Feature Weighting by Maximum Distance Minimization

Jens Hocke and Thomas Martinetz

University of Lübeck - Institute for Neuro- and Bioinformatics
Ratzeburger Allee 160 23538 Lübeck - Germany
hocke@inb.uni-luebeck.de

Abstract. The k-NN algorithm is still very popular due to its simplicity and the easy interpretability of the results. However, the often used Euclidean distance is an arbitrary choice for many datasets. It is arbitrary because often the data is described by measurements from different domains. Therefore, the Euclidean distance often leads to a bad classification rate of k-NN. By feature weighting the scaling of dimensions can be adapted and the classification performance can be significantly improved. We here present a simple linear programming based method for feature weighting, which in contrast to other feature weighting methods is robust to the initial scaling of the data dimensions. An evaluation is performed on real-world datasets from the UCI repository with comparison to other feature weighting algorithms and to Large Margin Nearest Neighbor Classification (LMNN) as a metric learning algorithm.

Keywords: feature selection, feature weighting, metric learning, k-Nearest-Neighbor, Relief, Large Margin Nearest Neighbor Classification.

1 Introduction

In pattern recognition tasks data is often described by measurements from different domains. Thus the feature dimensions of the datasets have an arbitrary scaling relative to each other. In many applications the k-Nearest-Neighbor classifier (k-NN) [1] is applied, because it is the simplest non-linear classifier and, which is even more important, the classification decisions are easily interpretable. The interpretability is due to the k nearest neighbors used for the classification decision. By inspecting these nearest neighbors one can discern the causes for the decision. However, the standard metric used for k-NN is the Euclidean metric, which does not measure the distance according to the relevance of each data dimension but according to their arbitrary scaling. To archive a performance close to state of the art classifiers, the dimensions need to be rescaled according to their relevance. A similar problem is solved by relevance learning in the context of LVQ classifiers [4].

An optimal rescaling has to minimize the classification error $E(X)$ of the k-NN algorithm. Often this problem is called the *feature weighting* problem. We want to find a weight vector $\boldsymbol{w} \in \Re^D, w_\mu \geq 0, \mu = 1, ..., D$ for some given dataset

V. Mladenov et al. (Eds.): ICANN 2013, LNCS 8131, pp. 420–425, 2013.

$X = \{\boldsymbol{x}_i \in \Re^D, i = 1, ..., N\}$ that helps the classifier to minimize $E(X)$. In case the Euclidean distance is used, the weighted distance between two data points $\boldsymbol{x}, \boldsymbol{x}'$ becomes

$$d(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x} - \boldsymbol{x}'||_{\boldsymbol{w}} = \sqrt{\sum_{\mu=1}^{D} w_\mu (x_\mu - x'_\mu)^2}, \tag{1}$$

also called the weighted Euclidean distance. In case there are data dimensions irrelevant for the classification, the weights for these dimensions will be decreased to zero and, through this process, the dimensionality of the data set is reduced. Such a dimensionality reduction is desirable to increase generalization performance and noise robustness.

Several methods for feature weighting have been proposed. Very well known is the Relief algorithm by Kira and Rendell [5]. It iteratively updates a weight vector based on the distance of a data point to the nearest neighbors of the same and different label. This concept has been extended in the Simba algorithm [3] by incorporating the weight vector into the distance measure for finding the nearest neighbors. While both methods were developed for selecting the most important dimensions, they use a weight vector to do so. Further feature weighting methods are I-Relief [7], which also extends Relief, and the loss-margin based algorithm (Lmba) [6], which was derived from the Large Margin Nearest Neighbor Classification described in the next paragraph.

Feature weighting can be seen as a subclass of the *metric learning* problem. In metric learning often a Mahalanobis distance

$$d(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x} - \boldsymbol{x}'||_W = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T W (\boldsymbol{x} - \boldsymbol{x}')} \tag{2}$$

is optimized to improve the k-NN classification. Note that here an entire positive semidefinite matrix W is optimized. The problem becomes equivalent to feature weighting, if W is restricted to a diagonal matrix. A well-known method for metric learning is Large Margin Nearest Neighbor Classification (LMNN) [8–11]. We will have a closer look at LMNN in the next section.

Here we present a simple linear programming approach for feature weighting. It archives a robust rescaling of the feature dimensions by a minimization of the maximum distance between data points of the same class under a minimal distance constraint for differently labeled points. We therefore name it Maximum Distance Minimization (MDM).

2 Methods

Before we introduce our approach, we first want to have a closer look at LMNN. Even though it is a metric learning method, whereas we present feature weighting, both methods are related by the goals they pursue to improve the k-NN classification performance.

2.1 LMNN

The LMNN algorithm is designed to optimize k-NN classification performance. To increase the classification performance, it pursues two goals directly derived from the k-NN. Since data points are classified by k-NN according to their k nearest neighbors, it would be best if for every class the data points from the same class are close together and points from different classes are far away. More precisely, the k closest points to every data point should have the same label. We name the k points $\boldsymbol{x}_j$ closest to point $\boldsymbol{x}_i$ and of the same class target neighbors. Any differently labeled point $\boldsymbol{x}_l$ which is closer than the most distant target neighbor (plus some margin) we name impostor. Now every triple of (i) data point, (ii) target neighbor and (iii) impostor is optimized. Additionally, the distance between all datapoints from the same class is minimized. The entire optimization can be done by the following positive-semidefinite program:

$$\min\ (1-\mu)\sum_{i,j\rightsquigarrow i} ||\boldsymbol{x}_i-\boldsymbol{x}_j||^2_W + \mu \sum_{i,j\rightsquigarrow i,l} (1-y_{il})\xi_{i,j,l} \text{ s.t.} \tag{3}$$

$$||\boldsymbol{x}_i-\boldsymbol{x}_l||^2_W - ||\boldsymbol{x}_i-\boldsymbol{x}_j||^2_W \geq 1-\xi_{i,j,l} \tag{4}$$

$$\xi_{i,j,l} \geq 0 \tag{5}$$

$$W \succeq 0. \tag{6}$$

The notation $j \rightsquigarrow i$ means $\boldsymbol{x}_j$ is a target neighbor of $\boldsymbol{x}_i$. To indicate that $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ have the same label, $y_{i,l}$ is used, which equals one if that is the case and zero otherwise. μ is a parameter for weighting the two terms, and $\xi_{i,j,l}$ are the slack variables. By restricting the matrix W to a diagonal matrix, the optimization problem can be cast into a linear program.

There are a couple of problems with the above formulation. First, the target neighbors need to be chosen prior to the optimization. This is usually done based on the initial Euclidean distance. By this selection the final result of the optimization depends on the initial scaling of the dimensions. Second, there are many parameters to optimize due to the large number of slack variables. And third, there is a large number of constraints to keep track of.

2.2 MDM

Our Maximum Distance Minimization (MDM) is a feature weighting method. The objectives are not as closely linked to the concept of the k-NN as it is the case for LMNN, but it avoids the target neighbor selection problem and has much fewer parameters to optimize. With its very general objective it might also be well suited as a preprocessing step for other classifiers. Like LMNN we try to minimize the distance of data points of the same class. However, in our case we do not look at local neighbors but try to minimize the maximum distance between all pairs of data points of the same class, while keeping the pairwise distance between data points of different classes large. This means that we aim to get all data points of the same class as closely together as possible. This is

a very global optimization. Formally, we are solving the following constrained optimization problem

$$||\boldsymbol{x}_i - \boldsymbol{x}_l||^2_{\boldsymbol{w}} \geq 1 \; \forall i, l : y_i \neq y_l \tag{7}$$

$$||\boldsymbol{x}_i - \boldsymbol{x}_j||^2_{\boldsymbol{w}} \leq r \; \forall i, j : y_i = y_j \tag{8}$$

$$\min_{\boldsymbol{w}} r \quad w_\mu \geq 0 \; \forall \mu, \tag{9}$$

where y_i, y_l, and y_j are the class labels of $\boldsymbol{x}_i$, $\boldsymbol{x}_l$, and $\boldsymbol{x}_j$.

The above problem can be formulated as a linear program[1]. For this formulation the number of constraints grows quadratically with the number of data points, but in contrast to LMNN we only have few parameters to optimize, namely only the weights w_μ. In contrast to LMNN, our optimization problem is always solvable, even without slack variables. While LMNN uses triples of points for optimization, here we only look at pairs. By summing the squared distances of target neighbors, LMNN uses a soft penalty for large distances. MDM has a hard penalty, which punishes only the most distant pairs. This may make MDM more sensible to outliers and noisy data. However, in our experiments we did not notice this problem. Of course softness can be added to MDM as well as target neighborhoods, but thereby some of the advantages of MDM would be lost.

3 Experiments

We evaluated MDM on datasets from the UCI repository [2]. MDM was compared with results based on standard Euclidean distance as well as obtained with the feature weighting algorithms Relief, Simba and the diagonally restricted LMNN (D-LMNN). As a reference, we also determined the result obtained with the unrestricted LMNN as a metric learning method. The parameter μ was chosen to be 0.5 for LMNN, because this was described by the authors to be a good choice. For D-LMNN μ was set to one, because for lower values it tended to reduce the dimensions too much. The datasets were split into 50% training points and 50% test points. After optimizing the weight vectors on the training data, the k-NN error rate was determined on the reweighted test data. Every algorithm was trained and tested on 10 different splits of every dataset. Table 1 shows the results obtained without any preprocessing of the data. In Table 2 the dataset dimensions were rescaled in a preprocessing step so that for every dimension the data points had a normalized distribution.

Without preprocessing (Table 1) MDM achieves the best error rates. It significantly improves the error rates of standard k-NN, while Relief and Simba often even deteriorate the classification performance. For the normalized datasets the results change drastically as it can be seen in Table 2. MDM is the only method for which there is almost no change compared to Table 1. LMNN and D-LMNN have a change in their results due to a different initial selection of target neighbors. Obviously, this selection depends on the intial scaling and can be improved

[1] A implementation is available at `www.inb.uni-luebeck.de/tools-demos/mdm/mdm.m`

Table 1. Results for unpreprocessed data. The top entry is the average test error followed by the variance in parentheses. Below the error rates the average rank, again followed by the variance, is given. In case of the feature weighting methods, the rank is equal to the non zero weights. The best results obtained with feature weighting are indicated by bold face.

	Euclidean	MDM	Relief	Simba	D-LMNN	LMNN
Iris	3.33(1.81)	**2.93(1.97)**	3.60(2.18)	4.40(1.89)	3.33(1.44)	3.07(1.89)
	4.00(0.00)	4.00(0.00)	4.00(0.00)	3.90(0.01)	4.00(0.00)	4.00(0.00)
Wine	31.00(4.52)	**4.00(2.23)**	34.67(3.05)	34.44(3.05)	4.33(1.99)	5.22(2.03)
	13.00(0.00)	11.50(0.52)	13.00(0.00)	12.90(0.01)	13.00(0.00)	12.30(0.21)
Breast	39.68(2.21)	**3.60(0.69)**	36.05(11.54)	39.77(2.21)	4.44(0.99)	3.80(0.57)
Cancer	10.00(0.00)	9.50(0.25)	9.90(0.01)	9.90(0.01)	10.00(0.00)	9.00(0.00)
Pima	30.78(1.99)	**28.49(1.87)**	30.08(2.19)	30.86(1.68)	29.14(2.62)	29.14(2.33)
Diabetes	8.00(0.00)	8.00(0.00)	7.50(0.08)	7.50(0.52)	8.00(0.00)	8.00(0.00)

Table 2. Results for the preprocessed datasets. The data dimensions of each dataset were normalized so that the data points have zero mean and a variance of one. The notation and structure of this table is the same as in Table 1.

	Euclidean	MDM	Relief	Simba	D-LMNN	LMNN
Iris	3.60(1.41)	2.93(1.97)	**2.53(1.33)**	3.73(1.64)	4.27(1.38)	2.67(1.99)
	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)
Wine	5.56(1.48)	**4.00(2.23)**	4.78(2.10)	4.56(2.06)	4.89(1.50)	2.67(1.90)
	13.00(0.00)	12.70(0.05)	13.00(0.00)	13.00(0.00)	13.00(0.00)	13.00(0.00)
Breast	3.92(0.63)	3.54(0.67)	3.65(0.57)	4.80(1.29)	**3.30(0.63)**	3.57(0.49)
Cancer	10.00(0.00)	10.00(0.00)	10.00(0.00)	9.70(0.05)	10.00(0.00)	9.70(0.05)
Pima	**27.76(1.44)**	28.49(1.87)	28.20(1.91)	29.35(2.97)	28.49(2.08)	27.92(1.71)
Diabetes	8.00(0.00)	8.00(0.00)	7.60(0.07)	6.60(3.32)	8.00(0.00)	8.00(0.00)

by an appropriate preprocessing. Especially LMNN relies on it. Relief and Simba work now as they are supposed to and become competitive. Even the standard Euclidean distance is becoming a quite good choice. Nevertheless, for all datasets except for the Pima Diabetes dataset the feature weighting and the metric learning methods perform best. MDM is still very competitive, but not clearly better than the others anymore. So the main advantage is that MDM is independent of the initial data scaling. Interestingly, LMNN performed significantly better than the feature weighting methods only on the Wine dataset. This shows that often the extra flexibility of learning the entire Mahalanobis distance is not needed.

4 Conclusion

We have presented a simple linear programming based method for feature weighting. By reweighting the dimensions of the dataset, the scaling is changed so that the classification performance of k-NN is improved significantly compared to using the standard Euclidean distance. However, if the dimensions of the datasets

are normalized according to their variance prior to testing, the performance improvement drops since the other methods profit from this normalization. The experiments on the UCI datasets show that feature weighting is for most datasets competitive to the metric learning algorithm LMNN. This shows that the flexibility of learning the entire Mahalanobis distance is often not needed. Therefore, for very high dimensional datasets with few training points it is advisable to use the simpler feature weighting methods.

The algorithm we presented uses hard boundaries. It does not apply softness by using slack variables. This keeps the number of parameters to optimize low. The missing softness had no negative effect on the datasets we tested, but very noisy datasets with outliers might need some softness to yield robust results. Also, it would be interesting to see the effect of using target neighborhoods in MDM, which would adapt MDM more closely to the k-NN algorithm.

References

1. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
2. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
3. Gilad-Bachrach, R., Navot, A., Tishby, N.: Margin based feature selection - theory and algorithms. In: Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004, pp. 43–50. ACM, New York (2004)
4. Hammer, B., Villmann, T.: Generalized relevance learning vector quantization. Neural Netw. 15(8-9), 1059–1068 (2002)
5. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proc. 9th International Workshop on Machine Learning, pp. 249–256 (1992)
6. Li, Y., Lu, B.L.: Feature selection based on loss-margin of nearest neighbor classification. Pattern Recogn. 42(9), 1914–1921 (2009)
7. Sun, Y., Li, J.: Iterative relief for feature weighting. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 913–920. ACM, New York (2006)
8. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems 19. MIT Press, Cambridge (2006)
9. Weinberger, K.Q., Saul, L.K.: Fast solvers and efficient implementations for distance metric learning. In: Proceedings of the 25th International Conference on Machine Learning, ICML 2008, pp. 1160–1167. ACM, New York (2008)
10. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. 10, 207–244 (2009)
11. Weinberger, K., Sha, F., Saul, L.: Convex optimizations for distance metric learning and pattern classification. Signal Processing Magazine 27, 146–158 (2010)