Pattern Recognition Letters 52 (2015) 48–52

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Maximum distance minimization for feature weighting ☆

Jens Hocke*, Thomas Martinetz

Institute for Neuro- and Bioinformatics, University of Lübeck, Ratzeburger Allee 160, 23538 Lübeck, Germany

ARTICLE INFO

Article history:
Received 17 April 2014
Available online 16 October 2014

Keywords:
Feature selection
Feature weighting
Metric learning
k-Nearest-Neighbor
Relief
Large Margin Nearest Neighbor Classification

ABSTRACT

We present a new feature weighting method to improve *k*-Nearest-Neighbor (*k*-NN) classification. The proposed method minimizes the largest distance between equally labeled data tuples, while retaining a minimum distance between data tuples of different classes, with the goal to group equally labeled data together. It can be implemented as a simple linear program, and in contrast to other feature weighting methods, it does not depend on the initial scaling of the data dimensions. Two versions, a hard and a soft one, are evaluated on real-world datasets from the UCI repository. In particular the soft version compares very well with competing methods. Furthermore, an evaluation is done on challenging gene expression data sets, where the method shows its ability to automatically reduce the dimensionality of the data.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The *k*-Nearest-Neighbor (*k*-NN) algorithm is a popular non-linear classifier [1]. The main advantage of this approach is, that it is simple and the results are easy to interpret. A major disadvantage is, however, that the classification results largely depend on the scaling of the input data and the measured features. The scaling may be very arbitrary, and the most commonly used distance measure, the Euclidean distance, may not be a good choice. Therefore, to assure good classification performance, we need to adjust the scaling and distance measure to fit the data.

By scaling the features through appropriate weighting, the classification performance can be improved significantly. This applies not only to the *k*-NN but also to many other distance based classifiers. The quality of the scaling can be measured by the *k*-NN error rate. An optimal rescaling should minimize the classification error E of the *k*-NN algorithm. The goal is to find a weight vector $\vec{w} \in \Re^D, w_\mu \geq 0, \mu = 1, \ldots, D$ that helps the classifier to minimize E. This problem is referred to as the *feature weighting* problem, which is similar to the problem of relevance learning in the context of LVQ classifiers [2]. It is, however, not as closely tied to one classifier as LVQ.

For the Euclidean distance, the weighted distance between two data points $\vec{x}, \vec{x}' \in \Re^D$ is given by

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

also called the weighted Euclidean distance. This distance measure takes the dimension relevance for classification into account. For irrelevant dimensions, the weights may become zero. This leads to a dimensionality reduction of the data set, which is desirable, because it increases the noise robustness and the generalization performance.

Several methods are available for feature weighting. The most simple approach is to rescale every dimension through normalization of the data distribution variance along every dimension. However, this does not take class label information into account. The Relief algorithm by Kira and Rendell [3] aims to account for this problem. It takes the distance of every data point to its nearest neighbors of the same and a different class. The ratio between these two distances is optimized iteratively such that data from the same class are grouped together. Simba [4] extends this concept. This is done by an iterative update of the nearest neighbors based on the current weight vector for the distance measure. Originally, both methods were developed to select the most important dimensions for classification by learning a weight vector. Other methods for feature weighting are the I-Relief [5] and the loss-margin based algorithm (LMBA). The I-Relief is another extension of Relief that uses the current weight vector when selecting the nearest neighbors. LMBA is derived from the Large Margin Nearest Neighbor Classification described in the next paragraph. For every data point, a circular area is spanned by a *k*-Neighborhood of equally labeled data points. By adapting the weight vector, LMBA tries to clear this area plus some margin from differently labeled data.

Instead of the weighted Euclidean distance, one can also optimize the Mahalanobis distance

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

to improve the *k*-NN classification. Here, an entire positive semidefinite matrix W is optimized. By doing so, we are solving a *metric*

☆ This paper has been recommended for acceptance by Jie Zou.

* Corresponding author. Tel.: +49 451 500 5509.
E-mail address: hocke@inb.uni-luebeck.de (J. Hocke).

http://dx.doi.org/10.1016/j.patrec.2014.10.003
0167-8655/© 2014 Elsevier B.V. All rights reserved.

learning problem, by which features can be decorrelated as a major benefit over feature weighting. In case we restrict W to a diagonal matrix, we again obtain the feature weighting problem. Thus feature weighting is a subclass of metric learning. Often, one wants to stick within this subclass. This is to avoid a mixing of all dimensions, which keeps and allows an interpretation of individual dimensions. Large Margin Nearest Neighbor Classification (LMNN) [6,7] is a well known metric learning method, which is closely linked to the concept of the k-NN classifier. Like LMBA, LMNN tries to free an area spanned by k equally labeled data points plus some margin from differently labeled data points. However, instead of the weighting vector a complete metric is adapted.

Learning a linear transformation of the data space and using the Euclidean distance in the transformed space is equivalent to using the Mahalanobis distance in the original space:

$$||\vec{x}-\vec{x}'||_W = \sqrt{(\vec{x}-\vec{x}')^T W(\vec{x}-\vec{x}')} \tag{3}$$

$$= \sqrt{(\vec{x}-\vec{x}')^T L^T L(\vec{x}-\vec{x}')} \tag{4}$$

$$= \sqrt{(L\vec{x}-L\vec{x}')^T(L\vec{x}-L\vec{x}')} = ||L\vec{x}-L\vec{x}'||_2, \tag{5}$$

with the transformation matrix L and a metric $W = L^T L$. To reduce the dimensionality, Principal Component Analysis (PCA) or Independent Component Analysis (ICA) [8] can find such a linear transformation, independent of the labeling. While PCA decorrelates the dimensions, ICA minimizes their statistical dependence. However, since the transformations are found without label information, the transformed space may not be optimal for classification. This is addressed by Linear Discriminant Analysis (LDA) [9]. It optimizes the ratio between scatter of equally labeled data points (intraclass) and differently labeled data points (interclass) in the transformed space. In the original formulation it was limited to a two class setting, but it has been extended to handle multiple classes [10]. These linear transformations share the mixing of dimensions with the metric learning approaches, making it harder to interpret the results in the transformed space.

It is important to note that often not only the weights are updated, but also prototypes which are used to improve the performance of k-NN on large datasets. For example, this can be done by a fuzzy-artificial immune system [11]. However, reduction of the training dataset is out of the scope of this work.

Here, we focus on feature weighting, which allows a better interpretation of the results, but without the decorrelation ability. In the following, we give a detailed description of our linear programming approach for feature weighting [12] and extend it by allowing soft constraints. Besides evaluating the methods on UCI data, we also demonstrate the dimension reduction capabilities on gene expression data in Section 3.

2. Methods

2.1. Maximum distance minimization

A good weighting vector $\vec{w}$ minimizes the classification error E of the k-NN algorithm. The idea of our approach is that the k-NN classification performance should improve, if equally labeled data points are close together and differently labeled data points are far apart. We want to achieve this by minimizing the maximum distance between all pairs of data points of the same class. To avoid the trivial solution $\vec{w} = \vec{0}$, a constraint on the minimum distance for data points of different classes is imposed. Due to these main ideas, which are illustrated in Fig. 1, we call our method Maximum Distance Minimization (MDM). Note, it is possible to do the opposite and impose a maximum intraclass distance while maximizing the minimum interclass distance (Minimum Distance Maximization). Both approaches are mathematically equivalent.

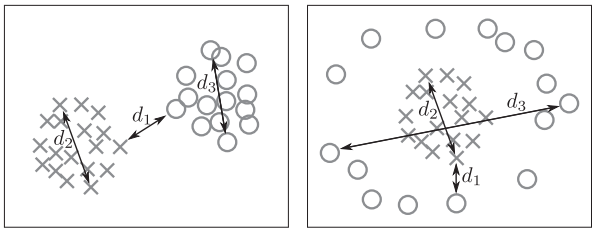

Fig. 1. Part (a) shows two different settings. d_1 denotes the shortest interclass distance. This distance is fixed to one by Eq. (6). The largest intraclass distance for class one (crosses) is d_2 and for class two (circles) d_3. The larger distance of the two (d_2) determines r in Eq. (7) and is minimized.

Given data points $\vec{x}_i \in \Re^D$ with class labels y_i, $i = 1, \ldots, N$, we formally solve the following constrained optimization problem:

$$||\vec{x}_i - \vec{x}_j||^2_{\vec{w}} \geq 1 \qquad \forall i,j : y_i \neq y_j \tag{6}$$

$$||\vec{x}_i - \vec{x}_j||^2_{\vec{w}} \leq r \qquad \forall i,j : y_i = y_j \tag{7}$$

$$\min_{\vec{w}} r \qquad w_\mu \geq 0 \; \forall \mu. \tag{8}$$

The above problem can be formulated as a linear program

$$\min_{\vec{v}} \vec{f}^T\vec{v} \text{ s.t. } A\vec{v} \leq \vec{b}, \;\; \vec{v} \geq \vec{0} \tag{9}$$

with $\vec{v}, \vec{f} \in \Re^{D+1}$, $b \in \Re^{N^2}$, and $A \in \Re^{N^2 \times D+1}$. The vector $\vec{v} = (\vec{w}, r)$ is optimized and with $\vec{f} = (\vec{0}, 1)$, only the r plays a role for the minimization. The constraints are imposed by

$$A_{I(i,j)} = \begin{cases} (-(\vec{x}_i - \vec{x}_j) \circ (\vec{x}_i - \vec{x}_j), 0) & \forall i,j : y_i \neq y_j \\ ((\vec{x}_i - \vec{x}_j) \circ (\vec{x}_i - \vec{x}_j), -1) & \forall i,j : y_i = y_j \end{cases} \tag{10}$$

and

$$b_{I(i,j)} = \begin{cases} -1 & \forall i,j : y_i \neq y_j \\ 0 & \forall i,j : y_i = y_j \end{cases}. \tag{11}$$

Here we use $I(i,j) = (i-1)N + j$ to index the row of A and the element of b. The Hadamard product $A \circ B = (a_{ij} \cdot b_{ij})$ is an element wise multiplication.

In this formulation, the number of constraints grows quadratically with the number of data points. The dimension of the vector $\vec{v}$ to be optimized is $D+1$. Note, that the above constraints can always be fulfilled and, therefore, our optimization problem is always solvable despite having hard constraints.

2.2. Soft maximum distance minimization

Maximum Distance Minimization as introduced above uses hard constraints. This punishes only the most distant pairs and may make MDM sensible to outliers and noisy data. The softness is achieved by introducing slack variables ξ_i for every data point x_i. A punishment for slack variables is added in the optimization criterion weighted by C. This soft approach allows some intraclass distances to be larger than r and hence to reduce the influence of outliers and noise. This is illustrated in Fig. 2. The new optimization problem becomes

$$||\vec{x}_i - \vec{x}_j||^2_{\vec{w}} \geq 1 \qquad \forall i,j : y_i \neq y_j \tag{12}$$

$$||\vec{x}_i - \vec{x}_j||^2_{\vec{w}} \leq r + \xi_i \qquad \forall i,j : y_i = y_j \tag{13}$$

$$\min_{\vec{w}} r + C\sum_i \xi_i \qquad w_\mu \geq 0 \; \forall \mu, \xi_i \geq 0 \; \forall i. \tag{14}$$

The linear program, introduced above, can easily be adopted to the soft version. $\vec{f}$ is extended to $\vec{f} = (\vec{0}, 1, \vec{C})$ and $\vec{v} = (\vec{w}, r, \vec{\xi})$ such that $\vec{f}, \vec{v} \in \Re^{D+1+N}$. The rows of A are changed as follows:

Download English Version:

https://daneshyari.com/en/article/536330

Download Persian Version:

https://daneshyari.com/article/536330

Daneshyari.com