Evolving a Locally Optimized Instance Based Learner

Ulf Johansson[1*], **Rikard König**[1] **and Lars Niklasson**[2]
[1]School of Business and Informatics, University of Borås, Sweden
[2] School of Humanities and Informatics, University of Skövde, Sweden

Abstract - *Standard kNN suffers from two major deficiencies, both related to the parameter k. First of all, it is well-known that the parameter value k is not only extremely important for the performance, but also very hard to estimate beforehand. In addition, the fact that k is a global constant, totally independent of the particular region in which an instance to be classified falls, makes standard kNN quite blunt. In this paper, we introduce a novel instance-based learner, specifically designed to avoid the two drawbacks mentioned above. The suggested technique, named G-kNN, optimizes the number of neighbors to consider for each specific test instance, based on its position in input space; i.e. the algorithm uses several, locally optimized k's, instead of just one global. More specifically, G-kNN uses genetic programming to build decision trees, partitioning the input space in regions, where each leaf node (region) contains a kNN classifier with a locally optimized k. In the experimentation, using 27 datasets from the UCI repository, the basic version of G-kNN is shown to significantly outperform standard kNN, with respect to accuracy. Although not evaluated in this study, it should be noted that the flexibility of genetic programming makes sophisticated extensions, like weighted voting and axes scaling, fairly straightforward.*

Keywords: Instance-based learner, kNN, Classification, Genetic programming.

1 Introduction

When performing predictive classification using *instance based learners* (or *lazy learners*), test instances are classified based on their similarity to existing training instances. The most common lazy approach is nearest neighbor classification. When classifying a novel instance, the algorithm first finds the majority class, C_m, among the k closest (according to some distance measure) training set instances. The test instance is then classified as belonging to class C_m. The value k is a parameter to the algorithm, and the entire technique is known as k-Nearest Neighbor (kNN).

kNN, consequently, does not, in contrast to techniques like neural networks and decision trees, use a global model covering the entire input space. Instead, classification is based on local information. This use of neighboring instances for the actual classification makes it, in theory, possible for kNN to produce arbitrarily shaped decision boundaries, while decision trees and rule-based learners are constrained to rectilinear decision boundaries.

Standard kNN, as described above, is a straightforward and frequently used classification technique. In practice, kNN normally performs quite well, in spite of its simplicity. Often, standard kNN is used as a first choice, if nothing else to obtain a lower bound estimation of the classification rate that should be achieved by more powerful methods, like neural networks or ensemble techniques; see e.g. [1].

Standard kNN has, however, at least two major weaknesses. First of all, the parameter value k is extremely important. If k is too small, the algorithm becomes very susceptible to noise. If k is too large, the locality aspect becomes less important, typically leading to test instances being misclassified based on training instances quite different from the test instance. Unsurprisingly, there is no "golden" k, that performs well on a majority of problems, although most data mining tools use a default value of k=10. One straightforward, and frequently used, way of determining k, is by means of cross-validation on the training data, which involves a significant computational cost.

The second drawback is slightly more subtle. Even for a single dataset, the optimal value for k most likely varies over the particular regions of the input space. If cross-validation is used to optimize k, this will most likely produce a k-value that sacrifices performance in some regions to obtain better overall performance. This problem becomes even more apparent if voting results are used to determine class probability estimates. If, as an example, kNN with k=11 is applied to a binary problem, a probability estimate based on the relative frequency interprets a 6-5 vote as a very close call. If, however, the six closest instances all voted for the majority class, this should intuitively be a very strong support for that class, in contrast to what is provided by the class probability estimate.

More sophisticated kNN algorithms have partly addressed these drawbacks of standard kNN, typically by using voting schemes where each vote is weighted with the distance to the test instance; see e.g. [2]. It should, however, be noted that these methods are still restricted to using a global k, i.e., one k-value for the entire dataset, rather than a locally

[*] U. Johansson and R. König are equal contributors to this paper.

optimized *k*. So, although these methods are slightly less sensitive to the actual value of *k*, the choice is still of major importance.

In this paper, we introduce a novel instance-based learner that does not require the number of neighbors to consider as a parameter. The suggested technique, named G-kNN, instead uses a decision tree, evolved by Genetic Programming (GP), to determine how many neighbors to use when classifying a specific test instance, falling into a particular leaf. G-kNN, consequently, uses a set of local *k*-values, instead of one global.

In the experimentation, 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, not considered in this study. As a matter of fact, the flexibility of GP makes it rather straightforward to, for instance, include a set of distance weights in each region (leaf node). GP would, in that case, evolve a decision tree determining the regions, where each region contains an evolved *k*-value and an evolved set of distance weights.

2 Method

As mentioned in the introduction, the overall idea of G-kNN is to use GP to evolve classification trees, where interior nodes represent splits, similar to decision trees like CART [3] and C5.0 [4]. Leaf nodes, however, do not directly classify an instance, but instead use some version of kNN for all test instances reaching that leaf node. In this study, we evaluate three different versions of the suggested technique.

When using *global* G-kNN, a test instance reaching a specific leaf is classified using the *k* nearest neighbors, even if some of those neighbors are not in the specific leaf; see Figure 1 below. In the example, where k=5, the test (black) instance is classified using the five closest instances (gray), despite the fact that four of these are in a different leaf than the test instance. Global G-kNN, consequently, optimizes the separation in regions, and the *k*-values for each region.

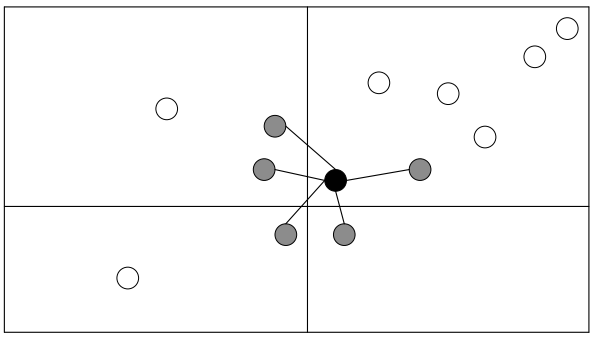

Figure 1: Selection of 5 neighbors using global G-kNN

The *local* G-KNN, on the other hand, only considers training instances falling in the same leaf node as the test instance; see Figure 2 below. Here, the test instance is classified using the five closest instances in the leaf. So, the local technique produces a genetically evolved decision tree, where classification is based on a voting among the *k* nearest instances in the specific leaf.

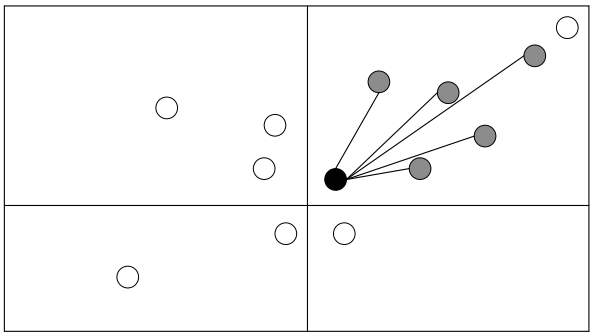

Figure 2: Selection of 5 neighbors using local G-kNN

Comparing local and global G-kNN, the global technique is more similar to standard kNN. The main difference is that several, locally optimized, *k*-values are used instead of just one global *k*. Local G-kNN, in contrast, is more similar to decision trees. Here, however, leaf nodes do not directly classify a test instance based on the majority class of all test instances reaching that leaf, but instead employs a local kNN, restricted to the instances in the specific leaf.

The third version, called *mixed*, simultaneously allows leaf nodes using either global or local kNN; i.e. the resulting tree may very well include leaf nodes of both kinds. Using this version, evolution will divide the input space in regions and then use either a kNN with a locally optimized *k*-value, or a voting among an optimized number of instances in that leaf node when classifying a novel instance.

Figure 3 below shows a sample G-kNN tree evolved from the Statlog Heart dataset. Here, G-kNN uses four different regions where the *k*-values range from 3 to 25. As an example, if the resting blood pressure is higher than 124, the model would use only 3 neighbors if the patient is older than 65, but 25 neighbors if the patient is younger than 65. The numbers enclosed in the parentheses show the number of instances reaching that specific leaf, in this particular run.

```
if resting_blood_pressure < 124.0
 |T: if age > 41
 |      |T: KNN13(71)
 |      |F: KNN1(12)
 |F: if age > 65
 |      |T: KNN3(18)
 |      |F: KNN25(142)
```

Figure 3: Sample global G-kNN tree from Heart dataset

The model evolved in Figure 4 below is a sample mixed G-kNN tree from the WBC dataset. As seen in the figure, G-kNN here uses both local and global kNN, depending on the different regions.

```
if Normal_Nucleoli < 7
 |T: if Marginal_Adhesion > 1
 |    |T: if Marginal_Adhesion > 3
 |    |    |T: LocalKNN5(84)
 |    |    |F: KNN9(85)
 |    |F: LocalKNN5(355)
 |F: KNN15(105)
```

Figure 4: Sample mixed G-kNN tree from WBC dataset

Figure 5 below describes the representation language used by G-kNN. The sets *F* and *T* describe the available functions and terminals, respectively. The functions are an *if-statement* and three relational operators. The terminals are attributes from the dataset, random real numbers, and *k*-values for kNN. The exact grammar used is also presented, using Backus-Naur form. It should be noted that the strategy (i.e. *global*, *local* or *mixed*) is determined by including different *kNN node* functions in the grammar.

```
F = {if, ==, <, >}
T = {i1, i2, …, in, ℜ, 1, 3, …, 25}

DTree    :- (if RExp Dtree Dtree) | kNN node
RExp     :- (ROp ConI ConC) | (== CatI CatC)
ROp      :- < | >
CatI     :- Categorical input variable
ConI     :- Continuous input variable
CatC     :- Categorical attribute value
ConC     :- ℜ
kNN node :- kNN k-value | LocalkNN k-value
k-value  :- 1, 3, …, 25
```

Figure 5: Representation language

The most natural fitness function to use for GP classification is probably accuracy on training and/or validation data. In this study, we settled for using only training accuracy, despite the fact that initial experimentation showed some risk of overfitting. Since kNN in itself is quite powerful, but also rather computationally intense, small populations and few generations were deemed to be sufficient. The GP settings used are found in Table 1 below.

Table 1: GP parameters

Parameter	Value
Crossover rate	0.8
Mutation rate	0.01
Population size	100
Generations	50
Persistence	20
Creation depth	5
Creation method	Ramped half-and-half
Fitness function	Training accuracy
Selection	Roulette wheel
Elitism	Yes

In the experimentation, we wanted to compare the different versions of G-kNN against normal kNN. Since several of the problems used are binary, we decided to use only odd *k*-values for both standard kNN and G-kNN. More specifically, the three *k*-values evaluated for normal kNN are *5*, *11* and *17*, while possible *k*-values for G-kNN are all odd numbers between *1* and *25*.

Normal kNN is, of course, deterministic. GP (and consequently G-kNN) is, on the other hand, inherently indeterministic; i.e. different runs on identical data can produce quite different models. Because of this inconsistency, we decided to run G-kNN 10 times on each fold in the 10-fold stratified cross-validation used for evaluating the standard kNNs. So, the results reported for G-kNN are the average test set accuracies over the ten folds, where the result on each fold is the average from 10 runs. It should be noted that it most certainly would be beneficial for G-kNN to instead select a specific model based on either training or validation accuracy, but, as mentioned above, we did not employ that strategy here.

2.1 Datasets

The 27 datasets used are all publicly available from the UCI Repository [5]. When preprocessing, all attributes were linearly normalized to the interval [0, 1]. For numerical attributes, missing values were handled by replacing the missing value with the mean value of that attribute. For nominal attributes, missing values were replaced with the mode; i.e. the most common value.

It is, of course, very important how distance is measured when using instance-based learners. In this study, standard Euclidean distance between instance vectors is used. For nominal attributes the distance is either 0 or 1; i.e. the distance is 0 if the values are identical and 1 otherwise.

Nominal and ordered categorical attributes could potentially be handled differently; e.g. ordered attributes could use the

same distance function as continuous attributes. On many UCI datasets, it is quite obvious from the problem descriptions that several categorical attributes actually are ordered. In a previous study, where we suggested another kNN algorithm, an effort was made to identify the character of each attribute in the datasets used. Although this was not explicitly targeted in the evaluation, it was obvious to us that this identification of attribute characteristics generally increased kNN accuracy; see [6]. In this study, however, all attributes marked as categorical were, for simplicity, treated as nominal. For a summary of dataset characteristics, see Table 2 below. *Inst.* is the total number of instances in the dataset. *Class* is the number of classes, *Con.* is the number of continuous input variables and *Cat.* is the number of categorical input variables.

Table 2: Datasets

Dataset	Inst.	Class	Con.	Cat.
Breast cancer (Bcancer)	286	2	0	9
Bupa liver disorders (Bld)	345	2	6	0
Cleveland heart disease (Cleve)	303	2	6	7
CMC	1473	3	2	7
Credit-A (Cred-A)	690	2	6	9
Credit-G (Cred-G)	1000	2	7	13
Cylinder bands (CB)	512	2	20	20
Ecoli	336	8	7	0
Glass	214	7	9	0
Haberman (Haber)	306	2	3	0
Heart disease Statlog (Heart)	270	2	6	7
Hepatitis (Hepati)	155	2	6	13
Horse colic (Horse)	368	2	7	15
Iono	351	2	34	0
Iris	150	3	4	0
Labor	57	2	8	8
Lymph	148	4	3	15
Pima Indian diabetes (PID)	768	2	8	0
Sick	2800	2	7	22
Sonar	208	2	60	0
TAE	151	3	1	4
Tictactoe (TTT)	958	2	0	9
Vehicle	846	4	18	0
Votes	435	2	0	16
Wine	178	3	13	0
Wisconsin breast cancer (WBC)	699	2	9	0
Zoo	100	7	0	16

3 Results

Table 3 below shows the results from the experimentation. The results reported are obtained using 10-fold stratified cross-validation. For G-kNN, as described above, the results for each fold is the mean accuracy from 10 runs.

First of all, it is interesting to note that all three versions of kNN obtained quite similar overall results. On single datasets, however, it was 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*-value is very important for kNN, but also that no specific value is substantially better overall.

Table 3: Accuracies

Dataset	kNN			G-kNN		
	k=5	k=11	k=17	Global	Local	Mix
Bcancer	0.6927	0.7485	**0.7518**	0.7332	0.7188	0.7251
Bld	0.6116	0.6234	0.6384	**0.6505**	0.6399	0.6427
Cleve	**0.8282**	0.8247	0.8278	0.8250	0.8253	0.8206
CMC	0.4650	0.4759	0.4732	0.4666	**0.5184**	0.5168
Cred-A	**0.8710**	0.8667	0.8638	0.8652	0.8644	0.8609
Cred-G	0.7320	**0.7390**	0.7330	0.7326	0.7322	0.7296
CB	0.7315	0.7130	0.6981	**0.8006**	0.7937	0.7526
Ecoli	0.8692	0.8662	0.8335	0.8706	0.8637	**0.8709**
Glass	0.6784	0.6357	0.6177	0.6944	0.6867	**0.7022**
Haber	0.6926	**0.7452**	0.7223	0.7248	0.7133	0.7185
Heart	0.8037	0.8148	**0.8222**	0.8155	0.8181	0.8104
Hepati	**0.8458**	**0.8458**	0.8200	0.8387	0.8346	0.8341
Horse	0.8098	0.8182	0.8264	**0.8307**	0.8258	0.8288
Iono	0.8547	0.8463	0.8434	0.8697	0.8729	**0.8789**
Iris	0.9533	0.9467	**0.9667**	0.9547	0.9540	0.9567
Labor	**0.8867**	0.8367	0.8000	0.8717	0.8573	0.8530
Lymph	0.8390	0.8262	0.8400	**0.8502**	0.8417	0.8444
PID	0.7291	0.7331	**0.7382**	0.7340	0.7321	0.7318
Sick	0.9634	0.9597	0.9557	0.9644	0.9640	**0.9660**
Sonar	0.8360	0.7260	0.7060	**0.8534**	0.8113	0.8240
TAE	0.5508	0.5104	0.4979	**0.5800**	0.5295	0.5455
TTT	0.8841	0.9530	0.9813	0.9904	0.9776	**0.9917**
Vehicle	**0.6975**	0.6786	0.6845	0.6889	0.6947	0.6911
Votes	0.9335	0.9289	0.9266	0.9344	**0.9584**	0.9544
Wine	0.9660	**0.9775**	**0.9775**	0.9741	0.9666	0.9694
WBC	0.9643	**0.9657**	**0.9657**	0.9571	0.9559	0.9594
Zoo	0.9500	0.8818	0.8827	**0.9651**	0.9362	0.9523
MEAN	**0.8015**	**0.7958**	**0.7924**	**0.8162**	**0.8106**	**0.8123**

Turning to G-kNN, all three versions obtained higher average accuracies than the three standard kNNs. Especially the global G-kNN performed remarkably well, winning 7 datasets and almost always (on 23 of 27 datasets) finishing among the top three techniques.

Naturally, the mean results, averaged over all datasets, give only a crude comparison between the techniques evaluated. With this in mind, ranks for each technique on every dataset are also presented in Table 4 below.

Table 4: Ranks

Dataset	kNN			G-kNN		
	k=5	k=11	k=17	Global	Local	Mix
Bcancer	6	2	1	3	5	4
Bld	6	5	4	1	3	2
Cleve	1	5	2	4	3	6
CMC	6	3	4	5	1	2
Cred-A	1	2	5	3	4	6
Cred-G	5	1	2	3	4	6
CB	4	5	6	1	2	3
Ecoli	3	4	6	2	5	1
Glass	4	5	6	2	3	1
Haber	6	1	3	2	5	4
Heart	6	4	1	3	2	5
Hepati	1	1	6	3	4	5
Horse	6	5	3	1	4	2
Iono	4	5	6	3	2	1
Iris	5	6	1	3	4	2
Labor	1	5	6	2	3	4
Lymph	5	6	4	1	3	2
PID	6	3	1	2	4	5
Sick	4	5	6	2	3	1
Sonar	2	5	6	1	4	3
TAE	2	5	6	1	4	3
TTT	6	5	3	2	4	1
Vehicle	1	6	5	4	2	3
Votes	4	5	6	3	1	2
Wine	6	1	1	3	5	4
WBC	3	1	1	5	6	4
Zoo	3	6	5	1	4	2
MEAN	**3.96**	**3.96**	**3.93**	**2.44**	**3.48**	**3.11**

One very important observation in Table 4 is the fact that the different versions of standard kNN (especially k=5 and k=17) are normally either very good (ranks 1-2) or very poor (ranks 5-6). This picture, together with the fact that it is so hard to estimate a good *k*-value in advance, pinpoints one of the major weaknesses for standard kNN. G-kNN performance, on the other hand, is not only better overall, but also much more stable. As a matter of fact, all three G-kNN versions perform at least fairly well (ranks 1-4) on a large majority of the data sets.

To determine if G-kNN performance is significantly better than standard kNN, we use the statistical tests recommended by Demsar [7] for comparing several algorithms against each other over a number of datasets; i.e. a Friedman test [8], followed by a Nemenyi post-hoc test [9]. The tests use the ranks displayed in Table 4 above.

Comparing six classifiers using 27 datasets, the critical distance (for α=0.05) is 1.45, so based on these tests, G-kNN global obtained significantly higher accuracy than all three kNN versions. Furthermore, the ranks also show that both Mixed and (to a lesser degree) Local G-kNN clearly outperformed standard kNN, even if the differences are not statistically significant.

4 Conclusions

We have in this paper suggested a novel instance based learner, called G-kNN, where genetic programming is used to evolve decision trees with kNN classifiers in the leaves. G-kNN is primarily designed to avoid the dependence on the parameter value *k* in standard kNN, but also to be able to produce accurate predictions, based on local information.

Although all three versions of G-kNN combine ideas from decision trees and kNN, and the only internal difference is the representation language used, their inductive biases are actually quite different. The global technique mostly resembles standard kNN, but locally optimized *k*-values are used instead of just one global *k*. Local G-kNN is more similar to decision trees, but the classification of a test instance is based on neighboring instances in the specific leaf. The mixed strategy, finally, permits a locally optimized combination of both the local and global strategy.

In the experimentation, G-kNN clearly outperformed the three versions of standard kNN used in the evaluation. As a matter of fact, G-kNN trees evolved using the global strategy were significantly more accurate than all kNN versions. In addition, the performance obtained by the other versions of G-kNN were clearly better than standard kNN, although the differences are not statistically significant.

5 Discussion and future work

It must be noted that the result reported for G-kNN in this study most likely should be regarded as baseline performance. Almost any reasonable heuristics for selecting specific G-kNN models would probably improve the performance significantly. With this in mind, it is a key priority of future studies to find and evaluate such heuristics.

As described above, we believe that G-kNN could benefit even more than standard kNN from extensions like weighted voting and axes scaling, due to the inherent ability to optimize the model based on local information. This, however, remains to be verified through experimentation.

The suggested method is obviously much more computationally intense than standard kNN. With this in mind, G-kNN performance must in future studies be compared to techniques like neural networks, SVMs and different ensemble techniques. Having said that, we believe that G-kNN, enhanced with, for instance, genetically evolved local attribute weighting and axes scaling, may be able to compete with the more powerful techniques.

ACKNOWLEDGMENT

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).

6 References

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

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

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

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

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

[6] U. Johansson, H. Boström and R. König, Extending Nearest Neighbor Classification with Spheres of Confidence, *21st Florida Artificial Intelligence Research Society Conference (FLAIRS 08)*, AAAI Press, pp. 282–287, 2008.

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

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

[9] P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.