Defining Relative Strengths of Classifiers at Randomly Generated Datasets

Harri M.T. Saarikoski

Department of Translation Studies
Helsinki University, Finland
Harri.Saarikoski@helsinki.fi

Abstract. Classification is notoriously computing-intensive, especially with large datasets. The common (mis)understanding is that cross-validation (CV) tests on subsets of the training data are the only recourse for the selection of best classifier for given dataset. We have shown in [9] that best classifier can be selected on basis of a few prediction factors (related to training volume, number of features and feature value distribution) calculated from the dataset. In this paper, we report promising results of a series of runs with a number of strong classifiers (Support Vector Machine kernels, Decision Tree variants, Naive Bayes and also K Nearest Neighbor Classifiers and Neural Networks) with randomly generated datasets (gensets). We outline the relative strengths of these classifiers in terms of four prediction factors and generalize the findings to different types of datasets (word sense disambiguation, gensets and a protein recognition dataset).

1 Introduction

Classification is the process of automatically resolving the conflicts that occur when two or more different things have the same name and are thereby ambiguous to machines (e.g. the English noun bank can refer among other things to a financial institution or a river embankment). It is a required procedure for machine-learning application fields as varied as data mining (e.g. recognition of the protein type), text mining and natural language processing (NLP). For instance, word sense disambiguation (WSD) is defined as the resolution of ambiguous words into meanings or senses). To resolve these 'classification tasks' automatically, training instances containing independent observations (instances) of ambiguous classes are gathered (or extracted from text in case of WSD). Then from this training data, a training model is learned by classifiers, and the model is tested or evaluated against unseen (i.e. test) instances. Classification accuracy (number of correctly classified test instances / total number of test instances) is the qualitative measure used in this paper for assessing the performance of classifiers.

In the next sections, we describe the investigated datasets, prediction factors and classifier differences. Then we present the summary of results from those classifiers at those datasets and explain the systematic findings in terms of those prediction factors and finally conclude on the importance of the findings.

Advances in Natural Language Processing and Applications
Research in Computing Science 33, 2008, pp. 107-120

Received 28/10/07
Accepted 07/12/07
Final Version 18/01/08

2 Dataset Profiles

Dataset elements are target classes (e.g. senses of word), training and test instances, features and feature values. In classification tasks such as weather prediction (e.g. sunny/rain, hail/snow), target class is predicted using either binary (0/1), numerical (0..1) or set-based (0,1,n) features as probabilistic pointers to those target classes.

Table 1. Dataset matrix (fictional values) for 2-grain (2 target classes) task.

Dataset	Instance	Feature 1 (binary)	Feature 2 (nominal)	Feature 3 (numerical)
Class 1	Instance 1	0	A	0.1
	Instance 2	0	A	0.2
	Instance 3	1	A	0.1
Class 2	Instance 4	1	B	0.9
	Instance 5	1	B	0.8
	Instance 6	1	C	0.7
Feature bias 1		0.890	-0.212	0.506

Datasets from different fields have been used to define the strengths of classifiers:

1. Word sense disambiguation (WSD) datasets. Word sense disambiguation is typically based on a combination of contextual-statistical methods (e.g. n-grams or parts of speech from around target word). 1-grams are single words around target word from the whole target word instance, 2-grams are consecutive words in the whole target word instance, parts of speech are POS tags from the local context around target word (see Listing 1).

Listing 1. Parts of speech feature set (pos3) (= parts of speech with one-word window around and including target word) for *authority.n* word in Senseval-4 in ARFF format from Weka [11].

```
@attribute 'R1-WDT' {0,1}
@attribute 'R1-WP' {0,1}
@attribute 'senseclass' {1, 3, 4, 5}
{3 1, 29 1, 43 1, 54 3} % authority.n 18:0@30@wsj02
{9 1, 22 1, 49 1, 54 1} % authority.n 32:0@0@wsj02
```

First two lines show two features (R2-WDT and R2-WP: words with those pos tags one word right (R) from target word). Third line lists the target classes ('senseclass'). Last lines show two instances where feature values are represented in sparse format (i.e.

[1]Positive/negative feature biases mean the feature points to class 1 / class 2 respectively as estimated by classifiers.

only showing 1's meaning the part of speech tag is present in current instance).

2. Gensets. To compare performance of classifiers at different datasets, we generated random datasets (with both binary features 0/1 or binfeats, and numerical features or numfeats). We varied the following relevant factors:

- number of training instances (train): 40, 60, 80, 100, 150, 200
- number of target classes (grain): 2, 5, 8, 10, 15, 20
- number of input features (nrfeat):5, 10, 25, 50, 100, 200

This constituted a batch of 216 datasets of both numfeat and binfeat type (totalling at 432 datasets). Example of such dataset follows in Listing 2.

Listing 2. Genset-binfeat with five features and two target classes (excerpt). 2

```
@attribute a8 {false,true}
@attribute a9 {false,true}
@attribute class {c0,c1}}
true, false, false, false, true, c0
true, true, false, false, false, c0
```

3. Protein Recognition. Online SCOP resource (http://scop.mrc-lmb.cam.ac.uk/scop/data/scop.b.html) is a protein categorization system with 126 numerical features and over a thousand proteins. Below is an excerpt (Listing 3).

Listing 3. Protein dataset (7 target classes as class grain).

```
@attribute comp_hyd_one real
@attribute comp_hyd_two real
@attribute class {53931, 51349, 48724, 46456, 56572, 56992,
56835}
@data 26.1745,...,5.36913,46456
```

Shown in this excerpt are two features (*comp_hyd_* prefixed), target class (protein) set in terms of protein id's in the SCOP system and one instance belonging to protein class 46456 (most of the 126 feature values largely omitted for reasons of space).

3 Prediction Factors

In this section, we define the case factors that predict the best system for word and define the properties of some classifiers that help define their respective strengths. Prediction factors are sums or averages or distributions of individual instances and features in the training set. In a previous article [9], we have already presented two of them: P (average number of positive training examples per target class) and N (total number of training examples minus P, i.e. number of negative training examples per

[2]Genset-numfeat datasets are just like binfeats but with fvals 0..1 instead of 0,1.

class). Additionally, in recent tests with gensets, we have discovered the following factors relevant:

1. Number of features (nrfeat). While training volume (P+N) and nrfeat already distinguish between some classifier pairs (families even), we still could not explain the difference of some classifiers different weight functions. We found that of all individual factors investigated thus far there exists highest correlation between nfeat and best classifier (0.35 in correlation coefficient). Even P and N remained less correlated but this can be explained by their codependency and interrelation in determining best classifier.

2. Feature quality (fqual) estimation. When we investigated the difference between different decision trees, we realized that they weight low/middle/high-quality features differently (see Classifier differences). Therefore, average fqual of features in the dataset has to make a difference in selecting best classifier. We do not here measure feature quality explicitly but only note the following difference between the datasets: n-grams are typically high in number but relatively low in quality (occurring very few times) while genset features converge to equal quality (due to randomness of fval assignment).

3. Feature value (fval) distribution. This factor characterizes the distribution profile of feature values: while gensets have equal distribution of fvals from 0 to 1 (i.e. with numfeats average fval of each feature converges to 0.5 and with binfeats both 0 and 1 fvals are almost equally probable). N-grams in WSD in turn are focused around value '0' (i.e. there are many features but each of them is not present in most instances). For example, 2-grams have a very high concentration of '0' fvals (features occur rarely) and pos3 the lowest (features occur more frequently). This distribution profile can therefore be characterized as very different from gensets: 'inverse Gaussian' distribution (high-low-high) with a skew toward value '0'. As for protein dataset (SCOP), we noticed that the distribution of approximately 80% of the numerical features enforces Gaussian distribution (low-high-low), i.e. fvals are focused in the middle range.

These two last factors (fqual and fval) will be shown below to be relevant in defining the strengths of different classifiers in terms of their feature / instance (kernel) weight functions respectively. Next we define the relevant differences between the selected classifiers.

4 Classifier Differences

We make the following relevant four divisions:

Difference 1. Weighted dataset element is either instances (containing features) or features (in isolation). We differentiate between feature and instance weighing classifiers:

(a) Feature-based classifiers (Decision Trees or Dtrees as a group name for Dstump, J48, REPTree, CART and PART classifiers from [8] and the well-known NB or Naive Bayes classifier [11]). These classifiers operate by estimating feature quality (or feature-class weights), e.g. probabilities that value TRUE in a binfeat dataset predicts class A vs class B. Formula for calculating feature quality however differs from

classifier to classifier. For instance, decision trees usually have two feature weight functions (linear and logarithmic) and corresponding NB formula is also presented 3:

- Information Gain Ratio used in J48, REPTree and PART [8]: `P(feature|class) = P(class) * log2 (P (class|feature))`
- Gini Index used in CART decision tree [3]]): `P(feature|class) = P(class) * P(class|feature)`
- Naïve Bayes formula used in NB [11]: `P(feature|class) = P(class) * P(class|feature)`

As we can see, the main difference between CART and Information Gain Ratio based classifiers, for example, is the feature weight function (linear vs logarithmic respectively). Note also that NB is comparable in terms of weight function (linear) to CART classifier (both have essentially the same formula), only differing in the number of features they include in the training model. When we look at the weight function curves (linear and logarithmic), we can categorize their adaptability with features of different fqual profiles as follows: linear weight function weights extremely low and high quality features more highly while logarithmic weight function weights middle-quality features more highly.

(b) Instance-based classifiers (SVM or support vector machine [10], kNN or k nearest neighbor classifier [1] and FNN or feed-forward neural network [6]): Instance-weighing classifiers use feature weight estimations to calculate distances between training instances in projected Euclidean feature space (either original or augmented). The weight function therefore determines how much emphasis is put on class cores vs borders respectively (we call this 'core-border ratio'). We further investigate the most popular ('k' or instance similarity) functions for SVM. The function `K(X,X')` optimizing the class borders differentiate kernel types:

- Linear (SVMlin): `||x·x'||`
- Polynomial (SVMpoly): `(gamma*||x·x'||)^degree)`
- Radial basis function or RBF (SVMrad): `exp(gamma*||x-x'||)^2`

Kernel function notation `K(x,x')` denotes two-class borders (called hyperplanes) X and X' and the optimizing function K itself that seeks to maximize the distance between X and X'. Inside the kernel function, ||x·x'|| denotes the dot product of any two instances projected into Euclidian space [10].

The fundamental difference between the kernel functions can be described as follows: polynomial function (x^2) upweighs the distance between high-difference instances (which by definition are concentrated in the class borders) and downweighs distances of instances with smallest differences (typically those located at class cores). In linear kernel (x), weights increase linearly with distance. When comparing these two kernels, polynomial kernel tends to equalize small differences (which are typical of core instances). Radial kernel does the opposite to polynomial kernel, upweighing small fval differences between instances (class core instances). Notice however that unlike linear and polynomial kernels, RBF does not transform features into nonlinear

---

[3]I.e. probability that a feature points to a class is probability of class in training data multiplied by logarithm / linear probability that a class points to a feature.

space: instead distances between instances are rather calculated from the sum of raw differences in fvals (x-x')[4].

Difference 2. Original vs augmented feature spaces. We further distinguish classifiers based on whether original input features are used (kNN, NB, Dtree, FNN) or whether feature space is transformed or augmented (SVM). SVM attempts to discover instances on the class border and reweighing the rest of the instances in relation (distance) to them, resulting in augmented feature space where classes can then be linearly separated. FNN in turn seeks to (gradually) place instances that are most related next to each other, attempting to classify instances in original feature space (as opposed to SVM). kNN also operates in original feature space, calculating pure Euclidean distances between instances. In other words, kNN does not transform feature weights like SVM does nor gradually iterate on instance similarity metric like FNN does.

Difference 3. Training model size. can also distinguish between classifiers that include all features in their training model (kNN, NB, FNN) as opposed to those who reduce features in various ways (Dtree, SVM, Dstump). [12] calls these generative (all features) vs discriminative (some features) classifiers respectively. Dstump only picks a single (best) feature and Dtree picks a majority of features (depending on the set amount of pruning). SVM in a way also reduces dimensions since its training model is constructed on the 'support vectors' or the class-border instances in transformed feature space (and hence features occurring in other instances than these are significantly downweighed). NB and Dtree in turn both estimate feature quality using entropy of `P(class|feature)` utilize class priors `P(class)`. There are therefore only two main differences between NB and Dtree: (a) Dtree creates a tree of conditional dependency (AND) rules consisting of several features pointing together to a class while NB uses feature weights in isolation (not expecting dependency between features, hence the name 'naive'). (b) Dtree discards features of poorest quality (under a configurable threshold) while NB uses all features.

Difference 4. Method of model optimization. are two types of model optimizers (or 'tree pruners' when dealing with Dtrees): (a) those that alter feature-class weights based on previous run misclassifications (error-based) and (b) those that generate multiple models and vote between them (vote-based). Additionally there are those that perform no pruning (let us call them 'single run based'). Within the decision tree family under scrutiny here, the corresponding classifiers are REPTree (error-based), CART (vote-based) and J48/PART (single run based). We further present and test two successful meta-learning schemes that implement these types of pruning (a) and (b) (these operate on the output of a selected base classifier):

- **Boosting** [5] is an error-based iteration method. It produces a series of models (set by number of boosting rounds, here 19) trained on different bootstrapped portions of the training data and performs weighted voting between then. More essentially, Boosting upweighs the features in misclassified instances of

[4]The gamma parameter can in this respect be considered a fine-tuning parameter, altering the angle of the function curve only slightly.

each boosting round. This is essentially the same procedure as with REPTree (reduced-error pruning).

- **Bagging** ('Bootstrap Aggregating') [2]. Bagging can be considered as a vote-based method like CART. In training phase, training data is first split into training and test subsets (set by size of split or bag, here 80%). Then, a chosen base classifier is trained on one bag and tested on the remainder. All these submodels then vote on their combined class decision.

5 Test Results [5]

To produce results that are also robust to any test instance, we ran 10-fold cross-validation 10 times on each of the above classifiers (including Boosting and Bagging), resulting in 100 separate test results for each classifier and each dataset. Cross-validation evaluation in Weka [11] also keeps the same target class distribution for the folds as in the whole dataset (= stratified cross-validation) to ensure that the folds are maximally alike and do not thus influence the classification accuracy (nor classifiers' different capability to deal robustly with new test data) dramatically. This is incidentally why we were motivated to omit training-test set divergence as prediction factor (although it was shown in [12] to differentiate classifiers), since 100 tests on all classifiers serves to downplay the role of divergence (as opposed to the 5-fold single-iteration tests done by [12]).

We here report results from tests of selected classifiers on gensets and proteins datasets. We also report a summary of OE tests, i.e. from utilizing prediction factors for the prediction of best classifier. Corresponding OE tests with WSD systems (both custom and Senseval [6]) have already been reported in [9].
(We discuss the significance of these results in next chapter). We can mention here further that OE gross and net gains with gensets were within range of those found for WSD datasets in [9], with net gain between 2-4%. For instance, accuracy for predicting between Dstump and NB was 0.89, which means that out of the potential (gross gain) of +4.1, a net gain of 0.89 * +4.1 = +3.2 was obtained. Prediction accuracies were again well above most frequent class baseline as they were in [9]. Best OE net gains with genset batches were in fact obtained from using at least one decision tree (CART).
A few notes of these results right away: While NB wins SVM at 5-grain, SVM gains over NB both with lower and higher grains. Polynomial kernel of SVM begins to gain steadily over linear kernel after 10-grain (ends up at stable +7% at higher grains). Linear decision tree (CART) using cross-validation based pruning performs better than logarithmic unpruning J48 at any grain as it did with gensets. Some other grains showed similar trends and were therefore omitted.

---

[5] All classifier runs were made with Weka implementations of these algorithms [11] (Weka is downloadable from http://www.cs.waikato.ac.nz/ml/weka/). Default configurations were used: complexity parameter for SVM was 1.0, confidence parameter for Dtrees was 0.25 (medium amount of tree pruning). Weka's AdaBoost.M1 was used for boosting with 10 iteration rounds and bag size 80% for Weka's Bagging algorithm.

[6] Evaluation framework for WSD systems (see www.senseval.org).

Table 2. Average accuracies of classifiers at two different types of gensets.

gensets-binfeat		gensets-numfeat	
Bagged CART	60	Bagged CART	80
Boosted CART	58	Boosted CART	78
CART	58	CART	77
SVMpoly [7]	56	REPTree	76
SVMlin	56	PART	75
NB	55	J48	75
REPTree	54	NB	69
J48	54	Dstump	66
PART	53	SVMpoly	65
Dstump	50	SVMlin	65
SVMrad	49	SVMrad	56

Table 3. Classifier results on different class grains of protein dataset.

grain	NB	CART	J48	SVMlin	SVMpoly
2	73	61	55	75	75
5	72	62	57	72	66
10	64	58	55	68	49
50	32	42	36	43	49

6 Classifier Strengths

In this section, we look at classifier pairs to see where their strengths lie with regard to three prediction factors [8].

Finding 1. SVM (linear kernel) vs NB form a hemispherical section in PN space. In all datasets involving linear SVM and NB so far investigated (see e.g. [9] for WSD systems, [4] for text classification systems), SVM tends to take C2-C1 corners (low-P) section of PN space over Dtrees and NB, leaving NB (and Dtree) C3-C4. At gensets we

[7] Using degree (exponent) of 2, i.e. Kernel function is thus ||x·x'||^2.

[8] Due to space constraints, prohibiting the inclusion of dozens of images generated using RapidMiner [7], formations (20) will be posted to the following site (www.mysenseval.com) until they are published in my dissertation later this year.

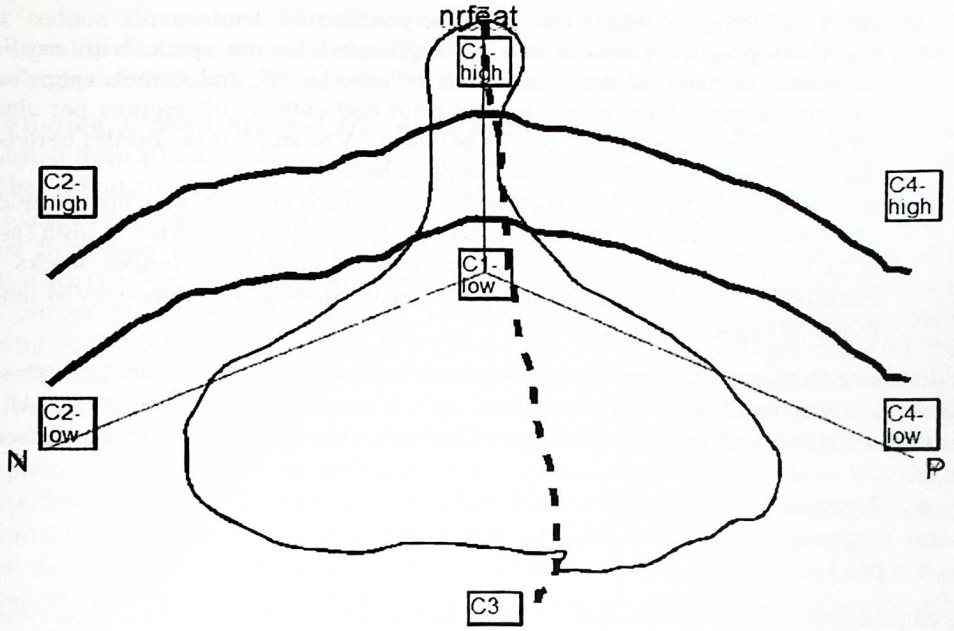

Figure 1. Three-dimensional space for representing classifier formations (= relative strengths) with P, N and nrfeat as dimensions. C stands for 'corner' (e.g. C4 is high-P but low-N, C2 is high-N but low-P) and the region remaining between corners we call 'midfield'. Shown are three types of formations: (a) single pear-shaped formation is called 'yin-yang formation' with one classifier taking the midfield (and the whole of C1 regardless of nrfeat), the other the P and N facets, (b) dotted line in the middle is called 'hemispherical formation' with one classifier taking P facet (C4) and the other N facet (C2), (c) two horizontal lines denote 'layered formation' (or layers), with one classifier taking both low and high nrfeats (regardless of P and N) and the other middle nrfeat range. (Cf. Discussion and [11] for another account on such 1-2-1 phenomena). We refer to these types of formations below when describing classifier performance differences.[9]

found SVM to restrict itself more to C1 (shifting from C2). In [9] with WSD datasets, we showed the border of SVM and NB to be almost linear in PN space.

This hemispherical formation was not as clear with gensets (probably due to the relative weakness of both SVM and NB at gensets), SVM tends to lose at C2 cases more, but it can be seen. There are many separate explanations to this:

- (a) SVM optimizes class-borders while NB makes no explicit difference between cores and borders. The higher the grain, the more borders there are, and so the better this expertise works. SVM is simply better equipped to classify cases where definition of borders are more crucial (i.e. high-grain

[9] We should note that while these are approximations of the actual formations, they are nonetheless clear-cut enough to fall into one of these three main formation types. Also unless specifically mentioned, the formations were similar for both binfeats and numfeats.

cases including C1 where training is very scarce).

- (b) SVM projects features into an augmented feature space with explicit definition of class borders, while NB retains the original feature space and weights features linearly using class-prior and entropy of features per class. Hence, we can consider the border between SVM and NB the border between linear and nonlinear feature space approaches.
- (c) NB operates largely on its class-prior, SVM has no such prior but considers all classes as equally important to classify correctly. When grain is high (as it is at C2), this upweighing of more frequent classes no longer works as effectively while SVM gains in strength with added grain. Hence SVM tends to win at higher grained tasks.

Furthermore, in recent WSD tests (with Senseval-4 English lexical sample dataset), we found the strengths of SVM (linear kernel, c=1.0) and NB different with regard to feature sets (number of word wins are quoted, note that if sum is not 100 rest have been tied):

- 2-grams: SVM 68, NB 23
- 1-grams: SVM 38, NB 4 (lot of ties)
- pos3: SVM 22, NB 35

We can see that SVM performs best when features are numerous but of low individual quality (2-grams) and NB best when features are relatively few but occur frequently (pos3), with both excelling equally at 1-grams (which in terms of both nrfeat and average number of occurrences of features is between 2-grams and pos3). This is an indication that fval distribution does define strength of at least these classifiers: SVM seems to able to perform better when features are very numerous (> 200) but of low occurrence, NB vice versa. This is the most likely reason why SVM did not perform well with gensets (see Table 2) but did perform well with WSD [9].

Finding 2. Dtrees, NB and Dstump form layers. We find clear-cut formation between these three classifiers per nrfeat factor. While Dstump is compatible with highest nrfeat, NB in the middle and Dtree (J48 in this case) takes the lowest nrfeat. We explain this as follows: Dtree operates on class-prior like NB, it additionally cuts (prunes) features of lowest quality, which would be essential in drawing class-borders. For this reason, Dtree cannot build a full-blown valid tree if nrfeat exceeds a certain limit (nrfeat=50) (we can call this 'tree building feasibility requirement') since at high-nrfeat, it becomes increasingly more difficult to find valid feature-class pairings (not to mention codependencies between them) (e.g. "IF rule 1 AND rule 2, THEN select class A"). With lower nrfeats, finding such interrelations between features is much easier (and the classification task itself is much easier), since then on average each feature is defined (validated) by many more training instances and thus the chance for an erroneous tree leaf propagating throughout the tree branch is significantly lower. Dstump in contrast is free of that tree-building feasibility requirement since it builds only a tree stump. Hence it works the better the more nrfeat is provided. As for why NB takes middle nrfeat layer between Dtree and Dstump, we found that this is the nrfeat range where both Dtree's and Dstump's strength degrades, i.e. is not particularly strong due to the abovementioned tree-building requirement. We need also to consider the fact that NB keeps all the features including those with lowest feature quality or fqual (does

not reduce dimensions like Dtree and Dstump), and so we can conclude that low-quality features are in a significant role at tasks with mid-nrfeat tasks (than at low/high-nrfeat tasks). This is why NB excels at such tasks in its own merit.

Finding 3. Most decision tree pairs make yin-yang formation. This formation resulted from all four decision tree variants when paired up. As to which classifier takes higher nrfeat position in the yin-yang formation seems to depend on their ranking order of average accuracies per dataset (i.e. CART > REPTree > J48, see results in Table 2). This formation type at least is therefore determined by the relative strength of the classifier (i.e. stronger classifier takes P and N facets at high-nrfeat and weaker one the C1 corner and low-nrfeat midfield). This formation results also when ensembling a strong decision tree like CART with SVM. As for defining the strength of CART (best overall classifier at gensets), we can infer that the datasets where linear decision tree (CART) performs best must have a relative profusion of either low- and high-quality features, since those are upweighed by the linear feature weight function in CART (and respectively mid-quality features must be in slight relative majority where logarithmic classifiers J48, REPTree and PART perform best). Although we have not deployed a quantification of fqual (as mentioned in Prediction factors), we find no other reason than the profusion of mid-quality features as factor that *must* explain why CART as a classifier that upweighs mid-quality features is the superior (decision tree) classifier with gensets.

Finding 4. Boosted vs Bagged decision trees make layers. This is the most clear-cut layer formation investigated in this study: Bagging takes high-nrfeat and Boosting low-nrfeat layer. To find causes for this layeredness, we ran boosters against baggers on two different decision trees with different weight functions (CART and J48 as base classifiers). We also ran error-pruned (equivalent of Boosting) version of J48 and tested two different values of the confidence parameter in J48, responsible for the amount of pruning. Pruned (CART, REPTree) vs unpruned decision trees (J48, PART) mentioned in Finding 3 were also obviously investigated. From these tests, we can confirm that both the amount of pruning and the type of pruning can be used to select best classifier for given dataset: pruning (especially cross-validation type) definitely works better at higher nrfeats. We can motivate this as follows: at high-nrfeat (which are tougher cases to classify anyway), feature-class pairings can only become fully validated, i.e. proper class cores and borders defined using those pairing, when the decision tree is trained multiple times on different subsets. Strength of Boosting in turn can be defined as follows: The primary issue for model optimization at low-nrfeat is determined by the number of outlier (misclassified) instances. Though not reported in the results, we found that the number of misclassified instances increases logistically as nrfeat is increased (i.e. classification accuracy is lower at high nrfeats). That is, at low-nrfeat there are simply fewer outlier instances. When Boosting (features and instances containing them) then upweighs them, they do not disturb the delicate weight balance of features occurring in the already correctly classified instances. Once the case becomes tougher as defined by added nrfeat, Boosting no longer can maintain this balance but gives overweight to misclassified instances (i.e. overfits the training model).

Finding 5. Linear and polynomial SVM kernels form no uniform type of formation in terms of PN+nrfeat factors. Strength of these kernels proved hard to define. Depending on whether gensets were binfeat or numfeat, the formation of the two kernels resembles either layers or hemispheres with some non-convex classifier regions. We explain this with their different instance weight (kernel) functions (see Classifier differences). From looking at the function curves of linear vs polynomial kernel (x vs x^2 as function of instance similarity in augmented Euclidean space), it is obvious that polynomial kernel by downweighing small distances tends to expect more core instances (i.e. core is wider and border is projected farther). In contrast, linear kernel expects rather a smooth transition from one class core to mutual border and over to another class core (i.e. that class cores and borders are of equal 'size'). Since this is their *only* difference, their respective strengths can be said to depend on this expectation.

Therefore, we can conlude that the dataset's core-border ratio (defined above), i.e. how many core vs border instances there are on average per target class, determines the best kernel. With gensets, as said, core-border ratio close to 1 qualifies linear kernel (core is of equal 'size' as borders on average for all classes) and ratios above 1 qualify polynomial kernel (core is larger and needs therefore to fit more instances and the farther we go from that core the less important the instances are estimated to be). In other words, polynomial weight function can be said to perform feature-reduction by way of projecting the far-away instances from the hyperplanes (class border) separating the target classes (cf. [11]).

Core-border ratio is an elusive individual quality of the dataset which could be estimated from fqual factor: high-quality features correlate highly with class core (low-quality features with class borders). However, since fqual measure is out of focus in this paper, the only thing that can be said is the following: gensets that have no pronounced core-border division (gensets) have a core-border ratio of 1 (i.e. there are no real cores vs borders since all fvals converge to equal fval distribution resulting in almost equal fquals for all features). Real-world datasets (such as protein recognition or word sense disambiguation) in contrast tend to have a more clear-cut ('natural') sense of class cores vs borders and a definite division between high and low quality features (e.g. 1-grams *and* and *money* are low- and high-quality features pointing to one of the senses of the English noun *bank*).

One kernel, however, is separable using these three prediction factors: radial kernel (with Gaussian weight function) sets apart from all other kernel types both in terms of formation and (lower) overall accuracy. We have confirmed that it performs best only at C4 cases in all datasets investigated and forms thus hemispherical formation with other kernel types. It seems only as long as class grain is low (2-3) and subsequently class borders are few, it is able to compete with other kernels and can even be superior to other kernel types. In our interpretation, there are two separate reasons for this that can be tracked down to RBF kernel design: (a) RBF makes no nonlinear transformation (border optimization) of feature weights like the other kernel types. (b) Gaussian weight function behind RBF kernel upweighs (counter-intuitively) the smallest differences between instances and downweighs largest ones. This effectively leads to an equalization of differences between all instances, and subsequently cores and borders are diluted. However, as we know [6], Gaussian based weight functions

perform well at unsupervised (clustering) tasks where borders are virtually non-existent and emphasis is on finding the cores (of clusters). The nature of classification vs clustering tasks is fundamentally different in the task outset: for clustering the algorithm mainly needs (and is able only) to create class *cores*, while classification algorithm is able to additionally define class *borders* from provided training data. We propose that since with clustering tasks all distances are relatively small, even smaller than with 2-grams, Gaussian style accentuation of small differences is the reason why the Gaussian function has empirically proven the most popular instance similarity function for clustering methods such as SOM and FNN [6].

7 Discussion and Conclusions

In this paper, we have presented probable explanations for the regional formations (relative strengths) of classifiers deriving from the properties of classifiers themselves. Our findings suggest that best classifier prediction is very much possible. Many of the formations and phenomena explained here were left unexplained in [9], (which featured mainly the clearly hemispherical formation of SVM vs NB).

We showed that classifiers respond to varying values of prediction factors (P, N, nrfeat), which furthermore can largely be attributed to their algorithm design (e.g. weight functions, size of training model). For instance, differences in weight functions result in the observed layered or '1-2-1' formation where one classifier outperforms the other at low/high ranges of a factors and the other at mid range. (Cf. definition of Dtree vs NB difference defined in [12]: 1-2-1 formation was shown in terms of P and N factors and 1-2 formation in terms of nrfeat).

To sum up the classifier formations, yin-yang formations result from ensembling decision tree pairs, layers occur when decision trees are ensembled with both NB and Dstump, hemisphere formations are mainly for SVM vs NB. SVM kernels make formations that depend on the core-border ratio of the dataset and cannot thus be well predicted using these three factors. As for prediction factors, the fact that feature-based classifiers are mostly separable by nrfeat (a feature-based factor) and instance-based classifiers (SVMs) are separable primarily by P and N (instance-based factors) is not very surprising after all. Instance-based classifiers kNN and FNN formed no discernible or convex region in PN+nrfeat space in either WSD or genset datasets.

The generalization of these formations to three types help the designers of OE method [9] to gain higher prediction accuracies and classifier developers to perhaps develop better classification algorithms that exploit this new-found knowledge of classifier regionality. Results from separate OE tests with gensets (this paper) and WSD datasets [9] result in standard 2-5% net gain (depending largely on selected best base classifiers which in turn depends on prediction factors). We can state, for example, that SVM and NB do not necessarily form maximal complements for all dataset types although they do for WSD [9] and text classification [4]. This is especially if the datasets are favorable to decision trees (as gensets are), in which case optimal ensembling of at least one decision tree comes into question.

As to why SVM and NB perform best at WSD [9] and Dtrees (CART) at gensets is an issue for further research. However, we can consider the equal fval distribution of

gensets as opposed to unequal fval distribution (binary features in WSD and Gaussian distribution in protein datasets) as definitely related to that adaptability. The fact that SVM can compete with gensets-binfeat (which are similar to WSD and protein datasets in their respective fval distributions) is proof of SVM's adaptability to sparse datasets. We can further generalize classifier adaptability to types of datasets: since with real-world datasets there is a marked difference between low- and high-quality features on one hand and core vs border instances on the other, we can define the competence of SVM as deriving from its explicit class border definition. As for decision trees, pruning of either type certainly helps them to higher accuracy, which is seen from the superiority of CART, REPTree Boosting and Bagging over J48 and PART at both gensets and protein dataset. The reason for relative strength of Dtrees over both SVM and NB at gensets can most likely be attributed to the fval distribution or correct fqual estimation, which in our interpretation and that of [12] are the only relevant factors remaining unquantified in this study and [9]. This (dataset-independent definition of classifier strengths) is a subject of further study.

References

1. Aha, D., Kibler, D. Instance-based learning algorithms. Machine Learning. 6:37-66. (1991)
2. Breiman, L. (1996). Bagging Predictors. Machine Learning, 24/2, 123-140.
3. Breiman, L,, Friedman, J., Olshen, R., and Stone, C. (1983). CART: Classification and Regression Trees. Wadsworth, NY.
4. Forman, G., and Cohen, I. Learning from Little: Comparison of Classifiers Given Little Training. In 15th European Conference on Machine Learning and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (2004)
5. Freund, Y. & R.E. Schapire (1996). Experiments with a New Boosting Algorithm, in Proceedings of the Thirteenth International Conference on Machine Learning.
6. Legrand, S., Pulido JGR. A Hybrid Approach to Word Sense Disambiguation: Neural Clustering with Class Labeling. Knowledge Discovery and Ontologies workshop at 15th European Conference on Machine Learning (ECML) (2004).
7. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. and Euler, T. YALE: Rapid Prototyping for Complex Data Mining Tasks. Proceedings of 12th ACM SIGKDD (2006)
8. Quinlan, R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers. (1993)
9. Saarikoski, H., Legrand, S., Gelbukh, A. 2006. Case-Sensitivity of Classifiers for WSD: Complex Systems Disambiguate Tough Words Better. In proceedings of CicLING 2006.
10. 10.Vapnik, V. N. The Nature of Statistical Learning Theory. Springer (1995)
11. 11.Witten, I., Frank, E. Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann (2005).
12. Yarowsky, D. and Florian, R. Evaluating sense disambiguation across diverse parameter spaces. Journal of Natural Language Engineering, 8(4) (2002) 293-311.