.803	0.794⊖	0.797	0.789⊖	0.801
vote	0.959	0.957⊖	0.957⊖	0.954⊖	0.960

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 9: Accuracies of random forest using clustering subclasses

	rf cluster 2	rf cluster 5	rf cluster 10	rf cluster 20	rf base
breast-cancer	0.677⊖	0.677⊖	0.676⊖	0.677⊖	0.692
breast-w	0.966⊕	0.966⊕	0.966⊕	0.967⊕	0.964
colic	0.845⊖	0.839⊖	0.830⊖	0.819⊖	0.848
credit-a	0.857⊖	0.844⊖	0.835⊖	0.832⊖	0.860
credit-g	0.734⊖	0.729⊖	0.727⊖	0.725⊖	0.748
diabetes	0.749⊖	0.736⊖	0.733⊖	0.731⊖	0.757
heart-statlog	0.808⊖	0.802⊖	0.805⊖	0.806⊖	0.816
hepatitis	0.830	0.822⊖	0.822⊖	0.813⊖	0.832
ionosphere	0.932	0.917⊖	0.899⊖	0.879⊖	0.933
kr-vs-kp	0.982⊖	0.976⊖	0.974⊖	0.973⊖	0.986
labor	0.860	0.841⊖	0.844⊖	0.853	0.857
mushroom	1	1	1	1	1
sick	0.979⊖	0.977⊖	0.975⊖	0.973⊖	0.980
sonar	0.815⊕	0.819⊕	0.825⊕	0.819⊕	0.801
vote	0.956⊖	0.954⊖	0.952⊖	0.950⊖	0.960

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

Table 10: Accuracies of random forest using error subclasses

	rf error	rf base
breast-cancer	0.679⊖	0.692
breast-w	0.965⊕	0.964
colic	0.851⊕	0.848
credit-a	0.858⊖	0.860
credit-g	0.742⊖	0.748
diabetes	0.754⊖	0.757
heart-statlog	0.814	0.816
hepatitis	0.830	0.832
ionosphere	0.931	0.933
kr-vs-kp	0.985⊖	0.986
labor	0.855	0.857
mushroom	1	1
sick	0.980	0.980
sonar	0.798	0.801
vote	0.958⊖	0.960

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

5.1.4 Comparison of the three base classifiers

Of all three classifiers, only naive Bayes manages to benefit from the subclass methods. A first explanation might be that the subclasses are able to increase the expressiveness of naive Bayes (see section 2.2.1), neither of which is true for C4.5 or random forest. A more detailed explanation will be presented in section 5.3.

Of all 3 classifiers, random forest is the only ensemble method, which should offer increased robustness to changes to the base classifier. Table 11 shows the mean dataset-wise variance of each algorithm. As can be seen, the random forest classifier has the smallest variance, which confirms the assumption. The small variance makes differences in performances (positive or negative) less likely. As random forest does not profit from the addition of subclasses anyway the subclasses have thus less negative influence on random forest than on C4.5.

5.2 Evaluation on the MNIST dataset

In addition to the experiments described in section 4, additional experiments have been carried out using a larger dataset. The MNIST dataset is an image labeling dataset offering 784 attributes (28 times 28 pixels), 70000 instances and 10 base classes (LeCun et al., 1998). It is thus much bigger than the datasets used this far. This dataset serves two purposes: First, the question is whether the findings of the section 5.1 are transferable to larger datasets. Secondly,

Table 11: Variance of performance per classifier

	Variance
Naive Bayes	4.8e−4
C4.5	1.7e−4
Random forest	6.5e−5

The variance shown for each base classifier has been computed as follows: On each dataset the variance of the performances of all methods using this classifier has been computed (this includes the performances of all subclass methods and the performance of the base classifier). This yields a variance estimate for each dataset for each base classifier. Then the mean over all datasets is taken.

it is of interest whether artificial subclasses can beat an already present subclass structure. To achieve this, the 10-class problem is converted into a 2-class one using the NDC technique of Melnikov and Hüllermeier (2018). This specific technique has been used as it is the computationally most efficient of all the non-agnostic techniques presented in the paper. To evaluate the second question, an additional classifier based on the original subclasses will be trained. For computational reasons, the evaluation has been changed to 5-fold cross-validation. The results are shown in Table 12.

First of all, the base classifiers performs best for C4.5 and random forest, which is in line with the findings from the previous sections. It is interesting that the classifier based on original classes performs slightly worse in both cases. Since the effect is small and the classes have been artificially merged, both classifiers can be said to perform equally well. For random forest, the overall deviation in performances is again very small. Therefore the subclassing methods result in no big differences. For C4.5, they all have a detrimental effect on the classification performance.

The results are again very different for naive Bayes. The base classifier performs badly, only achieving an accuracy in the 60s. Using random subclasses, naive Bayes performs even worse. Cluster and error, however, beat the base classifier by a large margin. So does the classifier based on the original classes. The best performance is obtained by using 20 clusters (the largest amount tested), which results in a total of 40 classes. This is 4 times the amount of the number of original classes. This subclass method even beats the classifier based on the original classes.

5.3 Theoretical discussion of results

This section presents theoretical explanations for the results seen. As naive Bayes is the only base classifier managing to benefit from the subclass methods, emphasis is put to carefully explain why and when an outperformance is achieved. C4.5 and random forest, on the other hand, do not benefit from subclasses. Their behavior is explained afterwards.

Table 12: Results on the MNIST dataset

	random 2	random 5	random 10	random 20	cluster 2	cluster 5	cluster 10	cluster 20	error	base	orig classes
naive Bayes	0.610	0.583	0.564	0.541	0.849	0.831	0.892	0.917	0.805	0.624	0.888
C4.5	0.960	0.954	0.951	0.945	0.959	0.960	0.957	0.957	0.959	0.962	0.958
random forest	0.989	0.987	0.987	0.984	0.989	0.988	0.987	0.987	0.989	0.989	0.988

Table 13: Max and mean deviations from base performance for random and cluster subclasses using naive Bayes

	random 2	random 5	random 10	random 20	cluster 2	cluster 5	cluster 10	cluster 20
Mean of positive deviations	0.0062	0.0169	0.0266	0.0306	0.025	0.0344	0.0348	0.025
Mean of negative deviations	-0.0039	-0.0094	-0.0191	-0.0271	-0.021	-0.0411	-0.048	-0.0433
Maximum positive deviation	0.0243	0.0659	0.0859	0.0924	0.07	0.1069	0.0845	0.0456
Maximum negative deviation	-0.0116	-0.0332	-0.0698	-0.1002	-0.0487	-0.0747	-0.133	-0.0753

Table 14: Number of nodes for C4.5

	random 2	random 5	random 10	random 20	cluster 2	cluster 5	cluster 10	cluster 20	error	base
breast-cancer	119	223	218	218	88	75	194	250	46	6
breast-w	131	261	283	327	63	93	119	151	29	27
colic	45	138	183	236	32	102	131	190	6	6
credit-a	262	494	529	642	78	148	274	301	71	42
credit-g	477	675	846	794	335	463	624	719	185	140
diabetes	293	477	543	573	133	189	215	279	169	39
heart-statlog	119	157	197	195	39	71	85	103	53	35
hepatitis	61	75	101	101	23	61	65	71	23	21
ionosphere	73	187	241	257	29	57	93	111	39	35
kr-vs-kp	672	1510	1719	1868	151	508	699	883	47	59
labor	13	33	32	28	14	27	26	29	3	5
mushroom	4652	10313	12732	14169	224	2208	2882	5093	30	30
sick	475	1813	2169	2365	57	213	265	462	69	61
sonar	73	113	129	151	43	57	73	91	51	35
vote	53	127	143	139	45	83	91	111	15	11

5.3.1 Naive Bayes

For this section, there are several important questions to answer. First of all, it is not apparent why introducing random subclasses should increase classification performance. This is examined first theoretically in section 5.3.1.1 and then using simulated datasets in section 5.3.1.2. Afterwards the different subclass methods are compared to explain differences in their performances.

5.3.1.1 Theoretical explanation of the performance of random subclasses

The most straightforward explanation is that introducing subclasses (no matter how) increases the expressiveness of the naive Bayes classifier (see section 2.2.1). Put more intuitively, as normal and multinomial distributions are both unimodal, naive Bayes can fit exactly one ellipsoid per class to explain the data. Certain datasets, however, might require naive Bayes to use several ellipsoids to better explain the data. This can be achieved by splitting each class into n subclasses which results in naive Bayes fitting n such ellipsoids per original class. Random subclasses thus can increase performance of a classifier like naive Bayes by increasing the expressiveness of the classifier. As each subclass follows the same distribution as the original class, no guidance is provided to the classifier on how to use this additional expressiveness.

The opposite explanation can be used for the datasets were the random subclasses worsen the performance. The dataset is best described by using a single ellipsoid per class. In most cases where the performance is worse for random subclasses it can be seen that it is monotonically decreasing the more subclasses are used. Regressing the attributes of Table 1 on the outperformance of random subclasses over the base classifier[6] the number of attributes is generally significantly[7] positive, while the other attributes offer no significance. As with increasing number of attributes the complexity of the dataset will generally increase, it makes sense that additional ellipsoids per class prove useful.

5.3.1.2 Simulated datasets for explaining the performance of random subclasses

Figure 5 shows an exemplary dataset that should be easy to solve using two subclasses.

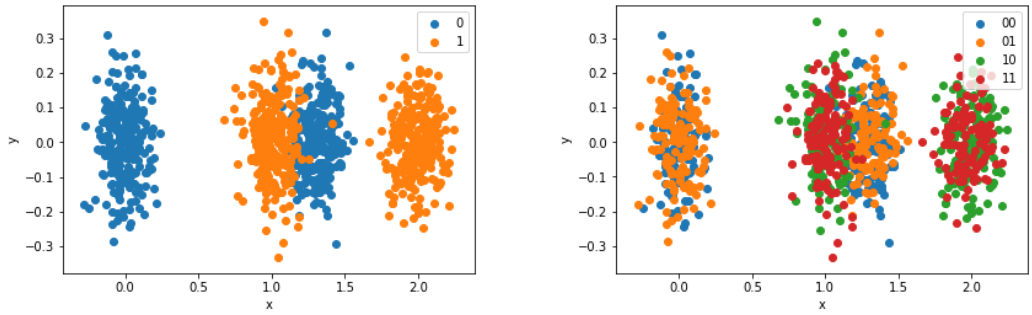
[6] The regression is built as follows:

$$p_1 - p_2 = \beta_0 + \beta_1 \text{num attributes} + \beta_2 \text{num instances} + \beta_3 \text{share majority class} + u$$
$$p_1 = \text{performance of a classifier using } n \text{ subclasses}$$
$$p_2 = \text{performance of the base classifier}$$

with the βs being the parameters and u the error term. As there are 4 different subclass configurations for random and clustering subclasses, there are four regressions for those subclass methods and four estimates for every parameter.

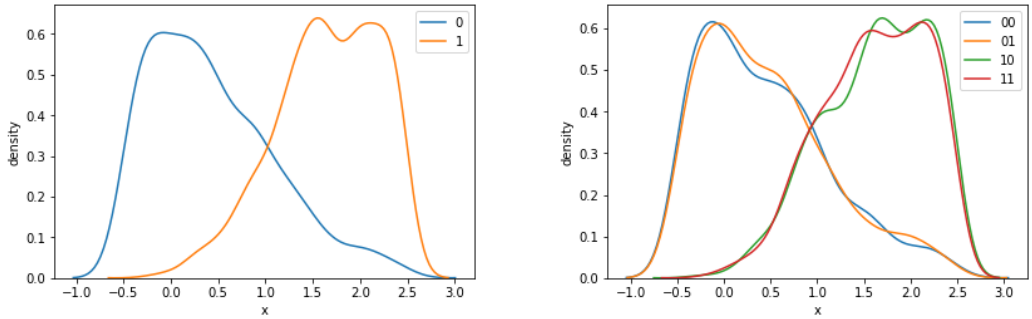
[7] When a significance is mentioned in this paper without a confidence level, the level of 5% is implied.

Figure 5: Example dataset where random subclasses fail to improve performance



Left the original dataset is shown, right the dataset that random subclasses (using 2 subclasses) produces. The accuracy decreases from 65% to 64% (using 10-fold cross-validation).

Figure 6: Density of naive Bayes using random subclasses

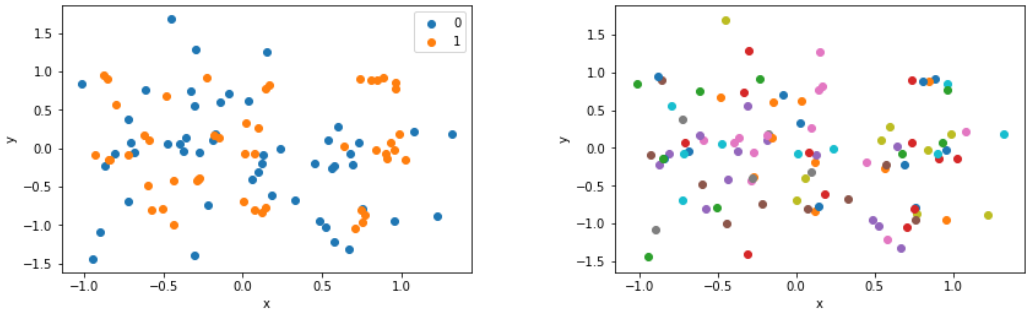


This figure shows $\tilde{p}(i|x) = \frac{p(i|x)}{\sum_{j=1}^{N} p(j|x)}$ for the horizontal axis of Figure 5. This is a normal distribution divided by a sum of normal distributions and therefore will not resemble a normal distribution nor be unimodal. Left the original classifier is shown, right the one using 2 subclasses. As can be seen on the right image the distributions of each two subclasses are very similar.

Using two random subclasses, the accuracy barely changes. The impression might have arisen that the classifier was free to chose where to put the additional ellipsoids since the random subclasses output subclasses with similar distributions as the original class. This is not the case. In reality, the classifier just fits the same distribution twice (see Figure 6 for illustration). As a result, the performance is almost identical to the original classifier. Mathematically speaking, the maximum likelihood estimation of the class likelihoods (see section 2.2.1) is independent of one another.

A new dataset has been simulated to show how naive Bayes can profit from the additional ellipsoids (see Figure 7). The base accuracy is 45%, using 20 random subclasses increases the accuracy to 52% (using 10-fold cross-validation). The dataset is very scattered. As a result, randomly assigning subclasses will make the distributions of the subclasses by chance vary enough to move the ellipsoids away from one another. Therefore the random subclasses enable outperformance if the dataset has a similar, scattered fashion. This allows the random subclasses to

Figure 7: Example dataset where random subclasses increase performance



Left the original dataset is shown, right the dataset that random subclasses (using 20 subclasses) produces. Class 0 has been simulated using a normal distribution. Class 1 has been simulated using several normal distributions with small variance but uniformly distributed center (class 1). The class labels are not shown on the right image as there are 40 labels.

assume different distributions[8]. These different distributions, however, also have to be helpful for the classification task at hand, otherwise the performance will deteriorate. As naive Bayes evaluates each attribute independently (see section 2.2.1), the number of attributes being often significantly positive in a regression on the outperformance makes sense (see previous section). Having more attributes increases the likelihood of having such a scattered landscape across one dimension. As the subclasses are assigned randomly, the subclass distribution across each dimension is independent of one another. On the other hand, this method should work better with a lower number of instances in the dataset as this heightens the chance of having different distributions in the subclasses. The data only partly confirms this. While the number of instances is always negative for the regression on the outperformance of the random methods, it is never significant. It is thus small in absolute value.

5.3.1.3 General comparison of subclass methods

As naive Bayes can profit from the sheer addition of subclasses it is now of interest whether the creation technique for the subclasses themselves is useful. As the performances of all three methods are quite similar on the 2-class problems, the expressiveness argument seems to be a strong one. There are differences, however. Especially for datasets having more attributes, clustering and error outperform random subclasses. This can be seen at the datasets such as *sonar*, *mushroom* and *kr-vs-kp*. A regression has been built regressing the difference of the outperformance of random subclasses and the outperformance of either cluster or error subclasses onto the dataset attributes of Table 1. The previous statement is confirmed as the regression

[8] As it seems, the performance is reliant on random perturbations in the subclass assignments. This should increase the variance of the performance estimates in comparison to other subclass methods. In fact the variance of naive Bayes using 20 random subclasses on the dataset of Figure 7 using 10-fold cross-validation varies substantially and can even drop to levels of the base classifier. However, repeating the evaluation 100 times also yields a value close to 52% which makes it a good estimate for the performance of the classifier.

yields an always negative and several times significant coefficient for the number of attributes[9]. The coefficient for the number of instances is also always negative and several times significant, which is in line with the argumentation of section 5.3.1.2.

Based on the results from the experiments using the 2-class problems, an increasing dataset complexity seems to favor a more expressive classifier (benefiting all split techniques). But it also favors guiding the position of the different ellipsoids (benefiting cluster and error over random subclasses). Then an increasing number of instances favors the clustering and error methods over random subclasses. One reason being that the first two methods can make use of the additional data to better identify suitable subclasses. Additionally, as mentioned in section 5.3.1.1, more data will reduce the variation in the subclasses. Random subclasses will thus have more difficulties to produce substantially different subclass distributions. Taking the results on the MNIST dataset into account puts emphasis on the last point. While clustering and error subclasses perform well, as expected, random subclasses worsen the performance for naive Bayes. Just increasing the expressiveness alone does not seem to work for such a large dataset. But by using meaningful subclasses which guide the classifier on how to use the additional expressiveness, performance improvements close to 30% can be achieved.

Performance-wise, every method has a niche, respectively datasets where the method performs best: random has 5 datasets, where any of the random subclass classifiers perform best, clustering has four, error and the base classifier have 3 each. The next sections will compare the different subclass methods in greater detail.

5.3.1.4 Comparing random and clustering subclasses

For illustration, the dataset from Figure 5 is used again to evaluate a clustering classifier using two subclasses. This is shown in Figure 8. The accuracy improved by 30% which is a big contrast to the performance of the random subclasses which did not yield any improvement (see section 5.3.1.2).

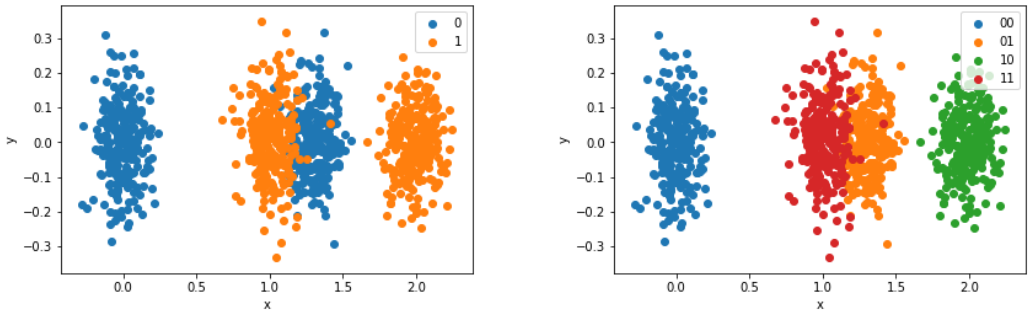
Even for the dataset used in Figure 7, clustering beats random at 64% using 20 subclasses (random achieved 52% with the same amount of subclasses). In fact, given the explanation in section 5.3.1.2, using random subclasses is unlikely to grant a 30% increase in accuracy as it relies on random fluctuations. At the same time, clustering methods can yield large improvements for datasets as the one in Figure 8. To confirm this, Table 13 lists the maximum and mean upward

[9] The regression is built as follows:

$$p_1 - p_2 = \beta_0 + \beta_1 \text{num attributes} + \beta_2 \text{num instances} + \beta_3 \text{share majority class} + u$$
$$p_1 = \text{outperformance of a naive Bayes classifier using random subclasses}$$
$$p_2 = \text{outperformance of a naive Bayes classifier using clustering or error subclasses}$$

with the βs being the parameters and u the error term. As naive Bayes using clustering or random subclasses exists in four different configurations there are several possible regressions using this formula

Figure 8: Example dataset where clustering subclasses increase performance



Left the original dataset is shown, right the dataset that clustering subclasses (using 2 subclasses) produces. Not surprisingly, the accuracy is increased from 65% to 97% (using 10-fold cross-validation).

and downward deviation from the base classifier performance for random and cluster methods using naive Bayes. As can be seen, the clustering subclass method has higher deviations in both directions. Random subclasses are therefore less likely to yield a strong outperformance, but they will also worsen the classifier by a smaller amount.

Lastly, clustering subclasses find subsets of data by jointly evaluating every dimension. On the other hand, random subclasses produce an independent subclass structure for each dimension. If the data has only a few important dimensions, random subclasses might have better chances of producing meaningful differences along those dimensions. Unlike clustering subclasses, random subclasses will not be distracted by the other, unimportant dimensions.

5.3.1.5 Comparing clustering and error subclasses

The comparison between error and clustering subclasses is interesting. Clustering has the flexibility of offering more subclasses than error. On the other hand, error might provide better guidance as it uses the class label information to build the subclasses. In fact, error always beats clustering in at least 10 of the 15 datasets, not always in the same, though. So the better guidance seems to offset the increased number of class centers cluster has to offer. This changes for a large dataset such as the MNIST dataset, however. The best clustering method on that dataset outperforms error by more than 10%. Using error subclasses can also produce large improvements as it achieves an accuracy of 94% on the dataset of Figure 8 (using 10-fold cross-validation).

5.3.2 C4.5 and random forest

This section aims to explain why C4.5 and random forest do not benefit from using subclasses. This could be seen clearly from the experiments in section 5.1 as well as section 5.2. As the algorithms are similar the following sections will focus on C4.5, but the same arguments apply to random forest. The addition being, that any effects are smaller on random forest because it is an ensemble method (see section 5.3.1.3).

5.3.2.1 Explanation of the performance of C4.5

C4.5, as shown in section 2.2.2, already has infinite VC dimension and therefore does not benefit from increased expressiveness as does naive Bayes. Put intuitively, it does not learn ellipsoids but partitions the space into axis-parallel splits. These splits can be placed arbitrarily independent of the number of classes to learn. However, not only random subclasses fail to improve performance but the other methods as well. Therefore the aspect of providing meaningful subclasses to the classifier does not work here either. An explanation is that the subclass creation methods create subclasses which are unnatural to learn for C4.5. This is easy to see for the clustering methods as k-means tends to produce round clusters (as it uses the euclidean distance) while C4.5 can only learn rectangles. It can also be shown more generally:

As C4.5 tries to minimize the entropy having the subclasses reducing the entropy would be helpful. However, the introduction of subclasses can be shown to always increase the entropy on the dataset, as will be proven shortly. The entropy is defined as:

$$I = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

with N being the number of classes and p_i the respective probability of class i. The logarithm is strictly monotonically increasing, so the negative logarithm is strictly monotonically decreasing. The logarithm is also a concave function which makes the negative logarithm convex. For convex functions, the following holds for a function f over a set X:

$$\forall x_1, x_2 \in X, \forall t \in [0,1] : f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

Now it will be shown that splitting up an existing class C into N subclasses will always increase the entropy. For starters, with p_i being the relative frequency of subclass i, so for class C

$p_C = \sum_{i=1}^{N} p_i$ holds. f will stand for the negative logarithm. The *proof*:

$$
\begin{aligned}
p_C f(p_C) &= (\sum_{i=1}^{N} p_i) f(\sum_{i=1}^{N} p_i) \\
&= (\sum_{i=1}^{N} p_i) f(p_1 + \sum_{i=2}^{N} p_i) \\
&= (\sum_{i=1}^{N} p_i) f(\frac{p_1}{\sum_{i=1}^{N} p_i} \sum_{i=1}^{N} p_i + \frac{\sum_{i=2}^{N} p_i}{\sum_{i=1}^{N} p_i} \sum_{i=1}^{N} p_i) \\
&\overset{\text{convexity}}{\leq} (\sum_{i=1}^{N} p_i) \frac{p_1}{\sum_{i=1}^{N} p_i} f(\sum_{i=1}^{N} p_i) + (\sum_{i=1}^{N} p_i) \frac{\sum_{i=2}^{N} p_i}{\sum_{i=1}^{N} p_i} f(\sum_{i=1}^{N} p_i) \\
&= p_1 f(\sum_{i=1}^{N} p_i) + (\sum_{i=2}^{N} p_i) f(\sum_{i=1}^{N} p_i) \\
&\overset{\text{strictly monontonically decreasing}}{<} p_1 f(p_1) + (\sum_{i=2}^{N} p_i) f(\sum_{i=2}^{N} p_i)
\end{aligned}
$$

Now this can be repeated:

$$
\begin{aligned}
p_1 f(p_1) + (\sum_{i=2}^{N} p_i) f(\sum_{i=2}^{N} p_i) &< p_1 f(p_1) + p_2 f(p_2) + (\sum_{i=3}^{N} p_i) f(\sum_{i=3}^{N} p_i) \\
&\dots \\
&< \sum_{i=1}^{N} (p_i f(p_i))
\end{aligned}
$$

Explanation: The convexity axiom was used setting $t = \frac{p_1}{\sum_{i=1}^{N} p_i}$ and therefore $1-t = \frac{\sum_{i=2}^{N} p_i}{\sum_{i=1}^{N} p_i}$. Furthermore set $x_1 = \sum_{i=1}^{N} p_i$ and $x_2 = \sum_{i=1}^{N} p_i$. This replaces the sum in front of the logarithm by the term that is inside it as the initial term in front of the logarithm will be put into the denominator. The term inside the logarithm is then corrected using the strict monotonic decrease of the negative logarithm. This is then repeated until the sum inside the logarithm has disappeared.

Consequently, introducing subclasses into the dataset (independent of the exact method) always increases the entropy and thus worsens the starting point for decision trees. The additional argument was given that the subclasses might introduce data splits which are unnatural to learn for C4.5. In fact, based on Table 14, C4.5 builds much larger trees for the subclass methods, especially for the random and cluster subclasses. The algorithm is therefore negatively affected by the presence of subclasses.

Table 15: Accuracies of C4.5 using clustering subclasses with manhattan distance

	manhattan 2	manhattan 5	manhattan 10	manhattan 20	tree base
breast-cancer	0.698⊖	0.691⊖	0.684⊖	0.677⊖	0.712
breast-w	0.941⊖	0.940⊖	0.938⊖	0.937⊖	0.943
colic	0.836⊖	0.832⊖	0.829⊖	0.827⊖	0.847
credit-a	0.842⊖	0.837⊖	0.833⊖	0.824⊖	0.847
credit-g	0.680⊖	0.669⊖	0.670⊖	0.667⊖	0.711
diabetes	0.707⊖	0.684⊖	0.681⊖	0.676⊖	0.728
heart-statlog	0.747⊖	0.739⊖	0.736⊖	0.734⊖	0.769
hepatitis	0.789	0.774⊖	0.768⊖	0.754⊖	0.794
ionosphere	0.871⊖	0.841⊖	0.823⊖	0.806⊖	0.886
kr-vs-kp	0.980⊖	0.967⊖	0.959⊖	0.953⊖	0.990
labor	0.786	0.772	0.754⊖	0.722⊖	0.779
mushroom	1.000	1.000⊖	1.000⊖	1.000	1.000
sick	0.981⊖	0.977⊖	0.975⊖	0.972⊖	0.984
sonar	0.711⊕	0.709⊕	0.695	0.670⊖	0.694
vote	0.952	0.951⊖	0.947⊖	0.942⊖	0.953

On the right the unmodified base classifier can be seen. A ⊕ signifies significant outperformance of the modified classifier over the base classifier, a ⊖ signifies significant underperformance. Both are calculated using a t-test with 5% confidence level.

5.3.2.2 C4.5 with clustering subclasses using manhattan distance

One of the arguments used in the previous section is that the round classes produced by the clustering subclasses are unnatural to learn for C4.5. Therefore additional clustering experiments have been done by again using clustering subclasses but substituting the euclidean by the manhattan distance. This ensures at least piecewise rectangular clusters and cluster boundaries which always lie along the data axes. The results are shown in Table 15. There is no big difference to the euclidean clustering method, comparing both in which method performs better yields 8 wins for the manhattan distance and 7 for the euclidean distance. So the difference is small. The exact cluster shape therefore has little effect and the other arguments mentioned in the previous section are still valid. Therefore the base classifier generally performs best.

6 Conclusion and further work

Based on the knowledge that a known subclass structure can help classification performance this paper tested if this can be exploited for unknown subclass structures. Experiments have been carried out on 15 different datasets, using naive Bayes, C4.5 and random forest as base classifiers. The artificial subclasses were introduced completely at random, by using clustering and by separating the regions where the classifier produces errors.

Naive Bayes has shown the capacity for improvement even when using random subclasses. Random subclasses even perform best the most times compared to the other classifiers, but cannot produce outperformances of magnitudes which are comparable to the other two (see section 5.3.1.3). The general performance of all three subclass methods has been explained by the fact that introducing subclasses increases the expressiveness of the classifier which can be beneficial on complicated datasets. The more complicated the dataset (measured in the number of attributes), the more the classifier benefits from having non-random, more meaningful subclasses. The result will then be large gains on the right datasets. This is confirmed by doing additional experiments on a large dataset (MNIST) where cluster and error subclasses improve the naive Bayes performance by up to 30%, but random subclasses worsen the performance.

C4.5 and random forest, however, do not benefit from using subclasses. One reason being that the algorithms do not increase their expressiveness with growing number of classes. Additionally, it has been shown that the subclasses introduce suboptimal data splits, forcing the algorithms to grow larger and worse trees (see section 5.3.2.1).

In conclusion, introducing artificial subclasses can improve classifiers if certain preconditions like the increased expressiveness with higher number of classes are met. Further work is needed to verify this on other classifiers. Additionally, only three basic techniques have been tested in this paper, additional, more powerful techniques can be researched. As this paper only researched the effect of artificial subclasses on 2-class problems, further work could look into whether the effect is different in a multi-class setting.

On the MNIST dataset the artificial subclasses managed to beat the original subclasses using naive Bayes. As the original subclass hierarchy was artificially created using the NDC method (see section 5.2), additional experiments using datasets with a natural hierarchy would be of interest.

Bibliography

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chen, Z., Ding, R., Chin, T.-W., and Marculescu, D. (2018). Understanding the impact of label granularity on cnn-based image classification. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 895–904. Institute of Electrical and Electronics Engineers.

Dietterich, T. G. and Bakiri, G. (1994). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Frank, E. and Kramer, S. (2004). Ensembles of nested dichotomies for multi-class problems. In *Proceedings of the twenty-first international Conference on Machine Learning*, page 39. Association for Computing Machinery.

Goldberg, P. W. and Jerrum, M. R. (1995). Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18(2-3):131–148.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Hand, D. J. and Yu, K. (2001). Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398.

Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling. In *Advances in Neural Information Processing Systems*, pages 507–513.

Hoffmann, A., Kwok, R., and Compton, P. (2001). Using subclasses to improve classification learning. In *European Conference on Machine Learning*, pages 203–213. Springer.

LeCun, Y., Cortes, C., and Burges, C. J. (1998). The mnist database of handwritten digits, 1998.

Luo, Y. (2008). Can subclasses help a multiclass learning problem? In *2008 IEEE Intelligent Vehicles Symposium*, pages 214–219. Institute of Electrical and Electronics Engineers.

Melnikov, V. and Hüllermeier, E. (2018). On the effectiveness of heuristics for learning nested dichotomies: an empirical analysis. *Machine Learning*, 107(8-10):1537–1560.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141.

Vapnik, V. (1999). *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264.

7 Appendix

7.1 Proof that naive Bayes using a normal distribution has a quadratic decision boundary

In case of two classes on a d dimensional dataset:

Be x a d-dimensional data vector. Having two classes A and B with posteriors $p(A|x)$ and $p(B|x)$ class A is chosen if $p(A|x) \geq p(B|x)$ and class B otherwise. As normal distributions are assumed for the likelihoods and the priors are constant factors, the following holds:

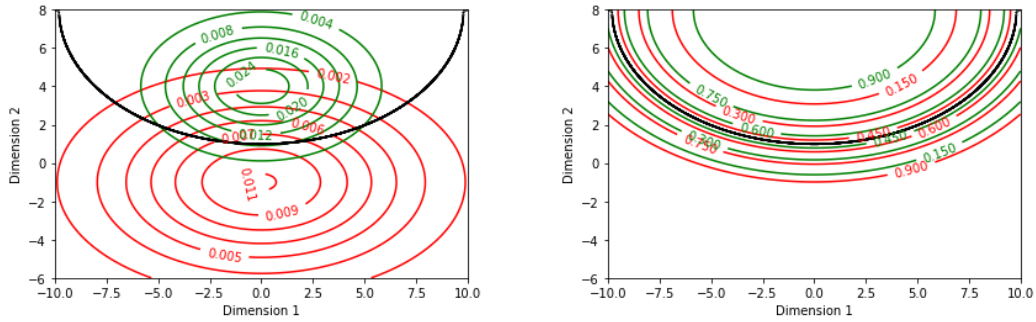
$$
\begin{aligned}
p(x|A) &= (2\pi)^{-\frac{d}{2}} \det(\Sigma_A)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_A)^T \Sigma_A^{-1}(x-\mu_A)} \\
p(x|B) &= (2\pi)^{-\frac{d}{2}} \det(\Sigma_B)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_B)^T \Sigma_B^{-1}(x-\mu_B)} \\
\ln(p(A)) &= l_A \\
\ln(p(B)) &= l_B
\end{aligned}
$$

with the μ being the means and the Σ the covariances. Therefore the decision boundary can be written as follows:

$$
\begin{aligned}
& p(A|x) \geq p(B|x) \\
\overset{\text{Bayes theorem}}{\Leftrightarrow} \quad & \frac{p(x|A)p(A)}{p(x)} \geq \frac{p(x|B)p(B)}{p(x)} \\
\Leftrightarrow \quad & p(x|A)p(A) \geq p(x|B)p(B) \\
\Leftrightarrow \quad & \ln(p(x|A)) + \ln(p(A)) \geq \ln(p(x|B)) + \ln(p(B)) \\
\Leftrightarrow \quad & -\frac{1}{2}\ln(\det(\Sigma_A)) - \frac{1}{2}x^T\Sigma_A^{-1}x + \mu_A^T\Sigma_A^{-1}x - \frac{1}{2}\mu_A^T\Sigma_A^{-1}\mu_A + l_A \\
& \geq -\frac{1}{2}\ln(\det(\Sigma_B)) - \frac{1}{2}x^T\Sigma_B^{-1}x + \mu_B^T\Sigma_B^{-1}x - \frac{1}{2}\mu_B^T\Sigma_B^{-1}\mu_B + l_B \\
\Leftrightarrow \quad & x^T(\Sigma_B^{-1} - \Sigma_A^{-1})x - 2(\mu_B^T\Sigma_B^{-1} - \mu_A^T\Sigma_A^{-1})x \\
& \geq \ln(\frac{\det(\Sigma_A)}{\det(\Sigma_B)}) - (\mu_B^T\Sigma_B^{-1}\mu_B - \mu_A^T\Sigma_A^{-1}\mu_A) + l_B - l_A
\end{aligned}
$$

As all terms containing x are on the left side and all others on the right, it can be seen that the decision boundary is quadratic in x. In a multi-class setting, each boundary between adjacent classes is quadratic, hence the overall decision boundary is piecewise quadratic. In case of naive Bayes, the covariance matrices are diagonal which does not affect the proof, however. An exemplary decision boundary is illustrated in Figure 9.

Figure 9: Visualization of a naive Bayes decision boundary



This figure illustrates a quadratic decision boundary that would arise using naive Bayes. Left the class likelihoods (green and red) are shown together with the decision boundary (black). Right the posterior distributions (again green and red) are shown together with the decision boundary. As can be seen, the black decision boundary forms a half circle (the rest of the circle is not shown). The green class has been set to having $\mu_{\text{green}} = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$ and $\Sigma_{\text{green}} = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$. The red class has $\mu_{\text{red}} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ and $\Sigma_{\text{red}} = \begin{pmatrix} 5 & 0 \\ 0 & 3 \end{pmatrix}$. Both priors have been set to 0.5.

7.2 Proof of the VC dimension of naive Bayes using a normal distribution

As shown in section 7.1, naive Bayes has a quadratic decision boundary in a 2-class setting. The proof will first be illustrated in two dimensions and then be generalized:

In two dimensions (x_1, x_2), a quadratic function can be mapped into the five dimensional space $(a_1, a_2, a_3, a_4, a_5) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$ in which it becomes a linear function. Its VC dimension is thus $5 + 1 = 6$ (see section 2.1).

In a general setting with d dimensions the quadratic function can be mapped into a $2d + \frac{d(d-1)}{2}$ dimensional space in which it becomes a linear function (The first d dimensions are the original dimensions, then add d squares, then $\frac{d(d-1)}{2}$ cross-products. The division by 2 is necessary for the last term as the ordering is not important). Therefore the VC dimension of naive Bayes using a normal distribution in a d dimensional space is $2d + \frac{d(d-1)}{2} + 1$

7.3 Proof of the lower bound of the VC dimension of naive Bayes using subclasses

Be m the VC dimension of the base naive Bayes estimator using a normal distribution (see section 7.2). The VC dimension is calculated in a 2-class setting, the classes will be named A and B. There exists a set of m data points which the base classifier can shatter. The subclass classifier will use n subclasses per original class. The procedure for the subclass classifier to shatter a set of nm points is as follows:

1. Take a set of m data points which can be shattered by the base classifier. Assign one subclass of each class to these m data points. These subclasses behave like a normal naive Bayes classifier for these m data points and can thus separate any possible 2-class problem on this set.

2. Copy the first m data points (or draw another equivalent set) and move them far away from the first one. Assign another subclass of each class to this new set.

3. Repeat these steps for all subclasses, the result are n separated sets of m data points.

For each of the n subsets the assigned subclasses can separate any possible 2-class problem on this set. Furthermore, the separation of the sets is trivial, therefore all 2-class problems on the nm data points constructed this way can be separated so this set can be shattered by naive Bayes using n subclasses. The VC dimension of naive Bayes using n subclasses is therefore at least nm.

7.4 Proof of the VC dimension of a decision tree

A decision tree has theoretical infinite VC dimension. This can be shown by the following tree building procedure, which can shatter a set of any number n of vectors of any dimension d:

1. Draw n data points and arange them on one of the base axes.

2. Build a tree with each data point as a leaf node. This can easily be done by placing a split between any two data points.

For every possible 2-class problem, the decision tree can now assign the classes at the leaf nodes accordingly and thus perfectly separate the classes.

7.5 Proof of the VC dimension of a random forest

This is similar to section 7.4. The random forest classifier uses q decision trees on a subset of attributes A and a subset of points P. As control over the building process of the classifier is assumed here, one can just build q trees according to the procedure laid out in section 7.4 and choose A and P accordingly. Therefore for any number n there exits a set of n points which random forest can shatter by using this procedure. Hence the classifier has infinite VC dimension.

Förmliche Erklärung

Hiermit versichere ich, Nathan Valenti, die vorliegende Studienarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 19.06.2019

Nathan Valenti

(Unterschrift)