Extending Nearest Neighbor Classification with Spheres of Confidence

Ulf Johansson[1], Henrik Boström[2], Rikard König[1]

[1]School of Business and Informatics, University of Borås, Sweden
[2]School of Humanities and Informatics, University of Skövde, Sweden
ulf.johansson@hb.se, henrik.bostrom@his.se, rikard.konig@hb.se

Abstract

The standard kNN algorithm suffers from two major drawbacks: sensitivity to the parameter value *k*, i.e., the number of neighbors, and the use of *k* as a global constant that is independent of the particular region in which the example to be classified falls. Methods using weighted voting schemes only partly alleviate these problems, since they still involve choosing a fixed *k*. In this paper, a novel instance-based learner is introduced that does not require *k* as a parameter, but instead employs a flexible strategy for determining the number of neighbors to consider for the specific example to be classified, hence using a local instead of global *k*. A number of variants of the algorithm are evaluated on 18 datasets from the UCI repository. The novel algorithm in its basic form is shown to significantly outperform standard kNN with respect to accuracy, and an adapted version of the algorithm is shown to be clearly ahead with respect to the area under ROC curve. Similar to standard kNN, the novel algorithm still allows for various extensions, such as weighted voting and axes scaling.

Introduction

A data miner performing predictive classification has many different techniques to choose from. Most approaches consist of a two-step process: first an *inductive* step, where a model is constructed from data, and then a second, *deductive*, step where the model is applied to test instances.

An alternative approach is, however, to omit the model building and directly classify novel instances based on available training instances. Such approaches are named *lazy learners* or *instance based learners*. The most common lazy approach is *nearest neighbor classification*. Given an instance to classify, the algorithm first finds the majority class C_m among the *k* closest (according to some distance measure) data points in the training set. The new instance is then classified as belonging to class C_m. The value *k* is a parameter, and the entire technique is known as *k-Nearest Neighbor* (kNN) classification.

kNN, in contrast to techniques like neural networks and decision trees, performs classification based on local information rather than using a global model covering the entire input space. kNN can, in theory, produce arbitrary shaped decision boundaries, while decision trees and rule-based learner are constrained to rectilinear decision boundaries.

All in all, kNN is a simple and frequently used technique, especially for classification problems. In spite of its simplicity, kNN often performs quite well. One purpose of using kNN is to get an idea of the classification rate that should, at the very least, be achieved by more powerful methods like neural networks or ensemble techniques; see e.g. (Bishop, 1995).

kNN does, however, suffer from two major drawbacks. First of all, the parameter *k* is extremely important. If *k* is too small, the algorithm becomes very susceptible to noise. If *k* is too large, a test instance may be misclassified based on training instances quite far away. Needless to say, there is no "golden" *k*, performing well on a majority of problems, although most data mining tools use a default value of *k=10*. The second drawback is slightly more subtle. Even for a single problem, the optimal value for *k* may vary depending on the particular region in which an example to be classified falls. For example, a 7-4 result for *k=11*, where the seven closest instances vote for a specific class, should intuitively be a very strong support for that class, in contrast to what is provided by a class probability estimate based only on the relative frequency of that class among all votes. A related consequence of how kNN operates, is that the only way of increasing the confidence in the class probability estimates obtained by votes from nearest neighbors is by increasing *k.* As a result, the increased confidence often comes at the cost of a decreased predictive performance, following from the sub-optimal choice of *k*.

More sophisticated kNN algorithms have partly addressed these issues by using voting scheme where each vote is weighted with the distance to the test instance; see e.g. (Zavrel, 1997). However, it should be noted that again the classifications still involve a fixed number of votes, and although several of these methods are less sensitive to the actual choice of *k* compared to standard kNN, the choice can still be of major importance. One straightforward way of finding out which *k* to use, is by means of cross-validation on the training data, which however involves a

significant computational cost. Furthermore, weighted voting methods are still restricted to using a global *k*, i.e., for the entire dataset, rather than a local *k* that is tailored for each instance to classify.

In this paper, we introduce a novel instance-based learner that does not require the number of neighbors as a parameter, but instead employs a flexible strategy for how many neighbors to consider for a specific example that falls into a particular region (hence it uses a local instead of a global *k*). In this study, we investigate how the novel algorithm, in its basic form, compares to standard kNN. It should be noted, though, that the suggested algorithm can easily be extended with more elaborate strategies like distance-weighted voting or axes scaling based on attribute importance, that are not considered in this study.

Method

In this section we first introduce the BuLL (**Bu**bble **L**azy **L**earner) algorithm. In the second part, we describe the details regarding the experiments conducted.

The basic idea of BuLL is to first create spheres ("bubbles") of confidence around each training instance. When later classifying a novel (test) instance, all spheres of confidence covering the test instance are considered. Exactly how the spheres are constructed, and how they are combined when classifying a test instance, varies slightly between the different versions described below.

When constructing a sphere of confidence around a training instance (the *center instance*) in the basic version, all instances are first ordered based on their proximity to the center instance. The radius of the sphere of confidence is then set to the distance to the last instance before the first conflicting instance. A conflicting instance in this context is simply an instance with a different classification than the current instance. This approach creates the largest possible sphere containing only instances classified identically to the center instance. It should be noted that, in this version, the spheres of confidence may very well contain several instances classified identically to the center instance, but no conflicting instances. Typically, spheres in "easy" regions of the input space, will be large and contain many instances, while spheres in harder regions would be smaller and even overlapping with spheres centered around instances with another classification.

When classifying a test instance, the basic version (called *sphere aggregation*) counts only the class labels of each sphere covering the example, and assigns the most frequent class to the test example; i.e. this method does not consider the number of instances covered by each sphere.

The overall aim of the BuLL algorithm is to produce an instance-based learner, not suffering from the two major drawbacks of standard kNN, as described in the introduction. Clearly, the number of spheres participating in a BuLL classification will depend on both the distribution and the density of instances around the test instance. We argue that this is an obvious advantage compared to the use of a predefined, global, parameter like *k* in kNN. In addition, BuLL offers a potentially more informative decision than kNN. A kNN decision will always consist of a fixed number of votes (e.g. 11) while a BuLL decision, in principle, could be based on any number of votes. Voting results from both kNN and BuLL could straightforwardly be used to produce a class probability distribution for each instance. This would permit the examples to be ranked according to the probability of belonging to a certain class; i.e. allowing the method to be evaluated using OC-analysis. In addition, the number of votes (i.e. the number of covering spheres) also gives a clear indication of whether the test instance is in a dense or sparse part of the input space, something that probably should influence the reliability of the classification. Specifically, a test instance might occasionally fall outside all existing spheres of confidence; a clear indication that this is in a part of the input space not well covered by the BuLL model. Exactly how this should be handled if a classification is needed must be determined when implementing the algorithm; the point is that this information is provided by BuLL. In the current study, BuLL uses a standard kNN with *k=5* to break a tie, when necessary to get a prediction for a specific instance.

In this study, we consider several slightly different versions of the basic algorithm described above. In particular, two alternative ways of constructing the spheres, together with one slightly more sophisticated combination strategy are outlined below.

When *eager construction* is used, the radius of a sphere of confidence can be greater than the distance to the closest conflicting instance. More specifically, the idea is to incrementally increase the radius to the next conflicting instance, *as long as the resulting distribution of positive and negative instances is improved*. Since the first sphere, containing only positive examples, would have an unbeatable proportion of 100% positive instances, the algorithm uses the so called *Laplace estimate* for the comparison of distributions.

The main reason for using a Laplace estimate in general is that the basic (maximum-likelihood) estimate does not consider the number of training instances supporting a specific decision, just the proportions. Intuitively, a decision based on many training instances is a better estimator of class membership probabilities. With this in mind, the Laplace estimator calculates the estimated probability as:

$$p_{class\ A} = \frac{k+1}{N+C} \qquad (1)$$

where k is the number of training instances belonging to class A, N is the total number of training instances involved in the decision and C is the number of classes. Specifically, the Laplace estimate does not assign a zero probability to a class even if it is not supported at all. It should be noted that the Laplace estimator in fact introduces a prior uniform probability for each class; i.e. when $k=N=0$, the probability for each class is $1/C$.

The Laplace estimate has previously been used for, among other things, producing *probability estimation trees* (Provost & Domingos, 2003) from decision trees like CART (Breiman et al., 1984), and C4.5/C5.0 (Quinlan, 1993). When applying the Laplace estimate to a decision tree, the calculation is performed in the specific leaf reached by the test instance; i.e. k is the number of training instances supporting class A and N is the total number of training instances reaching that leaf.

When using eager construction, BuLL employs the Laplace estimate in a similar way. Here, however, k is the number of instances supporting the center class and N is the total number of instances in the sphere under consideration. Consequently, BuLL increases the sphere until another extension would decrease the Laplace estimate. For an illustrative example see Figure 1 below. In the figure, x and O represents instances classified as two different classes. The three instances closest to the center instance, **X**, would produce the first Laplace estimate of 5/6 for a sphere covering these four instances. The second possible sphere, covering 15 instances, has a Laplace estimate of 15/17 which is slightly higher. The third sphere considered, however, has a lower Laplace estimate of 17/20; so the resulting radius is the distance to the fourteenth x, which is shown in bold below.

Xxxx Oxxxxxxxxx**x** Oxx O

Figure 1: Eager construction

The third construction strategy, called *total*, works similarly to *eager*, but considers all possible spheres centered at the current instance, with radius equal to the distance to conflicting instances. The sphere chosen is the sphere with the highest Laplace estimate.

As mentioned above, *sphere aggregation* considers only one class count for each of the covering spheres when classifying a test instance. *Instance aggregation*, on the other hand, uses the total number of instances covered by the spheres covering the test instance. More specifically, the class probability distributions from all covering spheres are averaged; i.e. the "vote" of each covering sphere is the Laplace estimate for that sphere.

Datasets and preprocessing

The 18 datasets used in this study are all from the UCI Repository (Blake and Merz, 1998). It should be noted that only binary (two-class) classification problems are used. When preprocessing, all attributes were linearly normalized to the interval [0, 1]. Missing values were, for all numerical attributes, handled by replacing the missing value with the mean value of that attribute. For nominal attributes, missing values were replaced with a new, specific value.

Clearly, it is very important how distance is measured when using instance-based learners. In this study, standard Euclidean distance between instance vectors is used. It should be noted, however, that for nominal attributes the distance is either 0 or 1; i.e. the distance is 0 if the values are identical and 1 otherwise.

Since nominal and ordered categorical attributes are handled differently, an effort was made to identify the character of each attribute in the datasets. Table 1 below summarizes the dataset characteristics. *Ins* is the number of instances. *Con* is the number of continuous input variables, *Bin* is the number of binary input variables, *Ord* is the number of ordered input variables and *Nom* is the number of nominal input variables.

Dataset	Ins	Con	Bin	Ord	Nom
Liver disorders (BLD)	345	6	0	0	0
Breast cancer (BC)	286	0	3	1	5
Cleveland heart disease (Cleve)	303	6	3	0	4
Crx	690	6	4	0	5
German (Germ)	1000	7	2	8	3
Heart disease Statlog (Heart)	270	6	3	0	4
Hepati	155	6	13	0	0
Horse colic (Horse)	368	7	2	5	8
Hypothyroid (Hypo)	3163	7	18	0	0
Iono	351	34	0	0	0
Labor	57	8	3	5	0
Diabetes (PID)	768	8	0	0	0
Sick	2800	7	21	0	0
Sonar	208	60	0	0	0
Spambase (Spam)	4601	57	0	0	0
Tic-Tac-Toe (TTT)	958	0	0	0	9
Votes	435	0	16	0	0
Wisconsin breast cancer (WBC)	699	9	0	0	0

Table 1: Datasets

Experiments

In the experiments, both *accuracy* and *area under the ROC curve (AUC)* are used for evaluation. While accuracy is based only on the final classification, AUC measures the ability to rank instances according to how likely they are to belong to a certain class; see e.g. (Fawcett, 2001). AUC can be interpreted as the probability of ranking a true positive instance ahead of a false positive; see (Bradley, 1997).

For actual experimentation, standard 10-fold cross-validation is used. The reported accuracies are therefore averaged over the ten folds. Following the standard procedure for AUC evaluation, only one ROC curve based on all test set instances from each fold is produced per dataset. In the first experiment, the basic version of BuLL (i.e. using *original construction* and *sphere aggregation*) is compared to standard kNN, with k values of 5, 11 and 17. The second experiment evaluates the different versions of BuLL.

Results

Table 2 below shows the accuracy results for Experiment 1.

	kNN5		kNN11		kNN17		BuLL	
	Acc.	***R***	***Acc.***	***R***	***Acc.***	***R***	***Acc.***	***R***
BLD	0.6000	4	0.6324	1	0.6265	3	0.6324	1
BC	0.7286	3	0.7250	4	0.7357	2	0.7500	1
Cleve	0.8033	4	0.8267	2	0.8267	2	0.8400	1
Crx	0.8638	1	0.8638	2	0.8638	2	0.8623	4
Germ	0.7130	4	0.7290	3	0.7310	2	0.7370	1
Heart	0.8148	4	0.8296	1	0.8185	3	0.8185	2
Hepati	0.8533	1	0.8467	3	0.8267	4	0.8533	1
Horse	0.8222	4	0.8333	1	0.8306	2	0.8306	2
Hypo	0.9737	2	0.9728	3	0.9699	4	0.9756	1
Iono	0.8486	2	0.8400	3	0.8371	4	0.8800	1
Labor	0.9000	1	0.8600	3	0.7800	4	0.9000	1
PID	0.7474	3	0.7434	4	0.7500	2	0.7605	1
Sick	0.9600	2	0.9575	3	0.9518	4	0.9611	1
Sonar	0.8350	2	0.7150	3	0.6800	4	0.8450	1
Spam	0.8848	2	0.8809	3	0.8770	4	0.9083	1
TTT	0.9147	4	0.9663	2	0.9821	1	0.9516	3
Votes	0.9256	2	0.9256	2	0.9186	4	0.9349	1
WBC	0.9710	1	0.9652	4	0.9681	3	0.9696	2
Mean rank	**2.56**		**2.61**		**3.00**		**1.44**	

Table 2: Experiment 1 - Accuracy results. R is ranks.

First of all, it is interesting to note that all three versions of kNN obtain quite similar results overall. On single datasets, however, it is sometimes clearly better to use *k=5* and sometimes clearly better to use *k=11* or *k=17*. This demonstrates not only that the *k* parameter value is very important for kNN, but also that no specific value is better on all datasets.

In this experiment, BuLL clearly outperforms all kNN variants, winning 13 of 18 datasets. To determine if BuLL performance is significantly better than the kNNs, we use the statistical tests suggested by Demšar (2006) for comparing one classifier against several others over a number of datasets; i.e. a *Friedman test* (Friedman, 1937), followed by a Bonferroni-Dunn post-hoc test (Dunn, 1961). With four classifiers and 18 datasets the critical distance (for α=0.05) is 1.03, so based on these tests, BuLL significantly outperformed all three kNN variants.

Table 3 below shows the AUC results. Using average ranks, the only statistically significant difference is that kNN5 is outperformed by all three competitors. This is not very surprising since the mere five votes will only be able to produce five different probability distributions; i.e. many instances will have equal rank, making the ROC curve too simple.

	kNN5		kNN11		kNN17		BuLL	
	AUC	***R***	***AUC***	***R***	***AUC***	***R***	***AUC***	***R***
BLD	0.6365	3	0.6511	2	0.6698	1	0.6179	4
BC	0.6345	4	0.6611	2	0.6489	3	0.6856	1
Cleve	0.8667	4	0.8969	2	0.9016	1	0.8869	3
Crx	0.9082	4	0.9109	1	0.9108	2	0.9107	3
Germ	0.7176	4	0.7502	2	0.7509	1	0.7314	3
Heart	0.8756	4	0.8880	2	0.8946	1	0.8803	3
Hepati	0.8479	4	0.8562	2	0.8678	1	0.8481	3
Horse	0.8547	4	0.8685	2	0.8653	3	0.8712	1
Hypo	0.8758	4	0.8957	3	0.9094	2	0.9222	1
Iono	0.9214	4	0.9323	3	0.9341	2	0.9771	1
Labor	0.9342	4	0.9708	1	0.9533	2	0.9500	3
PID	0.7869	4	0.8011	2	0.8056	1	0.7966	3
Sick	0.8939	4	0.9153	3	0.9225	1	0.9191	2
Sonar	0.9204	2	0.8537	3	0.7978	4	0.9402	1
Spam	0.9348	4	0.9426	3	0.9428	2	0.9524	1
TTT	0.9679	4	0.9974	2	0.9997	1	0.9869	3
Votes	0.9684	4	0.9703	3	0.9722	2	0.9783	1
WBC	0.9881	4	0.9913	2	0.9917	1	0.9890	3
Mean rank	**3.83**		**2.22**		**1.72**		**2.22**	

Table 3: Experiment 1 - AUC results. R is ranks.

Figures 2 to 4 below show ROC curves obtained by kNN5, kNN17 and BuLL on the Iono dataset.

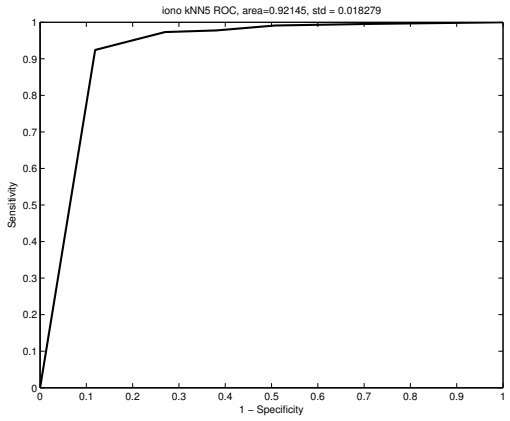

Figure 2: ROC curve kNN5 on IONO. AUC=0.9214

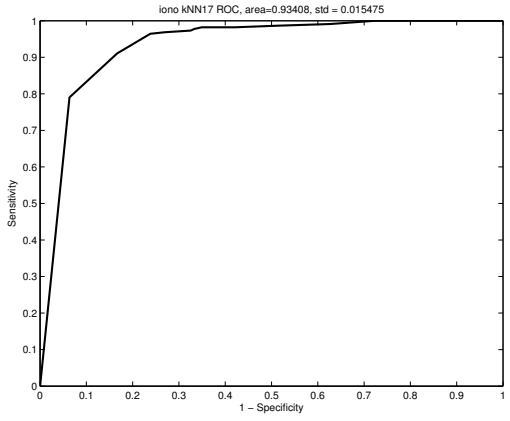

Figure 3: ROC curve kNN17 on IONO. AUC=0.9341

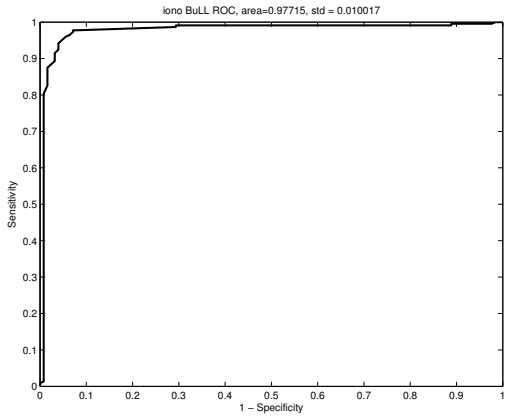

Figure 4: ROC curve BuLL on IONO. AUC=0.9757

Although the average rank is a little lower for kNN17, BuLL still wins 7 of 18 datasets. When comparing BuLL only to kNN11, BuLL wins 8 of 18 datasets. The fact that BuLL is unable to keep the edge when considering AUC instead of accuracy is clearly interesting. With this in mind, one key priority becomes to produce a BuLL variant performing better when using AUC for evaluation.

The overall result from Experiment 1 is nevertheless that BuLL significantly outperforms the kNN variants regarding accuracy, while the results regarding AUC are inconclusive between BuLL, kNN11 and kNN17.

For Experiment 2, we, due to the limited space available, elect to present only obtained ranks and pair-wise comparisons. Table 4 below shows all six BuLL variants, evaluated against each other and the three kNN algorithms. The BuLL variants are describes as *X/Y*, where *X* is the construction strategy (**O**riginal, **E**ager or **A**ll) and *Y* is the combination strategy (**S**pheres or **I**nstances).

	kNN			**BuLL**					
	5	***11***	***17***	***O/S***	***O/I***	***E/S***	***E/I***	***A/S***	***A/I***
BLD	8	3	5	4	6	1	2	8	7
BC	4	5	3	1	5	2	7	9	8
Cleve	8	5	5	1	4	7	1	9	1
Crx	3	3	3	6	7	1	2	9	8
Germ	7	6	4	2	1	3	5	8	8
Heart	7	1	5	2	8	6	2	9	2
Hepati	3	4	7	1	1	5	5	8	8
Horse	7	3	4	5	1	6	2	9	8
Hypo	3	4	5	2	1	6	7	8	8
Iono	5	6	7	3	3	1	2	9	8
Labor	3	8	9	3	3	3	3	1	1
PID	5	7	3	1	2	4	6	8	8
Sick	2	3	6	1	4	5	7	9	8
Sonar	6	8	9	2	6	1	2	5	2
Spam	7	8	9	4	5	3	6	1	1
TTT	4	2	1	3	8	5	7	6	9
Votes	4	4	7	2	1	3	6	9	8
WBC	3	8	6	4	2	1	4	9	7
Mean Rank	**4.9**	**4.9**	**5.4**	**2.6**	**3.8**	**3.5**	**4.2**	**7.4**	**6.1**

Table 4: Experiment 2 - Ranks accuracy

The original BuLL variant is still clearly the most accurate. In addition, all BuLL variants, except the two using *All construction* have lower average ranks than all three kNNs. A deepened analysis showed that *BuLL A/S* and *BuLL A/I*, for some datasets, classified all instances as belonging to the majority class. In other words, the algorithm, when allowed to optimize class distributions in the spheres, produced a set of spheres classifying every test instance identically. Table 5 below shows pair-wise comparisons. The values tabulated are wins for the row technique against the column technique. Using 18 datasets, a sign-test (α=0.05) requires 13 wins for statistical significance. Statistically significant number of wins are underlined.

	kNN			**BuLL**					
	5	***11***	***17***	***O/S***	***O/I***	***E/S***	***E/I***	***A/S***	***A/I***
kNN5	-	11	9	2	6	4	8	14	12
kNN11	7	-	12	5	5	7	7	15	13
kNN17	8	7	-	3	5	6	6	15	13
BuLL O/S	15	13	15	-	11	11	11	16	14
BuLL O/I	12	12	13	6	-	8	10	14	13
BuLL E/S	13	11	12	6	9	-	14	16	14
BuLL E/I	9	11	12	6	7	4	-	15	14
BuLL A/S	3	3	3	2	4	2	3	-	4
BuLL A/I	6	5	5	3	5	4	3	14	-

Table 5: Experiment 2 - Pairwise comparisons accuracy

As seen in Table 5, BuLL O/I and BuLL E/S are both very close to significantly outperforming all three kNNs. Another interesting observation is that BuLL E/S actually is significantly more accurate than BuLL E/I. This, together with the results obtained by BuLL O/S, indicates that the simpler combination strategy of using just the spheres and not the instances, appears to be better when targeting accuracy. Table 6 below shows the ranks obtained using AUC.

	kNN			**BuLL**					
	5	***11***	***17***	***O/S***	***O/I***	***E/S***	***E/I***	***A/S***	***A/I***
BLD	5	3	1	7	6	4	2	9	8
BC	8	6	7	1	2	4	5	9	3
Cleve	9	4	3	6	8	5	7	2	1
Crx	7	3	4	5	9	1	6	8	2
Germ	9	3	2	6	8	4	5	7	1
Heart	7	4	2	5	8	6	9	1	3
Hepati	8	5	3	7	9	4	6	2	1
Horse	8	4	5	3	9	2	6	7	1
Hypo	9	8	3	1	6	2	5	7	4
Iono	9	8	7	1	6	2	5	2	4
Labor	9	2	7	8	1	5	4	6	3
PID	8	4	2	5	6	3	1	9	7
Sick	9	5	2	4	8	6	7	3	1
Sonar	5	8	9	4	3	2	6	7	1
Spam	7	6	5	2	4	3	1	8	9
TTT	7	2	1	5	8	9	6	3	4
Votes	9	7	5	2	3	1	6	3	8
WBC	9	4	2	7	6	8	3	5	1
Mean Rank	**7.9**	**4.8**	**3.9**	**4.4**	**6.1**	**3.9**	**5.0**	**5.4**	**3.4**

Table 6: Experiment 2 - Ranks AUC

Somewhat surprising, the best mean rank is obtained by BuLL A/I. So, although this variant performed poorly when classifying, it is evidently able to rank the instances based on how likely they are to belong to a certain class. This may appear to be an anomaly, but some preliminary follow-up experiments showed that it would probably be possible to obtain much better classification accuracy, just by using a variable threshold. The two algorithms with the best performance overall are BuLL O/S and BuLL E/S. Bull E/S actually has the second best mean rank in both the accuracy and the AUC evaluation. Table 7 below show the pair-wise AUC comparisons. Leaving kNN5 out, there are very few statistically significant differences. It should, nevertheless, be noted that BuLL A/I wins at least 11 of 18 datasets against all competitors.

	kNN			**BuLL**					
	5	***11***	***17***	***O/S***	***O/I***	***E/S***	***E/I***	***A/S***	***A/I***
kNN5	-	1	1	1	6	1	2	6	2
kNN11	17	-	5	10	11	8	9	10	6
kNN17	17	13	-	11	12	9	12	11	7
BuLL O/S	17	8	7	-	14	8	11	11	7
BuLL O/I	12	7	6	4	-	4	5	7	6
BuLL E/S	17	10	9	10	14	-	12	11	7
BuLL E/I	16	9	6	7	13	6	-	11	4
BuLL A/S	12	8	7	7	10	6	7	-	5
BuLL A/I	16	12	11	11	12	11	14	13	-

Table 7: Experiment 2 - Pairwise comparisons AUC

Conclusions

We have in this paper suggested a novel instance based learner called BuLL. BuLL is primarily designed to avoid the dependence of the parameter value *k* in standard kNN, but also to be able to produce accurate probability distributions, based on local information.

The experimentation showed that the basic version of BuLL was significantly more accurate than standard kNN. When using accuracy for evaluation, all versions of BuLL not using the *All* combination strategy, have a positive won-loss record against all kNN variants.

When using AUC instead of accuracy for evaluation, the results are inconclusive. Still, most BuLL variants outperform kNN even regarding AUC. The best BuLL approach is in fact very close to having significantly higher AUC than all kNN variants. Finally, an interesting observation is that the approach having the second worst accuracy, actually obtained the highest AUC.

Discussion and future work

First of all it should be noted that BuLL in this paper was only compared to standard kNN, instead of the more sophisticated versions available. The reason for this is that improvements like weighted voting and axes-scaling based on attribute importance are equally applicable to BuLL. Having said that, one priority is to determine whether these modifications in fact are beneficial for BuLL. There are also, however, other possible modifications, specific to BuLL, that should be explored. One example is to investigate the use of a variable threshold when performing classification. Another option is to introduce more sophisticated voting schemes based on, for instance, each training instance ranking all test instances after how likely they are to belong to a certain class.

Acknowledgement
This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with the Swedish Knowledge Foundation under grant 2003/0104. URL: http://www.infofusion.se.

References

C. Bishop, 1995. *Neural Networks for Pattern Recognition*, Oxford University Press.

C. L. Blake and C. J. Merz, 1998. *UCI Repository of machine learning databases*. University of California, Department of Information and Computer Science.

A. Bradley, 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(6):1145-1159.

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

L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, 1984. *Classification and Regression Trees*, Wadsworth International Group.

J. Demšar, 2006. Statistical Comparisons of Classifiers over Multiple Data Sets, *Journal of Machine Learning Research*, 7:1–30.

O. J. Dunn, 1961. Multiple comparisons among means, *Journal of the American Statistical Association*, 56:52-64.

T. Fawcett, 2001. Using rule sets to maximize roc performance, *15th International Conference on Machine Learning*, pp. 445-453.

M. Friedman, 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *Journal of American Statistical Association*, 32:675-701.

F. Provost and P. Domingos, 2003. Tree induction for probability-based ranking, *Machine Learning*, Vol. 52:199-215.

J. R. Quinlan, 1993. *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

J. Zavrel, 1997. An empirical re-examination of weighted voting for k-nn, *7th Belgian-Dutch Conference on Machine Learning*, pp. 139-148.