Dissertation

Towards Understanding Stacking
Studies of a General Ensemble Learning Scheme

ausgeführt zum Zwecke der Erlangung des
akademischen Grades eines
Doktors der technischen Naturwissenschaften

betreut von ao.Univ.-Prof.Dr. Gerhard Widmer,
Institut für Medizinische Kybernetik und
Artificial Intelligence (IMKAI)

eingereicht an der Technischen Universität Wien,
Fakultät für Naturwissenschaften und Informatik

von

Dipl.-Ing. Alexander K. Seewald
alexsee@oefai.at http://alex.seewald.at alex@seewald.at

wohnhaft Anton-Krieger-Gasse 78/5, 1230 Wien
Matrikelnummer 9425006
Geboren am 08. Dezember 1975 in Wien

Wien, im Februar 2003 ______________________________

Deutsche Kurzfassung

Diese Dissertation besteht aus komplementären Studien, die den Ensemble-Lernalgorithmus Stacking (Wolpert, 1992) untersuchen. Verschiedene Facetten seines Verhaltens und seine Beziehung zu anderen Ensemble-Lernalgorithmen werden untersucht, letzteres in zweifacher Weise: Einerseits indem wir zeigen, daß es üblicherweise die beste Wahl ist[1]; andererseits durch Aufzeigen der Äquivalenz von Stacking mit vielen anderen Ensemble-Lernalgorithmen (Kapitel 7) – d.h. der Allgemeinheit von Stacking.

Im ersten Kapitel wird erklärt, was Stacking ist und wie es funktioniert, gefolgt von einer Übersicht der Kapitel mit zusammengefaßten Ergebnissen.

Im Kapitel 2 befindet sich eine Übersicht der verwandten Forschungsliteratur, die einen Querschnitt der aktuellen Forschung auf diesem Gebiet darstellt.

Im Kapitel 3 erforschen wir den Parameterraum von Stacking. Wir untersuchen Stacking systematisch mit einem vollständigen Set von Level-0 Lernalgorithmen, verschiedenen Level-1 Lernalgorithmen und zwei verwandten Arten von Meta-Daten. Wir schlagen theoretisch und empirisch fundierte Defaultwerte für alle Parameter vor. Dieses Kapitel basiert auf (Seewald, 2002c).

Im Kapitel 4 untersuchen wir Meta-Regressionslernen, d.h. die Voraussage von Stackings Fehlerrate aus einer Anzahl von Charakterisierungen von zu lernenden Daten und Level-0 Lernalgorithmen. Wir untersuchen auch Meta-Klassifikationslernen, d.h. die direkte Voraussage von signifikanten Unterschieden zwischen verwandten Ensemble-Lernalgorithmen. Dieses Kapitel basiert auf (Seewald, 2002b).

Im Kapitel 5 stellen wir die Variante StackingC vor, welche die Genauigkeit von Stacking verbessert, den Lernvorgang beschleunigt und zusätzlich eine signifikante Schwäche behebt. Dieses Kapitel basiert auf (Seewald, 2002a).

Im Kapitel 6 stellen wir die Ergebnisse eines alternativen Paradigmas zum Vergleich zwischen Lernalgorithmen vor. Wir untersuchen die Hypothese, daß StackingC stabiler ist als andere Ensemble-Lernalgorithmen, d.h. daß seine Lernkurve über allen anderen Lernkurven liegt. Überraschenderweise finden wir heraus, daß in diesem Fall keine signifikanten Unterschiede zwischen den untersuchten Lernalgorithmen auftauchen.

Im Kapitel 7 zeigen wir, daß viele Ensemble-Lernalgorithmen wie StackingC, Grading (Seewald & Fürnkranz, 2001) und sogar Bagging (Breiman, 1996) von Stacking simuliert werden können. Wir geben in den meisten Fällen äquivalente Definitionen als Level-1 Lernalgorithmen für Stacking an.

Im Kapitel 8 stellen wir das Forschungsgebiet Informationsvisualisierung vor und wenden es an, um unsere 26 Lernprobleme und umfangreiche experimentelle Ergebnisse zu visualisieren. Wir finden heraus, daß die Mehrzahl der falsch klassifizierten Beispiele dadurch entstehen, weil keiner der Level-0 Lernalgorithmen korrekt vorhersagt. Obwohl Stacking potentiell in der Lage wäre, selbst in dieser Situation zu lernen, wird dies nicht beobachtet. Ganz im Gegenteil, Stacking sagt sogar dann falsch voraus, wenn eine Mehrheit der Level-0 Lernalgorithmen richtig liegt. Graphiken von allen Lernproblemen sind im Appendix A dargestellt.

In Kapitel 9 geben wir abschließend eine Zusammenfassung unserer hauptsächlichen Ergebnisse, generelle Schlußfolgerungen und einen Ausblick auf zukünftige Forschungen an.

[1] Wir konnten dieses Ergebnis mittels StackingC noch verbessern, siehe Kapitel 5.

Abstract

This thesis consists of complementary studies concerned with the ensemble learning scheme Stacking (Wolpert, 1992) We will explore various aspects of its behaviour and also clarify its relation to related ensemble learning schemes in two ways: by showing that it is usually the best choice among ensemble learning schemes[2] and also by demonstrating that most ensemble learning schemes can be simulated by Stacking (Chapter 7), making it the most general ensemble learning scheme.

In the first chapter, we give an overview of what Stacking is and how it works, combined with a short roadmap to this thesis.

In Chapter 2, we present an overview of related research for Stacking, considering both state of the art approaches and unique contributions to this field.

In Chapter 3, we explore the parameter state space of Stacking. We systematically investigate Stacking with an exhaustive set of base classifiers, diverse meta classifiers and two related types of meta data. We propose default settings of all these parameters, grounded by empirical and theoretical arguments. This chapter is an extended version of (Seewald, 2002c).

In Chapter 4, we investigate regression meta learning, trying to predict Stacking's accuracy from a variety of dataset-related and base-classifier-related features. We also investigate classification meta learning, trying to predict significant differences between related ensemble learning schemes directly on a dataset-by-dataset basis. This chapter is an extended version of (Seewald, 2002b).

In Chapter 5, we introduce a variant, StackingC, which improves Stacking's performance further, reduces computational cost and also resolves a significant weakness found during the course of our experiments. This chapter is an extended version of (Seewald, 2002a).

In Chapter 6, we present results from an alternative paradigm to compare classifiers. We investigate the hypothesis that StackingC is more stable than other ensemble learning schemes, i.e. that its learning curve is at the uppermost level of all learning curves. Surprisingly, we find that there is no significant difference between all considered schemes within this paradigm.

In Chapter 7, we show that all ensemble learning systems, including StackingC, Grading (Seewald & Fürnkranz, 2001) and even Bagging (Breiman, 1996) can be simulated by Stacking. For this we give functionally equivalent definitions of most schemes as meta classifiers for Stacking.

In Chapter 8, we shortly introduce the field of Information Visualization and apply it to the problem of visualizing our twenty-six datasets and extensive experimental results. We find that the majority of examples are misclassified because none of the base classifiers predict correctly. Although Stacking would potentially be able to learn from such a setting, this is not observed. On the contrary, Stacking even predicts incorrectly when a majority of base classifiers predicts correctly. Full page figures for all datasets can be found in Appendix A.

Finally, in Chapter 9, we conclude this thesis with a summary of our main findings, an overall conclusion and an outlook on future research.

[2] We were even able to improve on this result with our variant StackingC, see Chapter 5.

Contents

Acknowledgements

This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture.

In no particular order, I'd like to thank Sašo Džeroski, Bernard Zenko and Ljupco Todorovski for one week of research collaboration, many helpful hints and discussions, inspiration for StackingC, valuable feedback on a preliminary version of this thesis and for introducing me to a lot of interesting people at recent conferences; Gerhard Widmer for letting me find a suitable research field on my own, for helping me write my first paper – and for employing me in the first place; Eduard Gröller for accepting to be my second examiner and for pointing me towards some interesting proceedings; Johann Petrak for helping me write my first paper, too, some feedback on later papers and a lot of technical support; Johannes (Juffi) Fürnkranz for helpful tips, valuable feedback and a lot of useful references; Elias Pampalk for valuable feedback on a preliminary version of Chapter 3; Simon Dixon for sharing with me his tricks on squashing papers in LaTeX – fortunately, this thesis does not need them; Arthur Flexer for pointing me towards Erich Mittenecker's book on statistical evaluation of experiments, which finally enlightened me as to how to implement all these statistical tests – hopefully properly; and especially Robert Trappl who manages our small research institute so well that even my many travels to various far-away conferences in nice places such as Key West, Cascais, Sydney, Helsinki and Maebashi could be easily supported.

Thanks also to my family, who supported me in so many ways that I will not list them out of fear of forgetting something; to all my friends for motivation, distraction and encouragement; and to myself for finally finding the willpower to push this through – driven by deadlines as usual.

Special thanks to Sonja Reitinger for tangible and intangible support – e.g. in the form of letting me use her notebook to finish this thesis in bed.

Chapter 1

Introduction

This chapter serves as an introduction to this thesis. We will begin by explaining the motivation behind this thesis, continue by introducing important concepts and terms based on an example of Stacking, and conclude with a detailed roadmap to the subsequent chapters of this thesis to facilitate quick access to interesting material.

1.1 Motivation

A variety of machine learning algorithms are available, e.g. decision tree learners such as C4.5 (Quinlan, 1993a), instance based learners such as IBk or KStar (Cleary & Trigg, 1995), simple learners based on conditional probabilities such as NaiveBayes and linear discriminants such as MLR (*multi-response linear regression*) – to name just a few. However, which one gives optimal or even acceptable results for a given dataset at hand is as of now a *black art*. *Meta-Learning* approaches (Brazdil, Gama & Henry, 1994; Pfahringer et al., 2000) aim to solve this problem by learning which classifier to choose from dataset characterization features and the performance of simple landmark classifiers with mixed success, but so far no reliable patterns have emerged. Some researchers rely on fine-tuning a single classifier which they presumably know best, while others try to decide this question empirically on a case-by-case basis.

The predominant approach to choose classifiers empirically is to estimate the accuracy of candidate algorithms on the problem, usually via crossvalidation[1], and select the one which seems to be most accurate. Schaffer (1993) has investigated this approach in a small study with three learning algorithms on five UCI datasets. His conclusions are that on the one hand this procedure is on average better than working with a single learning algorithm, but, on the other hand, the crossvalidation procedure often picks the wrong base algorithm on individual problems. This problem is expected to become more severe with an increasing number of classifiers.[2]

[1] Crossvalidation randomly splits the dataset into a fixed number of equal-sized parts, or folds. All but one fold is used for training and the remaining fold for testing each classifier. This procedure is repeated so that each fold is used for testing exactly once. The average accuracy over all test folds is the crossvalidation's estimate of the classifier's accuracy.

[2] In rank comparisons, see e.g. Table 3.1, we have found that selection by crossvalidation is usually the worst ensemble learning scheme – even with just four classifiers.

As crossvalidation essentially computes a prediction for each example in the training set, it was soon realized that this information could be used in more elaborate ways than simply counting the number of correct and incorrect predictions. One general way to achieve this is Stacking (Wolpert, 1992), which learns from predictions of base (=level-0) classifiers, via a single meta (=level-1) classifier. The basic idea of Stacking is to use the predictions of the base classifiers as attributes in a new training set that keeps the original class labels. This new meta training set is then used to train the meta classifier which learns to predict the final class. So for the additional cost of running an appropriate meta classifier it is possible to utilize all the output generated by a crossvalidation. Furthermore, the dimensionality of the meta dataset is equal to the number of classes multiplied by the number of base classifiers[3] and thus fairly independent of the dimensionality of the original dataset. The additional training cost for the meta classifier is usually much smaller than the training costs for the base classifiers, especially for large, high-dimensional datasets.

A straightforward extension proposed by (Ting & Witten, 1999) is using class probability distributions instead of predictions. This allows each base classifier to express uncertainty by returning estimated probabilities for all classes instead of just the one predicted class. Each prediction is thus replaced by a vector of class probabilities. In this case MLR (*multi-response linear regression*) is proposed as meta classifier. We follow their approach because all our base classifiers are equipped to output class probability distributions. In the second part of chapter 3 concerned with meta classifier choice, we will show that this extension is indeed competitive to Stacking with predictions. Furthermore, in Chapter 5 we will introduce a new variant derived from this extension which performs significantly better, learns faster and compensates for a previously unknown weakness in the mentioned extension.

We have chosen to investigate Stacking because it is the most general ensemble learning scheme available – in Chapter 7 we will demonstrate that it is able to simulate most ensemble learning schemes – and because its successful application is still a matter of *black art*. The latter will be comprehensively addressed in Chapter 3. As further motivation Stacking and especially our new variant StackingC seem to be the best performing ensemble learning schemes – see Table 3.1 for a comparison against other ensemble learning schemes including Bagging (Breiman, 1996) and AdaBoostM1 (Freund & Schapire, 1996). However, other methodologies for comparing classifiers find less or even no differences between related ensemble learning schemes, see Chapter 6.

1.2 How does Stacking work?

We will now explain Stacking in detail, based on a simple example. Figure 1.1 shows Stacking on a hypothetical dataset with three classes, n examples and N diverse base classifiers. Figure 1.1(a) shows the original dataset. Each example from the dataset consists of an attribute vector of fixed length, followed by a class value.

First, all base classifiers are evaluated via crossvalidation on the original dataset. Basically, a crossvalidation splits the dataset into J equal-sized folds, then uses $J-1$ folds for training and the remaining fold for testing. This

[3] For StackingC, see Chapter 5, it is just the number of base classifiers.

Attrs	*Cl.*
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
⋮	⋮
$AttrVec_n$	a

(a) original training set

a	b	c
0.90	0.05	0.05
0.15	*0.70*	0.15
0.10	*0.80*	0.10
0.20	0.20	*0.60*
⋮	⋮	⋮
0.80	0.10	0.10

(b) sample class probability distribution

$Classifier_1$			$Classifier_2$				$Classifier_N$			
a	b	c	a	b	c		a	b	c	*class*
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	...	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	a
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	...	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	b
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	...	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	b
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	...	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	c
	⋮			⋮				⋮		⋮
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	...	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	a

(c) training set for Stacking's meta classifier

Figure 1.1: Illustration of Stackingon a dataset with three classes (a, b and c), n examples and N base classifiers. $P_{i,jk}$ refers to the probability given by classifier i for class j on example number k

process is repeated J times so that each fold is used for testing exactly once, thus generating one prediction for every example.[4] One classifier's output is therefore a class probability distribution for every example.[5] Figure 1.1(b) shows how such a reasonable class probability distribution may look. The rows correspond to the examples from the original training set, see Figure 1.1(a).

The concatenated class probability distributions of all base classifiers in a fixed order, followed by the class value, forms the training set for Stacking's meta classifier, see Figure 1.1(c). After training the meta classifier, the base classifiers are retrained on the complete training data.[6]

For testing, the base classifiers are first queried for their class probability distributions on the test example. These form a meta-example for the meta classifier which outputs the final class prediction.

[4] $J = 10$ throughout this thesis.

[5] In the original proposal by Wolpert (1992), this would have been one predicted class value for each example. However, in this thesis we use mainly the extension by Ting & Witten (1999), unless otherwise noted.

[6] Interestingly, *not* retraining the base classifiers yields slightly worse results.

The choice of the following parameters for a Stacking classifier has initially been described as *black art* by Wolpert (1992). Ting & Witten (1999) have proposed settings for the meta classifier and the type of meta data to be used. In Chapter 3, we will revisit their results, run our own more extensive experiments which disagree on some of their points and additionally address the remaining issue of base classifier choice.

- One or more base classifiers, which may be arbitrary machine learning algorithms – in case of Stacking with probability distributions, they should be able to output class probability distributions.[7] The issue of appropriate choice of base classifiers seems to have not been investigated systematically previously.
- A single meta classifier. According to Wolpert (1992), relatively global and smooth classifiers should perform well. Ting & Witten (1999) propose MLR (*multi-response linear regression*). However, models such as decision trees have also been used successfully (e.g. (Skalak, 1997)), although they are clearly not smooth.
- The choice of meta data to use, i.e. the extension by Ting & Witten (1999) using probability distributions or the original proposal by Wolpert (1992) using only the predictions of base classifiers. Both approaches have their strengths and weaknesses.

More focussed and detailed introductions on Stacking can be found in the following chapters where appropriate. For the sake of comprehensibility, some redundancy is clearly inevitable.

1.3 Short Roadmap

In the following Chapter 2 we present papers from related research concerning ensemble learning schemes, structured along three different dimensions. Focussing on Stacking, we consider state of the art approaches and unique contributions to the field of ensemble learning schemes.

Afterwards, we address the issue which may have hampered the adoption of Stacking within the research community, namely the choice of parameters – base classifier, meta classifier choice and type of meta data to use. So, in Chapter 3, we propose default settings for Stacking which are empirically and theoretically grounded. Notice also that all base classifiers have been used with their default settings, so we propose a single set of settings for all experiments with Stacking. The mentioned chapter is an extended version of (Seewald, 2002c).

As the mechanism behind Stacking is largely unknown and no theoretical estimation of errors can be easily obtained – contrary e.g. to unweighted voting – trying to find such approximate estimations via statistical techniques and meta-learning seems appropriate. In Chapter 4, we investigate two approaches: learning Stacking's accuracy from dataset-related and base classifier related features, and predicting the best ensemble learning scheme for a given dataset using the same feature set. Meta-Learning in this sense is usually hard – nevertheless we found some surprisingly simple models and can even offer a tentative

[7] Class probability distributions may easily be constructed from a predicted class, but these are seldom of use.

explanation for one of them. This chapter is an extended version of (Seewald, 2002b).

During our experiments with Stacking, we found a previously unreported weakness, namely worse performance on multi-class datasets. In Chapter 5 we give empirical evidence for this weakness, trace it to its source and propose a new stacking variant, StackingC, which improves Stacking's performance while also resolving this weakness. We also discuss results on applying the same approach to related variants. This chapter is an extended version of (Seewald, 2002a).

Usually, classifiers are compared via accuracy, statistical tests such as t-Test or χ^2 test, or minimum description length. Stability of classifiers, in the sense of graceful performance degradation for smaller and smaller training sets, does not seem to have been studied systematically. So we chose to investigate the hypothesis that StackingC is the most stable ensemble learning scheme via learning curves. However, we were surprised to find no significant differences between related ensemble learning schemes. Chapter 6 presents our results, discussion and some tentative conclusions from this experiment.

In Chapter 7, as a step towards a theoretical framework for ensemble classification, we show that most ensemble learning schemes – Selection by Cross-validation (X-Val), Voting, StackingC, Grading (Seewald & Fürnkranz, 2001) and even Bagging (Breiman, 1996) – can be simulated by Stacking. To achieve this, we give functionally equivalent definitions of these schemes as meta classifiers for Stacking. On the way, we also show how Grading can be radically simplified without sacrificing its unique performance, thus making it amenable for such a simulation. Regrettably, sequential ensemble learning schemes such as AdaBoostM1 (Freund & Schapire, 1996) and Cascading (Gama & Brazdil, 2000) are not amenable to such a simulation for a variety of reasons.

In Chapter 8, we visualize our twenty-six datasets and some of our extensive experimental results by innovative methods from information visualization and point out some interesting patterns which may aid future studies. We find that the majority of examples are misclassified because none of the base classifiers predict correctly. Although Stacking would potentially be able to learn from such a setting, this is not observed. On the contrary, Stacking even predicts incorrectly when a majority of base classifiers is right. Thus we found the field of information visualization to be a valuable addition and inspiration for our research and are looking forward to applying its methods to real-life problems in the near future.

Appendix A contains full-page figures for all our datasets, taken from Chapter 8: Information Visualization. Appendix B contains a recent Curriculum Vitae for the author.

Chapter 9 concludes this thesis with a summary of our main findings, overall conclusions and outlook on future research.

Chapter 2

Related Research

There are many approaches to combine multiple models without resorting to ensemble learning schemes. Best known are simpler ensemble methods such as Bagging (Breiman, 1996) and AdaBoost (Freund & Schapire, 1996), which rely on training a set of diverse base classifiers (typically via different subsamples of the training set), whose predictions are then combined by more-or-less elaborate voting techniques, see e.g. (Bauer & Kohavi, 1999). Another group of techniques, *Meta-Learning*, focusses on predicting the right algorithm for a particular problem based on characteristics of the dataset (Brazdil, Gama & Henry, 1994) or based on the performance of other, simpler learning algorithms (Pfahringer et al., 2000). Finally, another common decision procedure (especially with larger datasets) is to take a subsample of the entire dataset and try each algorithm on this sample. This approach was analyzed by Petrak (2000).

However, for the purposes of this chapter, we will concern ourselves with research that is directly or indirectly concerned with Stacking, or other ensemble learning schemes which also employ diverse component classifiers. Research of this kind can roughly be grouped into three categories:

1. *Choosing parameters for Stacking*, i.e. base classifiers, meta classifier and type of meta data to use. As this provides *insight* into how Stacking works, we also have included other papers concerned with investigating aspects of its behaviour.

2. *Alternative Ensemble Learning Schemes.* An example is Selection by Crossvalidation (X-Val) which we mentioned earlier, and common unweighted voting. We chose to restrict ourselves to those schemes which incorporate more than a single learning algorithm bias and thus do not include Bagging (Breiman, 1996) and AdaBoost (Freund & Schapire, 1996) here.

3. *Extending non-ensemble learners*, e.g. C4.5, to incorporate more and different biases. In a way this is complementary to Stacking: whereas Stacking combines the output from its component classifiers without making any assumptions about their internal structure, the mentioned approach focusses on meshing classifiers to hopefully combine their strengths and mitigate their weaknesses. So we also consider these extended learners to be – somewhat distant – relatives to Stacking.

We will now consider each of these categories in turn, citing relevant papers and explaining their relation to our work where appropriate. To emphasize recent work, the papers appear in reverse chronological order.

2.1 Choosing Parameters & Providing Insight

Džeroski and Zenko (2002) investigate Stacking in the extension proposed by Ting & Witten (1999). They conclude that, when comparing against other ensemble learning schemes, Stacking with MLR as meta classifier is at best competitive to selection by crossvalidation (X-Val) and not significantly better as some papers claim. They propose a comparative study to resolve these and other contradictions in the literature. One possible explanation may be that, as Dietterich (1998) indicates, the widely used ten-times tenfold crossvalidated t-Test is fatally flawed, and a five-times two-fold crossvalidated t-Test should be employed instead.[1] Another explanation may be that Stacking is subtly preferred when compared via crossvalidation, since it also uses crossvalidation internally. Results from Chapter 6 hint that other approaches to compare Stacking may be less able to show its superiority, which is consistent with both hypotheses.

Seewald (2002a) investigates Stacking in the extension proposed by Ting & Witten (1999). He claims a weakness of this extension which is not apparent in the original version of Stacking and introduces a new variant, StackingC, in order to compensate for this weakness. Empirical evidence is given and supports these claims. An analysis into the reasons for improvement yields some interesting insights, most notably that the reason for this improvement is not mainly the dimensionality reduction of the meta dataset, but also the higher diversity of the class models. An extended version of this paper can be found in Chapter 5

Dietterich (2000a) is also concerned with classifier ensembles where the component classifiers are of the same type. It reviews Bagging, Randomization and Boosting. Surprisingly, Dietterich's definition of ensembles as given in (Dietterich, 2000b) does not to include Stacking. Thus, none of his papers are concerned with Stacking – a regrettable oversight.

Ting & Witten (1999) deal with the type of generalizer suitable to derive the higher-level model and the kind of attributes it should use as input. Their main conclusion is that using class probability distributions is superior to using predictions for the meta data, at least when MLR is the meta classifier. In Chapter 3 we were able to confirm their result that MLR is the best meta classifier for probability distribution meta data. But we also found that Stacking using prediction meta data is competitive to using probability distribution meta data for almost all meta classifiers – which disagrees with one of their main conclusions. This can be explained by taking a close look at their paper and finding that their definition of MLR differs from the common definition. In fact, they compare a variant similar to StackingC to Stacking with predictions, which biases their results strongly in favor of the former and thus in favor of Stacking with probability distribution meta data – which explains their incorrect conclusion. Using probability distributions instead of just predictions is essential to our variant StackingC, which will be introduced in Chapter 5, and other state-of-the-art

[1] In case of this alternative test, almost no significant differences are found between related ensemble learning schemes (personal communication by B.Zenko)

variants such as sMM5 (Džeroski & Zenko, 2002). Ting & Witten also investigated the usefulness of non-negativity constraints for feature weights within linear models, but found that these are inessential for best performance. However they found non-negativity constraints may be useful to facilitate human comprehension of these models. Since our focus is on performance and not on comprehensibility, we did not use non-negativity constraints.

Fan, Stolfo & Chan (1999) introduce a conflict-based accuracy estimate for Stacking and evaluate them on four datasets, two of them artificial. Stacking is used with one rule-based and two tree-based base classifiers and an unpruned decision tree as meta learner. While they claim that their measure is better than all other metrics previously proposed, it is not clear if their results will also hold on a larger number of datasets or with other meta learners. Furthermore, their approach relies on using predictions as meta data and would fail for class probability distributions.

Bauer & Kohavi (1999) compare voting classification algorithms. While they use Bias-Variance analysis to achieve insights about the sources of error reduction, their analysis only investigates Bagging, Boosting and other ensemble methods combining the same base algorithm, while our work is concerned with Stacking which combines diverse base algorithms. Furthermore, they use the zero-one loss decomposition by Kohavi & Wolpert (1996) which has been severely critized.[2]

Skalak (1997) includes an excellent overview about methods for constructing classifier ensembles. His other main contribution consists of investigating ensembles of coarse instance-based classifiers storing only a few prototypes per class, using ID3 as meta classifier. His results inspired us to re-evaluate meta classifier choice for Stacking, see Chapter 3.

Ting (1997) proposes to use the predictions of base classifiers for learning a function that maps the algorithms' internal confidence measure (e.g., instance typicality for nearest neighbor classifiers or a posteriori probabilities for Naive Bayes classifiers) to an estimate of its accuracy on the output which could then potentially be used to combine their expertise. If the classifiers output class probability distributions which are strongly related to their internal confidence measures – as would be expected –his approach is similar to Grading, investigated by Seewald & Fürnkranz (2001); however, in his case a definitive learning algorithm is neither given nor evaluated.

Brodley & Lane (1996) illustrate empirically that integration approaches such as Stacking seem unable to exploit the diversity as measured by classification overlap of the classifiers. They also describe a metric for choosing base classifiers in order to maximize coverage. Diversity was achieved by different random orderings and initializations of linear machines while Stacking achieves diversity by exploiting machine learning bias of diverse base classifiers, so their results are not directly applicable to our work.

2.2 Alternative Ensemble Learning Schemes

Džeroski and Zenko (2002) investigate Stacking in the extension proposed by Ting & Witten (1999). They introduce a new variant sMM5 which they claim to

[2] Their case for a definitive zero-one loss function is much weaker than it appears (multiple personal communications) Unfortunately we were unable to find a citation which states this.

be *in a league of its own.* Their new variant is quite competitive to our variant StackingC (Chapter 5) but much slower, according to unpublished experiments.

Seewald & Fürnkranz (2001) re-evaluated a scheme called Grading that associates a meta-level classifier to each base classifier. Essentially, Grading trains a referee for each base classifier which aims to predict when its base classifier fails, on an example-by-example basis. This decision is based on the original dataset's attributes. A weighted voting of the base classifiers' predictions gives the final class prediction. The voting weight is the confidence for a correct prediction of a base classifier, which is estimated by its associated meta classifier. In Chapter 7, we will argue that Grading is essentially similar to accuracy-weighted voting, which explains the observation that very different meta-level classifiers still perform equally well.

Another approach with a similar goal, *meta decision trees* by (Todorovski & Džeroski, 2000), aims at directly predicting which classifier is best to classify an individual example. To this end, it uses information about statistical properties of the predicted class distribution as attributes and predicts the right algorithm from this information. The approach is not modular (in the sense that any algorithm could be used at the meta-level) but implemented as a modification to the decision tree learner C4.5. While their approach may make the combining scheme more comprehensible by learning an explicit decision tree decision tree, it is unclear whether this leads to better insight as well. Stacking and StackingC using MLR as meta-learner also allow to determine relative importance of base learners per class, simply by inspecting the weights of meta-level attributes after training – this has e.g. been done by Ting & Witten (1999).

Cascading by Gama & Brazdil (2000) is a related variant to Stacking where the classifiers are applied in sequence and there is no dedicated meta classifier. Each base classifier, when applied to the data, adds its class probability distribution to the data and returns an augmented dataset, which is to be used by the next base classifier. Thus, the order in which the classifiers are executed becomes important. Cascading does not use an internal crossvalidation like most other ensemble learning schemes and is therefore claimed to be at least three times faster than Stacking. On the other hand in Stacking the classifier order is not important, thereby reducing the degrees of freedom and minimizing chances for overfitting. Furthermore, cascading increases the dimensionality of the dataset with each step whereas Stacking's meta dataset has a dimensionality which is independent of the dimensionality of the dataset – i.e. the number of base classifiers multiplied with the number of classes.

Merz (1999) studies the use of correspondence analysis and lazy learning to combine classifiers in a stacking-like setting. He compares his approach, SCANN, to two stacking variants with NaiveBayes resp. a backpropagation-trained neural network as meta-learner. MLR was not considered as meta-learner. According to experiments with synthetic data, his approach is equivalent to plurality vote if the models make uncorrelated errors. However, in practice this is seldom the case. Moreover, his approach is limited to using predictions as meta-level data and would fail for the class probability distributions which we use. Still, correspondence analysis as a means of less static dimensionality reduction of stacking meta-level data may have its merits.

Chan & Stolfo (1995) propose the use of *arbiters* and *combiners.* A combiner is more or less identical to Stacking. Chan & Stolfo (1995) also investigate a related form, which they call an *attribute-combiner.* In this architecture, the original attributes are not replaced with the class predictions, but instead they are added to them. As Schaffer (1994) shows in his paper about bi-level stacking, this may result in worse performance.

An arbiter (Chan & Stolfo, 1995) is a separate, single classifier, which is trained on a subset of the original data. This subset consists of examples on which the base classifiers disagree. They also investigate *arbiter trees*, in which arbiters that specialize in resolving conflicts between pairs of classifiers are organized in a binary decision tree. Arbiters use information about the disagreement of classifiers for selecting a training set.

2.3 Extending non-ensemble learners

Combinations of decision tree with other learning algorithms have been studied in various ways before. An early example of a hybrid decision tree algorithm is presented by Utgoff (1988). Here, a decision tree learner is introduced that uses linear threshold units at the leaf nodes; however, pruning is not considered as the algorithm was expected to work only on noise-free domains.

In (Seewald et al., 2001), a decision tree learner is described which replaces subtrees in post-pruning not only with leafs, but also with alternative leaf classifiers. Four common classifiers are considered. To be able to evaluate different leaf classifiers, a separate pruning set was used, similar to Reduced Error Pruning (Quinlan, 1987). The results are somewhat promising.

In (Gama & Brazdil, 1999) and (Gama, 1999) a decision tree learner named LTree is introduced that computes new attributes as linear, quadratic or logistic discriminant functions of attributes at each node; these are then also passed down the tree. The leaf nodes are still basically majority classifiers, although the class probability distributions on the path from the root are taken into account via smoothing techniques. Results indicate that this offers significant improvements, comparable to that by Stacking, while retaining some comprehensibility.

Kohavi (1996) introduces a decision tree learner named NBTree which has Naive Bayes classifiers as leaf nodes and uses a split criterion that is based directly on the performance of Naive Bayes classifiers in all first-level child nodes (evaluated by crossvalidation) — an extremely expensive procedure. The tree size is determined by a simple stopping criterion and no postpruning is done. The results are mildly positive; in most cases, NBTree outperforms one, in a few cases both of its constituent base learners. In (Kohavi, 1996) it is also pointed out that the size of the trees induced by NBTree is often substantially smaller than the original C4.5 trees. This is not of obvious practical significance: The hybrid trees may be smaller, but that does not necessarily make them more comprehensible to the user, due to the more complex models at the leaves.

A recursive bayesian classifier is introduced by Langley (1993). The main idea is to split the data recursively into partitions where the conditional independence assumption holds. The experimental results are somewhat disappointing – the author managed to show superiority of his method over simple Naive Bayes only on synthetic (and noise-free) data specifically generated for his experiments. Pre- and postpruning are also not implemented.

2.4 Conclusion

While we saw there is a lot of research indirectly concerned with ensemble learning schemes, only little research is directly related to investigating Stacking, the most general such scheme. A stronger research focus – outside the scope of this thesis – seems to be on simpler ensemble learning schemes such as Bagging (Breiman, 1996) and AdaBoost (Freund & Schapire, 1996), which combine only a single kind of classifier. While we believe that the current variants StackingC and sMM5 are very close to the optimum[3], we hope that our comprehensive overview on related research stimulates further work in ensemble learning and applying ensembles in other research areas such as Game Playing, Self-Diagnosing Systems and BioInformatics.

[3] In fact several experiments concerned with improving performance further, e.g. by combining both current variants, amply demonstrated the law of diminishing returns.

Chapter 3

Exploring the Parameter State Space

Ensemble learning schemes are a new field in data mining. While current research concentrates mainly on improving the performance of single learning algorithms, an alternative is to combine learners with different biases. Stacking is an ensemble learning scheme which tries to combine learners' predictions or confidences via another learning algorithm. However, the adoption of Stacking into the data mining community is hampered by its large parameter space, consisting mainly of other learning algorithms: (1) the set of learning algorithms to combine, (2) the meta-learner responsible for the combining and (3) the type of meta data to use: confidences or predictions. None of these parameters are obvious choices. Furthermore, little is known about the relation between parameter settings and performance of Stacking. By exploring all of Stacking's parameter settings and their interdependencies, we intend to make Stacking a suitable choice for mainstream data mining applications. This chapter is based on the paper (Seewald, 2002c).

3.1 Introduction

The basic idea of the ensemble learning scheme Stacking is to use the predictions of base classifiers as attributes in a new training set that keeps the original class labels. Stacking thus utilizes a *meta* classifier to combine the predictions from several *base* classifiers which were estimated via crossvalidation on the training data. Any classifier may be used as base and/or meta classifier. We shall refer to the type of meta data consisting of base classifiers predictions as *preds*.

A straightforward extension of this approach is using class probability distributions of the original classifiers[1] which convey not only prediction information, but also confidence for all classes. We shall call the meta data of this extension *class-probs*. This approach was evaluated and found to be superior to Stacking with predictions in (Ting & Witten, 1999), provided *multi-response linear regression* (MLR) is used as meta classifier.

[1] Every prediction is replaced by a vector of probabilities, consisting of one probability value for each class.

Table 3.1: This table shows a ranking of all ensemble learning schemes. Ranks are given as significant Wins/Losses with the wins counting for the algorithm in the row (=BetterScheme). Significant differences were determined via a χ^2 test after McNemar with 95% significance level, based on data from a single tenfold crossvalidation. All schemes use the set of four diverse base classifiers from section 3.3; except Bagging and AdaBoostM1 which both use C4.5 with twenty-five bags resp. iterations.

BetterScheme	X-Val	Grading	StackingC	Voting	AdaBoostM1	Bagging
X-Val	0/0	3/3	1/3	4/3		
Grading	3/3	0/0	3/4	0/2		
StackingC	3/1	4/3	0/0	5/2	6/0	5/1
Voting	3/4	2/0	2/5	0/0		

While approaches such as boosting (Freund & Schapire, 1996) and bagging (Breiman, 1996), which combine classifiers of the same type, have been used extensively in data mining, Stacking has not.

As short motivation, consider that in terms of significant performance differences on our datasets, the most recent scheme StackingC[2] wins five times and loses only once against AdaBoostM1 and wins six times and never loses against Bagging. More results can be found in Table 3.1. StackingC performs better than its competitors, even though much less research has been focused on improving Stacking! Why then has Stacking not been adopted more widely? Tentatively we can suggest some reasons: For once, Stacking requires an integrated workbench including common machine learning classifiers. As of the time of writing this chapter, two of the largest commercially available data mining tools lack a basic machine learning classifier, NaiveBayes. The added complexity of managing ensemble schemes may also play a role. But most significantly, Stacking requires a lot of non-obvious parameters: which base classifiers to choose, which meta classifier to choose and also the type of meta data – either predictions as in the original proposal by Wolpert (1992) or complete probability distributions as in the extension proposed by Ting & Witten (1999).

Thus we felt it was time to investigate parameter settings for Stacking systematically to see how various parameters contribute to Stacking's performance, in order to see where sensible areas for further improvement may lie, but also to give useful proposals for *all* parameter settings. We investigate both the original Stacking (Wolpert, 1992) and the extension (Ting & Witten, 1999) here.

3.2 Overview

At first, Section 3.3 will present the experimental setup, the datasets and classifiers considered and details on significance tests and alpha-confidence levels.

In Section 3.4 we address the choice of base classifiers, using MLR as meta classifier as proposed by (Ting & Witten, 1999). We show that it is quite hard to significantly beat the trivial choice of using all available base classifiers and

[2] i.e. Stacking with a fixed specialized meta classifier based on MLR, see Chapter 5.

even harder to beat an informed choice of four base classifiers chosen via a priori and a posteriori arguments on diversity and base classifier performance. This makes base classifier choice the least influential factor on Stacking's performance. Intuitively, we would also expect that the *know-how* to combine the output of classifiers is more important than *which* classifiers to combine, as long as a reasonably large and diverse set is chosen.

In Section 3.5 we address both meta classifier choice and type of meta data to be used, on two subsets of base classifiers. We show that MLR is indeed the best classifier for *preds* meta data, among those we considered. We notice that the performance differerences of those variants using *preds* meta data are much smaller than of the variants for *class-probs* meta data, which indicates that the learning problem for *preds* is easier for most classifiers. NaiveBayes seems a reasonable if somewhat arbitrary choice for *preds* meta data. At last we conclude that Stacking with predictions meta data is competitive to using probability distribution meta data. We point out that Ting & Witten (1999) may have used a variation of MLR similar in spirit to StackingC in their experiments which yields a biased comparison and may explain why their conclusion as to the merits of the different meta data types differs from ours.

In Section 3.6, Related Research, we give a short overview on relevant research. We will now proceed to shortly characterize our experimental setup.

3.3 Experimental Setup

For our empirical evaluation we chose twenty-six datasets from the UCI Machine Learning Repository (Blake & Merz, 1998), shown in Table 3.2. These datasets include fourteen multi-class and twelve two-class problems. Reported accuracy estimates are from a single ten-fold stratified crossvalidation. Significant differences were evaluated by a χ^2-test after McNemar[3] with significance level of 95%, unless otherwise noted.

As base classifiers for Stacking we considered the following seven base learners, which were chosen to cover a variety of different biases. For figures, classifier numbers are used instead of proper names.

1. J48: a Java port of C4.5 Release 8 (Quinlan, 1993a)

2. KStar: the K* instance-based learner (Cleary & Trigg, 1995)

3. MLR: a multi-class learner which tries to separate each class from all other classes by linear regression (*multi-response linear regression*)

4. NaiveBayes: the Naive Bayes classifier using kernel density estimation over multiple values for continuous attributes, instead of assuming a simple normal distribution.

5. DecisionTable: a decision table learner.

6. IBk: the IBk instance-based learner with $K = 1$ nearest neighbors, in order to offset KStar with a maximally local learner.

7. KernelDensity: a simple kernel density classifier.

[3] Dietterich (1998) proposes this test when the investigated algorithm can be run only once.

Table 3.2: The used datasets with number of classes and examples, discrete and continuous attributes, baseline accuracy (%) and entropy in bits per example (Kononenko & Bratko, 1991).

Dataset	cl	Inst	disc	cont	bL	E
audiology	24	226	69	0	25.22	3.51
autos	7	205	10	16	32.68	2.29
balance-scale	3	625	0	4	45.76	1.32
breast-cancer	2	286	10	0	70.28	0.88
breast-w	2	699	0	9	65.52	0.93
colic	2	368	16	7	63.04	0.95
credit-a	2	690	9	6	55.51	0.99
credit-g	2	1000	13	7	70.00	0.88
diabetes	2	768	0	8	65.10	0.93
glass	7	214	0	9	35.51	2.19
heart-c	5	303	7	6	54.46	1.01
heart-h	5	294	7	6	63.95	0.96
heart-statlog	2	270	0	13	55.56	0.99
hepatitis	2	155	13	6	79.35	0.74
ionosphere	2	351	0	34	64.10	0.94
iris	3	150	0	4	33.33	1.58
labor	2	57	8	8	64.91	0.94
lymph	4	148	15	3	54.73	1.24
primary-t.	22	339	17	0	24.78	3.68
segment	7	2310	0	19	14.29	2.81
sonar	2	208	0	60	53.37	1.00
soybean	19	683	35	0	13.47	3.84
vehicle	4	846	0	18	25.41	2.00
vote	2	435	16	0	61.38	0.96
vowel	11	990	3	10	9.09	3.46
zoo	7	101	16	2	40.59	2.41

All algorithms are implemented in WEKA Release 3.1.8. Each of them returns a class probability distribution, i.e. they do not predict a single class, but give probability estimates for each possible class. Parameters for learning schemes which have not been mentioned were left at their default values.

These algorithms can be clustered into four natural groups[4] by their internal structure, where the first member tends to give better results than the others.

- J48, DecisionTable
- NaiveBayes
- MLR
- KStar, IBk, KernelDensity

Surprisingly – as we found out during our experiments – this can also be supported empirically. In particular, the statistical correlation of accuracies within

[4] A group with a single member is still considered a group by us, although in that case inner-group correlation and diversity are trivial to compute. However, intra-group measures are still reasonable.

each group is always greater than 0.95 while it is much smaller between classifiers from different groups. Thus, the structure of correlations allows us to determine these same groups empirically.[5] So we considered not only the trivial set of all seven base classifiers, but also the set of four base classifiers J48, NaiveBayes, MLR and KStar by choosing from each correlated subgroup the classifier which performed best by geometric mean of accuracy ratio.

We define a variant as a specific Stacking algorithm of which all parameters – type of meta data, meta classifier, and the set of base classifiers – are known. Meta data *class-probs* is signified by the prefix *St*, *preds* is signified by the prefix *StP*. After this prefix, *7B* refers to the full set of base classifiers while *4B* refers to the set of four diverse base classifiers mentioned in Section 3.3. After this, a hyphen precedes the meta classifier's name or an abbreviation. E.g. *St7B-MLR* refers to Stacking with the full set of seven base classifiers, MLR as meta classifier and *class-probs* as type of meta data while *StP7B-MLR* refers to the same variant with *preds* as meta data. In Section 3.5, we will define stacking groups as those variants which just differ in the meta classifier. These are named without reference to the meta classifier.

3.4 Base Classifier Choice

In this section we investigate Stacking variants with MLR as meta classifier[6] and any non-empty subset of our seven base classifiers as base classifiers[7]. Because of the large number of comparisons, we used a significance level of 99% here to reduce our alpha-error. If we consider using all seven base classifiers (*St7B-MLR*) as gold standard, about one tenth of our variants are significantly better than *St7B-MLR* on any dataset; the rest shows no significant differences or are frequently even significantly worse. Figure 3.1 shows all our variants as accuracy ratios vs. *St7B-MLR*. One variant corresponds to a specific instantiation of Stacking with a non-empty subset from our seven base classifiers, always with MLR as meta classifier, on *class-probs* meta data. We have grouped the variants according to the size of the subset of base classifiers, from one to six.

Although at least some variants seem to offer considerable improvements on accuracy, this by no means assures a significant difference. We have considered two approaches to take significance into account.

At first we determined those variants which, over all datasets, never lose significantly against *St7B-MLR* and win as often as possible. It turns out that the maximum number of wins is only three, i.e. 11.5% of our datasets, which is somewhat disappointing. If we compare against *St4B-MLR*[8] instead, the maximum number of wins falls to zero. Concluding, dataset-independent variant resp. base classifier choice is not able to improve upon *St4B-MLR* here, even by hindsight.

[5] A preliminary analysis of the κ-statistic – a measure of diversity due to (Dietterich, 2000b) – is also compatible with this finding: the inner-group diversity between classifiers from the same group is generally smaller than the diversity between classifiers from different groups.

[6] ...in the extension proposed by (Ting & Witten, 1999): using complete class probability distributions as meta-level data.

[7] 128 variants per dataset.

[8] Stacking with the set of four diverse base classifiers we mentioned earlier. As we will see, this variant is slightly better than *St7B-MLR*.

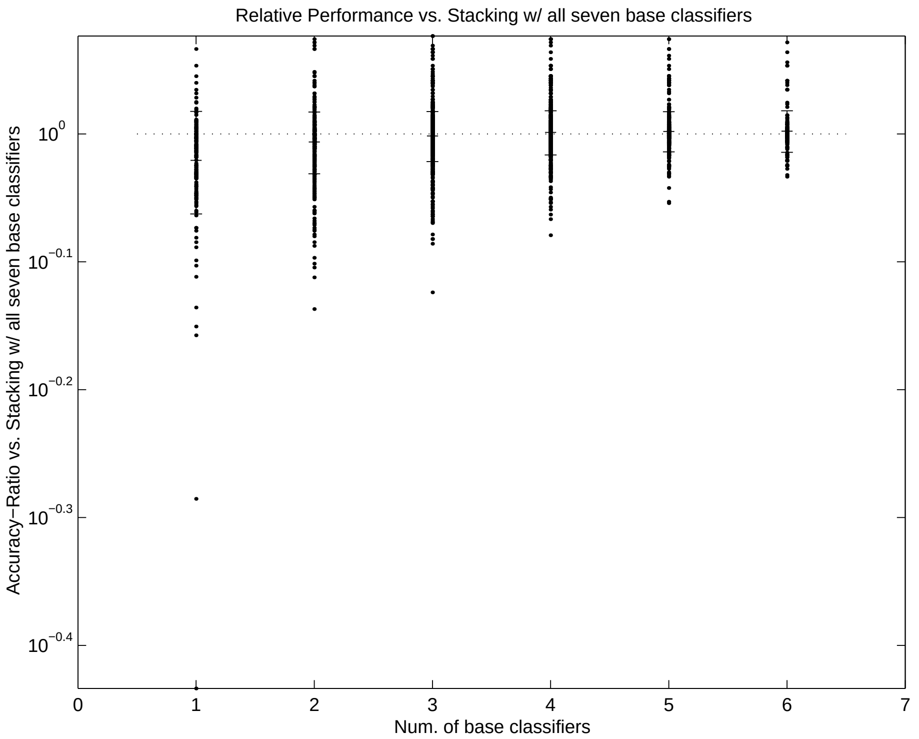


Figure 3.1: This figure shows the improvement of Stacking variants with different numbers of base classifiers and MLR as meta classifier, where all possible subsets of base classifiers were considered, as $\frac{Acc_{StVariant}}{Acc_{St7B-MLR}}$. The dotted line indicates a ratio of 1.0. Each • shows the ratio from one dataset and variant. Average and standard deviation over all subsets with the same number of base classifiers are shown as error bars.

If we were to consider a dataset-dependent choice of variants resp. base classifiers, i.e. possibly choosing a different variant for each dataset, the results are still quite disappointing. While the maximum number of wins is 7 (26.9%) vs. *St7B-MLR*, it is only 2 (7.7%) vs. *St4B-MLR*.[9] Thus, even if we would resort to Meta-Learning and found a model which lets us choose variants as good as by hindsight – which is doubtful to say the least – the improvements would still be quite insignificant.

So, in the remainder of this chapter, we just consider two sets of base classifiers: the trivial choice of using all seven (those used by *St7B-MLR*), but also the subset of four diverse ones (those used by *St4B-MLR*) we mentioned in the last section. This is motivated by the wish to investigate both meta classifier choice and base classifier choice, even if quite coarse-grained, in one setup.

[9] The maximum number of wins is even just 1 (3.9%) for StackingC with the full set of seven base classifiers.

Table 3.3: This table shows the significant wins minus losses for each variant, against the other three meta classifiers.

Variant	J48	KStar	MLR	NB
St7B	2	-26	16	8
St4B	-2	-19	19	2
StP7B	-1	0	1	0
StP4B	2	-9	1	6

Table 3.4: This table shows the improvements of Figures 3.2, 3.3, 3.4 and 3.5, in that order, in terms of significant wins and losses for the first mentioned variant.

Comparison	J48	KStar	MLR	NB
St7B vs. *StP7B*	1/5	1/14	4/3	1/4
St4B vs. *StP4B*	1/4	1/11	2/0	1/4
St4B vs. *St7B*	2/2	3/2	4/0	1/2
StP4B vs. *StP7B*	2/0	1/3	2/1	3/4

3.5 Meta Classifier Choice

In this section, we investigate Stacking variants along three independent dimensions – which are indeed *all* possible dimensions for Stacking's parameters.

1. Different meta classifiers, chosen from our set of four diverse classifiers from Section 3.3 (numbers 1-4).

2. Type of meta data: either predictions = *preds* or complete class probability distributions = *class-probs*.

3. Base Classifiers: the set of four resp. seven base classifiers from Section 3.3, as maximally coarse-grained base classifier choice.

What concerns us most of all are the first two dimensions, i.e. to determine which meta classifier is best, depending on type of meta data; and whether or not one type of meta data can be considered unconditionally superior. However, the third dimension can still roughly tell us the susceptibility of each meta classifier to changes in its set of base classifiers and thus give us the opportunity to also investigate coarse-grained base classifier choice at little additional cost.

The second and third dimension gives the name to our Stacking group – i.e. *St4B*, *St7B* for *class-probs* meta data[10] and *StP4B*, *StP7B* for *preds* meta data. The meta classifier is usually shown as an additional dimension in tables or figures, so all parameters are hereby accounted for.

To determine which meta classifier is best, we resorted to a standard ranking of all meta classifiers, separately for each stacking group. Table 3.3 shows *wins* minus *losses* of each meta classifier versus all other meta classifiers, for each group separately.

[10] which we consider to be the default case

A short glance at the table reveals some insights: For *class-probs* meta data MLR is clearly the best meta classifier, which was also shown by Ting & Witten (1999). However, for *preds* meta data *StP7B* and *StP4B* disagree – the latter considers NaiveBayes superior while the former prefers MLR; with NaiveBayes and KStar both on a very close second place. The wins and losses are smaller than for *St7B* and *St4B* which indicates that meta classifier choice seems to make less difference for predictions meta data. This is to be expected since most classifiers are best suited to process nominal data, thus their performance as meta classifiers is better and – since there is an upper limit on accuracy – tends to be more similar than for continuous data.

If we had to choose any one classifier for prediction meta data, NaiveBayes seems the logical choice – it is not inconceivable that NaiveBayes ended up on second place because the ranking may be a little noisy; after all the difference is only one. A priori, the Bayesian approach inherent in NaiveBayes seems a suitable way to combine confidences from the base classifiers, but in practice this seems to work well only for *preds* meta data. It may be that the modelling of continuous attributes by multiple kernels is not appropriate for this task and that a simple normal distribution may be more useful. This would explain why in our case, NaiveBayes performs clearly better with *preds* – as does J48 – and not equally well on *class-probs* and *preds*, as Ting & Witten (1999) found. Interestingly, both KStar and IBk do not perform better on *class-probs* meta data – rather, they are those meta classifiers with the highest number of significant losses on *class-probs* vs. *preds*. This also contradicts Ting & Witten (1999), who concluded that IB1 performs better on *class-probs*. Further work is needed to resolve these contradictory results.

Now we will investigate whether one type of meta data can be considered superior. For this, we determined accuracy ratios of *St7B* vs. *StP7B* and *St4B* vs. *StP4B* respectively, see Figure 3.2 and 3.3. The ranking in Table 3.4 shows it more clearly: While MLR performs better on *class-probs* meta data, all other considered meta classifiers perform better on *preds*, i.e. nominal, meta data. The latter is to be expected since most machine learning algorithms are best suited to process nominal data.

This seems to contradict Ting & Witten (1999) who concluded that no meta classifier on prediction meta data offered satisfactory performance. However, their definition of MLR is different: each linear model had as input data only those partial class probability distributions concerned with the class it was trying to predict. So the dimensionality of the input data was smaller for MLR by a factor equal to the number of classes, which makes the mentioned comparison somewhat biased. If we try to replicate their results and replace MLR with their modified version[11], it is easily the best meta classifier with a large margin – which explains their early dismissal of Stacking with *preds* meta data. Concluding, Stacking with probability distributions (*class-probs*) is competitive to classic Stacking with predictions (*preds*). However, some Stacking variants based on *class-probs* perform significantly better than both, as shown in Chapter 5. That is not to say that Stacking with predictions cannot be improved in the future – it remains an interesting research topic and should not yet be dismissed.

[11] i.e. equivalent to StackingC.

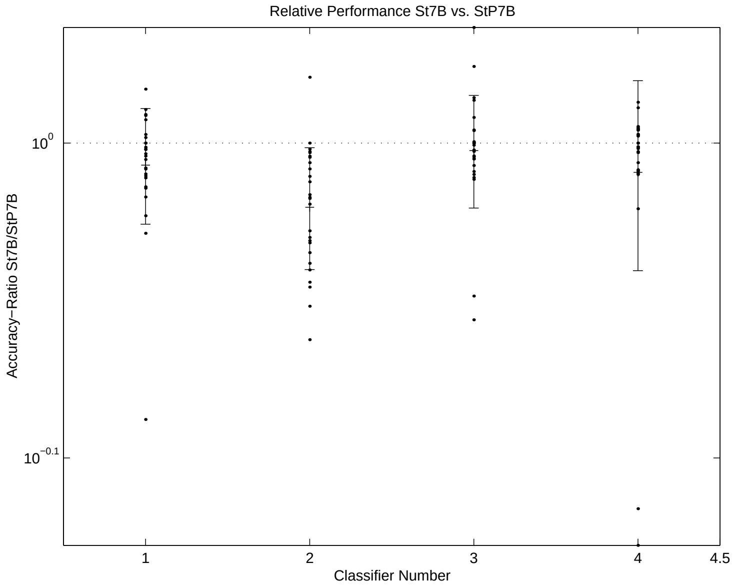


Figure 3.2: Improvement of *St7B* over *StP7B*, i.e. using probability distributions meta data vs. prediction meta data, as $\frac{Acc_{St7B}}{Acc_{StP7B}}$. The dotted line indicates a ratio of 1.0. Each • shows the ratio from one dataset. Average and standard deviation over all datasets are shown as error bars.

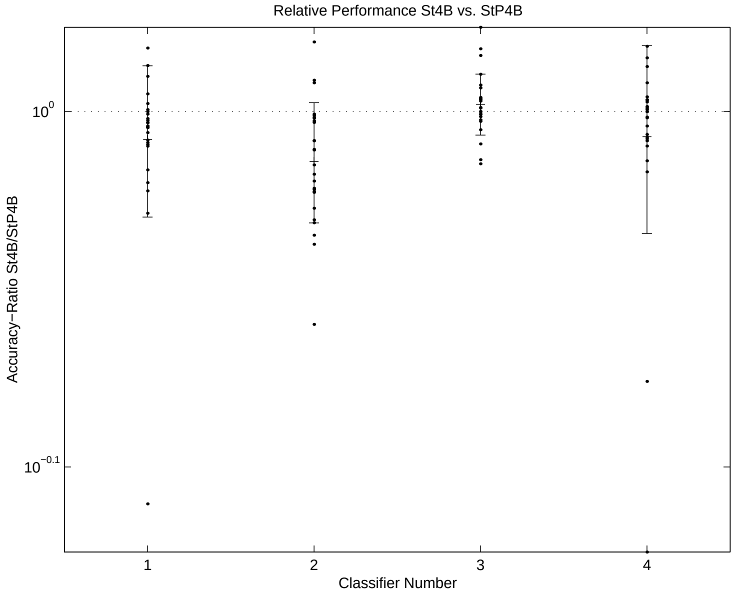


Figure 3.3: Improvement of *St4B* over *StP4B*, i.e. using probability distributions meta data vs. prediction meta data, as $\frac{Acc_{St4B}}{Acc_{StP4B}}$. The dotted line indicates a ratio of 1.0. Each • shows the ratio from one dataset. Average and standard deviation over all datasets are shown as error bars.

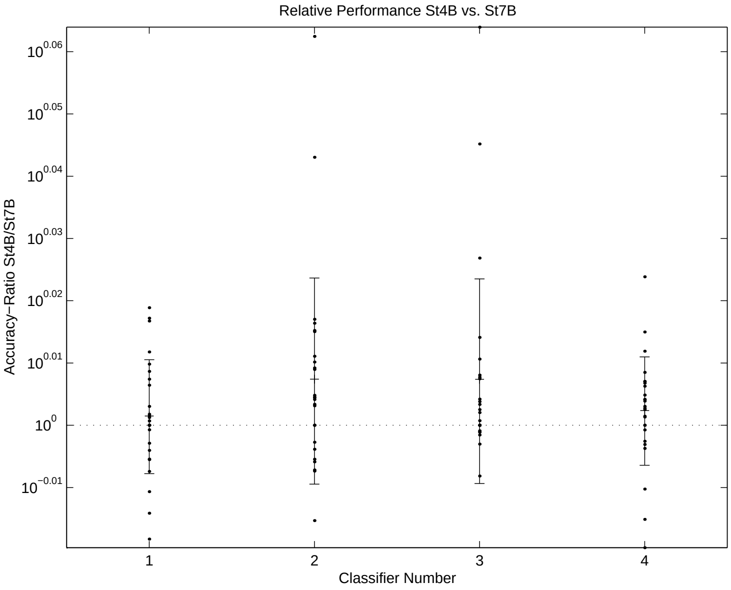


Figure 3.4: Improvement of *St4B* over *St7B*, i.e. using only four base learners vs. using all seven for probability distributions meta data, as $\frac{Acc_{St4B}}{Acc_{St7B}}$. The dotted line indicates a ratio of 1.0. Each • shows the ratio from one dataset. Average and standard deviation over all datasets are shown as error bars.

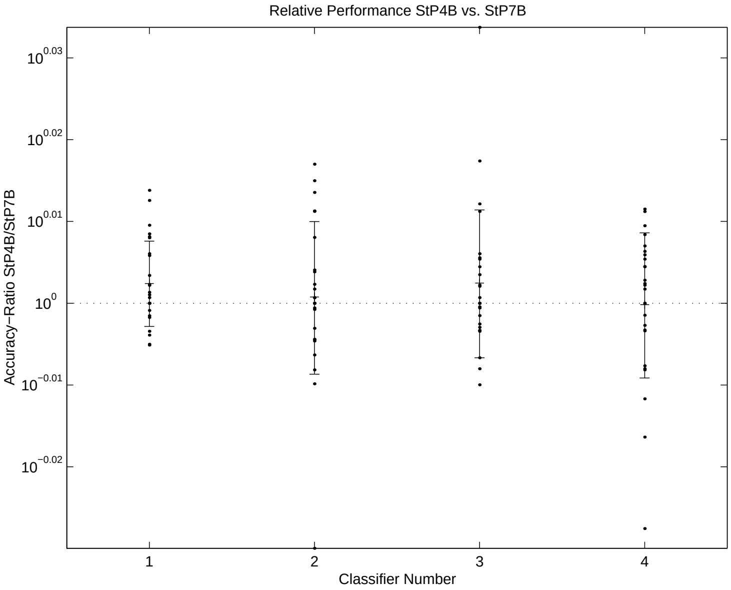


Figure 3.5: Improvement of *StP4B* over *StP7B*, i.e. using only four base learners vs. using all seven for predictions meta data, as $\frac{Acc_{StP4B}}{Acc_{StP7B}}$. The dotted line indicates a ratio of 1.0. Each • shows the ratio from one dataset. Average and standard deviation over all datasets are shown as error bars.

Now we were interested in the difference between using four base and all our seven base classifiers, see Figure 3.4 and 3.5. In the former case, we see that *St4B* is usually better and in four cases even significantly better when using MLR as meta classifier. In the latter case we see the same picture, but even less difference.[12] So we tenatively conclude that on average Stacking does seem to work better with a smaller set of less similar base classifiers, especially when using continuous meta-level data.

3.6 Related Research

Seewald (2002a) investigates Stacking in the extension proposed by Ting & Witten (1999). He claims a weakness of this extension which is not apparent in the original version of Stacking and introduces a new variant, StackingC, in order to compensate for this weakness. Empirical evidence is given and supports these claims. An analysis into the reasons for improvement yields some interesting insights, most notably that the reason for this improvement is not mainly the dimensionality reduction of the meta dataset, but also the higher diversity of the class models.

Džeroski and Zenko (2002) investigate Stacking in the extension proposed by Ting & Witten (1999). They conclude that, when comparing against other ensemble learning schemes, Stacking with MLR as meta classifier is at best competitive to selection by crossvalidation (X-Val) and not significantly better as some papers claim, while their new variant sMM5 clearly beats X-Val. They propose a comparative study to resolve these contradictions in the literature.

Seewald & Fürnkranz (2001) propose a scheme called Grading that learns a meta-level classifier for each base classifier. Grading trains a meta classifier for each base classifier which tries to predict when its base classifier fails. This decision is based on the dataset's attributes. A weighted voting of the base classifiers' prediction gives the final class prediction. The voting weight is the confidence for a correct prediction of a base classifier, which is estimated by its associated meta classifier.

Cascading by Gama & Brazdil (2000) is a related variant to Stacking where the classifiers are applied in sequence and there is no dedicated level 1 classifier. Each base classifier, when applied to the data, adds his class probability distribution to the data and returns an augmented dataset, which is to be used by the next base classifier. Thus, the order in which the classifiers are executed becomes important. Cascading does not use an internal crossvalidation like most other ensemble learning schemes and is therefore claimed to be at least three times faster than Stacking. On the other hand in Stacking the classifier order is not important, thereby reducing the degrees of freedom and minimizing chances for overfitting. Furthermore, cascading increases the dimensionality of the meta dataset with each step whereas Stacking's meta dataset has a dimensionality which is independent of the dimensionality of the dataset, i.e. the number of base classifiers multiplied with the number of classes.

Todorovski & Džeroski (2000) introduce a novel method to combine multiple models. Instead of directly predicting the final class as all combining schemes

[12] We noted that for StackingC, the difference is even less, i.e. 1.0012±0.0117 for the accuracy ratio of 4B vs. 7B – making it the scheme least susceptible to our coarse base classifier choice.

we considered, their meta-learner MDT, based on C4.5, specifies which model to use for each example based on statistical and information theoretic measures computed from the class probability distribution. While their approach may make the combining scheme more comprehensible by learning an explicit decision tree decision tree, it is unclear whether this leads to better insight as well.

Merz (1999) studies the use of correspondence analysis and lazy learning to combine classifiers in a stacking-like setting. He compares his approach, SCANN, to two Stacking-variants with NaiveBayes resp. a backpropagation-trained neural network as meta-learner. MLR was not considered as meta-learner. According to experiments with synthetic data, his approach is equivalent to plurality vote if the models make uncorrelated errors. However, in practice this is seldom the case. Moreover, his approach is limited to using predictions as meta-level data and would fail for the class probability distributions which we use.

Ting & Witten (1999) deal with the type of generalizer suitable to derive the higher-level model and the kind of attributes it should use as input. Comments and remarks concerning their paper can be found throughout this chapter. Additionally, they investigated the usefulness of non-negativity constraints for feature weights within linear models, but found that it is not essential to get the best performance. However they found this may be useful to facilitate human comprehension of these models. Since our focus was on performance and not on comprehensibility, we did not use a non-negativity constraint.

Skalak (1997) includes an excellent overview about methods for constructing classifier ensembles. His other main contribution consists of investigating ensembles of coarse instance-based classifiers storing only a few prototypes per class.

Chan & Stolfo (1995) propose the use of *arbiters* and *combiners*. A combiner is more or less identical to Stacking. Chan & Stolfo (1995) also investigate a related form, which they call an *attribute-combiner*. In this architecture, the original attributes are not replaced with the class predictions, but instead they are added to them. As Schaffer (1994) shows in his paper about bi-level stacking, this may result in worse performance.

3.7 Conclusion

We have explored the parameter state space of Stacking. Concerning the choice of base classifiers, we have found a set of four base classifiers, chosen by a priori and a posteriori arguments, which performs best. However, using all available base classifiers also remains an acceptable option, although the performance may be slightly worse[13] since the dimensionality of the meta dataset is increased. When using predictions meta data, the performance difference tends to be smaller, probably because most machine learning algorithms are better suited to deal with nominal data and therefore seem to exhibit less vulnerability to curse-of-dimensionality.

[13] In our case, a penalty of at most four significant losses on twenty-six datasets is observed. Dependent on the type of meta data and meta classifier which is used, this may be much less – e.g. for StackingC, it is only one loss.

Concerning the choice of meta classifier and choice of meta data to be used, we have found that MLR is indeed the best meta classifier for probability distribution data. We showed probability distribution meta data and predictions meta data to perform comparably. For predictions meta data, the best meta classifier has less advantage because of less performance variation among the meta classifiers. NaiveBayes is a reasonable choice since it is once on first and once on a very close second place.

We believe that repeating our extensive base classifier experiments with a current variant, StackingC (see Chapter 5), will not yield new insights. Because StackingC can be viewed as meta classifier for probability distribution meta data, we can see it as alternative meta classifier. As such, we have investigated the distribution of accuracy ratios *StC4B* by *StC7B* and found it to be quite symmetric around 1.0, with the smallest standard deviation of all meta classifiers for *St4B* by *St7B* (Fig.3.4). Thus, StackingC seems to be least influenced by specific sets of base classifiers, which leads us to expect that base classifier choice has even less influence on StackingC than it has on Stacking. It still remains to be investigated whether this is also true for sMM5.

We hope that future research in Stacking will stay as exciting and interesting as it has been in the past and that our parameter proposals will bring the general ensemble learning scheme Stacking nearer to main-stream data mining.

Chapter 4

Meta-Learning for Stacking

In this chapter we describe new experiments with the ensemble learning method Stacking. The central question in these experiments was whether meta-learning methods can be used to accurately predict various aspects of Stacking's behaviour. The resulting contributions of this chapter are two-fold: When learning to predict the accuracy of Stacking, we found that the single most important feature is the accuracy of the best base classifier. A simple linear model involving just this feature turns out to be surprisingly accurate. When learning to predict significant differences between Stacking and three related ensemble learning methods, we have found simple models, all but one of which are based on single features which can be efficiently computed directly from the dataset. For one of these models, we were able to offer a tentative interpretation. These models may ultimately be used to decide in advance which ensemble learning scheme to use on a given dataset, since neither of them is always the best choice. Furthermore, aiming to understand these models can lead to new insights into Stacking's behaviour. This chapter is an extended version of (Seewald, 2002b).

4.1 Introduction

Meta-Learning[1] focusses on predicting the right algorithm for a particular problem based on characteristics of the dataset (Brazdil, Gama & Henry, 1994) or based on the performance of other, simpler learning algorithms (Pfahringer et al., 2000). Here we are concerned with Meta-Learning of *ensemble learning schemes*. Stacking can be considered the most general such scheme and was introduced in (Wolpert, 1992). However, here we use the extension due to Ting & Witten (1999). We take a more general view of Meta-Learning and use it to predict two aspects of Stacking's behaviour: accuracy as estimated via ten-fold crossvalidation; and also significant differences vs. related ensemble learning schemes as estimated via t-Test.

We will now describe our experimental setup and our two sets of features describing dataset characteristics and base classifier accuracy & diversity. Then we will give results for predicting the accuracy of Stacking, followed by Meta-

[1] The term *Meta-Learning* is used elsewhere to refer to learning from the output of component classifiers in ensemble learning schemes – so in a sense we achieve to relate both aspects of this term in this chapter.

Learning of significant differences. Afterwards we give a short overview on related Meta-Learning research.

4.2 Experimental setup

In our experiments, we used twenty-six datasets from the UCI machine learning repository (Blake & Merz, 1998). Details can be found in Table 3.2. We used Stacking with all of the following seven base classifiers for our experiments, which were chosen in an attempt to maximize diversity. All algorithms were taken from the Waikato Environment for Knowledge Analysis (WEKA[2]), Version 3-1-8. Learning algorithm parameters which have not been mentioned have been left at their default settings.

- DecisionTable: a decision table learner.
- IBk: the IBk instance-based learner using K=1 nearest neighbors. K=1 was chosen to offset the K* algorithm with a maximally local learner.
- J48: a Java port of C4.5 Release 8 (Quinlan, 1993a)
- KernelDensity: a simple kernel density classifier.
- KStar: the K* instance-based learner (Cleary & Trigg, 1995), using all nearest neighbors and an entropy-based distance function.
- MLR: a multi-class learner based on linear regression, which tries to separate each class from all other classes by linear discrimination (*Multi-response Linear Regression*)
- NaiveBayes: the Naive Bayes classifier using multiple kernel density estimation (-K) for numeric attributes.

We used the following four ensemble learning schemes.

- Stacking is the stacking algorithm as implemented in WEKA, which follows (Ting & Witten, 1999). It constructs the meta dataset by adding the entire predicted class probability distribution instead of only the most likely class. We used MLR as the level 1 learner.[3]
- X-Val chooses the best base classifier on each fold by an internal ten-fold CV. This is just the selection by crossvalidation we mentioned in the beginning.
- Voting is a straight-forward adaptation of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to the base classifier's estimate of the class probability distribution for the example. I.e. the mean class distribution of all classifiers is calculated. It is the only scheme which does not use an expensive internal crossvalidation.

[2] The Java source code of WEKA has been made available at `www.cs.waikato.ac.nz`.

[3] Relatively global and smooth level-1 (=meta) generalizers should perform well (Wolpert, 1992; Ting & Witten, 1999).

- Grading is an implementation of the grading algorithm evaluated in (Seewald & Fürnkranz, 2001) which uses IBk ($K = 10$) as meta classifier. Basically, Grading trains a meta classifier for each base classifier which tries to predict when its base classifier fails. This decision is based on the dataset's attributes. A weighted voting of the base classifiers' prediction gives the final class prediction. The voting weight is the confidence for a correct prediction of a base classifier, which is estimated by its associated meta classifier.

We used seventeen dataset-related features which uniquely characterize the dataset. These were inspired by the StatLOG project (Brazdil, Gama & Henry, 1994). Space restrictions prevent us from giving exact formulas for each case, but a reference implementation is available from the author upon request.

- *Inst*, the number of examples.
- $ln(Inst)$, the natural logarithm of *Inst*.
- *Classes*, the number of classes.
- *NumAttrs*, the number of attributes (excluding the class)
- *PropNomAttrs*, number of nominal attributes as a proportion of *NumAttrs*: $\frac{\#NominalAttrs}{NumAttrs}$
- *PropContAttrs*, number of numeric attributes as a proportion of *NumAttrs*: $\frac{\#ContinuousAttrs}{NumAttrs}$
- *PropBinAttrs*, number of binary-valued nominal attributes as a proportion of *NumAttrs*: $\frac{\#BinaryAttrs}{NumAttrs}$
- *ClassEntropy*, the entropy of the class attribute: $-\sum_{i=1}^{Classes} xldx\frac{\#Class_i}{Inst}$ where $\#Class_i$ is the number of examples with class i and $xldx(y) = y * log_2 y$. For proper handling of zero counts, we force $xldx(0) = 0$.
- *AttrEntropy*, mean attribute entropy: $-\frac{1}{NumAttrs}\sum_{j=1}^{NumAttrs}\sum_{\forall i} xldx\frac{\#Attr_jVal_i}{Inst}$ $\#Attr_jVal_i$ is the number of examples where the jth attribute takes on its ith value. For this and the next feature, continuous attributes were discretized via equal-width binning and $\frac{Inst}{10*Classes}$ bins.
- *MutualEntropy*, mean mutual entropy between attributes and the class: $AttrEntropy+\frac{1}{NumAttrs}\sum_{j=1}^{NumAttrs}\sum_{k=1}^{Classes}\frac{\#Class_k}{Inst}\sum_{\forall i} xldx\frac{\#Attr_jVal_iClass_k}{\#Class_k}$, where $\#Attr_jVal_iClass_k$ is the number of examples where the jth attributes takes on its ith value and the class is equal to k.
- *EquivAttrs*, the equivalent number of attributes, $\frac{ClassEntropy}{MutualEntropy}$
- *RelEquivAttrs*, $\frac{EquivAttrs}{NumAttrs}$
- *S/N*, the signal-to-noise ratio: $\frac{MutualEntropy}{AttrEntropy-MutualEntropy}$
- *MeanAbsCorr*, the mean absolute pairwise correlation over all unique pairs of numeric attributes and each class separately, where $Corr(x, y) = \frac{\sum_{\forall i}(x_i-\overline{x})*(y_i-\overline{y})}{\sqrt{|\sum_{\forall i}(x_i-\overline{x})^2\sum_{\forall i}(y_i-\overline{y})^2|}}$

- *MeanAbsSkew*, the mean absolute skew over all numeric attributes, where $Skew(x) = \frac{1}{n}\sum_{i=1}^{n} (\frac{x_i - \overline{x}}{stDev(x)})^3$

- *MeanAbsKurtosis*, the mean absolute kurtosis over all numeric attributes, where $Kurtosis(x) = \frac{1}{n}\sum_{i=1}^{n} (\frac{x_i - \overline{x}}{stDev(x)})^4 - 3$

- *defAcc*, the default or base-line accuracy, i.e. the proportion of the most common class – the expected performance of a trivial classifier which always predicts the most common class from training data.

Additionally, we used the accuracies of our seven base-learners as features. We also calculated standard statistical features of this set of seven accuracies. Furthermore, we used the same statistical features over pairwise base classifier κ-statistics[4], a measure of diversity due to (Dietterich, 2000b).

- Seven accuracy values, one for each base classifier (*DT, IBk-K1, J48, KD, KStar, MLR, NB-K*)

- Eight statistical features describing the set of accuracy values (*MinAcc, MaxAcc, MeanAcc, StDevAcc, SkewAcc, SkewAcc*^{2}*, KurtosisAcc, relRangeAcc* $= \frac{MaxAcc - MinAcc}{StDevAcc}$)

- Eight statistical features describing the set of all pairwise κ-statistics between base classifiers (*MinK, MaxK, MeanK, StDevK, SkewK, SkewK*^{2}*, KurtosisK, relRangeK*)

- *relMeanAcc*$=\frac{AvgAcc}{defAcc}$, the ratio of average accuracy to default accuracy.

The above features were computed both on predictions estimated from the full data set (training set accuracy and diversity) and on predictions estimated via tenfold crossvalidation. For Meta-Learning of significant differences, we only used the latter set because it consistently offered better estimates during the first task. This also simplified the experimental evaluation. All statistical differences for Meta-Learning were computed via a t-Test with α=99%, based on the accuracies generated by ten-times ten-fold crossvalidation.

4.3 Estimating Stacking's Accuracy

This section is concerned with predicting the accuracy of Stacking. In order to obtain a reasonable estimate, a ten-fold CV was used for accuracy estimation. We first investigated the simplest models possible: based on only a single feature. Thus, we assumed linear relationships between each feature and the accuracy of our stacked classifier and characterized this relation by statistical correlation coefficients and mean absolute errors (MAE). Afterwards, we considered more complex and non-linear models obtained by various regression algorithms from machine learning.

[4] A value of 1.0 stands for identical predictions between two learners while a value of 0.0 represents random correlations. A negative value signifies systematic disagreement between classifiers.

Table 4.1: This table shows the correlations and mean absolute errors (MAE) for dataset-related features vs. the accuracy. The first column shows the correlation for the full meta dataset (26 examples), the second column shows the correlation estimated by leave-one-out crossvalidation. Best features are shown in **bold**.

Dataset	**All**		**CV**	
Features	Corr	MAE	Corr	MAE
$Inst$	0.26	0.084	0.03	0.090
$logInst$	0.11	0.087	-0.40	0.095
$Classes$	0.37	0.089	-0.09	0.100
$NumAttrs$	0.07	0.087	-0.58	0.094
$PropNomAttr$	0.29	0.087	-0.04	0.095
$PropContAttr$	0.29	0.087	-0.04	0.095
$PropBinAttr$	0.20	0.088	-0.30	0.098
$ClassEntropy$	0.16	0.089	-0.54	0.100
$AttrEntropy$	0.26	0.085	-0.05	0.094
$MutualEntropy$	0.49	0.072	**0.42**	**0.077**
$EquivAttrs$	**0.58**	**0.068**	0.40	0.079
$RelEquivAttrs$	0.43	0.075	0.26	0.082
S/N	0.23	0.083	0.00	0.088
$MeanAbsCorr$	0.08	0.087	-0.47	0.093
$MeanAbsSkew$	0.06	0.087	-0.45	0.093
$MeanAbsKurtosis$	0.06	0.088	-0.37	0.095
$defAcc$	0.01	0.087	-0.94	0.095

Table 4.2: This table shows the correlation and mean absolute errors (MAE) on base classifier related features vs. the accuracy. For the first two columns, all features are based on training set performance of the base classifiers. For the last two columns, a ten-fold crossvalidation was used to determine better but more costly estimates of base classifier performance. The first and third column show the correlations and MAE on the full meta dataset, the second and fourth column show correlations and MAE estimated via leave-one-out crossvalidation.

Classifier	**AllT**		**CV T**		**All**		**CV**	
Features	Corr	MAE	Corr	MAE	Corr	MAE	Corr	MAE
DT	0.77	0.055	0.67	0.062	0.83	0.046	0.79	0.051
$IBk-K1$	0.72	0.072	0.69	0.076	0.95	0.030	0.94	0.032
$J48$	0.79	0.055	0.73	0.061	0.84	0.042	0.81	0.046
KD	0.69	0.076	0.53	0.086	0.95	0.029	0.94	0.031
$KStar$	0.59	0.084	-0.21	0.102	0.93	0.037	0.92	0.040
MLR	0.50	0.072	0.07	0.084	0.32	0.083	-0.11	0.097
$NB-K$	0.81	0.050	0.73	0.056	0.77	0.052	0.68	0.059
$MinAcc$	0.58	0.069	0.27	0.080	0.40	0.078	0.00	0.091
$MaxAcc$	0.71	0.072	0.68	0.088	**0.96**	**0.021**	**0.96**	**0.022**
$MeanAcc$	**0.84**	**0.045**	**0.81**	**0.049**	0.92	0.033	0.91	0.036
$StDevAcc$	0.58	0.065	0.36	0.074	0.26	0.083	0.05	0.088
$SkewAcc$	0.46	0.073	0.31	0.079	0.32	0.083	0.02	0.091
$SkewAcc^2$	0.35	0.081	0.16	0.087	0.20	0.083	-0.10	0.089
$KurtAcc$	0.39	0.077	0.22	0.083	0.29	0.083	0.12	0.088
$relRangeAcc$	0.40	0.075	0.26	0.080	0.20	0.090	-0.20	0.097
$MinK$	0.56	0.065	0.46	0.069	0.40	0.075	0.23	0.081
$MaxK$	0.00	0.087	-0.19	0.096	0.18	0.085	-0.14	0.091
$MeanK$	0.70	0.059	0.61	0.064	0.64	0.060	0.57	0.065
$StDevK$	0.48	0.063	0.36	0.068	0.15	0.086	-0.20	0.100
$SkewK$	0.58	0.071	0.43	0.078	0.66	0.064	0.58	0.070
$SkewK^2$	0.61	0.082	0.32	0.094	0.57	0.075	0.42	0.083
$KurtK$	0.52	0.088	0.09	0.100	0.59	0.070	0.47	0.077
$relRangeK$	0.08	0.090	-0.81	0.099	0.35	0.076	0.21	0.081
$relMeanAcc$	0.28	0.082	0.08	0.087	0.38	0.078	0.27	0.082

4.3.1 Linear Models based on Single Features

We computed statistical correlation coefficients and mean absolute errors (MAE) for all our features, always versus the accuracy of Stacking. The dataset-related features can be found in Table 4.1, and the base-classifier-related features in Table 4.2. Correlations and MAEs were determined for all meta data (**All**) and also via leave-one-out crossvalidation (**CV**). In the former, this estimate was based on the output of one linear regression model computed from all meta-examples[5] In the latter case, this estimate was based on the output of twenty-six linear models which were trained using all but one meta-example and tested on the remaining meta-example. The latter case is a more reliable indicator of model performance on unseen data than the former.

In the case of base classifier related features, we have an additional dimension: we can estimate the base classifier accuracies on the full dataset (**AllT, CV T**, i.e. training set accuracies) or via tenfold crossvalidation (**All, CV**), yielding two different set of features. Since Stacking uses CV internally, we expect **All** and **CV** to be better predictors for stacked accuracy. This is indeed the case – a single feature, *MaxAcc*, already yields excellent results. However, computing a crossvalidation on the original dataset comes with a non-negligible computational cost. A computational cost reduction by an order of magnitude could be obtained by using training set output to compute our features – which motivates **AllT** and **CV T**. As expected, in this case we get less good but still acceptable results for best single feature, *MeanAcc*.

As should be expected from a high-bias linear model, all base classifier related features show a graceful degradation from **All** to **CV**. We were surprised to note that this is not always true for the dataset-related features - about half of the features have a negative correlation for **CV** whose absolute value is higher than the positive correlation for **All**. This higher negative correlation can unfortunately not be used to predict stacked accuracy[6] and is always coupled to a large MAE. It seems that a lot of the dataset-related features are not relevant to this task or that a one-dimensional linear model is not appropriate to find a relevant relation.

4.3.2 Models based on multiple features

In order to test how we may improve our results by using multiple features, we resorted to using standard machine-learning approaches for regression on our meta dataset. We created one meta dataset with accuracy estimation via training set (*MetaTrain*) and one estimated via tenfold CV (*MetaCV*). The dataset-related features were included in both cases. We evaluated linear regression, LWR (*locally weighted regression*), model trees[7], regression trees, KStar and IBk instance based learners at the meta-level. Linear regression and model

[5] Each meta-example consist of features computed from one original dataset, either directly or indirectly, followed by the accuracy of the stacked classifier.

[6] The maximum negative correlation appears in feature *defAcc* (-0.94; **CV**) This correlation is based on twenty-six different models, one per leave-one-out training fold. All data would have to be used to determine the final regression line, but then this result can no longer be validated and seems certainly too optimistic. This is also indicated by the high MAE – about 4x as high as for *MaxAcc*.

[7] M5Prime from WEKA, see (Wang & Witten, 1997)

trees proved superior[8]. However, we were still unable to find any model which performed better than the best linear model based on a single feature[9].

Additionally, we considered using principal components analysis for feature transformation. Since using all available data for PCA gives indirect feedback about feature distribution to the learning system[10] it is necessary to compute the PCA transformation on training data only. In preliminary investigations, we found that for our small meta dataset the PCA projection changes quite drastically from one leave-one-out fold to the next and thus seems too unstable. More meta data would be necessary to apply this technique appropriately.

Concluding, features derived from classifiers seem to be more relevant in the context of predicting accuracy than those derived directly from the datasets, which was also found in (Bensusan & Kalousis, 2001). However, in our case linear models based on single features were sufficient to achieve best results. For example, the following regression line predicts Stacking's accuracy with a correlation of 0.96 and a MAE of 0.022:

$$StAcc = 1.074 * MaxAcc - 0.082$$

Notice that although it seems at first glance that Stacking performs slightly worse than the best component classifier, this view is biased: *MaxAcc*, i.e. the best base classifier by hindsight, is a less fair comparison than accuracy of X-Val since its decision is based on all available data while X-Val and Stacking only see the training data from the leave-one-out CV, i.e. all but one meta-instance. Notice also that although computing *MaxAcc* leaves us with a lot of data which could be used directly by Stacking, this would only enable us to compute the training set accuracy for Stacking and not the ten-fold cv estimate we used here.

Given our results, it is surprising that other Meta-Learning approaches have not considered that quite simple models may suffice, but instead rely on complex models whose interpretation may be quite difficult.

4.4 Meta-Learning of Significant Differences

This section is concerned with predicting significant differences between Stacking and three other ensemble learning schemes. For each of Stacking vs. Voting, Stacking vs. Grading and Stacking vs. X-Val, we generated a separate meta dataset consisting of all dataset-related and classifier-related features[11] followed by a binary class variable, being 1 if Stacking is significantly better than the other scheme and 0 otherwise. In case of no significant difference, we removed the respective example from the meta dataset, under the premise that in this case

[8] Both were always top two by highest correlation and lowest MAE with the rest of the field – usually far – behind.

[9] We also evaluated a wrapper-based feature subset selection which systematically optimizes the performance of a learning algorithm by removing features, using the algorithm as its own model. For *MetaTrain* we were thus able to improve its performance – however, given the small size of the meta dataset we feel that these results may be strongly biased by overfitting.

[10] We found that using all data for the PCA does not enable us to predict Stacking's accuracy better, but it does allow us to solve all three learning problems from the next section equally well, each via a single threshold on the largest principal component – clearly too good to be true.

[11] Because of the much better results in predicting Stacking's accuracy and also to simplify our experiments, we only considered those classifier features estimated via crossvalidation.

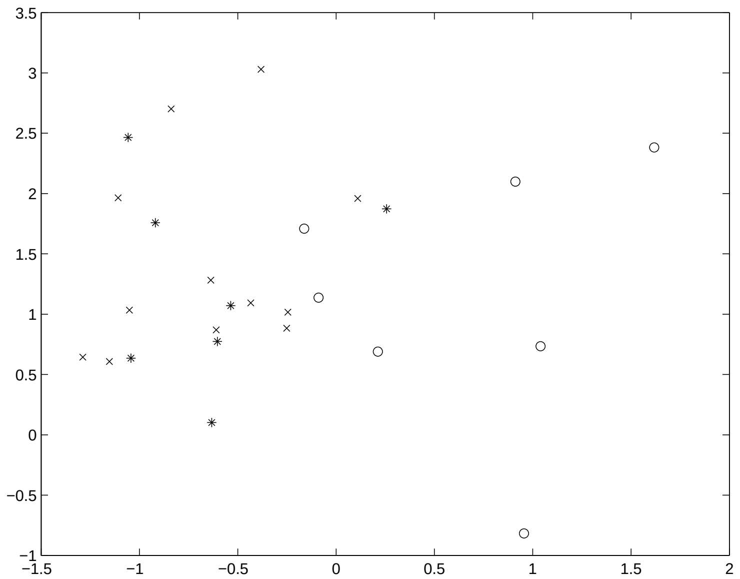


Figure 4.1: This figure shows a projection of the Stacking vs. Voting meta dataset onto two dimensions, via principal component analysis (53.1% of variance explained). $*$ stands for class=1 (where Stacking is better) and $\circ$ for class=0. Those datasets which were no part of the meta dataset (no significant differences between Stacking and Voting) are also shown, as $\times$.

we can consider both variants to be equivalent and thus judge either answer to be correct.

On these meta datasets, we evaluated a number of standard machine learning algorithms available in WEKA[12] via leave-one-out crossvalidation. We only discuss the best models which in most cases seem to be rather simple and based on single attributes only, hinting that they may be robust. Smaller values for crossvalidation (e.g. seven-fold, six-fold) were evaluated as additional measure for model robustness. In one case, we were even able to give a tentative interpretation of the model.

4.4.1 Stacking vs. Voting

For Stacking vs. Voting, there are twelve datasets without significant differences. After removing them from our meta dataset, we have fourteen instances, seven with class=1, seven with class=0. The baseline accuracy is thus 50%. Here, IBk is the best meta-learner with an accuracy of 92.86% and a single error for *vote*. A crossvalidation using only seven folds produces the exact same result. Figure 4.1 shows a projection via PCA of our meta dataset into two dimensions. The single $*$ in the $\circ$ territory (near X=0.5,Y=2) corresponds to the single error, dataset *vote*.

When removing the base classifier dependent features, IBk is still the best classifier with an additional error on *labor*, the smallest dataset. In this case

[12] All base learners plus 1R and DecisionStump

MLR, another high-bias and global learner, is equally good. So we may tentatively conclude that for this meta dataset, there seems to be no single feature which can predict the significant differences as well as a combination of all features.

4.4.2 Stacking vs. Grading

For Stacking vs. Grading, there are again twelve datasets on which there are no significant differences. After removing them from our meta dataset, we have fourteen instances whose classes are again equally distributed. Thus the baseline accuracy is also 50%. Here, J48 is the best choice with 92.86% accuracy and only a single error on the smallest dataset, *labor*. The training set model is based on a single attribute, *propNomAttr*. In all fourteen folds but two there is the same model, which also appears as the training set model.

$$\begin{array}{lcl} propNomAttr <= 0.684211 & : & class = 1 \\ otherwise & : & class = 0 \end{array}$$

In the two other folds, the same attribute appears in the same formula with 0.65 and 0.695652 resp. as value on the right side. It seems that the proportion of nominal attributes plays a role on the performance between Stacking and Grading: in case there about $\frac{2}{3}$ or less of the attributes are nominal, Stacking works significantly better than Grading.

A tentative explanation for this model may be that a smaller proportion of nominal attributes makes learning harder for the base-learners, since most of them are better equipped to handle nominal data. Stacking seems to be able to compensate for this, since its meta-level data is independent of the base-level data[13] and is processed by MLR which is well equipped to handle numeric data. However, Grading seems to be unable to compensate for this since its meta-level data contains just the base-level attributes. Thus its meta learner IBk can be expected to be susceptible to a smaller proportion of nominal attributes in the same way as the base learners.

4.4.3 Stacking vs. XVal

For Stacking vs. X-Val, seventeen examples offer no significant differences. Only nine examples remain for our experiments, the baseline accuracy is already 66.7%. Interestingly in this case the best model is from DecisionStump which learns a single J48 node, obtaining 88.9% accuracy, corresponding to a single error on dataset *balance-scale*. It seems J48 is prone to overfitting on this meta dataset. The training set model is based on *meanAbsSkew*. All but two times, the following model appears:

$$\begin{array}{lcrcl} meanAbsSkew & <= & 0.31 & : & class = 0 \\ meanAbsSkew & > & 0.31 & : & class = 1 \\ meanAbsSkew & & missing & : & class = 0 \end{array}$$

Once the same model appears with value 0.53 instead of 0.31. Once a model based on *numClasses* $<= 13$: *class* $= 1$ appears. The same overall accuracy

[13] The meta-level data for Stacking consists of class probability distributions from all base learners.

is also obtained in a six-fold crossvalidation. Still, this is probably those of our models which is least trustworthy.

4.5 Related Research

Up to now there is no research aiming to either predict the accuracy of ensemble learning schemes or to predict which ensemble learning scheme to use for a given dataset. Here we have investigated both tasks and found them to work quite well. We give some representative examples.

Using regression to predict the performance of basic learning algorithms was first investigated in (Gama & Brazdil, 1995), continuing the framework of StatLOG(Brazdil, Gama & Henry, 1994). They report poor results in terms of normalized mean squared error.

(Brazdil, Gama & Henry, 1994) have investigated meta-level learning to predict the best classifier for a given dataset. They use a confidence interval around the best accuracy to define *applicable* [14] and *inapplicable* classifiers for each dataset. Our approach uses the statistical t-Test instead. While their approach has to integrate possibly conflicting rules concerning applicability, making the evaluation quite complex, our approach can predict significant differences directly. Furthermore, the focus on using only decision trees and derived rules may have lead to inferior results as we had to use a variety of machine learning techniques to get best results. They also considered only one-against-all comparisons between candidate classifiers instead of the pairwise comparisons which we investigated.

(Pfahringer et al., 2000) investigated Meta-Learning using landmarkers, which are fast and simple learning algorithms used to characterize the dataset. The features which we derived from base classifiers can be considered similar. They report improved results by landmarking which we also observed for predicting the accuracy of Stacking. However, for predicting significant differences we found dataset-related features to be more appropriate. They report work on removing those examples from the meta dataset where no significant differences were found and show that in some cases this can hurt Meta-Learning performance.

(Bensusan & Kalousis, 2001) have recently investigated Meta-Learning in a similar setting, using features extended from STATLOG, histograms of numeric attributes and landmarking using seven learners, four of which are also used by us as base classifiers. They considered three meta-learners. They found that landmarking is superior to predict errors of classifiers which was confirmed in our experiments. However, they found that landmarking is not useful to generate good rankings for classifiers – no meta-learner is able to perform better than a default strategy. Our results indicate that for predicting significant differences, landmarking features are also less useful. Mean absolute errors are given, but no statistical correlation measure can be found for the regression experiments. Some regression rules from Cubist are shown and interpreted, even though they state that ranking determine from Cubist's error prediction always perform worse than the default.

[14] within confidence interval ≡ *applicable*

4.6 Conclusion

In this chapter we have investigated the use of machine learning techniques in the context of Meta-Learning both to predict stacked classifier accuracy and significant differences between Stacking and three other ensemble learning schemes. We used both dataset-related and base classifier related features in our tasks.

In the context of predicting classifier accuracy, we found that classifier-related features, namely some of those derived from accuracy, are excellently suited to this task, as have others e.g. (Bensusan & Kalousis, 2001; Pfahringer et al., 2000). A single feature, the accuracy of the best component classifier in the ensemble, is able to predict the accuracy of the stacked classifier quite well. Other Meta-Learning approaches seem not to take into account that such simple models may be competitive to more complex models.

In the second part we investigated the prediction of significant differences between Stacking and other ensemble learning schemes. In this case we found that features derived directly from the dataset were usually better suited. For the model which predicts significant differences between Grading and Stacking, insight into the inner workings of both schemes have enabled us to formulate a tentative explanation of the learned model.

Our Meta-Learning experiments were constructed on the premise to predict significant differences only where they appear. While removal of *ties* from the meta dataset was previously mentioned in work by (Brazdil, Gama & Henry, 1994), using a less ad-hoc and more appropriate statistical test to determine those ties seems to have been overlooked previously.

At last we have found that there is no single best meta classifier for predicting significant differences – a variety of machine learning algorithms had to be evaluated for best results. Although most of our best models were based on single features, it seems that no single learning algorithm is able to find all of them. This hints that learning problems which aim to distinguish between pairs of classifiers may have quite different properties, which could explain why meta-learning a single model for many classifiers at once is so hard.

Given that our meta-dataset is very small – only twenty-six examples – and shrinks even further when removing non-significant entries, all our results may be strongly biased by overfitting. Independent verification of our learned models in new domains is still necessary before the models can be taken seriously. However, we believe that our results on Stacking's accuracy will prove to remain valid, while we are far more sceptical of our models which predict significant differences.

Chapter 5

Improving upon Stacking: Stacking with Confidences

We investigated performance differences between multi-class and two-class datasets for ensemble learning schemes. We were surprised to find that Stacking, the most general such scheme, performs worse on multi-class datasets when using class probability distributions as meta-level data. In this chapter we will present results concerning this heretofore unknown weakness of Stacking. In addition we will present a new variant of Stacking, using meta-learner MLR, which is able to compensate for this weakness, improving Stacking significantly on almost half of our multi-class datasets. Two other related meta-learners could also be improved using the same idea. The dimensionality of the meta data set is reduced by a factor equal to the number of classes, which leads to faster learning. In comparison to other ensemble learning methods this improves Stacking's lead further, making it the most successful system by a variety of measures. This chapter is an extended version of (Seewald, 2002a).

5.1 Introduction

We have investigated the performance of four related ensemble learning schemes, including the mentioned Stacking variant, relative to multi-class vs. two-class datasets, in a ranking. We found no significant differences for Grading and Voting[1] – however, the mentioned Stacking variant showed a significant performance degradation for multi-class datasets. This performance degradation is also quite apparent by other accuracy-based measures. We will present these results and a variant of stacking which does not show this performance degradation.

First, we will present the basic concept behind Stacking and show how it can be extended towards StackingC for the meta-learner MLR. Then we will describe the experimental setup, base classifiers and ensemble learning schemes used and investigate empirically the claims that Stacking performs worse on multi-class datasets. Afterwards we will compare StackingC to Stacking in more detail, considering significant differences by dataset, influence of *number of classes* and shortly investigate the claim of faster learning for StackingC. At last we will

[1] X-Val seems to be worse on two-class datasets.

$Attrs$	$Cl.$
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
⋮	⋮
$AttrVec_n$	a

(a) original training set

	$Classifier_i$	
a	b	c
$P_{i,a1} = \mathit{0.90}$	$P_{i,b1} = 0.05$	$P_{i,c1} = 0.05$
$P_{i,a2} = 0.15$	$P_{i,b2} = \mathit{0.70}$	$P_{i,c2} = 0.15$
$P_{i,a3} = 0.10$	$P_{i,b3} = \mathit{0.80}$	$P_{i,c3} = 0.10$
$P_{i,a4} = 0.20$	$P_{i,b4} = 0.20$	$P_{i,c4} = \mathit{0.60}$
⋮	⋮	⋮
$P_{i,an} = \mathit{0.80}$	$P_{i,bn} = 0.10$	$P_{i,cn} = 0.10$

(b) sample class probability distribution

	$Classifier_1$			$Classifier_2$				$Classifier_N$		$class$
a	b	c	a	b	c		a	b	c	$= a?$
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	...	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	1
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	...	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	0
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	...	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	0
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	...	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	0
	⋮			⋮				⋮		⋮
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	...	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	1

(c) meta training set for class a, Stacking with MLR

$Classifier_1$	$Classifier_2$		$Classifier_N$	$class$
a	a		a	$= a?$
$P_{1,a1}$	$P_{2,a1}$	...	$P_{N,a1}$	1
$P_{1,a2}$	$P_{2,a2}$	...	$P_{N,a2}$	0
$P_{1,a3}$	$P_{2,a3}$	...	$P_{N,a3}$	0
$P_{1,a4}$	$P_{2,a4}$	...	$P_{N,a4}$	0
⋮	⋮		⋮	⋮
$P_{1,an}$	$P_{2,an}$	...	$P_{N,an}$	1

(d) meta training set for class a, StackingC with MLR

Figure 5.1: Illustration of Stacking and StackingC on a dataset with three classes (a, b and c), n examples and N base classifiers. $P_{i,jk}$ refers to the probability given by base classifier i for class j on example number k

discuss results about applying the same basic idea to other meta classifiers and point towards interesting research directions for the future.

5.2 StackingC

The main idea behind Stacking is using the output from a set of level-0 (=base) classifiers, estimated via crossvalidation, to learn a level-1 (=meta) classifier which gives the final prediction.

As Ting & Witten (1999) propose, we use Stacking with *multi-response linear regression* (MLR) as level-1 classifier. Basically, MLR learns a linear regression function for each class which predicts degree of confidence in class membership and can, after normalization, be interpreted as class probability. Other level-1 classifiers do not usually learn a distinct model for each class, but instead learn a single model for all classes at once, e.g. a decision tree.

During prediction, the base classifiers are queried for their class probability distributions which are then used as input for the regression models (one for each class). The output of the linear models is renormalized to yield a proper class probability distribution.

Figure 5.1 shows an example with three classes (a, b and c), n examples and N base classifiers. 1(a) shows the original training set with its attribute vectors and class values.

Figure 5.1(b) shows how a class probability distribution of one sensible classifier may look like. The maximum probabilities are shown in *italics* and denote the classes which would be predicted for each example. There is one such set of class probability distributions for each base classifier.

Figure 5.1(c) shows the meta training set for Stacking which is used to learn a linear regression function to predict the probability of $class == a$. We denote $P_{i,jk}$ to signify the probability given by base classifier i for class j on example number k. The classes are mapped to an indicator variable such that only class a is mapped to 1 and all other classes to 0. In our example there are of course two other such training sets for class b and c which differ only in the last column and are thus not shown.

The proposed variant, StackingC, differs in these points: for each linear model associated with a specific class, only the partial class probability distribution which deals with this very class is used during training and testing. While Stacking uses probabilities for all classes and from all component classifiers for each linear model, StackingC uses only the class probabilities associated with the class which we want our linear model to predict.[2]

Figure 5.1(d) shows the corresponding meta training set for StackingC which consists only of those columns from the original meta training set which are concerned with class=a; i.e. $P_{i,ak}$ for all i and k. While the meta training sets for Stacking's meta classifier differ only in the last attribute (the class indicator variable), those for StackingC have fewer attributes by a factor equal to the number of classes and also have no common attributes. Out of necessity, this leads to more diverse linear models which we believe to be one mechanism why it outperforms Stacking. Another one may simply be that with fewer attributes, the learning problem becomes easier to solve, as long as only irrelevant information is removed.

As can be easily seen, this modification should not change the performance for two-class datasets significantly. Since the sum of each class probability distribution has to be one, the probability of one class is one minus the probability of the other class, so one of these values is sufficient to encode the complete information[3] Thus we would expect two-class datasets to offer equally good performance under this modification, but train slightly faster[4] because of the inherent dimensionality reduction for meta data.

[2] We also switched off the internal feature subset selection in MLR since that seemed to slightly improve performance – possibly because all features from the focussed meta-level data are already relevant.

[3] A linear model is free to use either the one attribute with a positive weight or the other with a negative weight, using appropriate constant terms.

[4] The training costs for the base classifiers are of course unchanged.

5.3 Experimental Setup

We implemented StackingC in Java within the Waikato Environment for Knowledge Analysis (WEKA[5]). All other algorithms at the base and meta-level were already available within WEKA.

For an empirical evaluation we chose twenty-six datasets from the UCI Machine Learning Repository (Blake & Merz, 1998), shown in Table 3.2. These datasets include fourteen multi-class and twelve two-class problems. Reported accuracy estimates are the average of ten ten-fold stratified cross validations unless otherwise noted. Significant differences were evaluated by a t-test with significance level of 99%.[6]

We chose four ensemble learning schemes, including Stacking.

- Grading is the implementation of the grading algorithm evaluated in (Seewald & Fürnkranz, 2001). It uses the instance-based classifier IBk with ten nearest neighbors as meta-level classifier.

- X-Val chooses the best base classifier on each fold by an internal ten-fold CV on the training data. This is just Selection by Crossvalidation which we mentioned in the beginning.

- Voting is a straight-forward adaptation of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to the base classifier's estimate of the class probability distribution for the example. It is mainly included as a benchmark of the performance that could be obtained without resorting to the expensive CV of every other algorithm.

- Stacking is the stacking algorithm as implemented in WEKA, which follows (Ting & Witten, 1999). It constructs the meta dataset by adding entire class probability distributions instead of only the most likely class. Following (Ting & Witten, 1999), we also used MLR as the level 1 learner.

All ensemble learning schemes used the following six base learners, which were chosen to cover a variety of different biases.

- DecisionTable: a decision table learner.

- J48: a Java port of C4.5 Release 8 (Quinlan, 1993a)

- NaiveBayes: the Naive Bayes classifier using multiple kernel density estimation for continuous attributes.

- KernelDensity: a simple kernel density classifier.

- MLR: a multi-class learner which tries to separate each class from all other classes by linear regression (*multi-response linear regression*)

- KStar: the K* instance-based learner (Cleary & Trigg, 1995)

[5] The Java source code of WEKA has been made available at `www.cs.waikato.ac.nz`

[6] The used t-test has been shown to have a high type I error e.g. in (Dietterich, 1998). Although we obtained similar results using a single ten-fold crossvalidation and χ^2 test after McNemar – which should have a low type I error according to the same paper – our reported significant differences may still be too optimistic.

Table 5.1: This table shows the performance of Stacking with different meta-learners by four measures which are described in the text. Performance on multi-class and two-class datasets is shown separately, in two adjacent columns. All data is based on a single ten-fold crossvalidation.

	DecisionTable		J48		KernelDensity	
	2Cl	mulCl	2Cl	mulCl	2Cl	mulCl
Avg.acc.	0.842	0.709	0.839	0.827	0.815	0.828
by Acc_{best}	0.974	0.798	0.971	0.958	0.939	0.960
by Acc_{XVal}	0.988	0.809	0.985	0.971	0.953	0.973
by Acc_{Voting}	0.987	0.810	0.983	0.972	0.951	0.974

	KStar		MLR		NaiveBayes	
	2Cl	mulCl	2Cl	mulCl	2Cl	mulCl
Avg.acc.	0.816	0.810	0.856	0.826	0.852	0.825
by Acc_{best}	0.943	0.937	0.990	0.963	0.987	0.950
by Acc_{XVal}	0.956	0.949	1.005	0.976	1.001	0.963
by Acc_{Voting}	0.955	0.950	1.003	0.977	0.999	0.964

All algorithms are implemented in WEKA Release 3.1.8. Each of them returns a class probability distribution, i.e., they do not predict a single class, but give probability estimates for each possible class. Parameters for learning schemes which have not been mentioned were left at their default values.

5.4 Multi-class vs. two-class datasets

In this section, we present results concerning the inferior performance of Stacking with MLR on multi-class datasets and show that all but one meta-learner also suffer from the same weakness.

This weakness is apparent by a variety of measures. By average accuracy[7], Stacking with MLR performs slightly worse on multi-class datasets. However, since average accuracy is not a reliable measure because of the different baselines involved, we investigated three different ways of normalizating the accuracy ($\frac{Acc_{Stacking}}{Acc_{best}}$[8], $\frac{Acc_{Stacking}}{Acc_{XVal}}$, $\frac{Acc_{Stacking}}{Acc_{Voting}}$[9]) and computed the geometric mean[10] of these accuracy ratios, both for two-class and multi-class datasets separately. Detailed results can be found in Table 5.1, column MLR. All these measures agree that Stacking performs about 3% worse on multi-class datasets. A ranking of all ensemble learning schemes based on significant differences also shows this weakness, see Table 5.4. While the latter may be too optimistic due to the used t-test having a high type-I error, the overall agreement of these different measures seems to make this weakness quite obvious.

When using all available base learners also as meta-learners for Stacking, i.e. running six experiments which use the same set of base learners but different

[7] See Table 5.2 and Table 5.1.

[8] Acc_{best} = accuracy of best base classifier by hindsight according to X-Val, i.e. estimated on the complete dataset.

[9] X-Val and Voting are the accuracies of resp. ensemble learning schemes

[10] For ratio values, the geometric mean is more appropriate.

meta-learners, we found that in all but one case[11], again for average accuracy and all three normalization methods, the performance on multi-class datasets was worse than on two-class datasets. So we conclude that this may be a general weakness for Stacking with probability distributions as meta-level data, i.e. of the extension proposed by Ting & Witten (1999).

There are three reasonable explanations: Firstly, since the number of classes is proportional to the number of features in the meta-level dataset, a higher number of classes makes learning harder simply because there are more features which have to be considered – i.e. the curse of dimensionality. Secondly, since Stacking with MLR as meta-learner uses almost the same meta-level data to train each linear model (i.e. only the class indicator feature is different), a higher number of classes may decrease the diversity among those linear models. Thirdly, the base classifiers themselves may be susceptible to the curse of dimensionality and pass this susceptibility on to Stacking.

The previous result which hints at a general weakness for Stacking with probability distributions supports the first explanation and discounts the second, since e.g. J48 does not learn one model per class but one model for all classes at once and thus decreased diversity of class models cannot be used as explanationn.

If the first explanation is therefore correct, Stacking with predictions as base data should not suffer from this weakness. Accordingly, our experiments show that, when normalizing with Acc_{Voting}, there are three meta-learners which perform better on multi-class datasets and three meta-learners which perform better on two-class datasets – as would be expected by chance. The observed differences in performance are at most 1%. Average accuracy does not agree with this conclusion, but it is an unreliable indicator at best.

When we normalize with Acc_{XVal} or Acc_{best}, we also get similar conflicting results. However, according to our ranking (see Table 5.4), X-Val performs worse on two-class datasets while Voting offers more balanced results. Thus, when using Acc_{XVal} or related measure Acc_{best} for normalization, this leads to an systematic overestimation of the performance on two-class datasets and thus to a relative underestimation of the performance on multi-class datasets.

So we still conclude that Stacking with predictions does not seem to suffer from the mentioned weakness. This also discounts the third explanation. Thus, even though the base classifiers may still perform worse on multi-class problems, Stacking with predictions seems to be able to compensate for this bias – as is StackingC which we will see shortly.

5.5 StackingC versus Stacking

In this section we will compare StackingC and Stacking in detail, focussing on performance and runtime differences. Table 5.2 shows detailed accuracies, standard deviations and significant differences on all datasets and also average accuracy on two-class and multi-class datasets respectively. The last measure has to be interpreted carefully, since it combines problems with different baselines. Also, the ratio between runtimes[12] is given.

[11] When using KernelDensity as meta-learner, the performance on multi-class datasets is indeed better. However, all measures also agree that Stacking with KernelDensity performs worse than with MLR.

[12] This ratio has to be interpreted carefully, since the experiments were run on a heterogenous cluster of linux and sun machines with a dynamic mapping of tasks to machines. However,

Table 5.2: This table shows accuracies ± standard deviations of Stacking and StackingC. RRT shows $\frac{St}{StC}Runtime$, i.e. the runtime ratio, greater than one where StackingC is faster. Diff shows + and - for significant wins resp. losses of StackingC to Stacking and is empty in case of no significant differences.

DS	Cl.	RRT	StackingC	Stacking	Diff
aud	24	2.86	82.17%	76.02%	+
aut	7	2.00	84.20%	82.20%	+
b-s	3	1.00	90.22%	89.50%	+
b-c	2	1.97	72.13%	72.06%	
b-w	2	1.08	97.38%	97.41%	
col	2	2.21	84.70%	84.78%	
c-a	2	2.30	86.22%	86.09%	
c-g	2	1.65	76.24%	76.17%	
dia	2	1.15	76.48%	76.32%	
gla	7	1.67	77.20%	76.45%	
h-c	5	1.25	84.09%	84.26%	
h-h	5	1.27	85.10%	85.14%	
h-s	2	1.67	84.30%	84.04%	
hep	2	1.13	82.97%	83.29%	
ion	2	1.23	92.82%	92.82%	
iri	3	0.90	95.40%	94.93%	
lab	2	1.08	90.88%	91.58%	
lym	4	1.19	81.82%	80.20%	+
p-t	22	17.73	47.23%	42.63%	+
seg	7	2.00	98.10%	98.08%	
son	2	1.88	85.63%	85.58%	
soy	19	2.80	93.47%	92.90%	
veh	4	1.72	79.36%	79.89%	
vot	2	1.81	96.34%	96.32%	
vow	11	2.64	99.07%	99.00%	
zoo	7	2.37	96.24%	93.96%	+
Avg(2Cl)	2	1.60	85.51%	85.54%	
Avg(mCl)	9.14	2.96	85.26%	83.94%	6+

Table 5.3: This table shows the improvement of StackingC over Stacking as $\frac{Acc_{StackingC}}{Acc_{Stacking}}$ for meta-learners MLR, LWR and M5Prime. All data here is based on a single ten-fold crossvalidation.

Meta-learner	multi-class	two-class
MLR	1.0375	0.9983
LWR	1.0256	0.9997
M5Prime	1.0070	1.0000

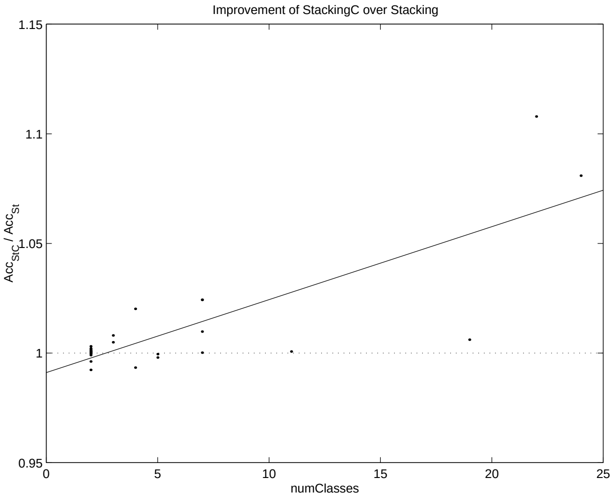


Figure 5.2: This figure shows the improvement of StackingC over Stacking as $\frac{Acc_{StackingC}}{Acc_{Stacking}}$ as a function of the number of classes. The solid line shows the obtained least squares fit for a one-dimensional linear model while the dotted line indicates a ratio of 1.0. Above the latter line, StackingC is better than Stacking and vice versa.

Summarizing the table, StackingC wins six times against Stacking, always on multi-class datasets, and never loses. In all but one case, StackingC is faster – on average it is 2.3 times faster. StackingC also improves on accuracy by an average of 3.75% for multi-class datasets; the performance difference for two-class datasets is negligible, see Table 5.3.

We would also expect StackingC to improve on Stacking more, if the number of classes is increased. This is usually the case, see Figure 5.2. The statistical correlation coefficient for the shown relation is 0.8. Our figure also shows the fitted regression line which has a mean squared error of 2.59E-04.

Concluding, StackingC improves on Stacking in terms of significant accuracy differences, accuracy ratios and runtime. These improvements are more evident for multi-class datasets and have some tendency to become more pronounced as the number of classes increases. StackingC also resolves the weakness of Stacking in the extension proposed by Ting & Witten (1999) and offers balanced performance on two-class and multi-class datasets.

5.6 Discussion

To find out whether or not our approach is also able to improve other meta-learners besides MLR, we conducted additional experiments.

Essentially, we tried two different approaches. One was to use two other regression learners instead of MLR to approximate the class membership func-

since all tasks were run ten times, each time on a different machine, we still consider these to be rough but useful estimates.

Table 5.4: This table shows a ranking of all ensemble learning schemes with Stacking and StackingC. We are mostly concerned with the differences between Stacking and StackingC on multi-class datasets. Separate rankings are given on multi-class and two-class datasets. Ranks are given as Wins/Losses with the wins counting for the algorithm in the row (Scheme).

Scheme		X-Val	Grading	Stacking	StackingC	Voting
X-Val	on multi-class ds.	0/0	3/3	3/3	2/6	5/5
Grading	on multi-class ds.	3/3	0/0	5/4	1/5	3/2
Stacking	on multi-class ds.	3/3	4/5	0/0	0/6	4/5
StackingC	on multi-class ds.	6/2	5/1	6/0	0/0	5/3
Voting	on multi-class ds.	5/5	2/3	5/4	3/5	0/0
X-Val	on two-class ds.	0/0	2/3	0/3	0/3	2/4
Grading	on two-class ds.	3/2	0/0	2/3	2/4	1/0
Stacking	on two-class ds.	3/0	3/2	0/0	0/0	3/2
StackingC	on two-class ds.	3/0	4/2	0/0	0/0	4/2
Voting	on two-class ds.	4/2	0/1	2/3	2/4	0/0

tions for each class separately. This worked quite well, see Table 5.3. MLR is the meta-learner which we previously used for StackingC, but we used only the first crossvalidation in order to enable a fair comparison to the other learners. LWR stands for *locally weighted regression*[13]and M5Prime stands for a model tree learner. Both learners are available within WEKA. In terms of absolute performance over all datasets, LWR is slightly better than MLR and M5Prime is slightly worse.

The second approach, namely modifying common machine learning algorithms[14] to use only partial probability distributions during prediction, yielded catastrophically bad results on some datasets.

These results indicate that the source of StackingC's improvement may lie more in the diversity of class models than in the dimensionality reduction, although both play a key role (see also Section 2, paragraph 8). Using one-against-all class binarization and regression learners seems to be essential. So we intend to look into other binarization methods in the future.

Another interesting venue for future research may be to find out why X-Val performs worse on two-class datasets as our ranking indicates. A tentative explanation may be that the base classifiers perform better on two-class datasets. Thus, their accuracies are more similar on these datasets, increasing the probability that X-Val will choose a suboptimal learner by its internal CV. Further research is needed to find out if this is indeed the case. This may have far-reaching consequences because of the ubiquitousness of X-Val as a method to choose the best classifier throughout the machine learning community.

[13] We used parameter -W 1 for inverse kernels $\frac{1}{x}$

[14] NaiveBayes, KStar, IBk and KernelDensity.

5.7 Related Research

Džeroski and Zenko (2002) investigate Stacking in the extension proposed by Ting & Witten (1999). They introduce a new variant sMM5 which they claim to be *in a league of its own*. Their new variant is quite competitive to our variant StackingC but much slower, according to unpublished experiments on our twenty-six datasets. However, combining both ideas does not improve performance.

Todorovski & Džeroski (2000) introduce a novel method to combine multiple models. Instead of directly predicting the final class as all combining schemes we considered, their meta-learner MDT, based on C4.5, specifies which model to use for each example based on statistical and information theoretic measures computed from the class probability distribution. While their approach may make the combining scheme more comprehensible by learning an explicit decision tree decision tree, it is unclear whether this leads to better insight as well. Stacking and StackingC using MLR as meta-learner also allow to determine relative importance of base learners per class, simply by inspecting the weights of meta-level attributes after training – this has e.g. been done by Ting & Witten (1999).

Ting & Witten (1999) deal with the type of generalizer suitable to derive the higher-level model and the kind of attributes it should use as input. Using probability distributions as they propose instead of just predictions is essential to our variant StackingC. They also investigated the usefulness of non-negativity constraints for feature weights within linear models, but found that it is not essential to get the best performance. However they found it may be useful to facilitate human comprehension of these models. Since our focus was on performance and not on comprehensibility, we did not use a non-negativity constraint. In the future it may be interesting to look into comparing their linear models to those found by StackingC to see where they differ.

5.8 Conclusion

We have presented empirical evidence that Stacking in the extension proposed by (Ting & Witten, 1999) performs worse on multi-class datasets than on two-class datasets, for all but one meta-learner we investigated.

This can be explained as follows: With a higher number of classes, the dimensionality of the meta-level data is proportionally increased. This higher dimensionality makes it harder for meta-learners to find good models, since there are more features to be considered. Stacking using meta-level data consisting of predictions does not suffer from this weakness, as would be expected.

In order to improve on the status quo, we have proposed and implemented a new Stacking variant, named StackingC, based on reducing the dimensionality of the meta dataset so as to be independent of the number of classes and removing a priori irrelevant features, and shown that it resolves this previously unreported weakness, for MLR and two other related meta-learners considered. We believe that the source of this improvement lies partially in the dimensionality reduction, but more importantly in the higher diversity of class models. Using one-against-all class binarization and regression learners for each class model seems to be essential.

Chapter 6

Learning Curves

In this chapter, we investigate the hypothesis that StackingC is the most stable ensemble learning scheme. We define the stability of a learner as its continued good performance in the face of a reduction in training data, i.e. its capability to find the best or a reasonably good model when confronted with less training data, and – in case of insufficient training data – a graceful degradation in performance. Our definition of stability is thus not directly related to the concept of *high bias* which is associated with learners such as MLR whose learned model is less influenced by small changes in the data. However, high bias may be a necessary or at least useful condition for a stable learner – provided it is able to find the correct model in the first place.

Based on the observation that Stacking, especially StackingC, seems able to improve upon simpler schemes such as Voting, we expected that meta-level learning in Stacking would also be able to compensate better for the worse estimation of meta-level data which occurs due to less training data than simpler schemes, e.g. unweighted voting. As framework to evaluate this hypothesis empirically we chose learning curves.

6.1 Introduction and Experimental Setup

Learning curves give a good picture of learning algorithms with respect to our concept of stability. By measuring the performance of our algorithms at different training set sizes, we are able to observe the reduction in accuracy directly. The error is computed via hold-out estimate from the same fixed test set. To increase stability and also to estimate variance for significance tests, the same procedure was repeated thirty times, with different random ordering. We used hold-out error estimates since crossvalidation on smaller and smaller training sets is problematic, because both the training and test sets get smaller, thus the error estimate rapidly becomes less reliable. By using the same fixed test set in each run, the error estimate is far more reliable; furthermore, far less correlation between training data sets exists in that case. Clearly, there is also an computational advantage, but we do not consider this a significant factor. At last, it is also interesting to see how StackingC performs when compared via other methodologies and not the ubiquitous ten-times ten-fold crossvalidated t-test.

The training and test sets for our learning curves were created as follows.

1. The datasets were split into 75% training set and 25% test set via stratified random sampling.
2. The training sets were successively reduced by factor 1.2 via random sampling, stopping when the training set size was approximately equal to the test set size. Seven training sets were thus generated for each dataset, each one 83% the size of the previous one.
3. Each ensemble learning scheme was trained on all the training sets and tested on the same test set.

For each of our thirty runs, these training and test sets were precomputed, each time with a different split into training and test set. Over all runs, mean and standard deviation of test set accuracies was computed. Significant differences were considered to be non-overlapping accuracy confidence intervals[1] which gives an approximate confidence level of 95%.

For empirical evaluation we chose twenty-six datasets from the UCI Machine Learning Repository (Blake & Merz, 1998), shown in Table 3.2. These were used in all other experiments, throughout this thesis. We chose four ensemble learning schemes, including StackingC, plus bestBase.

- StackingC is the stacking algorithm which was introduced in (Seewald, 2002a) (extended version see Chapter 5), one of the best-performing Stacking variants to date.
- X-Val chooses the best base classifier on each fold by an internal ten-fold CV on the training data. Thus, for every point on the learning curve and each run, a different classifier may possibly be chosen.
- Voting is a straight-forward extension of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to the base classifier's estimate of the class probability distribution for the example. It is mainly included as a benchmark of the performance that could be obtained without resorting to the expensive crossvalidation of all other algorithms.
- Grading is the implementation of the grading algorithm evaluated in (Seewald & Fürnkranz, 2001). It uses a simple baseline learner as meta-level classifier which speeds up computation by an order of magnitude while still retaining its unique performance. See Chapter 7 for a more comprehensive discussion of this optimization.
- bestBase is the best base classifier by hindsight. For each dataset, only one best base classifier was chosen, based on the premise that there is one optimal classifier for each dataset, whose expertise matches the learning problem best independent of training set size. The best base classifier is considered the one of the four base learners whose performance, averaged over all runs, is nearest to the performance that could be achieved by choosing the maximum performance from all the base learners.

[1] i.e. the difference between accuracy means is larger than the sum of the standard deviations of both accuracies involved.

Table 6.1: This table shows pairwise significant differences between ensemble learning schemes as wins/losses. Notice that for each entry 182 comparisons were made so that all but one of these differences are well below the expected alpha error of 5% and should therefore not be considered significant.

Scheme	StackingC	X-Val	Voting	Grading	bestBase
StackingC		0/0	5/0	3/0	1/6
X-Val	0/0		4/5	3/3	1/15
Voting	0/5	5/4		0/1	0/8
Grading	0/3	3/3	1/0		0/7
bestBase	6/1	15/1	8/0	7/0	

Table 6.2: This table shows pairwise correlation coefficients between ensemble learning schemes, over the full learning curve.

Scheme	StackingC	X-Val	Voting	Grading	bestBase
StackingC	1.000	0.994	0.984	0.986	0.982
X-Val	0.994	1.000	0.979	0.983	0.981
Voting	0.984	0.979	1.000	0.997	0.990
Grading	0.986	0.983	0.997	1.000	0.991
bestBase	0.982	0.981	0.990	0.991	1.000

All ensemble learning schemes used the following four base learners, which we found to be the best-performing diverse set of our seven original learners, see Chapter 3.

- J48: a Java port of C4.5 Release 8 (Quinlan, 1993a)
- NaiveBayes: the Naive Bayes classifier using multiple kernel density estimation for continuous attributes.
- MLR: a multi-class learner which tries to separate each class from all other classes by linear regression (*multi-response linear regression*)
- KStar: the K* instance-based learner (Cleary & Trigg, 1995)

All algorithms are implemented in WEKA Release 3.1.8. Each of them returns a class probability distribution, i.e., they do not predict a single class, but give probability estimates for each possible class. Parameters for learning schemes which have not been mentioned were left at their default values.

6.2 Results and Discussion

The complete learning curves can be found in Figures 6.1 to 6.4. The performance of a single ten-fold crossvalidation on the complete dataset is also shown in each case. For more general results, we also computed significant differences (Table 6.1) and pairwise correlation coefficients (Table 6.2).

Table 6.1 shows the pairwise significant differences between ensemble learning schemes, including bestBase. Each entry was computed by determining significant differences in accuracy over all twenty-six datasets and seven points on the learning curve, yielding 182 comparisons. Although some significant differences are found, all but one of them are well below the expected alpha error of 5%, i.e. 9.1. Concerning the single remaining entry *1/15* (bestBase vs. X-Val), six losses are on a single dataset – if we remove them, we are just a little above alpha error. Although a plausible explanation exists – i.e. we would expect X-Val to perform worse in selecting the best base classifier than hindsight, since its decision is based on less available data – this result seems unreliable and may be anecdotal. If we look at the learning curves themselves, we find the reason for this absence of truly significant differences in the consistently high standard deviations over all datasets, amounts of training data and ensemble learning schemes. Table 6.2 shows the correlation coefficients between the classifiers, computed over all points of the learning curves and datasets. Here, we also note that all the ensemble learning schemes perform quite similarily. So we must conclude that we found no significant differences between ensemble learning schemes. Thus, a further investigation into our concept of stability is meaningless – all schemes are equally stable.

Still, close inspection of the learning curves shows that most schemes *do* show a graceful degration when trained on ever smaller and smaller training sets. However, as mentioned the variance over the runs is quite high, so it would make sense to combine ensemble learning schemes with a variance-reducing technique such as bagging or boosting to increase stability.

6.3 Related Research

We are not aware of any other investigations into the stability as defined here, neither of ensemble learning schemes nor basic classifiers.

6.4 Conclusion

In light of the presented evidence, we cannot support our original hypothesis that StackingC offers more stable performance than other ensemble learning schemes. On the contrary, we found almost no significant differences between our ensemble learning schemes, including the best base classifier by hindsight. Thus by our definition of stability any of the schemes we considered is a viable choice. It seems that evaluation via hold-out set and evaluation via ten-times ten-fold crossvalidated t-test – which is widely used in the machine learning community – give contradicting results. This observation agrees with currently unpublished results from B. Zenko who found that, when comparing ensemble learning schemes via a five times two-fold crossvalidated t-test (proposed by Dietterich (1998)), there are also almost no significant differences to be found. Dietterich (1998) also shows that a variant of the commonly used ten times ten-fold crossvalidated t-test has a high alpha error, i.e. is quite likely to find significant differences where there are none. So it is not inconceivable that these results hold in general.

On a more positive note, we found that even the simplest scheme Voting is competitive to the best base classifier by hindsight. Crossvalidation via X-Val is also competitive, but much more costly. Even though it may potentially choose a different classifier for each point and run of the learning curve, it does not perform significantly better than bestBase, which in retrospect supports our decision to choose only one best classifier for each dataset.

As mentioned the variance over the runs from our learning curves is quite high. Thus it would seem to make sense to combine ensemble learning schemes with a variance-reducing technique such as bagging or boosting.

On a final note, we believe that stability is a desirable property of classifiers and may be complementary to classic accuracy estimates. A more quantitative measure of classifier stability will be our next step.

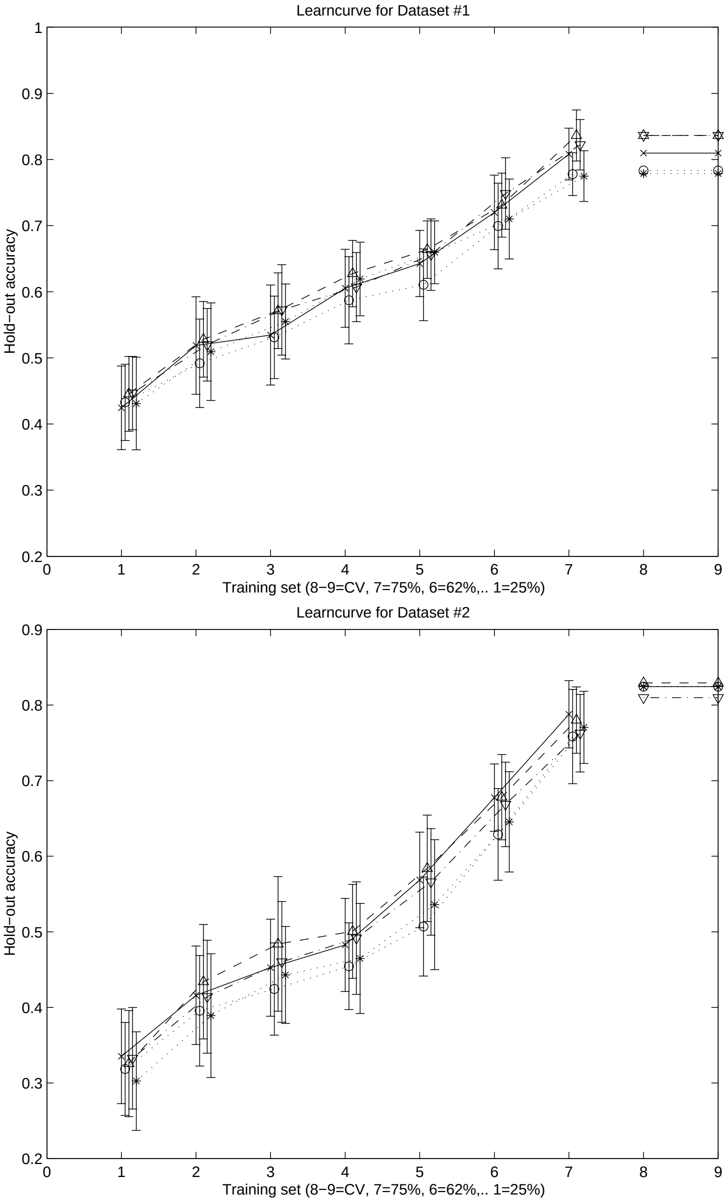


Figure 6.1: Learning curves for dataset *audiology* to *autos*. StackingC is the unbroken line with ×, X-Val is the dotted line with ∘, Voting is the dashed line with △, Grading is the dash-dotted line with ▽ and bestBase is the dotted line with ∗. The training set at x=1 is 25% and increases geometrically to 75% at x=7, see text. From 8-9, the crossvalidated accuracy estimate is shown for comparison. The schemes are shifted to the right from each point, in the given order, so that the confidence intervals can be more clearly observed. The dataset numbers were assigned in the order from Table 3.2, beginning at 1.

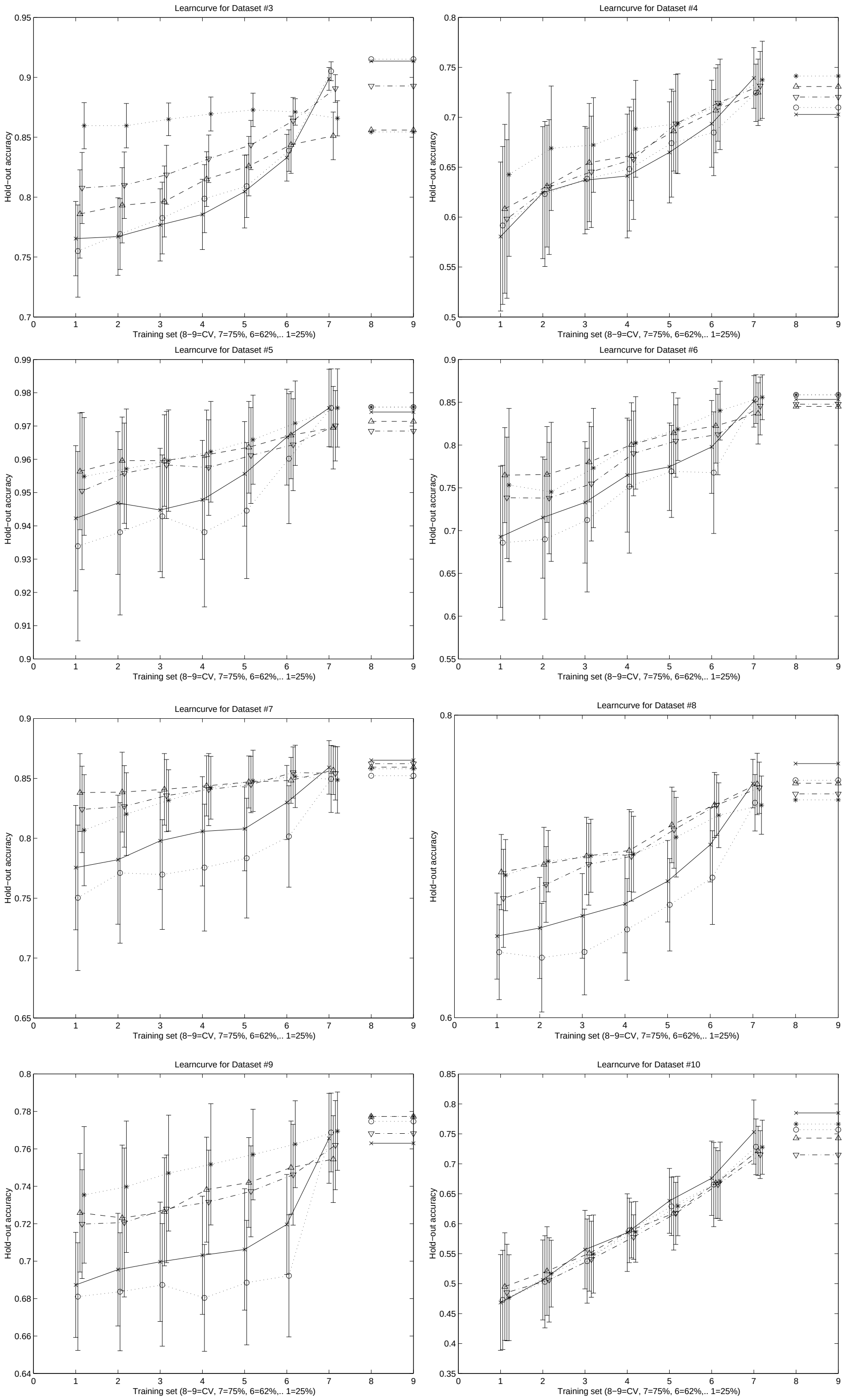


Figure 6.2: Learning curves for dataset *balance-scale* to *glass*.

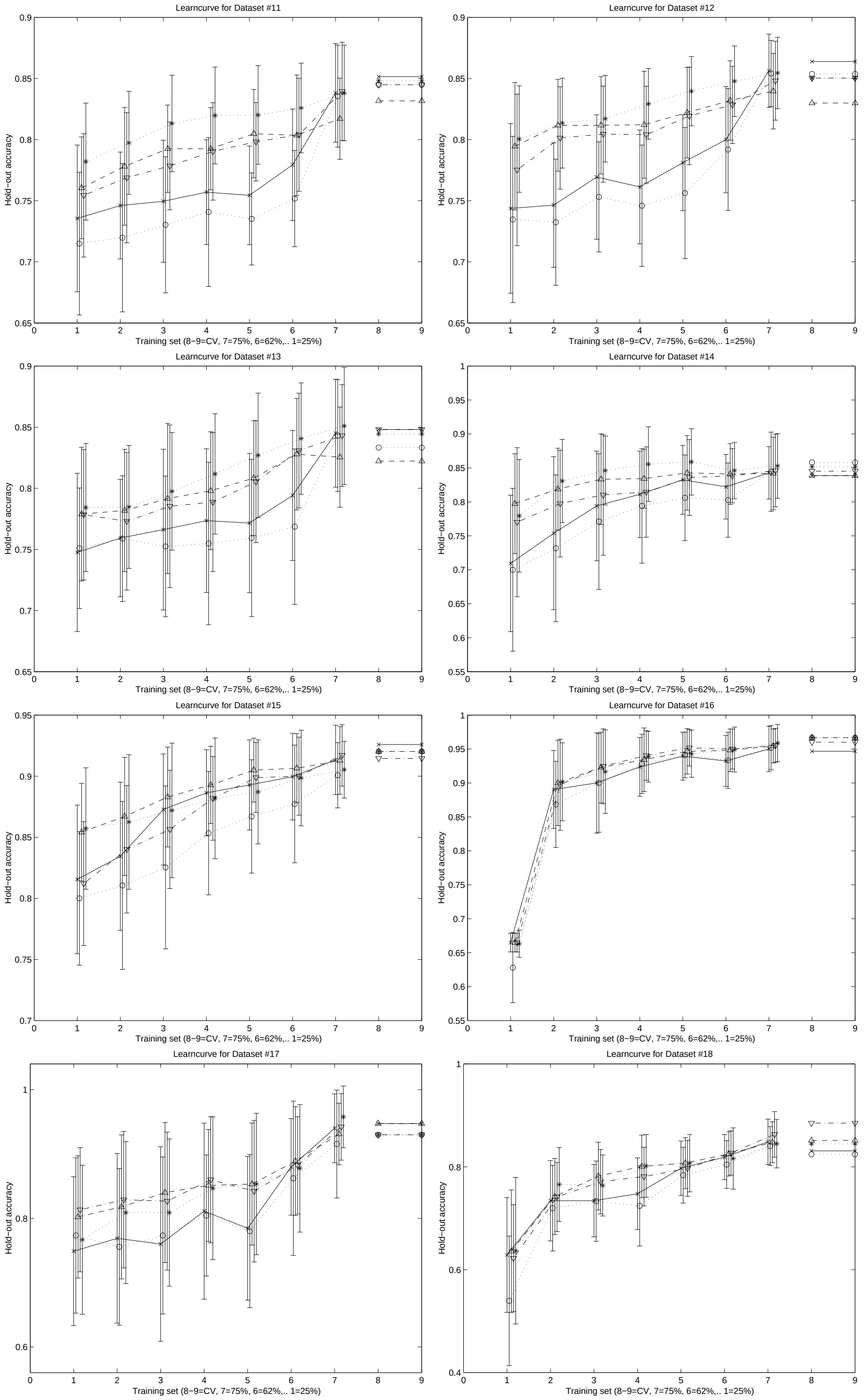


Figure 6.3: Learning curves for dataset *heart-c* to *lymph*.

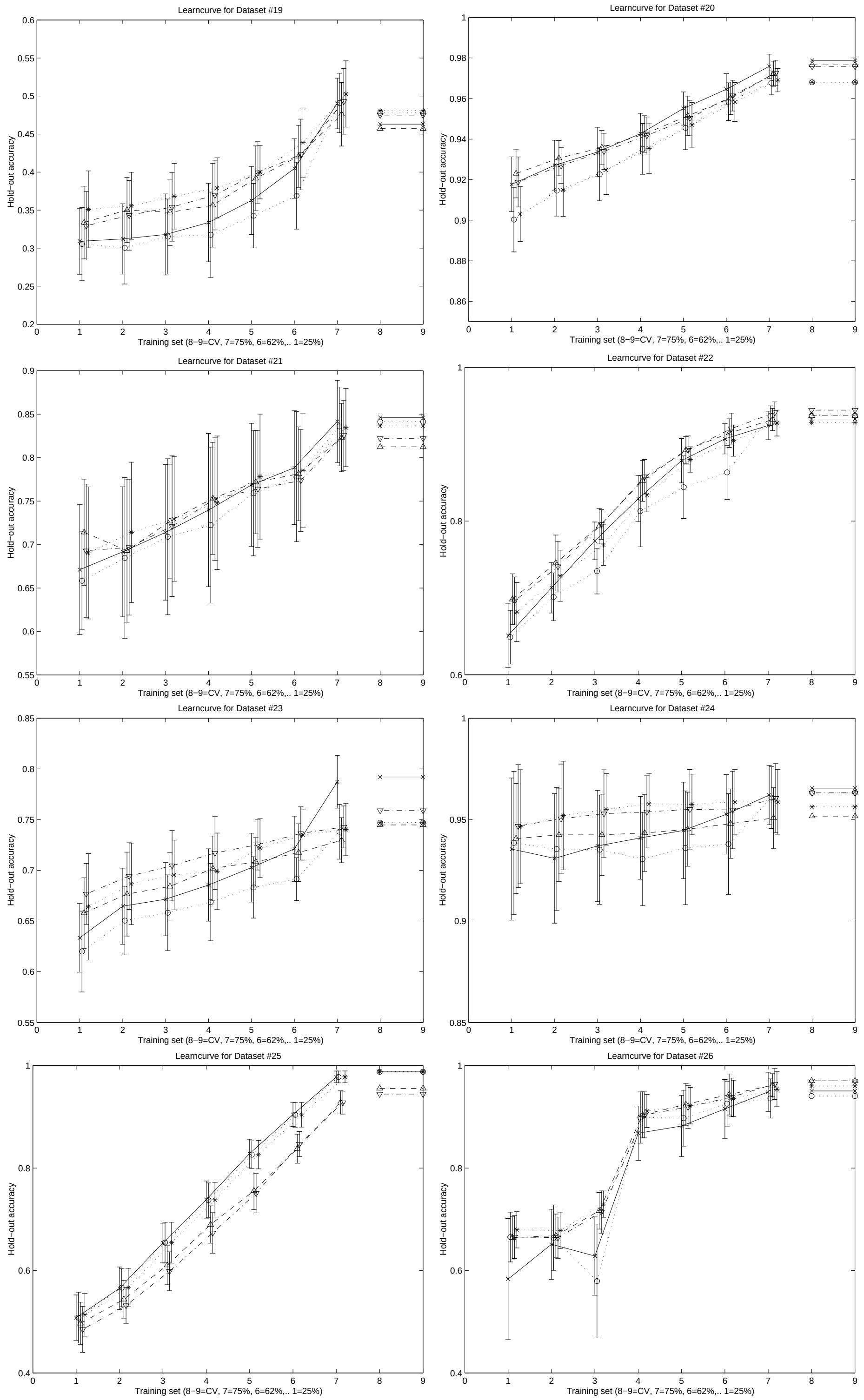


Figure 6.4: Learning curves for dataset *primary-tumor* to *zoo*.

Chapter 7

Towards a Theoretical Framework

In this chapter, we show that the ensemble learning scheme Stacking is universal in the sense that most ensemble learning schemes systems – Voting, *Selection by Crossvalidation* (X-Val), Grading and even Bagging – can be mapped onto Stacking via specialized meta classifiers. We present operational definitions of these meta classifiers for each scheme. In each case, running Stacking with the appropriate meta classifier simulates the respective ensemble learning scheme's operation perfectly, although – as with every simulation – the runtime performance may be worse.

For Grading (Seewald & Fürnkranz, 2001) which would ordinarily not be mappable, we show that an alternative parameter setting can transform it into a mappable form without sacrificing its unique performance.

Finally we show that, by additionally modifying the training set of the base classifiers with a simple wrapper, Bagging (Breiman, 1996) can also be simulated. Thus Stacking can be seen as a conceptual abstraction for most ensemble learning schemes.

7.1 Introduction

Let us recall a final time how Stacking works in order to lay the foundations for our functional definitions of meta classifiers later on. Figure 7.1 shows Stacking on a hypothetical dataset with three classes, n examples and N base classifiers. Figure 7.1(a) shows the original dataset. Each example consists of an attribute vector of fixed length, followed by a class value.

During training, all base classifiers are evaluated via crossvalidation on the original dataset. We follow the extension of (Ting & Witten, 1999) of using the base classifiers' class probabilities. Each classifier's output is therefore a class probability distribution for every example. Figure 7.1(b) shows how such a reasonable class probability distribution from a single classifier may look. The rows correspond to the examples from the original training set, see Figure 7.1(a).

The concatenated class probability distributions of all base classifiers in a fixed order, followed by the class value, forms the meta-level training set for Stacking's meta classifier, see Figure 7.1(c). After training the meta classifier,

$Attrs$	$Cl.$
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
⋮	⋮
$AttrVec_n$	a

(a) original training set

a	b	c
0.90	0.05	0.05
0.15	*0.70*	0.15
0.10	*0.80*	0.10
0.20	0.20	*0.60*
⋮	⋮	⋮
0.80	0.10	0.10

(b) sample class probability distribution

$Classifier_1$			$Classifier_2$				$Classifier_N$			
a	b	c	a	b	c		a	b	c	$class$
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	...	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	a
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	...	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	b
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	...	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	b
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	...	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	c
	⋮			⋮				⋮		⋮
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	...	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	a

(c) training set for Stacking's meta classifier

Figure 7.1: Illustration of Stacking on a dataset with three classes (a, b and c), n examples and N base classifiers. $P_{i,jk}$ refers to the probability given by classifier i for class j on example number k

the base classifiers are retrained on the complete training data.[1] Thus, Stacking's meta classifier is free to learn arbitrary complex models to predict the true class from the class probabilities of its base classifiers, making it clearly the most flexible ensemble learning scheme.

For testing, the base classifiers are queried for their class probability distributions. These form a meta-example for the meta classifier which outputs the final class prediction. Any meta classifier for Stacking must therefore learn a mapping from a vector of concatenated class probabilities to a predicted class value from training data, and later apply this learned mapping to a new vector.

7.1.1 Definitions

In this section we shall give some definitions for important concepts and terms. Without loss of generality let us assume that a fixed training dataset with n examples and k classes, and a single test instance[2], is given. N arbitrary base classifiers have already been cross-validated on this dataset, each yielding a prediction for all n examples, and have afterwards been retrained on the complete training dataset. We also assume that all base classifiers output class probability distributions, i.e. estimated probabilities for each class instead of deciding on a single class.

[1] Interestingly, *not* retraining the base classifiers usually yields slightly worse results.

[2] Since during testing, no learning takes place – i.e. the learned model of all classifiers remains constant – instances can be processed in arbitrary order. Thus, demonstrating equivalence for a single test instance is sufficient.

Then, P_{ijk} refers to the probability given by classifier i for class j on example number k during the internal cross-validation. Let us now denote $\overline{P}_{ik}$ to signify the complete class probability distribution of classifier i on instance k. If no k is given, $\overline{P}_i$ refers to the class probability distribution for classifier i on the unknown instance during testing. For completeness we also have to assume a fixed arbitrary order on the class values so that each class is at the same position in all $\overline{P}_{ik}$ considered, and a fixed arbitrary order on the N base classifiers. $Class_k$ denotes the true class for instance k from the training set. $AttrVec_k$ corresponds to the attribute vector of instance k. We consider all indices to be zero-based, e.g. the instance id k satisfies the equation $0 \leq k \leq n-1$.

$\delta(x, y)$ is the well-known delta function (7.1). In case the notation is not generally known, we also define arg max (7.2) to signify the first entry where the corresponding value is equal to the maximum. This allows to determine the predicted class from a given class probability distribution and also takes care of non-unique maximal values[3].

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} \tag{7.1}$$

$$\arg\max_i (A_i) \equiv \min\{i \mid A_i = \max_{\forall i} (A_i)\} \tag{7.2}$$

As we mentioned, we assume that all ensemble learning schemes return class probability distributions. If predictions are needed, the position of the maximum class probability in the vector – i.e. the predicted class – is easily obtained via formula (7.2)

We can now characterize every ensemble learning scheme by what features it extracts from the meta dataset during training and how these features define the mapping from meta dataset to final class probability distribution during testing. Thus, each meta classifier which simulates the corresponding ensemble learning scheme can be defined by two characteristic functions: One, the features which are extracted from the meta dataset and how they are computed. Two, how these features are used during testing to determine the final class probability distribution.

7.2 Mapping Ensemble Learning Schemes

We show that Stacking using class probability distributions is equivalent to the following ensemble learning schemes, given appropriate meta classifiers.

- Stacking is the stacking algorithm as implemented in WEKA, which follows (Ting & Witten, 1999). It constructs the meta dataset by adding the entire predicted class probability distribution instead of only the most likely class. Trivially, Stacking with predictions can also be simulated by transforming the class distributions meta dataset to predictions prior to applying the meta classifier via Formula (7.2)

- X-Val chooses the best base classifier on each fold by an internal ten-fold CV. This is also known as *Selection by Crossvalidation*, a widely used ensemble technique in machine learning.

[3] In that case, choosing the more common class is also a reasonable alternative

- Voting is a straight-forward extension of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to the base classifier's estimate of the class probability distribution for the example. I.e. the mean class distribution of all classifiers is calculated. It is the only scheme which does not use an internal crossvalidation. We also show that more common voting of predictions can be simulated by Stacking.

- Grading is the grading algorithm (Seewald & Fürnkranz, 2001). Basically, Grading trains one meta classifier for each base classifier which tries to predict when its associated base classifier fails. This decision is based on the original attributes from the dataset. A weighted voting of the base classifiers prediction gives the final class prediction. The confidence for a correct prediction of a base classifier, which is estimated by its associated meta classifier, is used as weight.

- Bagging (Breiman, 1996) is another common ensemble technique. Here, the same type of classifier is repeatedly trained on new datasets, which have been generated from the original dataset via random sampling with replacement. Afterwards, the component classifiers are combined via simple unweighted voting.

We shall proceed to show how every one of them can be simulated by Stacking with an appropriate meta classifier. We give functional descriptions of the mapping from meta dataset to features during training and from features to final prediction during testing. For Bagging, a simple wrapper is necessary around each base classifier, which simulates random sampling with replacement.

7.2.1 Voting

Voting is the simplest case. During training, no features are extracted from the meta dataset. In fact Voting does not even need the internal crossvalidation. Since after training the base classifiers are retrained on the complete training set, the base classifiers are then equivalent to those normally used in Voting.

During testing, Voting determines the final class probability distribution as follows, i.e. as mean class probability distribution for the current unknown instance.

$$\overline{pred} = \sum_{i=1}^{N} \frac{\overline{P}_i}{N} \tag{7.3}$$

Thus, it can be easily seen that the meta classifier defined by just computing the mean probability distribution of the base classifiers – as above – simulates Voting with probability distributions.

Voting with predictions can be mapped similarily. In this case, we use $\overline{P'}_i$ instead of $\overline{P}_i$ in Formula (7.3). $\overline{P'}_i$ is the vector of P'_{ij} for all j, where

$$P'_{ij} = \begin{cases} 1 & \text{if } j = \arg\max_j (P_{ij}), \text{ for given } i \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

In essence, this simplifies the class probability distribution to a vector of zeros with just a one where the most probable class was earlier. Summing over these simplified class probability distributions is clearly equivalent to counting the number of votes per class over all classifiers. Again, the class with the highest number of votes is chosen as final prediction. Concluding, we have shown Stacking to be equivalent to Voting in either variant, using the proposed meta classifier.

7.2.2 X-Val

For X-Val, we first determine the accuracy per classifiers as estimated by the internal crossvalidation, which can be computed directly from the meta-level dataset, see (7.6). Then, we derive as feature the id of the classifier with the maximum accuracy. Thus, the value of *bestC* corresponds to our learned model.

$$bestC = \arg\max_i \left(Acc_i\right) \tag{7.5}$$

where

$$Acc_i = \frac{1}{n}\sum_{k=1}^{n} \delta(\arg\max_j \left(P_{ijk}\right), Class_k) \tag{7.6}$$

During testing, X-Val simply returns the distribution from best base classifier *bestC*.

$$\overline{pred} = \overline{P}_{bestC} \tag{7.7}$$

7.2.3 Grading

For Grading, the case is quite difficult. Since Grading's meta classifiers[4] base their model on the original dataset's attributes, at first glance it seems to be impossible to map it onto Stacking as specialized meta classifier – at least without utilizing bi-level stacking (Schaffer, 1994).

However, during an evaluation of Grading we noted that there is very little difference between meta classifiers, always less than 1%, see Table 7.1. This was also found by the original authros, see the technical report of Seewald & Fürnkranz (2001). This puzzled us for some time and eventually prompted us to run our own experiments using a baseline learner as meta classifier which always outputs a fixed probability distribution based only on the most common class. The idea was to find out how much of the performance gain is due to the combining scheme and thus independent of the meta classifier.

We were surprised to note that this trivial meta classifier, ZeroR, is competitive to all other meta classifiers we evaluated and even once outperforms them all, see Table 7.1. So it is a reasonable alternative meta classifier to IBk with ten nearest neighbors, which was used in (Seewald & Fürnkranz, 2001) and we propose it for further experiments.

What does this alternate meta classifier mean for Grading? Basically, Grading does not *grade* – it works solely because reasonable meta classifiers will be as good as the baseline accuracy, while getting better than that seems to be extremely hard. Based on Grading's combining scheme, the predictions are

[4] Note that both Stacking and Grading have the parameter *meta classifier.* While Stacking has a single meta classifier, there is one for each base classifier in Grading– all of the same type.

weighted by $p(+)$, so in this setting Grading is essentially equivalent to accuracy-weighted voting of the base classifiers predictions, where the accuracy is estimated via cross-validation.[5]

Given our proposed new meta classifier, it is now possible to map Grading onto Stacking. During training, the accuracies of base classifiers are extracted using Formula (7.6). The accuracies of all our base classifiers correspond to our learned model. During testing, we build the combined class probability distribution similar to Voting using predictions but with an additional weight – namely the accuracy we extracted earlier.

$$\overline{pred} = \frac{1}{\sum_{i=1}^{N} Acc_i} \sum_{i=1}^{N} \left[Acc_i \frac{\overline{P'}_i}{N} \right] \tag{7.8}$$

where $\overline{P'}_i$ is again the vector of P'_{ij} for all j. P'_{ij} is taken from Formula (7.4).

A straightforward extension of this which we have not yet evaluated is accuracy-weighted voting of base classifier's class probability distributions, which is given in the following Formula.

$$\overline{pred} = \frac{1}{\sum_{i=1}^{N} Acc_i} \sum_{i=1}^{N} \left[Acc_i \frac{\overline{P}_i}{N} \right] \tag{7.9}$$

Thus, we have shown that Stacking can simulate Grading without sacrificing its unique performance.

7.2.4 Bagging

For Bagging, the same meta classifier as for Voting with predictions is used. The number of base classifiers is equal to the iteration parameter of bagging – each base classifier for Stacking corresponds to one instantiation of the base learner for bagging. In order to simulate the random sampling from the training set, the base learner's training sets have to be modified before training, via Formula (7.10).

$$\begin{aligned} newTrain &= \{[AttrVec_{i_j}, Class_{i_j}] | i_j = rand(0,n), \forall\ 0 \leq j < n\} \quad (7.10) \\ rand(a,b) &\ \ldots\ \text{generates integer random numbers from interval [a,b)} \end{aligned}$$

During training, Formula (7.10) is used to create – for each base classifier separately – a training set of the same size as the original training set via random sampling from the original training set, exactly as in Bagging. These training sets are then used to train the base classifiers. This approach can also be seen

[5] The meta datasets for Grading are different for each base classifier and consist of the original attributes, followed by a binary attribute which encodes whether the base classifier did (+) or did not (-) correctly predict the class of the respective instance during the cross-validation. The class distribution of this meta dataset is directly related to the cross-validated accuracy of its base learner, i.e. $p(+) = Acc$ and $p(-) = 1 - Acc$ – so the better the learner performs, the more unbiased the class distribution becomes. Under these circumstances it is not unreasonable to expect this to be a very hard learning task, which can seldom be solved better than the baseline of always predicting +. Based on the reasonable assumption that all classifiers have an accuracy of at least 50%, the most common class will be + and then Grading will be equivalent to accuracy-weighted voting.

as a probabilistic wrapper around each base classifier. No features are extracted from the meta-level dataset during training, as for Voting.

During testing, each base classifier gives a prediction. These predictions are then voted to yield the final prediction, exactly as for Voting with predictions, i.e. (7.3) modified via (7.4) – for more details refer to subsection 7.2.1. Concluding, we have shown the equivalence of Bagging and Stacking.

7.2.5 Others

By definition StackingC, see Chapter 5, can also be mapped onto Stacking via a specialized meta classifier. In fact, the current implementation is a subclass of a common meta classifier, MLR. Another recent variant, sMM5 (Džeroski & Zenko, 2002), is also implemented via a special meta classifier and thus amenable to the same kind of mapping. Therefore, Stacking is also equivalent to both of these new variants, given appropriate meta classifiers.

However, AdaBoost (Freund & Schapire, 1996) cannot be simulated by Stacking because of its mainly sequential nature.[6]

7.3 Experimental Issues

While the given formal definitions of meta classifiers are mainly intended to enable further theoretical work, a direct implementation is also feasible. On the assumption that our models are correct, an implementation could serve to investigate runtime performance, i.e. the cost penalty for the simulation. However, since training the meta classifier usually contributes little to the total runtime – the main cost is due to the internal cross-validation of all base classifiers – we consider this unnecessary. In most cases, the expected runtime cost of the simulation is expected to be comparable to that of the original system. In case of Grading it is even expected to be slightly less, since our proposed meta classifier is much faster than the original classifier proposed in (Seewald & Fürnkranz, 2001). Only for Voting, which does not need the internal cross-validation at all, the runtime cost of the simulation is expected to be about one order of magnitude higher. We believe that the advantage of having a comprehensive description of all these schemes within a single framework outweighs this cost penalty for Voting.

[6] Or, to be more precise: While its training phase could potentially be simulated, using bi-level Stacking (Schaffer, 1994), replacing internal cross-validation estimates with training set performance estimates and utilizing multi-level vertical stacking – i.e. putting each classifier on top of the last one: one level for each iteration – and some elaborate wrappers between adjacent levels, its testing phase can regrettably *not* be simulated. AdaBoost computes a weighted vote of its component classifiers, whereas in multi-level Stacking the predictions of the classifiers are propagated upwards, beginning at the lowest level, and are thus processed by each component classifier in turn. So, in order to simulate AdaBoost, we would have to either modify the component classifiers as well, or change the basic structure of Stacking. We opted to leave this problem open until a more complete understanding of combining methods offers a simpler approach.

7.4 Related Research

To our knowledge, there is no related research concerned with the theoretical equivalence or practical simulation of ensemble learning schemes. The conceptual closeness of most ensemble learning schemes is of course no surprise; however, we seem to have been the first to formalize this closeness towards achieving a general theoretical framework.

7.5 Conclusion

We have shown that Stacking is equivalent to most ensemble learning schemes, namely *Selection by Crossvalidation* (X-Val), Voting of either class probability distributions or predictions, and Grading. We have given functional descriptions of suitable meta classifiers for Stacking which simulate the operation of these ensemble learning schemes. By a simple wrapper we were also able to simulate Bagging. Recent variants such as StackingC(Seewald, 2002a), see also Chapter 5 and sMM5 (Džeroski & Zenko, 2002) can also be simulated in the same way. So all these schemes can essentially be reduced to Stacking with an appropriate combining scheme. Thus we conclude that our approach offers a unique viewpoint on Stacking which is an important step towards a theoretical framework for ensemble learning.

This endeavour also allows to reformulate the choice of best ensemble learning system as best meta classifier choice. As such, it is potentially amenable to an approach using pre-estimated meta data to choose between meta classifiers, which we initially tested for our experiments from Chapter 3.

Another possible venue for future research may be to build tailored meta classifiers for specific problems, using the definitions of other ensemble learning schemes as background knowledge to guide the search process. Research into alternative meta classifiers for Stacking seems also a reasonable course, given that two recent variants (StackingC, sMM5) have been successful in this area using quite simple approaches.

Table 7.1: Grading with different level 1 classifiers. The first column shows the accuracy of IBk(originally used in [Seewald and Fuernkranz, 2001]), all other columns show accuracy ratios for the respective meta-learners ($\frac{Acc(\mathsf{Meta}_i)}{Acc(\mathsf{IBk})}$). The last column shows the results for our baseline learner, ZeroR. Average accuracy and standard deviation per column are also shown.

Dataset	IBk	DecisionTable	J48	KernelDensity	KStar	MLR	NaiveBayes	ZeroR
audiology	84.51	0.990	0.990	0.979	1.011	0.990	1.005	1.000
autos	81.95	1.006	1.006	0.964	1.000	0.964	1.006	1.030
balance-scale	89.76	1.000	0.998	1.000	1.002	1.000	1.000	1.000
breast-cancer	74.48	1.000	0.986	1.005	1.009	1.009	1.000	0.991
breast-w	96.57	1.000	1.002	0.997	1.000	0.999	1.002	0.998
colic	84.78	0.994	1.003	0.997	1.003	0.984	0.997	1.000
credit-a	86.09	0.992	0.993	0.995	0.997	0.985	0.992	0.882
credit-g	75.90	0.993	0.990	1.000	0.983	0.979	0.988	1.134
diabetes	76.30	1.009	1.005	1.005	1.009	1.003	1.002	1.009
glass	73.36	1.019	1.000	1.026	1.026	1.038	1.013	1.032
heart-c	85.48	0.973	0.969	0.973	0.981	0.965	0.985	0.988
heart-h	83.33	1.012	1.012	0.976	0.984	1.004	1.004	1.008
heart-statlog	83.70	0.996	0.982	0.960	0.974	0.982	0.987	0.996
hepatitis	81.94	1.008	1.024	1.032	1.016	1.016	1.047	1.032
ionosphere	91.17	1.006	1.006	1.000	1.000	1.006	1.010	1.009
iris	95.33	1.000	1.000	1.007	1.000	1.007	1.000	1.000
labor	94.74	1.000	1.000	1.000	1.000	1.000	1.000	1.000
lymph	84.46	1.016	1.024	0.984	1.000	1.024	1.008	1.032
primary-tumor	49.26	0.964	0.922	0.892	0.964	0.964	1.000	0.958
segment	98.10	1.000	0.999	0.997	0.995	0.997	0.997	1.000
sonar	86.06	0.972	0.944	0.989	0.978	0.967	0.983	0.972
soybean	94.00	0.997	1.000	1.000	0.975	0.9984	1.003	0.998
vehicle	74.47	1.025	1.011	0.989	0.981	1.033	1.025	1.019
vote	96.09	0.998	1.000	0.998	0.998	0.995	0.995	1.000
vowel	98.79	0.997	1.001	0.997	0.998	0.994	0.998	0.991
zoo	97.03	1.000	1.000	1.010	0.990	0.990	1.010	1.000
Avg	85.29	0.999	0.995	0.991	0.995	0.996	1.002	1.003
±	10.81	0.014	0.022	0.026	0.014	0.020	0.013	0.040

Chapter 8

Information Visualization

In this chapter we will shortly introduce Information Visualization, which is a fairly new field that deals with visualization of high-dimensional data where no inherent spatial or temporal structure is present. We will then apply methods from this field to visualize all our datasets, noticing some irregularities and inconsistencies which we will discuss on a case-by-case basis. Afterwards we will visualize our experimental results for the best variant, StackingC from Chapter 5, based on the dataset visualization, and also a compressed representation. Icons are used to encode classifier performance on distinct instances. Afterwards, we briefly mention related research and conclude this chapter with an outlook on future applications of Information Visualization in Machine Learning.

8.1 Introduction

Within the scope of this thesis, we can only hope to give a short introduction to Information Visualization. (Card, Mackinlay & Shneiderman, 1999) offers a more comprehensive overview of this field.

In Information Visualization, contrary to classical Scientific Visualization, there is usually no inherent spatial or temporal structure present in the data. Information visualization deals with heterogenous, high-dimensional data, so the use of appropriate visual metaphors is necessary. The interaction with the user in context of real-time explorative data analysis offers the highest benefits – however, within this thesis we cannot offer any user interaction, so our choice of methods is more limited.

In Information Visualization literature, five general techniques can be discerned (Keim & Kriegel, 1996)

- **Geometric** – e.g. scatterplots and parallel coordinates.
- **Icon-based** – e.g. chernoff faces, stick figures and the glyphs we shall use later.
- **Pixel-based** – e.g. recursive pattern techniques and circle segments.
- **Hierarchical** – e.g. cone/cam trees and treemaps.
- **Graph-based** – e.g. polylines and curved lines.

Aonther dimension to be considered is interaction techniques. However, in our case no interaction with the user can take place, so within the scope of this thesis this is not a relevant dimension.

Within standard machine learning literature, simple graph-based techniques are most commonly used – e.g. learning curves, scatterplots, bar charts and line charts. Parallel coordinates are rarely used, possibly because this technique works best in an interactive setting. Hierarchical and graph-based visualization methods are also rarely used. Pixel-based techniques, while being the most dense methods – i.e. able to visualize the highest amount of data on a given area – are also the hardest to grasp.

We believe that both geometric and icon-based techniques show the greatest future potential for visualizing both experimental results and the respective datasets, and also for interactive exploration of results. Still, almost all journals and conference proceedings are printed in black and white; color pages are – if at all possible – very expensive and sometimes even high-resolution graphics (e.g. photos) can pose a problem. Under these circumstances it is understandable that most machine learning literature restricts itself to very simple visualization methods.

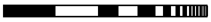
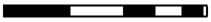
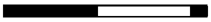
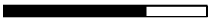
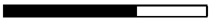
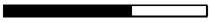
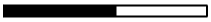
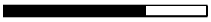
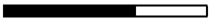
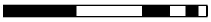
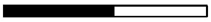
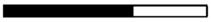
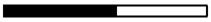
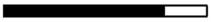
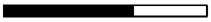
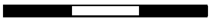
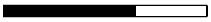
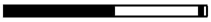
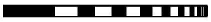
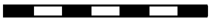
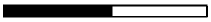
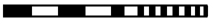
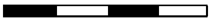
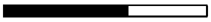
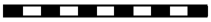
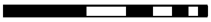
Of these methods, the hierarchical technique is not applicable to our data since there is no obvious hierarchical structure. The graph-based technique is also not applicable since we deal with independent examples – there are generally no meaningful relations between instances. Geometric methods have been used throughout this thesis and will also be used here, mostly in the form of line charts and scatterplots. Since all our datasets are small enough for alternative visualization methods, the additional denseness of pixel-based methods offers no significant advantages. Thus, we found icon-based methods most inspiring. So we will focus mainly on geometric and icon-based methods in the remainder of this chapter.

8.2 Visualizing Instance Space

In this section we take a close look at the twenty-six datasets which we chose for all experimental evaluations in this thesis, see Table 8.1. This section may seem somewhat of an afterthought – at first glance it would have made more sense to know our data *before* running extensive experiments. However, as proponents of *tabula rasa* machine learning, we treated our datasets as black boxes until now, in order to prevent even the temptation to contaminate our results, e.g. by parameter tuning, recoding of attributes, removing inconsistent instances or even removing complete datasets. We were mainly interested in those kinds of insight in Stacking which are *independent* of the datasets in question; thus it is clear that such an analysis should only be made after all experimental results are obtained.

For us, instance space is the multidimensional vector space of all possible examples which can potentially be represented in a given dataset. Each example, or instance, is thus a point in this usually high-dimensional space, where each dimension is either continuous (for numeric attribute) or discrete (for nominal attributes)

Table 8.1: The used datasets with number of classes and examples, discrete and continuous attributes, baseline accuracy (%) and entropy in bits per example (Kononenko & Bratko, 1991). Class frequencies are shown in descending order; the length of every black and white bar determines one class frequency each.

Dataset	cl	Inst	disc	cont	bL	E	class frequencies
audiology	24	226	69	0	25.22	3.51	
autos	7	205	10	16	32.68	2.29	
balance-scale	3	625	0	4	45.76	1.32	
breast-cancer	2	286	10	0	70.28	0.88	
breast-w	2	699	0	9	65.52	0.93	
colic	2	368	16	7	63.04	0.95	
credit-a	2	690	9	6	55.51	0.99	
credit-g	2	1000	13	7	70.00	0.88	
diabetes	2	768	0	8	65.10	0.93	
glass	7	214	0	9	35.51	2.19	
heart-c	5	303	7	6	54.46	1.01	
heart-h	5	294	7	6	63.95	0.96	
heart-statlog	2	270	0	13	55.56	0.99	
hepatitis	2	155	13	6	79.35	0.74	
ionosphere	2	351	0	34	64.10	0.94	
iris	3	150	0	4	33.33	1.58	
labor	2	57	8	8	64.91	0.94	
lymph	4	148	15	3	54.73	1.24	
primary-t.	22	339	17	0	24.78	3.68	
segment	7	2310	0	19	14.29	2.81	
sonar	2	208	0	60	53.37	1.00	
soybean	19	683	35	0	13.47	3.84	
vehicle	4	846	0	18	25.41	2.00	
vote	2	435	16	0	61.38	0.96	
vowel	11	990	3	10	9.09	3.46	
zoo	7	101	16	2	40.59	2.41	

For visualization of the instance space, we chose Sammon Mapping (Sammon, 1969; Borg & Groenen, 1997)[1], an iterative gradient-descent method which tries to reduce the stress in a mapping from high-dimensional to low-dimensional data. Missing values have been previously replaced by the mean resp. the mode of the full dataset. Nominal values were coded via 1-of-n coding. Numeric values were normalized to be within the interval [0,1] for all values. We used the recoded attributes without the class as input for Sammon Mapping and added the class afterwards. So, any apparent correlations between visible clusters and classes is present in the euclidean distance of examples and may indicate that an instance-based classifier is most appropriate for the respective dataset, as Sammon Mapping aims to preserve the euclidean distances between instances.

We rejected Principal Components Analysis after initial experiments because its linear projection on orthogonal axes is less general and the representation is noticeably worse for some datasets, e.g. *soybean*. We rejected self-organizing

[1] We used the Sammon Mapping implementation from Vesanto, Himberg, Alhoniemi & Parhankangas (2000), which was written in MatLab script language. It worked quite well for almost all datasets, but in two cases we had to reduce the learning rate significantly to obtain convergence.

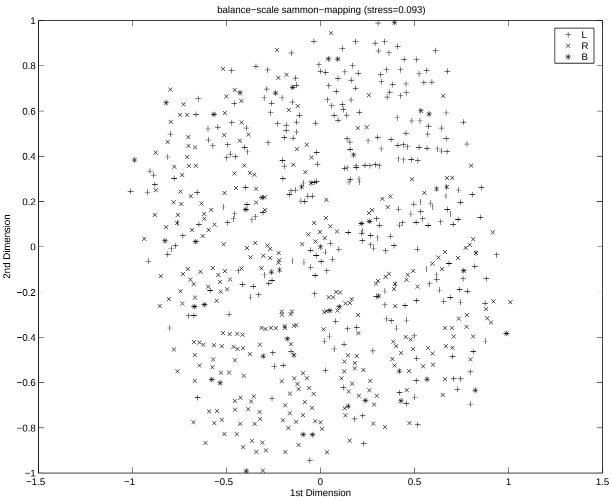


Figure 8.1: Instance space visualization of *balance-scale* via Sammon-Mapping. The regular structure of this visualization is striking. We found out that this is due to the fact that the input is exhaustive: for all four variables, all values from zero to one in steps on 0.25 appear. This is actually a synthetic dataset, with a non-linear perfect model and thus of questionable use in empirical experiments. Adding appropriate attributes makes the problem trivial to solve.

maps (SOMs), because Flexer (2001) showed that SOMs perform significantly worse in terms of quantization error, in recovering the structure of clusters and in preserving the topology of the multidimensional dataset. Also, Bezdek & Nikhil (1995) compared SOMs to principal components analysis and sammon mapping and show that it performs worse than either of the classical methods in the context of multidimensional scaling.

Visualization of the complete instance space via parallel coordinates was also rejected, mainly because high overlap between instances made it all but impossible to achieve a reasonable representation which shows all the examples and their classes clearly. However, for interactive exploration the parallel coordinates approach could have been quite useful. Appendix A shows all visualizations from this and the next section as full-page figures. Selected figures are duplicated here in smaller size for faster access. Observations are mentioned in the respective figures' captions, but for conciseness we will now summarize and interpret our findings.

1. We found a mismatch between reported number of classes – between the dataset documentation and those classes which really appear in the dataset – in five cases. In three of them, only one class does not appear. Since all these datasets have at least seven classes, this should be of marginal significance. However in two cases, *heart-c* and *heart-h*, the number of reported classes is five but the actual classes are two[2] – a much larger

[2] The true number of classes can e.g. be found out by counting the number of entries in the legend of the respective figures. It would in principle be possible that one or more classes

gap, possibly caused by a translation error to the ARFF format[3]. Thus, in Chapter 5, when we compared datasets with two- and more than two classes, these two datasets ended up on the wrong side of the comparison! As we have verified, this changes our experimental results only slightly and leaves the main conclusions intact, but still remains somewhat unsettling.

2. For quite some datasets with many classes, there are very few examples for the least-frequent classes – in some cases only *one*! Clearly, one example is not sufficient to learn anything, but even two examples are insufficient in most if not all cases. This has motivated us to visualize actual class distributions in Table 8.1, which convey this additional information. Mismatches between reported and actual classes (1) are also easily recognizable then, at least in the case of gross differences.

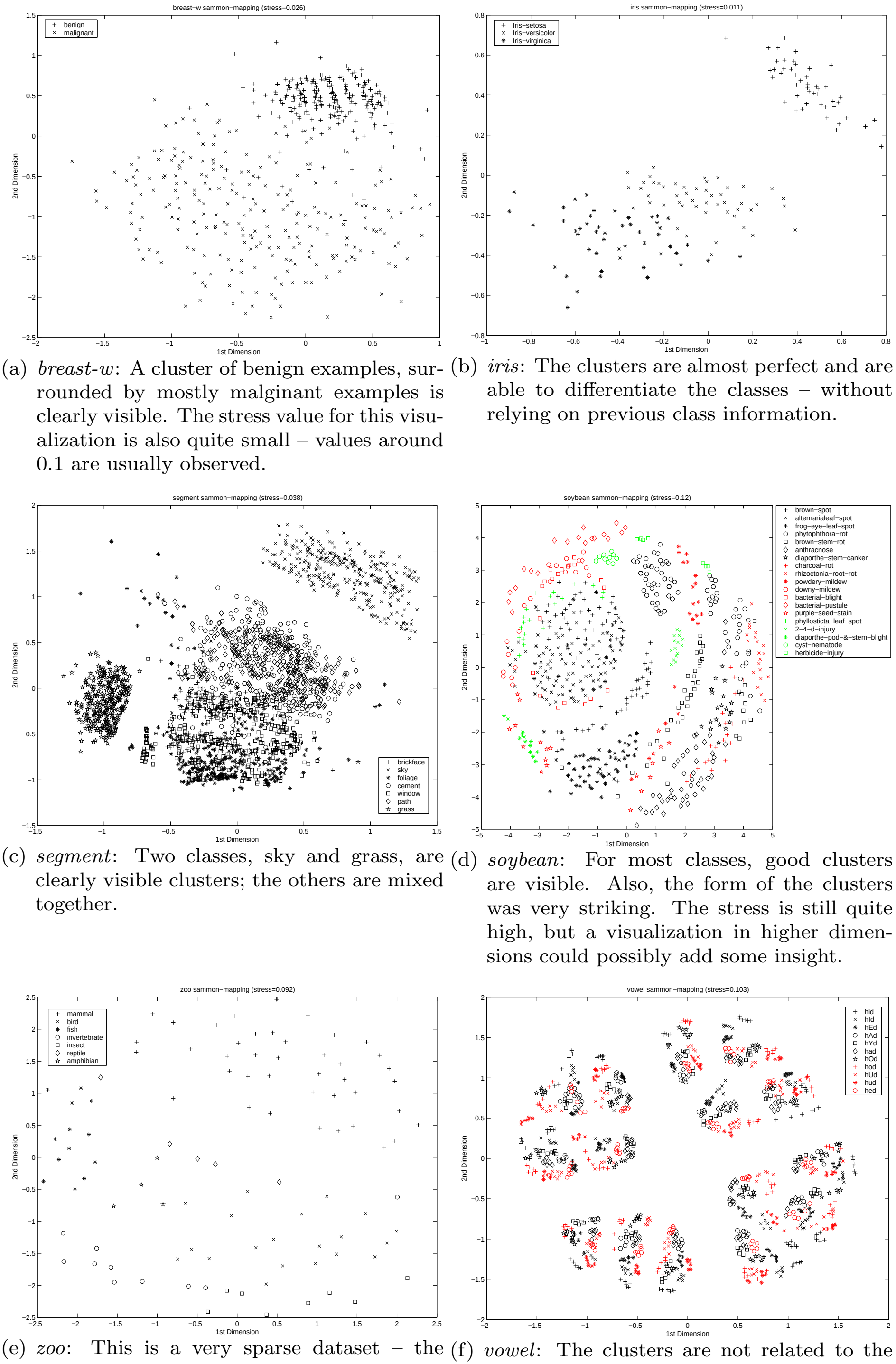
3. We found the regular structure of *balance-scale* (Fig.8.1) striking. By close inspection, we found out that this is due to the fact that the input is exhaustive: for all four variables, all values from zero to one in steps on 0.25 appear. When investigating the documentation, we found out that this is a purely synthetic dataset without noise, based on a non-linear model of a balance scale. Adding appropriate derived attributes could make the problem trivial to solve. This dataset is of questionable use for empirical studies since it is clearly *not* a real-life dataset.

4. For some datasets, Sammon Mapping shows some visible clusters corresponding to classes, mostly *breast-w*, *iris*, *segment*, *soybean* and *zoo* (Fig.8.2(a)-8.2(e)). Notice that this happended although the classes were not part of the input to the technique. In another case, *vowel* (Fig.8.2(f), the shown clusters correspond almost perfectly to the fifteen different speakers which were asked to pronounce vowels. This confirms what is widely known in the natural language community, namely the high variability of the speech signal between speakers which translates to large distances in instance space and thus also in the Sammon Mapping. Contrary to expectations we voiced earlier, we were not able to observe an advantage for instance based classifiers on the mentioned datasets – rather, the best classifiers for these datasets are almost uniformly distributed.

Concluding, Sammon Mapping seems to be a good means to get more insight into the structure of a given dataset. Although this visualization technique works best with continuous data, we noted that appropriate recoding of nominal attributes also yields good results. If we were concerned with real-life applications, we would now follow up on some of these interesting visual structures we observed. However, since this is an empirical study, we will now turn towards visualizing our experimental results – concentrating on those of the best variant, StackingC, and its components, the base classifiers.

overlap and thus some classes are hidden in the graph, although this has not been observed here. In any case, the legend always shows all non-empty classes.

[3] Both datasets were taken from the homepage of the WEKA machine learning group, `www.cs.waikato.ac.nz/~ml`, dataset-UCI.jar.



(a) *breast-w*: A cluster of benign examples, surrounded by mostly malginant examples is clearly visible. The stress value for this visualization is also quite small – values around 0.1 are usually observed.

(b) *iris*: The clusters are almost perfect and are able to differentiate the classes – without relying on previous class information.

(c) *segment*: Two classes, sky and grass, are clearly visible clusters; the others are mixed together.

(d) *soybean*: For most classes, good clusters are visible. Also, the form of the clusters was very striking. The stress is still quite high, but a visualization in higher dimensions could possibly add some insight.

(e) *zoo*: This is a very sparse dataset – the second-smallest one – however the clusters are quite good.

(f) *vowel*: The clusters are not related to the classes in this case, but correspond to the fifteen different speakers.

Figure 8.2: These subfigures show all datasets with striking clusters.

8.3 Visualizing Prediction Space

Instance space only depends on the dataset in question. However, prediction space as the space of possible predictions for instances includes not only the prediction of the respective scheme, StackingC, but also of its components, the base classifiers. All other parameters for the learner or its components would also be parts of prediction space, since these may influence predictive performance. In our case there are no additional parameters since we left all base classifiers at their default values. As simplification we chose to only consider two values for each prediction: correct or incorrect, compared to the true class. While inter-class differences may be worthwhile to study, in our case it would increase the information to be conveyed by an order of magnitude. Furthermore, the small size of most minor classes makes observation difficult. At last the independence of prediction space from the number of classes allows us to generate an overall result by averaging over all datasets, which would not be possible otherwise.

A feasible alternative would have been to use *InfoCrystal* (Spoerri, 1995), which is a generalization of classic Venn diagrams – able to show boolean relationships between an arbitrary number of sets, where Venn diagrams are only useful for up to three sets. However, no free implementation is available from the author.

We define prediction space as the vector space of possible predictions, for all four base classifiers and StackingC itself. Thus, each instance can be mapped onto a point in this five-dimensional space by determining for each dimension, i.e. scheme, whether or not it predicted the correct class for the mentioned instance, at the point where the instance was part of the crossvalidation test fold.[4] Since each dimension can take on only two values, this vector space is quite small – only 32 unique points in our case, so many instances will be mapped to the same point in prediction space[5]. We investigated two approaches to deal with this limitation.

1. We extended prediction space with instance space. Thus, each instance again gets a sufficiently unique location, which we can visualize via Sammon Mapping. We defined simple glyphs to capture the information from the original prediction space, see left side of Figure 8.3. These results can be found in Appendix A.

2. Another approach is to count the number of instances for each unique point in prediction space and visualize their number via glyph size. This has been done using the same glyphs as in (1), where their previously constant surface area is now proportional to the number of instances. This does not allow to distinguish between specific instances, but the prevalence of each category is apparent at a single glance. Figure 8.3 shows all glyphs in the same order as in the later figures. Figures 8.5 to 8.7 show the results separately for each dataset.

[4] Only in this case was the instance previously *unseen* and thus only then can the classifier's prediction be considered a reasonable estimate of its performance on *unseen* data.

[5] We are aware of another simplification: StackingC utilizes class probabilities from the base classifiers, where we consider just the predictions which is slightly less informative. So, we also looked at the results from StackingP which utilizes just the predictions of its base classifiers. We found them to be practically indistinguishable from the results reported here, which is also confirmed by Chapter 3, where we found that there are no significant differences between using predictions and probability distributions meta data

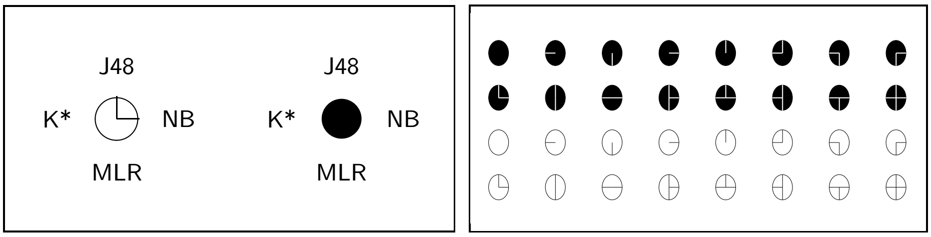


Figure 8.3: On the left side of the figure, we see two example glyphs. The main form is a circle. A filled circle indicates that StackingC predicted the wrong class; an empty circle indicates otherwise. Each direction (up, right, down and left) corresponds to a base classifier (J48, NaiveBayes, MLR and KStar) as shown, where the presence of a line indicates a correct prediction. The left glyph sample thus encodes an example where NaiveBayes, J48 and StackingC were correct; while the right glyph sample encodes an example where all base classifiers were incorrect, including StackingC. If one of the base classifiers were correct in this case, we would see a white line from the center of the filled circle into the appropriate direction, analogous to the black line for the empty circle. On the right side of this figure we see all thirty-two possible glyphs. The two upper rows correspond to instances where StackingC predicted the class wrong; the other two rows to correct predictions. The same order of glyphs is used later.

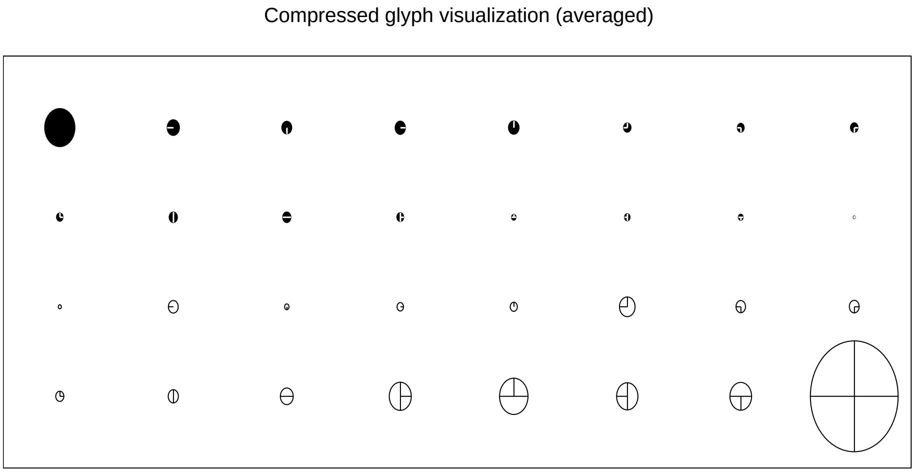


Figure 8.4: Averaged glyph visualization. The sizes of the corresponding glyphs have been averaged over all datasets.

For the first visualization approach, almost no patterns are readily apparent. What can be seen is that for almost all datasets the most frequent glyph is $\oplus$ which corresponds to *all base classifiers and StackingC correct*; glyphs which encode an incorrect prediction of StackingC (the filled circles, e.g. $\bullet$) are seldom found and do not seem to follow stable patterns. Because of the small number of errors by StackingC, significant inter-class differences in prediction performance were also not observed.

The prevalence of the $\oplus$ glyph can be seen more clearly in the second approach, Figure 8.5 to 8.7 – for all but two datasets, $\oplus$ is the glyph representing the highest number of instances. Also, for about half of our datasets, $\bullet$ is the second-most frequent glyph. Overall, the two largest groups of instances are those where all base classifiers and StackingC are already correct and those where none is correct. Although StackingC would potentially be able to predict correctly even when only a minority or even none of the base classifiers were correct, this is not observed often.

To obtain more general results, we chose to average the glyph sizes over all datasets. The result can be found in figure 8.4. As we can see by comparing the upper two to the lower two rows, there is a very small minority of cases where StackingC manages to predict correctly although all base classifiers are wrong – as expected, StackingC is still wrong on almost all cases where all the base classifiers are wrong. If we look at the cases where one base classifier is correct, StackingC manages to improve a bit, yielding almost equally many correct and incorrect predictions for the best case, when the single correct base classifier is KStar. For two correct base classifiers, StackingC becomes even better: the number of correct predictions is about twice the number of incorrect predictions, in the case of KStar and J48 being correct.[6]. For three correct base classifiers simple voting is always able to get the final class correct – however, StackingC is not quite as good. Although the minority of wrong predictions is quite small, in sum it cannot be neglected. The ability of StackingC to get correct final predictions in case of only one or two correct base classifiers has to be traded off against this weakness. Overall, StackingC still seems to outperform voting, but given the much higher computational cost this would be expected. Finally, if all four base classifiers are unanimously correct – overall the most frequent case – the number of cases where StackingC predicts a wrong class is so small as to be practically negligible.

8.4 Related Research

Card, Mackinlay & Shneiderman (1999) have collected a variety of papers on recent research from the field of Information Visualization. We recommend it as general introduction into this field.

Keim & Kriegel (1996) give a comprehensive overview on visualization techniques for data mining, citing essentially the same methods as in our introduction. Pixel-oriented techniqes (spiral, axes and grouping), parallel coordinates and stick figures were implemented and made available as the *VisDB* toolkit[7]

[6] In some of these cases simple voting may also be able to arrive at the right class, provided the number of classes is larger than two and the two remaining base classifiers each predict a different wrong class

[7] Binaries for HPUX and Linux are available at `www.dbs.informatik.uni-muenchen.de/dbs/projekt/visdb/visdb.html`. The source code does not seem to be forthcoming. We

Spoerri (1995) introduces InfoCrystal, a generalization of Venn diagrams for more than three sets which has also applications for visualization and creation of boolean queries. Their representation could have been applied to our prediction space visualization; however, their software is unavailable.

The data-mining toolkit WEKA[8] which we used for all our experiments, offers simple one- and two-dimensional scatterplots, with color as an additional dimension.

We now survey the techniques used or potentially useful to visualize models of classifiers. In our case, the composite model of StackingC includes all base classifiers' models and is thus too complex to visualize all-at-once[9]

While cam-trees and cone-trees (Robertson, Mackinlay & Card, 1991) could be used to visualize decision trees, we are not aware that this has been systematically investigated[10]. Most commercial data mining tools allow visualizing of decision trees as two-dimensional graphs, but the problem of visualizing very large or very unbalanced decision trees remains open.

Becker, Kohavi & Sommerfield (1997) proposes a visualization method for the NaiveBayes classifier. Conditional probabilities, relative to class and attribute values and the number of instances on which the probability is based are all visible at a single glance. Their method has been integrated into SGI MineSet, which is no longer available.

KStar and other instance-based classifiers such as IBk can easily be visualized by computing pairwise distances between instances, exactly as used internally by the instance-based classifier to determine nearest neighbors, and then visualizing this distance matrix via appropriate dimensionality reduction methods from multi-dimensional scaling, e.g. Sammon Mapping, Principal Components Analysis or Self-Organizing maps. Again, we are not aware that this has been investigated systematically.

Since MLR learns a weighted sum of attribute values, to predict each class probability separately, the visualization is quite straightforward. The weight of each attribute corresponds to its importance for determining the class; a weight near zero means that the influence is small and a large negative weight means that the attribute correlates negatively to the appropriate class probability. For a variant similar to StackingC Ting & Witten (1999) have shown the actual weights of the level-1 classifier, for each class separately, in a table. While this is only a trivial visualization, more useful approaches such as bar or pie charts (similar to (Becker, Kohavi & Sommerfield, 1997)) could easily be used. Again, we are not aware that this has been studied systematically.

attempted to run the software, but it seems that a slight change in the X-server interfaces since compilation prevents its use.

[8] available freely on www.cs.waikato.ac.nz/\~ml

[9] The internal crossvalidation generates ten models for each base classifier, plus one on the complete training data. This is repeated for each step of the outer crossvalidation, yielding a overall number of 440 models for StackingC – on each dataset. Even if we restrict ourselves to the models used to obtain the predictions, there would still be 40 models to visualize for each dataset.

[10] Although we are aware that the author programmed a tentative 3D visualization for decision trees as a student, for the lecture *Visualisierung* at the Institute of Computer Graphics and Algorithms, Technical University of Vienna. ;-)

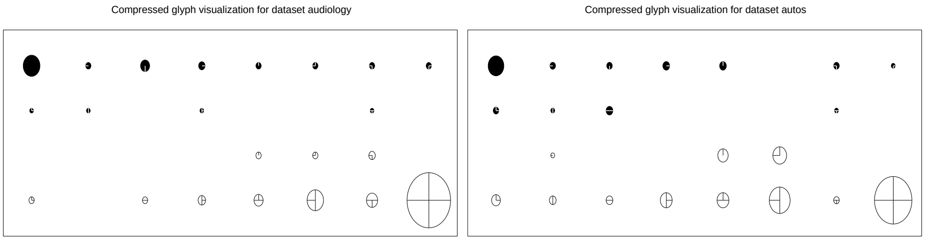


Figure 8.5: Glyph visualization for datasets *audiology* and *autos*. The area of each glyph is proportional to the number of instances in the corresponding category.

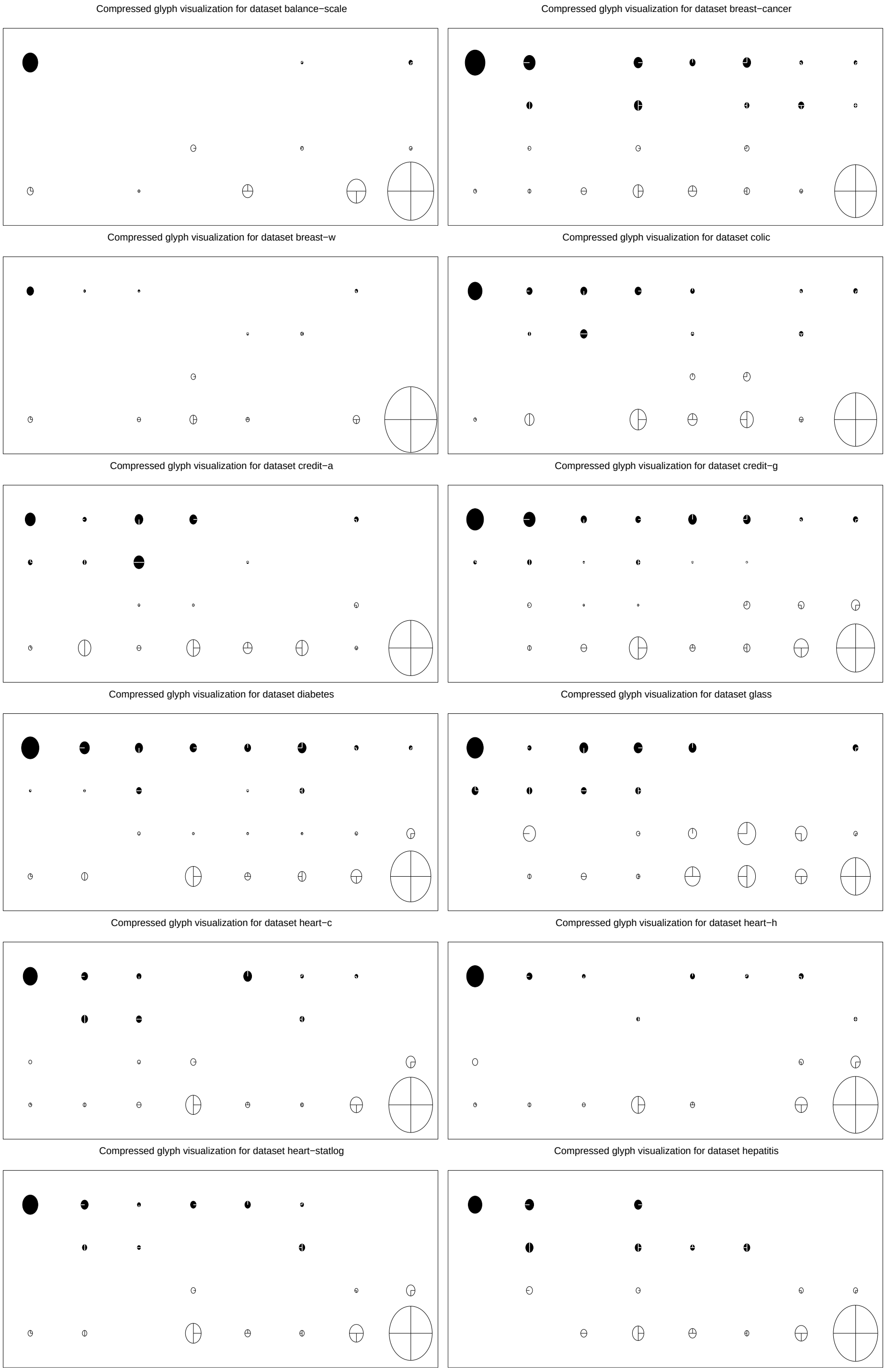
8.5 Conclusion

We have introduced Information Visualization and applied it to the problem of visualizing datasets (instance space) and also to visualize some of our experimental results (prediction space). By means of these representations, we found and discussed some interesting structures.

For instance space, we found a significant mismatch of the number of reported and actual classes for two datasets. Also, we found quite some minority classes with very few instances – in some cases, only one. Furthermore, we found and identified one synthetic dataset, *balance-scale*, via its strikingly regular pattern. Finally, we found that Sammon Mapping is able to capture partial class structure in some datasets, although class information was not part of its input – however, this does not correlate well with the performance of the instance-based KStar classifier as may have been expected. This can be explained because KStar's internal distance measure varies significantly from our transformations of categorical and numeric attributes.

For prediction space, we found that the two most frequent groups of instances are first those where all base classifiers and StackingC are correct and then those where all base classifiers and StackingC are wrong in their predictions. While some cases exist where StackingC is able to find the correct class even when none or a minority of base classifiers are correct, this has to be discounted against its slightly worse performance in cases where the majority of base classifiers is correct. Overall, StackingC still outperforms simpler methods such as Voting but given the much higher computational cost one would have hoped for a larger margin of benefit.

Concluding, we have found the field of Information Visualization to be a valuable addition and inspiration for our research and hope that it will be more widely applied in the future. However, free research tools for basic visualization methods are quite hard to find – sadly, sometimes even the code for quite recent PhDs is either no longer available or too old to run on current machines. Therefore, we had to implement quite some methods from scratch. In our case they are freely available – please contact the author if interested.



Compressed glyph visualization for dataset balance−scale

Compressed glyph visualization for dataset breast−cancer

Compressed glyph visualization for dataset breast−w

Compressed glyph visualization for dataset colic

Compressed glyph visualization for dataset credit−a

Compressed glyph visualization for dataset credit−g

Compressed glyph visualization for dataset diabetes

Compressed glyph visualization for dataset glass

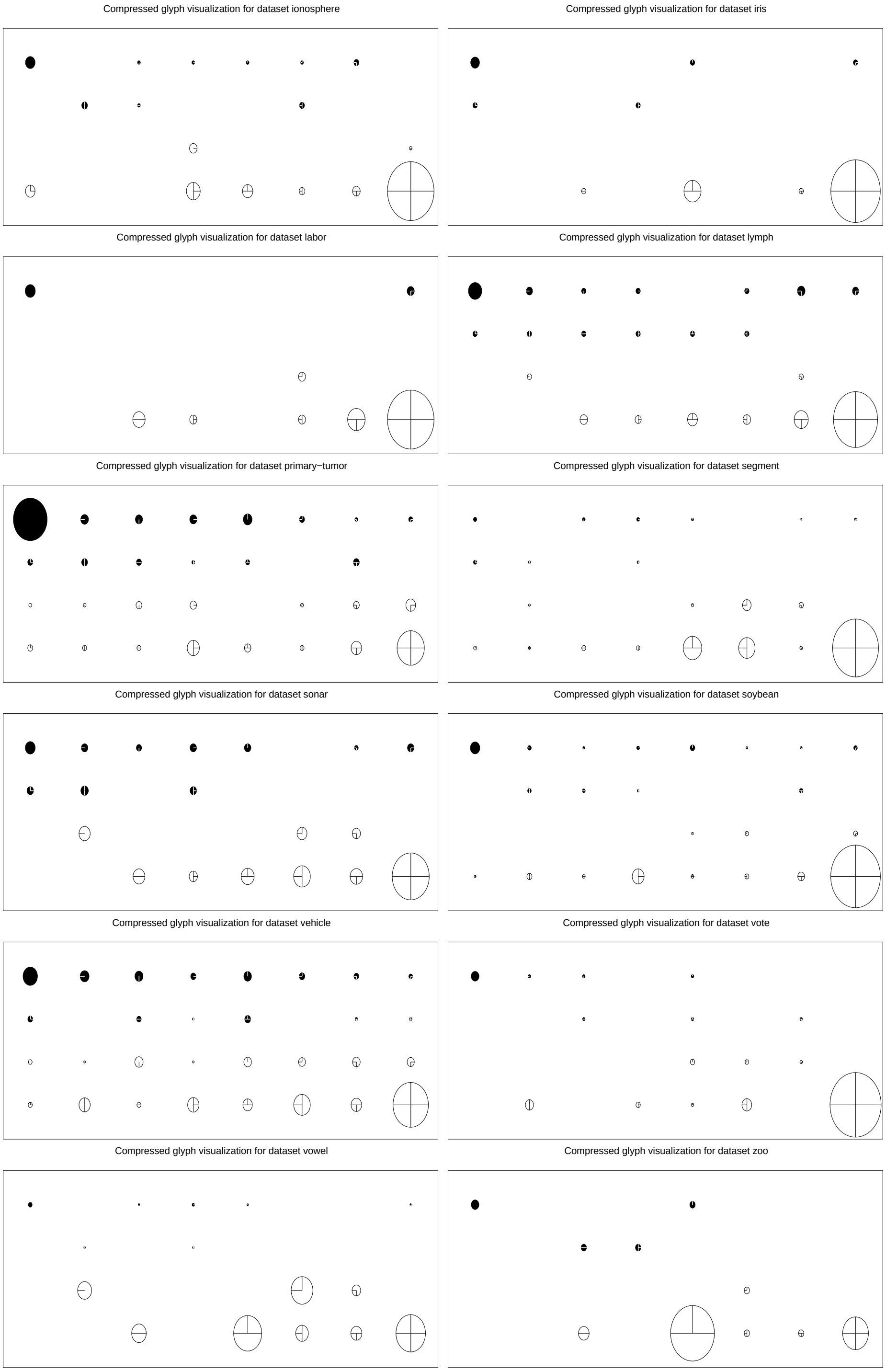
Compressed glyph visualization for dataset heart−c

Compressed glyph visualization for dataset heart−h

Compressed glyph visualization for dataset heart−statlog

Compressed glyph visualization for dataset hepatitis

Figure 8.6: Glyph visualization for datasets *audiology* to *hepatitis*.



Compressed glyph visualization for dataset ionosphere

Compressed glyph visualization for dataset iris

Compressed glyph visualization for dataset labor

Compressed glyph visualization for dataset lymph

Compressed glyph visualization for dataset primary−tumor

Compressed glyph visualization for dataset segment

Compressed glyph visualization for dataset sonar

Compressed glyph visualization for dataset soybean

Compressed glyph visualization for dataset vehicle

Compressed glyph visualization for dataset vote

Compressed glyph visualization for dataset vowel

Compressed glyph visualization for dataset zoo

Figure 8.7: Glyph visualization for datasets *ionosphere* to *zoo*.

Chapter 9

Summary of Conclusions

The conclusions of the preceding chapters have already shown what we have achieved. Still, we would like to summarize our main results here, in the order of appearance. More detailed conclusions can be found in the corresponding chapter. At the end of this chapter we will give overall conclusions and an outlook towards future research.

9.1 Exploring the Parameter State Space

We have extensively explored the parameter state space of Stacking. Concerning the choice of base classifiers, we have found a set of four base classifiers, chosen by a priori and a posteriori arguments, which performs best. However, using all available base classifiers also remains an acceptable option, although the performance may slightly deteriorate.

Concerning the choice of meta classifier and choice of meta data to be used, we have found that MLR is indeed the best meta classifier for probability distribution data. We showed probability distribution meta data and predictions meta data to perform comparably. For predictions meta data, NaiveBayes is a reasonable choice.

StackingC seems least influenced by specific sets of base classifiers, which leads us to expect that base classifier choice has even less influence on StackingC than it has on Stacking. Thus we believe that repeating our experiments with StackingC would not lead to new insights. It still remains to be investigated whether this is also true for sMM5 by Džeroski and Zenko (2002). More details can be found in Chapter 3.

9.2 Meta-Learning for Stacking

Here, we have investigated whether Meta-Learning methods can be used to accuractely predict various aspects of Stacking's behaviour.

In the context of predicting Stacking's accuracy, we found that classifier-related features, namely those derived from accuracy, are excellently suited to this task, as have others e.g. (Bensusan & Kalousis, 2001; Pfahringer et al., 2000). A single feature, the accuracy of the best component classifier in the ensemble, is able to predict the accuracy of the stacked classifier suprisingly well.

We also investigated the prediction of significant differences between Stacking and other ensemble learning schemes. Contrary to expectations, we found that features derived directly from the dataset were better suited in this case. For the model which predicts significant differences between Grading and Stacking, insight into the inner workings of both schemes have enabled us to formulate a tentative explanation of the learned model.

At last we have found that there is no single best meta classifier for predicting significant differences – a variety of machine learning algorithms had to be evaluated for best results. This hints that learning problems which aim to distinguish between pairs of classifiers have quite different properties, which could explain why Meta-Learning a single model for many classifiers at once is so hard. More details can be found in Chapter 4.

9.3 Improving upon Stacking: Stacking with Confidences

We have presented empirical evidence that Stacking in the extension proposed by (Ting & Witten, 1999) performs worse on multi-class datasets than on two-class datasets, for all but one meta-learner we investigated. This can be explained as follows: With a higher number of classes, the dimensionality of the meta-level data is proportionally increased. This higher dimensionality makes it harder for meta-learners to find good models, since there are more features to be considered. Stacking using meta-level data consisting of predictions does not suffer from this weakness, as would be expected.

In order to improve on the status quo, we have proposed and implemented a new Stacking variant, StackingC, based on reducing the dimensionality of the meta dataset so as to be independent of the number of classes and removing a priori irrelevant features, and shown that it resolves this previously unreported weakness, for MLR and two other related meta-learners considered. We believe that the source of this improvement lies partially in the dimensionality reduction, but more importantly in the higher diversity of class models. Using one-against-all class binarization and regression learners for each class model seems to be essential. This variant is competitive to other current variants, e.g. sMM5 by Džeroski and Zenko (2002). More details can be found in Chapter 5.

9.4 Learning Curves

In this chapter, we investigate the hypothesis that StackingC is the most stable ensemble learning scheme. We define the stability of a learner as its continued good performance in the face of a reduction in training data, i.e. its capability to find the best or a reasonably good model when confronted with less training data, and – in case of insufficient amounts of training data – a graceful degradation in performance.

In light of evidence presented in this chapter, we cannot support the original hypothesis that StackingC offers more stable performance than other ensemble learning schemes. On the contrary, we found almost no significant differences between our ensemble learning schemes and even vs. the best base classifier by hindsight. On a more positive note, we found that even the simplest scheme Voting is competitive to the best base classifier by hindsight, even though Voting does not use the expensive crossvalidation of all other schemes.

On a final note, we believe that stability is a desirable property of classifiers and may be complementary to classic accuracy estimates. Detailed results can be found in Chapter 6.

9.5 Towards a Theoretical Framework

Here, we have shown that Stacking is equivalent to most ensemble learning schemes, namely *Selection by Crossvalidation* (X-Val), Voting of either class probability distributions or predictions, and Grading. We have given functional descriptions of suitable meta classifiers for Stacking which simulate the operation of these ensemble learning schemes. By a simple wrapper we were also able to simulate Bagging. Recent variants such as StackingC (Seewald, 2002a), see also Chapter 5 and sMM5 (Džeroski & Zenko, 2002) can also be simulated in the same way. So Stacking can be seen as conceptual abstraction of all these schemes. Thus we conclude that our approach offers a unique viewpoint on Stacking which is an important step towards a theoretical framework for ensemble learning. Detailed results can be found in Chapter 7.

9.6 Information Visualization

In this chapter, we have introduced Information Visualization, a fairly new field that deals with visualization of high-dimensional data where no inherent spatial or temporal structure is present. We have applied methods from this field to the problem of visualizing datasets (instance space) and also to visualize some of our experimental results (predition space).

For instance space, we found a significant mismatch of the number of reported and actual classes for two datasets. Also, we found quite some minority classes with very few instances – in some cases, only one. Furthermore, we found and identified one synthetic dataset, *balance-scale*, via its strikingly regular pattern.

For prediction space, we found that the two most frequent groups of instances are first those where all base classifiers and StackingC are correct and then those where all base classifiers and StackingC are wrong in their predictions. While some cases exist where StackingC is able to find the correct class even when none or a minority of base classifiers are correct, this has to be discounted against its worse performance in cases where the majority of base classifiers is correct.

Concluding, we have found the field of Information Visualization to be a valuable addition and inspiration for our research and hope that it will be more widely applied in the future. However, free research tools for basic visualization methods are quite hard to find – sadly, sometimes even the code for quite recent PhDs is not longer available or too old to run on current machines. So we were forced to reimplement quite some methods from scratch.[1] Detailed results can be found in the Chapter 8

[1] In our case they are freely available, please contact the author.

9.7 Conclusion and Outlook

While we believe that the current variants StackingC (see Chapter 5) and sMM5 are very close to the achievable optimum[2], we are disappointed that the competitive advantage over simpler methods such as unweighted Voting is quite small – and in some comparisons does not exist at all. Given that all Stacking variants rely on expensive crossvalidation, which makes them at least an order of magnitude slower than Voting, this is quite unsatisfactory. Possibly neurophysiological research will once reveal how the brain is able to work efficiently as a very large ensemble – as of now, we have exhausted the issue of ensemble learning in the guise of Stacking quite thoroughly.

We still hope that this thesis stimulates further work in understanding ensemble learning and applying ensembles in other research areas such as Game Playing, Self-Diagnosing Systems and BioInformatics. Overall, the concept of ensembles has been a great inspiration to our work and still remains fruitful. We also hope that our results concerning parameter choice will help Stacking to become more visible within and outside the machine learning community, instead of Neural Networks and Support Vector Machines which are at least as computationally inefficient and opaque. Finally, we hope that our results from Information Visualization motivates other researchers to create if not more colorful then at least more creative and inspiring figures.

[2] In fact several experiments concerned with improving performance further, e.g. by combining both current variants, amply demonstrated the law of diminishing returns.

Bibliography

Bauer, E., Kohavi, R. (1999): An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36, pages 105-169.

Becker, B., Kohavi, R., Sommerfield, D. (1997): Visualizing the Simple Bayesian Classifier. In KDD 1997 Workshop on Issues in the Integration of Data Mining and Data Visualization.

Bensusan, H., Kalouis, A. (2001): Estimating the Predictive Accuracy of a Classifier. In Proceedings of the twelveth European Conference on Machine Learning, Freiburg, Germany. Springer Verlag.

Bezdek, J.C., Nikhil, R.P. (1995): An index of topological preservation for feature extraction. Pattern Recognition, Vol.28, No.3, pages 381-391.

Blake, C. L., Merz, C. J (1998): UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (1998). Department of Information and Computer Science, University of California at Irvine, Irvine CA.

Borg, I., Groenen, P. (1997): Modern Multidimensional Scaling. Springer-Verlag, New York.

Brazdil, P. B., Gama, J., & Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. *Proceedings of the 7th European Conference on Machine Learning (ECML-94)* (pages 83-102), Catania, Italy, Springer-Verlag.

Breiman, L. (1996): Bagging Predictors. Machine Learning (24), 123-140.

Brodley, C., Lane, T. (1996): Creating and Exploiting Coverage and Diversity. Integrating Multiple Learning Model Workshop, AAAI-96, Portland, Oregon.

Card, S. K., Mackinlay, J., Shneiderman, B., eds. (1999): Readings in Information Visualization: Using Vision to Think. San Francisco, Morgan Kaufmann.

Chan, P. K., & Stolfo, S. J. (1995). A comparative evaluation of voting and meta-learning on partitioned data. *Proceedings of the 12th International Conference on Machine Learning (ICML-95)* (pages 90-98). Morgan Kaufmann.

Cleary, J. G., Trigg, L. E (1995): K*: An instance-based learner using an entropic distance measure. In Prieditis, A., Russell, S., Proceedings of the 12th International Conference on Machine Learning, pages 108-114, Lake Tahoe, CA.

Dietterich, T.G. (1998): Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10 (7), pages 1895-1924.

Dietterich T. G. (2000a): An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Machine Learning 40(2), pages 139-158.

Dietterich, T. G. (2000b): Ensemble methods in machine learning. In Kittler, J., Roli, F., First International Workshop on Multiple Classifier Systems, pages 1-15. Springer-Verlag.

Džeroski S., Zenko B. (2002): Is Combining Classifiers Better than Selecting the Best One?, in Proceedings of the 19th International Conference on Machine Learning, ICML-2002, Morgan Kaufmann Publishers, San Francisco.

Fan, W., Stolfo, J. S., Chan, P. K. (1999): Using Conflicts Among Multiple Base Classifiers to Measure the Performance of Stacking. In Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work (1999) 66-73. Jozef Stefan Institute, Ljubljana, Slovenia.

Flexer A. (2001): On the use of self-organizing maps for clustering and visualization, Intelligent Data Analysis 5(5), pages 373-384.

Freund, Y., Schapire R.E. (1996): Experiments with a new boosting algorithm, Proceedings of the International Conference on Machine Learning, pages 148-156, Morgan Kaufmann, San Francisco.

Fürnkranz J. (2001): Round Robin Rule Learning, in Brodley C.E., Danyluk A.P., Proceedings of the 18th International Conference on Machine Learning (ICML-01), Morgan Kaufmann, San Francisco, pages 146-153.

Gama, J. (1999). Discriminant Trees. In Bratko I. & Džeroski S.(eds.), Proceedings of the 16th International Conference on Machine Learning, ICML'99, pages 134-142. Morgan Kaufmann, Los Altos/Palo Alto/San Francisco.

Gama, J., Brazdil, P. (1995): Characterization of classification algorithms. In Proceedings of the 7th Portugese Conference in AI, EPIA 95, pages 83-102.

Gama, J.; and Brazdil, P. (1999): Linear Tree. Intelligent Data Analysis 3(1):1-22.

Gama J., Brazdil P. (2000): Cascade Generalization, Machine Learning 41(3), pages 315-344.

Hansen, L. K., Salamon, P. (1990): Neural Network Ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12:10, pages 993-1001.

Keim, D.A., Kriegel, H-P (1996): Visualization Techniques for Mining Large Databases: A Comparison. In IEEE Transactions on Knowledge and Data Engineering, Vol.8, No.6, Dec. 1996.

Kohavi, R. (1996): Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In Simoudis E. & Han J.(eds.), KDD-96: Proceedings Second International Conference on Knowledge Discovery & Data Mining, pages 202-207. AAAI Press/MIT Press, Cambridge/Menlo Park.

Kohavi, R., Wolpert, D. (1996): Bias plus variance decomposition for zero-one loss functions. In Saitta L.(eds.), ICML-96: Proceedings of the Thirteenth International Machine Learning Conference, Bari, Italy. Morgan Kaufmann, San Francisco.

Kononenko, I., Bratko, I. (1991): Information-based evaluation criterion for classifier's performance. Machine Learning 6, pages 67-80.

Langley, P. (1993): Induction of Recursive Bayesian Classifiers. In Brazdil P.B.(ed.), Machine Learning: ECML-93, pages 153-164. Springer, Berlin/Heidelberg/New York/Tokyo.

Merz, C. J. (1999): Using correspondence analysis to combine classifiers, Machine Learning 36, pages 33-58.

Petrak, J. (2000): Fast subsampling performance estimates for classification algorithm selection. *Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination* (pages 3-14). Barcelona, Spain.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000): Meta-learning by landmarking various learning algorithms. *Proceedings of the 17th International Conference on Machine Learning (ICML-2000).* Stanford, CA.

Quinlan, J.R. (1987): Simplifying Decision Trees. International Journal of Man-Machine Studies 27:221-234.

Quinlan, J.R. (1992): Learning with Continuous Classes. Proceedings of the Australian Joint Conference on Artificial Intelligence, pages 343-348. World Scientific, Singapore.

Quinlan, J. R. (1993a): C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

Quinlan, J. R. (1993b): Comparing Connectionist and Symbolic Learning Methods: Constraints and Prospects, MIT Press.

Robertson, G., Mackinlay, J., Card, S. (1991): Cone Trees: Animated 3D Visualizations of Hierarchical Information, Proc. CHI'91 Human Factors in Comp. Systems.

Sammon, J.W. (1969): A nonlinear mapping for data structure analysis. IEEE Transactions on Computers, C-18:401-409, 1969.

Schaffer, C. (1993): Selecting a classification method by cross-validation. Machine Learning 13(1), pages 135-143.

Schaffer, C. (1994): Cross-validation, stacking and bi-level stacking: Meta-methods for classification learning. In P. Cheeseman and R. W. Oldford (Eds.), *Selecting models from data: Artificial Intelligence and Statistics IV*, pages 51-59. Springer-Verlag.

Seewald A.K., Fürnkranz J. (2001): An Evaluation of Grading Classifiers, in Hoffmann F. et al. (eds.), Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001, Proceedings, Springer, Berlin/Heidelberg/New York/Tokyo, pages 115-124. Also available as Technical Report (extended version) TR-2001-01, Austrian Research Institute for Artificial Intelligence, Vienna, Austria. `www.oefai.at`

Seewald A.K., Petrak J., Widmer G. (2001): Hybrid Decision Tree Learners with Alternative Leaf Classifiers: An Empirical Study, Proceedings of the 14th International FLAIRS Conference (FLAIRS-2001), AAAI Press, Menlo Park, California.

Seewald A.K. (2002a): How to make Stacking Better and Faster While Also Taking Care of an Unknown Weakness, in Proceedings of the 19th International Conference on Machine Learning, ICML-2002, Morgan Kaufmann Publishers, San Francisco.

Seewald A.K. (2002b): Meta-Learning for Stacked Classification, in Bohanec M. et al. (eds.), Second International Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2002), University of Helsinki, Department of Computer Science, pages 123-128.

Seewald A.K. (2002c): Exploring the Parameter State Space of Stacking, in Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM-2002), Maebashi City, Japan.

Skalak, D. B. (1997): Prototype Selection for Composite Nearest Neighbor Classifiers. PhD Dissertation, University of Massachusetts, Amherst, Massachusetts.

Spoerri A. (1995): InfoCrystal - A Visual Tool For Information Retrieval, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Ting, K. M. (1997): Decision combination based on the characterisation of predictive accuracy. *Intelligent Data Analysis*, *1*, pages 181-206.

Ting, K. M., Witten, I. H. (1999): Issues in stacked generalization. Journal of Artificial Intelligence Research 10, pages 271-289.

Todorovski, L., & Džeroski, S. (2000): Combining multiple models with meta decision trees. *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-2000)* (pages 54-64). Lyon, France: Springer-Verlag.

Utgoff, P.E. (1988): Perceptron Trees: A Cast Study in Hybrid Concept Representations. In Proceedings of the 7th National Conference on Artificial Intelligence, pages 601-605. Morgan Kaufmann, Los Altos/Palo Alto/San Francisco.

Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J. (2000): SOM Toolbox for Matlab, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland.

Wang, Y., Witten, I. H. (1997): Induction of model trees for predicting continuous classes. Proceedings of the poster papers of the European Conference on Machine Learning. University of Economics, Faculty of Informatics and Statistics, Prague.

Wolpert, D. H. (1992): Stacked generalization. Neural Networks 5(2), pages 241-260.

Appendix A

InfoVis Figures

This appendix shows complete figures of Chapter 8, Information Visualization, for all twenty-six datasets, each one with instance space and prediction space visualization. For the sake of completeness, the figures already shown in Chapter 8 are duplicated here in full-page format.

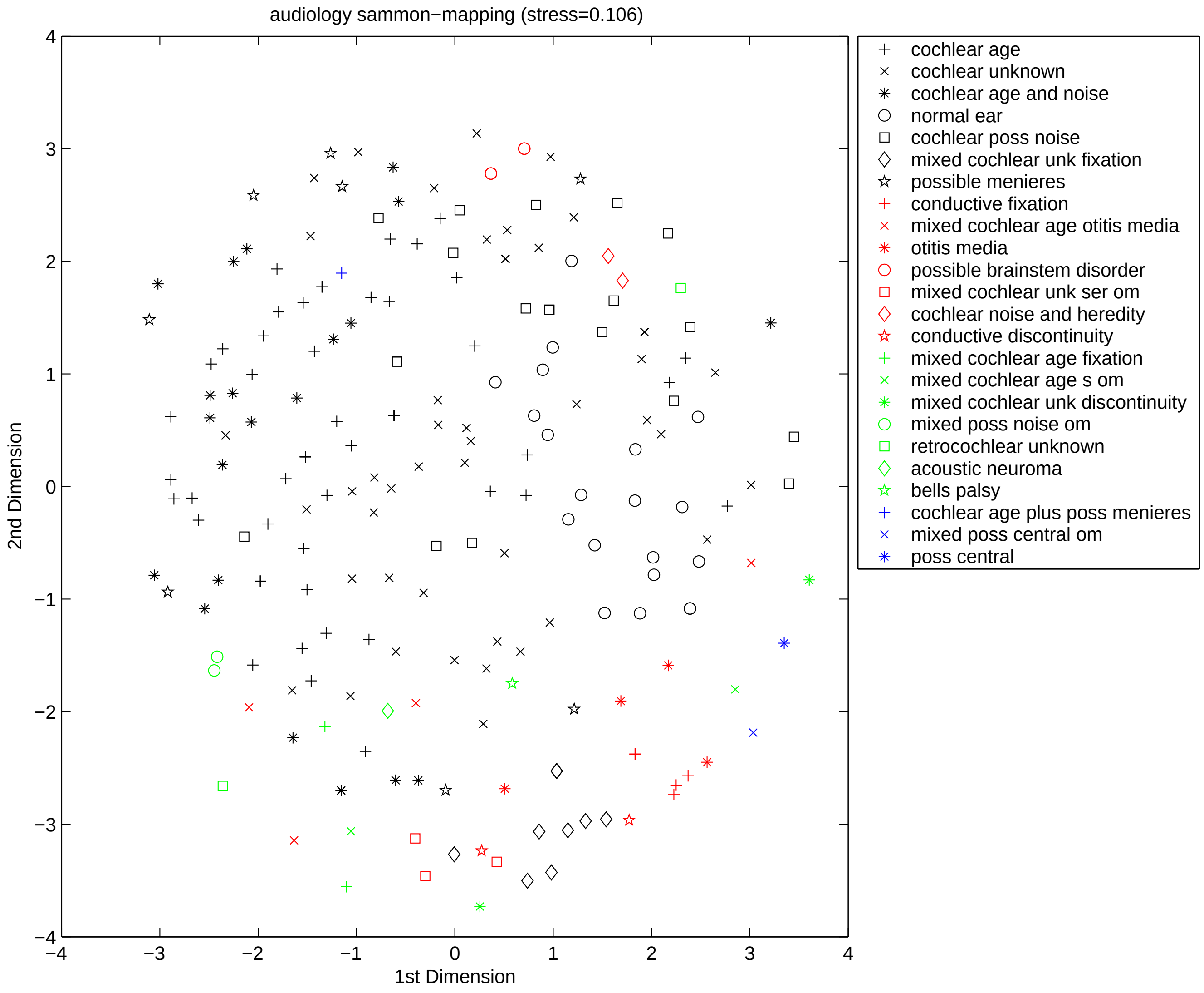


Figure A.1: Instance space visualization of *audiology* via Sammon-Mapping. The legend is sorted by descending frequency, i.e. the first line corresponds to the most frequent class. As we can see there are several classes with very few examples. Some classes are locally clustered, but this is quite indistinct.

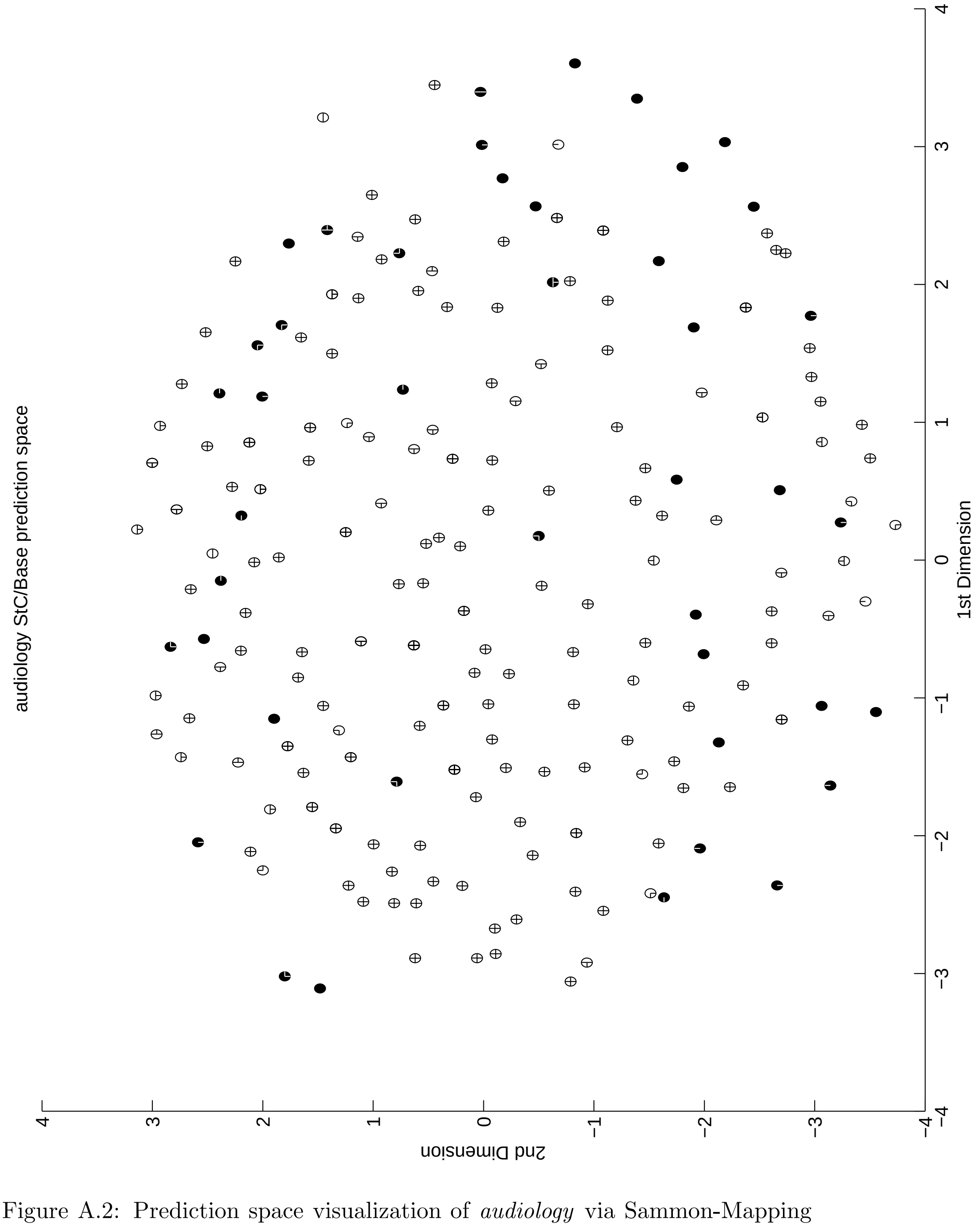


Figure A.2: Prediction space visualization of *audiology* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

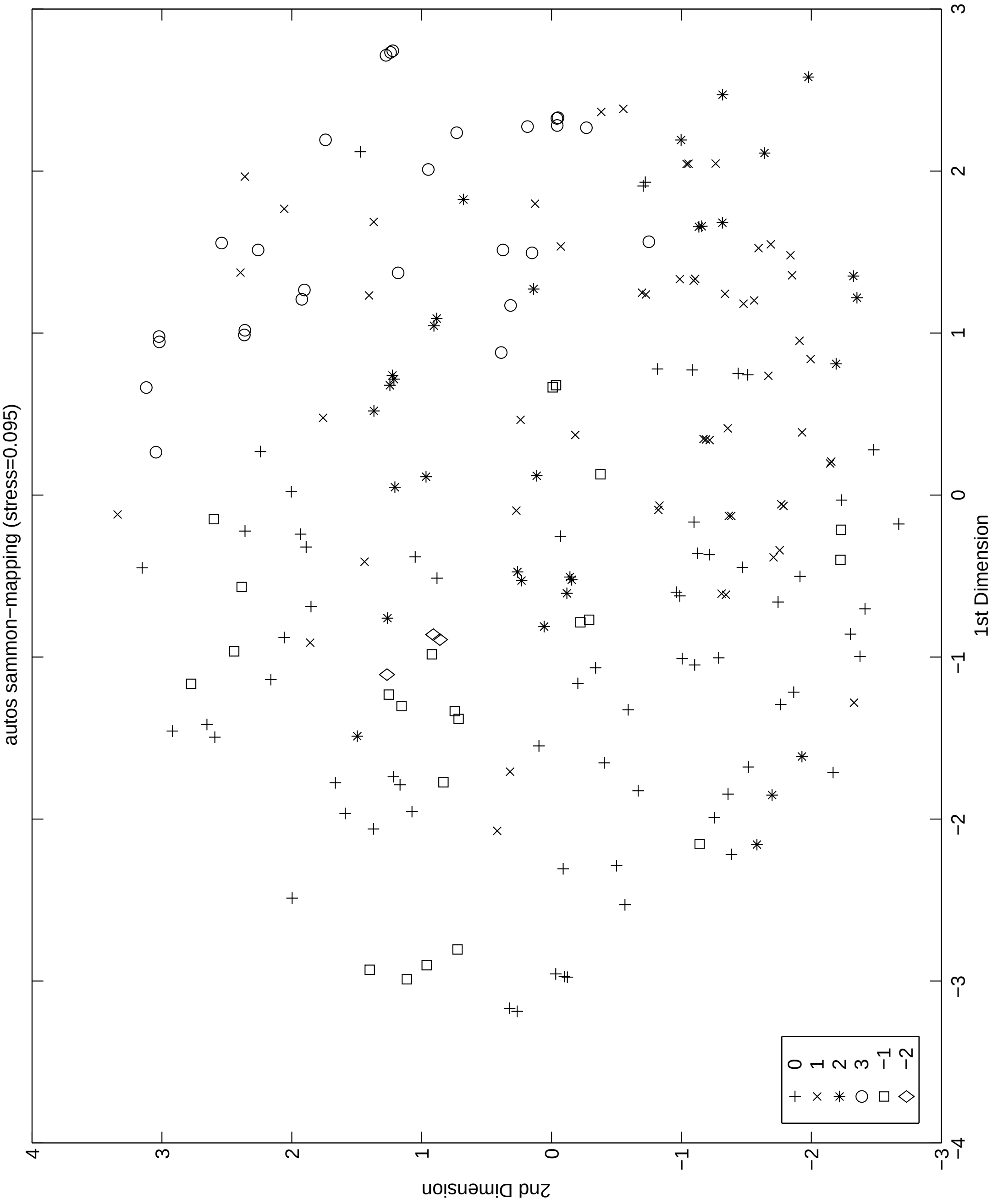


Figure A.3: Instance space visualization of *autos* via Sammon-Mapping. Class -3 never appears in the dataset; thus this is in fact a six-class dataset. The examples are sparsely distributed, but there are some local changes in density. The high number of two or more examples within small distance which share the same class is somewhat striking.

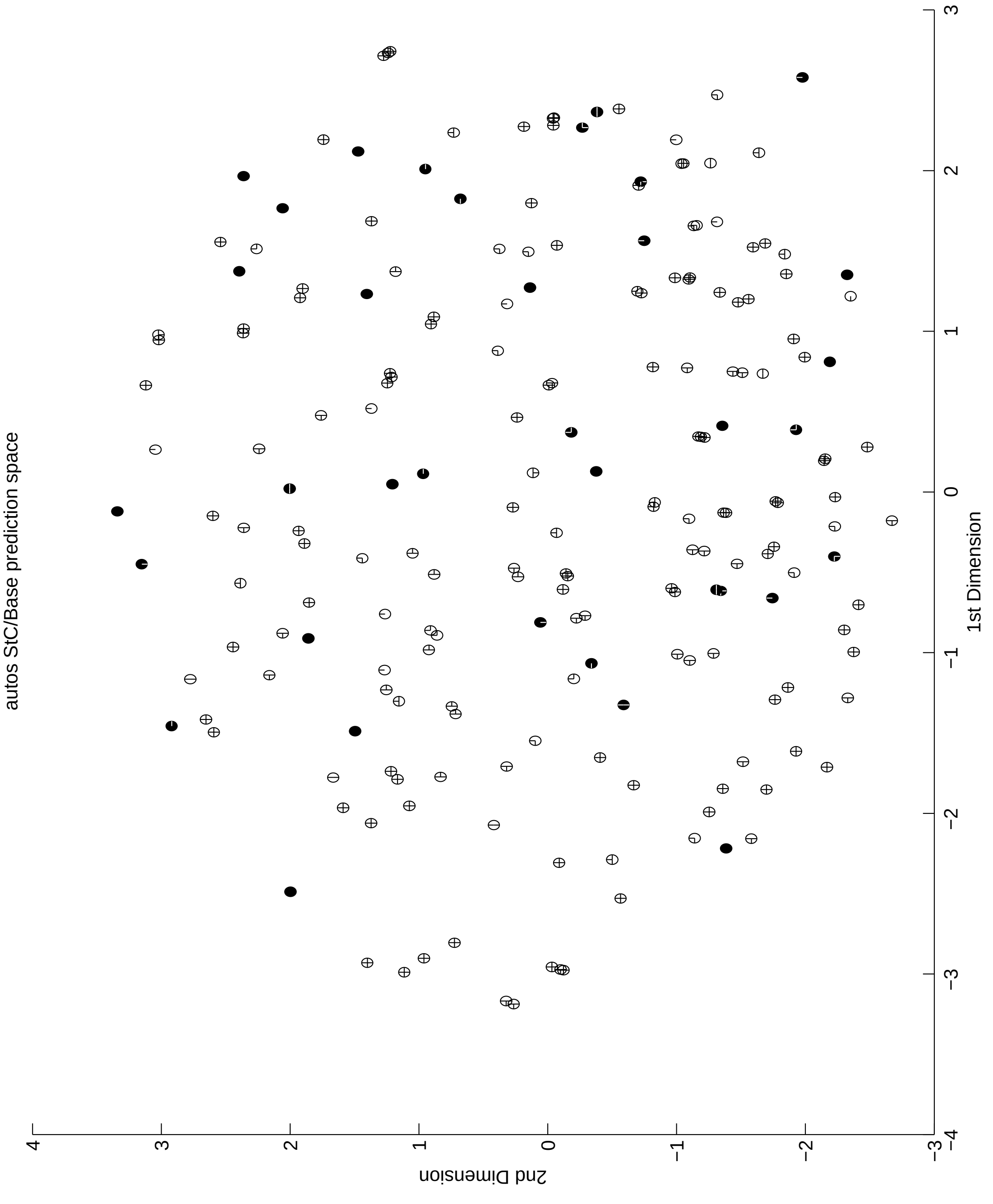


Figure A.4: Prediction space visualization of *autos* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

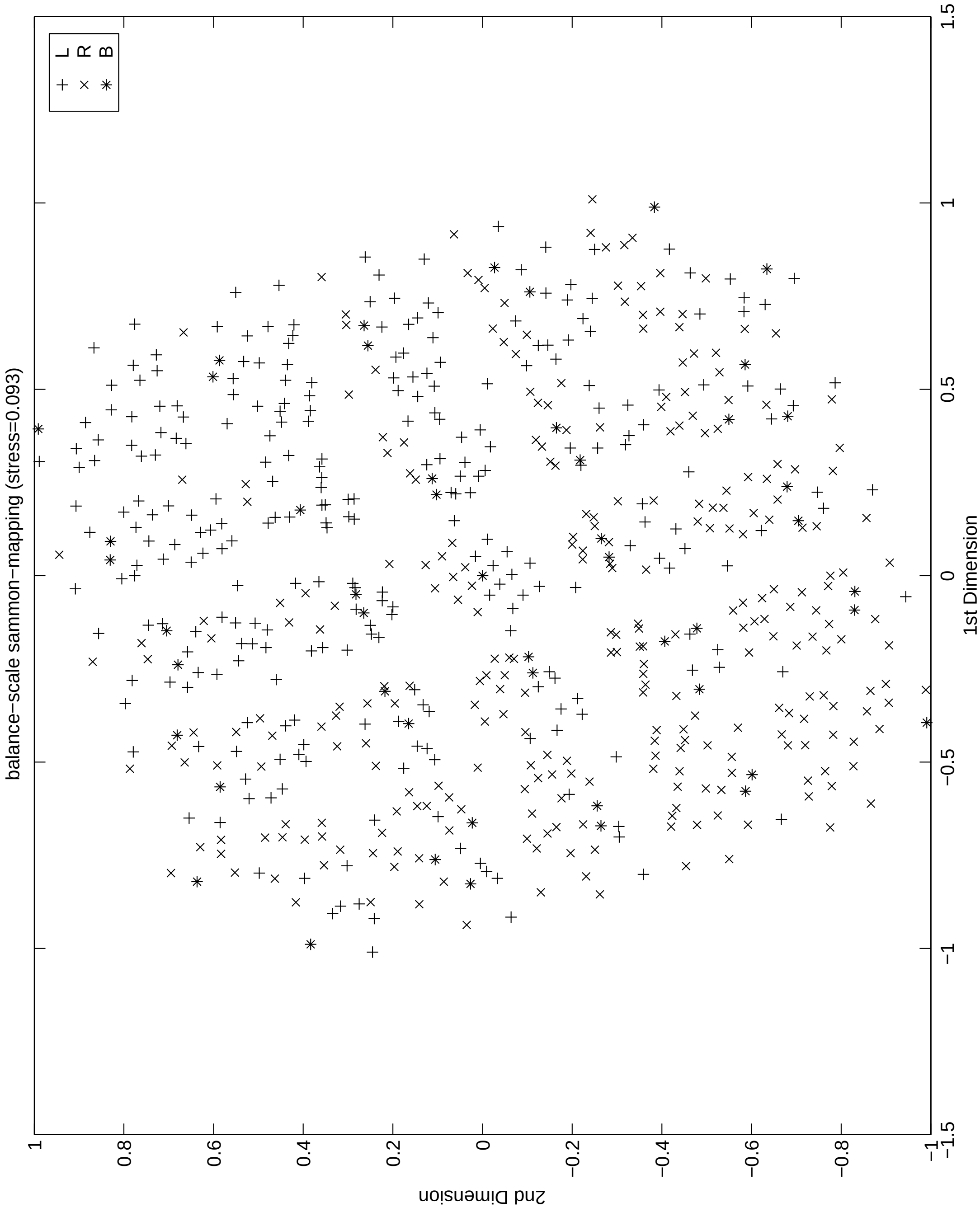


Figure A.5: Instance space visualization of *balance-scale* via Sammon-Mapping. The regular structure of this visualization is striking. We found out that this is due to the fact that the input is exhaustive: for all four variables, all values from zero to one in steps on 0.25 appear. This is actually a synthetic dataset, with a non-linear perfect model and thus of questionable use in empirical experiments. Adding appropriate attributes makes the problem trivial to solve.

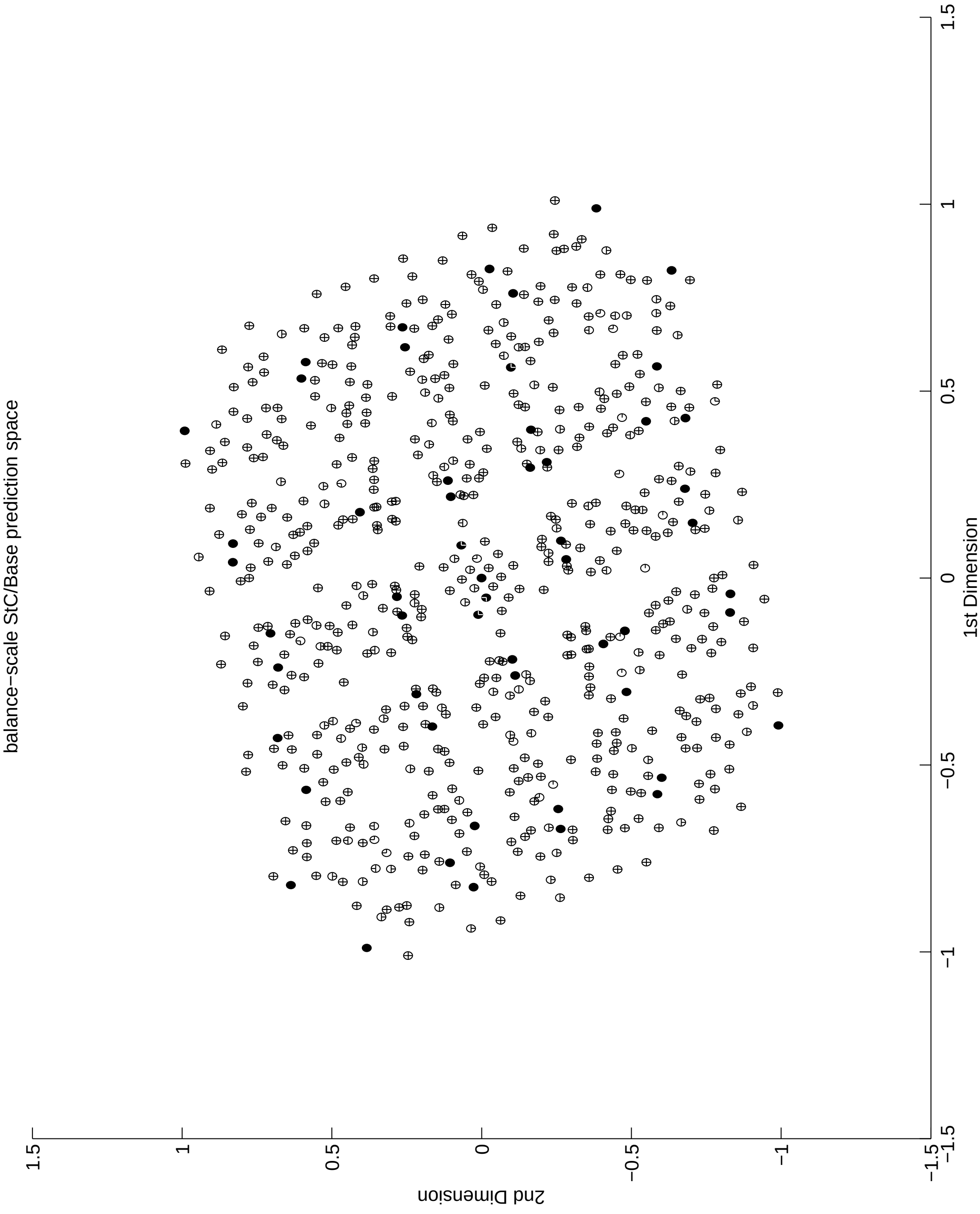


Figure A.6: Prediction space visualization of *balance-scale* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

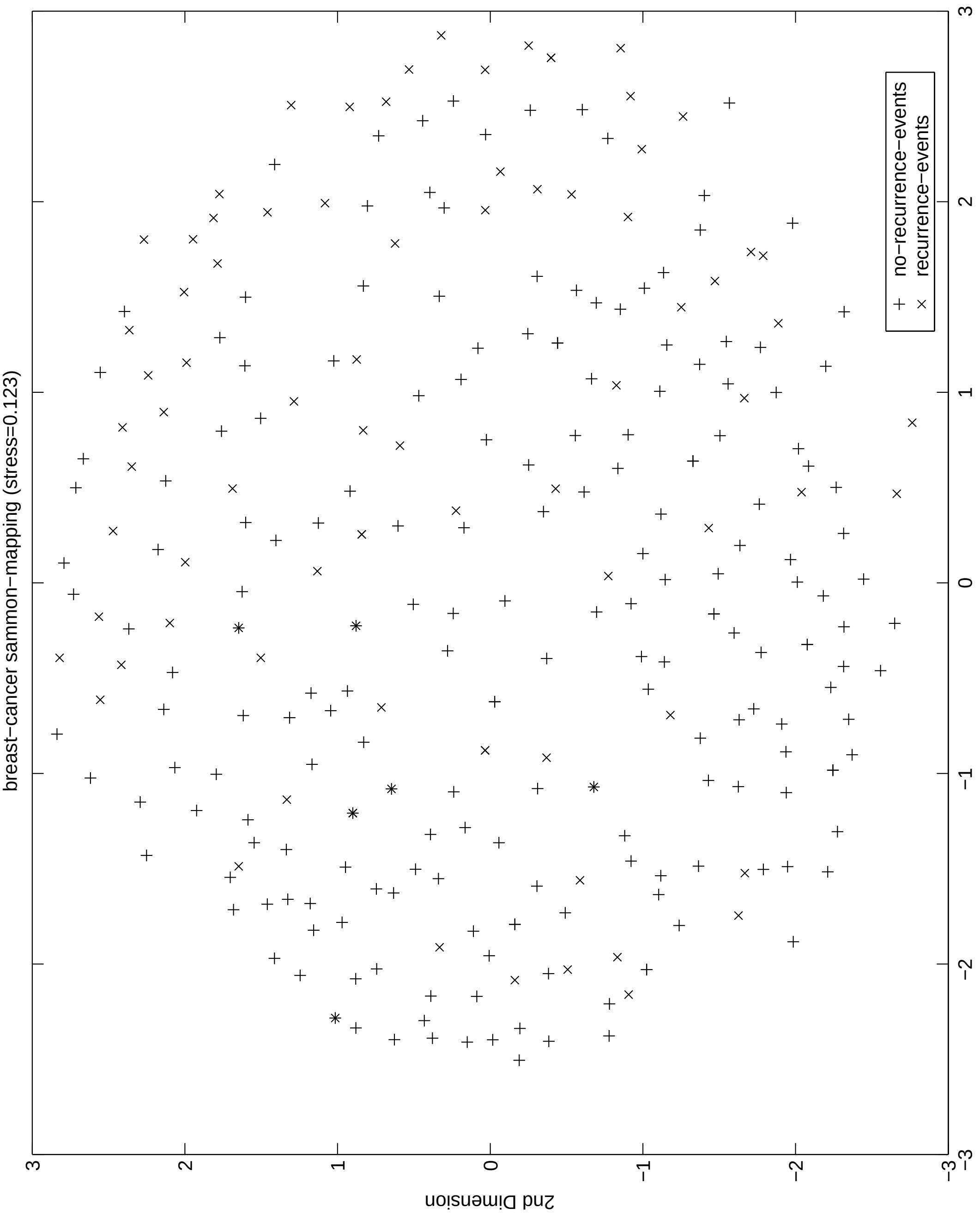


Figure A.7: Instance space visualization of *breast-cancer* via Sammon-Mapping. The examples appear quite sparse, but not completely uniformly randomly distributed. Some empty spaces between examples strike us as interesting.

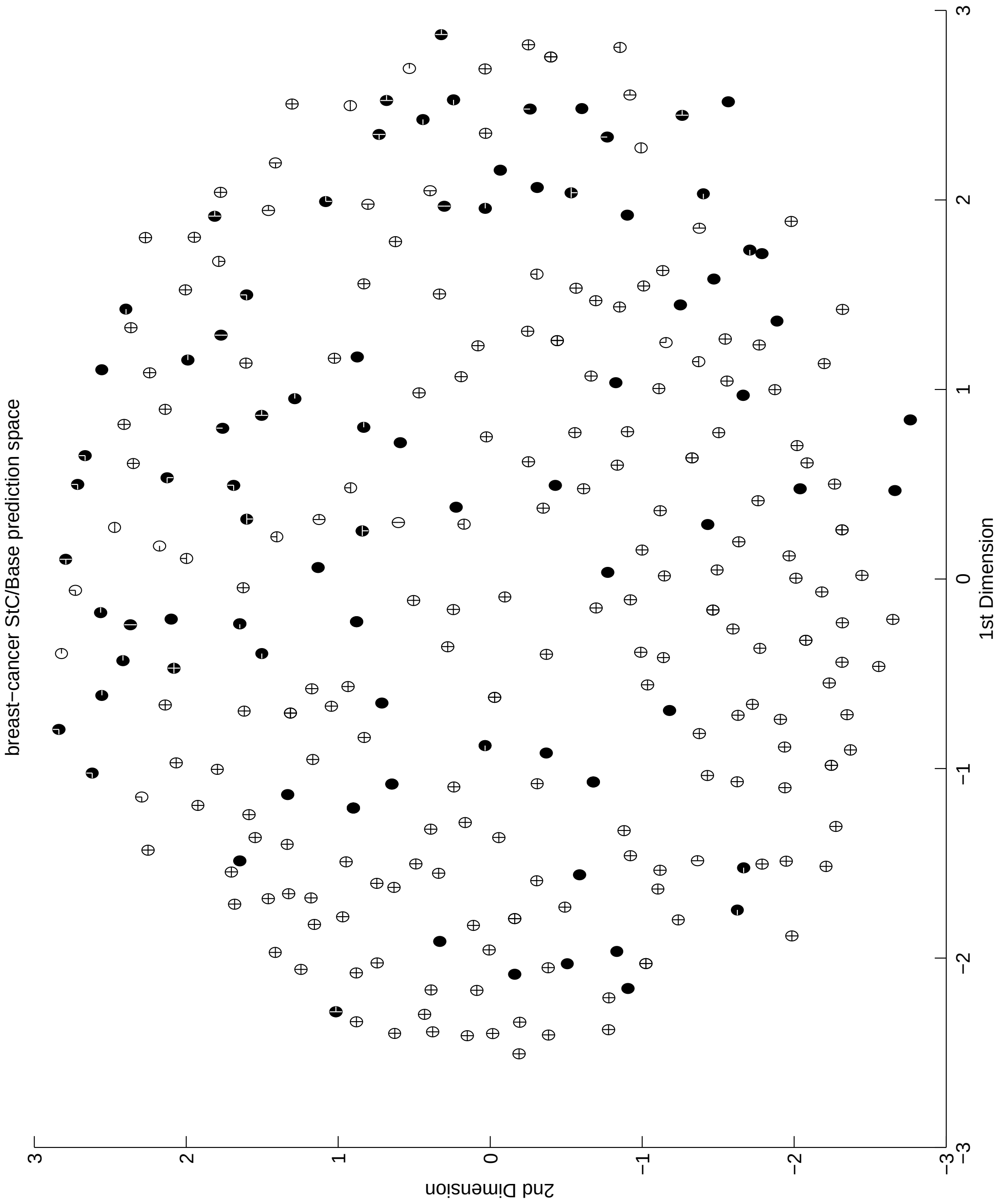


Figure A.8: Prediction space visualization of *breast-cancer* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

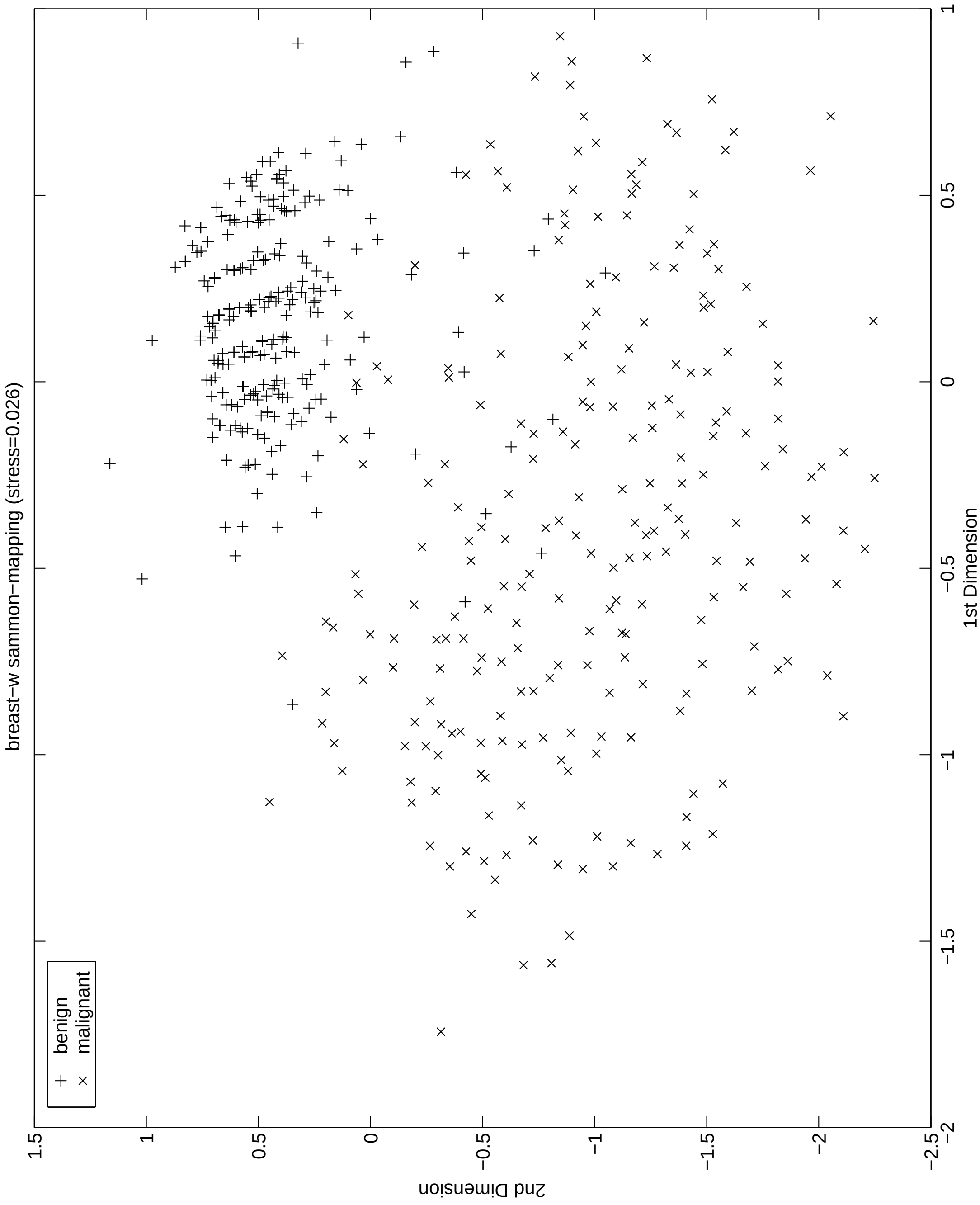


Figure A.9: Instance space visualization of *breast-w* via Sammon-Mapping. A cluster of benign examples, surrounded by mostly malginant examples is clearly visible. The stress value for this visualization is also quite small – values around 0.1 are usually observed.

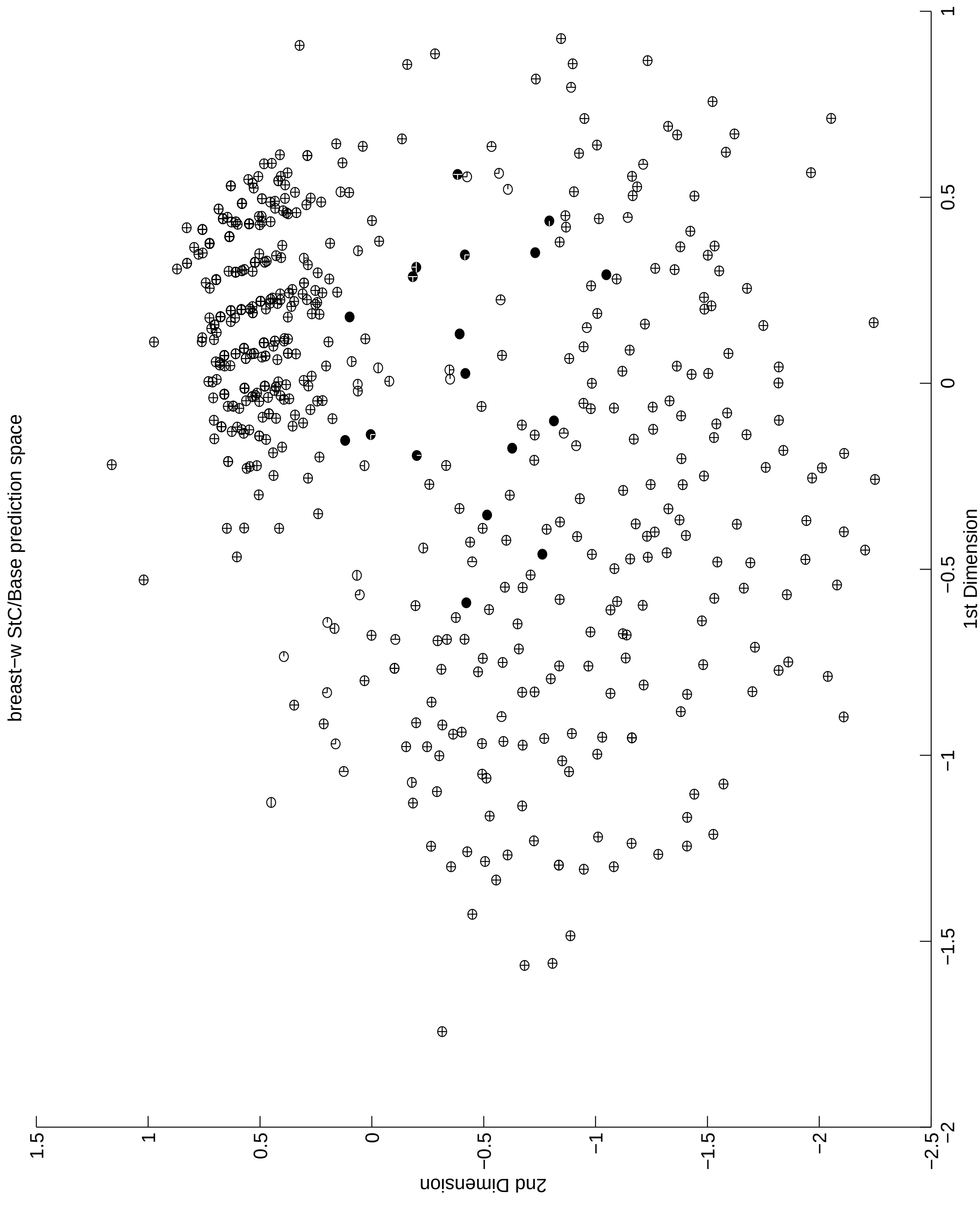


Figure A.10: Prediction space visualization of *breast-w* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

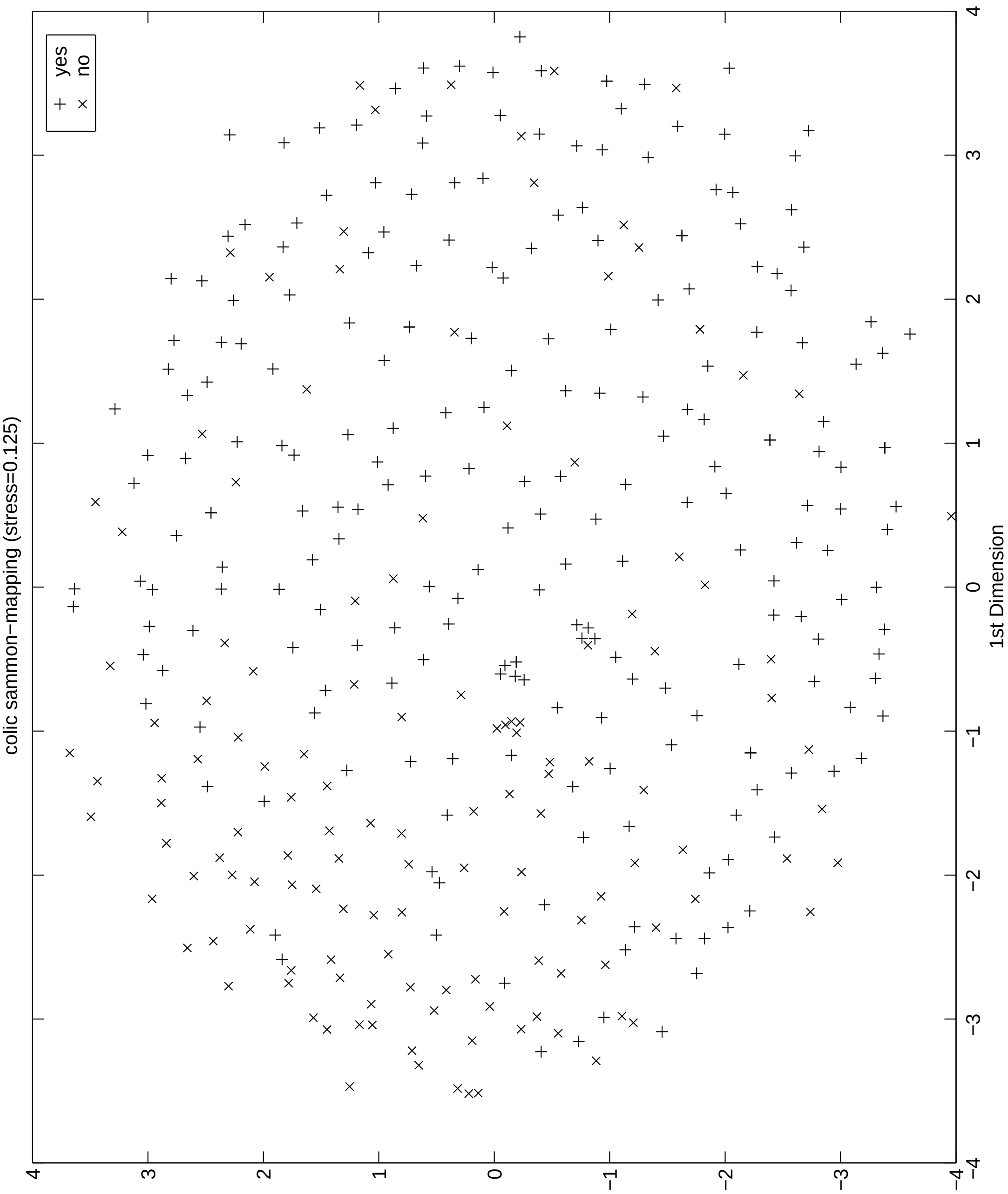


Figure A.11: Instance space visualization of *colic* via Sammon-Mapping. Towards negative values on the 1st dimension axis, a higher proportion of class no is observed. Apart from that, the distribution seems quite uniformly random.

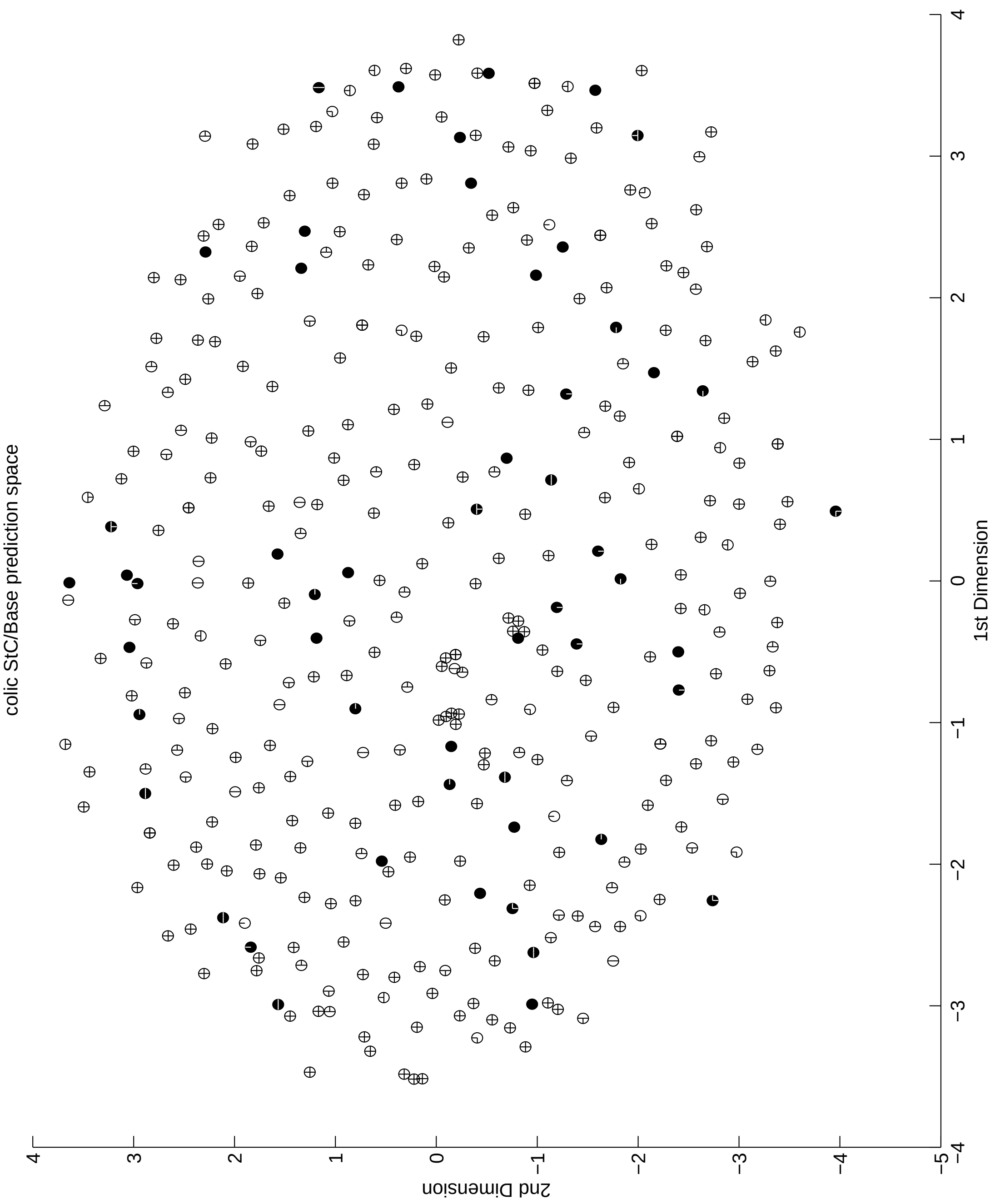


Figure A.12: Prediction space visualization of *colic* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

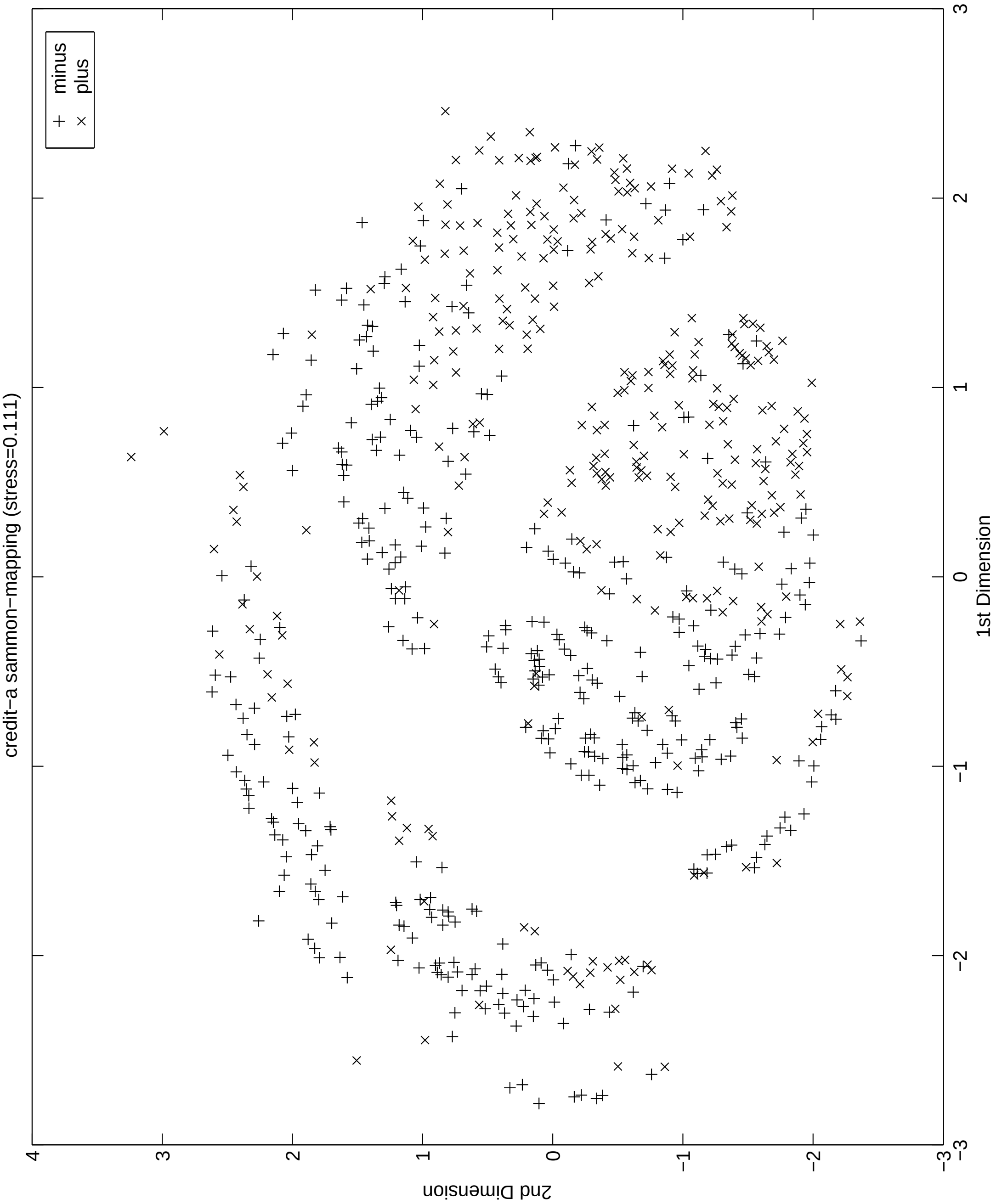


Figure A.13: Instance space visualization of *credit-a* via Sammon-Mapping. Some clusters with different proportions of plus and minus can be observed, with wide gaps between them. However, stress is still quite high, so this is of unclear significance.

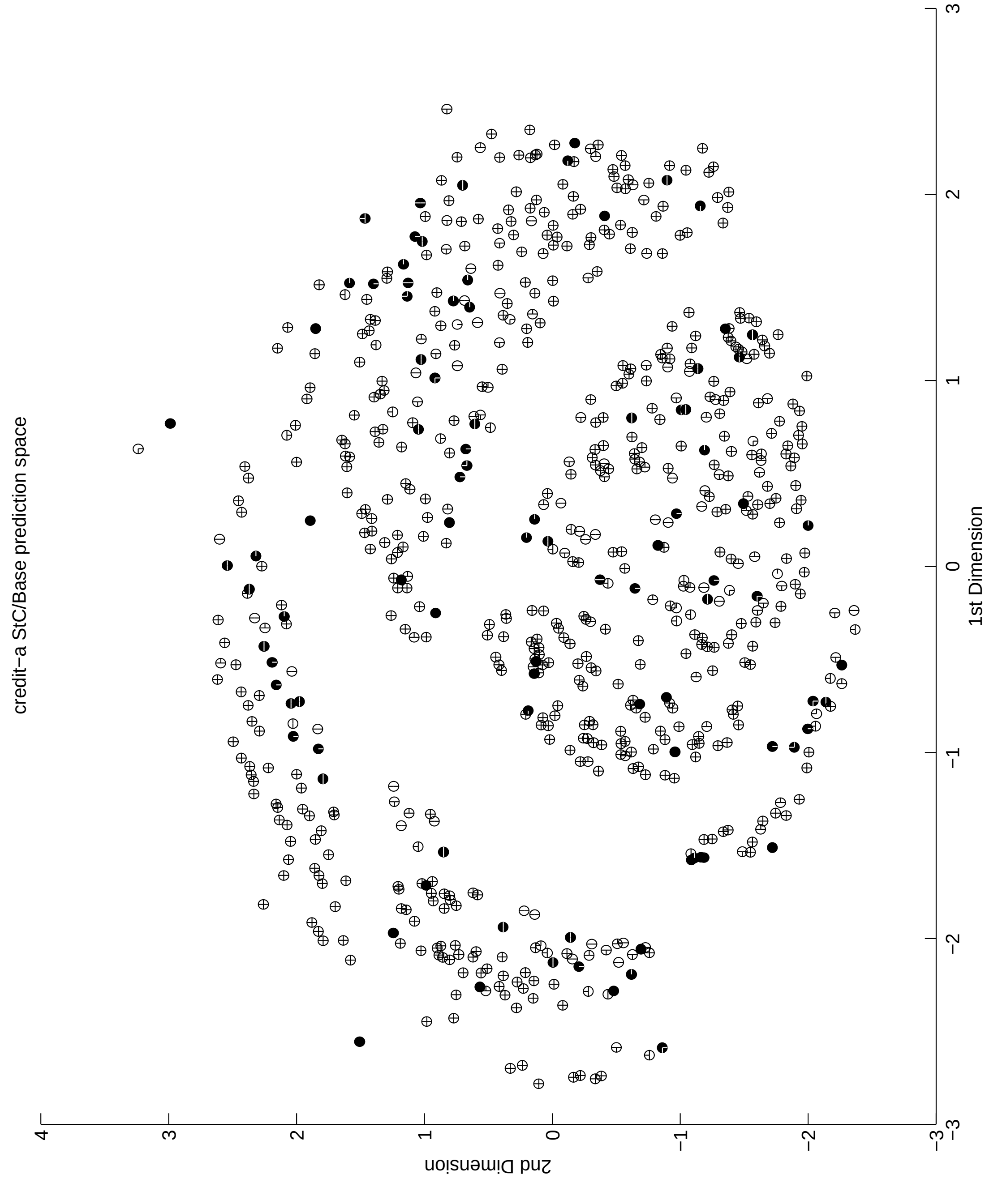


Figure A.14: Prediction space visualization of *credit-a* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

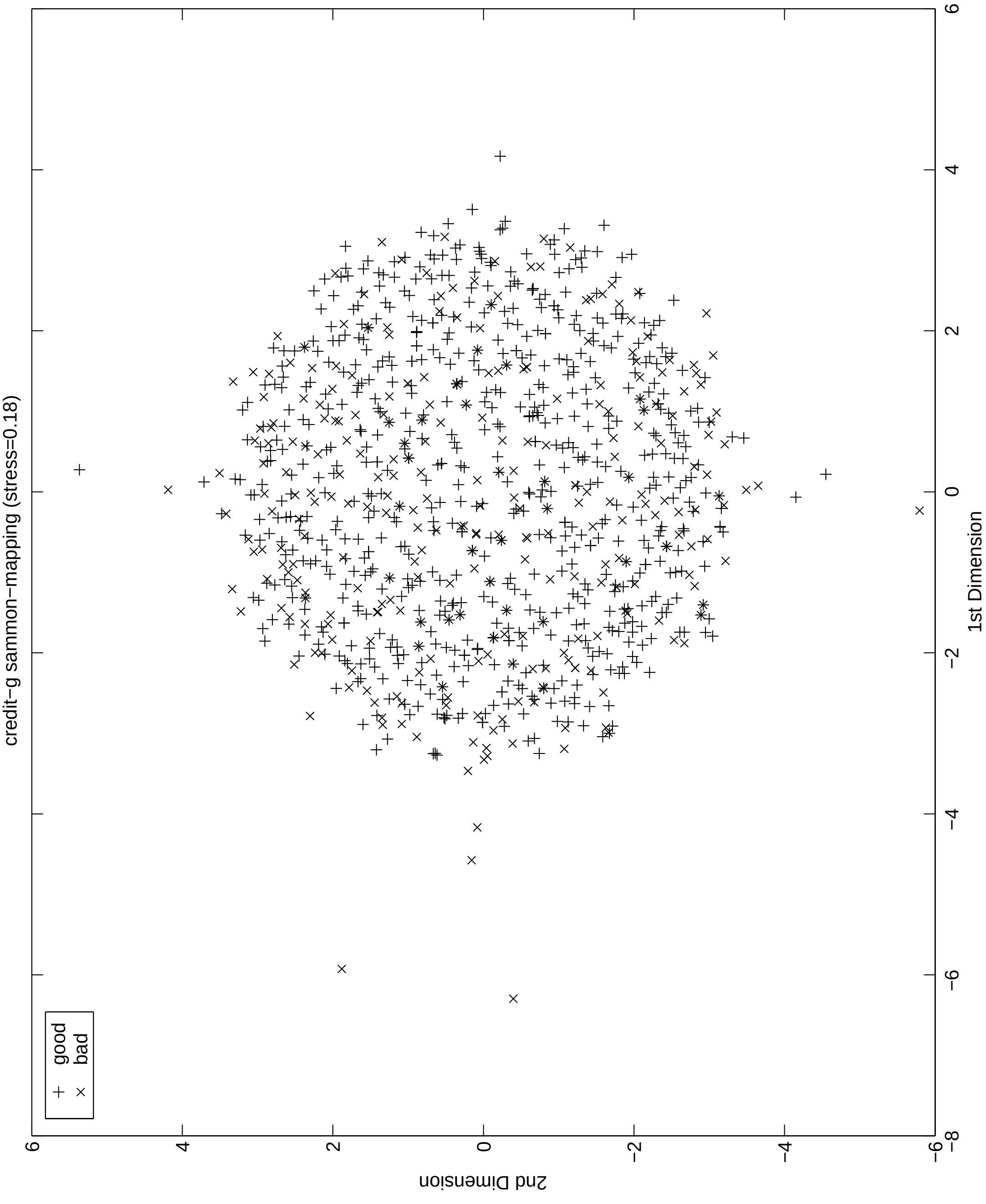


Figure A.15: Instance space visualization of *credit-g* via Sammon-Mapping. No observable clusters are visible, the examples are quite uniformly randomly distributed in a circle around the origin as would normally be expected only from random data.

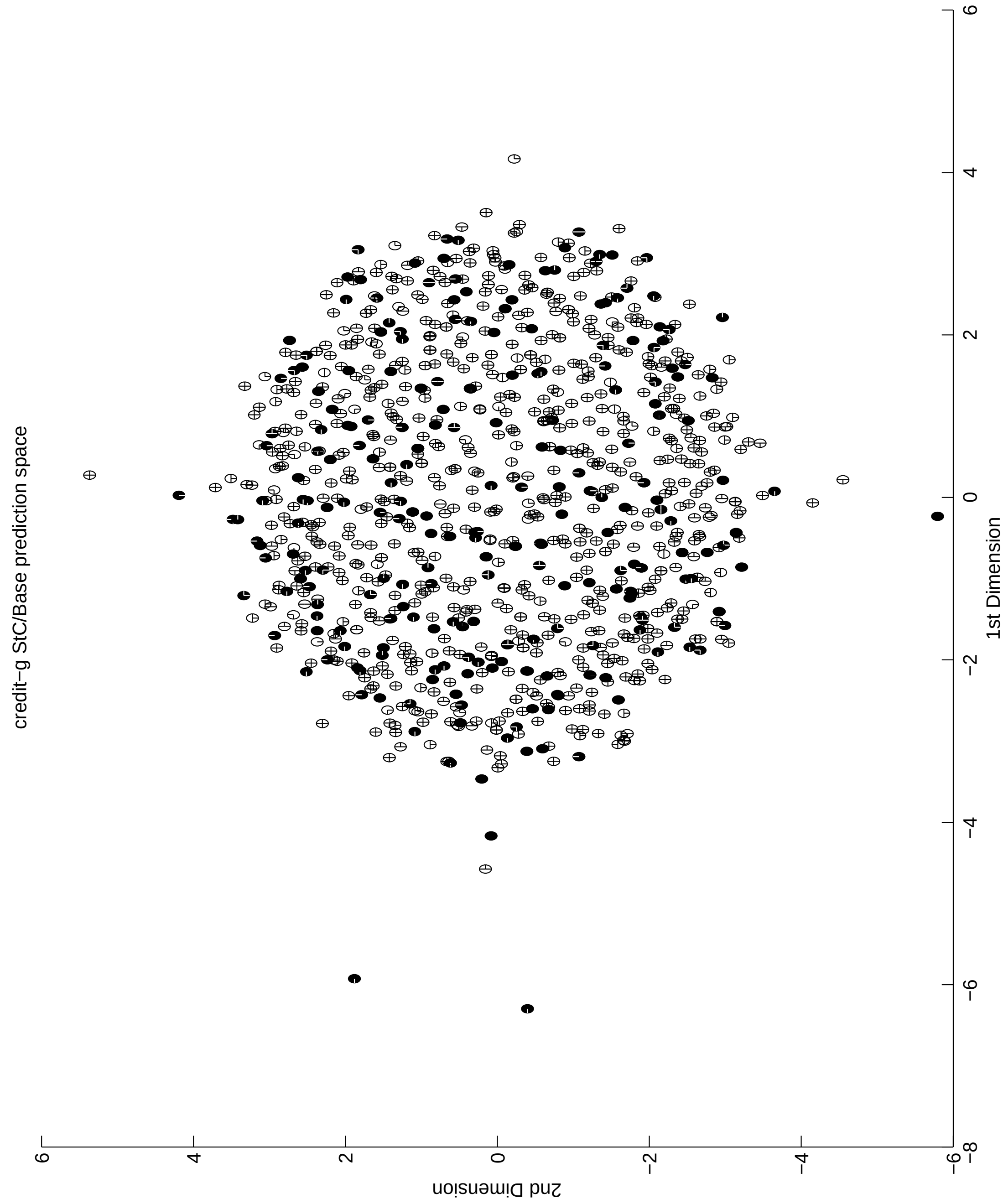


Figure A.16: Prediction space visualization of *credit-g* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

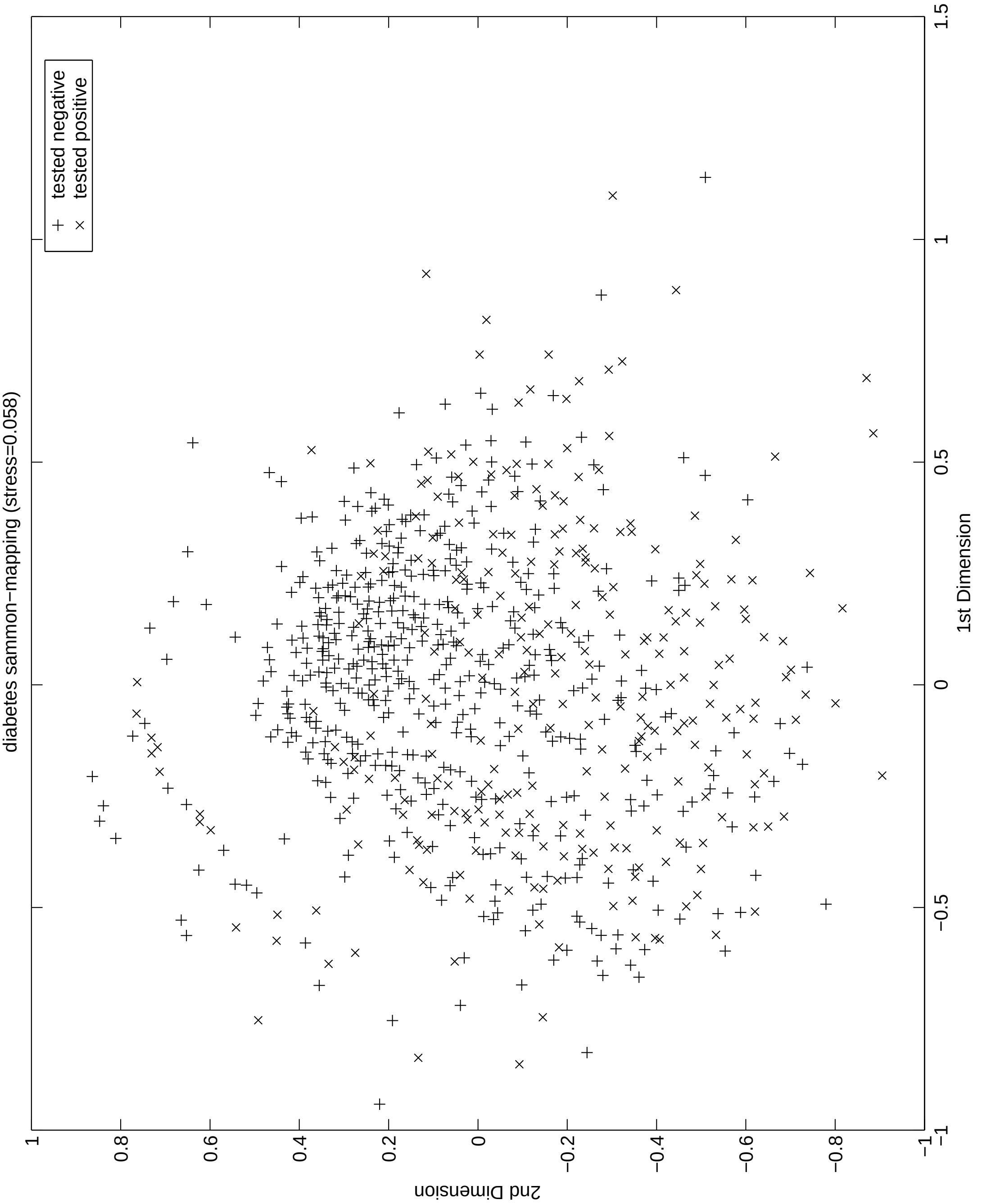


Figure A.17: Instance space visualization of *diabetes* via Sammon-Mapping. Some structure in the example distribution is observed. It seems that there are slightly more tested positive examples towards the negative 2nd dimension.

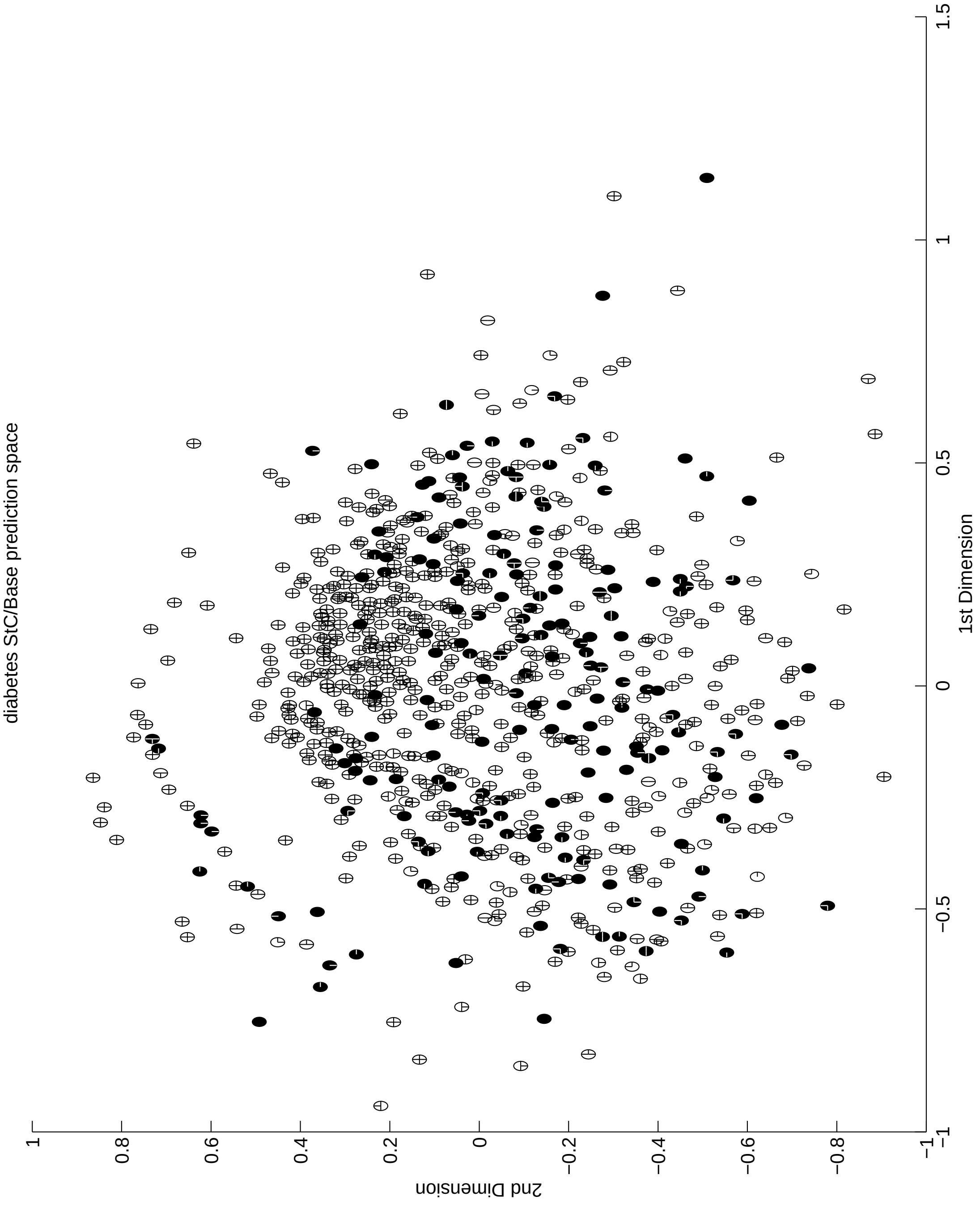


Figure A.18: Prediction space visualization of *diabetes* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

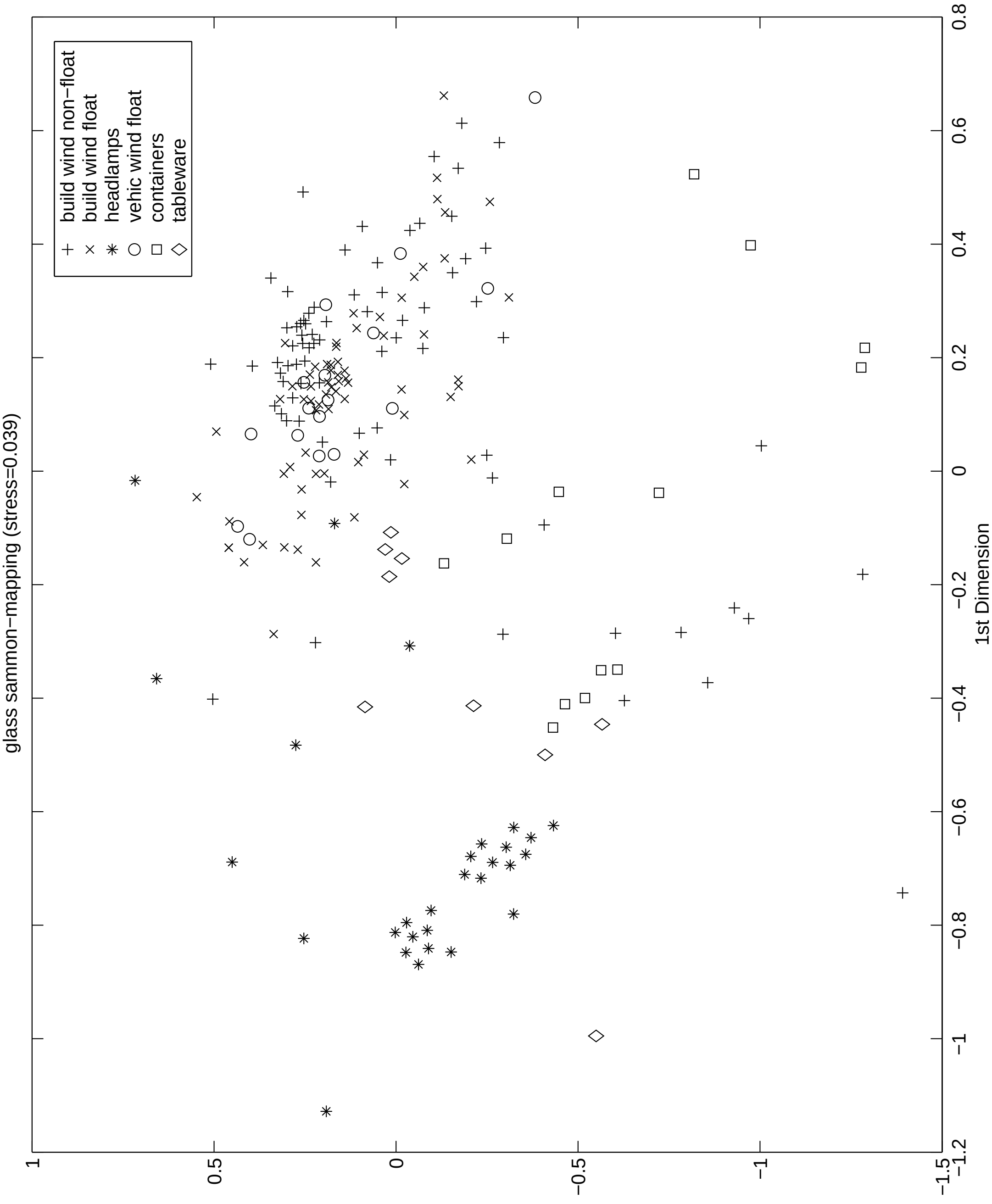


Figure A.19: Instance space visualization of *glass* via Sammon-Mapping. The examples are very non-uniformly distributed. There is one cluster, headlamps; all the others overlap to a lesser or greater degree. One class does not appear in the dataset.

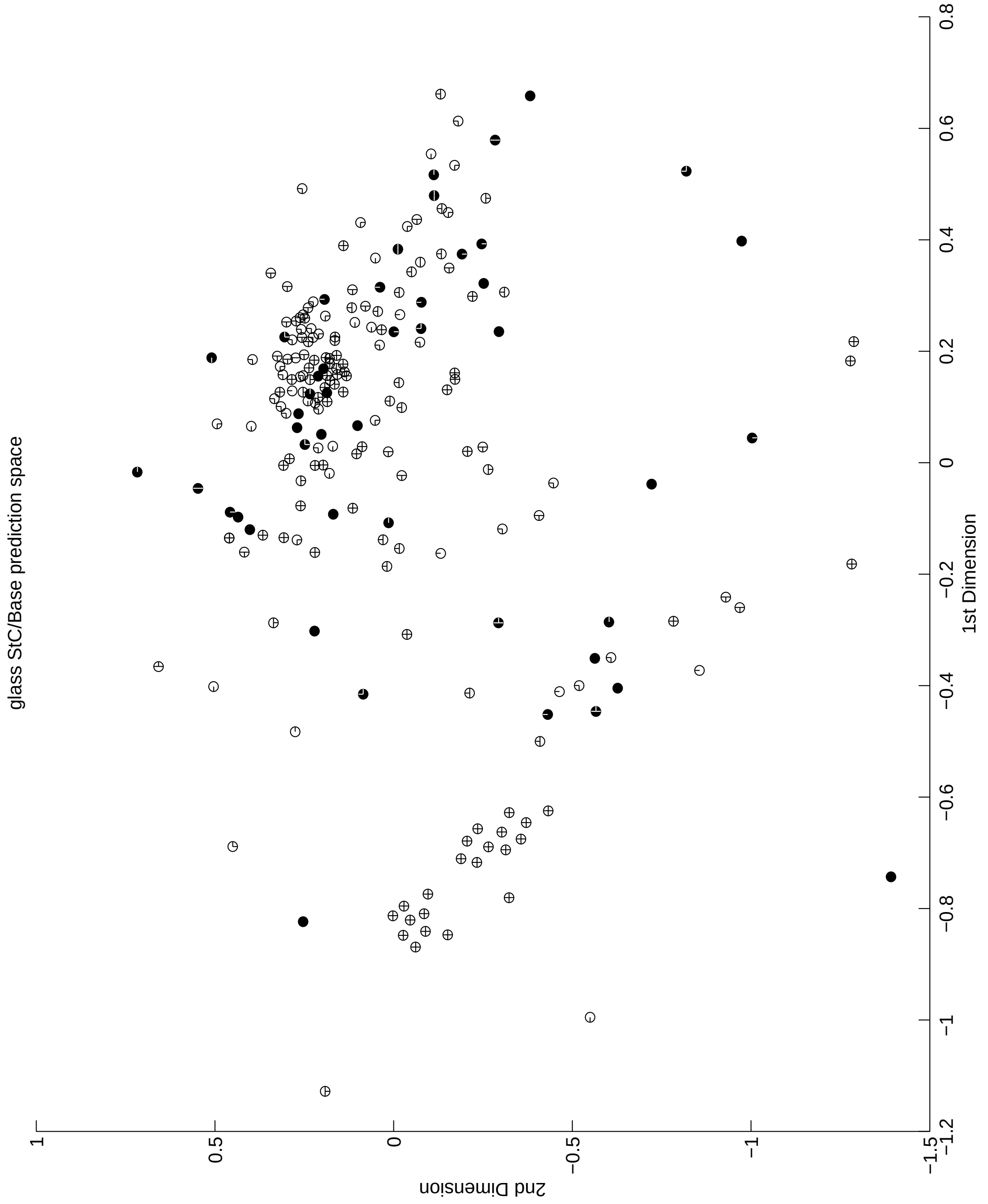


Figure A.20: Prediction space visualization of *glass* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

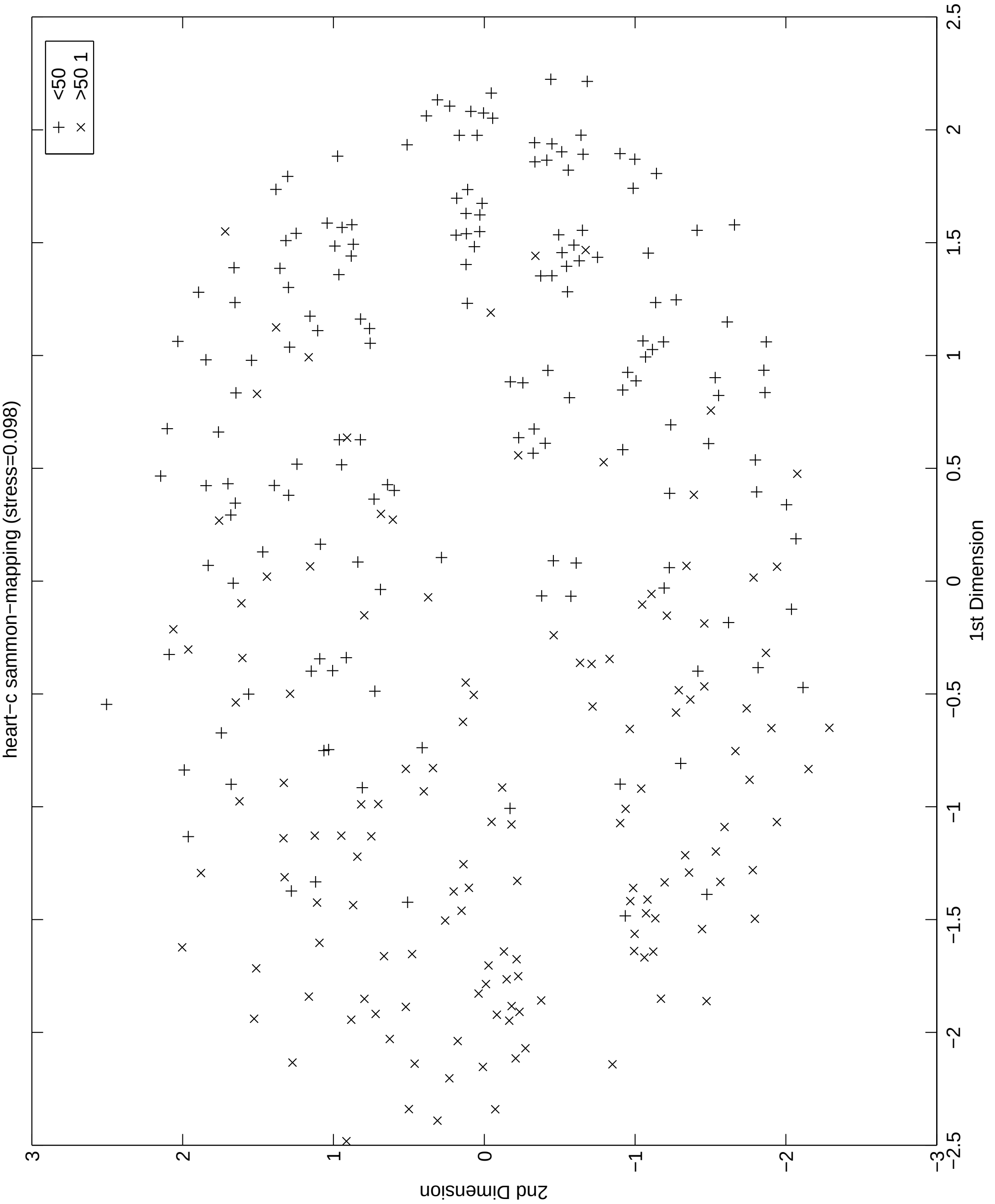


Figure A.21: Instance space visualization of *heart-c* via Sammon-Mapping. Towards the negative 1st dimension, more > 50 1 examples can be found. Some structure seems to be present in the distribution of examples. What is more striking, however, is that this dataset has five class values in the header, but only two of the classes are really used in the examples!

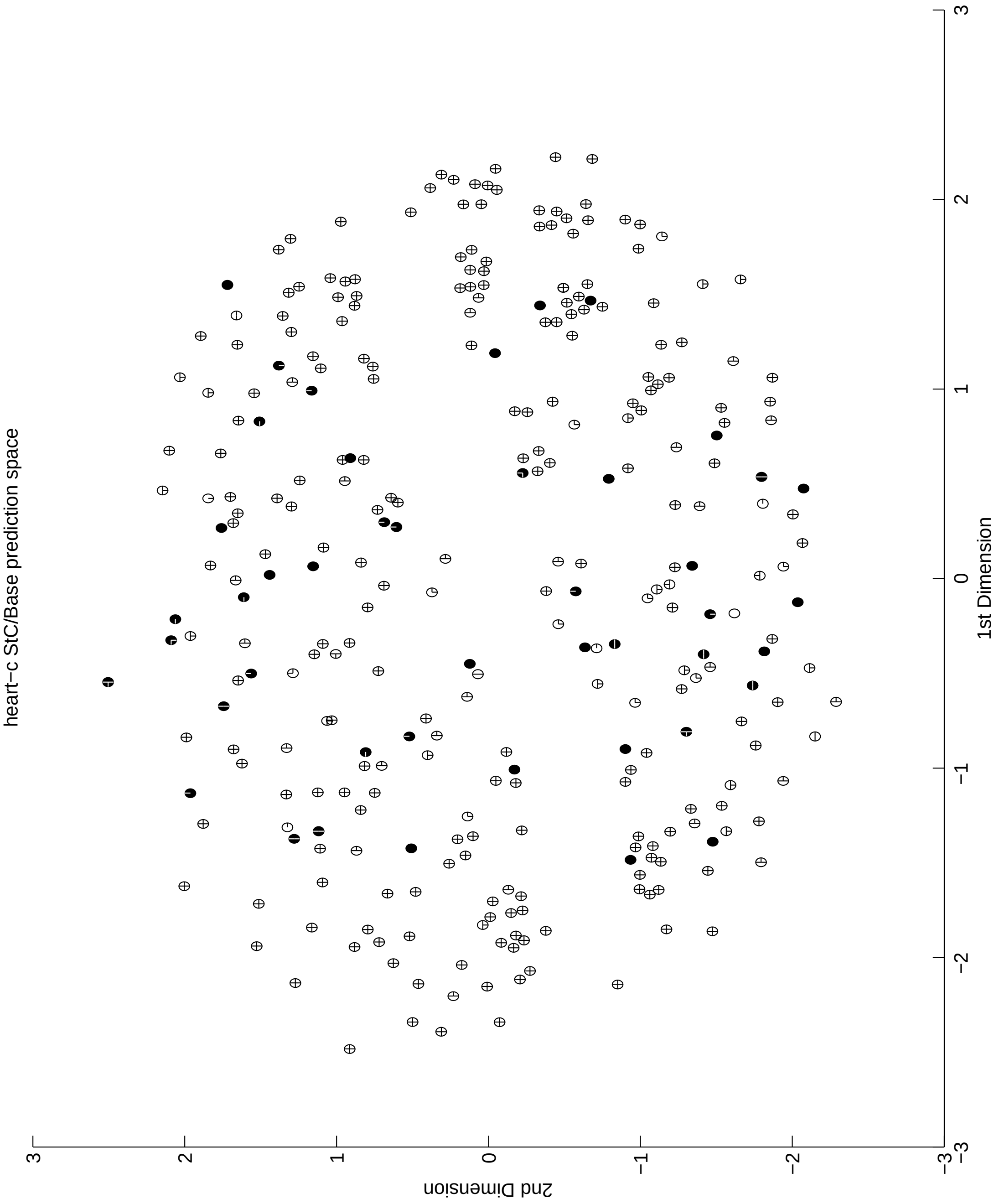


Figure A.22: Prediction space visualization of *heart-c* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

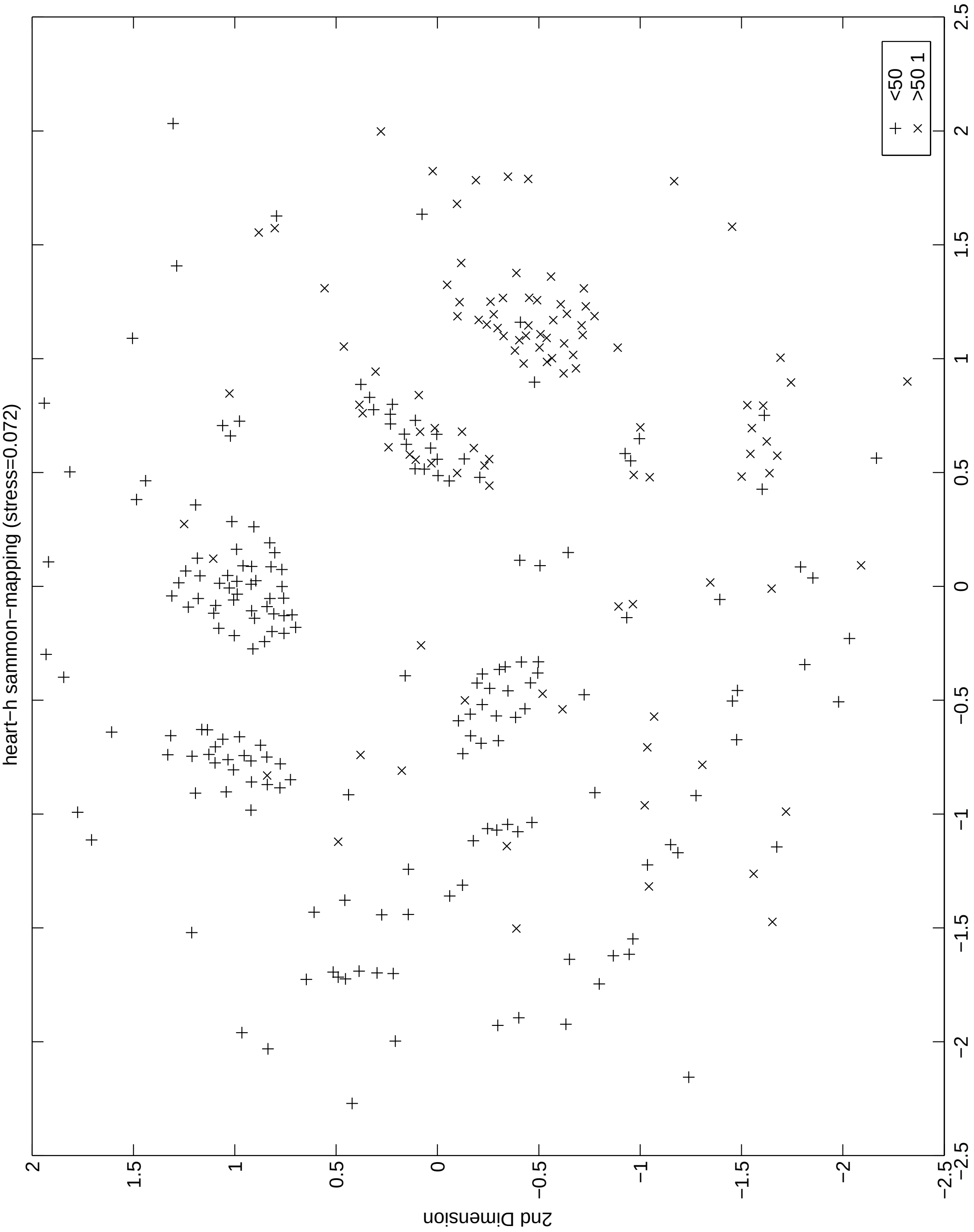


Figure A.23: Instance space visualization of *heart-h* via Sammon-Mapping. There are some clusters with different proportions of < 50 and > 50 1, but most of the data seems randomly distributed. Towards the positive 1st dimension, more > 50 1 examples can be found. However, again this is a supposedly five-class dataset where only two classes are used in the examples.

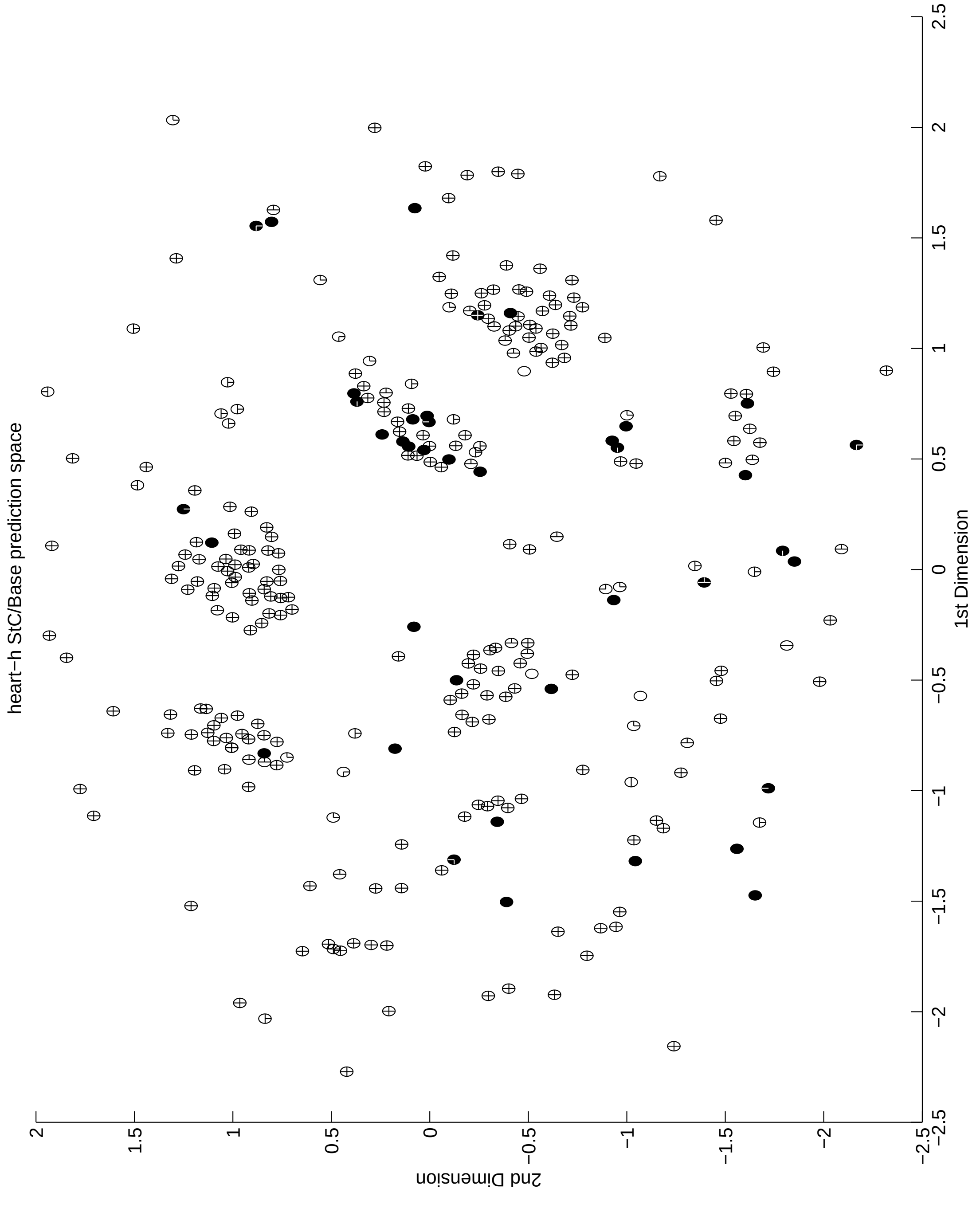


Figure A.24: Prediction space visualization of *heart-h* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

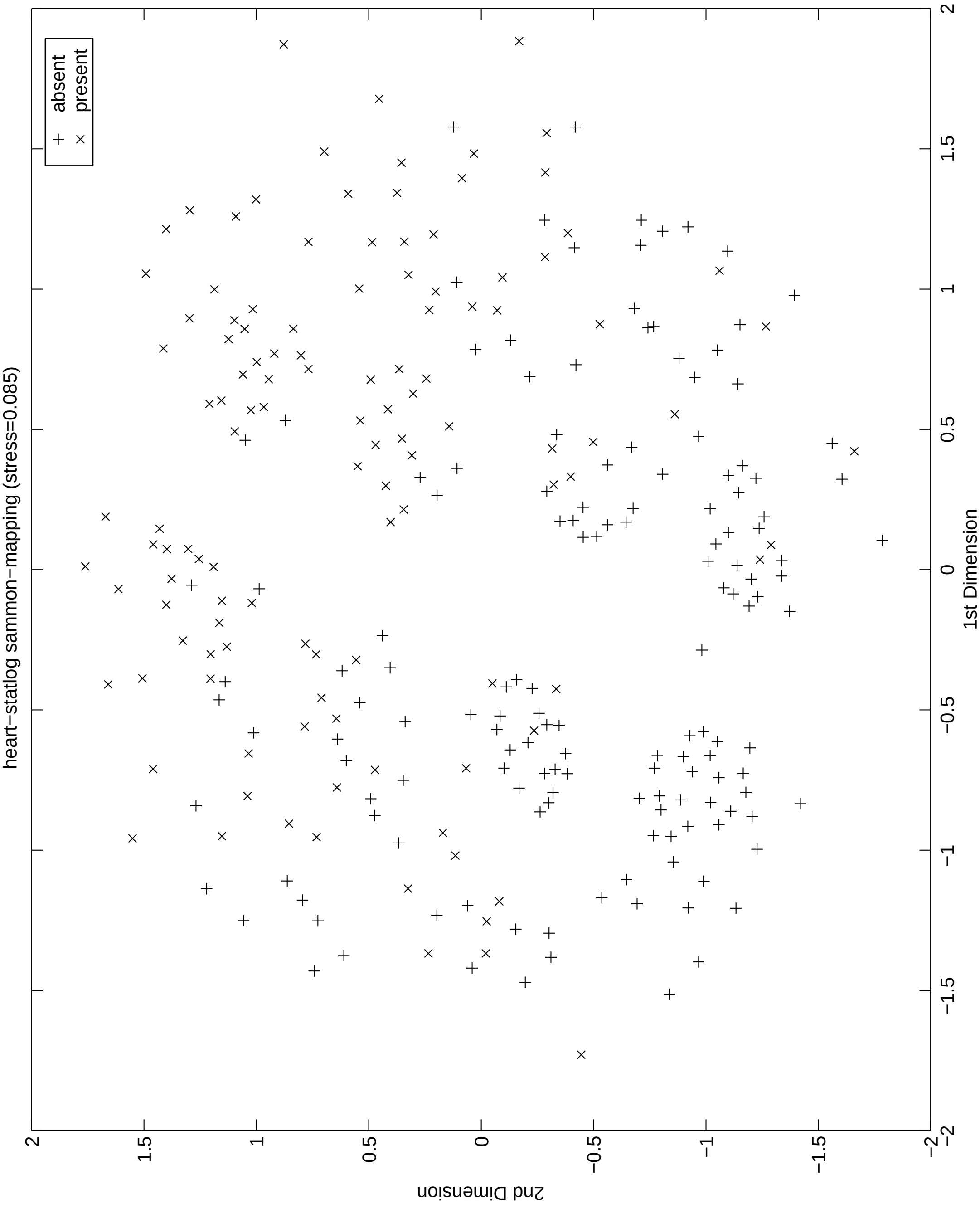


Figure A.25: Instance space visualization of *heart-statlog* via Sammon-Mapping. Some interesting structure seems to be present in the distribution of examples.

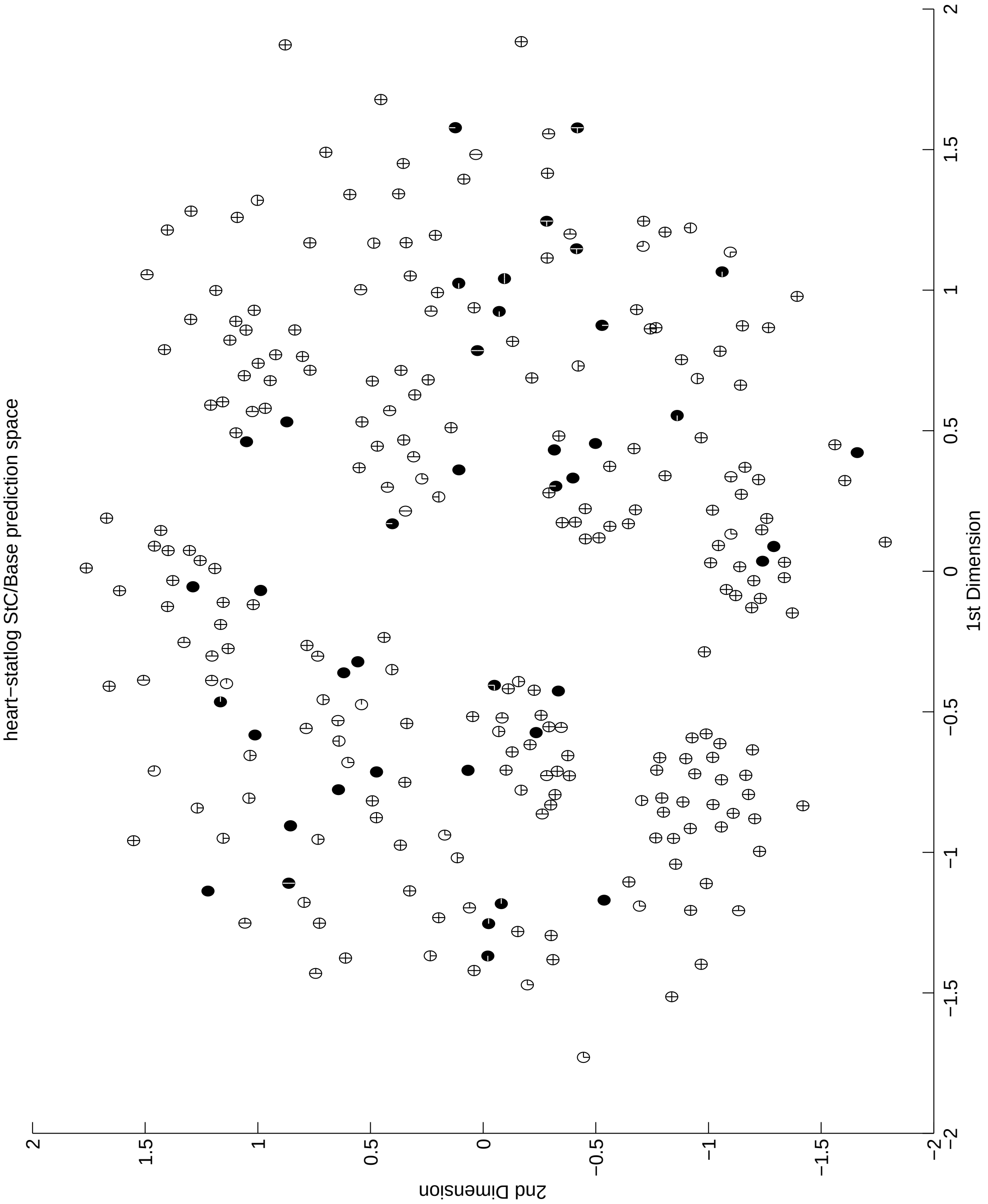


Figure A.26: Prediction space visualization of *heart-statlog* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

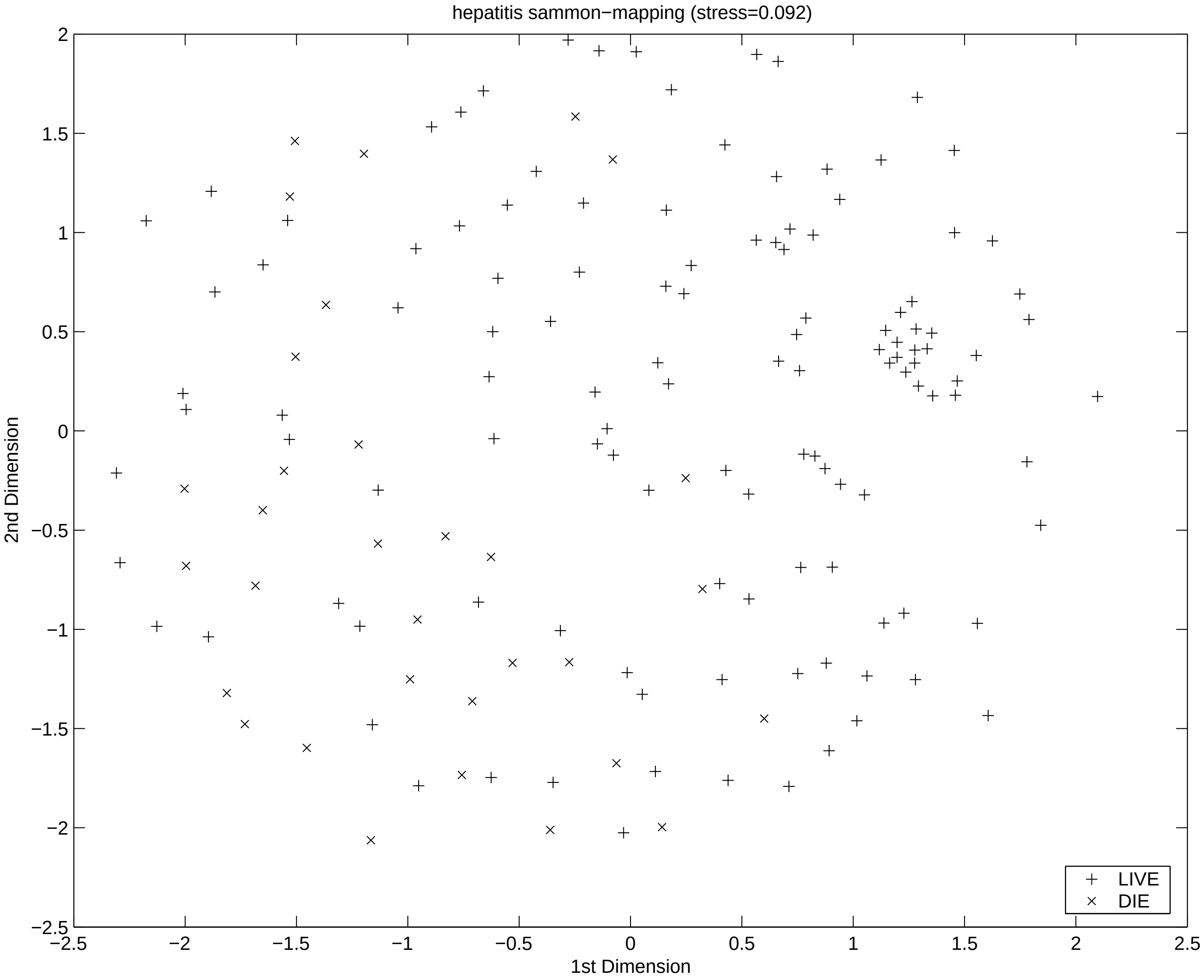


Figure A.27: Instance space visualization of *hepatitis* via Sammon-Mapping. This dataset seems quite sparse.

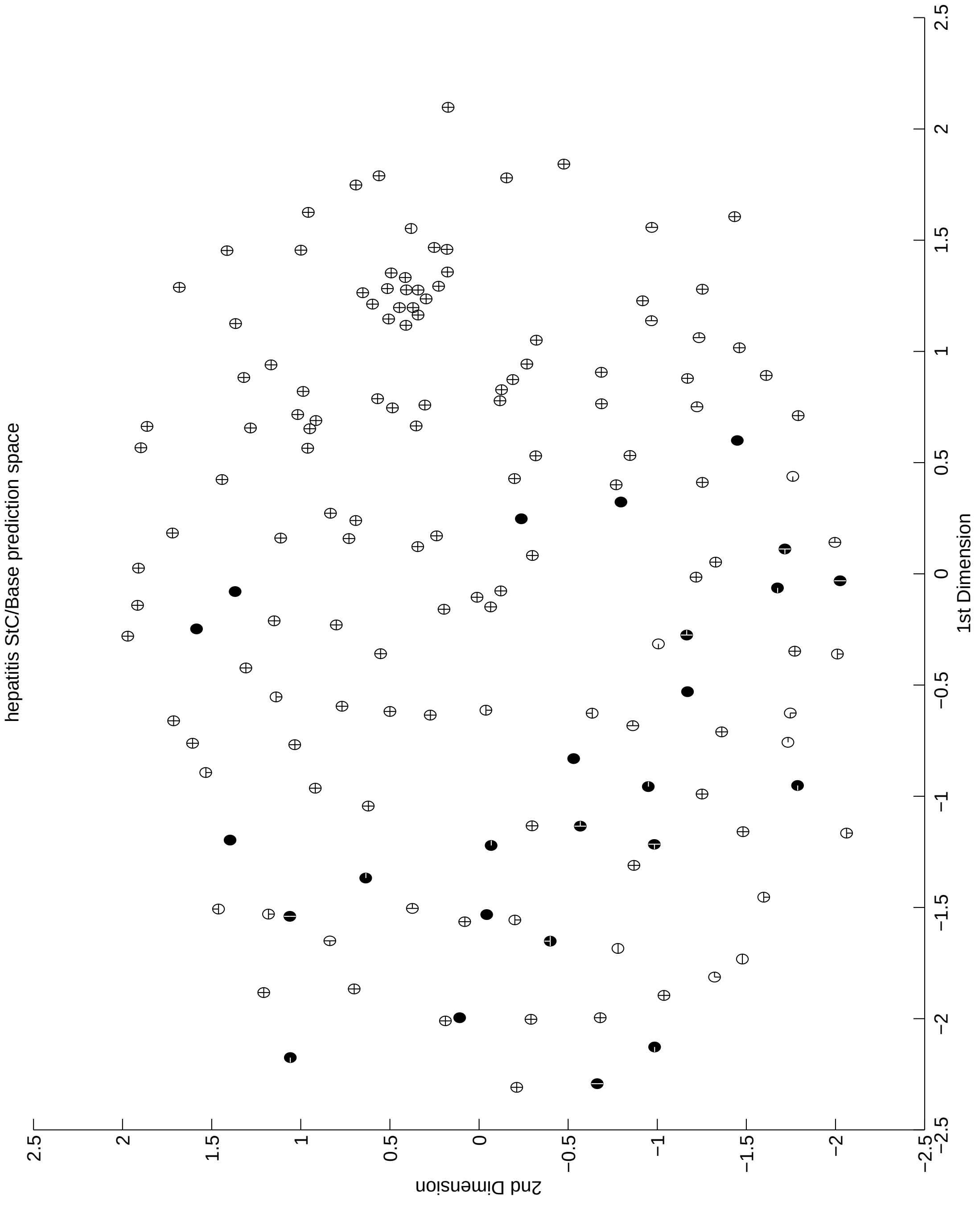


Figure A.28: Prediction space visualization of *hepatitis* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

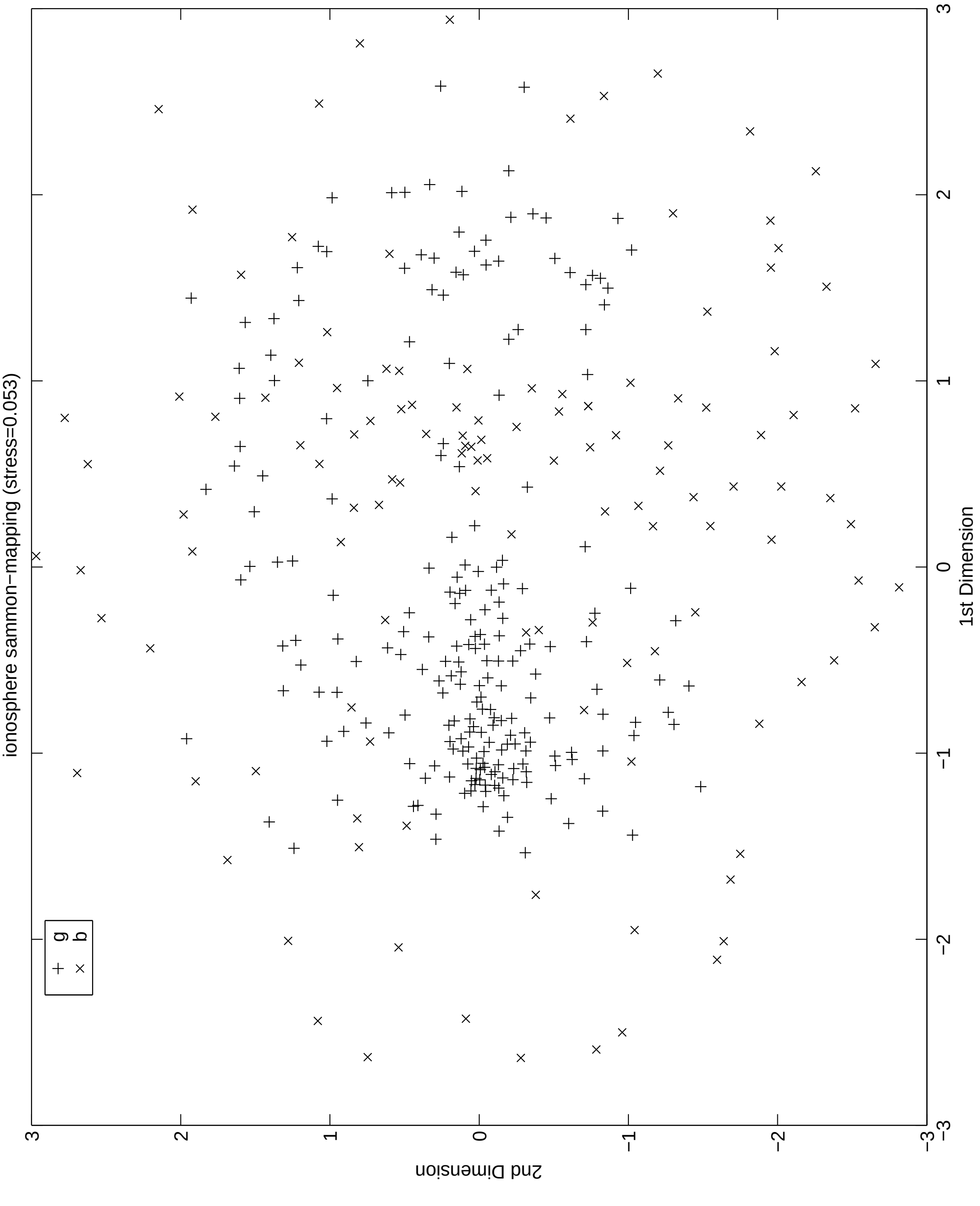


Figure A.29: Instance space visualization of *ionosphere* via Sammon-Mapping. Some interesting structure seems to be present in the distribution of examples.

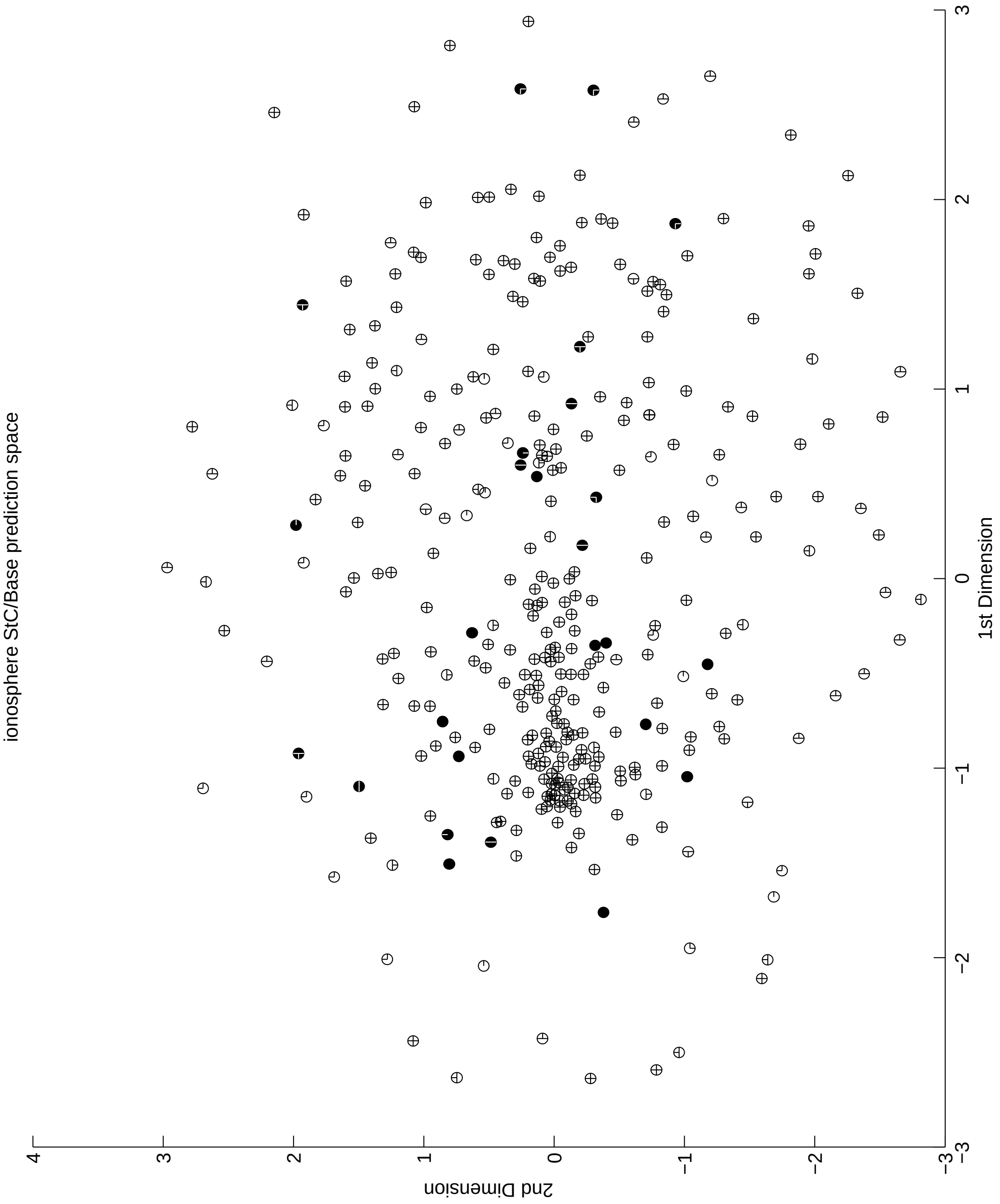


Figure A.30: Prediction space visualization of *ionosphere* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

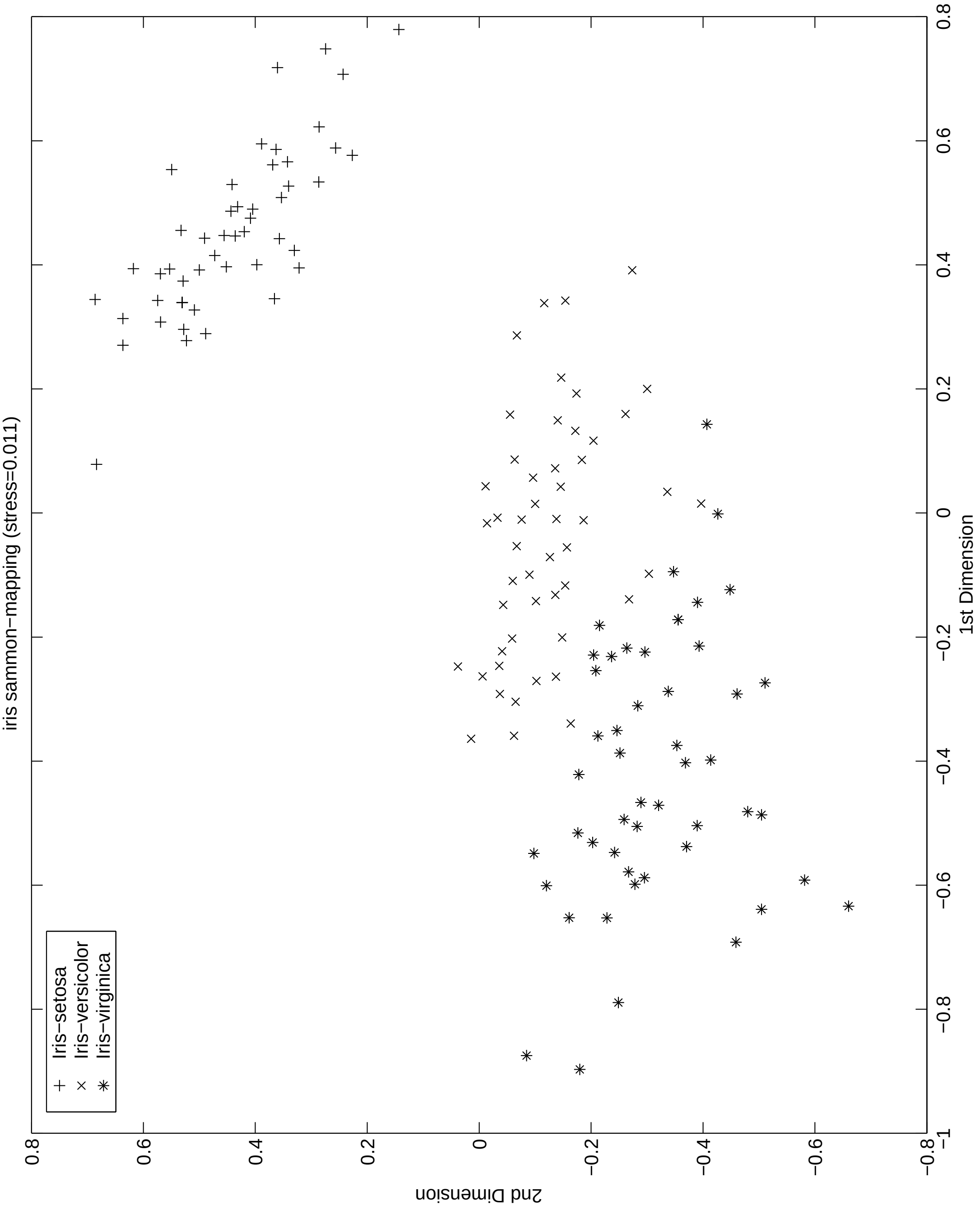


Figure A.31: Instance space visualization of *iris* via Sammon-Mapping. The clusters are almost perfect and are able to differentiate the classes – without relying on previous class information.

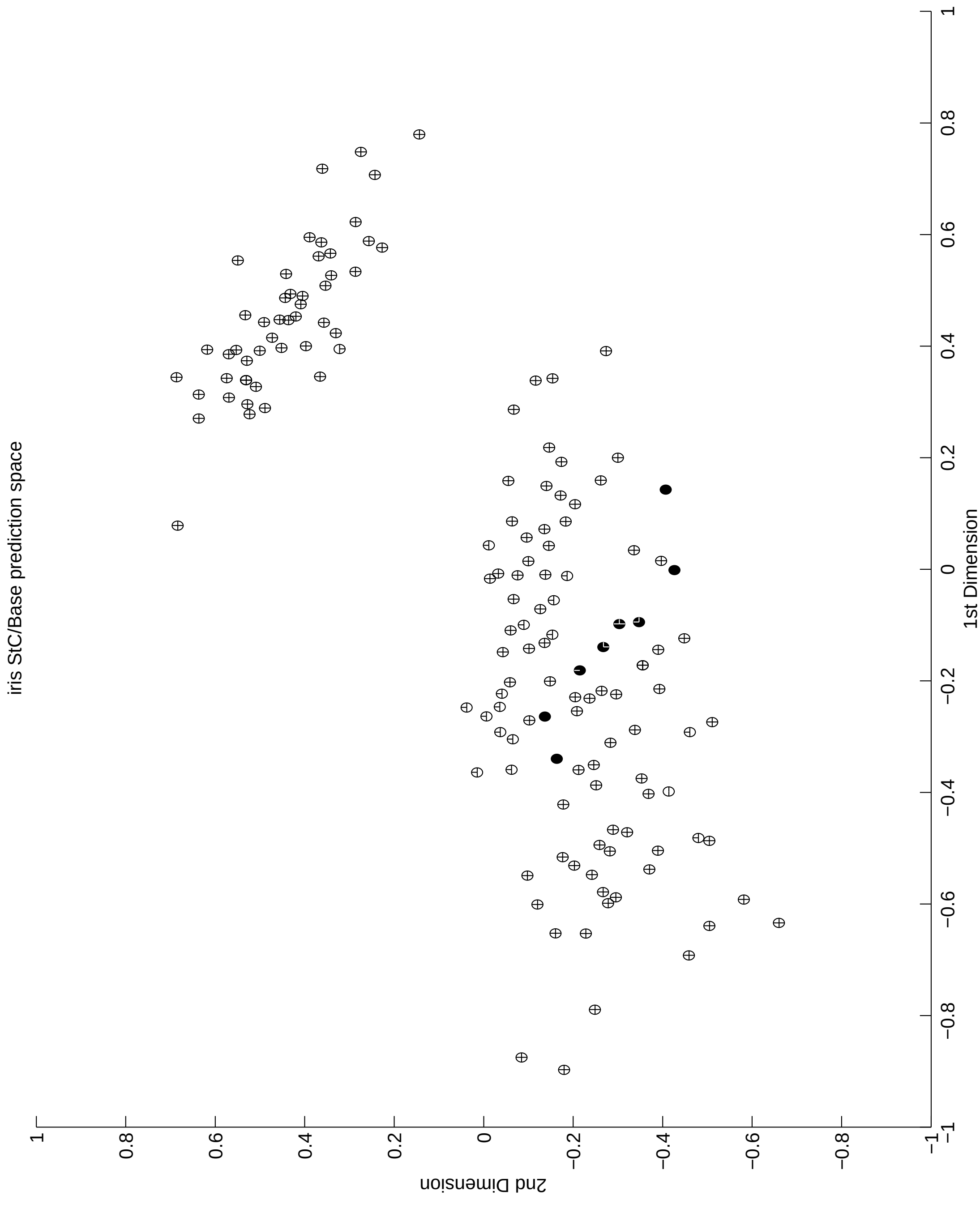


Figure A.32: Prediction space visualization of *iris* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

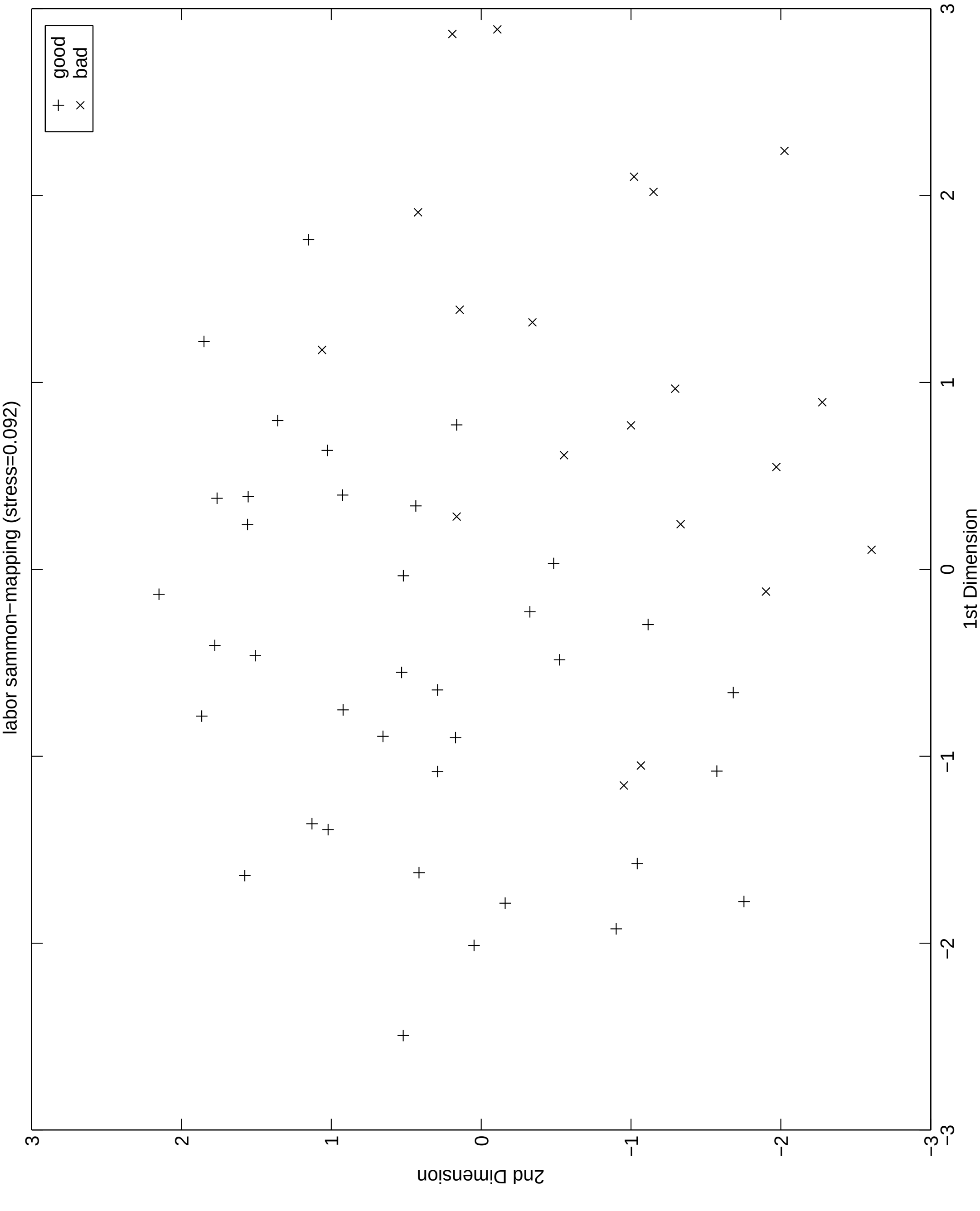


Figure A.33: Instance space visualization of *labor* via Sammon-Mapping. The examples are very sparse – in fact, this is our smallest dataset.

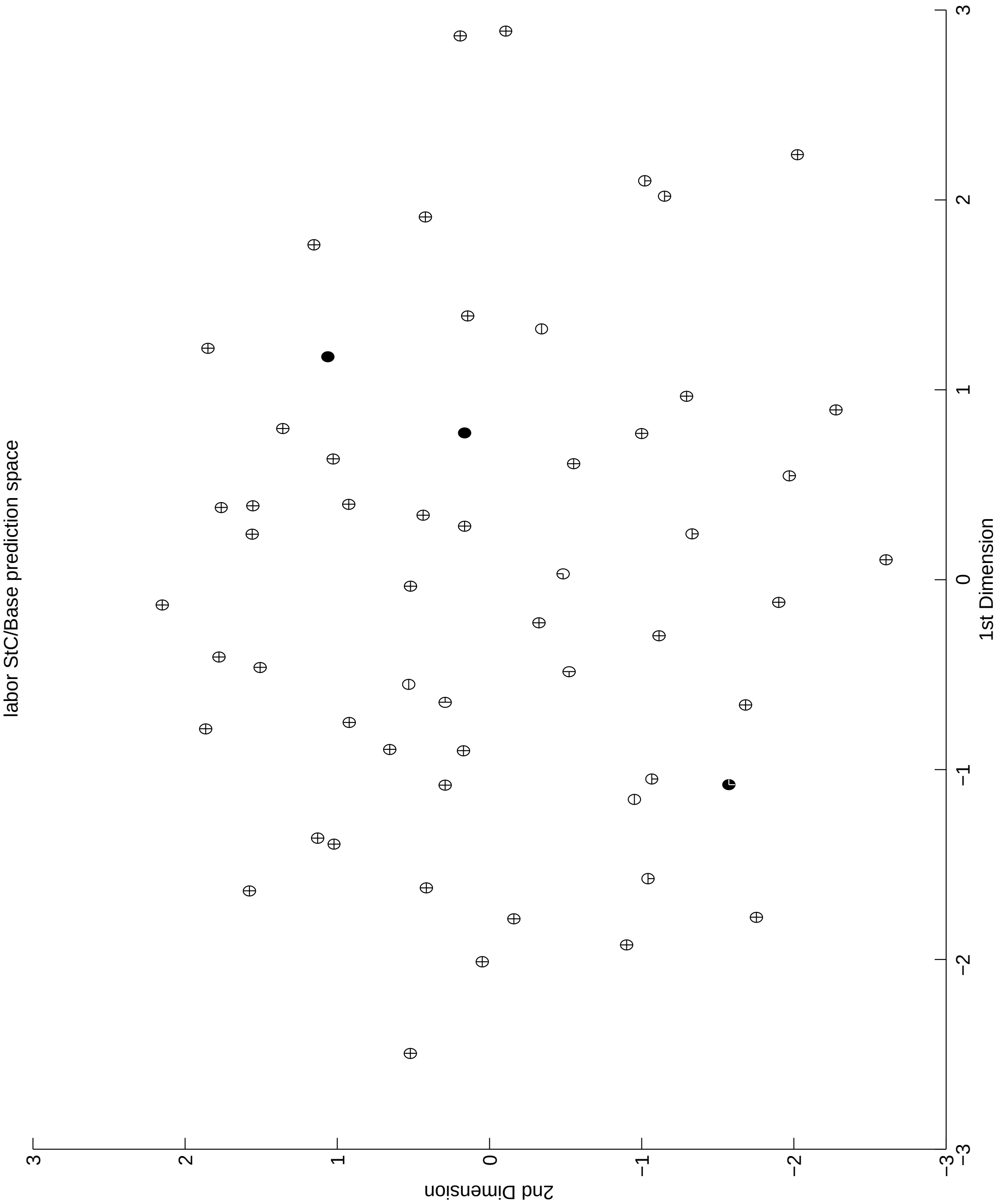


Figure A.34: Prediction space visualization of *labor* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

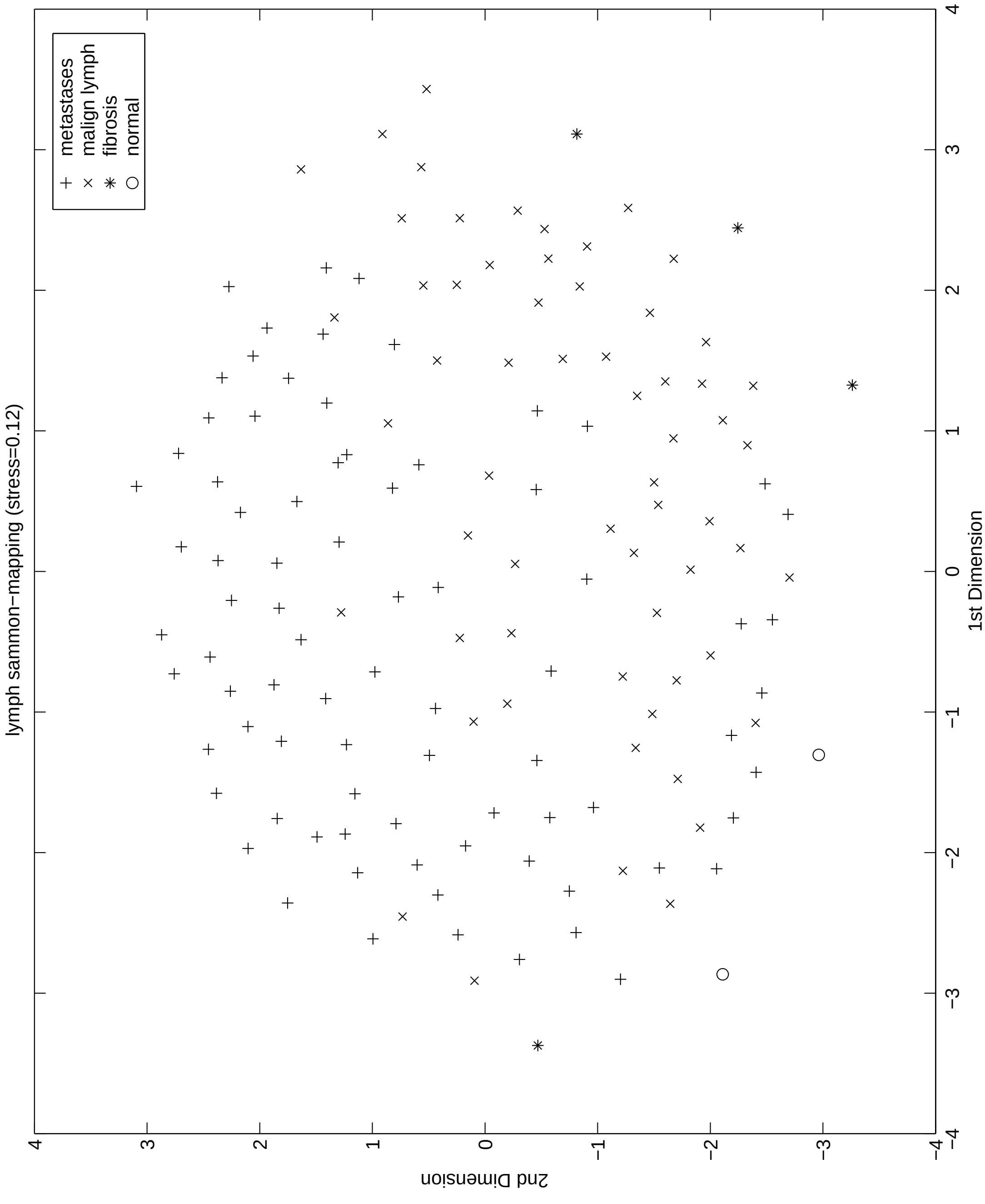


Figure A.35: Instance space visualization of *lymph* via Sammon-Mapping. This dataset is also quite sparse. Notice that the classes normal and fibrosis have only two resp. 4 examples. This is clearly insufficient data for any kind of meaningful learning.

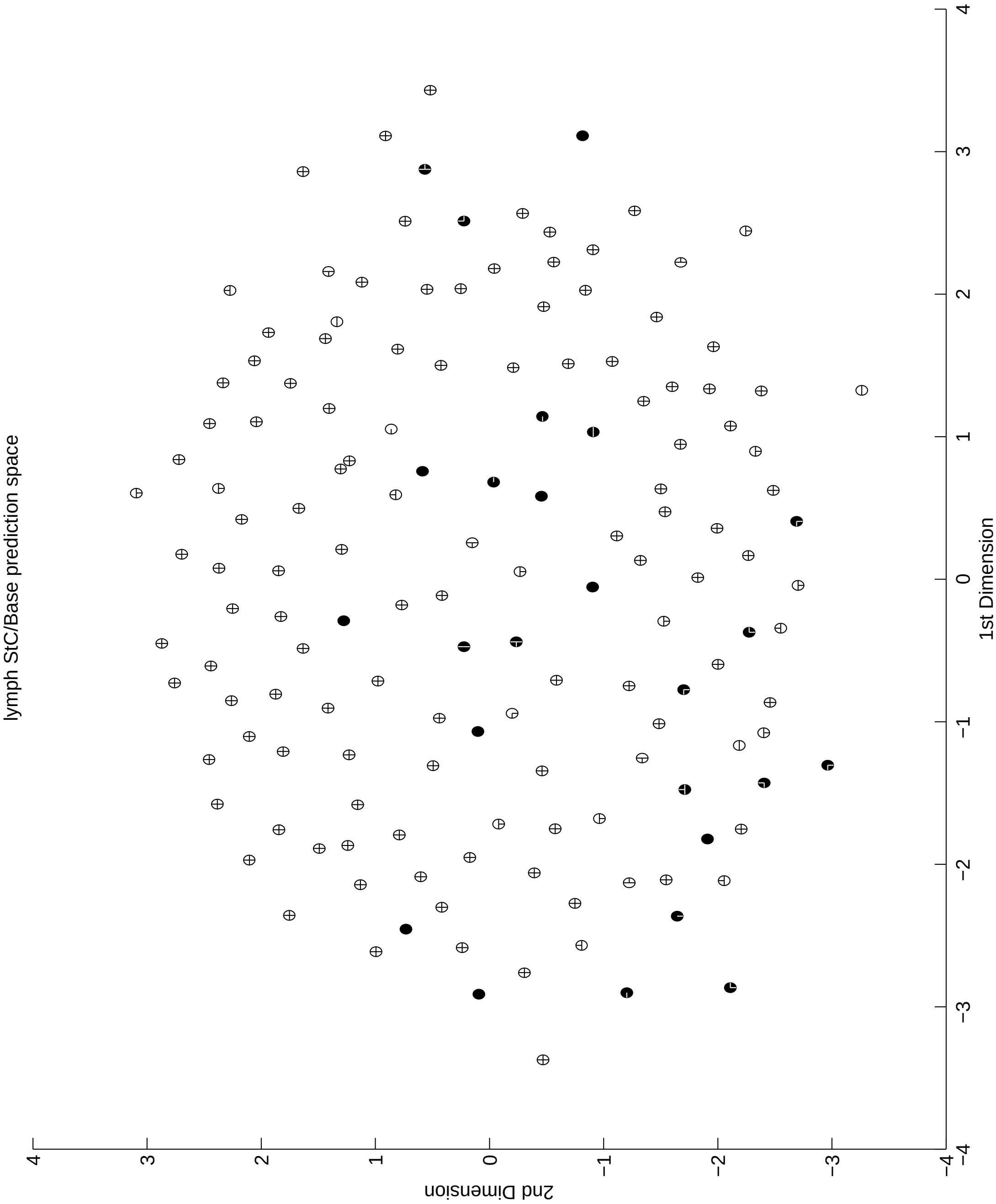


Figure A.36: Prediction space visualization of *lymph* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

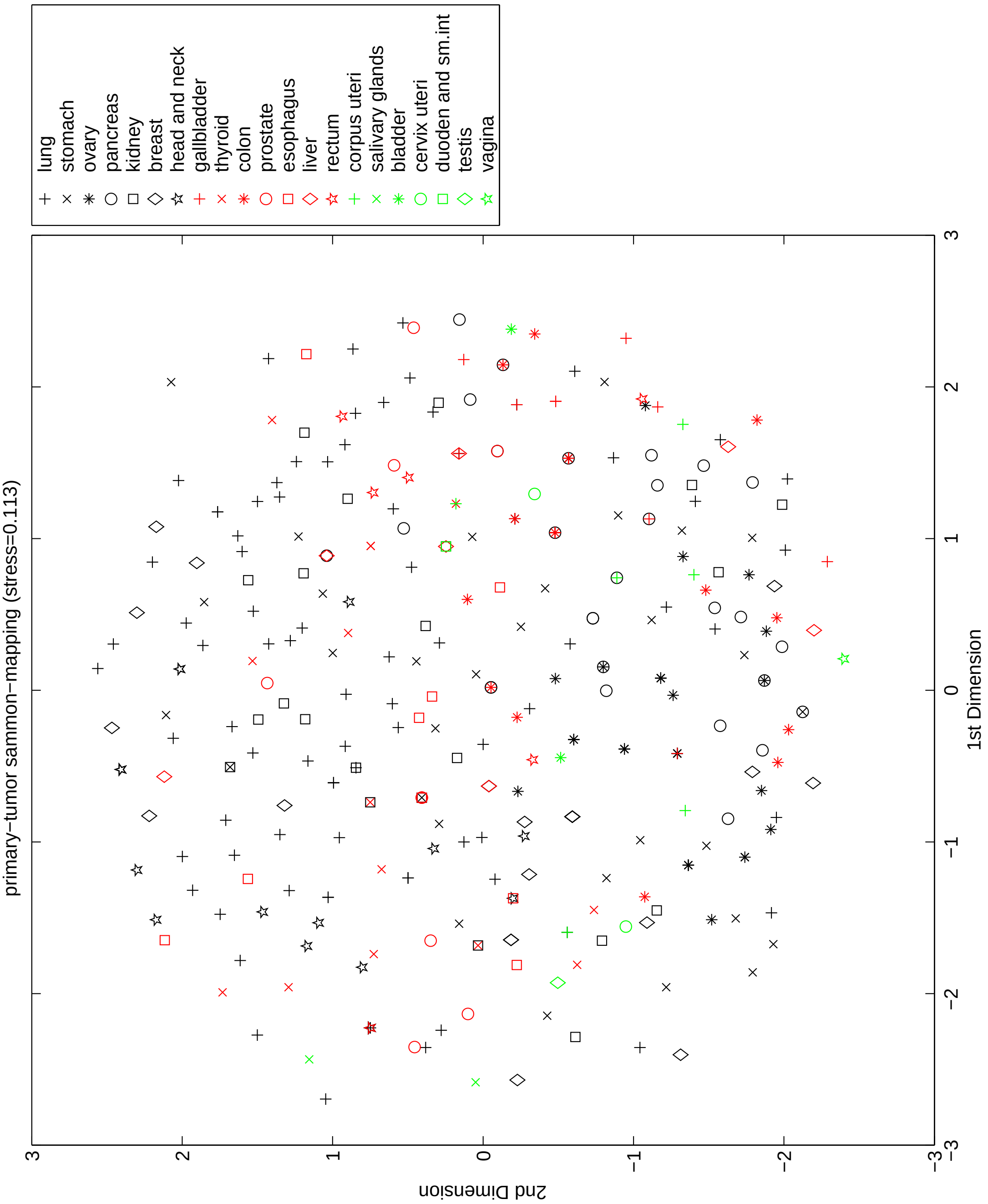


Figure A.37: Instance space visualization of *primary-tumor* via Sammon-Mapping. The visualization seems quite confusing and colorful, because of the twenty-four different classes. One of these classes does not even appear in the full dataset. Only *one* example is present for the three least-frequent classes; two are present for the next three and six for corpus uteri. Learning requires at least one training example, two are the absolute minimum. Notice also that some of the classes seem related, e.g. corpus uteri and cervix uteri, which may be exploited by recoding the class values.

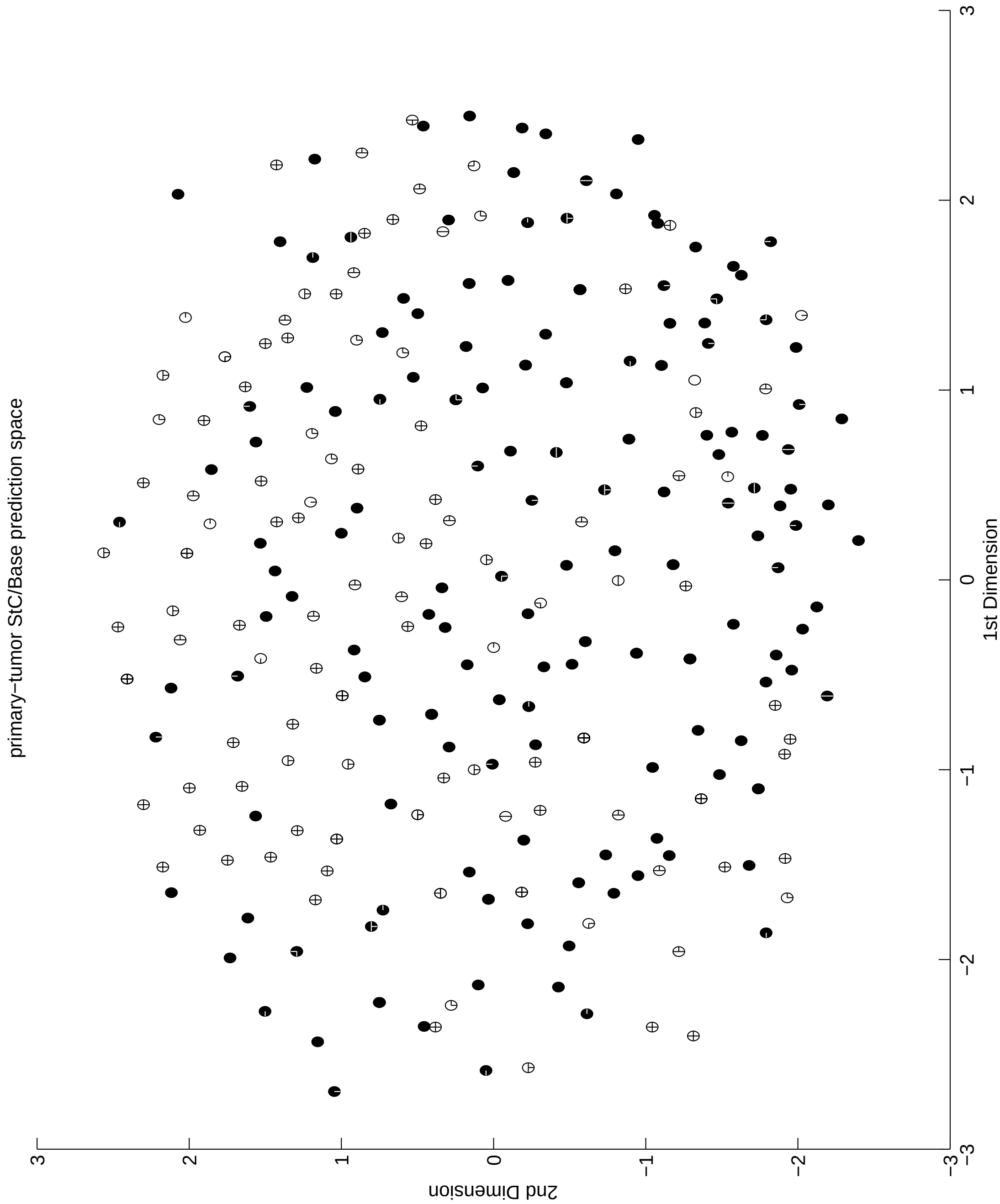


Figure A.38: Prediction space visualization of *primary-tumor* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

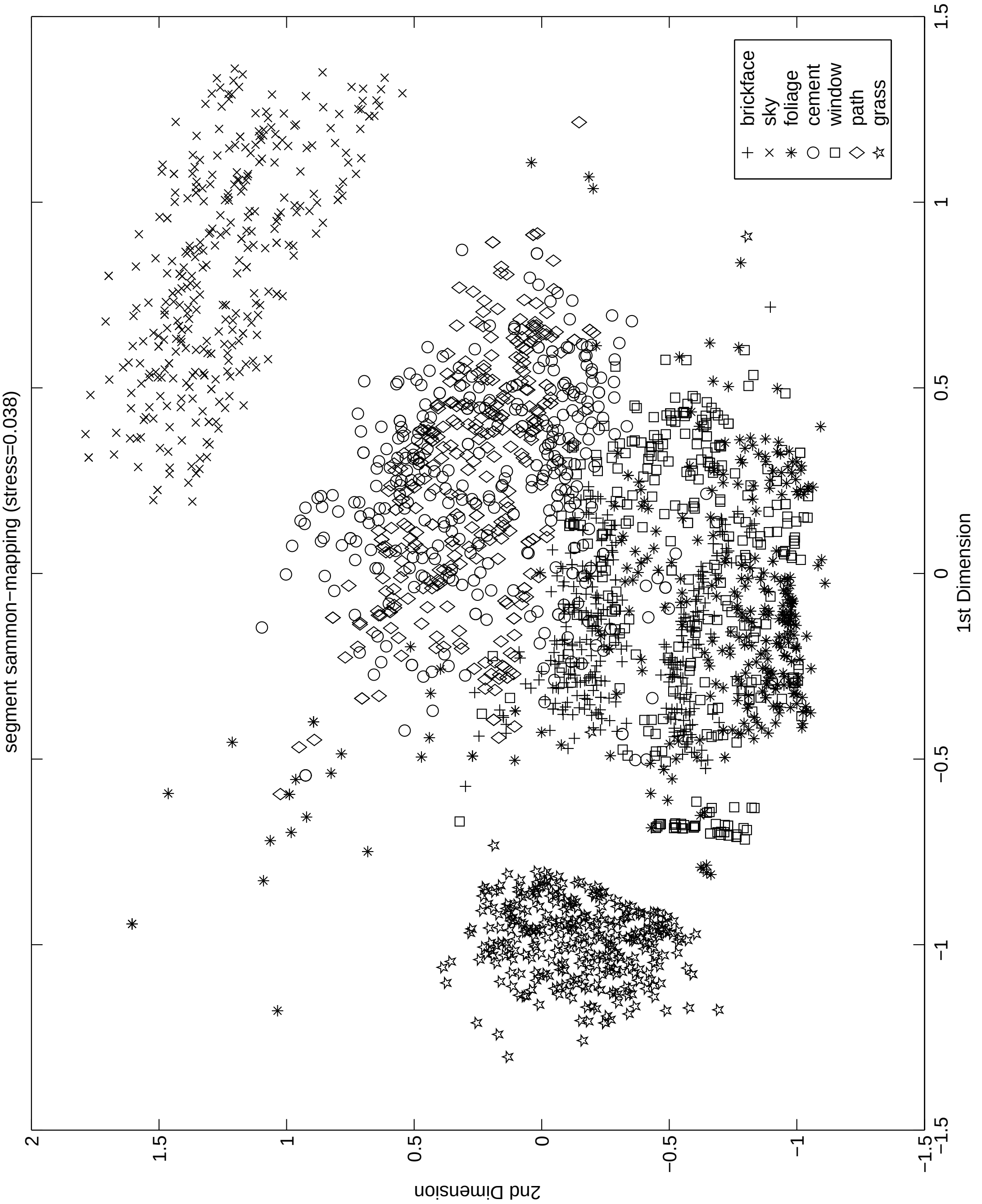


Figure A.39: Instance space visualization of *segment* via Sammon-Mapping. Two classes, sky and grass, are clearly visible clusters; the others are mixed together.

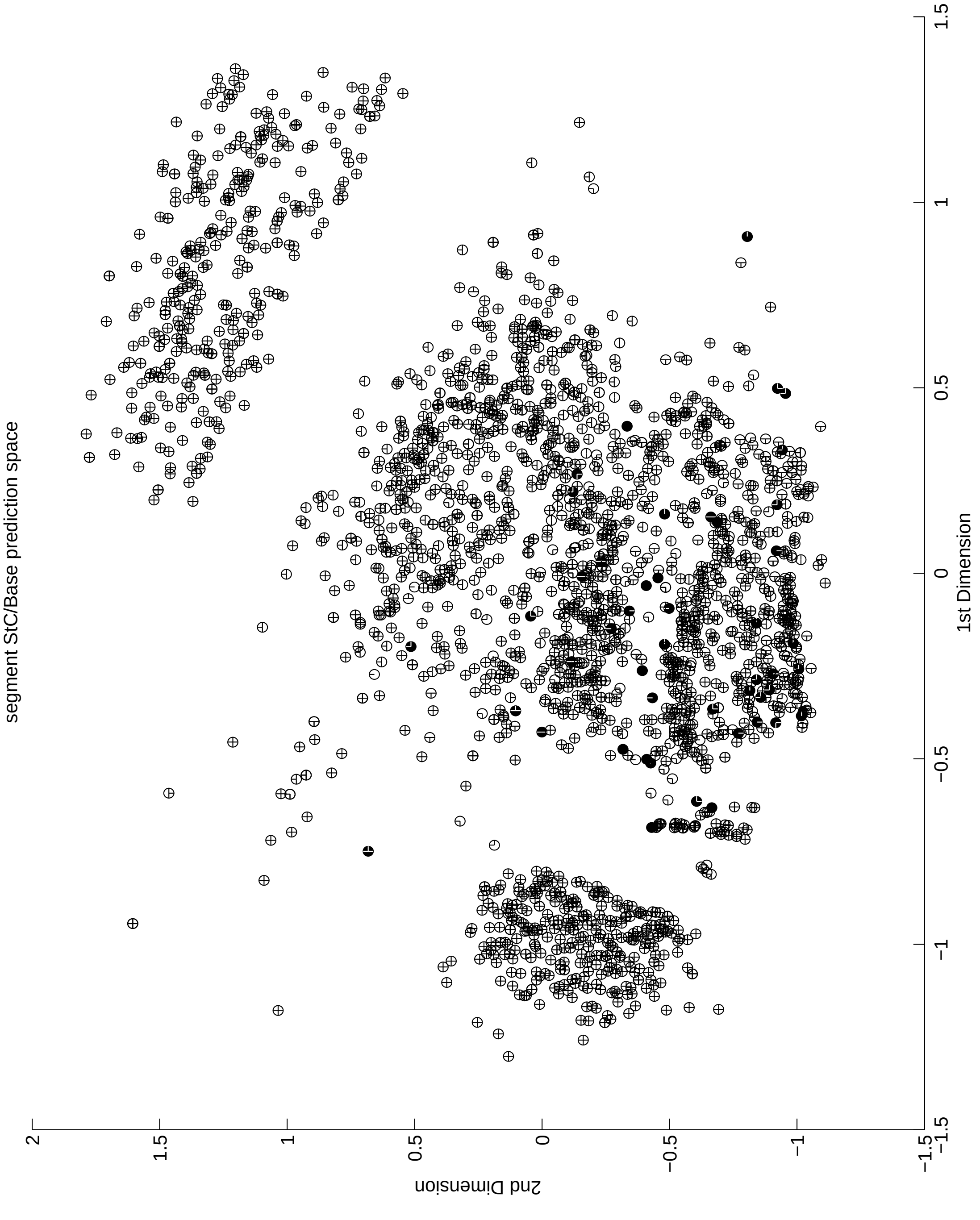


Figure A.40: Prediction space visualization of *segment* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

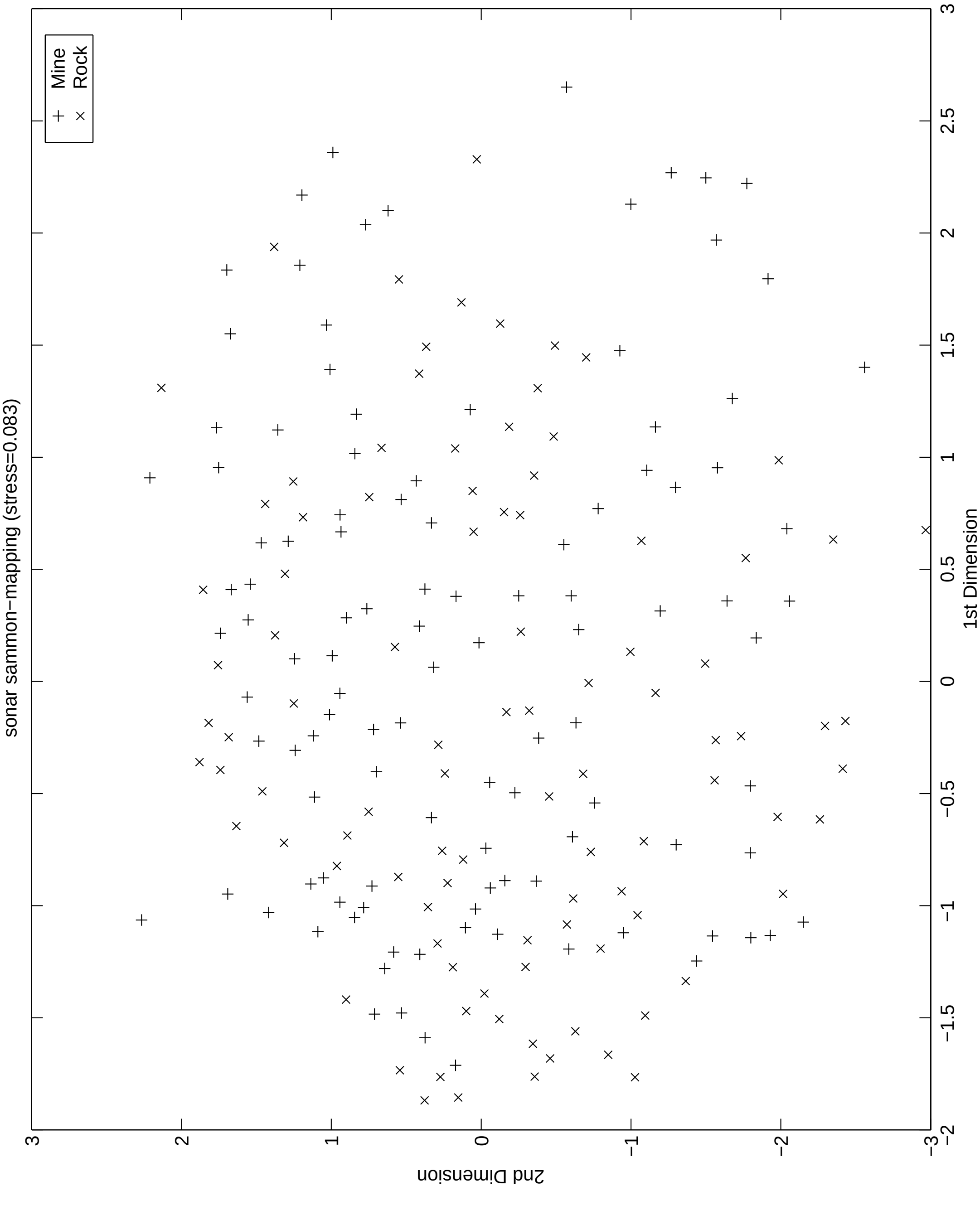


Figure A.41: Instance space visualization of *sonar* via Sammon-Mapping. The examples seem very sparse and almost randomly distributed.

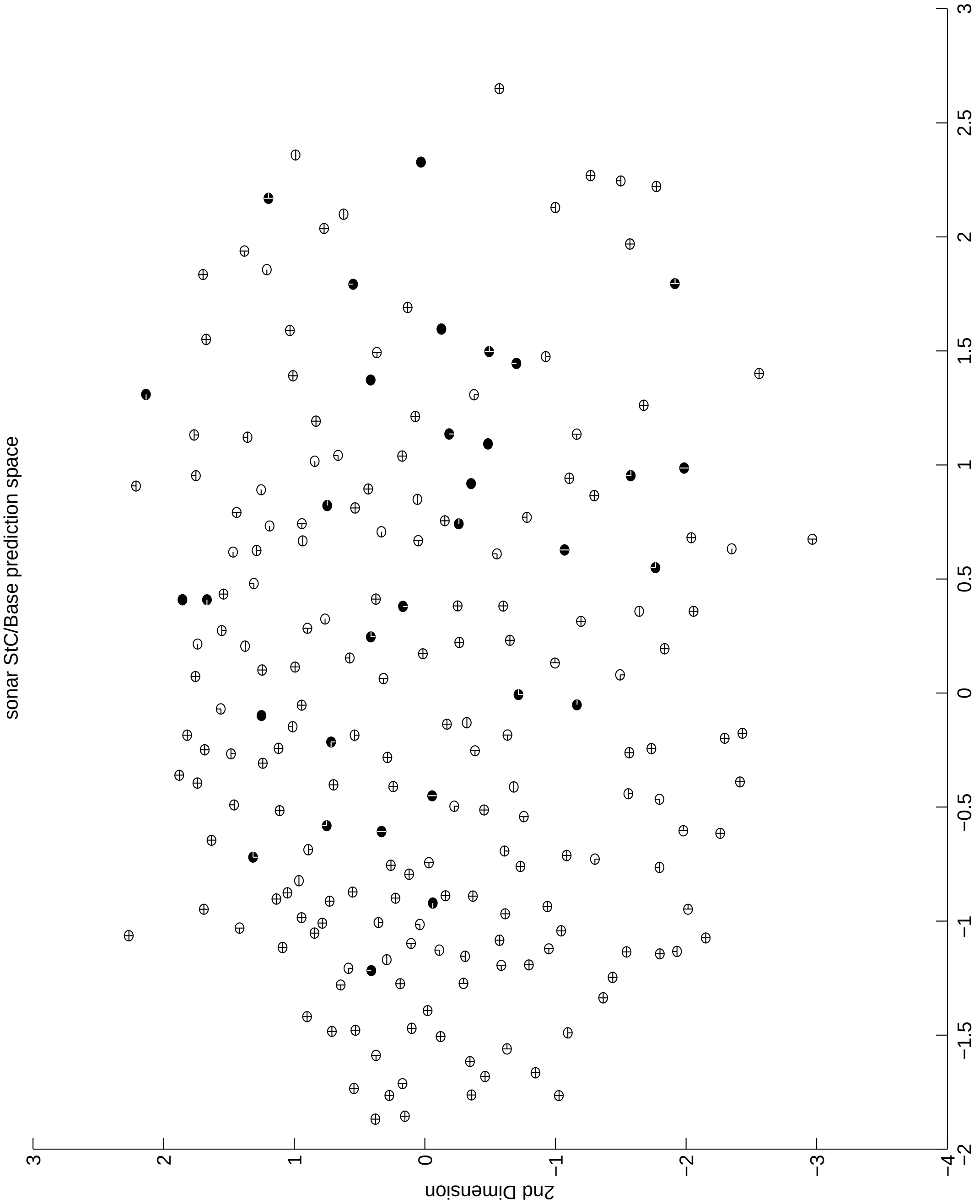


Figure A.42: Prediction space visualization of *sonar* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

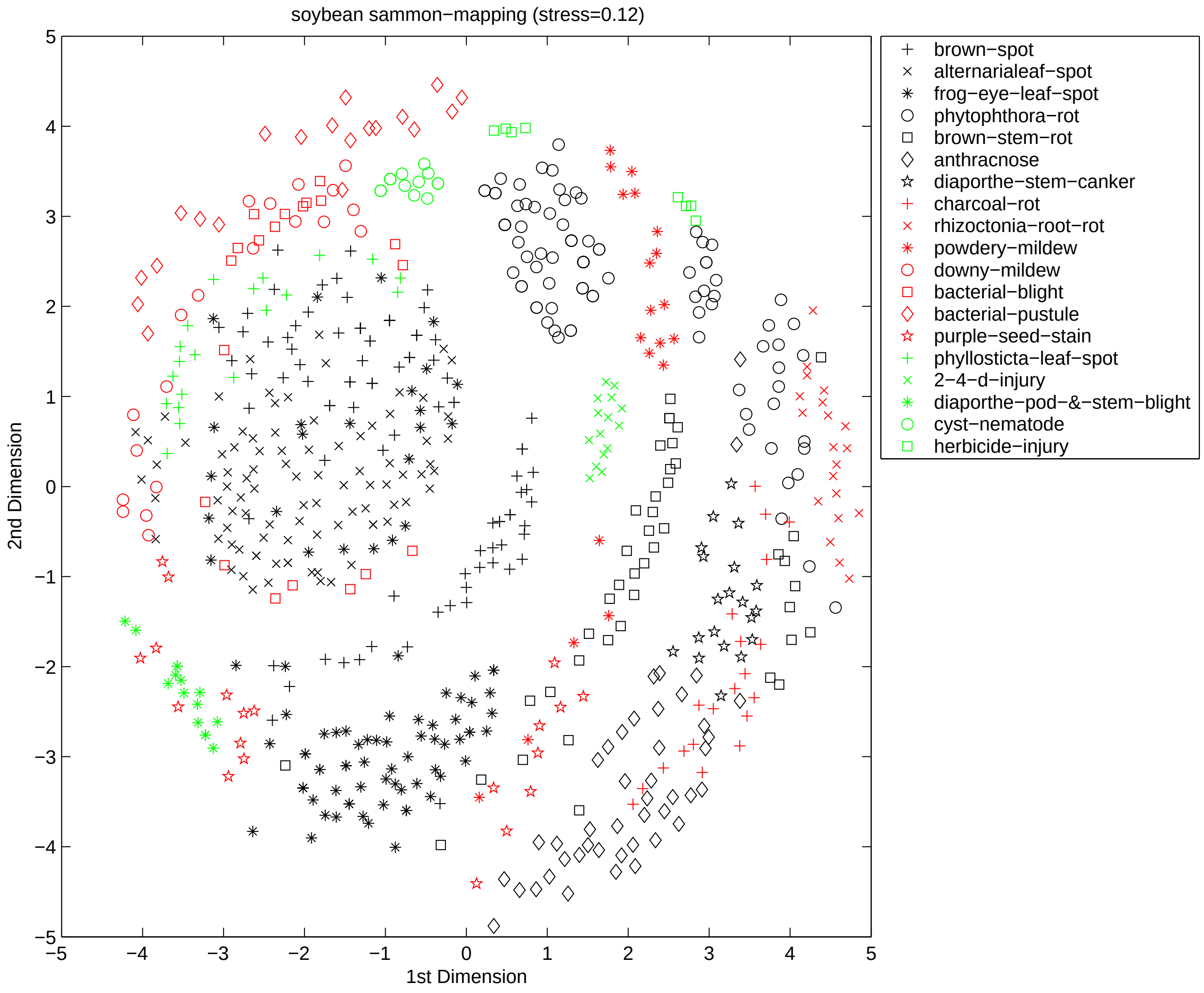


Figure A.43: Instance space visualization of *soybean* via Sammon-Mapping. For most classes, good clusters are visible. Also, the form of the clusters was very striking. The stress is still quite high, but a visualization in higher dimensions could possibly add some insight.

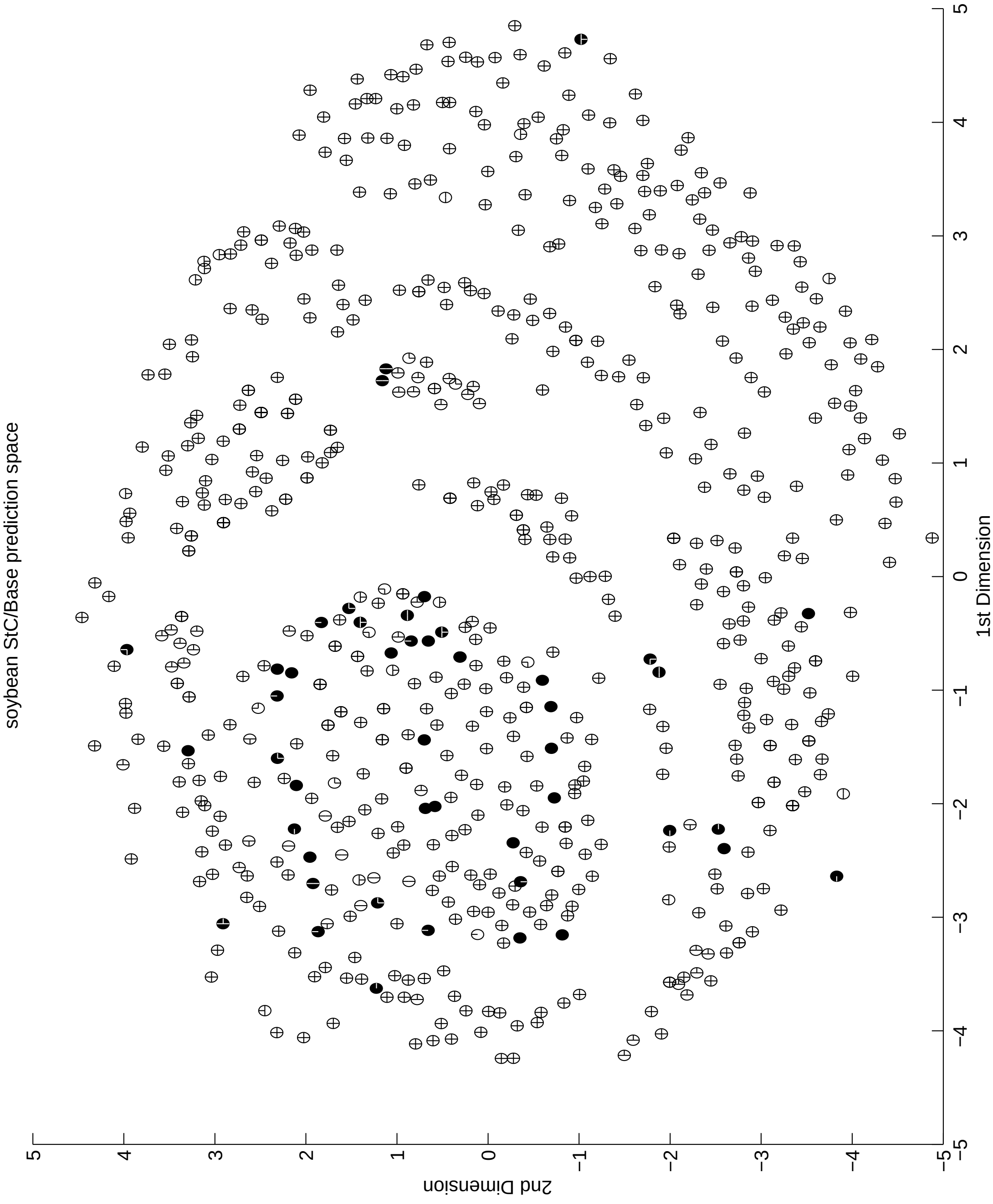


Figure A.44: Prediction space visualization of *soybean* via Sammon-Mapping and glyphs. ∘ stands for instances correctly classified by StackingC and • for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

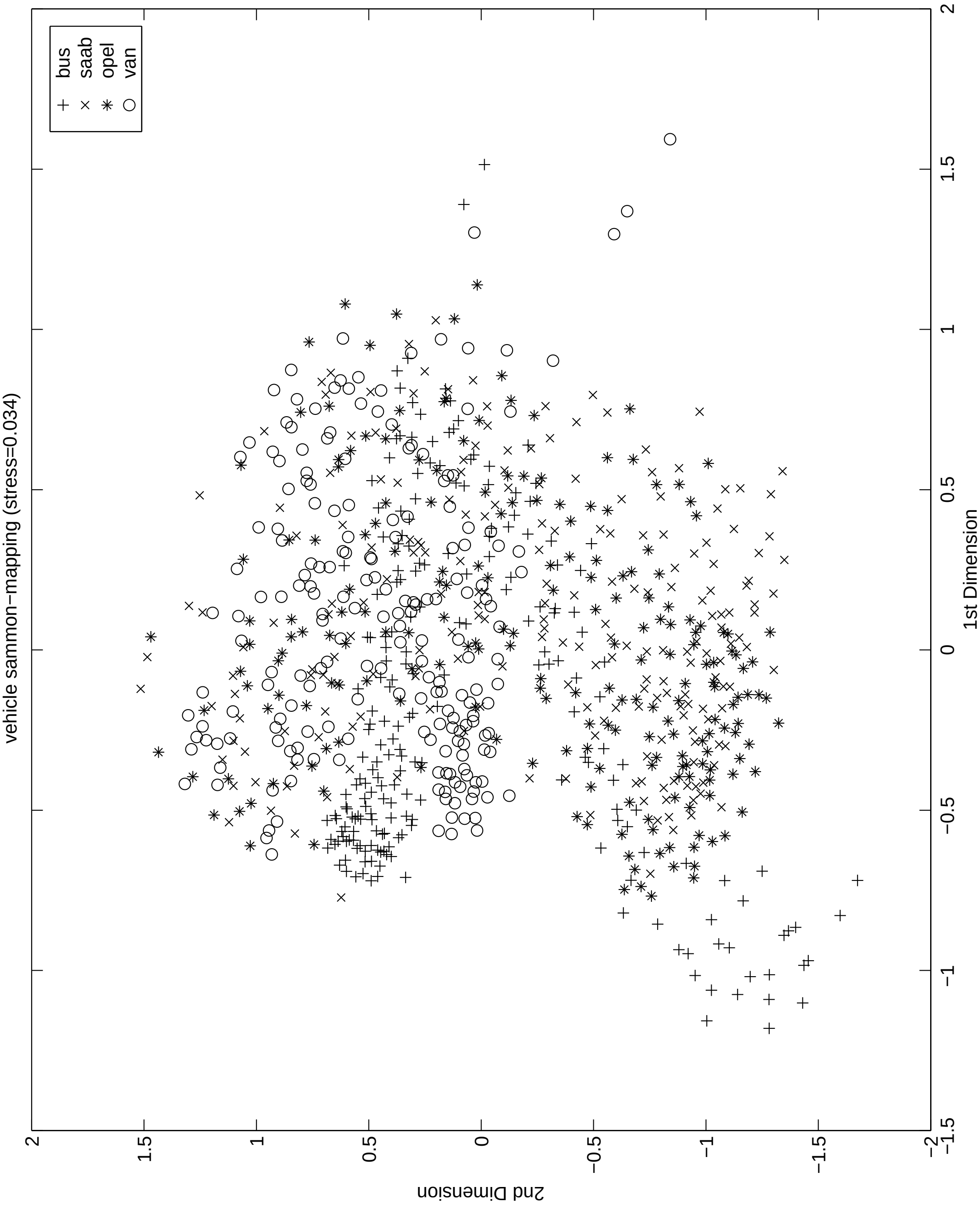


Figure A.45: Instance space visualization of *vehicle* via Sammon-Mapping. Some interesting structure is present in the distribution of examples; however, no meaningful clusters can be discerned.

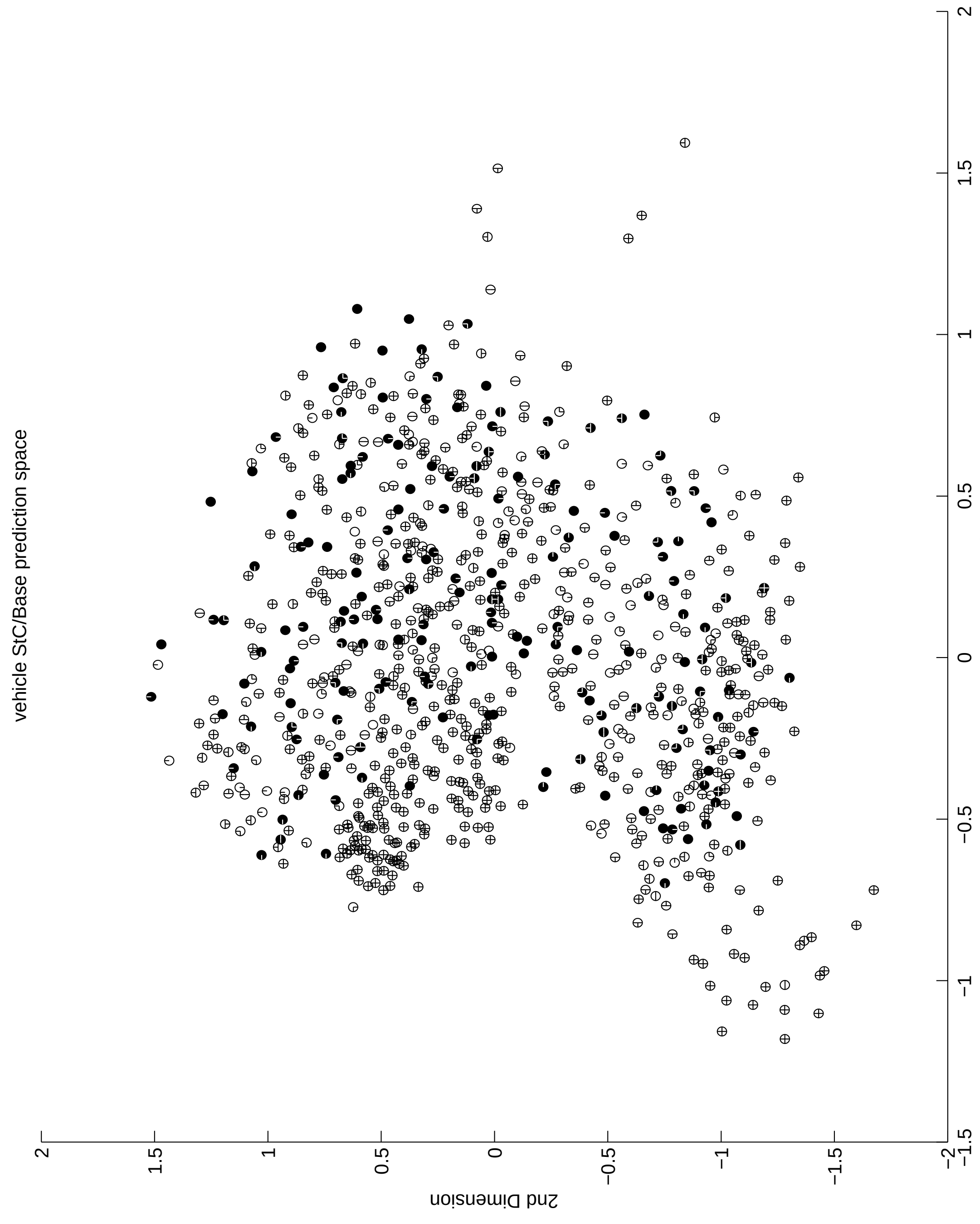


Figure A.46: Prediction space visualization of *vehicle* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

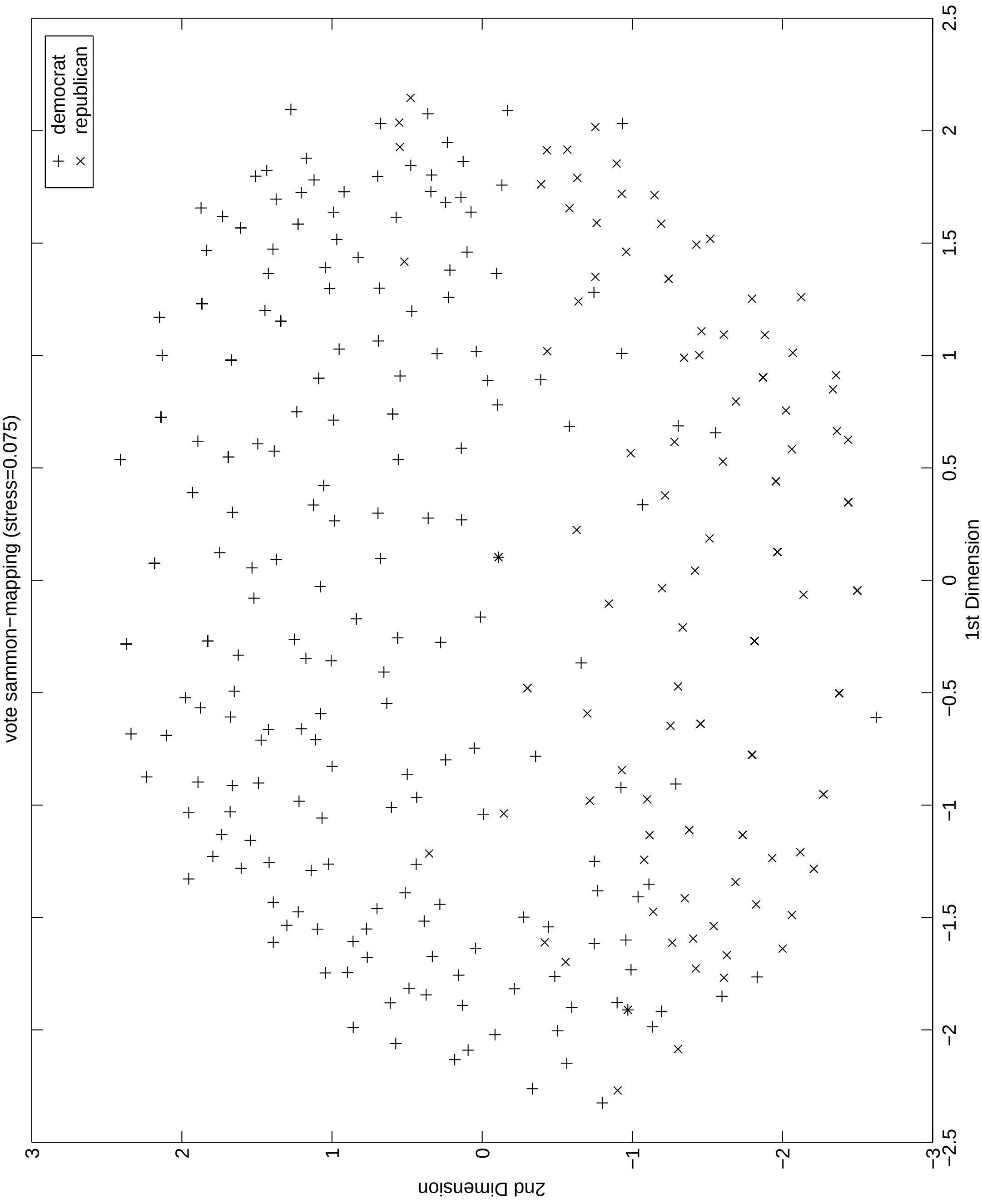


Figure A.47: Instance space visualization of *vote* via Sammon-Mapping. Towards minus the 2nd dimension, a higher proportion of republican voters can be observed. Apart from that, the distribution seems quite uniformly random in a circle around the origin.

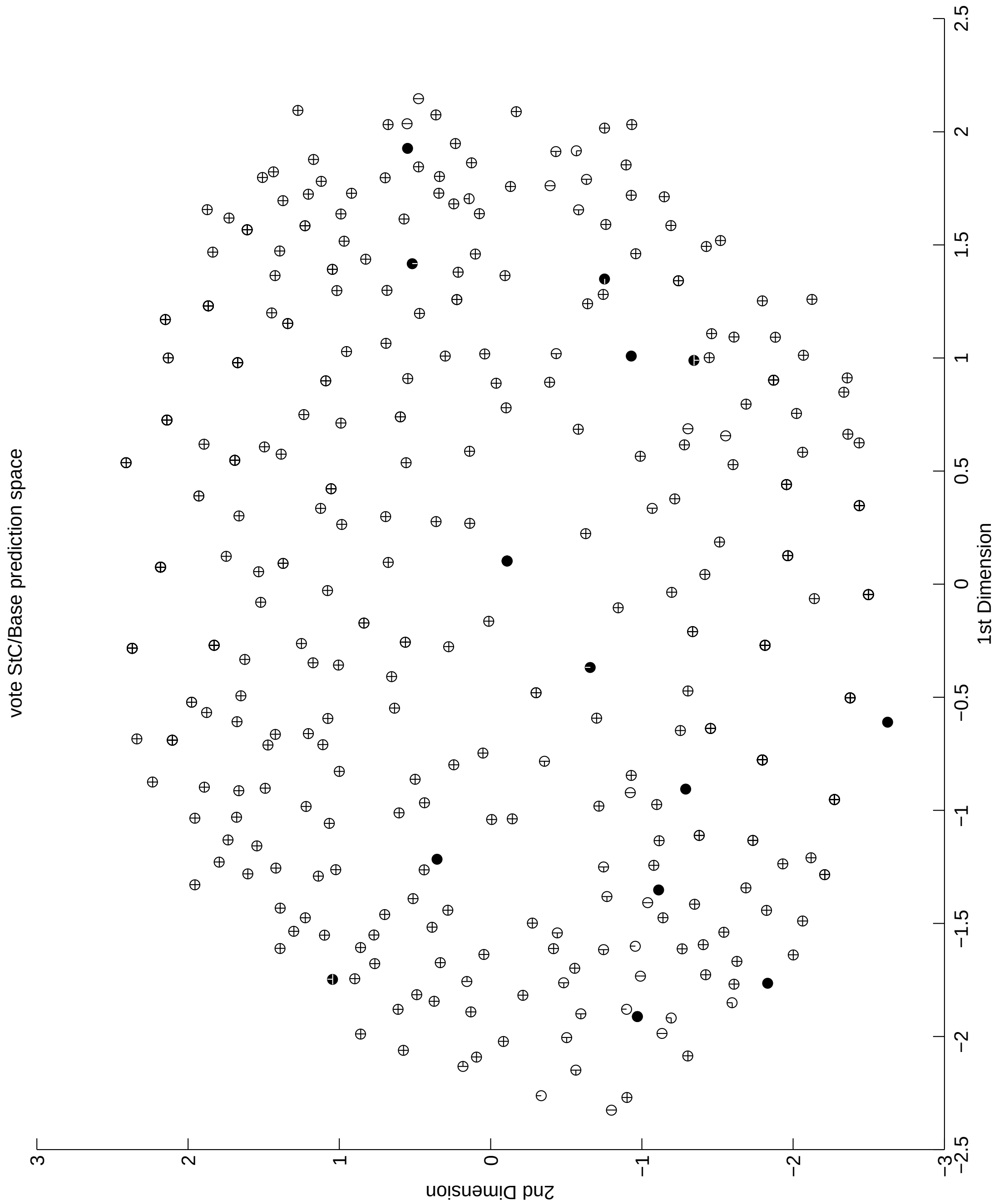


Figure A.48: Prediction space visualization of *vote* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

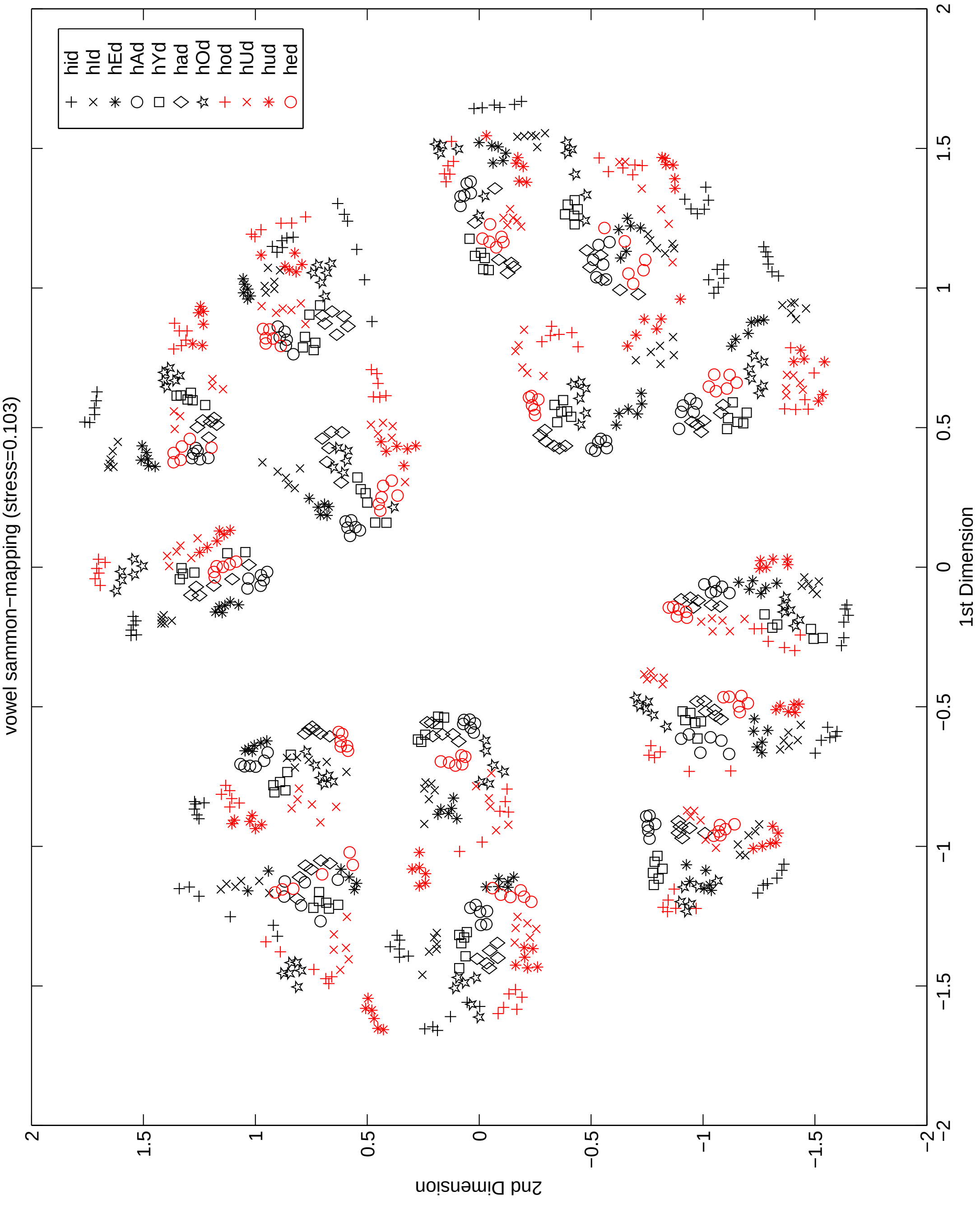


Figure A.49: Instance space visualization of *vowel* via Sammon-Mapping. The clusters are not related to the classes in this case, but correspond to the fifteen different speakers.

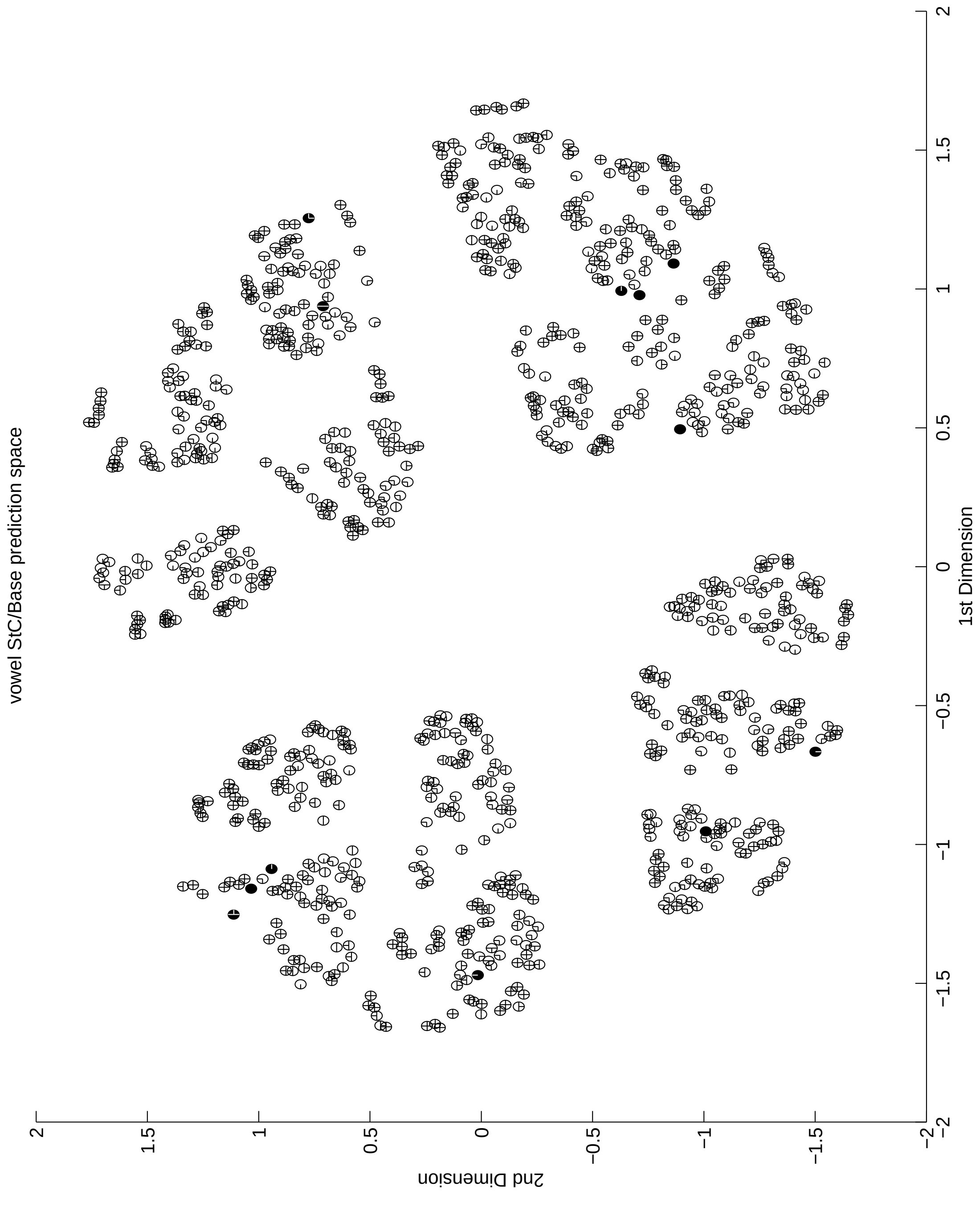


Figure A.50: Prediction space visualization of *vowel* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

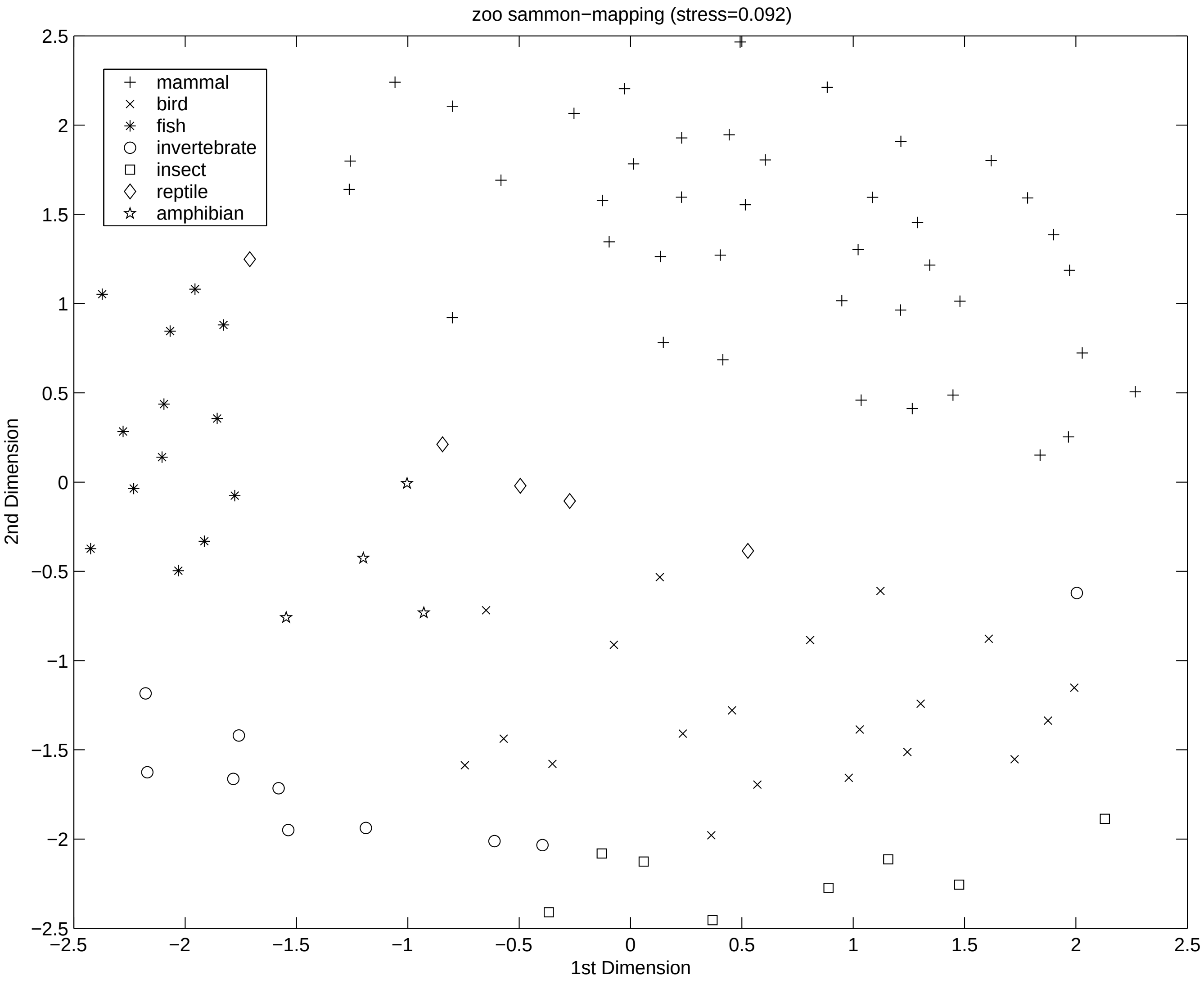


Figure A.51: Instance space visualization of *zoo* via Sammon-Mapping. This is a very sparse dataset – the second-smallest one – however the clusters are quite good.

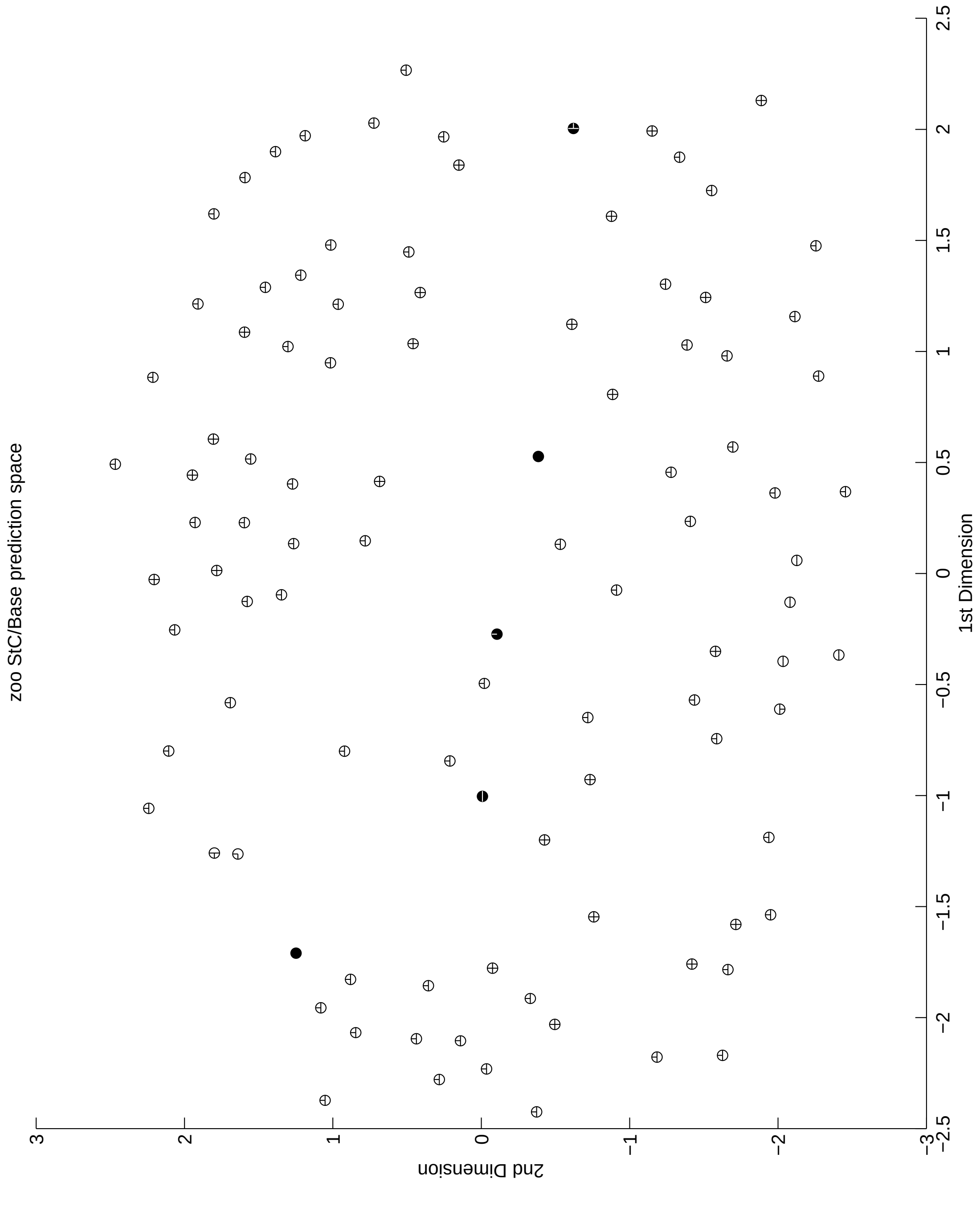


Figure A.52: Prediction space visualization of *zoo* via Sammon-Mapping and glyphs. ○ stands for instances correctly classified by StackingC and ● for incorrectly classified instances. Enclosed in each circle is a glyph which encodes the complete base classifier performance – for a more detailed explanation refer to the text.

Appendix B

Curriculum Vitae

This curriculum vitae is a selection of relevant entries. A more up-to-date list can be found at `http://www.oefai.at/~alexsee/cv.html` or on my private homepage, `http://alex.seewald.at`

B.1 Biographical data

- Full Name: Alexander K. Seewald.
- Born 8th December of 1975 in Vienna, Austria.
- Languages: proficient in German and English, basic knowledge of French.

B.2 Professional Career

- 1995-1996: Occupation as computer technician, troubleshooter and system administrator (SINIX 4.51) at Heinze
- Oct 1996-Jan 1997: Occupation as tutor at the Institute for Computer languages (basic programming course on Prolog)
- Jul 1997-Sep 1997: Summer job as computer technician at Teuber & Co
- Oct 1998-Sep 1999: Zivildienst (substitute for obligatory civil service) at the Österreichische Jungarbeiterbewegung, working as systems and applications administrator for NT/Win98-Network, implementing internal intranet-applications using MS Internet Information Server
- 2002: Data conversion from legacy systems for financial integration into a n AS/400 system using DKS, following the merger of Tarbuk with BLM (customer: Tarbuk)

B.3 Education & Scientific Career

- 1982-1986: Primary school (Volksschule) in Vienna
- 1986-1994: Secondary school (Gymnasium) at BRG Schmelz, Vienna

- June 1994: Graduation (Matura) at BRG Schmelz with highest distinction in mathematics, german, english and computer science (Fachbereichsarbeit on comparison of operating systems)
- Oct 1994: Enrollment in Computer Science (Informatik) at the Vienna University of Technology (TU Wien)
- Oct 1996-Jan 1997: Occupation as tutor at the Institute for Computer languages (Prolog LU)
- Mar 1997: 1. Diplomprüfung (B.Sc.) in Computer Science
- Oct 1999: Graduation as Diplomingenieur (M.Sc.) in Computer Science with the thesis *A mobile robot toy cat controlled by Vision and Motivation*
- Feb 2000: Employed as researcher at the Machine Learning Group at OeFAI for a two-year research project in Data Mining (KDD), *A New Modular Architecture for Data Mining*, funded by the FWF (Austrian research fund for basic research)
- Jan-Aug 2002: Employed by OeFAI as principal researcher for the EU IST project *3DSearch* (applied research with the Austrian company uma.information technology)
- Jan 2003-Dec 2004: Employed by OeFAI as secondary researcher for the three year EU project BioMinT (applied research on text mining for large proteomics databases)

B.4 Publications

See Bibliography.

B.5 Awards and stipends

- May 1991: EDV-Jugendpreis der Stadt Wien (award for innovative computer programs written by students in secondary school, granted by the city of Vienna)
- May 1992: Participation in the 23. Österreichische Mathematik-Olympiade, qualification for Gebietswettbewerb für Fortgeschrittene (competition focussing on creative solutions to mathematical problems)
- 1994: White rose from the Bundesrealgymnasium Schmelz for excellent performance during secondary school.
- 1996, 1997, 1998, 1999: Leistungsstipendium der Technisch-Naturwissenschaftlichen Fakultät (yearly stipend for excellent performance as student granted by TNF of Vienna University of Technology)
- 2002: $1000 Student travel stipend for visiting the ICML 2002 conference, presenting (Seewald, 2002a)