Intelligent Information Systems VIII
Proceedings of the Workshop held in
Ustroń, Poland, June 14-18 , 1999

Weighting and selection of features.

Włodzisław Duch and Karol Grudziński
Department of Computer Methods, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.
E-mail: duch,kagru@phys.uni.torun.pl

Abstract. Several methods for feature selection and weighting have been implemented and tested within the similarity-based framework of classification methods. Features are excluded and ranked according to their contribution to the classification accuracy in the cross-validation tests. Weighting factors used to compute distances are optimized using global minimization procedures or search-based methods. Our experiments show that, for some datasets, these methods give much better results than classical nearest neighbor methods.

Keywords: similarity based methods, kNN, optimization, feature selection, classification

1 Introduction.

Recently we have proposed a general framework for similarity based methods (SBM) [1, 2]. Probability $p(C_i|\mathbf{X};M)$ that a vector $\mathbf{X}$ of an unknown class belongs to class C_i requires specification of the classification model M (parameter values and procedures employed). If the model is based on similarity distance measures $d(\mathbf{X},\mathbf{X}^p)$ are used to estimate similarity of the vector $\mathbf{X}$ to some reference vectors $\mathbf{X}^p$. Parameters and procedures that minimize the probability of misclassification include the choice of $d(\cdot)$ function used to compute similarities (usually distances), the choice of the weighting function $G(d(\mathbf{X},\mathbf{X}^p))$ estimating contribution of reference vector $\mathbf{X}^p$ to the probability of classification, procedures for selection of the reference vectors $\mathbf{X}^p$ from the training set, and the choice of the total cost function that is minimized at the training stage, and several other factors and procedures.

An important aspects of SBM methods concerns selection and weighting of features. It is well known that instance (memory) based algorithms degrade in performance (prediction accuracy) when faced with many features that are not necessary for predicting the desired output. Many methods of feature selection and weighting have been developed so far. Wettschereck and Aha [3, 4] proposed a five-dimensional framework to characterize different methods of feature weighting. We have developed and tested several new feature selection methods useful for k–NN and other SBM methods. They are based either on the global minimization of cost functions [1] or on search techniques. In the framework of Wettschereck and Aha they use feedback, do not change

feature representation, use global weights, do not use task specific knowledge and perform both feature selection and feature weighting.

In this paper we concentrate on performance of the search-based methods, presented in the next section. In the third section some illustrative results are given. A short discussion closes this paper.

2 Feature selection and weighting methods.

In the **feature dropping** algorithm features are removed consecutively, one at a time, and the best-first search (BFS) strategy is used. The leave-one-out test is performed on the training file (in *k*-NN the cost of such test is equal to the cost of classification of all training vectors), and the change in accuracy is noted. Feature leading to the highest improvement of classification accuracy on the training file is selected as the least important and removed from the input set. If there is no improvement the feature which leads to minimal degradation is selected. The procedure is repeated with the number of features reduced by one until a single feature is left. If the number of features is large and there is no improvement for several levels the procedure is stopped, because there is little chance that results will improve at a later stage. This algorithm may be used with any similarity-based method. The number of the leave-one-out evaluations in the worst case is $N \cdot (N+1)/2 - 1$. After the selection procedure all features are ranked according to their importance.

The method is fast if the number of features is not to high but considering the datasets such as DNA with 180 attributes the selection of features requires over 16000 leave-one-out tests. An approximate ranking of features may be done at a lower cost. If all features were completely independent and the effects of feature removal were additive N tests would be sufficient. To make the method more robust we rank features after averaging the results of crossvalidation tests with a single feature removed. An alternative is to perform the feature dropping BFS algorithm using a subset of features identified as promising during the first evaluation, for example using those features that may be removed without degradation of the accuracy of *k*-NN. The feature dropping method may be efficiently parallelized to reduce the time of calculations. Other search strategies, such as beam-search, may be used if the number of features is not too large. After calculation of feature ranks crossvalidation tests with first M best features are performed for $M = 1..N$. Usually the best results are obtained with those features that on average were found useful (did not increase the accuracy after being dropped).

We have also used the **search strategies for feature weighting**. The cost function is simply the number of classification errors. Since features have real-valued weights they have to be initially quantized, either with fixed precision (for example 0.1) or precision that is steadily increased during the progress of the search procedure. Several search schemes were implemented, aiming at speeding up calculations and finding better and simpler weights. Our experiments show that using search algorithm for some databases it is possible to obtain weights leading to higher classification accuracy than those obtained in minimization procedures, in addition saving computational costs. For some databases for which weighting using minimization methods did not improve results best–first search methods performed well. For example in the hypothyroid dataset [5] case it was possible to increase the classification accuracy from 94% to more than 98%.

We have used three minimization procedures: simplex method (a local method), adaptive simulated annealing (ASA, a global method) and multisimplex global minimization method. The local simplex method usually requires less than 100 evaluations and is the fastest but results have large variance. It is a good method to start from to see whether optimization of feature weights works for a particular database. For a dataset having separate training file ASA or multisimplex method converge to similar results in all calculations (the advantage of global minimization). However, global optimization methods are expensive and may require a large number of evaluations of errors for convergence. Therefore we have tested three best first search methods for feature weighting, called here $S0$, $S1$ and the weight-tuning method. The last algorithm is used to search for optimal weighting factors starting from a solution obtained by other methods. Our experiments indicate that the search-based algorithms are faster and usually give better accuracy than minimization methods. Since weighting factors are real-valued they are quantized first, either with fixed precision (in method $S0$ and $S1$) or precision that is steadily increased during the progress of the search procedure (weight-tuning method).

The $S0$ algorithm is 'a mirror image' of the $S1$ algorithm: in the first features are added, starting from a single one, and in the second features are dropped, starting from all features. $S1$ algorithm usually works better and therefore it is described below. In the initial ranking of the features all weighting factors are set to 1 and evaluation with a single feature turned off (weighting factor $w_i = 0$) is made for $i = 1 \ldots N$. Thus the ranking is done in the same way as in the feature dropping selection method. The most important feature has a fixed value of the weighting factor $s_k = 1$ and the optimal weighting factor for the second feature w_l in the ranking is determined by the search procedure. The classifier's performance is evaluated with $s_l = m\Delta \in [0,1]$ (the default is $\Delta = 0.05$) and the remaining weighting factors are all fixed to 1. The search is repeated for feature that has the next-highest rating, with optimal values of weighting factors for higher ranked features kept fixed and lower-ranked features left at 1. Finally after determination of all weighting factors performance of the classifier is evaluated on the test set. The total number of evaluations on the training set is on the order of N/Δ. The method corresponds to quantized version of the line search optimization procedure.

The weight-tuning method is used to tune the weighting factors already found by some other procedure (either by a minimization method or one of the methods described above). In this method weighting factors are changed without initial ranking, from first to the last one. Weights are changed for every feature by adding or subtracting a constant value $w_i \leftarrow w_i \pm \delta$ where δ is a parameter given by the user (by default $\delta = 0.5$). If change of the weighting factor leads to an improvement of the classification accuracy the factor is updated, otherwise it remains unchanged. After the last weighting factor is checked in this way the δ parameter is divided by 2 and the whole procedure repeated. The algorithm is terminated if the difference in classification accuracy during two subsequent iterations is smaller than a given threshold.

3 Results

A simple test to check the feature selection and weighting algorithms is to add a function of a class number as one of the input features. This feature should be selected as the most important one enabling 100% classification accuracy.

In the well known hypothyroid dataset [5] despite the high number of training cases (3772) the *k*-NN method (after selection of *k* and selection of the distance measure) performs only slightly better than the majority classifier, giving classification accuracy of 94.4% on the test set (3428 cases). This is much worse than most other algorithms, including neural networks and logical rules [6]. After applying the BFS feature dropping algorithm from 21 features present in the original dataset only 4 remain (f3, f8, f17, f21), increasing the classification accuracy on the test set by nearly 3% to 97.9% (a significant improvement since the number of cases in the test set is large). Applying weighting method $S1$ and tuning the weighting factors we were able to increase the classification accuracy further to 98.1%. This is probably the best result for this database obtained so far with the minimal distance method although still significantly worse than the result obtained with logical rules [6].

Cleveland Heart data was taken from the UCI [5] repository. 6 of the 303 cases contain missing attributes, therefore they are usually removed in most tests. There are 13 attributes (4 continuos, 9 nominal), and two classes (healthy or sick), with 164 (54.1%) healthy, and 139 (45.9%) cases with various degrees of heart problems.

Although Wettschereck and Aha [3] use untypical testing procedure (30% of data is used for tests and the rest for training) while we use standard 10-fold crossvalidation (averaged 10 times to obtain variance) it should be noted that their feature selection methods have always decreased the accuracy of the *k*-NN classification. On the other hand our approach ranked 3 features (thal, ca, cp or features 13, 12 and 3) as the most useful, giving $80.1\pm0.2\%$ (note the small variance). These are the same features as those used in logical rules [6]. After adding weights to the 3 selected features averaged results are $83.9\pm0.5\%$, very close to the best results, $84.2\%\pm0.4\%$ obtained with only a single feature dropped, and quite close to $83.6\%\pm1.0\%$ obtained with all features used (note the large variance for all features).

Another interesting database is the **hepatobiliary disorders** data obtained from Tokyo Dental and Medical University [6] (536 cases, including 163 test cases, 9 features, 4 classes). *k*-NN with Manhattan distance function gives 77.9% accuracy for this dataset which is already much better than other methods [6] (for example MLP trained with RPROP gives accuracies that are below 70%). After applying feature selection method 4 features were removed (features 2, 5, 6 and 9), increasing accuracy to 79.1%. Using weighted *k*-NN methods it was possible to increase the accuracy further to 82.8%. These results are significantly better than for all other classifiers applied to this data (including IB2-IB4, FOIL, LDA, DLVQ, C4.5, FSM, Fuzzy MLP and K* methods). This data has also been analyzed by Mitra, De and Pal [7] using a knowledge-based fuzzy MLP system with results on the test set in the range from 33% to 66.3%, depending on the actual fuzzy model used.

We have also tried the 3 artificial datasets for the **Monk problems**, popular in machine learning community [5]. For the Monk1 problem (124 train, 432 test cases, 2 classes, 6 features) *k*-NN with Euclidean metric, $k = 1$, gives 89.5% accuracy on the test set, while minimization of the weighting factors with the simplex method increases accuracy to 97.2%, and with multisimplex even to 100%. The best-first search weighting method or feature selection gives also 100% accuracy. For the Monk2 problem (169 train, 432 test cases, 2 classes, 6 features) *k*-NN gives 82.6% which is increased by weighting to 84.5%, still rather disappointing since many rule-based methods may find the original classification rules achieving 100% accuracy, and MLPs are also capable of perfect accuracy. Perhaps this is a difficult problem for *k*-NN and all memory-based

methods without optimization of reference vectors. Accuracy of many other methods does not exceed 80% for this problem [5] although some systems may achieve 100%. For the Monk3 problem (122 train, 432 test, 2 classes, 6 features) *k*-NN gives 89.1% accuracy and is significantly improved by using the simplex minimization (93.3%), multisimplex (92.4%) or ASA (94.2%). The best results, 97.2% accuracy on the training set, was obtained using best-first feature weighting followed by tuning the weighting factors.

4 Summary

As a step towards implementation of general framework for similarity-based methods several algorithms for feature selection and determination of the feature weighting factors have been investigated. Search-based techniques were used instead of minimization both for feature selection and weighting. Test results presented here are only preliminary. We expect that global optimization applied to feature weighting should give even better results although at higher computational costs. Feature selection and weighting methods described in this paper significantly improve results obtained by the straightforward *k*-NN approach, giving additional information about the importance of different features.

A natural network realization of *k*-NN method introduced very recently [8] leads to a model with more parameters and should allow to improve the results even further. It is applicable to problems with an infinite number of output classes and may take into account costs of misclassifications. However, even if such more sophisticated similarity-based methods are used performing feature selection and weighting at the level of simple *k*-NN method may be the only practical solution due to the computational efficiency.

References

1. Duch W. *A framework for similarity-based classification methods*, Intelligent Information Systems VII, Malbork, Poland, 15-19.06.1998, pp. 288-291
2. Duch W, Grudziński K. *A framework for similarity-based methods*. Second Polish Conference on Theory and Applications of Artificial Intelligence, Łódź, 28-30 Sept. 1998, pp. 33-60; Duch W, Grudziński K. *Search and global minimization in similarity-based methods*, International Joint Conference on Neural Networks, Washington, July 1999 (in print)
3. Wettschereck D and Aha D.W. *Weighting Features*. In: 1st Int. conf. on Case-based Reasoning (ICCBR-95), Lisbon, Portugal: Springer-Verlag; Wettschereck, D, Aha, D.W. and Mohri, T. *A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms*, Artificial Intelligence Review 11 (1997) 273-314
4. Aha, D.W. *Feature weighting for lazy learning algorithms*. In: H. Liu and H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective. Norwell MA: Kluwer, 1998.
5. C.J. Mertz, P.M. Murphy. *UCI repository of machine learning databases*, http://www.ics.uci.edu/pub/machine-learning-data-bases.
6. Duch W, Adamczak R, Grąbczewski K, Żal G, Hayashi Y. *Fuzzy and crisp logical rule extraction methods in application to medical data*. In: Computational Intelligence and Applications. Springer (Studies in Fuzziness and Soft Computing, Vol. 23, in print)
7. Mitra S, De R, Pal S. *Knowledge based fuzzy MLP for classification and rule generation*, IEEE Transactions on Neural Networks 8 (1997) 1338-1350
8. Duch W, Grudziński K. *The weighted k-NN method with selection of features and its neural realization*, Fourth Conference on Neural Networks and Their Applications, Zakopane, May 1999 (in print)