Global Metric Learning by Gradient Descent

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 classifier can be very competitive if an appropriate distance measure is used. It is often used in applications because the classification decisions are easy to interpret. Here, we demonstrate how to find a good Mahalanobis distance for k-NN classification by a simple gradient descent without any constraints. The cost term uses global distances and unlike other methods there is a soft transition in the influence of data points. It is evaluated and compared to other metric learning and feature weighting methods on datasets from the UCI repository, where the described gradient method also shows a high robustness. In the comparison the advantages of global approaches are demonstrated.

Keywords: Metric Learning, Feature Weighting, k-Nearest-Neighbors, Neighborhood Component Analysis, Large Margin Nearest Neighbor Classification, Relief.

1 Introduction

In many pattern recognition problems, we have datasets with statistical regularities that can be used as prior knowledge. For example, there may be measurements from different domains, which makes the relative scaling of the dimensions in the given dataset arbitrary. Also often the data from different classes lie on submanifolds. If some class labels are available for the data, this information can be captured by a distance metric. This prior knowledge can be used to improve the performance of clustering [13], learning vector quantization [6], or k-Nearest-Neighbor (k-NN) classification [4]. We will focus here on the broadly used k-NN classifier [1]. That often allows competitive non-linear classification, even though it is very simple.

The k-NN classifier labels unknown data points to the most frequently occurring label of the k closest points. Which points are the closest, depends on the distance measure used. A standard choice is the Euclidean distance. However, to increase the probability of correct classification, it might be advantageous to adapt the metric to the data. Very popular for this purpose is the Mahalanobis distance

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T M (\boldsymbol{x}_i - \boldsymbol{x}_j)}, \tag{1}$$

S. Wermter et al. (Eds.): ICANN 2014, LNCS 8681, pp. 129–135, 2014.

where M is a positive semidefinite matrix to be learned. Instead of adapting a distance measure, a matrix W can be learned, that projects the data to a more suitable space. The distance in that space is

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||W\boldsymbol{x}_i - W\boldsymbol{x}_j|| = ||\boldsymbol{x}_i - \boldsymbol{x}_j||_W. \tag{2}$$

This is equivalent to the Mahalanobis distance with $M = WW^\top$. However, W does not need to be positive semidefinite.

The well known metric learning methods either optimize W or M. Probabilistic Global Distance Metric Learning (PGDM) by Xing et al. [13] maximizes interclass distances, while keeping intraclass distances below some threshold. Neighborhood Component Analysis (NCA) [4] as well as Large Margin Nearest Neighbors (LMNN) [11,12] try to free a neighborhood around every data point from differently labeled data points.

Related to metric learning is feature weighting. Here only the data dimensions are rescaled. This is equivalent to optimizing only the diagonal elements of M or W. All other elements are set to zero. Methods designed specifically for this task are Relief [8], Simba [3] and MDM [7]. Relief and Simba both use the closest same and differently labeled data points for optimization, while MDM uses the globally largest distances for same labeled data and the shortest distances for differently labeled data.

In the following we describe a gradient method to find W and evaluate it using UCI datasets [2] in both metric learning and feature weighting tasks.

2 Global Metric Learning

Our goal is to find a matrix $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n) \in \mathbb{R}^{n \times m}$ that projects data points $\boldsymbol{x}_i \in \mathbb{R}^n$ with a given label y_i to a m dimensional space, where the k-NN classification performance is improved. This becomes more likely when for every data point the same labeled points (intraclass) are close together and the differently labeled data points (interclass) are far away. To achieve this we will use a cost function consisting of two parts weighted by a parameter α. The first part of the cost function punishes small distances of pairs from the set of interclass tuples $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : y_i \neq y_j\}$. If the distance one is reached, the cost will become zero due to a cutoff. We chose a squared cost term to penalize close interclass pairs significantly more than far apart pairs and to make a smooth transition at the cutoff. In Figure 1 the squared error term is compared to a linear term, where the gradient is constant and also not continuous at the cutoff. The second part punishes large distances of intraclass tuples from the set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : y_i = y_j\}$. We use the distance measure $||\boldsymbol{x}_i - \boldsymbol{x}_j||_W = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top WW^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)}$ in the cost function

$$E(W) = \alpha \sum_{(i,j)\in\mathcal{D}} \left(1 - \min\left(||\boldsymbol{x}_i - \boldsymbol{x}_j||_W, 1\right)\right)^2 + (1-\alpha) \sum_{(i,j)\in\mathcal{S}} ||\boldsymbol{x}_i - \boldsymbol{x}_j||_W^2. \tag{3}$$

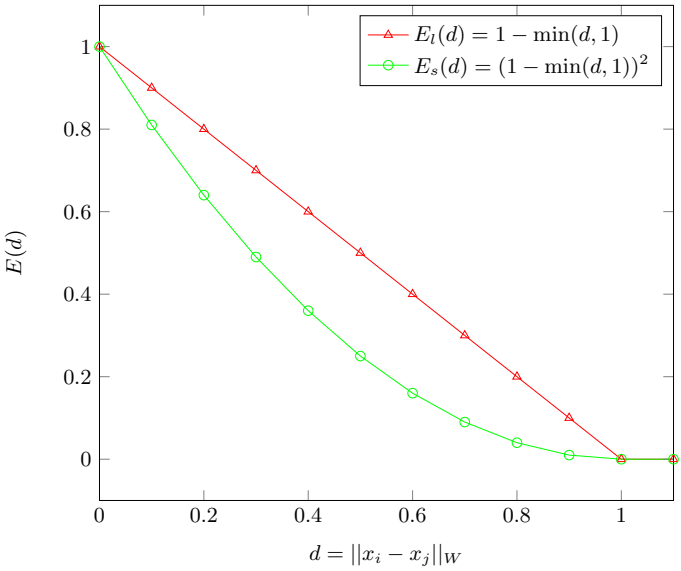

Fig. 1. Interclass cost function comparison. The linear function $E_l(d)$ has a constant gradient. Therefore, all interclass pairs influence the projection matrix W equally. For the non-linear function $E_s(d)$, close by pairs have a larger influence due to their steeper gradient compared to the far apart pairs. In addition, the non-linear function is continuously differentiable at the cutoff at $d = 1$.

The gradient for any weight vector $\boldsymbol{w}_k$ is given by

$$\frac{\partial E(W)}{\partial \boldsymbol{w}_k} = -2\alpha \sum_{(i,j)\in\mathcal{D}} \frac{(1 - \min(||\boldsymbol{x}_i - \boldsymbol{x}_j||_W, 1))}{||\boldsymbol{x}_i - \boldsymbol{x}_j||_W} \left(\boldsymbol{w}_k^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)\right)(\boldsymbol{x}_i - \boldsymbol{x}_j) + 2(1-\alpha) \sum_{(i,j)\in\mathcal{S}} \left(\boldsymbol{w}_k^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)\right)(\boldsymbol{x}_i - \boldsymbol{x}_j). \quad (4)$$

Due to this purely cost function driven design without any hard constraints, there is always a trade off between minimizing and maximizing distances influenced by many tuples. There is a soft transition between tuples close to their desired distance with little influence and tuples far from their desired distance with large influence. All intraclass tuples are taken into account, making this a global approach with a Gaussian prior on the intraclass distances. Due to this property we coin this approach *Global Metric Learning* (GML).

There are of course many optimizers available, that will find good solutions. We use stochastic gradient descent (SGD), because it works fast in case of redundant data. To avoid the need to select a learning rate, we applied variance-based SGD [10,9].

3 Relations to Other Methods

GML is like PGDM [13] a global approach with the same idea of pulling together all intraclass tuples and setting a desired distance for interclass tuples. PGDM, however, enforces the interclass distances as constraints. This makes PGDM dependent on few interclass tuples with small distances and, therefore, may make it more sensitive to noise and outliers. The same idea was also pursued by MDM [7] for feature weighting, where for both intraclass and interclass distances a hard rule was used.

NCA [4] and LMNN [11,12] are in contrast both local methods that try to free the neighborhood of every data point from differently ones. Similar are NCA and GML in that they both use soft transitions, and they optimize a projection matrix instead of the Mahalanobis distance directly.

In the context of visualizing high-dimensional data, Hadsell et al. [5] use almost the same cost term to find a non-linear mapping. Instead of a linear transform, they optimize a siamese architecture of convolutional networks, which may be interesting for finding non-linear metrics.

4 Experiments

For evaluation we used datasets from the UCI repository described in Table 1. The datasets were split into 50% training data and 50% test data. 10 different splits were generated for each dataset. To compare our method to LMNN, NCA and PGDM, we determined the k-NN classification error rates with $k = 3$. The metrics were learned on the training sets and used by k-NN while testing. As a reference the Euclidean metric was also tested.

Table 1. Description of the UCI datasets

Name	Samples	Dimensions	Classes
Iris	150	4	3
Wine	178	13	3
Breast Cancer	683	10	2
Pima Diabetes	768	8	2
Balance Scale	625	4	3
Parkinsons	195	22	2
Seeds	210	7	3

Our method has only the weighting parameter, which we set to $\alpha = 0.9$ to emphasize the interclass distances. The optimal α depends on the data distributions, and it may be beneficial to tune it for every dataset, e.g. by cross-validation. However, this is out of the scope of this work. The weighting parameter of LMNN was set according to the authors advise to $\alpha = 0.5$. NCA and PGDM are parameter free.

In Table 2 the 3-NN classification error rates after metric learning without preprocessing the data are shown, followed by the standard deviation in parentheses. We can see that GML performs well on all datasets. In three cases it is slightly outperformed by PGDM, and only on the Balance Scale data set it is significantly outperformed by NCA. In the three cases where GML is the best, it is by far the best. NCA is the worst on all other data sets and improves the classification performance only marginally or even deteriorates it compared to the standard Euclidean distance.

Table 2. Classification results after metric learning. There was no preprocessing applied to the data sets. The error rates are given in percent followed by the STD in parentheses. The best results are marked in bold face and the worst in italic.

	LMNN	NCA	PGDM	GML	Euclidean
Iris	3.07(1.89)	*4.13(1.93)*	**2.27(1.99)**	2.53(1.72)	3.33(1.81)
Wine	5.22(2.03)	*29.78(5.87)*	5.89(3.67)	**2.89(1.75)**	31.00(4.52)
Breast Cancer	3.80(0.57)	*39.71(2.24)*	**3.60(0.44)**	3.68(0.68)	39.68(2.21)
Pima Diabetes	29.14(2.33)	*31.12(1.94)*	**27.19(1.34)**	27.89(1.74)	30.78(1.99)
Balance Scale	*15.50(1.95)*	**7.60(2.87)**	10.00(1.14)	9.33(1.31)	21.73(1.36)
Parkinsons	14.18(2.90)	*17.24(2.47)*	16.63(3.57)	**10.10(4.04)**	16.73(2.46)
Seeds	6.95(2.88)	*7.71(1.76)*	6.86(2.14)	**4.10(1.19)**	11.33(2.64)

When preprocessing is done by scaling each dimension such that the data distribution has variance one, the results for some methods change dramatically. This is shown in Table 3. While all other methods benefit clearly from the preprocessing, there are only small changes in the results of GML and PGDM. In fact, due to the global cost function, there should be no change at all. However, the stochastic gradient descent may not find the exact optimum in case of GML, and also the small changes in PGDM seem to be due to convergence issues. GML still achieves the best results in three cases, and in the other four cases is never much worse the best one. Note, that for all methods except for GML, there is always one dataset where it performs significantly worse than all the others (the worst results are marked in *italic*).

Table 3. Classification results after metric learning on preprocessed data. The dimensions of the datasets were normalized to variance one.

	LMNN	NCA	PGDM	GML	Euclidean
Iris	2.80(1.93)	*3.20(2.10)*	**2.27(1.99)**	**2.27(1.89)**	3.60(1.55)
Wine	3.00(2.10)	5.67(1.85)	*5.89(3.67)*	**2.67(1.67)**	5.67(1.52)
Breast Cancer	3.65(0.57)	*4.71(0.71)*	**3.60(0.44)**	3.83(0.76)	3.74(0.76)
Pima Diabetes	27.97(1.33)	*29.69(1.77)*	**27.19(1.34)**	27.92(1.76)	27.84(1.93)
Balance Scale	*14.41(1.90)*	**6.71(1.72)**	10.03(1.25)	9.74(1.08)	18.95(0.85)
Parkinsons	**9.49(2.26)**	10.71(2.86)	*15.10(3.36)*	10.71(3.13)	10.00(3.22)
Seeds	6.67(2.06)	7.43(1.33)	*6.86(2.14)*	**4.29(1.63)**	8.29(2.50)

We also tested the feature weighting performance of GML. In Table 4 the results for preprocessed datasets are listed. The experimental set up and the preprocessing is the same as for metric learning, however, GML was only used to optimize the diagonal elements of W, leaving the off diagonal elements to zero. For comparison the feature weighting methods MDM, Relief, and Simba were used. Also in this feature weighting scenario GML performs well compared to the other methods and is again the best in three out seven cases. Of course, for feature weighting there is the same effect as observed for metric learning: Global methods are robust to the initial scaling, while local methods are effected heavily. Because the best results for the local methods were obtained in the preprocessed setting, we only show those.

Table 4. Results for feature weighting. In a preprocessing step the dimensions of the datasets were normalized to variance one.

	MDM	Relief	Simba	GML	Euclidean
Iris	**2.93(1.97)**	3.20(2.10)	**2.93(1.51)**	*3.33(2.01)*	3.60(1.55)
Wine	4.00(2.23)	*4.22(2.39)*	4.00(2.11)	**2.67(2.11)**	5.67(1.52)
Breast Cancer	**3.54(0.67)**	4.06(1.02)	*4.12(0.71)*	4.06(0.79)	3.74(0.76)
Pima Diabetes	*28.49(1.87)*	**27.16(1.43)**	27.86(2.04)	28.26(1.64)	27.84(1.93)
Balance Scale	**18.95(0.85)**	19.17(1.31)	19.23(1.24)	*21.31(2.00)*	19.23(0.95)
Parkinsons	10.00(4.13)	9.18(2.50)	*10.82(3.73)*	**8.78(3.01)**	10.00(3.22)
Seeds	8.29(3.56)	9.62(3.06)	*10.19(3.39)*	**7.33(2.50)**	8.29(2.50)

Interestingly, when the metric learning and the feature weighting results are compared (Tables 3 and 4), most error rates are quite close, showing that often a proper scaling of the dimensions is most important for good classification. Only for the Balance Scale and the Seeds datasets there are significant improvements when using the more powerful metric learning. Scaling only the original dimensions has the advantage that the dimensions are not mixed, which makes it easier to interpret the results, e.g. relevant dimensions.

5 Conclusion

We showed that an optimized similarity metric can easily be obtained by gradient descent. Two weighted cost terms represent the inter- and intraclass distances. Optimizing their sum yields a trade off between both. All intraclass pairs are taken into account, making this a global approach which is independent of the initial scaling. By optimizing a projection matrix instead of the Mahalanobis distance, we avoid to introduce a constraint to enforce the optimized matrix to be positive semidefinite. While GML is not always the best method on the datasets we used, it was the best method most often (together with PGDM), and it never performed much worse than the others. In this sense it was the most robust method, at least on the datasets we used for comparison.

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. Goldberger, J., Roweis, S.T., Hinton, G.E., Salakhutdinov, R.: Neighbourhood components analysis. In: NIPS (2004)
5. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1735–1742. IEEE (2006)
6. Hammer, B., Villmann, T.: Generalized relevance learning vector quantization. Neural Networks 15(8), 1059–1068 (2002)
7. Hocke, J., Martinetz, T.: Feature Weighting by Maximum Distance Minimization. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E.P., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 420–425. Springer, Heidelberg (2013)
8. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proceedings of the Ninth International Workshop on Machine Learning, pp. 249–256 (1992)
9. Schaul, T., LeCun, Y.: Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients. CoRR abs/1301.3764 (2013)
10. Schaul, T., Zhang, S., LeCun, Y.: No More Pesky Learning Rates. In: ICML, vol. (3), pp. 343–351 (2013)
11. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems, vol. 19. MIT Press, Cambridge (2006)
12. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. 10, 207–244 (2009)
13. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. Advances in Neural Information Processing Systems 15, 505–512 (2002)