

Master's Thesis

Random Subspace Method for One-Class Classifiers

Thesis Committee:

Prof. dr. ir. M.J.T. Reinders

Dr. D.M.J. Tax

T.A. van der Donk, MSc

Dr. ir. R.P.W. Duin

Prof. dr. drs. L.J.M. Rothkrantz

Author	Veronika Cheplygina
Email	v.cheplygina@gmail.com
Student number	1217666
Thesis supervisor	Dr. D.M.J. Tax
Date	November 15, 2010



Preface

This thesis is submitted in partial fulfillment of requirements for the Master of Science degree in Media and Knowledge Engineering at the Delft University of Technology. It contains the results of my graduation project done at the Pattern Recognition Lab in collaboration with Prime Vision from March to November 2010. This document consists of a research article which highlights the most important findings of my work, and a final report which includes all the performed research.

I would like to thank my supervisors David Tax and Theo van der Donk for all their help during this project, and the Pattern Recognition Lab and Prime Vision for making this project possible. A thank you is also in order to the people of the Pattern Recognition Lab who have sparked my interest in pattern recognition and machine learning during the last years of my study.

Delft is not only a place to study. I am grateful for all the things I did next to my education: a board year and several committees at CH, the study visit to the United States and playing soccer at Ariston'80. I have learned a lot from these activities and made many friends. A thank you goes out to all these friends who have made my time here an awesome experience: 51, Wocky!, my soccer team, people from my OWee group whom I've stayed in touch with and many others. Last but not least, I would like to thank my parents and Lennert for supporting me during my whole study and especially during the last stressful days before graduation.

Veronika Cheplygina
November 7, 2010

Random Subspace Method for One-Class Classifiers

Veronika Cheplygina

Pattern Recognition Lab, Delft University of Technology

Abstract. The goal of one-class classification is to distinguish the target class from all the other classes using only training data of the target class. It may be difficult for a single one-class classifier to capture all the characteristics of the target class, which may result in misclassifications of new objects. In order to create more accurate classifiers, combining several one-class classifiers is possible. Previous research has shown that the Random Subspace Method, in which classifiers are trained on different subsets of the feature space, can be effective for one-class classifiers. However, there are indications that it is not straightforward to achieve improved performance with this method. In ensembles of traditional classifiers, it has been shown that pruning inaccurate classifiers can be more effective than using all the classifiers. We propose to apply pruning to RSM of one-class classifiers. Results from the performed experiments show that the pruned ensemble has better and more stable performance than the complete ensemble. A comparison of the proposed ensemble with other ensemble techniques also indicates very competitive performance.

1 Introduction

The goal of one-class classification is to create a description of one class of objects (called target objects) and distinguish this class of objects from all objects, not belong to this class (called outliers) [19]. One-class classification is particularly suitable for situations where one of the classes is not represented well in the training set. This is common in applications where examples of one class are more difficult to obtain, such as detecting credit card fraud, intrusions or a rare disease [5].

For any single classifier, it might be difficult to create a perfect model of the data using some training examples. A poor model may lead to classifying new objects incorrectly. Combining several classifiers might create a better description and thus improve classification performance [8]. This is also the case for one-class classifiers (OCCs), as there have been indications that combining OCCs may reduce the classification error [19, 18].

A popular approach of creating an ensemble of classifiers is to combine the same base classifier, trained on different versions of the training data, as in Bagging [3] or Random Subspace Method (RSM) [11]. Although these methods were originally designed for traditional classifiers, in recent years a number of

attempts have been made to apply RSM to OCCs [13, 14, 2]. However, the variety of classifiers and parameters in these works suggests it is difficult to improve performance of OCCs with RSM. We perform a series of preliminary experiments to support this statement.

Although it was originally believed that larger ensembles are more effective, it has been demonstrated that using a subset of classifiers might be better than the using the whole set [23]. Pruning inaccurate classifiers in RSM has been shown to be effective for traditional classifiers [4], however, to our knowledge, this method has not been examined for OCCs. We investigate whether pruning is effective for RSM when applied to OCCs.

In section 2 more information about OCCs and RSM is provided. Our proposed approach is described in section 3. The experimental setup is described in section 4, followed by results and analysis in section 5. Conclusion and suggestions for further research are presented in section 6.

2 Related Work

2.1 One-Class Classification

Supervised classification consists of approximating the true function $y = f(\mathbf{x})$ for some object-label pairs $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ with a hypothesis h (i.e. training a classifier), and using h to assign y values to previously unseen objects $\mathbf{x}$. Let us assume there are two classes, i.e. $y_i \in \{+1, -1\}$. A traditional classifier needs labeled objects of both classes in order to create h , and its performance will suffer if one of the classes is absent from the training data. If one of the classes is not present in the training data, h will not reflect the “true” situation, and new objects of this class might be classified incorrectly. A one-class classifier, however, only needs objects of one class to create h . It is thus trained to accept objects of one class (target class) and reject objects of the other (outlier class).

A one-class classifier consists of two elements: the “similarity” of an object to the target class, expressed as a posterior probability $p(\omega_t|\mathbf{x})$ or a distance (from which $p(\omega_t|\mathbf{x})$ can be estimated), and a threshold θ on this measure, which is used to determine whether a new object belongs to the target class or not. In [19] several strategies of deriving these elements are described. In this work we choose two classifiers, each following a different strategy: the Gaussian (Gauss), Nearest Neighbor (1-NN) and k -Means one-class classifiers. Gauss is a *density* method which fits a normal distribution to the data and rejects objects on the tails of this distribution. 1-NN is a *boundary* method which uses distances between objects to calculate the local densities of objects, and rejects new objects with a local density lower than that of its nearest neighbor. k -Means is a *reconstruction* method which assumes that the data is clustered in k groups, finds k prototype objects for these groups, and creates a boundary out of the hyperspheres placed at these objects.

2.2 Ensembles of (One-Class) Classifiers

For a single classifier it may be difficult to find a good approximation of f because of limited training data or the properties of a particular classifier. By combining several hypotheses in an ensemble, it may be possible to find a better approximation of f , thus creating a more accurate classifier [8, 22]. It has been demonstrated that combining classifiers can also be effective for one-class classifiers [19, 18].

In order to produce an ensemble classifier, it is necessary to combine the outputs of several classifiers into a single output. Several methodologies (combining methods M) for doing so exist [12]. Here we focus on *averaging* or *mean* of the posterior probabilities, i.e. $f_{avg}(\mathbf{x}) = \frac{1}{L} \sum_{i=1}^L p_i(\omega_t|\mathbf{x})$.

There is no point in combining the same classifier, so a set of sufficiently different classifiers is needed. This set can be obtained by using different base classifiers or by training the classifiers on different data, such as in Bagging [3] or RSM [11]. An extensive survey can be found in [17]. Here we only examine RSM in more detail.

RSM In RSM [11] or feature bagging [13], each classifier is trained on a random sample of a fixed size, such as $d/2$, of the d features, i.e. the samples contain N distinct objects with $d/2$ features. RSM is expected to benefit in problems suffering from the *curse of dimensionality* because of the improved object/feature ratio for each classifier. RSM is successfully applied to a range of one-class classifiers in [13, 14, 2].

Pruning ensembles After the “many-could-be-better-than-all” principle has been demonstrated in [23], many ensembles combining only a subset of the available classifiers have been proposed [17]. The most simple approach is to evaluate L classifiers individually according to a specified criterion (such as accuracy on the training set) and select the L_s best classifiers (i.e. prune the inaccurate classifiers). This approach has been successfully applied to RSM in [4]. In our research, we have not encountered any works where pruning has been applied to ensembles of one-class classifiers.

3 Motivation and Proposed Approach

As described in the previous section, there are indications that RSM of one-class classifiers may be successful. This seems intuitive: in a problem with few objects in two classes (with objects in each class), a one-class classifier has access to even less training data than a two-class classifier. However, the variety of classifiers, parameters and algorithm modifications in the literature suggests that it is not straightforward to achieve better performance with this method.

This is also illustrated in Table 1, where we show results of RSM for two datasets (which are described in the next section), while varying the classifier, the number of features d_s , the number of subspaces L and the combining method M .

From the amount of bold results in Table 1 it is clear that RSM is able to significantly outperform the base classifiers, but this improvement is quite unpredictable, i.e. it is difficult to say what settings need to be chosen to achieve the best performance per dataset. A good number of features differs across the datasets (25% for Imports, 50% for Concordia) and is also influenced by other parameters. It is difficult to say that it is beneficial to use more subspaces, as mean can produce good results even with 5 subspaces. On the other hand, there does seem to be a trend in the combining methods: mean produces better results than voting in most situations. However, a disadvantage of mean is that it is unpredictable as its performance goes up and down with number of subspaces.

Table 1. AUC performances ($\times 100$) of RSM with different parameters. Next to the classifier names, we report the default performances of the base classifier on the two datasets. Bold indicates results significantly better than the base classifier ($\alpha = 0.05$), italics indicates results significantly worse than the base classifier.

Classifier	$d_s, M / L$	Concordia dataset					Imports dataset				
		5	10	20	50	100	5	10	20	50	100
Gauss	25% mean	<i>90.6</i>	<i>91.5</i>	94.8	94.4	94.3	80.5	78.1	74.8	<i>71.2</i>	<i>68.6</i>
	93.7/74.1 25% vote	<i>88.1</i>	<i>90.1</i>	94.0	93.7	93.8	80.7	79.2	77.0	75.6	73.9
	50% mean	<i>93.2</i>	93.9	94.8	94.8	94.7	<i>69.0</i>	<i>70.7</i>	<i>70.5</i>	<i>69.5</i>	<i>68.2</i>
	50% vote	<i>91.4</i>	<i>93.2</i>	94.4	94.9	94.8	72.3	73.5	74.5	73.7	73.1
	75% mean	93.9	94.1	94.6	94.4	94.4	<i>70.8</i>	<i>70.7</i>	<i>70.9</i>	<i>70.6</i>	<i>71.7</i>
	75% vote	<i>78.5</i>	<i>81.5</i>	<i>83.9</i>	<i>86.4</i>	<i>88.0</i>	<i>71.2</i>	<i>71.8</i>	72.6	73.2	74.0
1-NN	25% mean	<i>90.1</i>	<i>91.7</i>	93.8	<i>93.3</i>	<i>93.4</i>	90.6	90.7	89.8	87.9	87.5
	93.9/87.4 25% vote	<i>87.0</i>	<i>90.0</i>	<i>93.5</i>	<i>93.3</i>	<i>93.6</i>	86.8	86.2	86.8	<i>85.8</i>	<i>84.8</i>
	50% mean	<i>92.5</i>	<i>93.6</i>	94.1	94.0	93.9	87.6	88.2	88.2	87.2	87.2
	50% vote	<i>89.1</i>	<i>91.8</i>	<i>93.4</i>	<i>93.6</i>	<i>93.6</i>	<i>78.7</i>	<i>79.7</i>	<i>81.1</i>	<i>82.1</i>	<i>82.8</i>
	75% mean	<i>92.6</i>	<i>93.4</i>	93.9	93.9	93.9	88.5	87.8	87.9	88.0	88.0
	75% vote	<i>86.9</i>	<i>90.0</i>	<i>91.6</i>	<i>92.3</i>	<i>92.7</i>	<i>72.4</i>	<i>73.1</i>	<i>76.6</i>	<i>78.2</i>	<i>79.1</i>
k -Means	25% mean	<i>89.0</i>	<i>90.5</i>	92.7	92.0	92.0	83.2	79.7	76.5	73.0	72.2
	91.5/66.2 25% vote	<i>83.0</i>	<i>87.0</i>	91.9	91.7	91.7	79.6	77.6	77.1	74.2	73.7
	50% mean	<i>91.0</i>	92.3	92.6	92.3	92.2	69.2	69.6	68.7	68.0	68.0
	50% vote	<i>84.4</i>	<i>88.0</i>	<i>90.0</i>	<i>90.4</i>	<i>90.6</i>	<i>59.6</i>	64.3	66.1	66.4	66.5
	75% mean	<i>91.0</i>	91.7	92.1	92.1	92.1	66.8	66.8	67.8	66.9	67.0
	75% vote	<i>82.0</i>	<i>84.9</i>	<i>87.2</i>	<i>88.2</i>	<i>88.8</i>	<i>60.5</i>	65.0	65.2	<i>63.8</i>	<i>63.5</i>

For a classifier to be succesful it should have good performance *and* be predictable in terms of how its parameters may influence its performance: it should be possible to choose parameters based on knowledge of the data and classifier, rather than trying all possible parameter combinations, as in Table 1. Based on our observations, the only parameter we could choose with some certainty is mean (although other combining methods might be more effective), thus still leaving two “difficult” parameters. However, if the performance of mean was

more stable, it would be possible to also choose a good number of subspaces, leaving only the one “difficult” parameter.

We suspect that RSM produces subspace classifiers of different quality. Averaging may be sensitive to such variations [16], therefore causing the performance of the whole ensemble to go up and down. By excluding the less accurate classifiers, the performance of the ensemble should stabilize. In this case, we would be able to choose a reasonable number of subspaces and therefore create a “better” classifier (with the same performance, but less parameters).

We propose PRSM, a pruned version of RSM¹, and apply it to several OCCs. The implementation of PRSM is shown in Table 2. We use AUC performance as an evaluation measure for the method to be applicable to OCCs. However, this evaluation measure requires the validation set to contain some outliers, i.e. PRSM is not strictly a one-class method. Because this information is used, the performance estimates for PRSM might be more optimistic than if no outlier examples were available. However, PRSM-OCC could theoretically be equipped with a method for generating artificial outliers in order to be applicable even these types of problems.

Using a validation set has another implication: there may be too little data to use a separate validation set. Another possibility is to evaluate using the training set as in [4], i.e. $x_v = x_t$. However, this is not possible for classifiers which have 100% performance on the training set, such as 1-NN. In our implementation, we use 90% of the training set for training and the remaining 10% for validation. After ranking the classifiers, we retrain the classifiers using the whole training set.

Table 2. PRSM Algorithm

Input: Training set x_t , validation set x_v , base classifier w , number of classifiers L , number of selected classifiers L_s , subspace dimensionality d , combining method M
Output: Ensemble E

```

for  $i = 1$  to  $L$  do
   $subspace(i) \leftarrow select(x_t, d)$ 
   $clasf(i) \leftarrow train(subspace(i), w)$ 
  if  $L_s < L$  then
     $score(i) \leftarrow test(x_v, clasf(i))$ 
  end if
end for
 $clasf \leftarrow sortByScore(clasf, score)$ 
return  $E \leftarrow combine(clasf(1)...clasf(L_s), M)$ 

```

¹ Note that when no pruning is necessary ($L_s = L$), this classifier reduces to RSM-OCC.

4 Experimental Setup

All experiments are performed in Matlab using PRTools [9] and the Data Description toolbox [20]. We use 16 datasets from [21] which originate from the UCI Machine Learning Repository [1], but have been modified in order to contain a target and an outlier class. The PV dataset is provided by Prime Vision B.V. in Delft, The Netherlands. A summary of the datasets is included in Table 3. We included datasets with both high and low object-to-feature ratios, i.e. the choice of datasets is not biased for using subspaces.

Table 3. List of datasets

Dataset	Targets	Outliers	Features	Dataset	Targets	Outliers	Features
Cancer	47	151	33	PV	1425	575	72
Concordia	400	3600	256	Sonar	111	97	60
Diabetes	500	268	8	Spambase	1813	2788	57
Glass	144	70	9	Spectf	95	254	44
Heart	164	139	13	Vehicle	212	634	18
Hepatitis	123	32	19	Vowel	48	480	10
Housing	48	458	13	Wave	300	600	21
Imports	71	88	25	Wine	71	107	13
Ionosphere	225	126	34				

All experiments are performed using 5x stratified 10-fold cross-validation. The results are reported as the average AUC and its standard deviation (both $\times 100$) over the 50 runs. Due to space considerations, in this paper we present only the average AUC.

RSM vs. PRSM We investigate whether PRSM is at least as accurate than RSM, but more stable in the number of subspaces, while keeping other parameters constant. A separate comparison is performed for every combination of dataset and base classifier, therefore only a selection of datasets is used for this experiment. For each dataset, we choose the best case scenario for RSM: mean and a reasonable number (25% or 50%) of features and compare the performances of RSM with L_s classifiers to PRSM with L_s classifiers out of $L = 100$.

Classifier Comparison Furthermore, we compare PRSM to other classifiers across a range of datasets. We use the base classifier and RSM, Bagging and AdaBoost² ensembles of the same base classifier. A separate comparison is performed for each base classifier. For the comparison, we use the Friedman test [10] and the post-hoc Nemenyi test [15], as recommended in [7].

² Similarly to PRSM, AdaBoost is not strictly a one-class method

5 Experimental Results

RSM vs. PRSM A selection of the results are presented in Fig. 1. The additional results can be found in [6]. Although RSM is sometimes able to produce slightly better results than PRSM, PRSM performs better overall. Furthermore, PRSM tends to produce more regular, “convex” graphs than RSM. Due to this property, it becomes easier to pick a good number of subspaces. Selecting 10 out of 100 subspaces has good results for all the datasets we use here.

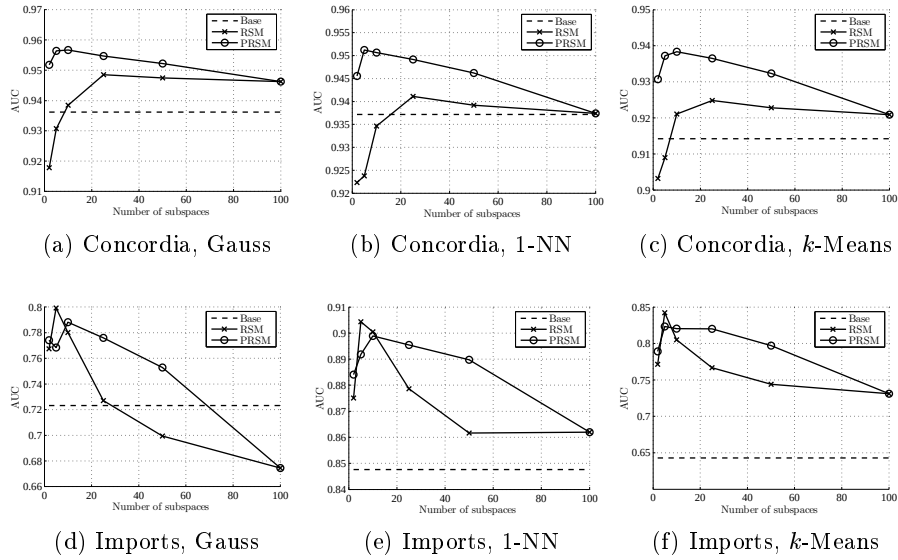


Fig. 1. PRSM vs. RSM

Comparison Based on the previous results, we use 10 out of 100 subspaces for PRSM and 100 subspaces for RSM. For both classifiers, we use the mean combining rule and a variable feature size (in the hope that this is more adequate than a particular fixed value). For Bagging (Bag) and AdaBoost (ABv), we use the implementations from PRTools [9] with default parameters. Due to our intuitions about the combining methods, we also add AdaBoost with mean (ABm) to the comparison.

The results of the comparison are shown in Table 4. A key observation is that overall, the classifiers have competitive performance, however there is no overall winning classifier. For some datasets, such as Sonar, it is even best to just use the base classifier as no ensemble is able to improve on this performance. This is counterintuitive, because with its low object/feature ratio, we would expect Sonar to at least benefit from using subspaces.

The results of using the variable feature size did not have the effect we had hoped for, as a number of datasets suffer from this setting. For instance, for the Concordia dataset we have seen that the performance of PRSM with a fixed feature size is at least that of the base classifier, while here this is not the case for the Gaussian classifier. This suggests that there is no universal setting for this parameter, and that the only solution is to find a fitting value for every dataset, either empirically or by using knowledge about the dataset.

Another interesting observation is that ABm often significantly outperforms ABv. Although ABv is sometimes the better of the pair, this difference is usually quite modest, which suggests that mean might more effective than vote for ensembles of one-class classifiers in general, not just for RSM.

Table 4. AUC performances (x100) of the base classifier and ensemble methods for the Gauss, 1-NN and k -Means base classifiers.

	Base	RS	PRS	Bag	ABv	ABm		Base	RS	PRS	Bag	ABv	ABm
Canc.	60.8	69.4	65.1	68.7	64.7	68.0	PV	93.9	92.7	93.5	93.2	86.6	93.8
Conc.	93.8	92.3	93.2	91.1	80.4	92.1	Sonar	70.3	66.6	69.4	68.0	69.6	69.2
Diab.	70.3	70.9	71.7	69.8	57.0	69.9	Spam.	63.3	78.4	82.3	69.9	50.0	69.9
Glass	78.3	78.4	78.1	76.5	76.2	75.9	Spec.	92.7	88.3	87.2	88.0	87.5	89.0
Heart	76.0	69.2	66.0	63.4	58.8	62.7	Veh.	71.6	69.0	73.3	71.8	64.3	71.5
Hep.	64.4	75.9	81.1	64.3	53.0	61.1	Vowel	99.2	93.8	94.4	97.0	97.5	97.1
Hous.	80.2	87.0	88.3	88.8	87.3	86.9	Wave	89.1	92.4	91.9	90.6	87.0	90.3
Imp.	72.5	82.9	82.4	83.6	86.4	84.2	Wine	83.1	86.6	90.5	88.0	86.2	82.8
Ion.	96.4	93.0	90.1	94.8	92.9	94.1							
Canc.	68.7	69.0	66.3	70.0	49.7	51.4	PV	95.5	95.3	95.8	95.7	92.6	96.5
Conc.	93.9	94.0	95.2	94.0	90.7	94.4	Sonar	84.7	82.6	83.6	83.4	80.9	82.4
Diab.	68.2	70.6	71.7	68.6	66.5	67.3	Spam.	75.8	76.5	80.1	73.2	50.0	60.7
Glass	86.5	85.9	85.3	86.1	80.7	87.0	Spec.	95.2	95.4	95.3	86.1	88.1	96.1
Heart	59.6	52.9	67.3	60.7	61.7	62.6	Veh.	64.2	72.1	76.4	64.6	64.7	65.0
Hep.	65.6	50.0	72.6	66.3	56.9	56.7	Vowel	99.6	99.6	99.6	99.4	99.6	98.8
Hous.	83.4	90.1	91.8	83.1	81.9	82.0	Wave	90.6	91.7	91.2	90.9	86.2	88.9
Imp.	87.4	86.4	87.8	84.0	78.0	81.3	Wine	82.9	90.4	89.1	84.5	80.9	82.6
Ion.	95.9	96.2	96.7	96.2	90.8	94.7							
Canc.	50.4	64.2	66.3	71.8	68.9	71.7	PV	92.4	92.9	93.7	92.9	86.7	92.7
Conc.	91.7	92.0	93.3	92.1	85.6	91.9	Sonar	71.4	75.6	75.1	74.1	72.1	74.8
Diab.	65.7	66.4	67.3	66.5	62.6	66.5	Spam.	48.6	65.3	72.7	50.8	50.0	51.0
Glass	79.6	80.8	81.4	80.9	79.1	79.7	Spec.	86.0	89.3	88.5	88.7	86.9	90.2
Heart	59.0	68.9	68.1	62.2	61.6	61.8	Veh.	57.4	68.6	69.1	68.4	52.5	68.1
Hepat.	53.9	42.7	72.9	56.7	51.8	57.5	Vowel	96.4	99.1	98.3	98.5	98.9	98.7
Hous.	66.3	92.3	91.5	89.5	90.0	90.0	Wave	89.1	91.4	91.2	90.6	87.9	90.6
Imp.	66.7	88.7	86.5	88.6	86.8	86.6	Wine	75.1	88.1	90.9	92.8	91.7	86.6
Ion.	96.6	90.6	89.9	94.3	93.0	92.7							

The results of the statistical tests are shown in Table 5. For each comparison, the F statistic value indicates there are significant differences between the classifiers. In each comparison, PRSM is ranked first, however, RSM and Bagging are close competitors as their ranks are very similar for Gauss and k -Means classifiers. Using the post-hoc test, PRSM is found to be significantly better than ABv for Gauss, ABv and ABm for 1-NN and ABv and the base classifier for k -Means.

Table 5. Ranks and F statistics of the Friedman test. The critical value (CV) for the Friedman test for 17 datasets and 6 classifiers is 2.33. The critical difference for the Nemenyi test is 1.83.

	Base	RS	PRS	Bag	ABv	ABm	F	Significant?
Gauss	3.29	2.94	2.65	3.47	4.82	3.82	3.23	Yes, $F > CV$
1-NN	3.47	3.06	1.76	3.41	5.41	3.88	10.71	Yes, $F > CV$
k -Means	5.24	2.65	2.47	2.76	4.59	3.29	9.61	Yes, $F > CV$

6 Conclusions

In this paper we investigated the effect of pruning on RSM ensembles on the Gaussian, Nearest Neighbor and k -Means one-class classifiers and compared this classifier to RSM and other popular ensemble techniques. Reported experiments have shown that PRSM has comparable, but more stable performance than RSM, and that PRSM has very competitive performance with other ensembles of the examined one-class classifiers. These results suggest that combining a few accurate classifiers may be more beneficial than combining all available classifiers. Furthermore, the more stable performance of PRSM reduces the number of parameters that need to be set for RSM, making it a more applicable classifier. Unfortunately, the effect of the subspace dimensionality is still unclear and requires further investigation. Other issues that could be explored in the future are turning PRSM into a strictly one-class method or the pruning of other ensembles such as Bagging.

References

1. Asuncion, A. and Newman, D.J.: UCI Machine Learning Repository (2007), <http://www.ics.uci.edu/~mllearn/MLRepository.html>
2. Biggio, B., Fumera, G., Roli, F.: Multiple classifier systems under attack. Multiple Classifier Systems pp. 74–83 (2010)
3. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
4. Bryll, R., Gutierrez-Osuna, R., Quek, F.: Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition 36(6), 1291–1302 (2003)

5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41(3), 1–58 (2009)
6. Cheplygina, V.: Random subspace method for one-class classifiers. Master's thesis, Delft University of Technology (2010)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7, 1–30 (2006)
8. Dietterich, T.: Ensemble methods in machine learning. *Multiple Classifier Systems* pp. 1–15 (2000)
9. Duin, R.: PRtools (2009), version 4.1.5
10. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association* 32(200), 675–701 (1937)
11. Ho, T.: The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8), 832–844 (1998)
12. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(3), 226–239 (2002)
13. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp. 157–166. ACM (2005)
14. Nanni, L.: Experimental comparison of one-class classifiers for online signature verification. *Neurocomputing* 69(7-9), 869–873 (2006)
15. Nemenyi, P.: Distribution-free multiple comparisons. Ph.D. thesis, Princeton (1963)
16. Oza, N., Tumer, K.: Classifier ensembles: Select real-world applications. *Information Fusion* 9(1), 4–20 (2008)
17. Rokach, L.: Ensemble-based classifiers. *Artificial Intelligence Review* 33(1), 1–39 (2010)
18. Tax, D., Duin, R.: Combining one-class classifiers. *Multiple Classifier Systems* pp. 299–308 (2001)
19. Tax, D.: One-class classification; Concept-learning in the absence of counter-examples. Ph.D. thesis, Delft University of Technology (June 2001)
20. Tax, D.: DDtools, the Data Description toolbox for Matlab (2010), version 1.7.4
21. Tax, D.: OC classifier results (2010), <http://homepage.tudelft.nl/n9d04/occ/index.html>
22. Valentini, G., Masulli, F.: Ensembles of learning machines. *Neural Nets* pp. 3–20 (2002)
23. Zhou, Z., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. *Artificial Intelligence* 137(1-2), 239–263 (2002)

Master's Thesis

Random Subspace Method for One-Class Classifiers

Veronika Cheplygina

November 15, 2010

Notation

N	Number of objects in a dataset
d	Dimensionality of a dataset
C	Number of classes
$\mathbf{x}$	Object, represented by a vector $\{x_1, \dots, x_d\}$, often an object from the training set.
$\mathbf{z}$	Object, represented by a vector $\{z_1, \dots, z_d\}$, often an object from the test set.
y_i	Label for object $\mathbf{x}_i$
$\mathcal{H}$	Space of hypotheses
f	True hypothesis for the data
h	Approximation of f by a classifier
$h(\mathbf{x})$	Output of classifier for input $\mathbf{x}$
$p(\omega_t \mathbf{x})$	Posterior probability of $\mathbf{x}$ belonging to the target class
E	Ensemble classifier
L	Number of classifiers in an ensemble

Contents

1	Introduction	1
1.1	Problem Description	1
1.2	Proposed Solution	3
1.3	Research Questions	3
1.4	Outline	4
2	Related Research	5
2.1	One-class Classification	5
2.1.1	Basic classification problem	5
2.1.2	One-class vs two-class	5
2.1.3	Properties	6
2.1.4	Approaches to one-class classification	8
2.2	Ensembles of Classifiers	8
2.2.1	Motivation	9
2.2.2	Ensemble architecture	10
2.2.3	Ensemble design	12
2.3	Random Subspace Method	13
2.3.1	Motivation	13
2.3.2	Implementation	13
2.3.3	Applications to one-class problems	14
3	One-class Classification for Prime Vision	17
3.1	Introduction	17

3.2	Data	17
3.2.1	Objects	17
3.2.2	Features	20
3.3	Experimental Setup	21
3.4	Results	23
3.4.1	Feature set	23
3.4.2	Generalization	25
3.4.3	Error analysis	25
3.5	Conclusion	28
4	Random Subspace Method	29
4.1	Data	29
4.2	Classifiers	30
4.3	Experimental Setup	31
4.3.1	Evaluation	32
4.4	Results	33
4.4.1	Parameters RSM	33
4.4.2	Parameters PRSM	38
4.5	Conclusion	40
5	Further Analysis	42
5.1	Data-dependency	42
5.1.1	Experimental setup	44
5.1.2	Results	45
5.2	Diversity	47
5.3	Comparison	54
5.3.1	Experimental setup	54
5.3.2	Results	56
6	Conclusions	60
6.1	Discussion	60

6.1.1	Prime Vision data	60
6.1.2	Data-dependency	60
6.1.3	Parameter influence	61
6.1.4	Diversity	61
6.1.5	Other ensemble techniques	61
6.1.6	Evaluation	62
6.2	Future Research	62
6.2.1	Prime Vision data	62
6.2.2	Data dependency	62
6.2.3	Parameter influence	62
6.2.4	Other ensemble techniques	63
6.3	Main Conclusions	63
A	Results on Additional Data	68
A.1	RSM	69
A.2	PRSM	81
A.2.1	RSM vs PRSM	81
A.2.2	Pool size	82

Chapter 1

Introduction

Prime Vision is one of the leading companies in the field of optical character recognition (OCR) techniques. Prime Vision's areas of expertise include address recognition, processing of financial documents, identification of vehicles, prevention of identity fraud and others. One of Prime Vision's products is an automated parcel sorting solution, which is currently in use at TNT Post, the biggest postal company in The Netherlands, but also in other countries.

With over 100,000 parcels being delivered every night to one of the postal sorting centres in the Netherlands, creating a system to process all the parcels correctly and on time is a challenge. Prime Vision's parcel sorting solution is already delivering excellent results at the sorting centre as 98.5% of the parcels get delivered the next day [1]. However, efforts are still being directed into further improving the system, as an improvement of just 0.1% would mean an improvement of 100 parcels daily, or 36,500 parcels yearly. And these are just the numbers for one of the three sorting centres in The Netherlands!

Address recognition and thus sorting of parcels has several difficulties due to the varying layout of the parcels [2]. For instance, the address might be written elsewhere on the parcel instead of the designated address label or might be (partially) obscured by a company logo, causing difficulties for OCR. Addressing such issues is a necessary step in order to further improve the performance of the system.

1.1 Problem Description

The current Prime Vision system functions as follows. The input of the system are binary images of letters or parcels. These images are first processed by the clustering algorithm, which identifies meaningful clusters, such as address blocks, within each image. Each of the clusters is then labeled by the classifier as a typed text or a written text cluster. This label is needed to start the correct OCR process, as typed text and written text have different requirements in order to be recognized. The output of the OCR step is then analyzed using country-specific rules to determine whether it contains a valid address. After this step, valid addresses can be used to sort the letters or parcels according to their destination. Mail with invalid addresses will have to be sorted manually.

A problem of this system is that it is assumed that the output of the clustering algorithm always

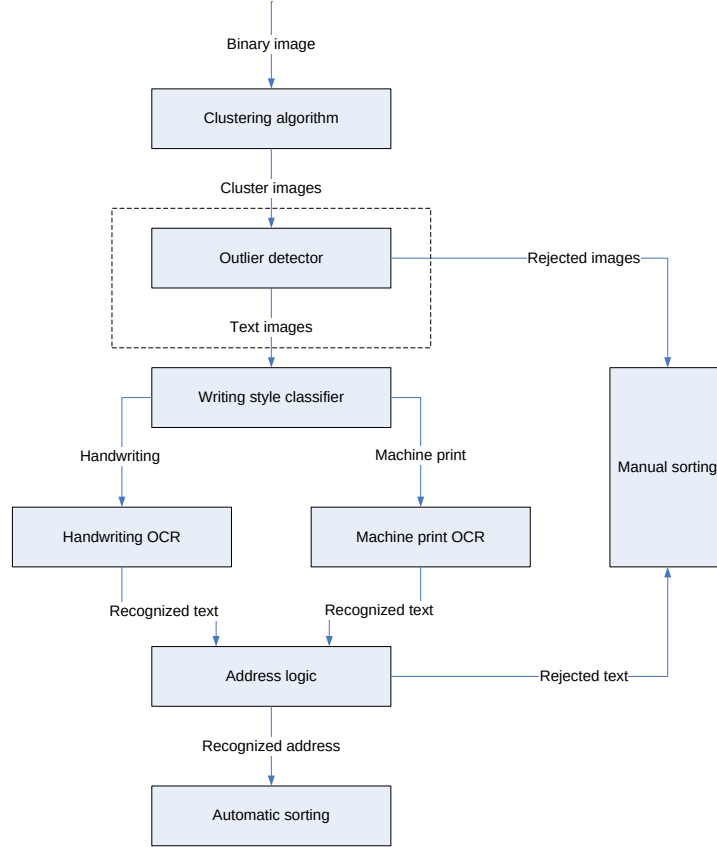


Figure 1.1: Current Prime Vision system and (dotted line) proposed component to be implemented.

produces meaningful text clusters which contain either typed or written text. Unfortunately, this is not always true in practice and clustering algorithm also outputs clusters which do not contain text, such as pictures, stamps, barcodes and so forth. For instance, a barcode which is input to the OCR could produce output such as “1111II” which could be a postal code in The Netherlands. If this is a valid postcode, the letter could get delivered to the wrong address. If this postal code is invalid, the letter would be sorted manually and arrive to the correct destination.

This is not a desirable situation, because time is wasted while trying to process “meaningless” clusters. In order to avoid this, it would be beneficial to detect whether there is any text to be processed *before* OCR is applied. If no text is found, the letter would be processed manually straight away. There are two modifications to the current systems which could enable such functionality:

- Modify the classifier so it will output three labels: “typed text”, “written text” and “non-text”.
- Insert a new classifier which will output two labels: “text” (both “typed” and “written”) and “non-text”. Items labeled as “text” can then be processed by the existing classifier. This approach is illustrated in Fig. 1.1.

1.2 Proposed Solution

The approach of implementing a new classifier is more interesting to Prime Vision and Delft University of Technology both practically and academically. On the practical side, the current Prime Vision classifier is already optimized for two classes, so its performance could decrease if a new class is introduced. By inserting an outlier detector before the classifier, the current classifier would remain intact. Furthermore, the classifier code is copyright protected so it would be more difficult to use it in research and publish results.

On the academic side, introducing a classifier with outputs which can be interpreted as “target” (for text) and “outlier” (for non-text) offers an opportunity to replace the classifier by a so called one-class classifier or outlier detector. The problem of outlier detection is less explored than the problem of classification. At Delft University of Technology, outlier detection (or one-class classification) is an ongoing research topic, whereas at Prime Vision, only traditional classification is used. Therefore, researching the applications of outlier detection to this problem has potential to provide a significant contribution to the understanding of the subject at both institutions.

There are several existing one-class classifiers or even classification toolboxes available today. Simply evaluating all available classifiers and selecting the best one would not be a very interesting research direction. A more interesting solution is to implement a new classifier which uses the existing classifiers as building blocks. This approach is called an ensemble of classifiers, also an on-going research direction. One of the methods which has received attention because of its ability to improve on the performance of the individual classifier is called the Random Subspace Method (RSM). While there has been quite a lot of research on RSM for traditional classifiers, only few works have been dedicated to applying RSM to one-class classifiers.

1.3 Research Questions

Our choice of solution forms two main research questions for this project: ***“Are one-class classifiers suitable for the Prime Vision problem?”*** and ***“How does the Random Subspace Method affect the performance of one-class classifiers?”***.

The first question is more specific to Prime Vision and their data and only examines whether one-class classifiers are a suitable tool for classifying text and non-text images. In order to answer this question, we need to examine several sub-questions, such as *“Is the available data suitable for classifying text and non-text clusters?”*, *“What is the best case scenario for classifying the available data?”* and finally *“Are one-class classifiers better than conventional classifiers for this problem?”*.

The second question is more general. It aims to investigate whether RSM can influence the performance of one-class classifiers on Prime Vision data, but also whether this is also true for other datasets. Furthermore, especially if the effect of RSM varies across different datasets, we wish to investigate why this is the case.

1.4 Outline

The outline of this thesis is as follows. The next chapter presents an overview of theoretical background in one-class classification and ensemble learning. In particular, the random subspace method as an ensemble method will be covered.

The following two chapters describe the experiments and analysis related to one-class classification and the random subspace method respectively. These chapters can be read as independent investigations, although conclusions from chapter 3 are relevant for chapter 4. Chapter 5 provides additional experiments to aid the understanding of results from chapter 4.

Finally, chapter 6 presents a discussion of the results and propositions for further research.

Chapter 2

Related Research

2.1 One-class Classification

2.1.1 Basic classification problem

A standard classification problem consists assigning class labels to a set of inputs. In order to do so, a classifier needs to construct a model of the data using some examples, i.e. the classifier needs to be trained. Using this model, the trained classifier will be able to assign class labels to new, unlabelled examples.

More formally, classification consists of approximating the function $y = f(\mathbf{x})$ for some inputs $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and some class labels $\{y_1, \dots, y_C\}$ with a hypothesis h , and using h to predict y values for previously unseen $\mathbf{x}$ values [3]. Often the training set is of the form $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$. This type of learning is referred to as *supervised learning*.

In principle, there can be an arbitrary number of classes or class labels C . However, it has been demonstrated that a classification problem with C classes can be decomposed into several problems with two classes [4]. Therefore we consider the problem with $C = 2$ as a fundamental problem for classification.

2.1.2 One-class vs two-class

Let us examine a simple supervised classification problem with two classes. We create an artificial dataset with 75 objects (50 in one class, 25 in the other) and two features. The first step is to train a classifier, for instance the linear classifier, using these training points. The classifier uses information from both classes in order to find the best line that separates these classes. The trained linear classifier is shown as a dotted line in Fig. 2.1a. Now points to the left of the boundary will be classified as class 1, while the rest will be classified as class 2.

Just as in this example, in traditional classification problems, training objects from both classes are used in order to construct a model of the data. The decision boundary of whether a new object belongs to class 1 or class 2 is therefore supported by both classes. However, in real-life classification tasks, it may be often difficult to obtain sufficient data from both classes. For instance,

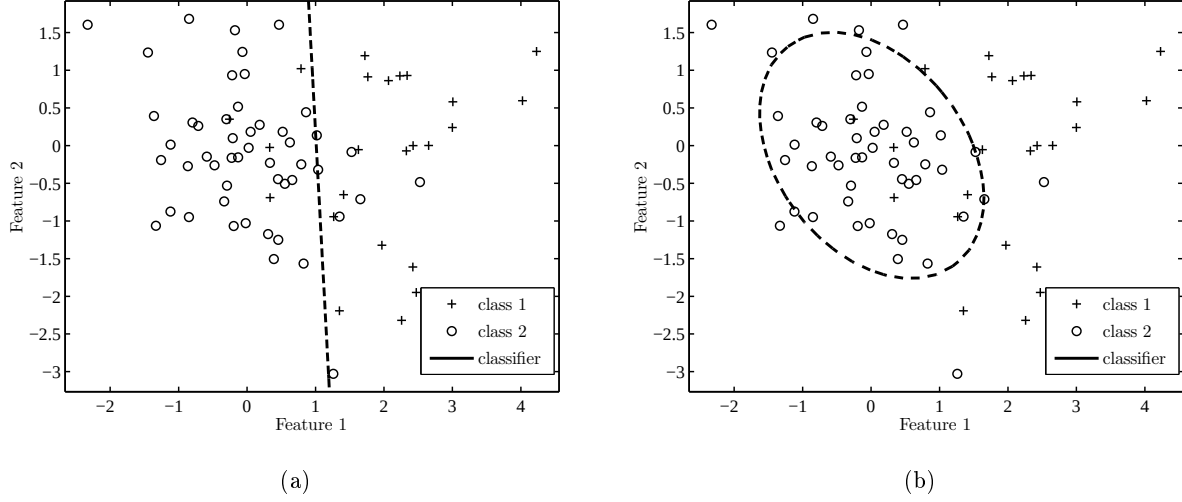


Figure 2.1: Traditional and one-class classifiers

in detecting credit card fraud, a lot of data concerning normal transactions is available, but only a few fraudulent transactions may be available to be used as training data. In this case, it would be possible to only create a description of normal transactions, and classify transactions not matching this description as fraudulent. This type of problem is called one-class classification [5] or outlier detection.

A simple example is shown in Fig. 2.1b. Now only points from one of the classes are used to create a model of the data. The points inside the circular boundary will be classified as class 1, points outside the boundary will be classified as class 2. Note that although points of class 2 are not used for training the classifier, they are necessary for evaluating classifier performance.

2.1.3 Properties

A classifier does not *really* draw a boundary and then examine where points are located geometrically in relation to the boundary. The boundary is a concept to aid the understanding of how a classifier decides whether to accept or to reject an object. One-class classification is really done using two elements: the “similarity” of an object $\mathbf{z}$ to the target class, and a threshold on this measure. This “similarity” may be expressed as a distance $d(\mathbf{z})$ or a probability $p(\mathbf{z})$. The thresholds on these measures are denoted as θ_d and θ_p respectively. For a new object, the classifier does the following:

$$f(\mathbf{z}) = \begin{cases} \text{target} & \text{if } d(\mathbf{z}) < \theta_d \\ \text{outlier} & \text{otherwise} \end{cases} \quad (2.1)$$

Or, in the case with probabilities:

$$f(\mathbf{z}) = \begin{cases} \text{target} & \text{if } p(\mathbf{z}) > \theta_p \\ \text{outlier} & \text{otherwise} \end{cases} \quad (2.2)$$

The boundary around the target class is thus the set of points where the measure is equal to the threshold. This threshold can be varied in order to “tighten” or “loosen” the boundary around the target class. This property is important for the errors the classifier will make.

Consider the example in Fig. 2.2. On the left, the classifier has a very loose boundary, thus accepting all the target objects (true positives), but also quite a few outlier objects (false positives). On the right, the classifier has a tight boundary, rejecting more outlier objects (true negatives), but also rejecting a few target objects (false negatives). When we say the threshold of a classifier is varied, we actually speak in terms of these quantities rather than actual distances or probabilities. For instance, the classifiers in Fig. 2.2 reject 10% and 25% of the target objects.

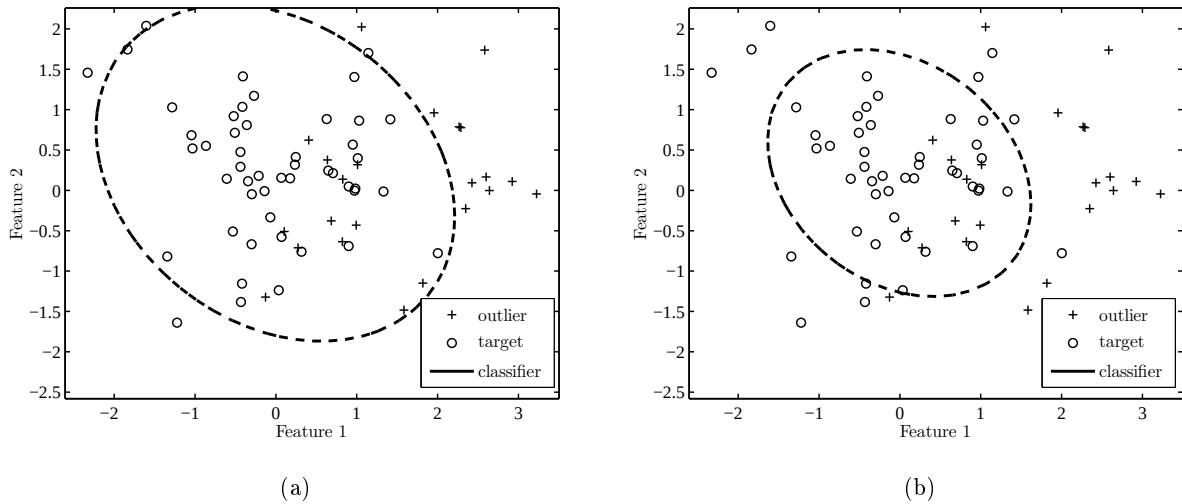


Figure 2.2: Effect of threshold on classifier boundary

The false positive and false negative rates are also important for evaluation of the classifier. A traditional approach would be to just sum these quantities up to obtain the accuracy of a classifier. However, this may be problematic when only few objects of one of the classes exist. Therefore, a more popular approach for evaluating one-class classifiers is to examine several thresholds of a classifier and its corresponding error rates [6]. Together, these points form the Receiver Operating Characteristic (ROC) curve. Because such curves may be difficult to compare, the integrated area under the ROC curve (AUC) is used as a measure to compare classifiers [7].

An example of an ROC curve is shown in Fig. 2.3. Here a “relevant” part of curve (high true positive rate, low false positive rate) is shown. Ideally, the true positive rate would be 1 and the false positive rate would be 0, i.e. all target objects and no outlier objects would be accepted. In this case, the AUC would be equal to 1.

Note that a 1% increase in AUC performance does not translate to 1% increase in accuracy for a particular threshold value of the one-class classifier. For instance, in Fig. 2.3, at some threshold values both classifiers have the same true positive and false positive rates, whereas at other thresholds one classifier outperforms the other by 5%.

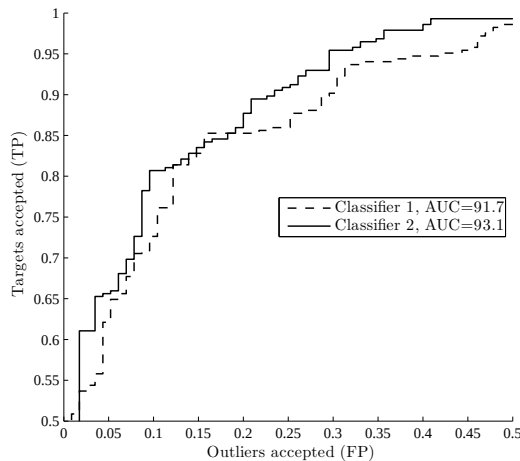


Figure 2.3: ROC curve

2.1.4 Approaches to one-class classification

The example in the previous section is not the only approach to one-class classification. In fact, there are several ways in which a model of the data can be created using only objects from one class. In [5] three main strategies are identified:

Density Density methods assume that the data follows a particular distribution and use the available training data to estimate the parameters of the distribution. An example is a Gaussian classifier, which determines its threshold by estimating the mean and covariance matrix of the data distribution. Such methods typically need a lot of training data to estimate the parameters correctly.

Boundary Boundary methods use distances between objects to construct a boundary around the target class. An example is the Nearest Neighbor one-class classifier which compares the distance between an unseen object $\mathbf{z}$ and its nearest neighbor $\mathbf{x}_1$ to the distance between $\mathbf{x}_1$ and *its* nearest neighbor $\mathbf{x}_2$. Such distance ratios can be thresholded to find a boundary around the target objects. Such methods typically need less data than the boundary methods, but are sensitive to the scaling of the features.

Reconstruction Reconstruction methods assume a model that generated the data and describe unseen objects using this model. An example is the k -Means classifier which assumes that the data is clustered in k groups, which can be described by k prototype objects, and creates a boundary out of the hyperspheres, placed at these prototype objects.

2.2 Ensembles of Classifiers

An ensemble of classifiers is a set of classifiers whose individual decisions are combined to classify new examples. Ensembles have received a lot of attention in the last decade because of they are often more accurate than the individual classifiers they are built from [3, 8, 9]. Next to empirical

studies, several theoretical explanations for the advantages of using ensembles have been proposed [10, 3].

In this section we examine further the motivation behind using ensembles and explore the the most typical approaches for building ensembles.

2.2.1 Motivation

The Condorcet Jury Theorem [11] describes that a committee of voters is more accurate on average than any of the individual voters, provided each voter is correct at least half of the time, and that the decisions of the voters are independent. Dietterich [3] translates this into the following: if the error rates of L hypotheses h_l are all equal to $p < 0.5$ and if the errors are independent, the probability that the majority vote will be wrong will be the area under the binomial distribution where more than $L/2$ hypotheses are wrong, or

$$p_{error} = \sum_{i=L/2}^L \binom{L}{i} p^i (1-p)^{L-i} \quad (2.3)$$

From this follows that:

$$\lim_{L \rightarrow \infty} p_{error} = 0 \quad (2.4)$$

In other words, with a very large number of voters, the probability of the committee's decision being wrong will be very low. An important assumption here is that the errors must be independent. There is no point in combining classifiers which have the same errors. However, it also does not help if the classifiers's output are completely different, because the the $p < 0.5$ condition will not be satisfied for every classifier. Therefore, classifiers in an ensemble must be both accurate *and* diverse, i.e. make errors on different parts of the input space [12, 13], in order to improve performance.

Diversity concerns two or more classifiers and is therefore more difficult to define [14]. An example of a simple diversity measure is disagreement, which is the percentage of items that two classifiers disagree on. For an ensemble of more than two classifiers, the average disagreement between any two classifiers would be calculated. This is a so-called *pairwise* measure. Several *non-pairwise* measures such as entropy also exist [15, 14].

However, diversity it is debatable to which extent diversity is important for ensembles. Some researchers explicitly use diversity as an evaluation measure for ensembles [16], while others are unable to find a real relationship between diversity and the success of an ensemble [14].

Another explanation for the success of classifier ensembles is that they are able to find a better approximation of the true function f for the data [3, 17]. On the one hand, with limited data the space of hypotheses $\mathcal{H}$ is discrete and the true hypothesis f may not correspond to any of the hypotheses formed by the single classifiers. By combining different data models, ensembles are able to find a better approximation for the true hypothesis. This situation is illustrated in Fig. 2.4a.

On the other hand, it is possible that f is not present in $\mathcal{H}$. Combining hypotheses may be able to expand $\mathcal{H}$ so that it includes the true hypothesis or a better approximation of it. This situation

is illustrated in Fig. 2.4b.

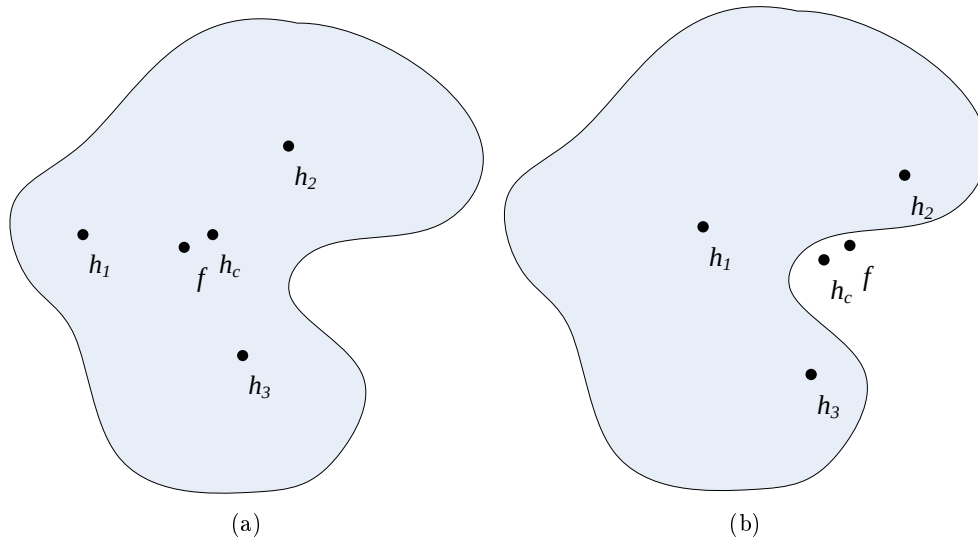


Figure 2.4: Approximating the true hypothesis with ensembles. The figures are adaptations from [3].

2.2.2 Ensemble architecture

There are two main components to an ensemble: a set of diverse classifiers and a combining method. For both components, there are a number of choices that can be made.

Diverse classifiers

Ensembles exploit the local different behavior of the base classifiers to improve the accuracy and robustness of the overall classification [17]. Different behaviour of classifiers can be achieved by varying two things: the base classifier or the input to each classifier. For instance, we can use different versions of the same classifier, but initialized with different parameters, or use fundamentally different classifiers.

A more popular approach is to vary the data each base classifier is trained on. The main approach is illustrated in Fig. 2.5. This can be done by training each classifier on a bootstrapped sample from the original training set as in *bagging* [18], or training each classifier on a random subset of the available features, called *random subspace method* [19]. Due to conceptual similarity to bagging, the Random Subspace Method has also been called *attribute bagging* [20] or *feature bagging* [21]. These approaches are sometimes grouped under the term *independent* [22] because each classifier is trained independently, and thus classifiers may be trained in parallel.

The classifiers can also be trained in series, each time using the result of the previous classifier to train the next classifier. An example of such a method is called *boosting* [23]. The first classifier is trained on objects with equal weights. Then the weights of the items misclassified by this classifier

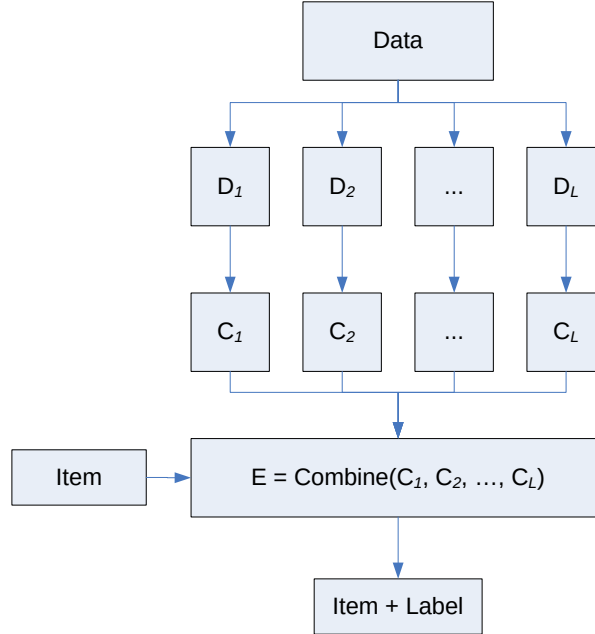


Figure 2.5: Ensemble of classifiers, trained on different subsets of the data

are increased, so they receive more emphasis when training the next classifier. This is a *dependent* approach [22] because the classifiers are trained in series.

Using a number of these methods simultaneously is also possible. For instance, Breiman uses both bagging and randomized versions of decision trees in a technique called *random forests* [24]. Other approaches combining bagging and random subspace method have been proposed in [25, 26].

In-depth explanations of bagging and boosting can be found respectively in [18] and [23]. In section 2.3 we will explain RSM in more detail.

Combining method

In order to combine classifiers, it is necessary to transform the outputs of these classifiers into a single output. Xu et al identify three types of classifier outputs in [27]: a discrete label, a ranked list of discrete labels or a list of real valued class posterior probabilities. The type of classifier outputs determines the ways in which these outputs can be combined. Note that it is possible to convert posterior probabilities into discrete outputs [28], but the opposite may be more difficult [5].

So how can we combine several labels or probabilities into one output? Several taxonomies describing different combining methods can be found in [22, 29, 16]. However, here we only describe a few commonly used combining approaches.

A very conceptually simple approach is called *voting*. This approach is suitable for labels, but can also be applied to posterior probabilities. For each item to be classified, each classifier “votes” for a class label. The item receives the label with the most voters. This way, all classifiers have equal votes. Alternatively, some classifiers may have receive more weight according to their

accuracy, hence the term *weighted voting*. However, this approach is more expensive because each classifier has to be evaluated separately.

When the outputs of classifiers are real-valued, i.e. posterior probabilities $p_i(\omega_t|\mathbf{x})$ or estimates thereof, it is possible to apply mathematical rules to them. In [10, 5] several derivations are shown. For example, it is possible to *average* the posterior probabilities over all classifiers. This is illustrated in equation 2.5, where $f_{avg}(\mathbf{x})$ can be used as a standard classifier output.

$$f_{avg}(\mathbf{x}) = \frac{1}{L} \sum_{i=1}^L p_i(\omega_t|\mathbf{x}) \quad (2.5)$$

Another possibility is to apply the *product rule* to the posterior probabilities. This is illustrated in equation 2.6.

$$f_{prod}(\mathbf{x}) = \frac{\prod_{i=1}^L p_i(\omega_t|\mathbf{x})}{\sum \prod_{i=1}^L p_i(\omega_t|\mathbf{x})} \quad (2.6)$$

In [29] it is argued that voting and averaging suffer in problems where not all classifiers have comparable performance. An alternative approach is to use *order statistics* such as the maximum value, which are said to be less influenced by a few wrong outputs.

In the case where some classifiers are better at classifying certain areas of the data, it might be advantageous to use *meta-learners*. Meta-learners are classifiers which are trained on the classifier outputs and the correct labels and thus learn which classifiers are better at classifying which samples. However, this is a very expensive technique which might also be prone to overfitting [29].

2.2.3 Ensemble design

An ensemble designer now has several methods to generate diverse classifiers and several combining methods to choose from. A selection of a method from each category, such random subspace method and averaging, and the choice of the number and type of classifiers, can be used to build an ensemble. The possibilities for ensembles that can be created in this way are endless. Although some methods are attributed with particular properties, there are few guidelines on which ensemble methods to use for a particular classification problem [30]. Furthermore, it has been shown that a subset of the classifiers in an ensemble may be as strong as the whole ensemble [31].

These reasons have led to the introduction of the “overproduce and choose” [30] paradigm, in which only a subset of an initial set of classifiers are selected to be in the final ensemble in order to achieve the highest accuracy. The classifiers that are not selected are therefore *pruned* [32]. Here, Rokach identifies two types of pruning methods:

- In *ranking-based* methods, the ensemble members are all evaluated using a specified criterion, and only the ensemble members with values over a threshold value are selected. For instance, Bryll et al [20] select the best subsets based on their accuracy on the training set.
- In *search-based* methods, the space of possible ensembles is traversed using a search algorithm (such as forward or genetic search) while evaluating each possible ensemble in order to find

the best-performing ensemble. During such search, there may be a focus on the diversity (or disagreement) of the ensemble members [16].

This is somewhat analogous to feature selection: ranking-based methods are like univariate feature selection, where each “item” is evaluated separately, whereas search-based methods are similar to multivariate feature selection, where sets of “items” are evaluated.

We note that there is also a combined approach, i.e. search methods can also be applied to ensemble members, leading to the task of *ensemble feature selection* or EFS [33]. The starting point of EFS is the random subspace method. Then for several iterations, each ensemble member is iteratively modified and evaluated in order to create an overall better ensemble [34, 16]. Therefore, ensemble selection and EFS are similar in the goal of finding the best set of base classifiers, but their search spaces are different. Ensemble selection searches in the space of ensembles (Which members should I select for my ensemble?), while EFS searches in the space of ensemble members (Which features should I select for each ensemble member?).

2.3 Random Subspace Method

2.3.1 Motivation

The random subspace method is a type of ensemble method where sampling features is applied to produce a set of diverse classifiers. Next to the motivations of using ensemble methods in general, described in section 2.2, there main reason why random subspace method is advocated is that it reduces the dimensionality of the data in each training set. Thus, RSM is able to reduce problems caused by the “*curse of dimensionality*” [35]. Furthermore, sampling features instead of using bootstrap samples objects as in Bagging has the added advantages of creating smaller training sets (i.e. faster training) in which all classes are represented equally well (i.e. more accurate training) [20].

In [36], the authors emphasise the importance of using lower-dimensional feature subspaces for one-class classification. They propose that many one-class classifiers use distances to nearest neighbours and that such distances may be less meaningful (i.e. less reliable) in high dimensionality or in presence of noisy features. They suggest that one-class classifiers are more effective in lower dimensions, however, they do not promote feature selection as it might be difficult to choose one best subset of features.

2.3.2 Implementation

The random subspace method has been originally proposed to construct diverse decision tree classifiers [19]. However, the idea of training classifiers on different feature sets can in principle be applied on any classifier.

The algorithm for training a random subspace classifier is presented in 1. There are four inputs: the training set x , the base classifier w , number of subspaces L and number of subspaces d_s . We describe these parameters in more detail in Table 2.1.

Table 2.1: Parameters of RSM

Parameter	Description
w	Base classifier. This can in principle be any classifier, or even different classifiers, as long as it is possible to combine their outputs in some way.
L	Number of subspaces. In theory, this number should be as large as possible [11] to provide the best accuracy, but in practice, up to 100 subspaces are used.
d_s	Number of features per subspace. This number is often fixed to a value such as $\sqrt{N}$ ([24]) or a number which is linearly related to the number of features. Another possibility is to vary the number of features in each subspace, as in [37]. Some algorithms do not specify $nFeat$ beforehand, but evaluate results using different number of features and then continue using the most optimal value, as in [20].
M	Combining method. Although the original RSM implementation uses majority voting [19], it is also possible to apply other methods, such as averaging or order statistics.

In RSM, L subsets of d_s features are randomly generated and stored in *subspace*. Then the base classifier is trained on each of the subsets in *subspace* in order to produce L different classifiers. These classifiers are combined to produce an ensemble classifier E .

Algorithm 1 RSM(x, w, d_s, L, M)

```

for  $i = 1$  to  $L$  do
     $subspace(i) \leftarrow select(x, d_s)$ 
     $clasf(i) \leftarrow train(subspace(i), w)$ 
end for
return  $E \leftarrow combine(clasf(1, 2, \dots, L), M)$ 

```

Alternative versions of RSM have been proposed in [20, 32]. Here the trained subspace classifiers are first evaluated on the training or validation set, and then sorted according to their performance. Only L_s best performing classifiers are then combined into an ensemble, which we call PRSM for Pruned RSM. The algorithm is described in 2.

2.3.3 Applications to one-class problems

RSM has originally been designed for decision trees, a “two-class” classifier. At the time, one-class classification was still an emerging topic [5]. Therefore, initial additional research on RSM also focused on traditional classifiers. For instance, it has been demonstrated that RSM is particularly effective for “weak” (two-class) classifiers. Classifiers using RSM have also been used in some real-life applications, such as EEG classification [38], microarray data [39, 40] and MRI classification [41].

However, attempts at combining RSM with one-class classifiers have also been made. In [5]

Algorithm 2 PRSM(x, w, d_s, L, L_s, M)

```
for  $i = 1$  to  $L$  do  
     $subspace(i) \leftarrow select(x, d_s)$   
     $clasf(i) \leftarrow train(subspace(i), w)$   
     $score(i) \leftarrow test(x, clasf(i))$   
end for  
 $clasf \leftarrow sortByScore(clasf, score)$   
return  $E \leftarrow combine(clasf(1, 2, \dots, L_s), M)$ 
```

it is shown that combining one-class classifiers trained on different subspaces is more useful than combining different one-class classifiers trained on the whole feature space.

More recently, combining one-class classifiers trained on different feature sets has also been applied to several real-life one-class problems. In order to gain a better understanding of what has been done in this area, we provide an overview of how various works combine one-class classifiers. 2.2. This summary is not intended to provide complete descriptions of these research papers because their focus is often different and because often some domain-specific approaches are used. However, we are interested in how these papers produce ensembles of one-class classifiers.

Although we doubt that this is an exhaustive list, we can say that there is a number of different applications where the success of training one-class classifiers on feature subsets has been demonstrated. Furthermore, these papers vary in their choice base classifier and other parameters. This suggests that using subspaces might be beneficial for one-class classifiers in general.

On the other hand, such a variety of parameters might also suggest that it is not straightforward to achieve an increased performance using subspaces. In fact, we have not yet encountered any papers which provide guidelines on how to apply RSM (or other ensembles) to one-class classifiers, i.e. for what type of data or what parameters it might be effective. Such guidelines seem to be somewhat more clear for traditional classifiers, however, it would be wrong to assume that the same properties hold for one-class classifiers. For instance, it has already been demonstrated that the best combining rules for multi-class classifiers are not the best for one-class classifiers [5]. Therefore it is difficult to conclude something about the effect of RSM on one-class classifiers in general.

Table 2.2: Examples of combining one-class classifiers.

Ref	Application		Classifier	Feature sets	Combining
[42]	anomaly	detection	Support Vector	Separate feature sets	voting
[43]	remote sensing images		Mixture of Gaussians Support Vector	Separate feature sets	mean product
[44]	intrusion	detection	Support Vector k -means Parzen Window	Separate feature sets	mean product minimum maximum
[36]	UCI repository		LOF LOCI	RSM ($d/2$ to $d - 1$)	weighted sum weighted vote
[45]	intrusion	detection	Support Vector k -means Parzen Window	Separate feature sets	mean product maximum minimum
[46]	fingerprint authentication	au-	Nearest Neighbor PCA Support Vector	Random Bands (similar to RSM) (100 of $0.5d$)	maximum
[47]	signature verification		Gaussian Mixture of Gaussians Nearest Neighbor PCA Support Vector Parzen Window Linear Programming	RSM (100 of $0.6d$)	maximum
[48]	spam filtering		Support Vector Logistic Regression	RSM (3 to 10 of $0.2d$ to $0.8d$)	weighted sum

Chapter 3

One-class Classification for Prime Vision

3.1 Introduction

In this chapter we answer the first main research question: *“Are one-class classifiers suitable for the Prime Vision problem?”*.

Firstly, we need to investigate whether the current data representation is sufficient for discriminating outliers or whether a different set of features is needed. To do so, we test a number of simple classifiers using the original or modified (for instance, normalized) data. If even in the best-case scenario the performance is “too low” (say, AUC of less than 0.9), probably the classes are not well-separable and a better feature set is needed.

Secondly, when a good feature set is available, we can investigate how one-class classifiers compare to two-class classifiers. We examine the performance of the classifiers, but also their robustness against previously unseen data, as this would be an important property in a real-life setting.

In section 3.2 we provide an overview of the available data. The performed experiments are described in section 3.3, followed by the results and analysis in section 3.4. A conclusion about using one-class classifiers on Prime Vision data is given in section 3.5.

3.2 Data

3.2.1 Objects

The data consists of images of clusters extracted from envelopes. A cluster may be a word, an address block, a stamp or barcode, or even a collection of either of these things. Therefore, each envelope image is split into a number of clusters, which may be subsets of each other. An example is shown in Fig. 3.1. In this and following examples, sections of names or addresses have been blurred out for privacy purposes.

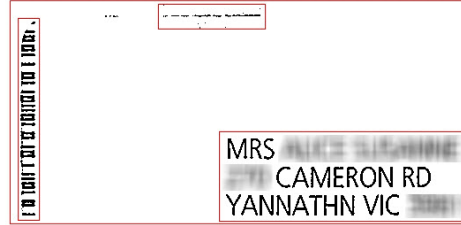


Figure 3.1: A postal image with five clusters.

Each cluster contains one or more connected components, such as letters or numbers. Although we focus on classifying the cluster as text or non-text, the connected components within the cluster may help us identify what class the cluster belongs to.

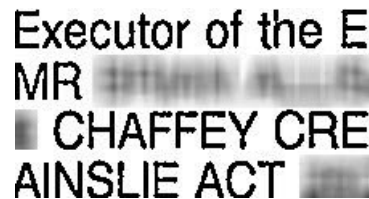
In order to distinguish between text and non-text clusters, we have identified the following classes:

- 1: Typed text** This class consists of examples of typed text. The text can consist of a single line or of multiple lines of black-on-white text and should be consistent in terms of font style and size. Text with a border around it or slightly “noisy” text (with pixels which do not belong to any of the letters) are also allowed.
- 2: Written text** This class is analogous to class 1, but concerns only handwritten text.
- 3: Noise** This class consists of images which do not contain any text, or only a minor proportion of text, such as stamps, logos and barcodes.
- 4: Multiple** This class consists of images with multiple parts, each of which may belong to any of the classes above. An example might be an address label which includes a barcode, or a letter with two addresses on it, separated by white space.

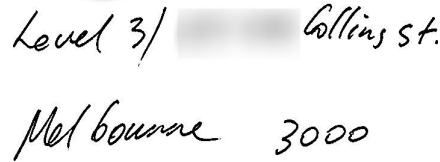
Classes 1 and 2 both represent text or the target class. Examples of each of these classes can be found in Fig. 3.2. Although all of these classes should be classified as text, distinctions were made between typed and written text. This distinction was made as a precaution in order to identify problems better. For instance, performance of a classifier might be low because written text data is misclassified as outliers. The availability of labels for typed text and written text would enable the user to test the performance of the classifier on only typed text and outliers.

Classes 3 and 4 represent non-text or outliers. Examples of each of these classes can be found in

Fig. 3.3. Here again, a distinction was made between two types of outliers: “single” noise images and “multiple” noise images, where both text and non-text elements may be present.



(a) Typed text

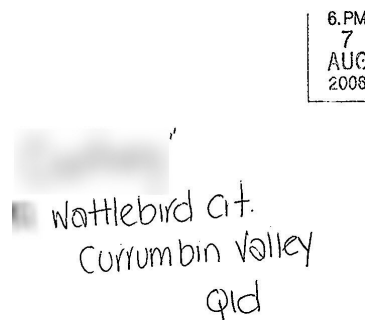


(b) Written text

Figure 3.2: Examples of text



(a) Noise



(b) Multiple

Figure 3.3: Examples of non-text

The dataset used for data analysis contains 2000 objects. In fact, Prime Vision has access to more (unlabeled) data, so in principle, even a larger dataset could be created. However, we feel that we should avoid using more data unless this is necessary. Learning curves of several simple (both two-class and one-class) classifiers have indicated that there is little or no performance improvement after the training set size has reached about 1500. To stay on the “safe side” and because some data is needed for testing, we decided to use 2000 objects in total.

The objects in the dataset are divided into four classes described above. The distribution of the objects per class is shown in Fig. 3.4a. However, for text/non-text classification, the dataset is modified to contain only a target class (text) and an outlier class (non-text). Therefore, the Typed and Written classes are grouped to create the target class, and the Noise and Multiple classes are grouped to create the outlier class. The resulting dataset is shown in Fig. 3.4b.

Class	Typed	Written	Noise	Multiple
Objects	1140	285	370	205

(a) Original dataset

Class	Target	Outlier
Objects	1425	575

(b) One-class dataset

Figure 3.4: Modifying the dataset

It must be noted that the labeling of the data may not be optimal for a classification problem because some images do not clearly belong to a particular class. For instance, a text image with 1% “noise pixels” (which do not belong to any of the letters in the text) should probably still be characterized as text. On the other hand, a text image with 50% noise pixels will probably not

have any resemblance to text for a classifier, even if a human could still be able to “guess” the text. It is difficult to say, for such cases, where the line between target and outlier should be drawn.

An example of this problem is shown in Fig. 3.5. Currently, Figs. 3.5a and 3.5b are labeled as target whereas the other Figs. are labeled as outlier. Although the two “extremes” (Figs. 3.5a and 3.5d) are quite different, it is possible to imagine that not-so-clear examples such as Figs. 3.5b and 3.5c might be closer to the other class and thus more difficult to classify.

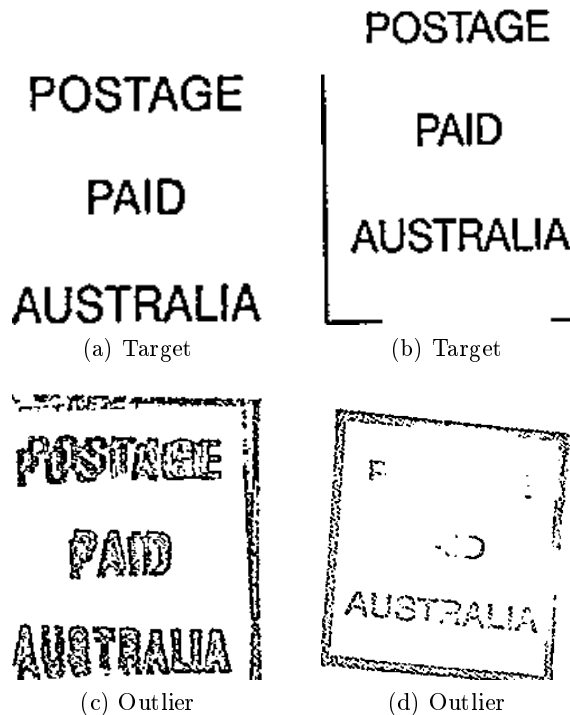


Figure 3.5: Drawing the line between target and outlier

3.2.2 Features

There are two feature sets provided by Prime Vision, describing the clusters and the connected components within each cluster. Here we describe the contents of each of these feature sets.

Cluster features

Each data object or cluster is described by a feature vector with 39 values. The list of original features provided can be found in Table 3.1. These features were originally meant for multi-class classification of address blocks, sender blocks, stamps and other clusters found on envelopes, not for classifying between text and non-text. Because of the nature of the original problem, position and size were very important feature for classifying clusters. For instance, address blocks and sender blocks may be very similar, but address blocks would typically be found in the bottom right corner, whereas sender blocks would be typically found in the top left corner.

However, it is not desirable to use position and size in classifying text from non-text, because

the assumption that an address is located in the bottom right corner and is landscape-oriented, may not always hold. Of course, position and size features might improve performance for a particular dataset, but on the other hand, they may be harmful when actual text is located in other parts of an image. To ensure that text/non-text classification is independent of position and size, a number of features were removed, leaving 23 features to be used for classification.

Connected Components features

We are also provided with a feature set for the connected components within each cluster. Each connected component is described by 8 features, which are shown in Table 3.2. In order to describe a component, an ellipse is formed around it. The features are derived from measurements relating to this ellipse.

However, because a cluster can contain a variable connected components, and classifiers typically need a fixed feature length, it is not possible to use this feature set for classification directly. Therefore we transform the features of all connected components per cluster into “statistic” features relating just to this cluster, such as average perimeter of the ellipse. The measures we compute are: mean, standard deviation, minimum, median, maximum and range. We also add the number of connected components per cluster as a feature. The resulting dataset contains $8 * 6 + 1 = 49$ features.

We note that this method is likely to create strongly correlated or redundant features. However, we wish to investigate whether the connected components information is at all suitable for separating the classes and the “noisy” dataset we create is sufficient for this purpose.

3.3 Experimental Setup

In the **Feature Set** experiment, we examine the performance of a number of classifiers on the available feature sets: cluster (denoted by Clust), connected components (denoted by Conn) and the combined feature set (denoted by Clust+Conn). Furthermore, we also test the normalized versions of these feature sets. Normalization is done by rescaling the 2-sigma interval of each feature f , $[f - 2 * std(f), f + 2 * std(f)]$, to $[0,1]$, and clipping values outside this domain.

We test each dataset using four classifiers: the linear two-class classifier (ldc), the k -nearest neighbour two-class classifier (k -nnc), the Gaussian one-class classifier (gauss) and the k -nearest neighbour one-class classifier (k -nndd). We use $k = 5$ as a reasonable value for the nearest neighbour classifiers. All other parameters are set to their default values.

Furthermore, we also consider using two dimensionality reduction techniques: Principal Component Analysis as a feature extraction method and Forward selection as a feature selection method. We know that some of the “generated” features may be very correlated or redundant, so dimensionality reduction techniques might be able to improve performance by disregarding such features. We compare classifier performances on the reduced feature sets to the performances on the whole feature set.

In the **Generalization** experiment, we test a number of “traditional” classifiers and one-class classifiers on their ability to generalize unseen data. All classifiers are trained using the following

Table 3.1: List of original features. The remaining features are marked with an *. $abs(A - B)$ denotes the absolute difference between A and B.

Name	Description
<i>NumberContours*</i>	Number of contours in the image
<i>BBWidth</i>	Width of bounding box
<i>BBHeight</i>	Height of bounding box
<i>BBCenterX</i>	Horizontal position of bounding box
<i>BBCenterY</i>	Vertical position of bounding box
<i>MeanCntrDx*</i>	Mean width of the contour
<i>MeanCntrDy*</i>	Mean height of the contour
<i>NNearLeft*</i>	Number of contours at the left side of the bounding box
<i>NNearRight*</i>	Number of contours at the right
<i>NNearTop*</i>	Number of contours at the top
<i>NNearBottom*</i>	Number of contours at the bottom
<i>Ori2*</i>	Indication for orientation of the script
<i>StdCntrDx*</i>	Standard deviation of the contour width
<i>StdCntrDy*</i>	Standard deviation of the contour height
<i>NKarX*</i>	Estimated number of characters per line
<i>NKarY*</i>	Estimated number of lines
<i>RelRegelLengte*</i>	Total line length normalized by the longest line length
<i>SymmX*</i>	Horizontal symmetry in black pixel distribution
<i>SymmY*</i>	Vertical symmetry in black pixel distribution
<i>UitLX*</i>	Horizontal outlining at the left side
<i>UitLY*</i>	Vertical outlining at the left side
<i>UitRX*</i>	Horizontal outlining at the right side
<i>UitRY*</i>	Vertical outlining at the right side
<i>RelX</i>	Horizontal relative position of the bounding box
<i>RelY</i>	Vertical relative position of the bounding box
<i>NClusters</i>	Number of clusters in the image
<i>BBCenterXMinY</i>	$abs(BBCenterX - BBHeight)$
<i>BBWidthMinHeight</i>	$abs(BBWidth - BBHeight)$
<i>MeanCntrDxMinDy*</i>	$abs(MeanCntrDx - MeanCntrDY)$
<i>StdCntrDxMinDy*</i>	$abs(StdCntrDx - StdCntrDY)$
<i>NKarXMinY*</i>	$abs(NKarX - NKarY)$
<i>SymmXMinY*</i>	$abs(SymmX - SymmY)$
<i>RelXMinY</i>	$abs(RelX - RelY)$
<i>NBlockLeft</i>	Number of blocks left of current block
<i>NBlockRight</i>	Number of blocks right of current block
<i>NBlockAbove</i>	Number of blocks above current block
<i>NBlockBelow</i>	Number of blocks below current block
<i>NumberInside</i>	Number of clusters on the inside of current cluster
<i>NumberOutside</i>	Number of clusters on the outside of current cluster

Table 3.2: List of Connected Componets features

Name	Description
<i>OppZwart</i>	Area with black pixels
<i>LSe</i>	Length of ellipse
<i>BrSe</i>	Width of ellipse
<i>Omtrek</i>	Perimeter of ellipse
<i>Ld</i>	Line thickness
<i>Slinger</i>	Ratio of perimeter to the area of ellipse
<i>OppZwartRel</i>	Ratio of area with black pixels to the area of ellipse
<i>RatioAB</i>	Ratio of the length to the width

subsets of the training data during each cross-validation fold:

Normal Both target and both outlier classes.

Target1 Only target class 1 and both outlier classes.

Outlier1 Both target classes and outlier class 1.

Outlier2 Both target classes and outlier class 2.

We expect that one-class classifiers are better at handling unseen data than traditional classifiers.

All experiments are performed using 5x stratified 10-fold cross-validation and the mean and standard deviation of the AUC, x100 (for space considerations) over these trials are presented as the results. We use the paired t-test with significance level of $\alpha = 0.05$ for comparisons. The type of comparison made will be given along with the results.

3.4 Results

3.4.1 Feature set

The results for different feature sets and their normalized versions are presented in Table 3.3. The results indicate that using the normalized dataset containing all the features gives the best results for each tested classifier. We therefore use this dataset for subsequent experiments. This is also the dataset that we will refer to as “Prime Vision data” from this point onwards.

The results of using dimensionality reduction methods on the Prime Vision data are shown in Table 3.4. Here we see that for all the classifiers, using the original features is never significantly worse than the best results per classifier.

Based on these results, we conclude that the original set of all the features is able to provide good performances for simple classifiers. We therefore choose this dataset as the “starting point” for the subsequent experiments. Note that we do not claim that this is the “best” possible dataset, as for other classifiers other datasets may be “best”. However, with the experiments that still

Table 3.3: Effect of feature set on AUC. Results in bold indicate results which are not significantly worse than the best result per classifier.

Data	Classifier			
	ldc	5-nnc	gauss	5-nnndd
Clust	88.1 (2.5)	91.4 (2.1)	89.3 (2.3)	89.4 (2.1)
Conn	91.4 (2.3)	89.2 (2.7)	86.8 (2.3)	83.2 (3.0)
Clust+Conn	93.8 (1.8)	89.4 (2.7)	89.4 (1.8)	84.6 (2.8)
Clust, normalized	88.0 (2.7)	90.9 (2.2)	91.0 (2.2)	92.7 (2.0)
Conn, normalized	92.5 (2.1)	92.8 (2.1)	90.1 (2.5)	90.8 (2.6)
Clust+Conn, normalized	94.7 (1.7)	94.0 (2.3)	93.9 (1.5)	94.7 (1.5)

Table 3.4: Effect of dimensionality reduction on AUC. PCA $k\%$ stands for the feature set after applying PCA to retain $k\%$ of the variance. Forward stands for the feature set obtained by forward feature selection. Results in bold indicate results which are not significantly worse than the best result, per classifier.

Features	Classifier			
	ldc	5-nnc	gauss	5-nnndd
Original	94.9 (1.7)	94.0 (2.3)	94.0 (1.6)	94.8 (1.5)
PCA 90%	91.6 (2.3)	93.8 (2.1)	92.9 (1.9)	94.4 (1.7)
PCA 95%	92.5 (2.2)	94.0 (2.5)	94.1 (1.6)	94.8 (1.6)
PCA 99%	93.1 (2.2)	94.2 (2.2)	94.4 (1.7)	94.9 (1.5)
Forward	94.3 (1.9)	94.9 (1.9)	93.3 (1.4)	94.5 (1.4)

have to be performed in mind, it would be infeasible to add dimensionality reduction as an extra “parameter” to each experiment.

3.4.2 Generalization

The results from the different generalization experiments are shown in Table 3.5. Here, we see that although two-class classifiers might perform slightly better than one-class classifiers when they have access to the outlier data, their performance drops from around 0.95 to 0.85 when they encounter previously unseen outliers during testing. For one-class classifiers, the presence or absence of outliers in the training data does not influence the results because only target objects are used during the training phase.

Table 3.5: Effect of modifying the training data on AUC. Results in bold indicate results which are not significantly worse than the best result, per classifier.

Training data	Classifier			
	ldc	5-nnc	gauss	5-mnndd
Normal	94.9 (1.8)	93.9 (2.1)	94.0 (1.4)	94.8 (1.4)
Target1	92.7 (1.5)	91.8 (2.1)	91.7 (1.7)	91.3 (1.7)
Outlier1	86.5 (2.9)	85.9 (2.6)	94.0 (1.4)	94.8 (1.4)
Outlier2	89.3 (3.1)	84.7 (3.1)	94.0 (1.4)	94.8 (1.4)

Our expectations about the generalization abilities of the classifiers were correct. This means that in a setting where new outliers are likely to occur, it would be a wise choice to use a one-class classifier, especially if this does not degrade the initial performance. Choosing between a very accurate two-class classifier and a significantly less accurate, but more robust one-class classifier would be more difficult. Fortunately, this is not the case for Prime Vision data, which leaves all the classifier options open.

3.4.3 Error analysis

We have seen that the examined classifiers are able to produce AUC performances of around 0.95. Because the performance still is not 100%, we wish to examine what kind of errors the classifiers make in order to determine how the performance can be improved.

We perform an experiment in which we count (during 5x stratified 10-fold cross-validation, i.e. each object is classified 5 times) how often each object is misclassified by each of the following classifiers: the linear, Nearest Neighbor and Nearest Mean (nnc) two-class classifiers, and the Gaussian, Nearest Neighbour and k -Means (kmeans) one-class classifiers. If an object has been misclassified at least once by a classifier, we label the combination of this classifier and object as “1” for misclassification.

A sample of the results of the experiment is shown in Table 3.6.

If an object does not get classified correctly by all classifiers (type 1), there are two other possibilities. It is possible that the object gets classified incorrectly by all classifiers (type 2), which

Table 3.6: Types of classifications of different objects.

Classifier	Object type		
	1	2	3
ldc	0	1	0
1-nnc	0	1	0
nmc	0	1	1
gauss	0	1	0
1-nnidd	0	1	1
<i>k</i> -means	0	1	0

would suggest that there is a problem with the data, i.e. the classes are not well separated. Another possibility is that some classifiers classify an object correctly, while others always fail to do so (type 3). This might indicate that none of the classifiers is able to find a good hypothesis for the data. Note that the generation of training and test sets also sometimes influences the classification of the object, but we do not examine this relationship.

We are mainly interested in the relationship between objects of type 2 and objects of type 3. If we discover that most errors are caused by objects of type 2, we would need to change the data representation, whereas errors caused by objects of type 3 might be improved by combining several hypotheses, as described in section 2.2.

We plot how many objects are misclassified (at least sometimes) by 0, 1, 2 and so forth classifiers in Fig. 3.6. We see here that only 38 objects at least sometimes get misclassified by all the classifiers, which means the majority of objects has at least one classifier that always classifies it correctly.

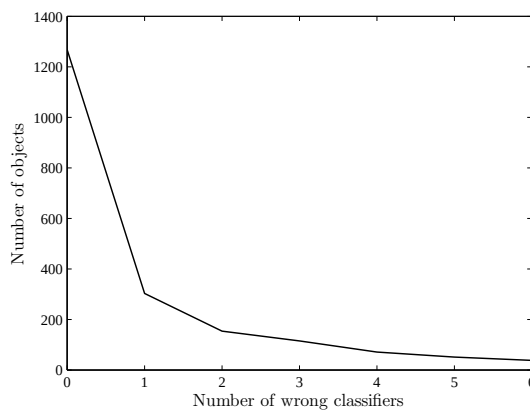


Figure 3.6: Number of misclassifications per number of classifiers

Next we examine the objects that get (at least sometimes) are misclassified by all of the classifiers. Most of these objects (36 out of 38) are objects that are “difficult” to label, i.e. they are target objects that are similar to outlier objects or vice versa. Let us call these misclassifications “logical”. The fact that most misclassifications are “logical” suggests that the features are able to

represent the differences between classes well.

Out of the “logical” misclassifications, 32 are false positives, i.e. outliers classified as target objects. These outliers are mostly (23 out of 32) of the Multiple class. In each of these images, text clusters are clearly present, but there are either several text clusters, or text clusters mixed in with some noise. The other “logical” outliers are of the Noise class, however, they are all examples of text which can still be readable (similar to Fig. 3.5c) but which is not as “perfect” as most text images. In other words, even though these images might not be detected as outliers, it would probably still be possible to extract text from them using OCR.

The other 4 “logical” misclassifications are false negatives, i.e. target objects wrongly classified as outliers. These objects are all of the Typed class, but contain some noise. All these objects are actually very similar to 3.5b).

These findings mean two things. First of all, it is clear that the feature representation is not perfect for the available classes because some objects close to the class boundary are often misclassified. However, most misclassifications made by a classifier are errors that a different classifier would not make, which suggests that a better classifier might improve the AUC even with the available features.

In order to deal with some of the “difficult” errors, we would recommend using more features that somehow reflect the variance of the “complexity” of the components in an image. Letters and numbers are quite “simple” elements because they consist of a few similar strokes. So in an image with only text of a particular size, the variance in this “complexity” should be low. Images containing different types of elements (such as the Multiple class) should have a higher variance.

Another possible feature is something like “average distance between components” in a cluster. This could help us solve the problem of images of the Multiple class which contain multiple clusters of text, but also a lot of white space, as here the average distance would be much higher than of a “dense” text image.

Instead of adding new features, it is also possible to examine whether different data representations are able to separate the classes better. For instance, we could use the pixels in an image as features. However, this requires all images to be of the same size, which might be problematic for Prime Vision data: an image with many lines of text is typically larger than an image with only one word, so if resized, these images would not be similar.

A different possibility is to use dissimilarities to “prototype” objects [49] as features. To do so, we would need to define typical text images, and compute (Euclidean) distances to these images using a set of features (such as the current feature set). These distances could then be used to classify new images as target or outlier. This method could be successful for Prime Vision data because it seems conceptually easy to choose good examples of the text class as prototypes. However, we cannot be sure that the combination of chosen prototypes and available features would result in a set of distances which would be better at distinguishing the classes than the original features.

Lastly, we could consider Multiple Instance Learning (MIL) [50] as data representation. In MIL, objects are “bags of instances”. Objects are not labeled directly, but receive a label according to the labels of its instances. This is very similar to classifying the components (using the Connected Components features) as text or outlier, and giving the cluster an appropriate label. For instance, clusters which contain only text components, would be considered text, whereas clusters containing

at least some noise components, would be considered noise.

3.5 Conclusion

We have demonstrated that the available features, when processed in a suitable manner, are sufficient to achieve relatively good performances (with AUCs around 0.95) for all tested classifiers. This means that, using these features, the target and outlier classes are already relatively well separated. However, these features are probably not sufficient to classify all objects correctly, as some objects lie on the class boundary. We suggested additional features and alternative data representations which might help be helpful in this situation.

For the “best” currently available feature set, we have demonstrated that most of the misclassifications did not depend on the data, but on the classifier used. Because different classifiers make different errors, it should be possible to combine their strengths to reduce the number of errors. Therefore, we expect that some performance improvements should still be possible with an ensemble classifier.

We have also shown that one-class classifiers are more robust against previously unseen outliers. Because for the chosen feature set, the tested classifiers have similar performance, we conclude that **one-class classifiers are suitable for classifying Prime Vision data.**

Chapter 4

Random Subspace Method

In this chapter we answer the second main research question: “*How does the Random Subspace Method affect the performance of one-class classifiers?*”.

In the previous chapter we have seen that one-class classifiers are suitable for classifying Prime Vision data. In this chapter we propose two ensembles for one-class classifiers, RSM and PRSM, and apply them to Prime Vision data. However, we also investigate how these classifiers affect the performance on one-class classifiers in general, and how the performances of RSM and PRSM depend on their parameters.

We describe the datasets which are also part of this investigation in section 4.1. The ensemble classifiers are presented in section 4.2. The experiments are described in section 4.3 and the results are presented in section 4.4. A conclusion is presented in section 4.5

4.1 Data

We are primarily interested in improving performance on Prime Vision data, so for the experiments we use the Prime Vision dataset from chapter 3. However, we also use several other datasets to be able to draw more general conclusions about RSM. We select several datasets from [51]. Most of these datasets originate from the UCI Machine Learning Repository [52], but have been modified in order to contain a target and an outlier class, and to deal with missing or categorical values.

A complete list of datasets, available to us, is presented in Table 4.1. Per dataset, we present the number of target and outlier objects and the number of features. The last column contains the best AUC performance obtained with one-class classifiers (including more complex classifiers such as the Support Vector one-class classifier) as reported in [51]. This performance indicates the best-case situation and thus how much improvement is still possible. Note that the best performance is reported as the mean AUC and its standard error, while we report the mean AUC and standard deviation in our results.

Table 4.1: Available datasets. The last column contains the best AUC performance and standard error (not standard deviation as in our results) of single one-class classifiers, as reported in [51].

Dataset	Targets	Outliers	Features	Best AUC
Concordia	400	3600	256	94.4 (0.9)
Hepatitis	123	32	19	82.1 (1.0)
Imports	71	88	25	87.6 (1.0)
Ionosphere	225	126	34	97.8 (0.2)
Prime Vision	1425	575	72	
Sonar	111	97	60	84.9 (0.7)
Spam	1813	2788	57	89.9 (0.1)

4.2 Classifiers

We propose two ensemble classifiers, RSM and PRSM. We implement the classifiers according to the algorithms in 1 and 2 except a slight modification for PRSM. In 2, the subspaces are evaluated using the whole training set. However, there are two important assumptions which must hold for this method to be effective:

- The performance on the training set must be correlated with the performance on the test set.
- The performance on the training set must not be 100%.

The second assumption is important because if all subspace classifiers have equal (100%) performance on the training set, there will be nothing to sort. In this case PRSM will reduce to RSM. In order to see different performances, we must use unseen data. Ideally, we should use a separate validation set, which is not used for training or testing the final classifier. However, this might be problematic for datasets which already have very few items. To overcome this problem, we split each training set into 90% training and 10% validation set. This way the classifiers are still trained using a sufficient amount of items and their performance is evaluated using previously unseen data.

Another implementation issue which must be addressed is the subspace generation procedure. As the name of the algorithm suggests (*random* subspace), this procedure should be random. This is also the case with our implementation, however, we chose to reset the random seed every time subspaces are generated for the purpose of comparison. In other words, the subspaces that are generated for a particular dataset are the same during every experiment.

Base classifier

The implementations of RSM and PRSM can in principle be applied to any classifier, or even to different classifiers for each subspace. We will use the same base classifier in each ensemble. However, we will test RSM for several one-class classifiers. We propose to use the following classifiers available in the Data Description toolbox [53]:

- Gaussian

- k -nearest neighbour
- k -means

A description of these classifiers is given in section 2.1.4.

Our choice of classifiers is based on their diverse data representation (density, boundary and reconstruction [5]). We expect that some classifiers will be more suitable to model the data in some datasets than in others, and that this will influence the fact whether the classifier will benefit from using RSM. If a classifier is able to model the data well in the complete feature space, there will probably be no increase in performance using RSM. On the other hand, if a classifier is unable to deal with the complete feature space, using subspaces might be a good way to increase the space of models, representable by the classifier, thus possibly improving the performance.

For each classifier, we will use the default or reasonable parameters. For the Gaussian classifier, the default regularisation parameter $r = 0.01$ is used. For the k -Means classifier, the default number of prototypes $k = 5$ is used. The default setting of the k -nearest neighbour classifier is to optimize k using a leave-one-out procedure. However, this is very time-consuming, especially for larger datasets. Unless stated otherwise, $k = 1$ is used as parameter, this classifier is then also denoted by 1-NN. When the optimization is used, the classifier is denoted as k -NN.

Clearly, these parameters might not be optimal for the various datasets and therefore the performance of the base classifier might be lower than the best case scenario for a particular classifier. However, it is not our purpose at this stage to investigate which classifier and which parameters will provide the optimal performance. At this point, we are interested in whether RSM is able to improve performance of a basic one-class classifier such as one of the above.

An alternative method for avoiding very good or very bad parameters would be to randomize both feature subsets and classifier parameters as is done in [25]. If the classifiers are evaluated before being included in an ensemble as with PRSM, only good combinations of features and classifier parameters would be chosen.

4.3 Experimental Setup

We want to find out whether RSM and PRSM improve the performance of one-class classifiers, and how the improvement depends on the parameters of these ensembles.

Parameters RSM

We investigate the influence of parameters, such as number of features, number of subspaces or the combining method, on the performance of RSM. First, we train 100 subspaces of 25%, 50% and 75% of features per subspace. Then we create ensembles of the first $nSub \in [2, 5, 10, 20, 50, 100]$ subspaces. This way, each ensemble of size s is a subset of a larger ensemble of size $s + 1$. Each ensemble is then combined using four available combining methods from [54]: mean (meanc), product (prodc), voting (votec) and the maximum rule (maxc).

We expect that these settings will influence the performance of an ensemble strongly. Our

intuition (based on previous research on ensembles) is that ensembles with many subspaces with “not too few, but not too many” features and a “good” combining method will perform the best. However, it is not yet clear what the terms in parentheses might be for one-class classifiers. We also suppose that these terms might depend on the dataset in question.

In this experiment, we compare performances using pairwise comparisons with the base classifier.

Parameters PRSM

We also perform two different experiments to investigate the performance of PRSM. In the first experiment, we train 100 subspace classifiers and use only a selection of the best performing classifiers in the final ensemble. Thus we investigate whether it is worthwhile to build a smaller ensemble with better classifiers or a large ensemble with “average” classifiers.

Furthermore, we investigate how PRSM behaves with an increasing number of classifiers to choose from. The other parameters (number of features, combining method) will be set to reasonable parameters from experiment 1.

We expect that by increasing the number of subspaces that we can choose from, or “pool size”, and keeping the number of chosen subspaces constant, the performance of the ensemble will improve. An important assumption here is that we have a good method to evaluate individual subspaces, i.e. subspaces that have high scores will also contribute to a better ensemble. However, this improvement will stop at some point because duplicate subspaces will be added to the pool and ensembles will no longer be able to benefit from a large pool size. This is especially dangerous for data with just a few (< 10) dimensions.

Notice that if this expectation is correct and in experiment 1 we indeed observe that larger ensembles are more accurate, it does not immediately hold that an ensemble satisfying both conditions (large ensemble size, large pool size) will perform the best. In that case, PRSM would not have any advantages because it, by definition, creates smaller ensembles. Therefore we also vary the number of selected subspaces in this experiment.

4.3.1 Evaluation

All experiments are performed using 5x stratified 10-fold cross-validation in order to obtain good estimates of the classifier performances. The results are reported as the average AUC and its standard deviation (both $\times 100$ for space considerations) over the 50 runs.

The dependent paired t-test is used to test statistical significance between pairs of classifiers on a particular dataset, typically the base classifier and an ensemble classifier. We use the dependent test because we compare the performance of two classifiers at each cross-validation run, i.e. for the same training and test sets, thus forming meaningful pairs. For the t-test, statistical significance level $\alpha = 0.05$ is used. In this setting, the paired t-test is not the most powerful method to use [55, 56], however, it will give us an idea of whether RSM or PRSM are better than the base classifiers, per dataset.

4.4 Results

We conducted a number of experiments using RSM and PRSM with different parameters and on several datasets. Here, we discuss all of the results, however, only the results for the Prime Vision dataset are presented due to space considerations. The full results can be found in the appendix.

4.4.1 Parameters RSM

Prime Vision dataset

The results for the Prime Vision dataset are shown in figures 4.1, 4.2 and 4.3. If we compare the results of the RSM methods to the base classifier, indicated by 'default', the results are disappointing. For the Gaussian and 1-NN classifiers, RSM is not able to outperform the base classifier. For the k -Means classifier, RSM does improve on the base classifier. However, the improved performance is still lower than the performances of the other classifiers.

Our expectation that using more subspaces will increase the accuracy, was not entirely correct. This is only true when only very few subspaces (up to 20) are used. After that, the performance is influenced a lot by the combining rule used. For `meanc` and `maxc` the performance stays more or less constant. For `prodc`, the performance decreases and for `votec`, it increases. The performances using voting are also the most predictable in terms of the graphs it produces. However, RSM with voting is never able to outperform the base classifiers. This is surprising because voting is often used to combine traditional classifiers.

The combining rule also influences the optimal number of features per subspace. For `meanc` and `prodc`, more features per subspace results in a better performance, whereas for `votec`, it is better to use less features. This suggests that `meanc` and `prodc` benefit from more accurate classifiers, whereas `votec` is more effective in combining less accurate classifiers.

Both of these observations are true for all classifiers in the experiment. This suggests that the parameters of RSM may be more important than the choice of the base classifier, at least for these one-class classifiers.

Other datasets

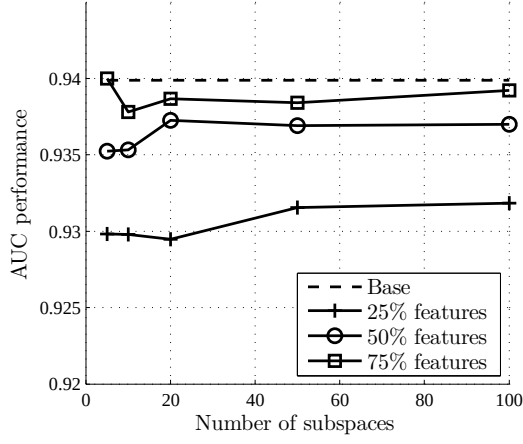
The results of the other datasets can be found in appendix A. A quick look at the results as a whole leads to the following observations:

- For some datasets, RSM is able to significantly outperform the base classifiers.
- The influence of parameters on performance is not the same across datasets.

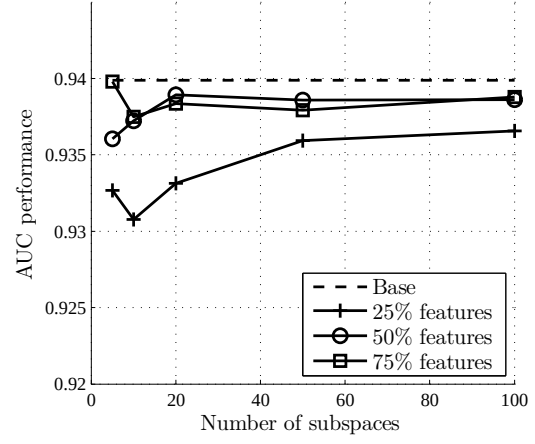
Despite the limited success of RSM on Prime Vision data, RSM can improve the performance on other datasets significantly. The performances on Hepatitis, Imports and Spam datasets are increased the most (sometimes the AUC is increased by 0.2, such as from 0.6 to 0.8). The improvements on Ionosphere, Sonar and Concordia datasets are more modest, however, significant differences are still often found.

Table 4.2: AUC of RSM, Prime Vision dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

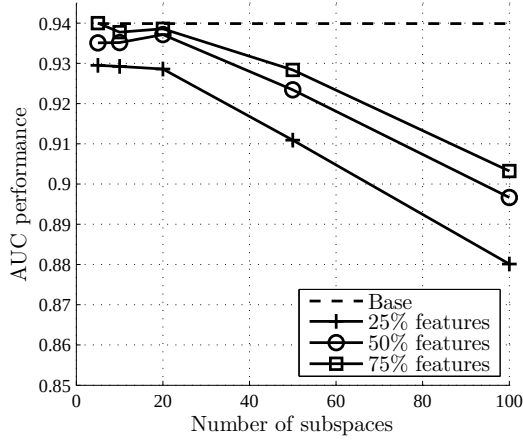
		Subspaces					
Classifier		2	5	10	20	50	100
Gauss, 94.0 (1.4)	25% mean	<i>92.6 (1.7)</i>	<i>93.0 (1.6)</i>	<i>93.0 (1.6)</i>	<i>92.9 (1.6)</i>	<i>93.2 (1.6)</i>	<i>93.2 (1.6)</i>
	25% prod	<i>92.5 (1.7)</i>	<i>93.0 (1.6)</i>	<i>92.9 (1.6)</i>	<i>92.9 (1.7)</i>	<i>91.1 (2.2)</i>	<i>88.0 (2.9)</i>
	25% vote	<i>84.1 (3.2)</i>	<i>87.4 (2.9)</i>	<i>88.7 (2.6)</i>	<i>90.1 (2.4)</i>	<i>91.3 (2.3)</i>	<i>91.7 (2.2)</i>
	25% max	<i>92.6 (1.7)</i>	<i>93.3 (1.7)</i>	<i>93.1 (1.6)</i>	<i>93.3 (1.6)</i>	<i>93.6 (1.6)</i>	<i>93.7 (1.6)</i>
	50% mean	<i>93.5 (1.5)</i>	<i>93.5 (1.5)</i>	<i>93.5 (1.5)</i>	<i>93.7 (1.5)</i>	<i>93.7 (1.5)</i>	<i>93.7 (1.5)</i>
	50% prod	<i>93.4 (1.5)</i>	<i>93.5 (1.5)</i>	<i>93.5 (1.5)</i>	<i>93.7 (1.5)</i>	<i>92.3 (1.9)</i>	<i>89.7 (2.3)</i>
	50% vote	<i>84.4 (3.0)</i>	<i>86.4 (2.8)</i>	<i>87.9 (2.6)</i>	<i>88.8 (2.6)</i>	<i>89.2 (2.6)</i>	<i>89.6 (2.6)</i>
	50% max	<i>93.5 (1.6)</i>	<i>93.6 (1.5)</i>	<i>93.7 (1.5)</i>	<i>93.9 (1.5)</i>	<i>93.9 (1.4)</i>	<i>93.9 (1.4)</i>
	75% mean	94.0 (1.4)	94.0 (1.4)	<i>93.8 (1.5)</i>	<i>93.9 (1.5)</i>	<i>93.8 (1.5)</i>	<i>93.9 (1.4)</i>
	75% prod	94.0 (1.4)	94.0 (1.4)	<i>93.8 (1.5)</i>	<i>93.9 (1.5)</i>	<i>92.8 (1.9)</i>	<i>90.3 (2.2)</i>
	75% vote	<i>84.6 (3.1)</i>	<i>85.6 (3.1)</i>	<i>86.2 (2.8)</i>	<i>86.7 (2.7)</i>	<i>87.0 (2.7)</i>	<i>87.5 (2.6)</i>
	75% max	94.0 (1.4)	94.0 (1.4)	<i>93.7 (1.5)</i>	<i>93.8 (1.5)</i>	<i>93.8 (1.5)</i>	<i>93.9 (1.5)</i>
1-NN, 96.2 (1.3)	25% mean	<i>95.0 (1.5)</i>	<i>95.5 (1.5)</i>	<i>95.8 (1.4)</i>	<i>95.6 (1.5)</i>	<i>95.9 (1.4)</i>	<i>95.9 (1.3)</i>
	25% prod	<i>95.0 (1.5)</i>	<i>95.4 (1.5)</i>	<i>95.7 (1.4)</i>	<i>95.5 (1.5)</i>	<i>95.3 (1.8)</i>	<i>93.5 (2.0)</i>
	25% vote	<i>90.7 (2.1)</i>	<i>93.1 (2.1)</i>	<i>94.4 (2.0)</i>	<i>94.9 (1.6)</i>	<i>95.6 (1.4)</i>	<i>95.9 (1.3)</i>
	25% max	<i>95.0 (1.4)</i>	<i>95.2 (1.5)</i>	<i>95.6 (1.3)</i>	<i>95.5 (1.4)</i>	<i>95.7 (1.2)</i>	<i>95.8 (1.2)</i>
	50% mean	<i>95.3 (1.5)</i>	<i>95.8 (1.3)</i>	96.2 (1.2)	<i>96.0 (1.3)</i>	96.2 (1.3)	96.2 (1.3)
	50% prod	<i>95.3 (1.5)</i>	<i>95.8 (1.4)</i>	96.1 (1.3)	<i>96.0 (1.4)</i>	<i>96.0 (1.4)</i>	<i>94.8 (1.7)</i>
	50% vote	<i>90.4 (2.4)</i>	<i>92.6 (2.0)</i>	<i>93.9 (1.8)</i>	<i>94.4 (1.6)</i>	<i>95.0 (1.5)</i>	<i>95.1 (1.5)</i>
	50% max	<i>95.4 (1.5)</i>	<i>95.9 (1.3)</i>	96.1 (1.3)	<i>96.1 (1.3)</i>	96.1 (1.3)	96.1 (1.2)
	75% mean	<i>96.1 (1.3)</i>	<i>96.0 (1.3)</i>	<i>96.1 (1.3)</i>	<i>96.0 (1.4)</i>	<i>96.1 (1.3)</i>	96.2 (1.3)
	75% prod	<i>96.0 (1.3)</i>	<i>96.0 (1.3)</i>	<i>96.1 (1.3)</i>	<i>96.0 (1.4)</i>	<i>96.0 (1.4)</i>	<i>95.0 (1.7)</i>
	75% vote	<i>91.2 (2.2)</i>	<i>92.4 (1.8)</i>	<i>92.7 (1.8)</i>	<i>93.5 (1.8)</i>	<i>93.9 (1.7)</i>	<i>94.2 (1.7)</i>
	75% max	<i>96.1 (1.3)</i>	<i>96.0 (1.3)</i>	<i>96.0 (1.3)</i>	<i>96.0 (1.4)</i>	96.1 (1.3)	96.1 (1.3)
<i>k</i> -Means, 92.3 (1.9)	25% mean	<i>91.1 (2.3)</i>	92.3 (2.1)	92.6 (2.0)	92.7 (2.0)	93.1 (1.8)	93.1 (1.8)
	25% prod	<i>91.1 (2.3)</i>	92.3 (2.1)	92.6 (2.0)	92.7 (2.0)	93.1 (1.9)	<i>91.5 (2.5)</i>
	25% vote	<i>81.2 (3.1)</i>	<i>87.6 (3.0)</i>	<i>89.6 (2.8)</i>	<i>90.7 (2.4)</i>	<i>91.7 (2.3)</i>	92.1 (2.3)
	25% max	<i>91.2 (2.3)</i>	92.2 (2.2)	92.4 (1.9)	92.3 (1.8)	92.8 (1.8)	92.9 (1.8)
	50% mean	<i>91.9 (2.2)</i>	92.5 (2.0)	92.7 (1.9)	92.9 (1.9)	93.1 (1.9)	93.1 (1.9)
	50% prod	<i>91.9 (2.2)</i>	92.5 (2.0)	92.7 (1.9)	92.9 (1.9)	93.1 (1.9)	<i>92.0 (2.5)</i>
	50% vote	<i>81.8 (2.7)</i>	<i>86.1 (2.8)</i>	<i>87.6 (2.6)</i>	<i>88.9 (2.4)</i>	<i>90.2 (2.5)</i>	<i>90.6 (2.5)</i>
	50% max	<i>91.9 (2.1)</i>	92.5 (2.0)	92.6 (2.0)	92.8 (2.0)	93.1 (1.9)	93.0 (1.9)
	75% mean	92.4 (2.1)	92.8 (2.0)	92.8 (2.0)	92.8 (2.0)	92.9 (1.9)	93.0 (1.9)
	75% prod	92.4 (2.1)	92.8 (2.0)	92.8 (2.0)	92.8 (2.0)	92.9 (1.9)	<i>91.9 (2.5)</i>
	75% vote	<i>82.6 (3.6)</i>	<i>85.6 (3.4)</i>	<i>86.7 (2.9)</i>	<i>87.3 (2.9)</i>	<i>88.2 (2.9)</i>	<i>89.0 (2.8)</i>
	75% maxc	92.4 (2.1)	92.9 (2.0)	92.7 (2.0)	92.8 (2.0)	92.9 (2.0)	93.0 (1.9)



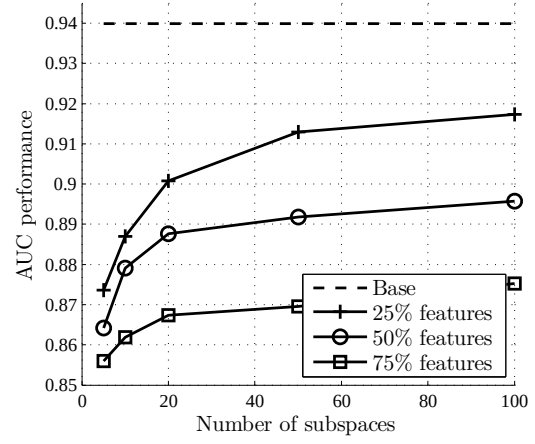
(a) mean



(b) max



(c) prod



(d) vote

Figure 4.1: Prime Vision, Gauss OCC

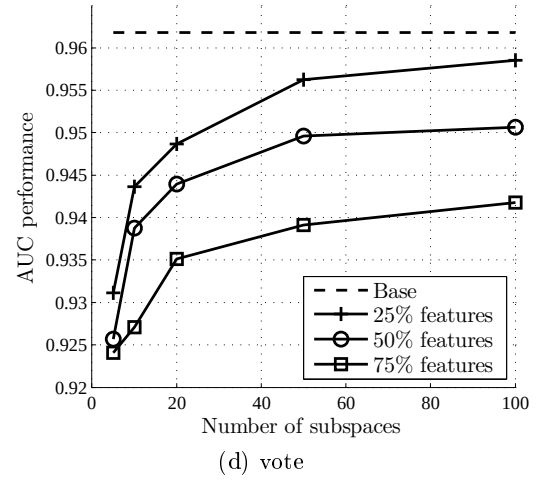
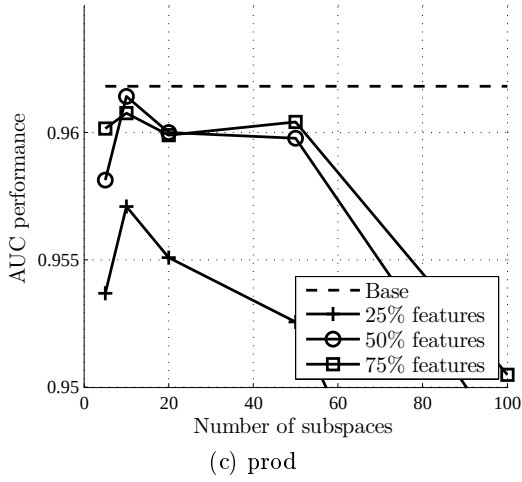
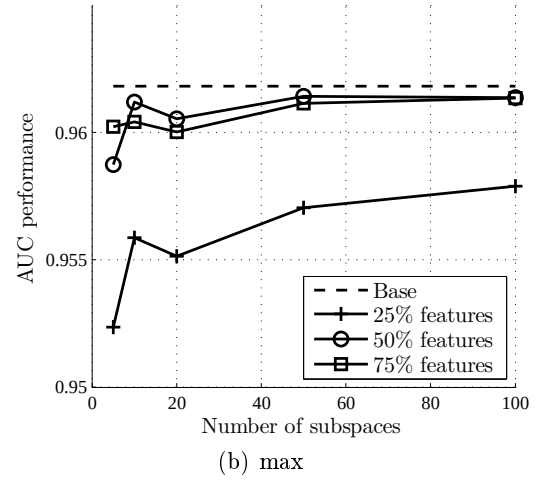
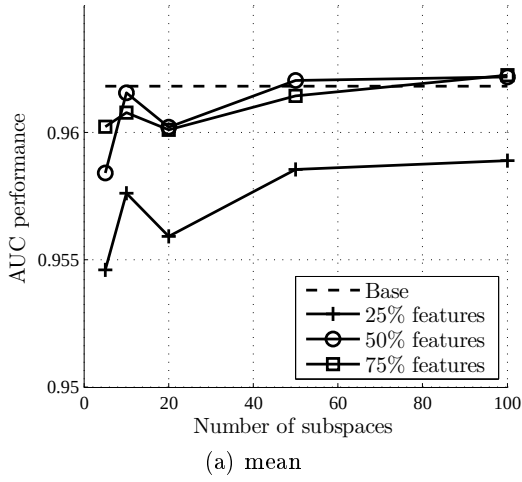


Figure 4.2: Prime Vision, 1-NN OCC

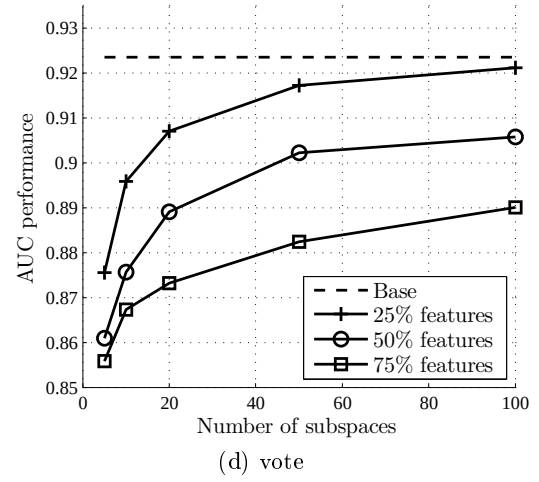
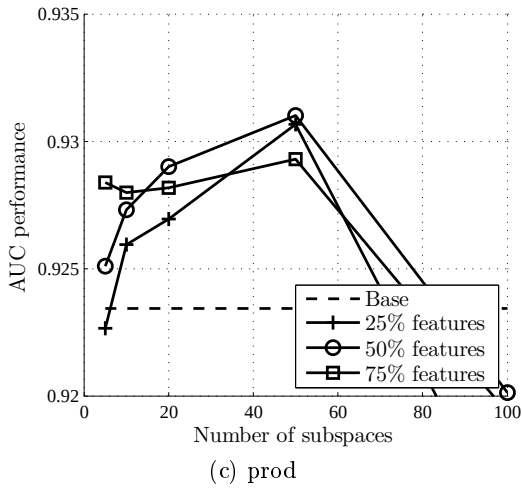
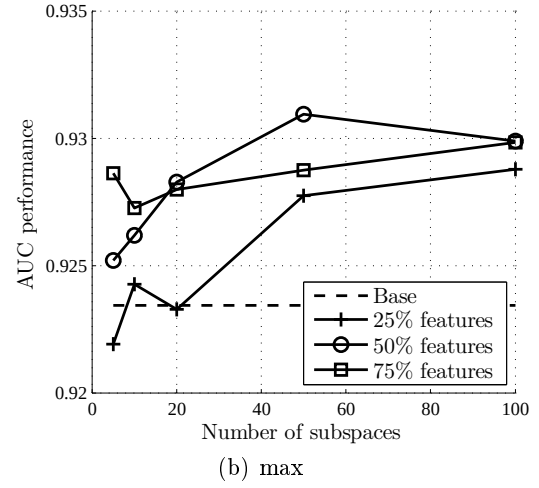
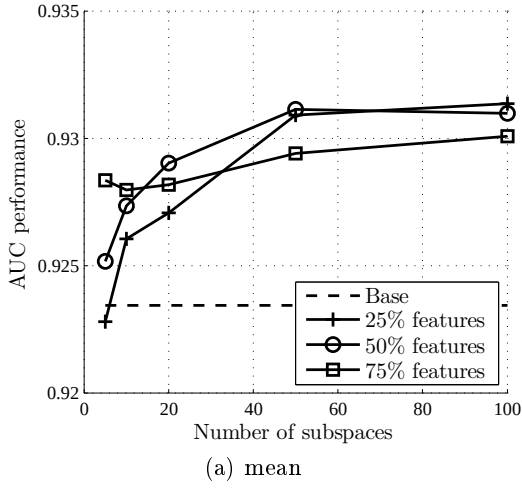


Figure 4.3: Prime Vision, k -Means OCC

Our findings about the effect of the number of subspaces on the different combining rules are not true for all of the datasets. In particular, the product and voting rules do not always display the same behavior. For instance, for Hepatitis, Imports and Spam datasets, RSM with voting is able to outperform the base classifiers significantly. Similarly, the performance with the product rule does not always degrade at higher numbers of subspaces. This can be observed in the Ionosphere, Sonar and Concordia datasets, particularly for the k -Means classifier.

However, it still holds that the mean and maximum rules are the least influenced by the number of subspaces, as we already observed with Prime Vision data. Also, it seems to be clear that it is definitely not an advantage to use as many subspaces as possible. It also seems to be the case that the effect of adding extra subspaces to an ensemble is unpredictable, as most graphs contain a lot of “jumps”.

Another puzzle is the influence of the number of feature per subspace on the performance. Some datasets (Hepatitis, Spam, Imports) clearly benefit from smaller subspaces, while for others (Ionosphere, Concordia, Sonar) there is no clear relationship. However, in these cases, it seems reasonable to use 50% of the features as this usually produces adequate results.

4.4.2 Parameters PRSM

Prime Vision dataset

The results of the first experiment are shown in Fig. 4.2. The performance of PRSM is plotted at different sizes of the final ensemble. Note that at 100 classifiers, PRSM reduces to RSM because all of the classifiers are selected for the ensemble.

The results indicate that PRSM is able to outperform the base classifier in cases where RSM was not able to do so (Gaussian and 1-NN classifiers). Furthermore, in all the plots PRSM outperforms RSM, although the improvement is very slight.

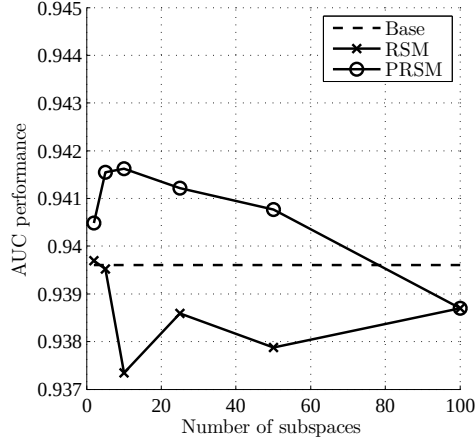
An important observation is that PRSM produces more predictable graphs than RSM, i.e. there are less “jumps” in performance. This suggests that sorting the subspaces by their performance is an adequate way to predict how accurate the overall ensemble will be. Here, we see that up to 25 subspaces in PRSM are optimal for all classifiers, whereas for RSM it is more difficult to pick such a optimal value because of the “jumps” in performance.

In the second part of the results we demonstrate how increasing the “pool size” of subspaces affects the performance of PRSM. We can see that, contrary to our expectations, the improvement from using more subspaces in the “pool” is very minimal. Furthermore, it is not more advantageous to select “more out of more” subspaces, as in the graph, using 10% of all the subspaces (i.e. 10 out of 100, 25 out of 250 and 100 out of 1000) produces almost the same result.

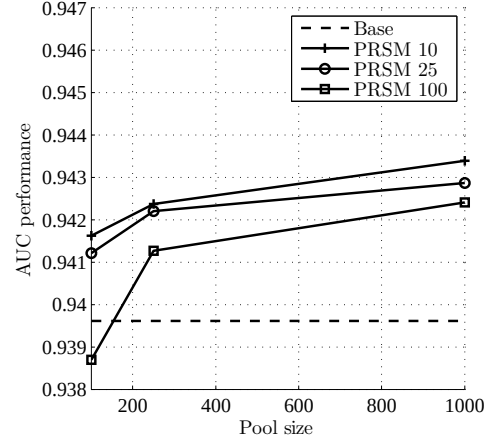
Other datasets

The results of PRSM experiments on the additional datasets are shown in appendix A.

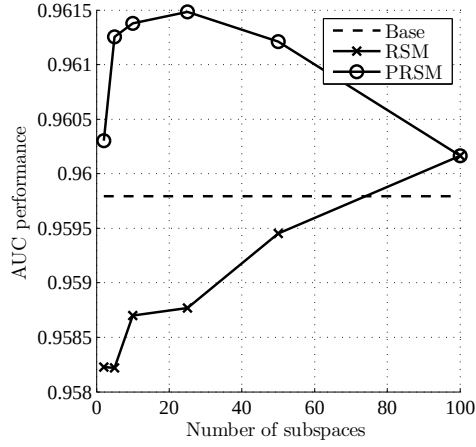
There is a noticeable pattern in the performances of PRSM as compared to RSM. Overall, PRSM outperforms RSM. Just as for the Prime Vision dataset, PRSM tends to produce more



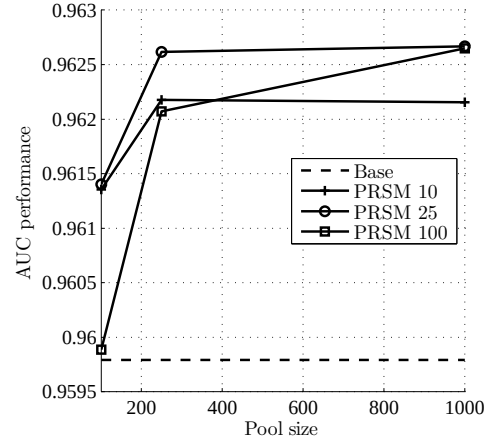
(a)



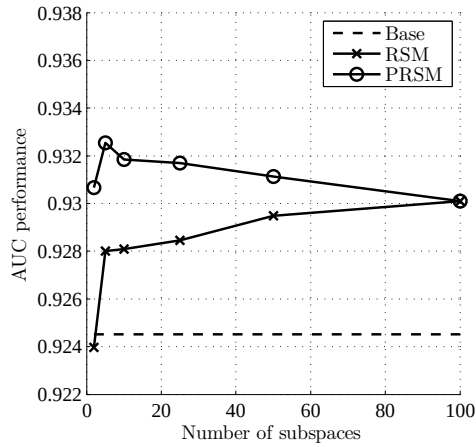
(b)



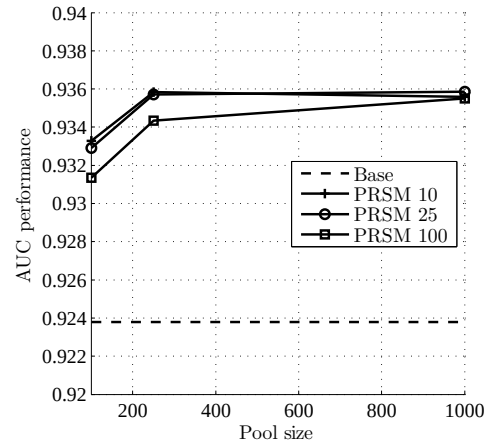
(c)



(d)



(e)



(f)

Figure 4.4: Results for PRSM parameters experiments on Prime Vision data

regular, “convex” graphs than RSM. Furthermore, we notice that it is a reasonable choice to use 10 to 25 subspaces because this produces the best results for most experiments.

Increasing the pool size for PRSM may improve performance slightly. However, for most datasets, using just 100 subspaces to select from was not statistically worse than using a larger pool size. Only with the Spam dataset, using a larger pool size led to significant improvements.

We can conclude that in most cases, although a larger pool size can slightly improve performance, it is sufficient to use a pool size of 100. This number is able to produce subsets that, together with the mean combining method, can form an effective classifier. We suppose that although with a larger pool size, we would be able to find subspaces which are more accurate individually, these subspaces would probably be less diverse and therefore only slightly contribute to the success of the combined classifier.

4.5 Conclusion

To answer the research question posed in the beginning this chapter, we can say that *the performance of one-class classifiers can be improved by RSM*. However, the following things need to be taken into account:

- The performance improvement depends on the dataset.
- The performance improvement depends on the parameters used.
- There are no “best” parameters.

Although we have seen that there is no best parameter set, there are some general observations that can be made about the various parameters we tested. First of all, we have seen that it is not always an advantage to use more subspaces, as we expected. The optimal number of subspaces depended on the combining rule used. Namely, for product rule, it was better to use very few subspaces, whereas for the voting rule, it was better to use a lot of subspaces. For the mean rule, the performance varied the least depending on the number of subspaces *and* the performance was quite good in most cases.

Secondly, we have seen that it is very difficult to pick a good number of features per subspaces. Some datasets benefited from using small subspaces, while in other datasets larger subspaces led to better results. In order to be able to apply RSM to real-life problems, we would need a way to determine which setting to use for the number of features: either by testing different settings in advance, or by deriving this number from some properties of the data.

The other classifier in this investigation, PRSM, was also able to outperform the base classifiers, and overall it performed better than RSM. This suggests that it is possible to build smaller, but more accurate ensembles.

For PRSM, we only tested the size of possible subspaces and the size of selected subspaces as parameters. The other parameters were set to reasonable values from the RSM experiments. For these parameters, we have found it was easier to set the size of the ensemble for PRSM than for RSM because the performance of PRSM varied less as new ensemble members were added. We have

found that combining 10 out of 100 subspaces was sufficient to obtain a reasonable performance for each dataset, whereas for RSM, it was more difficult to pick such a value.

As for the other parameter we investigated, we have seen no clear indication that increasing the number of initial subspaces has a significant effect on the performance of PRSM. Using 100 subspaces in total seems to be sufficient, however, if the resources allow it, it would not be a disadvantage to include more subspaces.

So, although significant improvements in performance can be achieved with RSM and PRSM, these are not straightforward methods to apply and there is no guarantee that the performance will be improved, as good parameters are needed for this. Ideally, we would want to be able to determine in advance what type of dataset would benefit from using subspace ensembles, and how to pick good parameters in order for the ensemble to be useful. We will examine these issues in more detail in chapter 5.

Chapter 5

Further Analysis

The previous chapter has focused on the performance of RSM and PRSM in terms of accuracy. We found that although RSM and PRSM are able to outperform one-class classifiers, it very much depends on the dataset how big this improvement may be. In particular, we observed that there was very little improvement on the Prime Vision data, while the improvement on the other datasets was more significant.

In this chapter we take a closer look at the reasons for these differences in performance. The question we want to answer here is, “*When is it worthwhile to apply RSM or PRSM to one-class classifiers?*” or “*Why are RSM and PRSM not always effective?*”.

In order to answer this question, we first examine the differences between the datasets for which RSM had no effect and the datasets for which RSM had significant improvements. We also discuss whether diversity is an important issue in ensembles, and thus whether the classifiers would improve if diversity is taken into account. Lastly, we compare RSM and PRSM to other ensembles which can be applied to other one-class classifiers in order to determine whether the benefits of these classifiers are due to their ensemble properties (if other ensembles perform equally well), or due to using subspaces (if other ensembles are less effective).

5.1 Data-dependency

A possible explanation of the limited success of RSM on Prime Vision data is that the dataset was already in the “best possible condition” because we selected the feature set that resulted in the best performances for several classifiers. The other datasets, on the other hand, were not processed in any way. Therefore, it may well be that processing the features in a simple way (such as by normalization) may also result in significant improvements in performance.

Thus, we hypothesise that RSM is able to benefit from “difficult” situations. When the situation is “easy”, a single classifier is also able to handle it, so there is no benefit from using an ensemble classifier. In other words, our main hypothesis is: *If a classifier is able to model the data well in the full feature space, there is no improvement from RSM. If a classifier is not able to do so, significant improvements from RSM can be expected.*

However, how do we know if the classifier is not able to model the data well in the full feature space? There are a few possible reasons:

- There are too little positive samples per dimension to produce an accurate model of the data. Thus, only a few hypotheses h which can be formed for the data using a single classifier. RSM increases the performance because it is able to produce new models which are closer to the true hypothesis. This is the main reason RSM is advocated as an ensemble technique.
- There are large differences in the variances of the features and a single (scale-sensitive) classifier is not able to produce an accurate model of the data. By using subsets of features with smaller differences in the variances, the classifier is able to produce better models, which are then combined for an increased accuracy. In this case, we expect that for a scale-sensitive classifier, normalizing the data would also result in improved performance.
- “Quality” of the features. If many of the features are “noisy” or irrelevant, RSM could have the potential to select several relevant subspaces for improved classification. In this case, feature selection would probably also have a positive effect on the results.

Of course, these properties are not mutually exclusive, i.e. a dataset could be poorly sampled, unnormalized, “oddly-shaped” *and* contain a lot of irrelevant features, or have none of these properties.

Unfortunately, most of these properties are not straightforward to measure. For sampling, we simply calculate the ratio of target objects to dimensions. For variances in the data, we first calculated the differences in the ranges of the smallest and largest dimensions. However, this measure is not be able to represent the differences in the largest *directions* of variance of the data. This is why we instead chose to calculate the difference between the smallest and largest eigenvalues of the covariance matrix of the data. We do not provide a measure for the “quality” of the features.

To measure the success of RSM, we measure the relative AUC performance increase, i.e. if a_1 is the initial AUC and a_2 is the AUC after applying RSM, we measure $\delta = 100 * \frac{a_2 - a_1}{1 - a_1}$. This is to reflect that improving from 0.95 to 0.96 is more significant than improving from 0.55 to 0.56. Also note that the improvements we calculate represent the improvements we have encountered in our experiments, not an oracle measure of how much improvement is possible.

These measurements are presented in Table 5.1. Next to the datasets from the previous chapter, we have added the Diabetes dataset here in order to have an example of a very well sampled dataset. To obtain performance estimates for this dataset, we performed the same experiments on this dataset as in the previous chapter.

It is immediately clear that the sampling is not the only factor influencing the performance of RSM. Although RSM is expected to perform better in problems with low object/feature ratios, here we see that the (well-sampled) Spam dataset also shows a lot of improvement. However, if we examine the sampling together with the variability, the picture becomes clearer. For well-sampled (say, > 5 samples per feature) datasets, little improvement is possible when the variability of the data is low, as with Prime Vision data. For undersampled datasets, a lot of improvement is possible. However, when the variability of the data is low (as in Sonar dataset), the improvement is less than in cases with large variability.

Table 5.1: Characteristics of datasets. δ denotes the best relative AUC improvement in %. The datasets are sorted by sampling ratio.

Dataset	Sampling	Variability	δ Gauss	δ 1-NN	δ k -Means
Diabetes	62.5	$1 * 10^4$	9.9	11.1	6.5
Spam	31.8	$4 * 10^5$	62.7	64.1	61.8
Prime Vision	19.8	1.1	3.3	2.5	10.7
Ionosphere	6.6	2.9	21.6	7.0	38.5
Hepatitis	6.5	$8 * 10^3$	43.4	45.3	38.7
Imports	2.8	9.4	31.4	42.1	55.7
Sonar	1.9	0.6	3.3	9.2	19.9
Concordia	1.6	3.0	43.7	22.2	27.9

Of course, this is not a clear-cut measure of whether RSM or PRSM is a good method to use because the actual relationship is probably much more complex and other factors contribute as well. Nevertheless, we attempt to verify this hypothesis with a few isolated experiments. We vary the sampling and the variability of the data (by normalization) and investigate whether the effect of these factors still holds in these new situations.

5.1.1 Experimental setup

Sampling

First, we test PRSM on undersampled Prime Vision, Diabetes and Ionosphere datasets. We expect that there will be more significant improvements in performance because the base classifiers will suffer from the curse of dimensionality and RSM /PRSM are better at dealing with this problem.

We perform 5x stratified 10-fold cross-validation. At each fold, the training sets are subsampled to 50, 20 and 25 objects for Prime Vision, Diabetes and Ionosphere respectively. The test sets remain the same in order to obtain a good picture of the differences in performance.

In the experiments in the previous chapter we have seen that all of these datasets performed better when larger feature subsets (50% of the features) are used. We suspect that this is the case because the datasets are well sampled. Therefore, for the subsampled versions, we use 25% of the features per subspace in order to achieve better sampling in each subspace. The number of subspaces is varied to obtain a few results for each dataset. The best result for each dataset is then recorded to indicate the best performance improvement.

Note that in this experiment we do not take into account the performances on the original datasets that we have obtained so far, because much more experiments have been done on the original datasets which increases the chances of obtaining a very good result.

Normalization

We also test PRSM on normalized Diabetes, Hepatitis and Spam data. Normalization directly affects the variances in the data, but also reduces the variability measure that we have selected. We expect that the best performance improvement will be lower than what we are observing with original versions of these datasets.

We use the same parameters for both versions of the dataset and vary the number of subspaces to obtain several results. The results of the base classifiers and the best results per dataset are recorded to calculate the best performance improvement. Again, we only use results from this experiment rather than taking into account the performances that we have obtained earlier.

Irrelevant features

We investigate how RSM and PRSM compare to the base classifier in datasets where many features are irrelevant to the classification. To do so, we add an increasing percentage of Gaussian distributed ($\mu = 0, \theta^2 = 1$) features to the original datasets. This is done only for normalized datasets so that the features are of the same scale as the other features: Prime Vision, Ionosphere, Imports and Sonar are used in the experiments. For each extended dataset, we test three classifiers: the base classifier, RSM and PRSM. We expect that RSM and PRSM will be better at dealing with large feature sets and that their performance will degrade less than that of the base classifier.

In this experiment, we use 2x stratified 5-fold cross-validation due to the lengthy training and evaluation times for the extended datasets.

5.1.2 Results

Sampling

The results of the sampling experiment are presented in Table 5.2. Contrary to our expectations, there is less improvement made for the subsampled datasets. This suggests that for these datasets, even with a few samples the base classifiers are still able to model the data relatively well. This also strengthens our suspicions about more factors than sampling alone contributing to the success of RSM.

On the other hand, it is also possible that the way we conduct the experiment influences the result. For instance, it is possible that the number of features used each dataset was not optimal for the subsampled version and more improvement could still be achieved.

Normalization

The performances and relative improvements of the normalization experiment are presented in Table 5.3. If we look at the performance improvements, we indeed notice that these are lower than the improvements for the original datasets. In fact, for two out of three classifiers on the normalized Diabetes dataset, the performances actually decrease if PRSM is used.

Table 5.2: Comparison of improvement between original and subsampled datasets. Next to the dataset, we provide the sampling in both situations. Base denotes the AUC of the base classifier, (P)RSM denotes the best obtained AUC with RSM or PRSM. δ denotes the relative improvement in AUC in %. The larger improvements are indicated in bold.

Data	Classifier	Original			Subsampled		
		Base	(P)RSM	δ	Base	(P)RSM	δE
PV 19.8/0.7	Gauss	93.9 (1.5)	94.1 (1.5)	3.3	92.3 (2.5)	92.6 (2.6)	3.9
	1-NN	96.0 (1.4)	96.2 (1.3)	5.0	92.3 (2.3)	92.5 (2.3)	2.6
	k -Means	92.3 (1.8)	93.4 (1.7)	14.3	91.4 (2.3)	92.1 (2.1)	8.1
Diabetes 62.5/2.2	Gauss	70.8 (5.9)	73.6 (6.1)	8.2	68.2 (7.0)	70.3 (7.2)	6.6
	1-NN	66.8 (5.8)	71.2 (5.6)	13.3	66.3 (7.1)	68.6 (6.8)	6.8
	k -Means	65.4 (7.0)	67.3 (8.0)	5.5	64.5 (7.5)	66.0 (8.7)	4.2
Ionosphere 6.6/0.8	Gauss	96.4 (3.0)	97.0 (2.9)	16.7	96.1 (3.1)	96.6 (3.0)	10.3
	1-NN	95.8 (3.3)	96.2 (3.2)	9.5	95.8 (4.0)	95.9 (4.0)	2.4
	k -Means	96.2 (2.9)	97.6 (2.5)	36.8	95.1 (3.6)	96.5 (3.0)	28.6

Table 5.3: Comparison of improvement between original and normalized datasets. Next to the dataset, we provide the sampling in both situations. Base denotes the AUC of the base classifier, (P)RSM denotes the best obtained AUC with RSM or PRSM. δ denotes the relative AUC improvement reduction, in %. The larger improvements are indicated in bold.

Data	Classifier	Original			Normalized		
		Base	(P)RSM	δ	Base	(P)RSM	δ
Diabetes $1 * 10^4/0.1$	Gauss	70.6 (5.0)	73.5 (5.3)	9.9	73.9 (5.2)	74.2 (5.2)	1.2
	1-NN	66.9 (6.7)	70.6 (6.5)	11.2	71.7 (6.4)	67.2 (7.4)	-15.9
	k -Means	64.8 (7.4)	67.1 (6.6)	6.5	71.5 (5.7)	71.4 (7.5)	-0.4
Hepatitis $8 * 10^3/0.2$	Gauss	63.8 (16.9)	79.5 (14.2)	43.4	82.2 (13.0)	83.1 (13.0)	5.1
	1-NN	58.9 (17.4)	77.5 (14.6)	45.3	78.3 (14.4)	80.1 (12.9)	8.3
	k -Means	51.8 (21.7)	79.0 (13.2)	56.4	76.5 (13.1)	84.2 (10.3)	32.8
Spam $4 * 10^5/0.1$	Gauss	63.3 (3.1)	86.3 (1.8)	62.7	82.0 (1.9)	82.5 (1.9)	2.8
	1-NN	65.2 (2.6)	87.5 (1.8)	64.1	76.5 (2.5)	81.9 (2.1)	23.0
	k -Means	49.0 (2.5)	80.5 (2.0)	61.8	49.7 (2.8)	63.9 (2.8)	28.2

The question jumps to mind whether it is perhaps better to apply a single base classifier on normalized data rather than to use subspaces. This seems to be the case for the Diabetes and Hepatitis datasets. On the other hand, for the Spam dataset PRSM still does better than a single classifier on normalized data. Another interesting observation is that for the Hepatitis dataset, applying PRSM on normalized data even further improves performance. In fact, for all three classifiers, PRSM on normalized data produces the best results.

These results are somewhat in line with our intuitions about the advantages of PRSM. With the “variability” factor being taken care of, PRSM has nothing to contribute to the Diabetes dataset because it is well-sampled and all the classifiers are able to model the normalized data well. However, we would then also expect the Spam dataset to have no further improvements. However, these improvements can still be observed. Perhaps here, another undiscovered advantage of PRSM is being exploited. On the other hand, the Hepatitis dataset is not very well-sampled like the Diabetes and Spam datasets. So although the variability factor is taken care of, PRSM is able to even further improve the performance.

Therefore, we can conclude that although PRSM is relatively less effective in normalized datasets, it can still be worthwhile to use PRSM in order to achieve better performances.

Irrelevant features

The results of the classifiers in presence of irrelevant features are shown in Figs. 5.1 and 5.2.

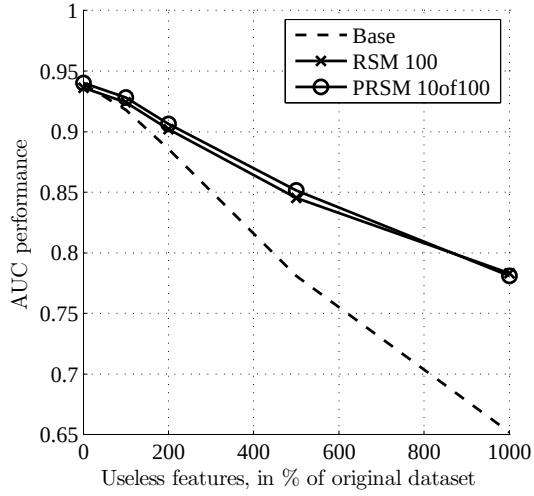
For Prime Vision and Ionosphere datasets, where the improvement for the Gaussian classifier from using subspaces was not very significant, we see that in presence of irrelevant features, the performance of the base classifier degrades a lot while this is not the case for RSM and PRSM. However, for both Imports and Sonar datasets, RSM and PRSM do not seem to have any advantages in the presence of irrelevant features. For the k -Means classifier, all the classifiers degrade in the presence of irrelevant features, i.e. there are also no advantages with using RSM and PRSM.

It is difficult to conclude something based on these results except that this percentage of irrelevant features is not a factor that can directly say something about the success of RSM or PRSM. In fact, the way the datasets are generated also influences the sampling of the data, so perhaps a better experiment would be to replace “good” features by irrelevant features, rather than add extra features to the data.

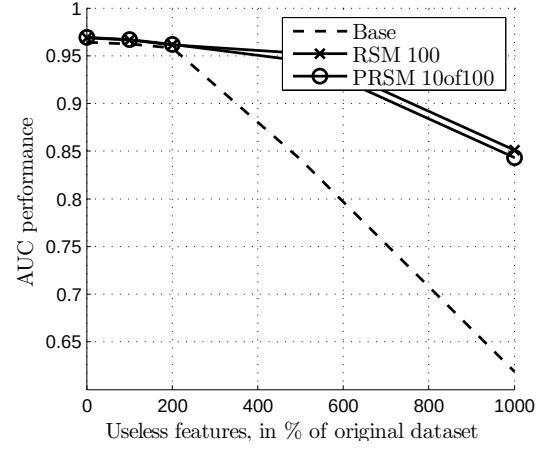
5.2 Diversity

Another possible argument for why PRSM does not always perform very well may be that there is no focus on diversity, which is often thought to be important for ensembles [15, 16]. Others have not been able to find this relationship in real-life data [14]. We agree that there is no point in combining identical classifiers, however, we expect that the random generation of subspaces still should produce sufficiently different subspaces, i.e. perhaps a special focus on diversity is not necessary.

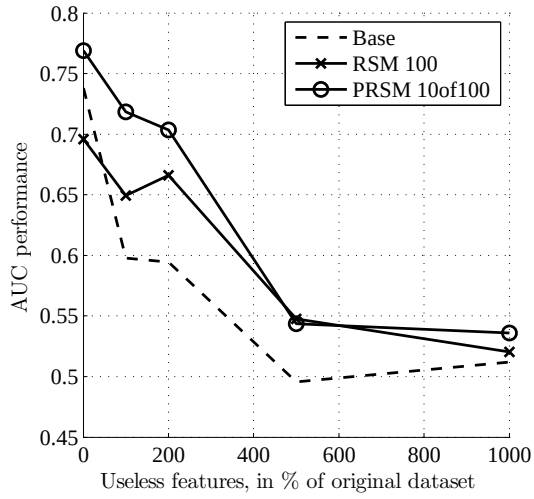
It is not straightforward to build a classifier which incorporates diversity into its evaluation procedure. On the one hand, we would have to switch from a ranking-based pruning in PRSM to



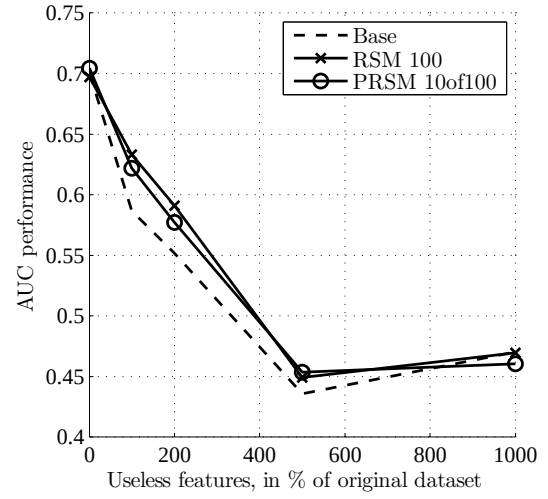
(a) Prime Vision



(b) Ionosphere

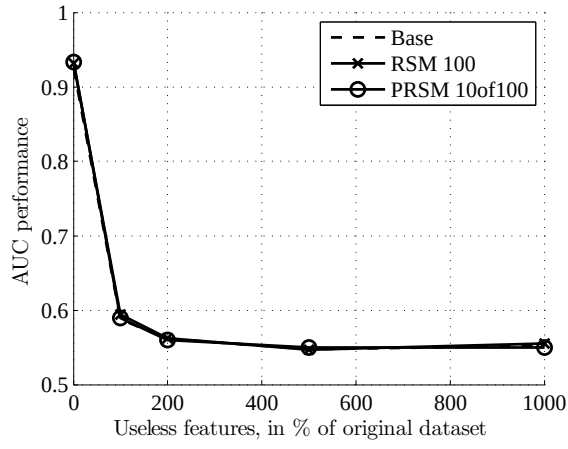


(c) Imports

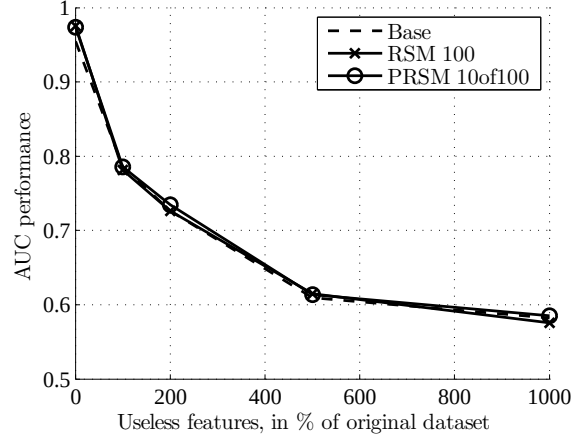


(d) Sonar

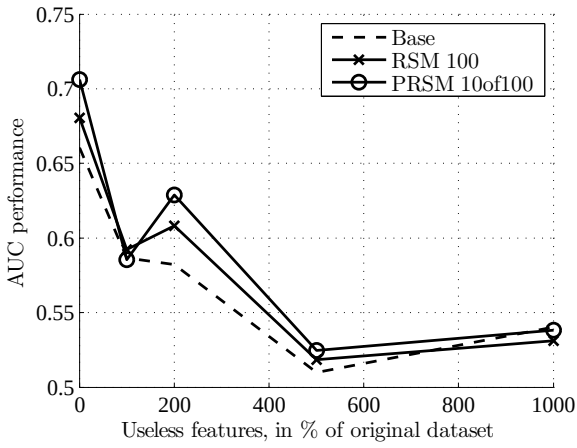
Figure 5.1: Performance in presence of useless features, Gauss



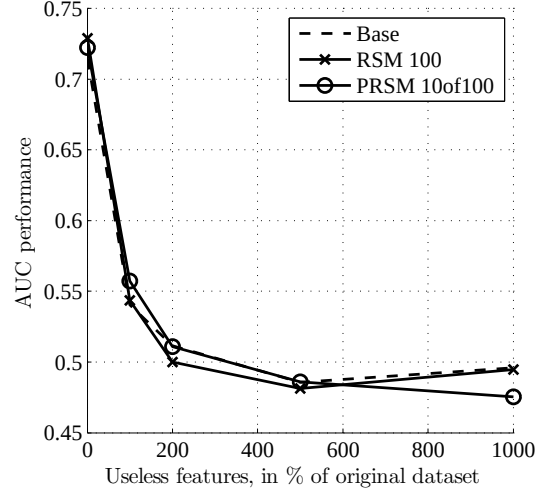
(a) Prime Vision



(b) Ionosphere



(c) Imports



(d) Sonar

Figure 5.2: Performance in presence of useless features, k -Means

a search-based pruning, because diversity is about the ensemble as a whole. On the other hand, deriving a good evaluation function containing both accuracy and diversity is not easy because we do not know the relative importance of these factors. In fact, in [16] this relationship was found to vary across datasets.

Before any such attempts are made, we decide to examine the relationship between diversity and ensemble performance. If such a relationship exists, it would be a signal to create a classifier which would take diversity into account.

We conduct the following experiment. For each dataset, we generate 100 subspaces with a fixed number of features in total, and then randomly create 50 ensembles of 10 subspaces. We perform 2x stratified 5-fold cross-validation because we are not interested in a good performance estimate but a just a general relationship. In each run, we measure the average AUC of the individual classifiers, the maximum AUC of the individual classifiers, the diversity and the AUC of the ensemble using the mean rule. For each of the 50 ensembles, these values are then averaged over the 10 runs.

Diversity is measured using Q-statistic (Q), as recommended in [14]. We also decide to use a different measure to ensure that the possible relationship does not only depend on the way that we measured diversity. Therefore we also use the double-fault measure (DF), because it has a low correlation with Q while other diversity measures are more correlated [14]. If we let N_{00} denote the number of items for which two classifiers are wrong, N_{01} the number of items for which only the first classifier is wrong and so forth, the two diversity measures are defined as follows:

$$Q = \frac{N_{11} * N_{00} - N_{01} * N_{10}}{N_{11} * N_{00} + N_{01} * N_{10}} \quad (5.1)$$

$$DF = \frac{N_{00}}{N_{00} + N_{01} + N_{10} + N_{11}} \quad (5.2)$$

These quantities indicate the diversity of two classifiers. To measure the diversity of an ensemble, the diversities of all $\frac{L*(L-1)}{2}$ pairs of L classifiers are averaged.

We wish to demonstrate the following relationships:

- Averaged individual AUC vs ensemble AUC
- Diversity vs ensemble AUC.

We expect there to be a relationship between the individual performances of the subspaces and the performance of the ensemble. This correlation is also implied by the way these quantities are measured. However, we suspect that some ensembles with the same average accuracy are better than others. This unexplained part could be a result of better diversity in the better performing ensemble.

Thus, we also examine the relationship of diversity and the ensemble AUC. However, it would be wrong to look at these measures in an absolute way. Even if there is a relationship, it might be due to the correlation of the measures (just as in the case with average AUC and ensemble AUC). Instead, we proceed in a manner similar to the method in [14]. We measure the improvement an ensemble has over its best single classifier. This way, if diversity is an important factor, we will be able to see this irrespective of the average quality of the ensemble.

First we demonstrate the relationship between the average AUC and the ensemble AUC in Fig. 5.3. We see that ensembles perform better if the individual classifiers perform better. Furthermore, we see that the results which outperform the single best classifier can be found at either ends of the spectrum. This suggests that the success of these ensembles might lie in their diversity.

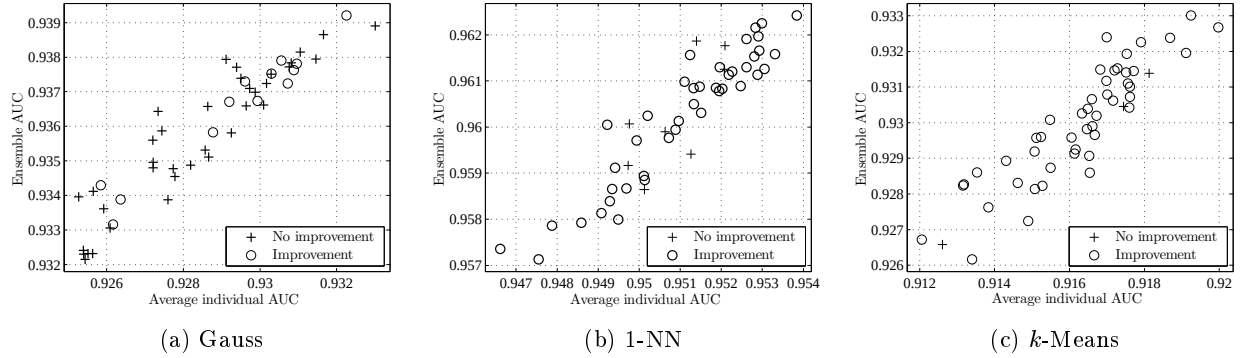


Figure 5.3: Relationship between average AUC and ensemble AUC for Prime Vision data.

Next we demonstrate the relationship between diversity and the ensemble AUC. The diversity plots for Q and DF are shown in Fig. 5.4. For Q, there does not seem to be a clear relationship with ensemble AUC. For DF, the relationship is more noticeable. However, even here, the fact that some ensembles are better than the single best, does not seem to depend on a high diversity (for DF, low values indicate high diversity).

We also do this for the other datasets, but instead of plots, we calculate correlation values of average AUC and the two diversity measures with the ensemble AUC and the improvement over single best. These results are presented in Table 5.4.

The average AUC and the ensemble AUC are highly correlated in all cases. The diversity measures are somewhat correlated with the ensemble AUC, however, this varies greatly per dataset and classifier. In some experiments the correlation is positive and not negative as in most other experiments. This situation is even worse if we look at the relationship between diversity and improvement over the best classifier. Therefore we cannot conclude that diversity leads to higher performances of ensembles.

These results do not point to a clear role of diversity for one-class classifiers. However, it is possible that the concept of diversity is not represented well by the diversity measures we used. Furthermore, it might be difficult to see a possible relationship because the ensembles in our experiment (given a dataset and a base classifier) were quite similar. Therefore we only have a small sample of the complete space of results: the ensembles are quite diverse because of the random choice of subspaces, and the individual classifiers are quite accurate. Perhaps if a bigger picture was available (including ensembles with very low diversity or with inaccurate classifiers) a trend could be noticed.

However, although a relationship between the concept of diversity and ensemble quality might exist, we feel that for our ensemble classifiers, incorporating a diversity measure would not be beneficial. If on the basis of a diversity measure, we cannot point out which ensembles will perform better and which ensembles will perform worse, using diversity to evaluate the selected classifiers in

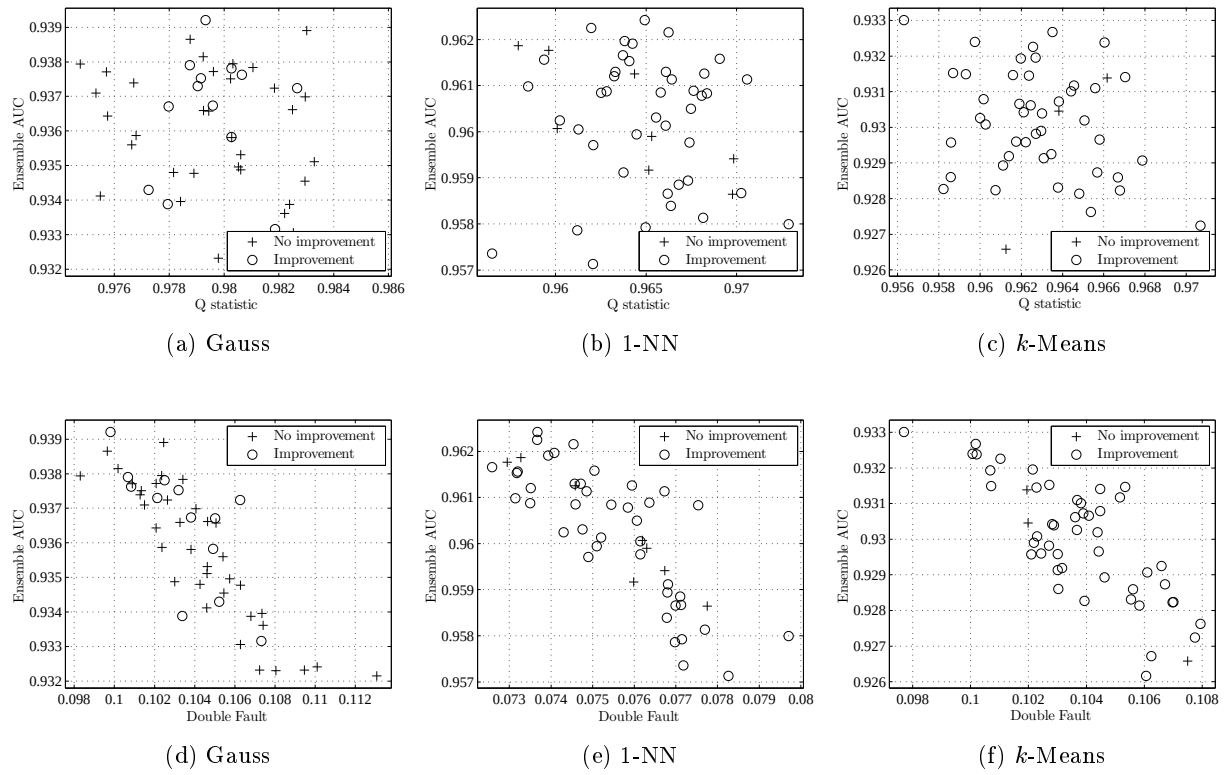


Figure 5.4: Relationship between diversity measures and ensemble AUC for Prime Vision data.

Table 5.4: Correlations (x100). Avg stands for average AUC, Ens stands for ensemble AUC, Q and DF stand for diversity measures, Imp stands for improvement over the best single classifier.

Data	Classifier	Measures				
		Avg Ens	Q Ens	DF Ens	Q Imp	DF Imp
Concordia	Gauss	88.1	43.9	-89.5	2.8	-5.9
	<i>k</i> -Means	96.4	-48.1	-90.1	-34.9	-24.7
	1-NN	96.9	-44.6	-93.6	-19.9	-28.6
Hepatitis	Gauss	87.7	5.5	-86.2	-9.5	-55.6
	<i>k</i> -Means	92.1	-27.3	-81.6	27.7	-13.8
	1-NN	83.8	-8.0	-11.6	4.9	-1.5
Imports	Gauss	96.8	-82.2	-90.9	-11.2	-24.6
	<i>k</i> -Means	95.3	-53.9	-75.2	-9.8	-16.3
	1-NN	89.1	-78.7	-84.7	-4.2	7.9
Ionosphere	Gauss	79.5	-1.3	3.6	-31.5	11.1
	<i>k</i> -Means	80.6	-43.2	-73.8	-39.5	-8.5
	1-NN	86.8	-33.7	-30.2	-0.8	4.2
Prime Vision	Gauss	94.0	-36.2	-85.3	-29.7	-47.4
	<i>k</i> -Means	90.0	-26.3	-78.6	-21.9	-9.4
	1-NN	92.2	-14.3	-81.5	7.4	4.7
Sonar	Gauss	92.3	-16.2	-75.5	6.7	-14.7
	<i>k</i> -Means	93.4	-35.8	-51.6	-1.6	-13.2
	1-NN	90.9	-22.0	-57.9	4.8	22.0
Spam	Gauss	76.2	-84.6	-87.3	-73.3	-68.4
	<i>k</i> -Means	88.7	-81.0	-82.3	-38.2	-37.0
	1-NN	89.9	-92.6	-97.0	-26.8	-26.5

PRSM would not necessarily lead to higher performance but would definitely add a computational overhead.

5.3 Comparison

5.3.1 Experimental setup

In this experiment we compare RSM and PRSM to a number of other classifiers: the base classifiers, base classifiers using the optimal feature set found by Forward Feature selection, and other ensembles which can be applied to any (thus also one-class) classifier. Note that although we can apply these methods to one-class classifiers, not all of these methods are strictly one-class themselves: forward feature selection, PRSM and AdaBoost also use information from the outlier class.

For RSM and PRSM, we use parameters which are found to be adequate in experiments in chapter 4. However, it is not possible to use different parameters for each dataset, as this would not be a statistically sound comparison. Therefore, before a comparison can be made, we need to establish which classifiers we want to compare, i.e. we need to either find parameters which are more or less suitable for any dataset, or to incorporate a way to determine parameters automatically.

As it is still not very clear how to choose good parameters (especially the number of features) for a dataset, we decide to choose a set of default parameters. Reasonable values for the number of subspaces (100 for RSM, 10 out of 100 for PRSM) and the combining method (averaging) were already found in the previous section. As for the number of features per subspace, we decided to use a *variable* number of features in the hope that this will be more appropriate for a range of datasets than using a fixed value.

The other ensembles that we use in the comparison are Bagging and AdaBoost. Descriptions of these methods are given in section 2.2. For both methods, we use their implementations from PRTools [54]. Except the base classifier, all other parameters are set to their default values: 100 classifiers and mean combining method for Bagging, and 100 classifiers and voting for AdaBoost. We denote this version of AdaBoost by AB-vote. Due to our findings from the RSM experiments, we also add a version of AdaBoost with the mean combining method to the comparison. This version is denoted by AB-mean.

To compare all of these methods, we do three comparisons in total: for the Gaussian, k -Means and 1-NN one-class classifiers. Although we did not see very large differences in effect of RSM or PRSM on these classifiers, it might be the case that such differences exist for other methods we are testing here.

In a comparison, we wish to use as many datasets as possible in order to be able to draw more statistically sound conclusions about the (absences of) differences between the classifiers. Therefore, we expand the collection of datasets that we use in the comparison. The extended list of datasets is presented in Table 5.5. Next to the information about target and outlier objects and the number of features, we include the best AUC as reported by [51] to create an idea of how much improvement is possible per dataset.

We modify the classifiers slightly from the versions that were used in the previous chapter. Instead of just using 90% of the training set for training and 10% for tuning, we now retrain the

Table 5.5: Datasets for the comparison experiment. The datasets with an * denote the datasets we have previously investigated.

Dataset	Targets	Outliers	Features	Best AUC
Cancer	47	151	33	63.3
Diabetes	500	268	8	75.6
Concordia*	400	3600	256	94.4
Glass	144	70	9	86.1
Heart	164	139	13	82.5
Hepatitis*	123	32	19	82.1
Housing	48	458	13	89.4
Imports*	71	88	25	87.6
Ionosphere*	225	126	34	97.8
Prime Vision*	1425	575	72	
Sonar*	111	97	60	84.9
Spam*	1813	2788	57	89.9
Spectf	95	254	44	95.8
Thyroid	93	3679	21	96.1
Vehicle	212	634	18	83.1
Vowel	48	480	10	88.8
Wave	300	600	21	93.0
Wine	71	107	13	95.1

classifiers on 100% of the training data after tuning has taken place.

For comparisons between multiple classifiers across a range of datasets, the paired t-test is not suitable [56]. In a statistical test, there is a probability that significance will be found when there is no significance. Because the paired t-test causes us to perform more comparisons ($\frac{L(L-1)}{2}$) than necessary, the probability of finding significance where there is none, is increased. Furthermore, the t-test only enables us to say something about how two classifiers compare to each other, not an overall evaluation of all classifiers in the comparison.

Therefore, we use the Friedman test [57], as recommended in [56] for such tasks. In this test, for each dataset the k algorithms are ranked according to their performance (1 is highest, k is lowest). The ranks are averaged over the datasets, and the average ranks are used to compute the the F statistic. If the statistic is above the critical value for the given number of datasets and algorithms, the null hypothesis can be rejected, i.e. there are statistically significant differences between the algorithms. In this case, the individual differences between the algorithms can be tested using a post-hoc test, such as the Nemenyi test, which requires the ranks of two classifiers to differ by at least the critical difference to be significant [58].

For each of our base classifiers, we compare the base classifier, forward feature selection, RSM, PRSM, Bagging and two versions of AdaBoost over the whole range of datasets described in 5.5.

5.3.2 Results

The performances of all the classifiers are shown in Tables 5.6, 5.7 and 5.8. Not all classifiers could produce a result for every dataset. In such cases, we used only the available values to obtain a performance estimate. However, in some cases (Thyroid dataset for the k -Means and 1-NN base classifiers) too many values were missing. This dataset is therefore omitted from two out of the three comparisons.

Table 5.6: Comparison of methods for the Gaussian base classifier.

Data	Classifier						
	Default	Forward	RSM	PRSM	Bagging	AB-vote	AB-mean
Cancer	60.8 (14.1)	68.9 (11.9)	69.4 (11.4)	65.1 (11.5)	68.7 (12.0)	64.7 (11.9)	68.0 (12.0)
Concordia	93.8 (2.6)	91.3 (2.8)	92.3 (2.5)	93.2 (2.3)	91.1 (2.8)	80.4 (3.4)	92.1 (2.7)
Diabetes	70.3 (7.9)	69.7 (8.4)	70.9 (7.8)	71.7 (8.5)	69.8 (8.4)	57.0 (3.8)	69.9 (8.3)
Glass	78.3 (9.3)	76.4 (7.1)	78.4 (7.3)	78.1 (8.4)	76.5 (7.1)	76.2 (8.2)	75.9 (7.1)
Heart	76.0 (8.1)	63.2 (9.4)	69.2 (9.7)	66.0 (9.8)	63.4 (9.5)	58.8 (7.2)	62.7 (9.4)
Hepatitis	64.4 (16.7)	64.2 (17.3)	75.9 (16.1)	81.1 (14.5)	64.3 (17.2)	53.0 (8.7)	61.1 (17.5)
Housing	80.2 (9.0)	89.0 (6.1)	87.0 (8.3)	88.3 (7.5)	88.8 (6.1)	87.3 (7.3)	86.9 (5.6)
Imports	72.5 (13.0)	83.8 (10.9)	82.9 (10.4)	82.4 (12.6)	83.6 (11.1)	86.4 (9.2)	84.2 (10.4)
Ionosphere	96.4 (3.1)	94.8 (3.8)	93.0 (4.8)	90.1 (5.8)	94.8 (3.9)	92.9 (4.4)	94.1 (4.2)
PV	93.9 (1.6)	93.2 (1.9)	92.7 (2.1)	93.5 (1.9)	93.2 (1.9)	86.6 (2.3)	93.8 (1.8)
Spambase	63.3 (2.1)	69.9 (2.2)	78.4 (1.9)	82.3 (1.6)	69.9 (2.2)	50.0 (0.0)	69.9 (2.2)
Spectf	92.7 (6.5)	87.8 (4.3)	88.3 (4.8)	87.2 (5.6)	88.0 (4.4)	87.5 (6.1)	89.0 (5.1)
Sonar	70.3 (11.5)	68.0 (11.6)	66.6 (11.5)	69.4 (10.7)	68.0 (11.7)	69.6 (9.8)	69.2 (11.7)
Thyroid	83.7 (8.4)	89.3 (6.6)	94.6 (4.2)	96.5 (3.7)	88.9 (6.8)	91.2 (4.8)	91.5 (5.6)
Vehicle	71.6 (5.1)	71.7 (6.4)	69.0 (6.0)	73.3 (5.7)	71.8 (6.4)	64.3 (4.9)	71.5 (6.5)
Vowel	99.2 (1.3)	96.9 (2.3)	93.8 (3.9)	94.4 (4.1)	97.0 (2.3)	97.5 (2.8)	97.1 (2.1)
Wave	89.1 (2.5)	90.6 (2.7)	92.4 (2.3)	91.9 (2.6)	90.6 (2.7)	87.0 (3.2)	90.3 (2.6)
Wine	83.1 (11.3)	87.7 (8.2)	86.6 (8.9)	90.5 (7.7)	88.0 (8.2)	86.2 (8.2)	82.8 (10.2)

Unfortunately, the variable feature size did not always have the effect we had hoped for. Although RSM and PRSM still perform well for some datasets that we have previously used, other datasets suffer quite a lot from this new setting. For instance, for the Prime Vision, Ionosphere, Sonar and Concordia datasets we have seen that the performance of PRSM with a good fixed feature size is at least that of the base classifier, while here this is not the case for the Gaussian and k -Means base classifiers. Interestingly, these are precisely the datasets for which it was better to use 50% of features per subspace. Our intuition is that here RSM and PRSM suffer from smaller subspaces being present as well.

An interesting observation is that for some datasets, Bagging and AdaBoost (both with voting and mean combining methods) are also very competitive classifiers. So, just as RSM and PRSM, these ensembles are data-dependent. Furthermore, the success of these method does not just depend on a particular dataset. In fact, if we examine the results for each cross-validation fold, we will see that the choice of training and test set also influences which method is the “best”. For instance, at some folds, the ensemble classifiers have (almost) the same performance as the base classifier, whereas at other folds, ensembles perform much better.

The Friedman test and post-hoc Nemenyi tests were performed on each of the three comparisons.

Table 5.7: Comparison of methods for the k -Means base classifier.

Data	Default	Forward	RSM	Classifier			
				PRSM	Bagging	AB-vote	AB-mean
Cancer	50.4 (15.8)	69.4 (12.5)	64.2 (12.4)	66.3 (16.0)	71.8 (11.4)	68.9 (11.1)	71.7 (11.7)
Concordia	91.7 (2.7)	91.9 (2.3)	92.0 (2.4)	93.3 (2.0)	92.1 (2.3)	85.6 (2.1)	91.9 (2.3)
Diabetes	65.7 (6.1)	64.7 (6.0)	66.4 (6.1)	67.3 (5.7)	66.5 (6.1)	62.6 (4.4)	66.5 (5.8)
Glass	79.6 (9.7)	81.9 (8.8)	80.8 (8.8)	81.4 (9.2)	80.9 (8.8)	79.1 (9.0)	79.7 (9.0)
Heart	59.0 (11.2)	60.0 (11.7)	68.9 (9.0)	68.1 (9.8)	62.2 (10.1)	61.6 (9.3)	61.8 (10.1)
Hepatitis	53.9 (14.1)	52.4 (20.2)	42.7 (24.8)	72.9 (22.2)	56.7 (19.0)	51.8 (14.1)	57.5 (17.6)
Housing	66.3 (14.1)	88.0 (8.0)	92.3 (4.5)	91.5 (4.8)	89.5 (5.9)	90.0 (5.5)	90.0 (5.5)
Imports	66.7 (13.8)	85.6 (10.0)	88.7 (8.8)	86.5 (9.4)	88.6 (8.7)	86.8 (10.4)	86.6 (9.9)
Ionosphere	96.6 (2.7)	93.9 (4.1)	90.6 (5.2)	89.9 (6.1)	94.3 (4.2)	93.0 (4.3)	92.7 (4.4)
PV	92.4 (2.1)	92.3 (2.1)	92.9 (2.0)	93.7 (1.9)	92.9 (2.0)	86.7 (3.0)	92.7 (2.0)
Spambase	48.6 (2.5)	53.7 (2.6)	65.3 (2.7)	72.7 (2.9)	50.8 (2.6)	50.0 (0.0)	51.0 (2.5)
Spectf	86.0 (6.0)	87.6 (4.6)	89.3 (4.8)	88.5 (5.2)	88.7 (4.5)	86.9 (5.3)	90.2 (4.5)
Sonar	71.4 (10.7)	72.4 (8.8)	75.6 (8.2)	75.1 (8.5)	74.1 (8.6)	72.1 (9.3)	74.8 (8.5)
Vehicle	57.4 (7.8)	64.2 (5.6)	68.6 (4.6)	69.1 (5.5)	68.4 (5.1)	52.5 (4.8)	68.1 (5.2)
Vowel	96.4 (2.7)	95.7 (3.7)	99.1 (1.0)	98.3 (1.9)	98.5 (1.3)	98.9 (1.3)	98.7 (1.2)
Wave	89.1 (2.5)	90.4 (2.9)	91.4 (2.5)	91.2 (2.6)	90.6 (2.6)	87.9 (3.2)	90.6 (2.7)
Wine	75.1 (12.1)	89.1 (8.1)	88.1 (9.1)	90.9 (7.5)	92.8 (6.7)	91.7 (7.0)	86.6 (10.0)

The Friedman test determines whether there is any statistically significant differences in the group of classifiers, whereas the Nemenyi test is used to compare two classifiers in case the null hypothesis (the hypothesis that there are no differences) is rejected.

The results of the Friedman tests are shown in Table 5.9. This table contains the average ranks of the classifiers, whether to reject the null hypothesis (i.e. there are significant differences between the classifiers) and the critical distance for the Nemenyi test.

For the Gaussian base classifier the F statistic shows that there are significant differences between the classifiers. According to the ranks, PRSM is the “best” classifier, closely followed by RSM and the base classifier itself. However, the ranks of the classifiers are still quite similar. This is also reflected by the low value of the F statistic. For pairwise comparisons using the Nemenyi post-hoc test, the critical difference (CD) is 2.12. Using this value, we can only conclude that there is a significant difference between PRSM and AB-vote, which is the worst classifier in this case.

For the k -Means base classifier the differences between classifiers are more significant, which is reflected by a higher value for the F statistic. Again, PRSM is ranked as the best classifier, however, the ranks of RSM, Bagging and AB-mean are also very similar. The base classifier is actually the worst classifier here. Using the Nemenyi post-hoc test, we can conclude that PRSM is significantly better than the base classifier and AB-vote.

For the 1-NN base classifier, there are also significant differences between the classifiers. PRSM is ranked first, followed by Forward feature selection and RSM. AB-vote is by far the worst classifier in the comparison. The post-hoc test shows that PRSM is significantly better than AB-mean and AB-vote.

These results indicate that PRSM is a very competitive classifier. Although PRSM on average

Table 5.8: Comparison of methods for the 1-NN base classifier.

Data	Default	Forward	RSM	Classifier			
				PRSM	Bagging	AB-vote	AB-mean
Cancer	68.7 (13.3)	68.9 (12.6)	69.0 (12.8)	66.3 (12.1)	70.0 (12.6)	49.7 (14.1)	51.4 (14.1)
Concordia	93.9 (1.6)	93.4 (1.9)	94.0 (1.7)	95.2 (1.4)	94.0 (1.6)	90.7 (2.7)	94.4 (1.4)
Diabetes	68.2 (5.9)	68.3 (5.0)	70.6 (6.2)	71.7 (6.8)	68.6 (6.3)	66.5 (6.2)	67.3 (6.2)
Glass	86.5 (8.5)	84.5 (7.7)	85.9 (8.2)	85.3 (8.5)	86.1 (8.3)	80.7 (10.9)	87.0 (7.8)
Heart	59.6 (7.2)	59.6 (9.1)	52.9 (24.9)	67.3 (10.3)	60.7 (8.1)	61.7 (9.4)	62.6 (9.0)
Hepatitis	65.6 (15.6)	67.2 (18.3)	50.0 (0.0)	72.6 (16.2)	66.3 (14.5)	56.9 (15.8)	56.7 (16.9)
Housing	83.4 (10.9)	94.2 (3.6)	90.1 (8.6)	91.8 (7.5)	83.1 (11.3)	81.9 (10.3)	82.0 (10.1)
Imports	87.4 (11.2)	94.0 (6.4)	86.4 (11.4)	87.8 (10.9)	84.0 (11.8)	78.0 (14.5)	81.3 (12.2)
Ionosphere	95.9 (2.8)	87.8 (8.0)	96.2 (2.5)	96.7 (2.5)	96.2 (2.6)	90.8 (4.3)	94.7 (3.9)
PV	95.5 (1.5)	95.4 (1.5)	95.3 (1.5)	95.8 (1.4)	95.7 (1.4)	92.6 (1.9)	96.5 (0.9)
Spambase	75.8 (1.9)	75.9 (2.6)	76.5 (1.9)	80.1 (1.6)	73.2 (1.8)	50.0 (2.3)	60.7 (2.6)
Spectf	95.2 (4.9)	96.4 (3.4)	95.4 (4.9)	95.3 (4.9)	86.1 (17.1)	88.1 (6.0)	96.1 (3.4)
Sonar	84.7 (9.2)	84.8 (7.2)	82.6 (9.6)	83.6 (9.8)	83.4 (9.5)	80.9 (9.6)	82.4 (9.3)
Vehicle	64.2 (6.5)	72.4 (5.1)	72.1 (4.4)	76.4 (4.8)	64.6 (6.4)	64.7 (6.2)	65.0 (6.0)
Vowel	99.6 (0.6)	99.8 (0.3)	99.6 (0.6)	99.6 (0.8)	99.4 (0.8)	99.6 (0.0)	98.8 (0.0)
Wave	90.6 (2.9)	90.9 (2.3)	91.7 (2.3)	91.2 (2.5)	90.9 (2.8)	86.2 (3.4)	88.9 (2.6)
Wine	82.9 (11.1)	88.2 (8.4)	90.4 (8.7)	89.1 (8.3)	84.5 (10.2)	80.9 (9.6)	82.6 (10.2)

Table 5.9: Results of the Friedman test and Nemenyi post-hoc test

Base	Def	Fwd	RSM	Ranks				Analysis	
				PRSM	Bag	AB-v	AB-m	Significant difference?	CD
Gauss	3.61	3.89	3.56	3.11	4.06	5.44	4.33	Yes, F=2.31, CV=2.19	2.12
k-Means	5.94	4.76	2.94	2.65	2.88	5.29	3.53	Yes, F=9.53, CV=2.19	2.19
1-NN	4.24	3.24	3.59	2.12	3.94	6.29	4.59	Yes, F=8.84, CV=2.19	2.19

has the best performances for the examined datasets and base classifiers, it would not be correct to say it is the best classifier in the comparison, as in some cases, it is outperformed by other classifiers that have a worse average rank.

It is clear that these results are strongly influenced by the available datasets. For instance, if the datasets where a classifier performs worse than others were excluded, the classifier’s rank would increase (i.e. be closer to 1). However, excluding datasets would also increase the CD value for the Nemenyi test, making it more difficult to find significant differences when less data is available. Similarly, if a lot of datasets are used, the CD value would be much smaller and more significant differences would be found.

Consider the following experiment: we duplicate the results for the 1-NN base classifier to obtain results for 34 datasets and 7 classifiers. We do not do any further “dataset selection” which might favor any particular classifier. Clearly, the average ranks of the classifiers stay the same as in Table 5.8. However, the CD value for the post-hoc test becomes 1.54. With this value, we could conclude that PRSM is also significantly better than Bagging and the base classifier.

The number of classifiers in comparison also influences whether significant differences are found

(more classifiers = higher CD). For instance, if out of the 1-NN comparison we would extract the results of PRSM and any other classifier, we would always conclude that there is a significant difference, as for two classifiers, $CD=0.47$.

So, the only thing that is possible to conclude in any such comparison is that there are significant differences between classifiers, *given a collection of datasets*. However, unless there is some evidence to believe that this collection is representative of all data, it would be wrong to conclude that one classifier is really “better” than the other.

Chapter 6

Conclusions

This chapter begins with discussion of the observations made during this project in section 6.1. Some ideas for future research are discussed in section 6.2. The main conclusions which answer the research questions, posed in the beginning of this project, are presented in section 6.3.

6.1 Discussion

6.1.1 Prime Vision data

We have seen that after processing the data, both one-class classifiers and two-class classifiers were able to achieve high performances on Prime Vision data. This suggests that the available features allow the target class to be modeled reasonably well by one-class classifiers. Furthermore, we have shown that one-class classifiers are more robust when dealing with previously unseen outliers.

However, we have not been able to achieve a large increase in performance (although the increase was significant using the t-test) using the Random Subspace Method. We have also seen that other types of ensembles also are not able to do so. This is surprising, because in chapter 3 we have seen that most misclassified objects can be classified correctly by a different classifier. Our expectation was that it would be possible to find a classifier that is able to deal with such errors. This means that the remaining errors are caused by the data, not by a particular classifier. Using the current feature representation, some target and outlier objects are too close to each other, causing misclassifications.

Based on the observed results, we recommend PRSM of the Nearest Neighbor classifier as this combination was able to achieve the highest AUC performances on the Prime Vision dataset.

6.1.2 Data-dependency

Although our proposed classifiers did not improve the performance on Prime Vision data significantly, we have seen that for other datasets, the Random Subspace Method was very powerful. We concluded that using subspaces is only beneficial when a single classifier is not able to model the data well using the whole feature space. However, it is difficult to say when this is the case.

We proposed that issues such as sampling, variances and redundancy in the features all have an effect on whether RSM will be able to improve performance of the base classifier. Unfortunately, we were not able to find a clear relationship between these factors. Nevertheless, we can conclude that RSM or PRSM are not only beneficial in problems suffering from a low object/feature ratio, as we have also seen significant improvements for well-sampled datasets.

6.1.3 Parameter influence

As expected, the performance of RSM depended on its parameters: number of features, number of subspaces and combining method. For the last two parameters, it was possible to pick values which were adequate for a range of datasets. In particular, we have seen that the mean combining method is able to produce good performances for most numbers of subspaces. For PRSM, this number was easier to determine because the variations of PRSM based on the number of subspaces were more predictable than with RSM. The number of features was more difficult to pick, as different settings were best for different datasets, and the best setting was sometimes influenced by other parameters. Using a variable number of features was also not a universal solution. Therefore, we were not able to find a good default setting for this parameter. Perhaps the only possibility is to pick a good value empirically, i.e. by evaluating several values beforehand.

Another observation was that in general, it was better to select a few accurate classifiers for an ensemble rather than combine all the possibilities, i.e. PRSM usually performed better than RSM. This means that it is better to combine a few accurate one-class classifiers than many less-accurate one-class classifiers. However, increasing the number of possibilities/classifiers to choose from had only a very limited effect on the performance.

6.1.4 Diversity

We have investigated whether diversity has a relationship to the quality of the RSM ensemble. For the two diversity measures investigated, we were not able to find such a relationship. We suppose that although diversity might be important, the randomness of RSM already provides enough diversity for the ensemble to be effective and that it is not necessary to pay special attention to this property.

6.1.5 Other ensemble techniques

In our comparison of several ensemble classifiers, we have noticed that Bagging performs similarly to RSM or PRSM for some datasets. This means that it may be worthwhile to use ensembles of one-class classifiers in general. AdaBoost also produces good results for some datasets, however, AdaBoost also uses information about the outlier class.

These findings lead us to a more general observation: there is no such thing as the best classifier. The properties of some classifiers are more suitable for some problems than others, so it is best to search for a good classifier for a particular problem rather than try to build an all-purpose classifier.

6.1.6 Evaluation

A general observation from these experiments is that it is difficult to generalize findings about classifiers, gathered from a range of datasets, and that it is not straightforward to compare classifiers on the basis of different datasets. The choice of evaluation measure (such as AUC) and statistical test are likely to influence conclusions about what is significant and what is not.

6.2 Future Research

6.2.1 Prime Vision data

In order to improve the classification of Prime Vision data, more attention needs to be spent on the data, not on a better classifier. It is necessary to provide a description of the data in which the target and outlier classes are more clearly separated.

One possibility is to add new features to the current feature set. Based on the content of the images that are often misclassified, we suggest that features describing the “variability” of components in an image are appropriate.

Another option is to investigate a different data representation altogether. For instance, Multiple Instance Learning is a good parallel to Prime Vision data: components are instances and clusters are bags of instances. Other choices for data representation include dissimilarities or pixels, however we doubt that the latter is a feasible option due to the amount of information contained in each image.

6.2.2 Data dependency

Clearly, a more thorough understanding of why ensembles work on some datasets, but not on others, is needed. This is a very challenging task because of the number of experiments that can be designed to compare the benefits of RSM for particular types of datasets, as we did in our investigation. However, for any experiment, we would recommend using a larger sample of datasets (perhaps a mix of real and artificial data).

In particular, datasets that still should be investigated are datasets with very high dimensionality. Perhaps RSM is more effective in such cases. However, in this case we might have to rethink our conclusions about RSM parameters, as for very high-dimensional datasets, it would probably be infeasible to use subsets with 25% or 50% of the features because such subspaces would still be very high-dimensional. Changing the number of features to a lower value might also affect the choices for the other parameters.

6.2.3 Parameter influence

The size of the subspaces turned out to be an important, and very data-dependent parameter for RSM. In order for RSM to be a more effective classifier in general, we need to gain more insight into how to choose a good value for this parameter. This could be done either by a better theoretical

understanding of using subspaces and picking a value, based on some properties of the dataset, or by empirically evaluating a few values and then selecting the best value.

Another parameter that could use more investigation is the combining method. Firstly, we only investigated a few simple combining methods, however, methods using weighting or meta-combining methods which were explained in section 2.2 might turn out to be more effective. Secondly, more understanding regarding the performances of different combining rules relative to the number of subspaces (for instance, why does performance using the product rule decrease?) is appropriate.

For PRSM, we only investigated selecting a fixed number of subspaces L_s . A different possibility for choosing L_s would be to increase L_s until no improvement is found, or to determine L_s based on a threshold on AUC.

6.2.4 Other ensemble techniques

We have seen that other ensembles such as Bagging and AdaBoost are also able to improve performances of one-class classifiers, sometimes in cases where RSM and PRSM are not able to do so. It would be interesting to investigate what causes these differences. Understanding more about the “areas of expertise” of these ensemble techniques might enable us to build even stronger ensembles by for instance combining advantages of Bagging and the Random Subspace Method.

6.3 Main Conclusions

The main research questions for this investigation were:

- *Are one-class classifiers suitable for the Prime Vision problem?*
- *How does the Random Subspace Method affect the performance of one-class classifiers?*

The Prime Vision classification problem is suitable for one-class classifiers because even simple, non-optimized one-class classifiers are able to obtain good performances on the dataset. Furthermore, a welcome property of one-class classifiers is that their performance will not degrade due to previously unseen outliers.

In general, RSM is able to improve the performance of one-class classifiers. However, this improvement depends on the dataset and the parameters used for the RSM classifier. Although we have a few intuitions about what datasets are more suitable for RSM, a more thorough investigation is necessary. The parameters for RSM may be difficult to choose. Pruning inaccurate classifiers from the RSM ensemble simplifies setting the parameters, and is also able to provide superior performances to RSM.

Next to these answers, there are two other conclusions that we would like to stress. Firstly, it is important to remember that there is such thing as the best classifier because some classifiers may be more suitable for some problems than others. Secondly, caution must be exercised when drawing conclusions about these differences between classifiers, because the evaluation method is very likely to influence these results.

Bibliography

- [1] Prime Vision, “Case study: TNT Post Parcel Service Deploys Prime Vision OCR & Video Coding Solutions Nationwide,” September 2010.
- [2] M. Rijcke, M. Bojovic, W. Homan, and M. Nuijt, “Issues in developing a commercial parcel reading system,” in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pp. 1015–1019, IEEE, 2005.
- [3] T. Dietterich, “Ensemble methods in machine learning,” *Multiple Classifier Systems*, pp. 1–15, 2000.
- [4] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic Press Professional, 1990.
- [5] D. Tax, *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, June 2001.
- [6] D. Tax and R. Duin, “Combining one-class classifiers,” *Multiple Classifier Systems*, pp. 299–308, 2001.
- [7] A. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.
- [8] T. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.
- [9] D. Opitz and R. Maclin, “Popular ensemble methods: An empirical study,” *Journal of Artificial Intelligence Research*, vol. 11, no. 169–198, p. 12, 1999.
- [10] J. Kittler, “Combining classifiers: A theoretical framework,” *Pattern Analysis & Applications*, vol. 1, no. 1, pp. 18–27, 1998.
- [11] N. de Condorcet, *Essai sur l’application de l’analyse la probabilit des dcisions rendues la pluralit des voix*. Imprimerie Royale, Paris, 1785.
- [12] L. Hansen and P. Salamon, “Neural network ensembles,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.
- [13] A. Krogh and J. Vedelsby, “Neural network ensembles, cross validation, and active learning,” *Advances in Neural Information Processing Systems*, pp. 231–238, 1995.
- [14] L. Kuncheva and C. Whitaker, “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy,” *Machine Learning*, vol. 51, no. 2, pp. 181–207, 2003.

- [15] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection," *Machine Learning: ECML 2000*, pp. 109–116, 2000.
- [16] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, no. 1, pp. 83–98, 2005.
- [17] G. Valentini and F. Masulli, "Ensembles of learning machines," *Neural Nets*, pp. 3–20, 2002.
- [18] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [19] T. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
- [20] R. Bryll, R. Gutierrez-Osuna, and F. Quek, "Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets," *Pattern Recognition*, vol. 36, no. 6, pp. 1291–1302, 2003.
- [21] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp. 157–166, ACM, 2005.
- [22] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, 2010.
- [23] R. Schapire and Y. Freund, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, p. 148, Morgan Kaufmann Pub, 1996.
- [24] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [25] H. Altınçay, "Ensembling evidential k-nearest neighbor classifiers through multi-modal perturbation," *Applied Soft Computing*, vol. 7, no. 3, pp. 1072–1083, 2007.
- [26] P. Panov and S. Džeroski, "Combining bagging and random subspaces to create better ensembles," in *Proceedings of the Seventh International Conference on Intelligent Data Analysis*, pp. 118–129, Springer-Verlag, 2007.
- [27] L. Xu, A. Krzyzak, and C. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems Man and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.
- [28] F. Roli and G. Giacinto, "Design of multiple classifier systems," *Series in Machine Perception and Artificial Intelligence*, vol. 47, pp. 199–226, 2002.
- [29] N. Oza and K. Tumer, "Classifier ensembles: Select real-world applications," *Information Fusion*, vol. 9, no. 1, pp. 4–20, 2008.
- [30] F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems," *Multiple Classifier Systems*, pp. 78–87, 2001.
- [31] Z. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

- [32] L. Rokach, "Collective-agreement-based pruning of ensembles," *Computational Statistics & Data Analysis*, vol. 53, no. 4, pp. 1015–1026, 2009.
- [33] D. Opitz, "Feature selection for ensembles," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 379–384, John Wiley & Sons Ltd, 1999.
- [34] A. Tsymbal, S. Puuronen, and D. Patterson, "Ensemble feature selection with the simple Bayesian classification," *Information Fusion*, vol. 4, no. 2, pp. 87–100, 2003.
- [35] M. Skurichina and R. Duin, "Bagging, Boosting and the Random Subspace Method for linear classifiers," *Pattern Analysis & Applications*, vol. 5, no. 2, pp. 121–135, 2002.
- [36] H. Nguyen, H. Ang, and V. Gopalkrishnan, "Mining Outliers with Ensemble of Heterogeneous Detectors on Random Subspaces," in *Database Systems for Advanced Applications*, pp. 368–383, Springer, 2010.
- [37] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Sequential genetic search for ensemble feature selection," *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2005.
- [38] S. Sun, "An improved Random Subspace Method and its application to EEG signal classification," *Multiple Classifier Systems*, pp. 103–112, 2007.
- [39] H. Ahn, H. Moon, M. Fazzari, N. Lim, J. Chen, and R. Kodell, "Classification by ensembles from random partitions of high-dimensional data," *Computational Statistics & Data Analysis*, vol. 51, no. 12, pp. 6166–6179, 2007.
- [40] R. Díaz-Uriarte and A. de Andrés, "Gene selection and classification of microarray data using random forest," *BMC Bioinformatics*, vol. 7, no. 1, p. 3, 2006.
- [41] L. Kuncheva, J. Rodríguez, C. Plumpton, D. Linden, and S. Johnston, "Random subspace ensembles for FMRI classification," *IEEE Transactions on Medical Imaging*, vol. 29, no. 2, pp. 531–542, 2010.
- [42] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems," in *Proceedings of the Sixth International Conference on Data Mining (ICDM'06)*, pp. 488–498, 2006.
- [43] J. Munoz-Mari, G. Camps-Valls, L. Gómez-Chova, and J. Calpe-Maravilla, "Combination of One-Class Remote Sensing Image Classifiers," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 1509–1512, Citeseer, 2007.
- [44] I. Corona, G. Giacinto, and F. Roli, "Intrusion detection in computer systems using multiple classifier systems," *Supervised and Unsupervised Ensemble Methods and their Applications*, pp. 91–113, 2008.
- [45] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli, "Intrusion detection in computer networks by a modular ensemble of one-class classifiers," *Information Fusion*, vol. 9, no. 1, pp. 69–82, 2008.
- [46] L. Nanni, "Experimental comparison of one-class classifiers for online signature verification," *Neurocomputing*, vol. 69, no. 7-9, pp. 869–873, 2006.
- [47] A. Lumini and L. Nanni, "Ensemble of on-line signature matchers based on OverComplete feature generation," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5291–5296, 2009.

- [48] B. Biggio, G. Fumera, and F. Roli, “Multiple classifier systems under attack,” *Multiple Classifier Systems*, pp. 74–83, 2010.
- [49] E. Pekalska and R. Duin, *The dissimilarity representation for pattern recognition: foundations and applications*. World Scientific Publishing Co Inc, 2005.
- [50] O. Maron and T. Lozano-Pérez, “A framework for multiple-instance learning,” in *Advances in Neural Information Processing Systems*, pp. 570–576, Citeseer, 1998.
- [51] D. Tax, “OC classifier results,” 2010.
- [52] Asuncion, A. and Newman, D.J., “UCI Machine Learning Repository,” 2007.
- [53] D. Tax, “DDtools, the Data Description toolbox for Matlab,” 2010. version 1.7.4.
- [54] R. Duin, “PRtools,” 2009. version 4.1.5.
- [55] S. Salzberg, “On comparing classifiers: Pitfalls to avoid and a recommended approach,” *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 317–328, 1997.
- [56] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [57] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.
- [58] P. Nemenyi, *Distribution-free multiple comparisons*. PhD thesis, Princeton, 1963.

Appendix A

Results on Additional Data

This appendix contains the results of the RSM and PRSM experiments on other datasets than Prime Vision. For the RSM experiments, we provide both a table of results and corresponding graphs. For PRSM experiments, we provide a graph of RSM vs. PRSM comparison and a table for the pool size experiment.

In tables, the average AUC performances and their standard deviations, both multiplied by 100, are given.

In figures, the legends and axes names are not displayed due to the size of the figures. However, the figures should be interpreted in the same way as the figures for Prime Vision data. Examples including the legend and axes names are shown in Fig. A.1.

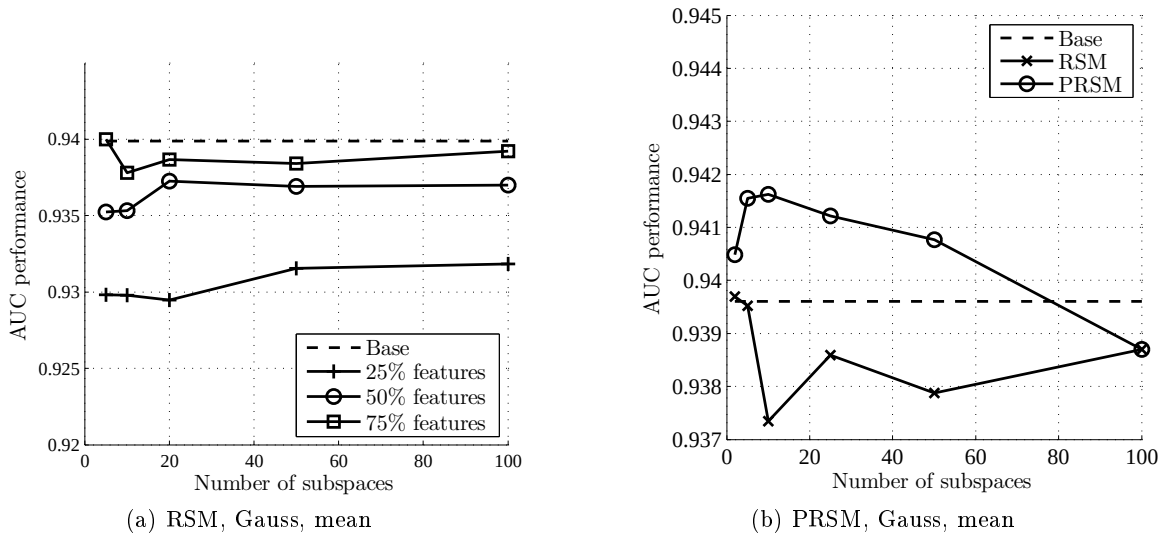
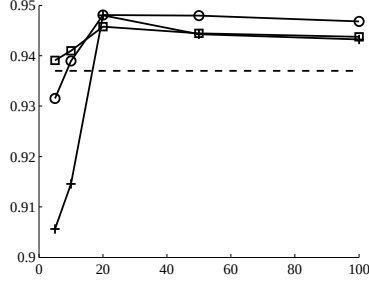
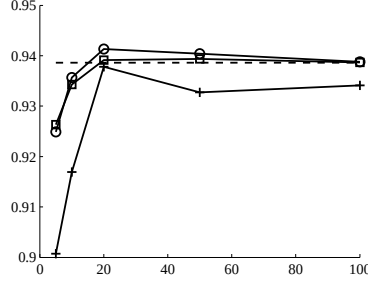


Figure A.1: Experiment results, Prime Vision

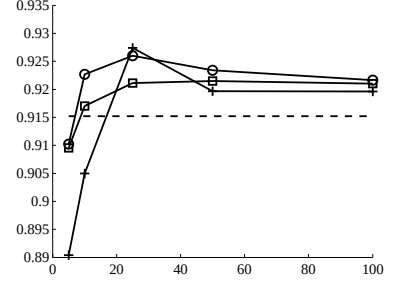
A.1 RSM



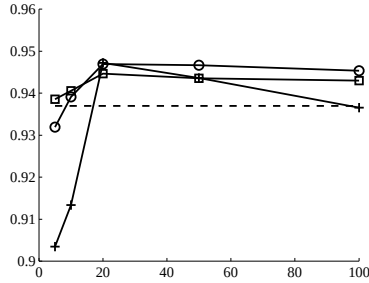
(a) Gauss, mean



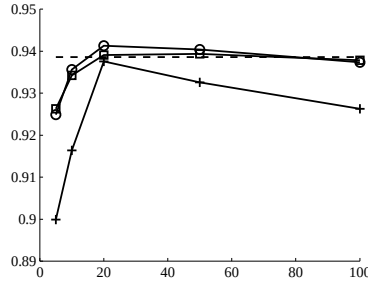
(b) 1-NN, mean



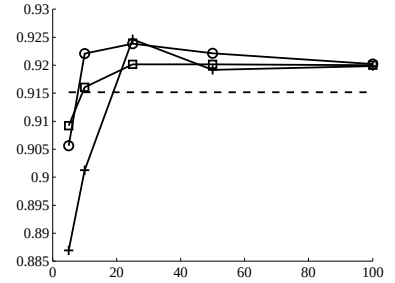
(c) k -Means, mean



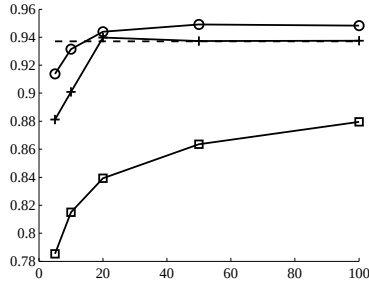
(d) Gauss, prod



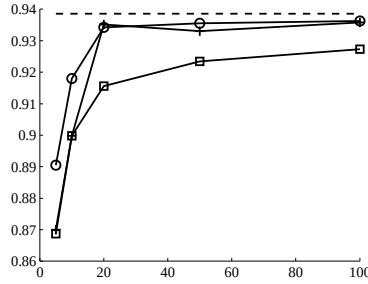
(e) 1-NN, prod



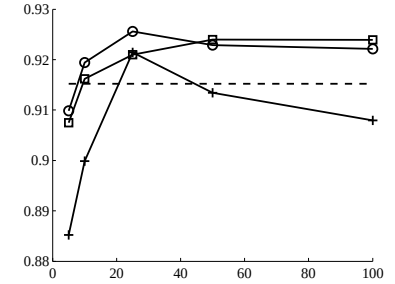
(f) k -Means, prod



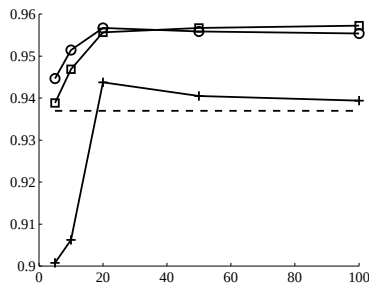
(g) Gauss, vote



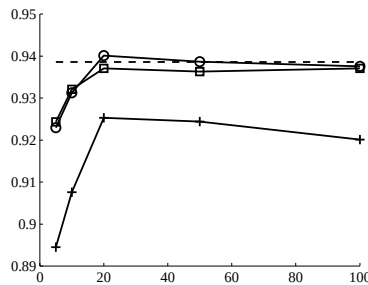
(h) 1-NN, vote



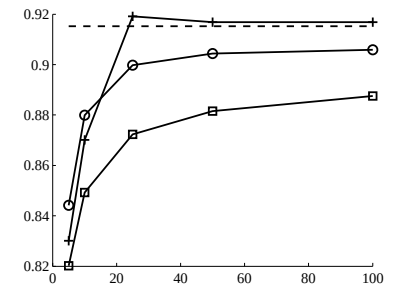
(i) k -Means, vote



(j) Gauss, max



(k) 1-NN, max



(l) k -Means, max

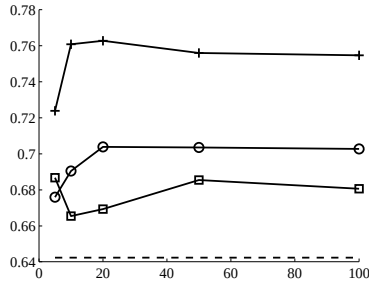
Figure A.2: Parameters RSM, Concordia dataset

Table A.1: AUC of RSM, Concordia dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

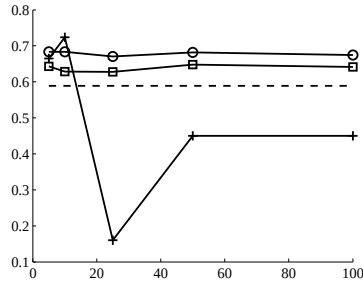
		Subspaces					
Classifier		2	5	10	20	50	100
Gauss, 93.7 (2.2)	25% mean	<i>89.0 (2.5)</i>	<i>90.6 (2.6)</i>	<i>91.5 (2.4)</i>	94.8 (1.6)	94.4 (1.7)	94.3 (1.8)
	25% prod	<i>88.9 (2.5)</i>	<i>90.3 (2.7)</i>	<i>91.3 (2.5)</i>	94.7 (1.7)	94.4 (1.7)	93.7 (1.8)
	25% vote	<i>83.8 (2.6)</i>	<i>88.1 (2.6)</i>	<i>90.1 (2.5)</i>	94.0 (1.9)	93.7 (2.0)	93.8 (2.0)
	25% max	<i>89.2 (2.4)</i>	<i>90.1 (2.6)</i>	<i>90.6 (2.9)</i>	94.4 (2.1)	94.1 (2.1)	93.9 (2.2)
	50% mean	<i>91.8 (2.8)</i>	<i>93.2 (2.7)</i>	93.9 (2.4)	94.8 (2.0)	94.8 (2.0)	94.7 (2.0)
	50% prod	<i>91.7 (2.8)</i>	<i>93.2 (2.8)</i>	93.9 (2.5)	94.7 (2.2)	94.7 (2.1)	94.5 (2.1)
	50% vote	<i>87.8 (3.7)</i>	<i>91.4 (2.7)</i>	<i>93.2 (2.2)</i>	94.4 (1.7)	94.9 (1.7)	94.8 (1.8)
	50% max	<i>92.0 (2.8)</i>	94.5 (1.9)	95.1 (1.7)	95.7 (1.5)	95.6 (1.4)	95.5 (1.5)
	75% mean	93.9 (2.3)	93.9 (2.2)	94.1 (2.2)	94.6 (2.0)	94.4 (2.0)	94.4 (2.1)
	75% prod	93.9 (2.3)	93.9 (2.2)	94.1 (2.3)	94.5 (2.1)	94.4 (2.1)	94.3 (2.1)
	75% vote	<i>74.6 (3.7)</i>	<i>78.5 (4.2)</i>	<i>81.5 (3.7)</i>	<i>83.9 (4.1)</i>	<i>86.4 (3.6)</i>	<i>88.0 (3.8)</i>
	75% max	93.7 (2.3)	93.9 (1.9)	94.7 (1.9)	95.6 (1.5)	95.7 (1.4)	95.7 (1.4)
1-NN, 93.9 (2.4)	25% mean	<i>87.5 (3.3)</i>	<i>90.1 (3.3)</i>	<i>91.7 (3.2)</i>	93.8 (2.7)	<i>93.3 (2.8)</i>	<i>93.4 (2.7)</i>
	25% prod	<i>87.5 (3.3)</i>	<i>90.0 (3.3)</i>	<i>91.6 (3.2)</i>	93.8 (2.7)	<i>93.3 (2.8)</i>	<i>92.6 (2.6)</i>
	25% vote	<i>80.8 (3.3)</i>	<i>87.0 (3.0)</i>	<i>90.0 (3.2)</i>	<i>93.5 (2.7)</i>	<i>93.3 (2.8)</i>	<i>93.6 (2.7)</i>
	25% max	<i>87.7 (3.3)</i>	<i>89.5 (3.3)</i>	<i>90.8 (3.4)</i>	<i>92.5 (3.1)</i>	<i>92.4 (3.0)</i>	<i>92.0 (3.1)</i>
	50% mean	<i>92.3 (2.9)</i>	<i>92.5 (2.9)</i>	<i>93.6 (2.8)</i>	94.1 (2.6)	94.0 (2.6)	93.9 (2.6)
	50% prod	<i>92.3 (2.9)</i>	<i>92.5 (2.9)</i>	<i>93.6 (2.8)</i>	94.1 (2.6)	94.0 (2.6)	93.7 (2.6)
	50% vote	<i>86.2 (2.8)</i>	<i>89.1 (2.8)</i>	<i>91.8 (2.8)</i>	<i>93.4 (2.6)</i>	<i>93.6 (2.6)</i>	<i>93.6 (2.6)</i>
	50% max	<i>92.3 (2.9)</i>	<i>92.3 (3.0)</i>	<i>93.1 (2.9)</i>	94.0 (2.7)	93.9 (2.6)	93.8 (2.7)
	75% mean	<i>92.9 (2.8)</i>	<i>92.6 (2.8)</i>	<i>93.4 (2.7)</i>	93.9 (2.5)	93.9 (2.5)	93.9 (2.5)
	75% prod	<i>92.9 (2.8)</i>	<i>92.6 (2.8)</i>	<i>93.4 (2.7)</i>	93.9 (2.5)	93.9 (2.5)	<i>93.8 (2.5)</i>
	75% vote	<i>85.3 (2.7)</i>	<i>86.9 (3.0)</i>	<i>90.0 (2.9)</i>	<i>91.6 (2.6)</i>	<i>92.3 (2.6)</i>	<i>92.7 (2.5)</i>
	75% max	<i>92.9 (2.8)</i>	<i>92.4 (2.9)</i>	<i>93.2 (2.8)</i>	<i>93.7 (2.7)</i>	<i>93.6 (2.6)</i>	<i>93.7 (2.6)</i>
<i>k</i> -Means, 91.5 (2.5)	25% mean	<i>86.8 (3.0)</i>	<i>89.0 (2.7)</i>	<i>90.5 (2.4)</i>	92.7 (2.2)	92.0 (2.2)	92.0 (2.3)
	25% max	<i>86.9 (3.1)</i>	<i>88.5 (2.9)</i>	<i>90.0 (2.6)</i>	92.1 (2.4)	91.3 (2.3)	<i>90.8 (2.2)</i>
	25% prod	<i>86.8 (3.0)</i>	<i>89.0 (2.7)</i>	<i>90.5 (2.4)</i>	92.7 (2.2)	92.0 (2.2)	92.0 (2.3)
	25% vote	<i>78.2 (2.5)</i>	<i>83.0 (2.3)</i>	<i>87.0 (2.3)</i>	91.9 (2.3)	91.7 (2.3)	91.7 (2.2)
	50% mean	<i>90.4 (2.6)</i>	<i>91.0 (2.8)</i>	92.3 (2.4)	92.6 (2.3)	92.3 (2.3)	92.2 (2.3)
	50% max	<i>90.4 (2.7)</i>	<i>91.0 (2.8)</i>	91.9 (2.5)	92.6 (2.3)	92.3 (2.3)	92.2 (2.2)
	50% prod	<i>90.4 (2.6)</i>	<i>91.0 (2.8)</i>	92.3 (2.4)	92.6 (2.3)	92.3 (2.3)	92.2 (2.3)
	50% vote	<i>81.1 (2.4)</i>	<i>84.4 (2.4)</i>	<i>88.0 (2.1)</i>	<i>90.0 (1.9)</i>	<i>90.4 (2.1)</i>	<i>90.6 (2.2)</i>
	75% mean	<i>90.8 (2.5)</i>	<i>91.0 (2.4)</i>	91.7 (2.4)	92.1 (2.3)	92.1 (2.3)	92.1 (2.3)
	75% max	<i>90.8 (2.5)</i>	<i>90.7 (2.3)</i>	91.6 (2.3)	92.1 (2.2)	92.4 (2.2)	92.4 (2.1)
	75% prod	<i>90.8 (2.5)</i>	<i>91.0 (2.4)</i>	91.7 (2.4)	92.1 (2.3)	92.1 (2.3)	92.1 (2.3)
	75% vote	<i>79.4 (2.1)</i>	<i>82.0 (1.8)</i>	<i>84.9 (2.1)</i>	<i>87.2 (2.1)</i>	<i>88.2 (2.2)</i>	<i>88.8 (2.0)</i>

Table A.2: AUC of RSM, Hepatitis dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

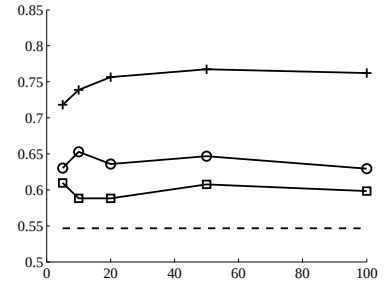
		Subspaces					
Classifier		2	5	10	20	50	100
Gauss, 64.2 (17.0)	25% mean	76.1 (16.2)	72.4 (16.2)	76.1 (15.4)	76.3 (14.9)	75.6 (15.0)	75.5 (14.9)
	25% prod	75.9 (16.0)	70.2 (16.6)	75.4 (16.1)	73.4 (16.6)	<i>55.5 (13.9)</i>	<i>48.9 (3.3)</i>
	25% vote	69.3 (15.9)	68.0 (15.8)	71.3 (14.7)	72.8 (16.0)	74.3 (15.1)	75.0 (15.0)
	25% max	76.6 (16.0)	73.6 (14.9)	76.6 (13.6)	76.9 (13.8)	76.1 (14.2)	75.8 (14.6)
	50% mean	67.5 (18.7)	67.6 (18.4)	69.0 (16.1)	70.4 (16.1)	70.4 (15.5)	70.3 (15.4)
	50% prod	65.4 (19.0)	65.8 (18.2)	68.4 (16.5)	67.5 (17.4)	<i>57.6 (15.6)</i>	<i>51.3 (8.9)</i>
	50% vote	63.3 (15.0)	62.8 (16.3)	62.8 (16.8)	62.7 (16.8)	67.2 (17.2)	69.3 (17.2)
	50% max	68.0 (18.7)	65.0 (17.8)	68.0 (17.0)	67.1 (16.9)	73.4 (16.9)	74.3 (16.2)
	75% mean	71.7 (14.4)	68.7 (16.6)	66.5 (17.0)	66.9 (17.0)	68.5 (16.2)	68.1 (16.2)
	75% prod	71.0 (15.2)	68.6 (17.2)	66.0 (17.4)	65.1 (18.7)	<i>56.5 (15.8)</i>	<i>54.1 (10.9)</i>
	75% vote	<i>58.9 (13.7)</i>	<i>59.5 (14.6)</i>	<i>58.9 (14.5)</i>	<i>58.9 (15.0)</i>	70.0 (14.4)	69.9 (14.3)
	75% max	71.8 (14.3)	72.4 (15.3)	72.1 (15.3)	71.9 (15.1)	78.0 (13.3)	77.8 (13.3)
<i>k</i> -Means, 54.6 (13.9)	25% mean	73.9 (11.6)	71.8 (13.3)	73.9 (13.4)	75.7 (13.1)	76.7 (11.8)	76.2 (12.3)
	25% prod	74.1 (12.5)	72.3 (13.5)	73.4 (13.3)	75.7 (12.8)	60.9 (11.2)	<i>49.9 (5.6)</i>
	25% vote	60.6 (14.0)	60.6 (13.5)	70.3 (12.2)	72.0 (13.5)	74.6 (11.4)	73.9 (11.6)
	25% max	72.2 (13.4)	69.9 (15.0)	73.9 (12.3)	75.1 (12.9)	76.1 (12.5)	74.4 (12.4)
	50% mean	64.4 (18.8)	63.0 (18.0)	65.3 (14.2)	63.6 (14.4)	64.7 (13.3)	62.9 (13.5)
	50% prod	61.3 (18.2)	61.9 (18.8)	64.6 (15.6)	64.1 (15.0)	54.0 (13.0)	<i>49.2 (5.7)</i>
	50% vote	59.7 (12.8)	58.1 (12.4)	59.4 (13.3)	58.8 (14.7)	66.3 (12.2)	64.9 (12.2)
	50% max	66.5 (16.3)	66.2 (14.2)	67.2 (15.1)	65.8 (14.6)	69.4 (13.6)	69.4 (13.8)
	75% mean	59.4 (16.2)	61.0 (15.7)	58.8 (15.5)	58.8 (14.4)	60.8 (14.3)	59.8 (14.4)
	75% prod	60.4 (16.7)	62.2 (15.2)	58.8 (15.2)	58.9 (14.5)	54.0 (15.1)	<i>47.6 (6.8)</i>
	75% vote	<i>49.2 (10.5)</i>	<i>49.5 (11.6)</i>	<i>48.7 (11.6)</i>	<i>50.2 (13.3)</i>	62.9 (13.5)	62.5 (12.8)
	75% max	57.5 (16.3)	61.9 (14.4)	61.1 (15.3)	60.4 (13.1)	68.9 (13.9)	69.0 (13.1)
1-NN, 58.9 (19.3)	25% mean	71.8 (12.1)	66.5 (18.9)	72.4 (17.7)	<i>16.0 (30.0)</i>	<i>45.0 (15.2)</i>	<i>45.0 (15.2)</i>
	25% prod	<i>49.8 (10.1)</i>	<i>50.2 (9.6)</i>	<i>51.0 (9.3)</i>	<i>39.7 (21.5)</i>	<i>50.0 (0.0)</i>	<i>50.0 (0.0)</i>
	25% vote	<i>52.8 (11.4)</i>	57.3 (17.6)	61.7 (17.1)	66.6 (17.0)	71.6 (15.4)	73.3 (15.3)
	25% max	71.7 (12.3)	65.5 (17.3)	64.9 (16.1)	69.9 (16.8)	74.9 (12.6)	73.8 (12.1)
	50% mean	64.9 (18.8)	68.3 (19.6)	68.4 (18.8)	67.0 (18.8)	68.2 (18.7)	67.4 (18.4)
	50% prod	65.0 (18.7)	70.1 (18.7)	68.9 (17.9)	61.1 (16.8)	<i>52.4 (13.8)</i>	<i>51.8 (9.2)</i>
	50% vote	<i>52.6 (12.7)</i>	56.2 (17.0)	60.3 (19.4)	61.5 (19.1)	64.6 (18.3)	65.2 (18.1)
	50% max	65.3 (19.4)	65.7 (20.4)	65.6 (20.0)	63.1 (20.2)	67.9 (18.6)	67.7 (19.0)
	75% mean	62.7 (18.9)	64.3 (18.3)	62.8 (18.4)	62.8 (19.2)	64.8 (18.6)	64.2 (19.0)
	75% prod	63.3 (18.9)	65.3 (18.4)	63.3 (18.8)	58.8 (18.6)	<i>51.6 (13.0)</i>	<i>50.2 (11.1)</i>
	75% vote	60.6 (14.8)	59.4 (14.2)	58.5 (14.3)	59.5 (15.7)	65.8 (17.8)	65.4 (18.0)
	75% max	63.2 (18.6)	64.8 (18.6)	63.7 (18.7)	67.1 (19.4)	70.8 (18.2)	70.6 (18.4)



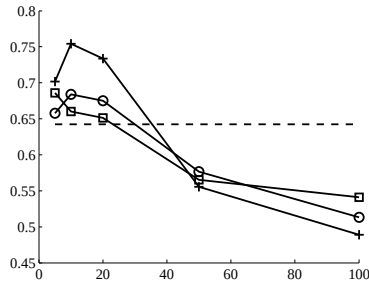
(a) Gauss, mean



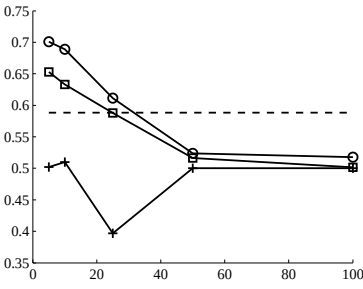
(b) 1-NN, mean



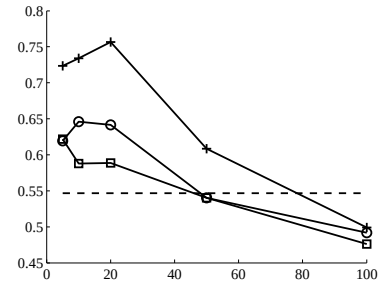
(c) k -Means, mean



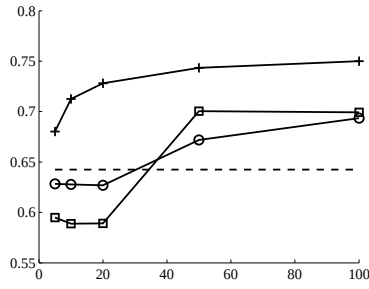
(d) Gauss, prod



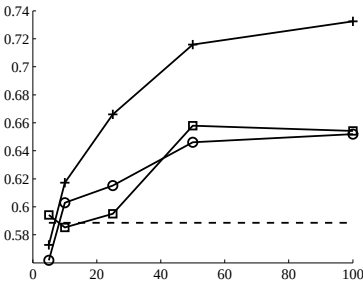
(e) 1-NN, prod



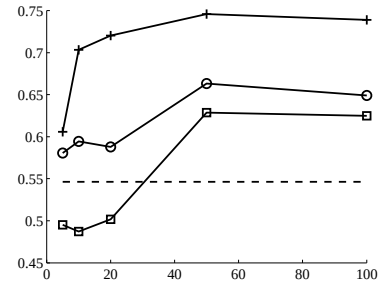
(f) k -Means, prod



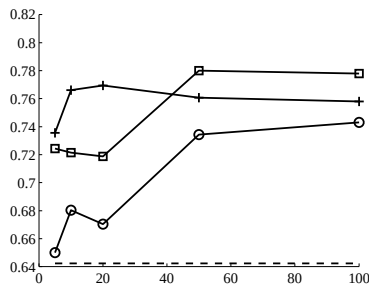
(g) Gauss, vote



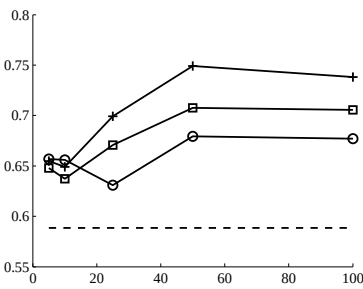
(h) 1-NN, vote



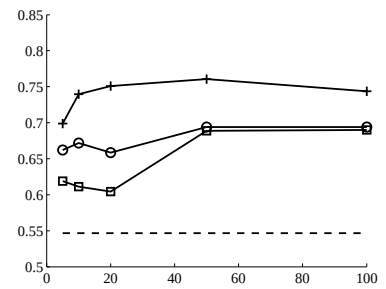
(i) k -Means, vote



(j) Gauss, max



(k) 1-NN, max

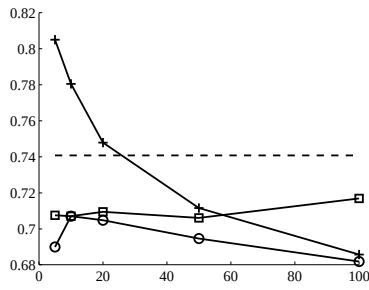


(l) k -Means, max

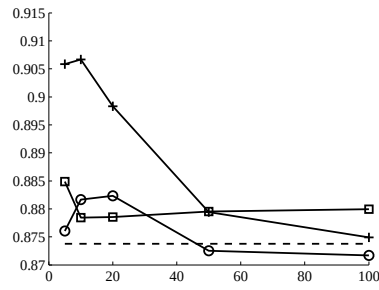
Figure A.3: Parameters RSM, Hepatitis dataset

Table A.3: AUC of RSM, Imports dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

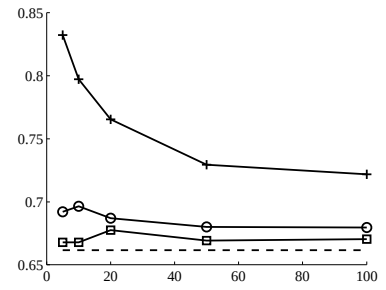
		Subspaces					
Classifier		2	5	10	20	50	100
Gauss, 74.1 (13.1)	25% mean	78.2 (10.6)	80.5 (11.2)	78.1 (11.3)	74.8 (11.8)	<i>71.2 (12.4)</i>	<i>68.6 (12.6)</i>
	25% prod	76.7 (10.4)	80.9 (10.5)	78.3 (10.8)	74.3 (11.4)	<i>68.8 (11.2)</i>	<i>62.1 (9.9)</i>
	25% vote	78.1 (8.8)	80.7 (10.2)	79.2 (11.5)	77.0 (12.0)	75.6 (12.9)	73.9 (13.1)
	25% max	79.5 (10.3)	77.7 (11.0)	74.6 (11.9)	74.6 (12.3)	72.9 (11.5)	<i>71.5 (12.1)</i>
	50% mean	<i>64.0 (12.6)</i>	<i>69.0 (13.4)</i>	<i>70.7 (13.6)</i>	<i>70.5 (13.1)</i>	<i>69.5 (12.9)</i>	<i>68.2 (12.6)</i>
	50% prod	<i>62.8 (12.8)</i>	<i>68.3 (13.4)</i>	<i>70.3 (13.4)</i>	<i>69.9 (13.0)</i>	<i>69.2 (13.4)</i>	<i>69.7 (11.8)</i>
	50% vote	<i>69.0 (10.8)</i>	72.3 (12.1)	73.5 (13.0)	74.5 (12.6)	73.7 (12.7)	73.1 (12.2)
	50% max	<i>65.8 (12.8)</i>	<i>66.3 (11.9)</i>	<i>68.0 (12.2)</i>	<i>69.1 (12.0)</i>	<i>69.3 (12.6)</i>	<i>68.8 (12.6)</i>
	75% mean	73.7 (13.1)	<i>70.8 (13.5)</i>	<i>70.7 (13.7)</i>	<i>70.9 (13.5)</i>	<i>70.6 (13.2)</i>	<i>71.7 (13.0)</i>
	75% prod	73.7 (13.1)	<i>70.5 (13.4)</i>	<i>70.5 (13.7)</i>	<i>70.7 (13.6)</i>	<i>70.3 (13.5)</i>	<i>70.9 (13.2)</i>
	75% vote	<i>71.2 (10.5)</i>	<i>71.2 (11.4)</i>	<i>71.8 (12.6)</i>	72.6 (12.4)	73.2 (11.5)	74.0 (11.4)
	75% max	73.4 (13.1)	<i>70.2 (13.6)</i>	<i>69.3 (13.6)</i>	<i>69.4 (13.3)</i>	<i>70.6 (13.3)</i>	<i>71.2 (13.1)</i>
1-NN, 87.4 (11.2)	25% mean	87.8 (7.9)	90.6 (7.8)	90.7 (7.9)	89.8 (8.0)	87.9 (9.0)	87.5 (8.6)
	25% prod	85.3 (8.3)	91.4 (7.4)	91.8 (7.0)	85.7 (8.1)	<i>65.7 (10.0)</i>	<i>62.0 (7.7)</i>
	25% vote	<i>78.4 (8.1)</i>	86.8 (8.9)	86.2 (9.1)	86.8 (9.2)	<i>85.8 (10.2)</i>	<i>84.8 (10.3)</i>
	25% max	88.4 (8.0)	90.1 (7.6)	87.3 (10.1)	86.6 (10.9)	<i>85.1 (11.3)</i>	<i>84.3 (11.9)</i>
	50% mean	<i>81.7 (9.7)</i>	87.6 (9.7)	88.2 (10.1)	88.2 (10.6)	87.2 (10.1)	87.2 (10.4)
	50% prod	<i>80.6 (9.9)</i>	86.8 (9.8)	88.4 (9.4)	88.4 (10.0)	<i>77.9 (10.4)</i>	<i>70.2 (9.9)</i>
	50% vote	<i>69.9 (9.8)</i>	<i>78.7 (10.2)</i>	<i>79.7 (11.2)</i>	<i>81.1 (12.0)</i>	<i>82.1 (11.1)</i>	<i>82.8 (11.3)</i>
	50% max	<i>81.4 (10.0)</i>	86.4 (10.8)	87.2 (10.7)	<i>85.9 (11.0)</i>	<i>85.3 (11.2)</i>	<i>84.8 (11.8)</i>
	75% mean	88.4 (10.0)	88.5 (10.6)	87.8 (11.3)	87.9 (11.4)	88.0 (10.8)	88.0 (10.7)
	75% prod	88.1 (10.1)	87.6 (10.6)	87.0 (11.2)	87.4 (11.1)	<i>80.3 (10.7)</i>	<i>72.9 (10.6)</i>
	75% vote	<i>72.0 (9.8)</i>	<i>72.4 (10.6)</i>	<i>73.1 (11.0)</i>	<i>76.6 (10.9)</i>	<i>78.2 (10.5)</i>	<i>79.1 (10.2)</i>
	75% max	88.9 (9.7)	87.5 (10.7)	87.4 (11.4)	87.2 (11.3)	87.3 (11.3)	87.8 (10.4)
<i>k</i> -Means, 66.2 (15.5)	25% mean	77.4 (12.7)	83.2 (13.4)	79.7 (14.4)	76.5 (14.7)	73.0 (14.9)	72.2 (14.9)
	25% prod	76.3 (12.8)	82.8 (13.3)	79.1 (14.0)	76.8 (14.1)	73.0 (14.9)	64.8 (11.5)
	25% vote	74.7 (12.2)	79.6 (12.4)	77.6 (14.4)	77.1 (14.0)	74.2 (15.5)	73.7 (15.7)
	25% max	78.8 (12.4)	79.8 (13.5)	74.9 (15.1)	74.1 (14.6)	73.0 (15.4)	73.1 (15.8)
	50% mean	67.0 (14.7)	69.2 (16.1)	69.6 (16.0)	68.7 (16.1)	68.0 (15.9)	68.0 (16.3)
	50% prod	66.8 (14.6)	69.3 (16.2)	69.9 (16.1)	68.8 (16.1)	68.0 (15.8)	69.3 (13.8)
	50% vote	<i>56.8 (11.8)</i>	<i>59.6 (13.9)</i>	64.3 (14.9)	66.1 (16.1)	66.4 (16.3)	66.5 (16.3)
	50% max	67.4 (14.9)	67.1 (16.1)	67.8 (16.1)	67.5 (16.2)	68.8 (15.6)	69.5 (15.8)
	75% mean	72.4 (13.8)	66.8 (15.5)	66.8 (15.9)	67.8 (15.7)	66.9 (16.1)	67.0 (16.0)
	75% prod	72.4 (13.8)	67.0 (15.5)	66.7 (16.0)	68.1 (15.7)	66.8 (15.9)	68.5 (14.6)
	75% vote	<i>61.8 (9.8)</i>	<i>60.5 (11.8)</i>	65.0 (13.9)	65.2 (13.5)	<i>63.8 (13.8)</i>	<i>63.5 (14.2)</i>
	75% max	71.7 (13.9)	66.1 (16.7)	67.2 (16.3)	67.9 (16.4)	68.0 (16.3)	68.6 (15.7)



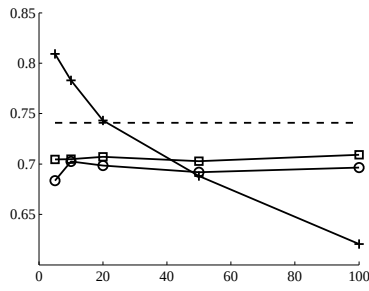
(a) Gauss, mean



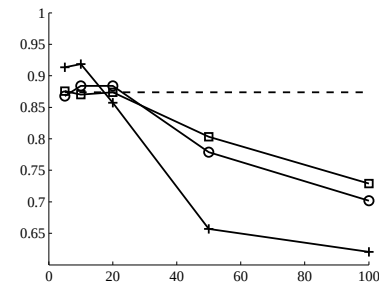
(b) 1-NN, mean



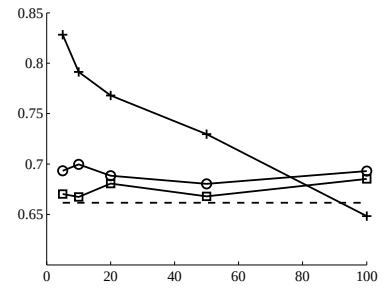
(c) k -Means, mean



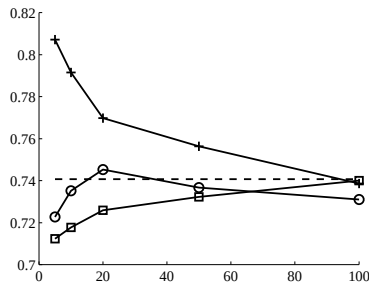
(d) Gauss, prod



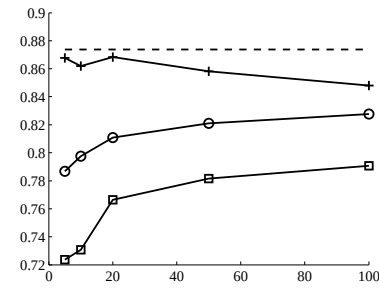
(e) 1-NN, prod



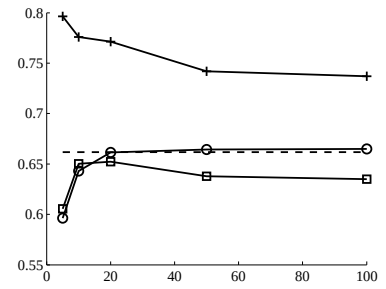
(f) k -Means, prod



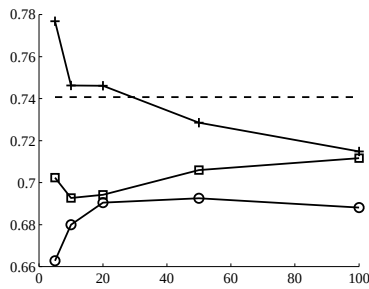
(g) Gauss, vote



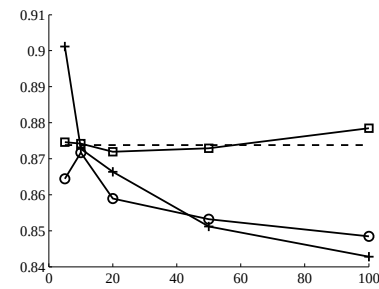
(h) 1-NN, vote



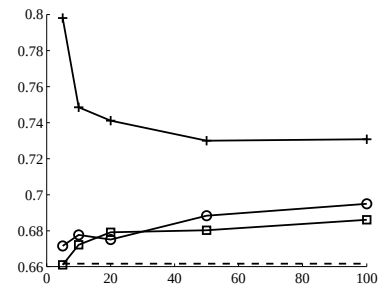
(i) k -Means, vote



(j) Gauss, max



(k) 1-NN, max

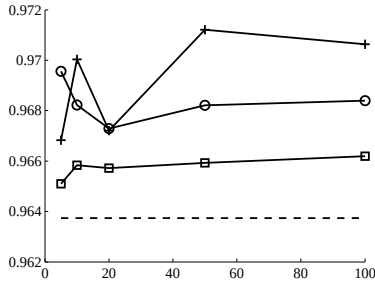


(l) k -Means, max

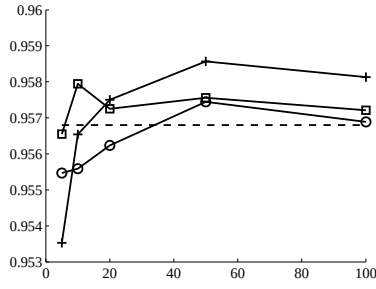
Figure A.4: Parameters RSM, Imports dataset

Table A.4: AUC of RSM, Ionosphere dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

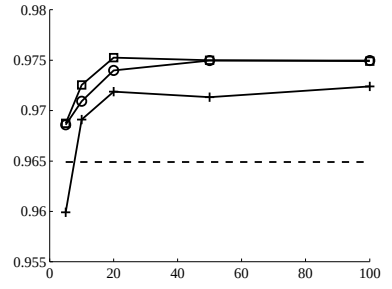
	Classifier	Subspaces					
		2	5	10	20	50	100
Gauss, 96.4 (3.5)	25% mean	<i>95.1 (3.4)</i>	96.7 (2.9)	97.0 (2.6)	96.7 (2.8)	97.1 (2.6)	97.1 (2.7)
	25% prod	<i>95.0 (3.4)</i>	96.7 (2.7)	97.0 (2.7)	96.6 (2.9)	96.2 (3.3)	<i>93.4 (4.8)</i>
	25% vote	<i>90.8 (5.7)</i>	<i>93.3 (5.1)</i>	<i>94.0 (4.2)</i>	<i>94.5 (4.0)</i>	<i>95.2 (4.1)</i>	<i>95.6 (3.8)</i>
	25% max	<i>95.4 (3.3)</i>	96.1 (3.1)	96.3 (2.9)	<i>95.8 (3.1)</i>	<i>95.8 (3.2)</i>	<i>96.0 (3.1)</i>
	50% mean	96.7 (2.7)	97.0 (2.8)	96.8 (2.9)	96.7 (2.9)	96.8 (2.9)	96.8 (2.9)
	50% prod	96.7 (2.7)	97.0 (2.8)	96.9 (2.8)	96.7 (2.9)	<i>95.4 (4.2)</i>	<i>94.1 (4.8)</i>
	50% vote	<i>91.3 (5.2)</i>	<i>92.5 (5.2)</i>	<i>92.9 (5.0)</i>	<i>93.7 (4.9)</i>	<i>94.1 (4.5)</i>	<i>94.2 (4.6)</i>
	50% max	96.7 (2.8)	96.5 (3.4)	96.4 (3.2)	96.5 (3.1)	96.6 (3.1)	96.3 (3.2)
	75% mean	96.5 (3.2)	96.5 (3.2)	96.6 (3.1)	96.6 (3.1)	96.6 (3.2)	96.6 (3.2)
	75% prod	96.5 (3.1)	96.5 (3.2)	96.6 (3.1)	96.5 (3.2)	<i>95.3 (4.2)</i>	<i>94.7 (4.5)</i>
	75% vote	<i>91.1 (5.3)</i>	<i>91.6 (5.3)</i>	<i>91.9 (5.3)</i>	<i>92.8 (5.1)</i>	<i>93.2 (5.0)</i>	<i>93.7 (4.9)</i>
	75% max	96.5 (3.2)	96.5 (3.2)	96.4 (3.2)	96.3 (3.1)	96.3 (3.2)	96.4 (3.2)
1-NN, 95.7 (4.0)	25% mean	<i>93.3 (4.9)</i>	<i>95.4 (4.2)</i>	95.7 (4.0)	95.7 (3.8)	95.9 (3.8)	95.8 (3.8)
	25% prod	<i>92.7 (5.2)</i>	<i>95.2 (4.4)</i>	<i>95.4 (4.2)</i>	<i>95.1 (4.5)</i>	<i>92.4 (5.5)</i>	<i>92.1 (5.2)</i>
	25% vote	<i>89.7 (5.5)</i>	<i>92.8 (4.9)</i>	<i>93.7 (4.9)</i>	<i>93.8 (4.8)</i>	<i>94.2 (4.8)</i>	<i>94.5 (4.8)</i>
	25% max	<i>93.6 (4.8)</i>	<i>94.9 (3.7)</i>	<i>95.1 (4.1)</i>	<i>95.0 (3.7)</i>	<i>94.8 (3.8)</i>	95.6 (3.7)
	50% mean	<i>95.3 (4.3)</i>	<i>95.5 (4.1)</i>	<i>95.6 (4.0)</i>	95.6 (4.0)	95.7 (4.0)	95.7 (4.0)
	50% prod	<i>95.2 (4.3)</i>	<i>95.5 (4.2)</i>	<i>95.5 (4.1)</i>	<i>95.2 (4.5)</i>	<i>94.4 (4.7)</i>	<i>94.2 (4.8)</i>
	50% vote	<i>90.6 (4.8)</i>	<i>92.2 (4.9)</i>	<i>92.6 (5.0)</i>	<i>93.7 (5.0)</i>	<i>94.1 (4.8)</i>	<i>94.3 (4.6)</i>
	50% max	<i>95.5 (4.1)</i>	<i>95.4 (4.2)</i>	<i>95.0 (4.2)</i>	<i>95.3 (4.1)</i>	<i>95.4 (4.1)</i>	95.5 (4.0)
	75% mean	95.9 (4.0)	95.7 (4.1)	95.8 (4.0)	95.7 (4.0)	95.8 (4.0)	95.7 (4.0)
	75% prod	95.9 (4.0)	95.7 (4.1)	95.8 (4.0)	<i>95.4 (4.5)</i>	<i>94.7 (4.8)</i>	<i>94.2 (4.8)</i>
	75% vote	<i>90.8 (4.9)</i>	<i>91.5 (5.2)</i>	<i>92.6 (5.0)</i>	<i>92.8 (4.9)</i>	<i>92.8 (4.7)</i>	<i>93.1 (4.8)</i>
	75% max	95.8 (4.0)	95.7 (4.1)	<i>95.5 (4.1)</i>	<i>95.3 (4.2)</i>	<i>95.4 (4.2)</i>	<i>95.4 (4.1)</i>
<i>k</i> -Means, 96.5 (3.5)	25% mean	<i>94.3 (3.9)</i>	96.0 (3.0)	96.9 (2.8)	97.2 (2.7)	97.1 (2.7)	97.2 (2.7)
	25% prod	<i>94.3 (3.9)</i>	<i>95.9 (3.1)</i>	96.9 (2.7)	97.2 (2.8)	97.0 (2.9)	96.8 (3.4)
	25% vote	<i>87.7 (5.3)</i>	<i>94.4 (4.0)</i>	<i>95.4 (3.6)</i>	96.3 (3.4)	96.7 (3.1)	96.8 (3.1)
	25% max	<i>94.5 (3.9)</i>	<i>95.5 (3.2)</i>	96.1 (3.0)	96.7 (3.0)	96.9 (2.8)	97.0 (2.8)
	50% mean	<i>95.7 (3.2)</i>	96.9 (2.8)	97.1 (2.8)	97.4 (2.5)	97.5 (2.5)	97.5 (2.5)
	50% prod	<i>95.7 (3.2)</i>	96.9 (2.8)	97.1 (2.8)	97.4 (2.5)	97.4 (2.7)	97.4 (2.7)
	50% vote	<i>89.1 (7.0)</i>	<i>93.2 (4.9)</i>	<i>95.0 (4.1)</i>	96.0 (3.4)	96.4 (3.3)	96.6 (3.2)
	50% max	<i>95.6 (3.3)</i>	96.8 (2.8)	97.0 (2.9)	97.2 (2.6)	97.2 (2.8)	97.1 (2.9)
	75% mean	96.3 (3.3)	96.9 (2.7)	97.3 (2.4)	97.5 (2.3)	97.5 (2.4)	97.5 (2.4)
	75% prod	96.3 (3.3)	96.9 (2.6)	97.3 (2.4)	97.6 (2.3)	97.4 (2.7)	97.4 (2.7)
	75% vote	<i>90.3 (6.6)</i>	<i>92.3 (6.0)</i>	<i>93.9 (5.0)</i>	<i>94.9 (4.9)</i>	<i>95.1 (4.6)</i>	<i>95.3 (4.5)</i>
	75% max	96.3 (3.3)	97.1 (2.6)	97.4 (2.6)	97.7 (2.6)	97.6 (2.5)	97.5 (2.6)



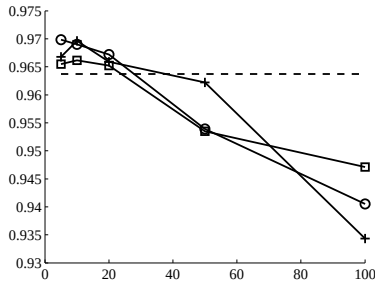
(a) Gauss, mean



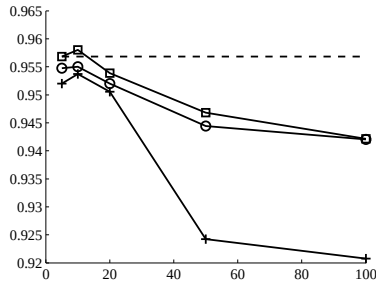
(b) 1-NN, mean



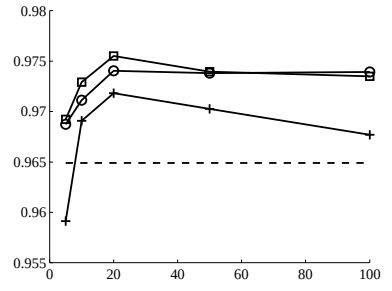
(c) k -Means, mean



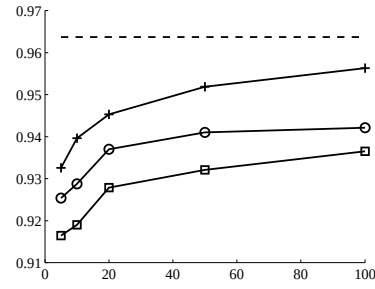
(d) Gauss, prod



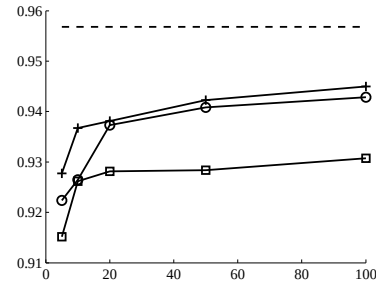
(e) 1-NN, prod



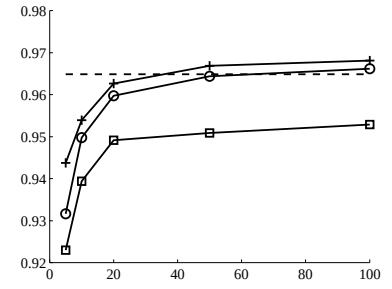
(f) k -Means, prod



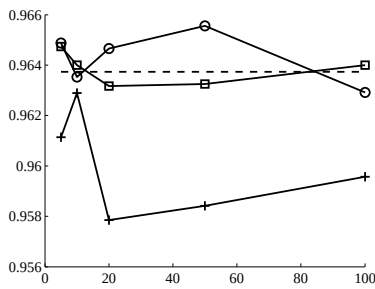
(g) Gauss, vote



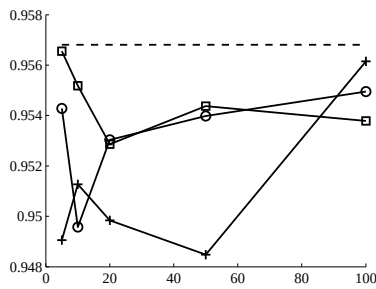
(h) 1-NN, vote



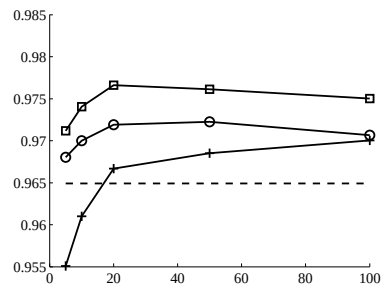
(i) k -Means, vote



(j) Gauss, max



(k) 1-NN, max

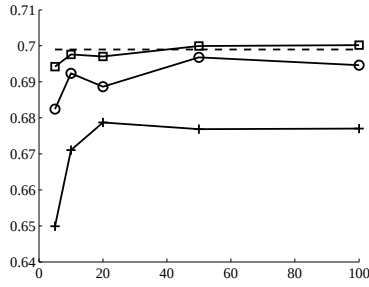


(l) k -Means, max

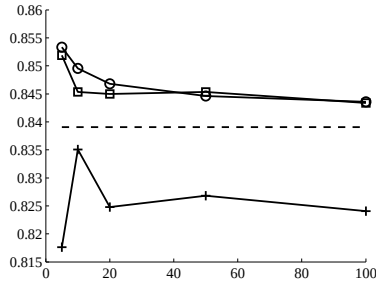
Figure A.5: Parameters RSM, Ionosphere dataset

Table A.5: AUC of RSM, Sonar dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

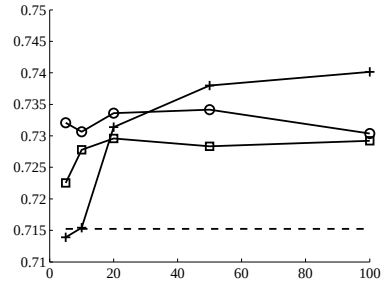
		Subspaces					
Classifier		2	5	10	20	50	100
Gauss, 69.9 (12.9)	25% mean	65.8 (13.0)	65.0 (12.5)	67.1 (11.8)	67.9 (12.2)	67.7 (11.9)	67.7 (12.0)
	25% prod	65.7 (12.9)	64.8 (12.4)	67.1 (11.7)	67.8 (12.2)	66.0 (11.7)	59.0 (9.4)
	25% vote	53.8 (9.4)	56.4 (9.7)	64.0 (11.0)	66.7 (10.8)	67.3 (12.0)	66.8 (11.9)
	25% max	66.0 (12.9)	65.4 (11.8)	70.2 (11.7)	70.0 (12.0)	69.2 (12.4)	69.6 (11.7)
	50% mean	68.5 (12.2)	68.2 (11.8)	69.2 (11.8)	68.9 (12.3)	69.7 (12.5)	69.5 (12.3)
	50% prod	68.5 (12.0)	68.2 (11.7)	69.2 (11.9)	69.0 (12.3)	69.0 (12.1)	67.5 (11.7)
	50% vote	62.7 (9.8)	63.5 (10.3)	66.6 (11.0)	66.5 (11.8)	67.8 (12.1)	68.2 (11.9)
	50% max	68.5 (12.4)	67.6 (12.3)	68.7 (11.8)	68.3 (12.5)	69.1 (13.2)	69.7 (13.0)
	75% mean	69.3 (12.8)	69.4 (12.3)	69.8 (12.5)	69.7 (12.7)	70.0 (12.7)	70.0 (12.6)
	75% prod	69.3 (12.8)	69.4 (12.3)	69.8 (12.5)	69.7 (12.7)	69.7 (12.8)	68.8 (12.1)
	75% vote	66.4 (11.6)	67.1 (11.8)	68.0 (12.2)	68.0 (12.6)	68.0 (12.2)	68.0 (12.7)
	75% max	69.3 (12.7)	69.6 (12.3)	69.7 (12.3)	69.9 (12.8)	70.4 (12.3)	70.1 (12.3)
1-NN, 83.9 (8.4)	25% mean	85.2 (8.1)	81.8 (8.4)	83.5 (7.8)	82.5 (8.6)	82.7 (8.7)	82.4 (8.6)
	25% prod	85.0 (7.9)	81.4 (8.2)	83.5 (7.7)	82.3 (8.6)	80.7 (8.9)	73.2 (9.5)
	25% vote	72.2 (9.7)	74.4 (8.7)	79.7 (8.6)	80.3 (9.1)	81.8 (8.8)	81.4 (8.8)
	25% max	84.9 (8.2)	81.4 (8.1)	83.2 (7.9)	81.3 (9.0)	82.3 (8.7)	80.8 (9.4)
	50% mean	85.0 (7.4)	85.3 (7.2)	85.0 (7.6)	84.7 (8.0)	84.5 (8.2)	84.4 (8.5)
	50% prod	84.9 (7.5)	85.3 (7.1)	85.0 (7.5)	84.7 (7.8)	83.0 (8.8)	76.8 (9.0)
	50% vote	73.3 (7.5)	75.6 (8.0)	76.1 (8.5)	76.1 (9.1)	78.0 (9.5)	78.9 (9.7)
	50% max	85.1 (7.4)	85.7 (7.1)	85.3 (7.5)	84.8 (8.1)	84.5 (8.3)	84.4 (8.3)
	75% mean	85.0 (7.9)	85.2 (8.0)	84.5 (8.1)	84.5 (8.4)	84.5 (8.5)	84.3 (8.4)
	75% prod	85.0 (7.9)	85.2 (8.0)	84.6 (8.1)	84.5 (8.4)	82.5 (9.0)	76.0 (8.7)
	75% vote	70.0 (9.2)	70.8 (8.7)	70.9 (9.1)	72.6 (9.6)	73.3 (9.2)	73.2 (9.4)
	75% max	84.8 (8.1)	84.9 (8.2)	84.5 (8.3)	84.3 (8.2)	83.9 (8.4)	83.8 (8.5)
k -Means, 71.5 (10.9)	25% mean	70.9 (10.7)	71.4 (10.6)	71.5 (10.3)	73.1 (9.7)	73.8 (9.8)	74.0 (10.3)
	25% prod	71.0 (10.7)	71.4 (10.6)	71.6 (10.3)	73.2 (9.6)	74.1 (9.7)	72.0 (10.3)
	25% vote	55.9 (10.1)	59.9 (10.0)	61.3 (10.7)	65.3 (11.0)	69.2 (9.8)	70.7 (10.1)
	25% max	70.4 (10.7)	70.8 (9.5)	71.7 (10.6)	72.1 (10.1)	73.6 (9.4)	75.5 (9.9)
	50% mean	72.5 (11.0)	73.2 (10.2)	73.1 (10.9)	73.4 (10.7)	73.4 (10.4)	73.0 (10.4)
	50% prod	72.5 (10.9)	73.3 (10.2)	73.1 (10.9)	73.5 (10.7)	73.6 (10.3)	72.1 (10.4)
	50% vote	60.2 (8.8)	62.7 (8.0)	64.2 (9.7)	65.1 (9.9)	66.8 (9.9)	68.3 (9.8)
	50% max	72.4 (11.0)	73.1 (9.9)	72.1 (11.4)	72.3 (11.3)	74.0 (10.4)	74.9 (9.0)
	75% mean	71.1 (9.9)	72.3 (10.7)	72.8 (10.2)	73.0 (10.4)	72.8 (10.2)	72.9 (10.2)
	75% prod	71.2 (10.0)	72.3 (10.8)	72.8 (10.2)	73.0 (10.5)	73.0 (10.1)	72.0 (10.2)
	75% vote	57.6 (8.8)	59.4 (8.9)	60.1 (8.9)	62.0 (9.3)	64.0 (9.5)	65.0 (9.2)
	75% max	71.1 (10.0)	72.1 (10.2)	72.9 (10.5)	73.2 (9.8)	73.8 (9.8)	74.4 (9.7)



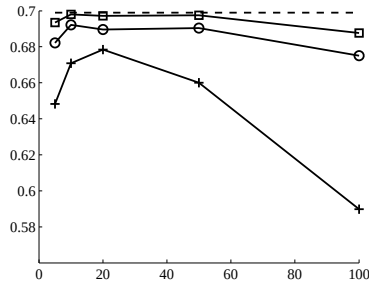
(a) Gauss, mean



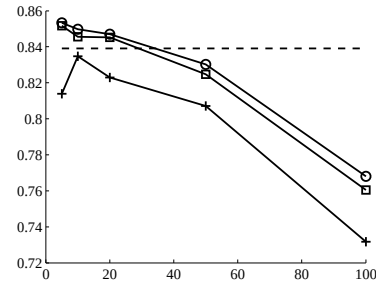
(b) 1-NN, mean



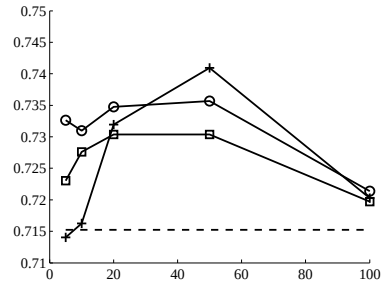
(c) k -Means, mean



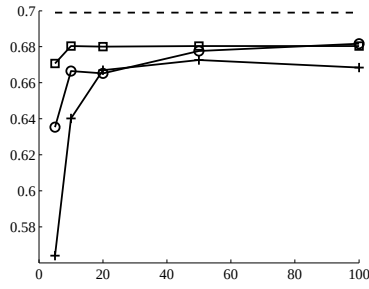
(d) Gauss, prod



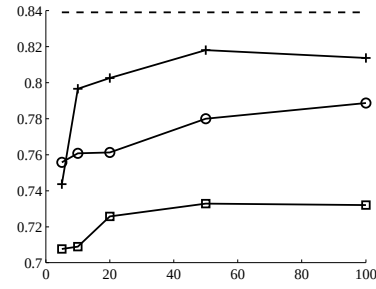
(e) 1-NN, prod



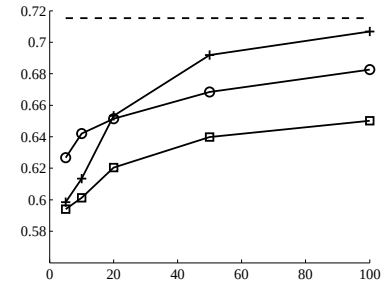
(f) k -Means, prod



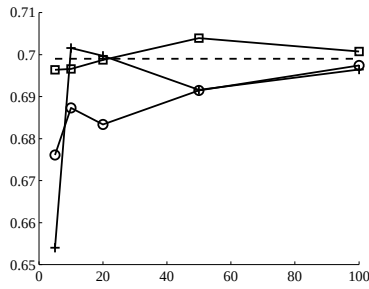
(g) Gauss, vote



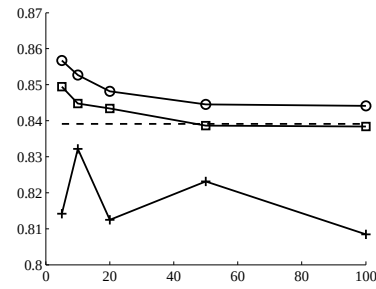
(h) 1-NN, vote



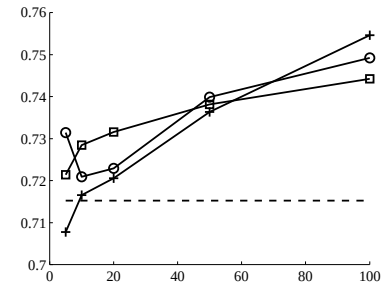
(i) k -Means, vote



(j) Gauss, max



(k) 1-NN, max

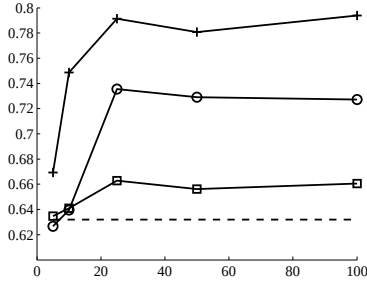


(l) k -Means, max

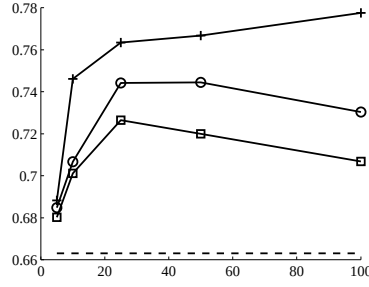
Figure A.6: Parameters RSM, Sonar dataset

Table A.6: AUC of RSM, Spam dataset. Bold results indicate significantly better performances than the base classifier, italic results indicate significantly worse performances than the base classifier.

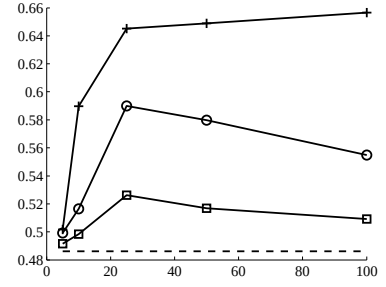
		Subspaces					
Classifier		2	5	10	20	50	100
Gauss, 63.2 (3.0)	25% mean	<i>60.6 (2.3)</i>	66.9 (1.9)	74.9 (1.9)	79.1 (1.9)	78.1 (1.9)	79.4 (1.9)
	25% max	<i>58.5 (2.3)</i>	67.9 (1.9)	77.6 (1.9)	82.5 (1.8)	81.7 (2.0)	81.9 (2.0)
	25% prod	<i>62.4 (2.2)</i>	68.7 (2.1)	76.4 (1.8)	68.6 (1.8)	<i>59.0 (1.1)</i>	<i>57.4 (1.1)</i>
	25% vote	<i>61.5 (1.7)</i>	67.0 (1.6)	75.8 (1.8)	79.4 (1.9)	78.1 (1.9)	79.0 (2.0)
	50% mean	<i>61.9 (3.2)</i>	<i>62.7 (3.0)</i>	64.0 (3.0)	73.5 (2.3)	72.9 (2.4)	72.7 (2.5)
	50% max	<i>62.1 (3.2)</i>	<i>60.1 (2.9)</i>	<i>61.6 (2.7)</i>	81.0 (2.0)	81.2 (2.0)	81.4 (2.0)
	50% prod	<i>61.8 (3.2)</i>	63.9 (3.0)	65.4 (3.0)	<i>58.8 (2.1)</i>	<i>52.2 (1.1)</i>	<i>51.4 (0.8)</i>
	50% vote	<i>47.7 (0.9)</i>	<i>49.6 (1.1)</i>	<i>51.9 (1.4)</i>	74.9 (2.4)	74.7 (2.4)	74.8 (2.4)
	75% mean	<i>62.0 (3.2)</i>	63.5 (3.1)	64.1 (2.9)	66.3 (2.7)	65.6 (2.8)	66.1 (2.9)
	75% max	<i>62.3 (3.1)</i>	<i>62.9 (2.9)</i>	<i>63.4 (2.7)</i>	66.5 (2.7)	66.7 (2.5)	77.4 (2.2)
	75% prod	<i>61.9 (3.2)</i>	63.4 (3.1)	64.3 (2.9)	<i>51.7 (1.6)</i>	<i>49.3 (1.2)</i>	<i>48.7 (0.9)</i>
	75% vote	<i>47.7 (0.9)</i>	<i>50.9 (1.4)</i>	<i>50.8 (1.4)</i>	<i>58.5 (2.1)</i>	<i>58.8 (2.1)</i>	69.2 (2.3)
1-NN, 66.3 (3.1)	25% mean	<i>63.9 (2.6)</i>	68.8 (2.7)	74.6 (2.7)	76.3 (2.6)	76.7 (2.5)	77.8 (2.5)
	25% max	<i>64.1 (2.6)</i>	70.3 (2.6)	78.2 (2.6)	81.3 (2.3)	82.3 (2.2)	83.9 (2.2)
	25% prod	<i>54.1 (2.4)</i>	<i>50.5 (2.3)</i>	<i>53.2 (2.3)</i>	<i>54.6 (1.6)</i>	<i>53.2 (1.3)</i>	<i>53.0 (1.2)</i>
	25% vote	<i>57.1 (2.1)</i>	<i>63.5 (2.4)</i>	73.0 (2.7)	77.5 (2.5)	77.8 (2.5)	78.9 (2.4)
	50% mean	<i>60.9 (3.1)</i>	68.5 (2.8)	70.7 (2.9)	74.4 (2.8)	74.4 (2.9)	73.0 (2.9)
	50% max	<i>61.6 (3.1)</i>	70.0 (2.8)	72.0 (2.8)	77.0 (2.7)	77.1 (2.6)	76.9 (2.5)
	50% prod	<i>57.8 (3.0)</i>	<i>64.0 (2.5)</i>	<i>63.9 (2.7)</i>	<i>58.7 (2.5)</i>	<i>53.5 (1.9)</i>	<i>50.9 (1.3)</i>
	50% vote	<i>50.4 (1.2)</i>	<i>58.9 (2.0)</i>	<i>63.3 (2.5)</i>	71.6 (2.6)	71.6 (2.6)	71.3 (2.6)
	75% mean	<i>64.7 (3.1)</i>	68.0 (3.2)	70.1 (3.0)	72.6 (2.9)	72.0 (2.9)	70.7 (3.0)
	75% max	<i>65.4 (3.1)</i>	71.1 (3.0)	71.1 (3.0)	73.7 (2.8)	73.2 (2.8)	73.6 (2.8)
	75% prod	<i>63.6 (3.1)</i>	67.4 (3.1)	67.4 (2.8)	<i>60.7 (2.6)</i>	<i>53.6 (1.7)</i>	<i>50.8 (1.2)</i>
	75% vote	<i>50.7 (1.1)</i>	<i>57.8 (2.1)</i>	<i>59.0 (2.3)</i>	<i>66.3 (2.5)</i>	<i>66.0 (2.5)</i>	<i>66.2 (2.4)</i>
<i>k</i> -Means, 48.6 (2.7)	25% mean	<i>48.6 (2.7)</i>	50.2 (2.5)	59.0 (2.4)	64.5 (2.6)	64.9 (2.5)	65.7 (2.4)
	25% max	<i>46.9 (2.8)</i>	49.8 (2.2)	60.5 (2.0)	68.6 (2.5)	68.8 (2.5)	69.3 (2.4)
	25% prod	50.4 (3.0)	52.1 (2.7)	59.3 (2.6)	64.2 (2.5)	51.8 (1.4)	50.6 (1.1)
	25% vote	52.9 (1.6)	54.2 (2.1)	61.6 (2.3)	67.0 (2.7)	65.4 (2.7)	66.6 (2.5)
	50% mean	<i>48.4 (2.7)</i>	49.9 (2.7)	51.6 (2.6)	59.0 (2.5)	58.0 (2.4)	55.5 (2.5)
	50% max	<i>48.3 (2.7)</i>	<i>48.3 (3.4)</i>	52.0 (2.5)	64.2 (2.0)	63.9 (2.0)	62.6 (2.1)
	50% prod	<i>48.5 (2.7)</i>	51.0 (2.6)	52.9 (2.5)	57.3 (2.3)	<i>46.9 (1.3)</i>	<i>47.3 (0.8)</i>
	50% vote	<i>47.9 (0.9)</i>	<i>46.0 (1.2)</i>	<i>47.8 (1.5)</i>	58.9 (2.3)	58.2 (2.3)	57.6 (2.3)
	75% mean	<i>48.5 (2.6)</i>	49.2 (2.7)	49.8 (2.8)	52.6 (2.6)	51.7 (2.6)	50.9 (2.7)
	75% max	<i>48.5 (2.7)</i>	<i>48.4 (3.5)</i>	<i>48.0 (3.1)</i>	60.1 (2.5)	59.7 (2.3)	59.7 (2.2)
	75% prod	<i>48.6 (2.6)</i>	49.8 (2.6)	51.3 (2.6)	51.1 (2.4)	<i>44.3 (1.3)</i>	<i>46.6 (1.1)</i>
	75% vote	<i>47.9 (0.9)</i>	<i>46.2 (1.2)</i>	<i>45.8 (1.3)</i>	50.9 (1.8)	50.8 (1.8)	52.1 (1.8)



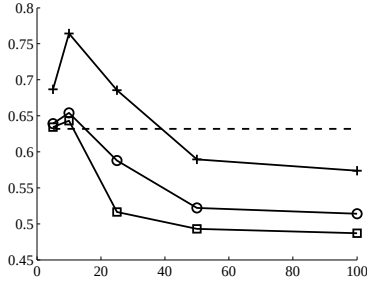
(a) Gauss, mean



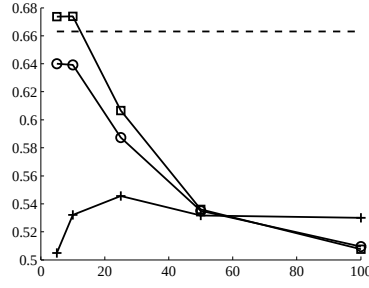
(b) 1-NN, mean



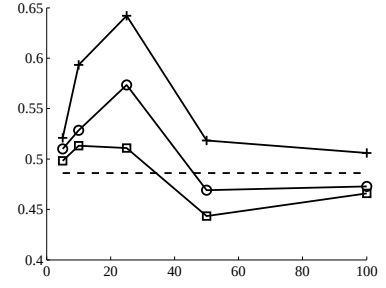
(c) k -Means, mean



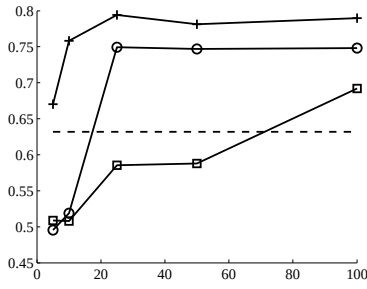
(d) Gauss, prod



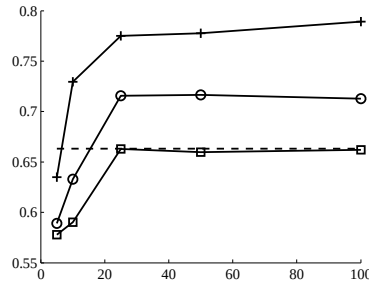
(e) 1-NN, prod



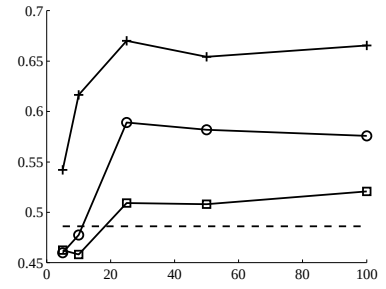
(f) k -Means, prod



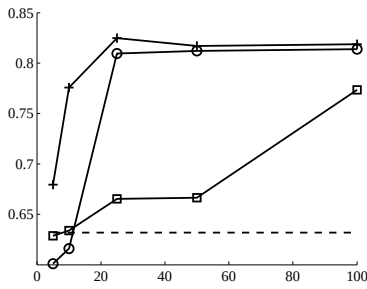
(g) Gauss, vote



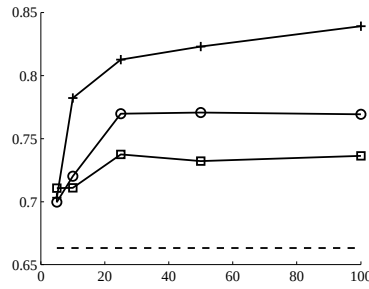
(h) 1-NN, vote



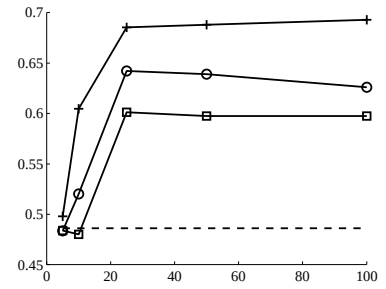
(i) k -Means, vote



(j) Gauss, max



(k) 1-NN, max



(l) k -Means, max

Figure A.7: Parameters RSM, Spam dataset

A.2 PRSM

A.2.1 RSM vs PRSM

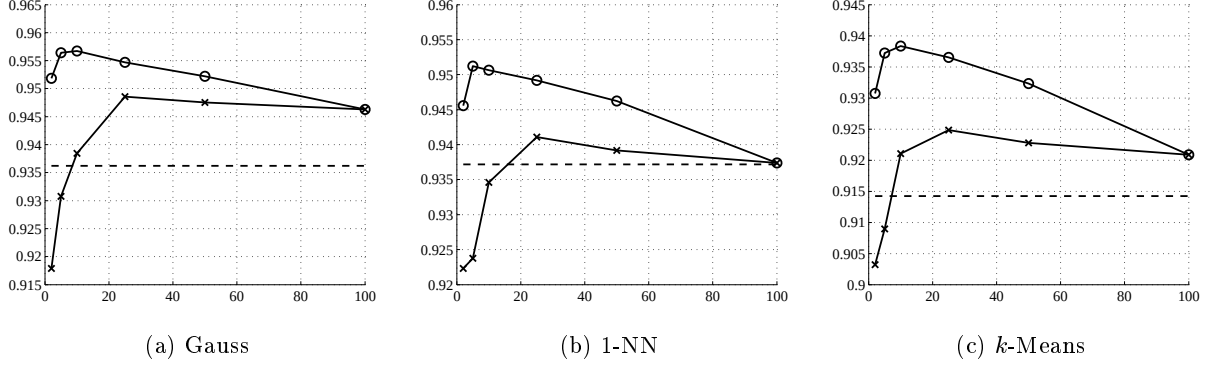


Figure A.8: RSM vs. PRSM, Concordia dataset

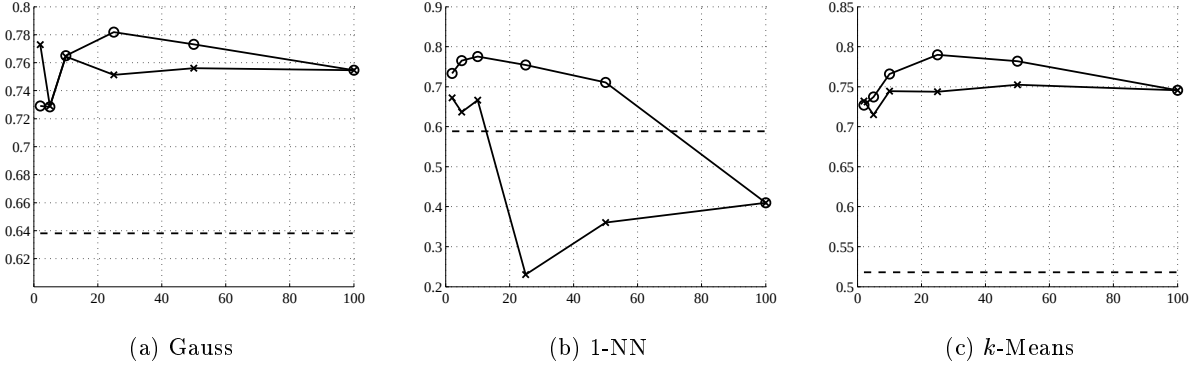


Figure A.9: RSM vs. PRSM, Hepatitis dataset

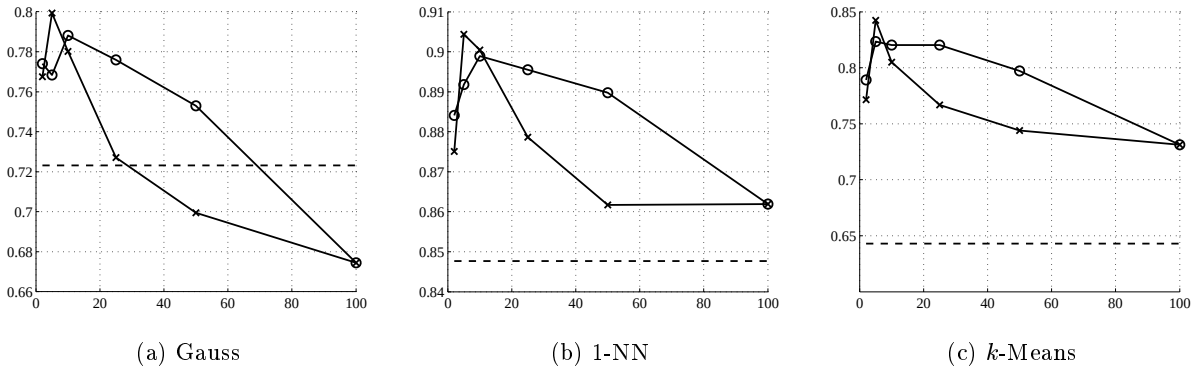
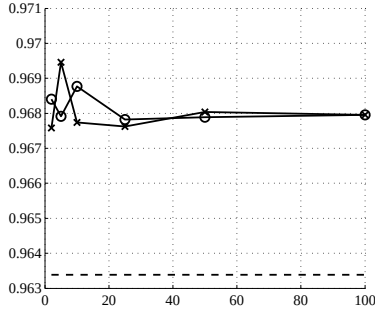
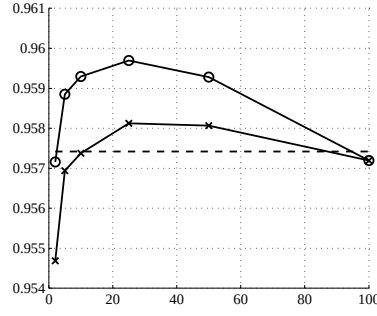


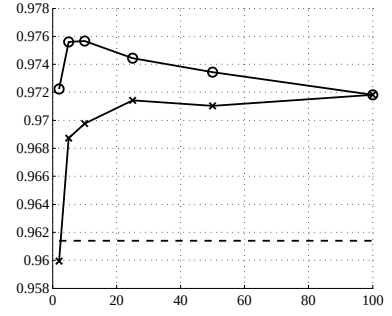
Figure A.10: RSM vs. PRSM, Imports dataset



(a) Gauss

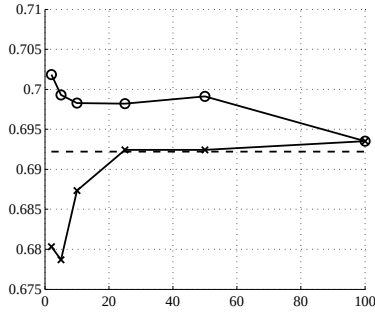


(b) 1-NN

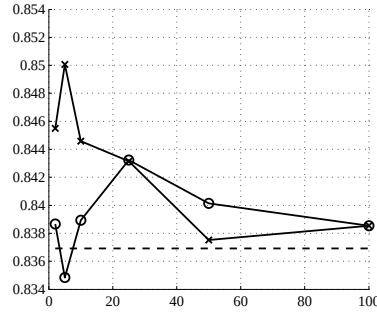


(c) k -Means

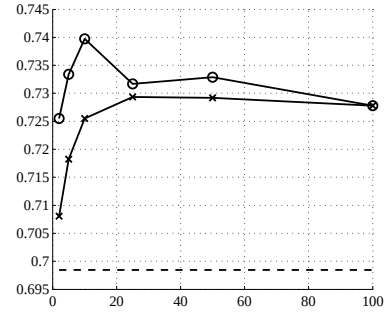
Figure A.11: RSM vs. PRSM, Ionosphere dataset



(a) Gauss

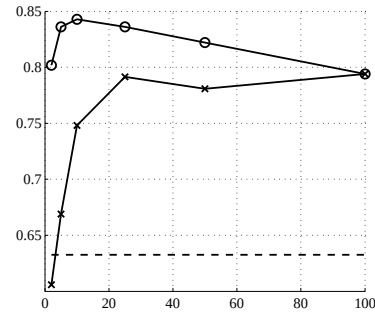


(b) 1-NN

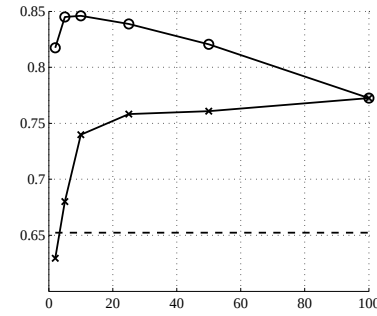


(c) k -Means

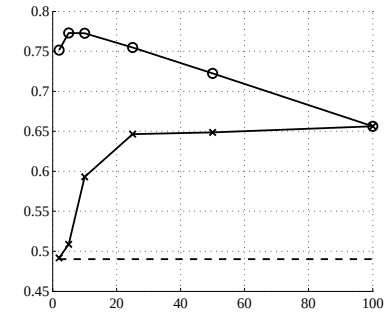
Figure A.12: RSM vs. PRSM, Sonar dataset



(a) Gauss



(b) 1-NN



(c) k -Means

Figure A.13: RSM vs. PRSM, Spam dataset

A.2.2 Pool size

Table A.7: AUC of PRSM, Concordia dataset. The bold numbers indicate the results which are not significantly worse than the best result per row.

Classifier	Select	Pool size		
		100	250	1000
Gauss 93.6 (2.5)	10	95.7 (1.9)	95.5 (1.9)	95.7 (1.9)
	25	95.5 (2.0)	95.5 (2.0)	95.6 (1.9)
	100	94.6 (2.3)	95.3 (2.0)	95.5 (2.0)
1-NN 93.7 (1.8)	10	95.1 (1.7)	95.2 (1.6)	95.5 (1.5)
	25	94.9 (1.7)	95.2 (1.6)	95.6 (1.5)
	100	93.7 (1.8)	94.8 (1.6)	95.4 (1.5)
<i>k</i> -Means 91.2 (3.0)	10	94.0 (2.3)	94.3 (2.2)	94.6 (2.4)
	25	93.7 (2.4)	94.2 (2.4)	94.6 (2.3)
	100	92.2 (2.6)	93.6 (2.5)	94.3 (2.3)

Table A.8: AUC of PRSM, Hepatitis dataset. The bold numbers indicate the results which are not significantly worse than the best result per row.

Classifier	Select	Pool size		
		100	250	1000
Gauss 63.8 (16.9)	10	76.5 (16.2)	76.5 (16.3)	76.2 (16.6)
	25	78.2 (15.0)	77.6 (16.6)	77.1 (16.1)
	100	75.5 (14.1)	79.2 (15.2)	77.6 (16.9)
1-NN 58.9 (17.4)	10	77.5 (14.6)	77.4 (15.3)	76.7 (15.6)
	25	75.4 (15.5)	77.3 (15.5)	77.9 (16.0)
	100	41.0 (19.4)	70.8 (19.8)	77.5 (15.5)
<i>k</i> -Means 54.9 (16.5)	10	73.4 (17.5)	74.9 (16.5)	76.7 (16.0)
	25	75.8 (17.0)	75.3 (17.2)	76.6 (16.3)
	100	76.3 (13.7)	76.3 (15.0)	76.9 (16.8)

Table A.9: AUC of PRSM, Imports dataset. The bold numbers indicate the results which are not significantly worse than the best result per row.

Classifier	Select	Pool size		
		100	250	1000
Gauss 72.3 (14.1)	10	78.8 (10.2)	78.2 (11.1)	78.6 (10.7)
	25	77.6 (11.2)	78.9 (11.1)	77.8 (12.0)
	100	67.4 (13.4)	76.7 (10.9)	78.0 (11.8)
1-NN 84.8 (12.6)	10	89.9 (8.3)	89.2 (9.3)	88.1 (9.5)
	25	89.5 (7.8)	89.7 (8.7)	89.6 (8.8)
	100	86.2 (10.0)	89.0 (8.9)	90.2 (8.3)
<i>k</i> -Means 63.8 (17.6)	10	81.5 (11.9)	82.2 (11.1)	81.7 (12.0)
	25	82.0 (12.1)	82.3 (11.7)	82.4 (12.1)
	100	73.4 (14.2)	80.6 (12.0)	82.6 (11.7)

Table A.10: AUC of PRSM, Ionosphere dataset. The bold numbers indicate the results which are not significantly worse than the best result per row.

Classifier	Select	Pool size		
		100	250	1000
Gauss 96.3 (4.4)	10	96.9 (4.0)	96.8 (4.2)	96.8 (4.2)
	25	96.8 (4.1)	96.7 (4.1)	96.7 (4.1)
	100	96.8 (4.0)	96.8 (4.0)	96.8 (4.1)
1-NN 95.7 (4.9)	10	95.9 (4.7)	96.1 (4.7)	96.1 (4.7)
	25	96.0 (4.7)	96.1 (4.7)	96.1 (4.7)
	100	95.7 (4.9)	96.0 (4.7)	96.1 (4.6)
<i>k</i> -Means 96.6 (2.5)	10	97.5 (2.4)	97.6 (2.3)	97.5 (2.3)
	25	97.5 (2.5)	97.5 (2.5)	97.6 (2.3)
	100	97.2 (2.5)	97.6 (2.3)	97.7 (2.2)

Table A.11: AUC of PRSM, Sonar dataset. The bold numbers indicate the results which are not significantly worse than the best result per row.

Classifier	Select	Pool size		
		100	250	1000
Gauss OCC 69.2 (12.2)	10	69.8 (12.0)	70.3 (11.9)	70.7 (11.7)
	25	69.8 (11.7)	70.2 (11.7)	70.5 (11.5)
	100	69.4 (12.3)	69.8 (11.9)	70.2 (11.9)
1-NN OCC 83.7 (9.2)	10	83.9 (9.7)	84.0 (9.8)	83.8 (9.7)
	25	84.3 (9.2)	84.2 (9.4)	84.1 (9.3)
	100	83.9 (9.1)	84.0 (9.3)	84.4 (9.5)
<i>k</i> -Means OCC 69.0 (11.4)	10	73.6 (12.0)	73.3 (12.2)	74.7 (12.4)
	25	73.0 (11.9)	73.3 (11.9)	74.0 (12.3)
	100	72.3 (11.5)	72.8 (11.7)	73.7 (12.0)

Table A.12: AUC of PRSM, Spam dataset. The bold numbers indicate the results which are not significantly worse than the best result per row.

Classifier	Select	Pool size		
		100	250	1000
Gauss OCC 63.3 (3.1)	10	84.3 (1.9)	83.8 (2.1)	86.2 (2.0)
	25	83.6 (1.6)	84.7 (1.7)	86.3 (1.8)
	100	79.4 (2.0)	82.6 (1.8)	85.6 (1.7)
1-NN OCC 65.2 (2.6)	10	84.6 (1.8)	85.9 (1.7)	87.5 (1.8)
	25	83.9 (1.8)	85.2 (1.7)	87.2 (1.8)
	100	77.3 (2.3)	82.6 (1.8)	85.7 (1.8)
<i>k</i> -Means OCC 48.8 (2.5)	10	77.5 (1.9)	78.9 (2.1)	80.2 (2.4)
	25	75.6 (1.8)	77.8 (1.9)	80.5 (2.0)
	100	65.7 (2.2)	72.4 (1.7)	77.8 (1.8)