Learning Weighted Metrics Method with Nonsmooth Learning Process

Jiřina, Marcel
2008

cAk

CENTRUM APLIKOVANÉ KYBERNETIKY

České vysoké učení technické v Praze - fakulta elektrotechnická

Learning Weighted Metrics Method with a Nonsmooth Learning Process

Technical report

Marcel Jiřina and Marcel Jiřina, jr.

www@c-a-k.cz

2008

Institute of Computer Science
Academy of Sciences of the Czech Republic

Learning Weighted Metrics Method with a Nonsmooth Learning Process

Marcel Jiřina and Marcel Jiřina, jr.

Technical Report No. V-1026

July 2008

Abstract
A new approach to the Learning Weighted Metrics method for optimized classification of data with 1-NN rule is proposed. New approach is based on application of updating rule similar to one of Madaline neural network, and on dynamic optimization of the step size similar to Runge's method of half step. A short theory is given and the classification ability is demonstrated.

Keywords:
multivariate data, pattern classification, 1-NN classifier, weighted distances, error minimization.

Learning Weighted Metrics Method with a Nonsmooth Learning Process

Marcel Jirina and Marcel Jirina, Jr.

[1] Institute of Computer Science AS CR, v.v.i., Pod vodárenskou věži 2, 182 07 Prague 8 – Libeň, Czech Republic, marcel@cs.cas.cz

[2] Faculty of Biomedical Engineering, Czech Technical University in Prague, Nám. Sítná 3105, 272 01, Kladno, Czech Republic, jirina@fbmi.cvut.cz

Contents

I. INTRODUCTION

We propose a simple nonsmooth updating algorithm for weighted features, prove its convergence and demonstrate its ability on artificial as well as on real-life classification tasks. The updating iterative process can be viewed as a string binding a gift box: it is straight on walls and changes direction on edges. So, it is smooth, even linear most of steps but nonsmooth on "edges", i.e. when specific conditions change.

Recently Paredes and Vidal [18] proposed a method for classification, the Learning Weighted Metrics method (LWM). The LWM method is based on assigning weights to individual coordinate differences of prototype (point, sample, pattern) of the training set and of a query point. Their method is based on learning with the use of the training set only. The aim is to find a proper metrics, which leads directly to better classification by the 1-NN method. In fact, the distances are weighted so that they are modified to be larger or smaller as needed. The weights are assigned to individual points of the training set during the learning process. The error function is based on leaving-one-out approach and is written as follows

$$J_T(W) = \frac{1}{N}\sum_{x\in T} h\left(\frac{d(x,x^{=})}{d(x,x^{\neq})}\right), \tag{1}$$

Where N is number of points of the training set, $h(.)$ is Heaviside step function, $d(.,.)$ is the distance, or dissimilarity metrics, between points, $x^{=}$ is the nearest neighbor of the same class as prototype x, and $x^{\neq}$ is the nearest neighbor of the any other class as prototype x. It is clear that if $d(.,.)$ is Euclidean distance, then the sum in the formula above gives just the number of errors of standard 1-NN method. In [18] it is proposed to multiply true distance between points by a weight. Because the weight is difficult to assign to particular distance in cases when the query point x is unknown in advance, the weights are associated to points of the training set. Thus $d(x, x^{=}) = w(x^{=}).r(x, x^{=})$ and $d(x, x^{\neq}) = w(x^{\neq}).r(x, x^{\neq})$, where $r(.,.)$ is Euclidean distance.

To find optimal weights Paredes and Vidal [18] use gradient descend method, which minimizes J_T (1). For it, the step function in (1) was approximated by sigmoid function to get derivatives. Finally an updating formula was derived for weight w_{ij}. $i = 1, 2, \ldots N$ (the number of points of the training set), $j = 1, 2, \ldots n$ (the dimension). Weight w_{ij} is the weight of particular variable or feature of the distance between points x and x_i. In the updating formula there is also the derivative of sigmoid with steepness factor (gain) β, and a ratio of distance to the neighbor of the same class as x and distance to the neighbor of the other class than x, $r(x) = \dfrac{d(x,x^{=})}{d(x,x^{\neq})}$. Moreover a weighting factor of j-th feature (coordinate or variable), i.e. relative portion of feature j in total squared distance $d(x, \tilde{x})$ between the query point x and a prototype of the training set $\tilde{x}$ is used, $R_j(x,\tilde{x}) = \dfrac{(x_j - \tilde{x}_j)^2}{d^2(x_j,\tilde{x}_j)}$.

With respect to potentially infinite number of distances and thus weights, the number of weights is reduced by two ways.

The first approach in [18] is sharing all the weights within each class, so called Class-dependent Weighting (CW). To each prototype of the training set of N points there are finally assigned as many weights as there are classes C getting total CN weights.

The other weights reduction approach uses sharing weights for each prototype, i.e. for each prototype of the training set, and is called Prototype-dependent Weighting (PW). In this approach it is assumed that all features have the same influence and thus weight is assigned to the prototype as a whole thus getting N weights only. These two approaches can be combined into Class and Prototype Weighting (CPW). In this case there are two sets of weights updated at the same time during learning cycle.

In different weighting approaches weights can be assigned either to coordinate difference of the query point y and prototype of the training set x or to a coordinate of prototype of the training set only. In the latter case prototype x is virtually "moved" from original to a new position. In the former case we can say that both the query point as well as prototype of the training set are "moved" either close one to another for weight lesser than one or farther one from another if the weight is larger than one.

These approaches have rather old origin. Going counter-time, the method of Zhang [29] looks for close neighbors to a query point and trains a local support vector machine that preserves the distance function on the collection of neighbors. The idea of [31] is to learn a function that maps input patterns into a target space such that the L_1 norm in the target space approximates the "semantic" distance in the input space. The method is applied to a face verification task. The learning process minimizes a discriminative loss function that drives the similarity metric to be small for pairs of faces from the same person, and large for pairs from different persons.

Weinberger [27] uses learning a Mahalanobis distance metric for k nearest neighbor classification method by semidefinite programming. The metric is trained with the goal that k nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. The cost function over the distance metrics has two competing terms. The first term penalizes large distances between each input and its target neighbors, while the second term penalizes small distances between each input and all other inputs that do not share the same class. The problem is then reformulated and it is shown that the global minimum of cost function can be efficiently computed.

Goldberger et al. [8] mention that the actual leave-one-out classification error of k-NN is discontinuous because small change of linear transformation matrix may change the neighbor graph and thus affect leave-one-out (LOO) classification performance by a finite amount. Instead, they introduce a differentiable cost function using a softmax function over Euclidean distances in the

transformed space.

Domeniconi et al. [5] search for the maximum margin boundary using the support vector machine (SVM) and determine local discriminant directions of query point neighborhood. The normal direction to local decision boundary identifies the orientation along which data points between classes are well separated. Moreover the gradient vector computed is used for measuring local feature relevance and weighting features. It is also shown that weighting in the case of small weights thus locally reduces dimensionality of the problem.

Peng et al. [19] introduce an adaptive kernel distance for nearest neighbor classification. In the method (AQKNN algorithm) a distance is estimated based on quasiconformal transformed kernels. Again as in already cited papers, the target of the kernel distance is to move points having similar class posterior probabilities to the query point closer to it, while moving points having different class posterior probabilities farther away from the query point. As a result, the class conditional probabilities tend to be more homogeneous in the modified neighborhoods.

A very close but different approach is local linear embedding [24]. The target is to use true dimensionality of data instead of original embedding dimension, and not adaptation of data points directly to need of better classification by 1-NN or *k*-NN method. The method uses weighted distances to *k* nearest neighbors and reconstructs these weighted distances in the target space. Dimensionality of the target space is estimated according to number of smallest eigenvalues [22]. It was shown [23] that the number of smallest eigenvalues should be equal to intrinsic dimensionality or to number of classes minus one.

Hastie and Tibshirani [10] use local discriminant analysis for each query point to estimate an effective metrics for searching neighborhoods. In fact, process linearly shrinks originally ball neighborhood in directions orthogonal to local decision boundaries and after that *k*-NN method is used for classification without dimensionality reduction.

McLeod et al. [12] propose a generalized version of much older classification rule [6] introducing a more complex weighting formula. According to original approach by Dudani [6] the weight of *j*-th nearest neighbor from query point is given by formula

$$w_j = \frac{d_k - d_j}{d_k - d_1} \text{ for } d_k \neq d_j, \text{ and}$$

$$w_j = 1 \qquad \text{for } d_k = d_j.$$

where d_k is a distance of *k*-th neighbor from the query point and d_1 a distance of the first nearest neighbor from the query point. This is done for all classes and class with the largest sum of weights is associated to the query point.

Note finally, a rather strange thing, which was pointed out e.g. in [2], namely the fact that asymptotic error rate of unweighted *k*-NN rule is better than of any weighted *k*-NN rule as proved By Bailey and Jain [1]. Also McLeod et al. [12] formulate hypothesis that the error rate of unweighted *k*-NN rule is lower than that of any weighted *k*-NN rule even when the number of training points is finite. On the other hand McLeod et al. in the same paper [12] proposed weighting rule similar to Dudani's [6] mentioned above. So neither they give credit to their own hypothesis. The conviction of success of weighted metrics follows from three facts. First we work with finite number of points, often with small number of them. Second, there is a proof that (only) half of the available information in an infinite set of points is contained in the nearest neighbor [4]. Finally, experiments often disprove the hypothesis above.

The target of this paper is to show that using a nonsmooth learning process one can eliminate "tuning" parameters especially steepness factor of sigmoid approximation of the step function as in [18], and to use full number of *Nn* weights, i.e. prototype and coordinate-dependent weighting.

The new method proposed here is based on three essential points.

- First, an approach similar to learning process of Madaline is used thus eliminating need of derivatives,
- Second, weights are associated to each feature of each prototype of the training set. Thus there are total *Nn* weights, i.e. the same number of scalar values as there are in the training set (the number of points *N* times dimensionality *n* not counting the class marks), and
- Third, the need of estimation of proper value of the steepness factor β of sigmoid and updating factor μ or factors μ_{ij} is eliminated. The steepness factor β is excluded by the use of nonsmooth analysis. The updating factor μ is set up automatically by approach similar to Runge's method of half-step well known from numerical methods, especially numerical integration of ordinary differential equations [9]. Thus the method has no tuning parameters.

As mentioned above, weights can modify either features of a prototype independently of a feature of any query point, or weights can modify a difference of corresponding features of the prototype and of the query point. In the first case in the end, a weight "moves" a prototype, in the second case a weight modifies a distance. In any case weights are assigned to features and prototypes.

In this paper a new nonsmooth approach to the Learning Weighted Metrics method (LWM) is formulated for case of weighting features as well as for case of weighting feature differences. For it there is proved existence of fixed point of mapping realized by nonsmooth LWM method and its quadratic convergence. A simple strategy of learning parameter control allows eliminate any tuning of parameters, and in Chap. VI the classification quality of new approach to Learning Weighted Metrics is compared with other methods using multivariate data from the Machine Learning Repository [13].

II. LEARNING PROCESS

Here we discuss problem how to optimize classifier, i.e. how to minimize nonsmooth function (1). In more detail, we discuss a problem how to change weights in the case of error function, which is not differentiable. We solve it by approach known from learning of systems with step transfer function like Madaline [20] where the need of derivatives must be eliminated.

Let the training set U of total N points be given in the form of a matrix X^T with N rows and n columns. Each prototype (sample) corresponds to one row of X^T and, at the same time, corresponds to a point in n-dimensional space R_n, where n is the data space dimension. The training set consists of points (samples, rows) of several classes c, i.e. each prototype (sample or row) corresponds to one class. Then, the training set $U = \{x_i\}$, $i = 1, 2 \dots N$, and $x_i = \{x_{ij}\}$, $j = 1, 2 \dots n$; c_i is a class label of this prototype.

A query point y = $\{y_j\}$, $j = 1, 2 \dots n$ is a prototype for which we look for a class c_y.

In the learning process we would like to minimize error criterion (1). Generally for function minimization two approaches exist. The first one based on Newton's method which derives the updating step directly from function minimized using derivatives. The other approaches use updating formula without respect to the function minimized. A one-dimensional example is the bisection method which works by repeatedly dividing an interval in half and then selecting the subinterval in which a root or minimum exists. So, the updating procedure and error criterion are not derived one from another. In contrast to Paredes and Vidal [18] which represents the first approach we use the second one.

We can think about the desirable change in position of the nearest neighbor of the same class and the nearest neighbor of any other class with respect to the query point x, which is considered fixed. This consideration we do individually for each variable (feature) x_{ij}. In fact, by assigning a weight to a variable we move the nearest prototype of the same class and the nearest prototype of any other class in directions of coordinates one coordinate after another individually.

Because the query point x is not known in advance we use all points of the training set successively as query points.

Variable $z = x_{ij}$ (the j-th coordinate of the i-th prototype of the training set) we denote z without indexes ij as well as the same variable of nearest neighbor of the same class $z^=$ and of any other class $z^{\neq}$.

In Table 1 six possible situations of move along one coordinate are summarized. Cases A1 and A2 correspond to z lying between $z^=$ and $z^{\neq}$. It may appear to be good or bad. In both cases in situation A1 we move both $z^=$ and $z^{\neq}$ to the right, i.e. to larger values, making z closer to $z^=$ and farther from $z^{\neq}$. In case A2 we move both $z^=$ and $z^{\neq}$ to the left. Cases B1 and B2 are bad in any case because z is nearer to $z^{\neq}$ of other class. Move of $z^=$ to the right and $z^{\neq}$ to the left (B1) or vice versa (B2) may situation change to a more desirable. Cases C1 and C2 are apparently good as z is nearer to $z^=$ of the same class than to $z^{\neq}$ and then we do nothing. In tables μ is a convergence constant from (0, 1). Its value we discuss later.

Table 1

Updating formulas for all six cases possible. Note that individual variables x_{ij} as well as $x_{ij}^=$ and $x_{ij}^{\neq}$ are written as z, $z^=$, $z^{\neq}$ respectively without indexes i, j here for simplicity.

Case	Description	Formula for $z^=_{new}$	Formula for $z^{\neq}_{new}$
A1	$z^= < z \le z^{\neq}$	$z^= + \mu(z^{\neq} - z^=)$	$z^{\neq} + \mu(z^{\neq} - z^=)$
A2	$z^{\neq} \le z < z^=$	$z^= - \mu(z^= - z^{\neq})$	$z^{\neq} - \mu(z^= - z^{\neq})$
B1	$z^= < z^{\neq} \le z$	$z^= + \mu(z - z^=)$	$z^{\neq} - \mu(z - z^=)$
B2	$z \le z^{\neq} < z^=$	$z^= - \mu(z^= - z)$	$z^= + \mu(z^= - z)$
C1	$z^{\neq} < z^= \le z$	No change	No change
C2	$z \le z^= < z^{\neq}$	No change	No change

Formulas in Table 1 can be rewritten in more comprehensive form, see Table 2, and in form where multiplication of old value by weight is apparent, see Table 3.

Table 2

Updating formulas for features.

Case	Formula for $z^=_{new}$	Formula for $z^{\neq}_{new}$
A	$z^=_{new} = z^= + \mu(z^{\neq} - z^=)$	$z^{\neq}_{new} = z^{\neq} + \mu(z^{\neq} - z^=)$
B	$z^=_{new} = z^= + \mu(z - z^=)$	$z^{\neq}_{new} = z^{\neq} + \mu(z^= - z)$
C	No change	No change

Table 3

Updating formulas for features in form with multiplication by weight.

Case	Formula for $z^=_{new}$	Formula for $z^{\neq}_{new}$

A	$z^{=}_{new} = z^{=}(1+ \mu(z^{\neq}/z^{=}-1))$	$z^{\neq}_{new} = z^{\neq}(1+\mu(1-z^{\neq}/z^{=}))$
B	$z^{=}_{new} = z^{=}(1+ \mu(z/ z^{=}-1))$	$z^{\neq}_{new}=z^{\neq}(1+\mu(z^{=}/z^{\neq}-z/z^{=}))$
C	No change	No change

The $z^{=}_{new}$ and $z^{\neq}_{new}$ according to Table 3 are at any time, in fact, products of original values $z^{=}$ and $z^{\neq}$ and of valid weight. Formulas in Table 4 are then updating formulas for corresponding weights. Thus no prototype of the training set needs to be truly moved, coordinates are changed by the use of weights only.

Table 4
Updating formulas for weights.

Case	Formula for $w^{=}_{new}$	Formula for $w^{\neq}_{new}$
A	$w^{=}_{new} = w^{=}(1+ \mu(z^{\neq}/z^{=}-1))$	$w^{\neq}_{new} = w^{\neq}(1+\mu(1-z^{\neq}/z^{=}))$
B	$w^{=}_{new} = w^{=}(1+ \mu(z/ z^{=}-1))$	$w^{\neq}_{new}=w^{\neq}(1+\mu(z^{=}/z^{\neq}-z/z^{=}))$
C	No change	No change

The updating (training) formulas according to tables above are applied successively to all points of the training set and for each prototype successively for each feature (coordinate). Then error according to (1) is evaluated. This is iteratively repeated until some stopping rule is fulfilled.

III. The Learning Process for Weighted Feature Differences

This case can be viewed as a formal modification of relation presented in Tables 1 – 4. It is easily seen in tables 5 – 8.

Table 5
Updating formulas for all six cases possible when feature differences are modified. Note that individual variables x_{ij} as well as $x_{ij}^{=}$ and $x_{ij}^{\neq}$ are written as z, $z^{=}$, $z^{\neq}$ respectively without indexes i, j here for simplicity.

Case	Description	Formula for $z^{=}_{new}$-z	Formula for $z^{\neq}_{new}$-z
A1	$z^{=} < z \leq z^{\neq}$	$z^{=}$-$z + \mu(z^{\neq} - z^{=})$	$z^{\neq}$-$z + \mu(z^{\neq} - z^{=})$
A2	$z^{\neq} \leq z < z^{=}$	$z^{=}$-$z - \mu(z^{=} - z^{\neq})$	$z^{\neq}$-$z - \mu(z^{=} - z^{\neq})$
B1	$z^{=} < z^{\neq} \leq z$	$z^{=}$-$z + \mu(z - z^{=})$	$z^{\neq}$-$z - \mu(z - z^{=})$
B2	$z \leq z^{\neq} < z^{=}$	$z^{=}$-$z - \mu(z^{=} - z)$	$z^{=}$-$z + \mu(z^{=} - z)$
C1	$z^{\neq} < z^{=} \leq z$	No change	No change
C2	$z \leq z^{=} < z^{\neq}$	No change	No change

Formulas in Table 1 can be rewritten in more comprehensive form, see Table 2, and in form where multiplication of old value by weight is apparent, see Table 3.

Table 6
Updating formulas for feature differences.

Case	Formula for $z^{=}_{new}$-z	Formula for $z^{\neq}_{new}$-z
A	$z^{=}_{new}$-$z = z^{=}$-$z + \mu(z^{\neq} - z^{=})$	$z^{\neq}_{new}$-$z = z^{\neq}$-$z + \mu(z^{\neq} - z^{=})$
B	$z^{=}_{new}$-$z = z^{=}$-$z + \mu(z - z^{=})$	$z^{\neq}_{new}$-$z = z^{\neq}$-$z + \mu(z^{=} - z)$
C	No change	No change

Table 7
Updating formulas for feature differences multiplied by weights.

Case	Formula for $z^{=}_{new}$-z	Formula for $z^{\neq}_{new}$-z
A	$(z^{=}-z)\,(1+ \mu(z^{\neq}-z^{=})/(z^{=}-z))$	$(z^{\neq}-z)(1+\mu(z^{\neq}-z^{=})/(z^{\neq}-z))$
B	$(z^{=}-z)\,(1 - \mu)$	$(z^{\neq}-z)(1+\mu(z^{=}-z)/(z^{\neq}-z))$
C	No change	No change

Formulas in Table 8 are updating formulas for weights of differences of corresponding features. Thus features (coordinates) are changed by the use of weights only.

Table 8

Updating formulas for weights for feature differences.

Case	Formula for $w^{=}_{new}$	Formula for $w^{\neq}_{new}$
A	$w^{=}(1+\mu(z^{\neq}-z^{=})/(z^{=}-z))$	$w^{\neq}(1+\mu(z^{\neq}-z^{=})/(z^{\neq}-z))$
B	$w^{=}(1-\mu)$	$w^{\neq}(1+\mu(z^{=}-z)/(z^{\neq}-z))$
C	No change	No change

Again, as in the previous case the updating (training) formulas according to tables above are applied successively to all points of the training set and for each prototype successively for each feature (coordinate). Then error according to (1) is evaluated. This is iteratively repeated until some stopping rule is fulfilled.

We show later that the use of weighted features (coordinates) of the training set usually leads to much better results than weighted feature differences (i.e., in the end, distances).

IV. Recall

After learning, a simple 1-NN method is used but not using features (coordinates) of points of the training set, but *features* multiplied by corresponding weights. Distance between query point $y = \{y_1, y_2, \ldots y_n\}$ and prototype $x_i = \{x_{i1}, x_{i2}, \ldots x_{in}\}$ of the training set is computed using corresponding weights $w_{i1}, w_{i2}, \ldots w_{in}$ using L2 metrics

$$d_i = \sqrt{\sum_{j=1}^{N} \left(y_j - w_{ij} x_{ij}\right)^2}$$

eventually L_1 metrics

$$d_i = \sum_{j=1}^{N} \left| y_j - w_{ij} x_{ij} \right|$$

for $i = 1, 2, \ldots N$.

The minimal value of distance d_{i*} corresponds to prototype x_{i*} of class c_{i*}, then the query point y is of class $c_y = c_{i*}$.

In the case of *feature differences* (Chap. III) the corresponding feature differences are multiplied by weights. Distance between query point $y = \{y_1, y_2, \ldots y_n\}$ and prototype $x_i = \{x_{i1}, x_{i2}, \ldots x_{in}\}$ of the training set is computed using corresponding weights $w_{i1}, w_{i2}, \ldots w_{in}$ using L2 metrics

$$d_i = \sqrt{\sum_{j=1}^{N} \left(w_{ij}(y_j - x_{ij})\right)^2}$$

eventually L_1 metrics

$$d_i = \sum_{j=1}^{N} w_{ij} \left| y_j - x_{ij} \right|$$

for $i = 1, 2, \ldots N$.

V. Properties of learning process

Here we first discuss questions of convergence and stability in classical sense using mostly work of Ortega [17] as reference. In the second part of this Chapter we use theoretical results on nonsmooth and semi-smooth mappings using works of Clarke [3], and Qi and Sun [21].

The case of weighted prototype features is considered; the case of weighted feature differences is nearly the same and need not be considered in detail here.

Learning process description

The learning process formulated above can be given by mapping G: $R^{nN} \to R^{nN}$ from real nN dimensional space (of weights) onto itself

$$w_{ij}^{(t+1)} = G w_{ij}^{(t)} \tag{2}$$

(i = 1, 2, .. N points, j = 1, 2, .. n dimension) and we will show that equilibrium point (fixed point) w_{ij}^{*} exist, $w_{ij}^{*} = G w_{ij}^{*}$. Note that the initial point is $w_{ij}^{(0)} = 1$ in all three cases A, B, C. At the same time, G depends on μ and individual coordinates of points of the training set.

For nearest neighbors of any prototype $x \in U$ of the same class $x^{=}$ and of any other class $x^{\neq}$ it holds that if $\hat{x}$ is its original position, the position in step t is given by product of original position and corresponding weight $\tilde{x}^{(t)} = w^{(t)} \hat{x}$. This relation is linear and under assumption that points of the training set do not change their case A, B, or C, the process is linear and thus has derivative. Because points may change their case, the process is piecewise linear and linear parts follow up one to another. The process can be viewed as a string binding a gift box: it is straight on walls and changes direction on edges. During process (supposing small μ) each point $\tilde{x}$ moves in the frame of its case or eventually changes case smoothly. Thus Fréchet (functional) derivative G' exists almost everywhere.

Local properties

Mapping G of our learning process can be represented in form of diagonal matrix of type nN x nN. Its elements have form $g_{ij} = 1 + \mu s_{ij}$, where s_{ij} is the term given by case A, B, or C as seen in Table 4. From Table 4 follows that for μ small enough all g_{ij} are positive, then G is positive definite. Moreover for μ small enough all products μs_{ij} are smaller than 1 and thus spectral radius $\rho\{G'(w_{ij}^{*})\}$ of derivative of G' in point w^{*}_{ij} is less than one. Thus the following lemmas are applicable to our case.

Lemma 1 (Ostrowski-Perron theorem [17]). If G: $R^{nN} \to R^{nN}$ has a Fréchet derivative $G'(w_{ij}^{*})$ at the fixed point w_{ij}^{*} and spectral radius $\rho\{G'(w_{ij}^{*})\} < 1$ then the process (2) is locally convergent to w_{ij}^{*}.

Lemma 2 [17, corollary 2.1]. If the conditions of previous lemma hold, then w_{ij}^{*} is exponentially stable.

These two theorems are valid in very small neighborhood of solution w_{ij}^{*} not farther than nearest margin between cases A, B, C where the Fréchet derivative need not exist. Results valid in much larger region follow.

Contraction mapping

For proof that our learning process with $\mu > 0$ is a contraction mapping [3] (see Appendix), we use Table 1.

Case A1 $z^{=} < z < z^{\neq}$. We have to prove that

$$|z^{=}_{\text{new}} - z| \le |z^{=} - z|$$

After substitution there is

$$|z^{=} + \mu(z^{\neq} - z^{=}) - z| \le |z^{=} - z|$$

And we get $\mu(z^{\neq} - z^{=}) \ge 0$.

Case B1 $z^{=} < z^{\neq} < z$. Now we have after substitution

$$|z^{=} + \mu(z - z^{=}) - z| \le |z^{=} - z|$$

and then $\mu(z - z^{=}) \ge 0$.

Case C1 $z^{\neq} < z^{=} \le z$. No change.

Analogous way we deal with cases A2, B2, and C2.

Thus it is seen that our learning process is contraction mapping on a nonempty complete metric space and then this mapping is Lipschitz continuous, and according to Banach fixed point theorem has a unique fixed point.

Alternatively, it can be shown that our mapping is directional contraction [3] and then has a unique fixed point[1].

It all means that our learning process even it is discontinuous (each change of case A, B, C is a discontinuity of derivative) has a unique solution.

[1] **Definition.** [3, Chap. 7.6] A map T: $V \to V$ in complete metric space (V, Δ) is said to be a directional contraction provided T is continuous and there exists a number σ in (0, 1) such that whenever v is such that $Tv \neq v$, there exists w in open $[v, Tv]$ such that

$$\frac{\Delta(Tv,Tw)}{\Delta(v,w)} \le \sigma .$$

Theorem. [3] Every directional contraction of a complete metric space admits a fixed point.

Note that in procedure above most essential is the change of $y^{=}_{new}$. The influence of $y^{\neq}_{new}$ lies in speeding up convergence by enlargement of updating factor $\mu(y^{\neq} - y^{=})$. When $y^{\neq}$ is updated and then $y^{=}$, we have

$$| y^{=} + \mu(y^{\neq}_{new} - y^{=}) - y| \leq |y^{=} - y|,$$

and then

$$\mu(1+\mu)(y^{\neq} - y^{=}) \geq \mu(y^{\neq} - y^{=}) \geq 0.$$

Newton method

The learning process can be viewed also as a Newton method

$$x_{(k+1)ij} = x_{kij} - V_k^{-1} H(x)$$

according to Theorem 1 (see Appendix). Here $H(x) = \text{diag}\,|x^{=} - x|$ which can be written also in form $H = [\,h_{ij}\,] = [|x_{ij}^{=} - x_{ij}|]$. The derivative of $H(x)$ is $h'_{ij} = 1$ for $x_{ij}^{=} \neq x_{ij}$. Let $V_{kij} = h'_{ij}$ and thus $V_k \in \partial H(x_k)$ is the generalized Jacobian. Moreover $H(x)$ is strongly semi-smooth as $V(d) - H'(x + d) \equiv 0$ for all d. From it follows that process converges quadratically.

VI. LEARNING PROCESS CONTROL

Here we discuss the problem of updating factor μ. This value must be carefully set up in advance or sometimes laboriously tuned, see theoretical conditions in previous chapter. To prevent this obstacle we use classical Runge's idea of half step [9]. It is based on an old observation and proof that in iterative processes with controlled step size the use of two steps of half-length of original single step makes error essentially smaller. Of course, it is paid by the cost of twice more computation. As a compromise between error and computation time a simple strategy is used. If error is estimated to be large, the step is halved to get acceptable error. If error is estimated to be very small, the step size is doubled to save time. This approach is often used for solving initial problems of ordinary differential equations [9].

For error control an error value J_T (1) is computed during iteration. As J_T varies, its smallest value found is stored as $J_{T\text{old}}$. At the same time, the corresponding weight matrix is stored.

Control criteria:

- For large error defined so that if $k = 3$ steps with $J_T > J_{T\text{old}}$ in series and for $\mu >$ some small constant (e.g. 0.001 not to have too small values), then the updating factor $\mu = \mu/2$ and restore last best values of weights.
- For small error defined so that if for $k = 3$ steps with $J_T = J_{T\text{old}}$ in series and for updating factor $\mu < 0.5$ (not to get μ too large), then $\mu = 2\mu$; and continue without returning to last best weights (in fact, the state are just the best weights up to now).

The value of $k = 3$ was chosen as compromise between too early reaction to error change and neglecting of apparent error change.

Stopping rule

For stopping the learning process one can use either the mean absolute value of difference of x and $x^{=}$ or function J_T directly.

In the case of the mean absolute value of difference $x - x^{=}$ one simply tests if $|x - x^{=}| < \varepsilon$. This is a standard stopping rule for convergent iterative processes.

In the case of direct use of function J_T, which is a „staircase", function reaching only finite number of values, we search for long interval of no change of its value. Our algorithm is normally convergent and finally for many steps the best (minimal) value $J_{T\text{best}}$ remains the same. However, algorithm combined with step control and with limited shortest step size, can finally slightly oscillate around the best value and thus for many steps there is $J_T > J_{T\text{best}}$. Therefore, unsuccessful steps are counted and after reaching some predefined number (~10-100) the procedure is stopped.

We combine both approaches. The process stops if either $|x - x^{=}| < \varepsilon$ or there is 100 unsuccessful adaptation steps.

VII. EXPERIMENTS

Experiments described below follow the procedures described by Paredes and Vidal [18] as truly thorough tests. Tests consist of three kinds of experiments. The first one is test with synthetic data set for which Bayes limit is known and one can estimate how close a particular approach allows to get close to this limit. The second uses real-life data from UCI Machine Learning repository [13] and also Statlog Project, see [33]. The third experiment deals with high dimensional task of text classification [32].

In all experiments rule of step control was used as described in Chap. VI. In the stopping rule the value of ε was set up to 0.001 and the number of unsuccessful steps was set up to 100. The process is stopped if any of these two conditions is reached. Especially the limit to the number of unsuccessful steps may cause worse results than it would be possible with larger limit and, of course, larger learning time.

A. Synthetic Data

Synthetic data are two dimensional and consist of three two dimensional normal distributions with identical a-priori probabilities. If μ denotes vector of means and $\boldsymbol{C}_m$ is the covariance matrix, there is

Class A: $\mu = (2, 0.5)^t$, $\boldsymbol{C}_m = (1, 0; 0, 1)$ (identity matrix)

Class B: $\mu = (0, 2)^t$, $\boldsymbol{C}_m = (1, 0.5; 0.5, 1)$

Class C: $\mu = (0, -1)^t$, $\boldsymbol{C}_m = (1, -0.5; -0.5, 1)$.

Fig. 1 shows results obtained by different methods for different learning sets sizes from 8 to 256 samples and testing set of 5000 samples all from the same distributions and independent. Each point was obtained by averaging over 100 different runs. Results are shown in Fig. 1. For other methods, i.e. 1-NN method with L2 metrics and variants of the LWM method by Paredes and Vidal [18] values were estimated from literature cited.

In Fig 1 it is seen that the use of weighted features differences gives rather large error while weighted features behave much better in this task. It is seen that in this synthetic experiment LWM1 method presented here outperforms all other methods shown in Fig. 2 and for large number of samples approaches fast to Bayes limit.

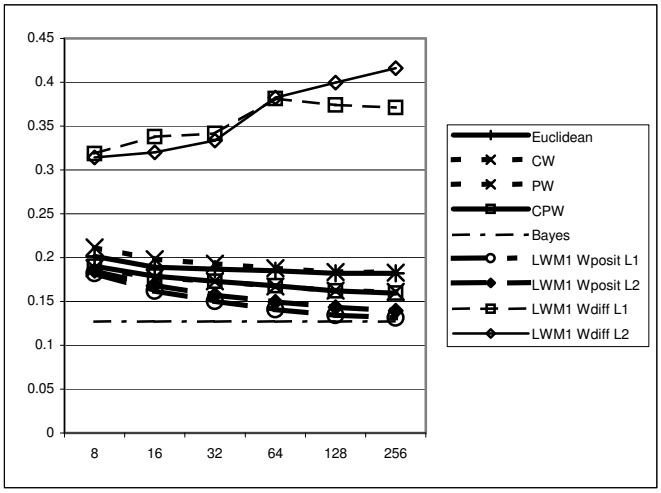

Fig. 1. Comparison of classification errors of synthetic data for different approaches. In legend Euclidean means 1-NN method with Euclidean metrics, CW, PW, and CPW are three variants of method by Paredes and Vidal; points are estimated from the reference [18]. "Bayes" means the Bayes limit. LWM1 means the method presented here, Wposit and Wdiff denote variant with weighted features (their positions) and weighted feature differences, respectively. L1 and L2 denote metrics used.

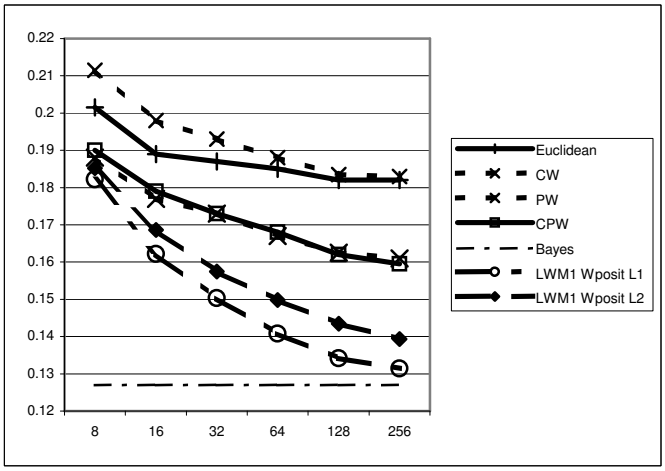

Fig. 2. Comparison of classification errors of synthetic data for different approaches, detail for successful methods. legend is the same as in Fig. 1.

Note that LWM1 method with L_1 metrics gives better results than with L_2 metrics here. We found this phenomenon rather often in distance-based methods especially for small training data sets.

B. Data from Machine Learning Repository

Data sets prepared just for run with a classifier were prepared by Paredes and Vidal and are available on the net [33]. We used all data sets of this corpus. Each task consists of 50 pairs of training and testing sets corresponding to 50-fold cross validation. For DNA data [34], Letter data (Letter recognition [13]), and Satimage (Statlog Landsat Satellite [13]) the single partition into training and testing set according to specification in [13] was used. We added also popular Iris data set [13] with ten-fold cross validation.

Results obtained by LWM1 approach presented here in comparison with data published in [18] are summarized in Table 9. Each row of the table corresponds to one task from [13]. For tasks where data are not available from [18] only results for 1-NN method with L2 metrics were amended.

In Table 9 it is seen that LWM1 as well as all variants of method [18] in all data sets outperforms 1-NN method with L_2 metrics. For some data sets LWM1 with weighted features outperforms all variants of [18], in others is well comparable. There is one exception, the Monkey1 data set for which LWM1 with weighted feature differences outperforms LWM1 with weighted features, i.e. positions of training set points. It can be also found that for these real-life data sets there is no special advantage of Manhattan (L1) metrics over Euclidean (L2) metrics.

Table 9.
Classification error rates for different datasets and different NN-based approaches by [18] and LWM1. Empty cells denote not available data.

Dataset	L2	CDM	CW	PW	CPW	posit. L1	posit. L2	diff. L1	diff. L2
australian	34.37	18.19	**17.37**	16.95	16.83	17.64	19.00	17.86	21.51
Balance	25.26	35.15	17.98	13.44	17.6	17.85	**16.17**	34.48	37.74
Cancer	4.75	8.76	3.69	3.32	3.53	17.70	**3.18**	26.46	26.49
diabetes	32.25	32.47	30.23	27.39	27.33	34.90	**26.49**	34.90	34.90
DNA	23.4	15	4.72	6.49	**4.21**	20.83	24.37	42.24	41.57
German	33.85	32.15	27.99	28.32	**27.29**	29.02	29.23	29.87	30.00
Glass	**27.23**	32.9	28.52	26.28	27.48	43.43	30.29	46.89	43.77
Heart	42.18	22.55	22.34	**18.94**	19.82	19.04	21.56	21.37	22.52
ionosphere	*19.03*					29.39	**17.58**	29.70	30.03
iris	*6.91*					**4.91**	6.93	25.82	11.82
led17	*20.5*					7.64	**2.67**	24.78	37.72
Letter	4.35	6.3	3.15	4.6	**4.2**	6.23	5.90	7.95	8.05
liver	37.7	39.32	40.22	**36.22**	36.95	40.96	42.00	40.70	40.43
monkey1	*2.01*					2.01	2.82	**1.45**	1.47
phoneme	*18.01*					14.72	**14.61**	29.27	29.27
Satimage	10.6	14.7	11.7	**8.8**	9.05	11.40	11.70	76.95	75.90
segmen	*11.81*					**5.18**	5.35	9.96	10.62
sonar	*31.4*					**21.11**	21.89	46.63	46.63
vehicle	35.52	32.11	29.38	**29.31**	28.09	30.48	31.01	36.83	34.96
vote	8.79	6.97	6.61	5.51	**5.26**	7.97	7.45	7.17	11.98
vowel	1.52	1.67	1.36	1.68	**1.24**	3.52	3.89	5.55	6.17
waveform21	*24.1*					**18.50**	18.63	25.56	25.19
waveform40	*31.66*					**20.50**	22.61	32.25	32.78
wine	24.14 1)	2.6	1.44	1.35	**1.24**	5.27	6.06	72.04	67.42

1) wine L2 24.14 is probably a misprint, it should be ~5.42.

C. Text Classification

Using text classification task one can test behavior of classification algorithm for very large dimensions (up to 10000). The WebKb (Web knowledge base [32]) contains web pages gathered from university computer science departments. We adopt approach known as "4 Universities WebKb". In this approach from the whole knowledge base 4199 web pages are selected. Selected pages are from four universities and are divided into four classes or categories, Student, Faculty, Course and Project.

Each web page is represented by a vector of word counts in that page. Thus different words correspond to individual features and value of a feature is 1 of a word is present in a page, 0 otherwise. Words were ordered in descending order according to their mutual information criterion [32] and for the same mutual information criterion according to total count in all 4199 pages. In tests a different size of vocabulary can be used. There are used 10, 50, 100, 500, 1000, 5000, and 10000 most frequent words. These numbers correspond to dimensionality of the task. As an average WebKb document contains about 80 different words out of

vocabulary of about 30000 words. The data set is rather sparse and some documents need not contain first 10, 50 or more first words in the vocabulary and therefore these documents have all features equal to zero.

These seven data sets we obtained directly from Roberto Paredes without split to training and testing set. We found that all sets have lot of samples equivalent. If one sample only would be in the training set, from principle of 1-NN method it follows exact recognition of equivalent samples of the testing set. We left one sample only of each group of equivalent samples. After this each data set was split to training and testing set of equal size. It can be found that for low dimensionality, the learning and testing sets have lesser number of samples than for larger dimensionality.

Comparison of results published in [18] and results obtained by approach presented here is shown in Fig. 3. Four variants were tested in experiments. The approach of weighted features i.e. weighted or even moved positions of points of the learning set and approach of weighted distances between the query point and points of the training set, in fact weighted differences of corresponding coordinates (features) of the query point and point of the training set.

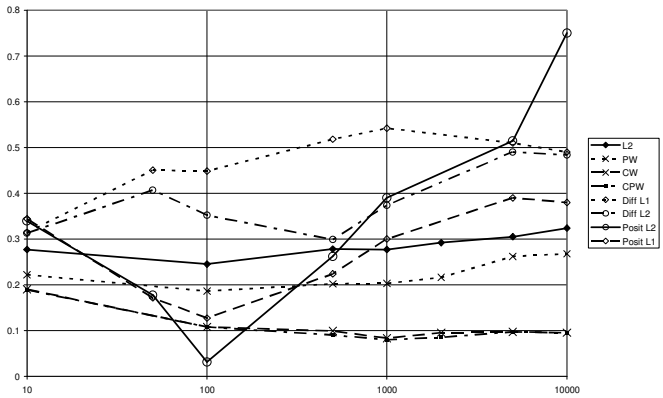

Fig. 3. WebKb nearest Neighbor classification results Euclidean (L2) method, four methods of Learning Weighting Metrics proposed in [18] and for method presented here with weighted positions (Posit L2 and Posit L1), weighted feature (coordinate) differences (Diff L2 and Diff L1) with the use of Euclidean and Manhattan (L1) metrics.

In Fig. 3 it is seen that the use of weighted feature (coordinate) differences with the use of Euclidean or Manhattan (L1) metrics gives worse results than the approach with weighted feature values, i.e. positions of points of the training set. Results are better when Manhattan (L1) metrics is used for weighted feature values.

Again it is verified that very large number of features may cause degradation of classification efficiency. Adding more features may add more noise than new information to the data set.

VIII. Conclusion

The Learning Weighted Metrics [18] is a nice idea how to improve efficiency of popular nearest-neighbor method. Its basic idea is to modify distance between the query point and point of the training set so that error of leave-one-out test is minimized. Thus error of 1-NN classification procedure is lowered. This approach probably can be also useful for other methods based on measuring and evaluating distances between a query point and points of the training set.

The target of this paper was to show that using nonsmooth analysis the sigmoid function approximation of step function could be eliminated. Thus a steepness factor (gain) β is eliminated which otherwise must be properly estimated in advance or even tuned to get good results. The steepness factor (gain) β does not exist when a nonsmooth analysis is used. The convergence parameter μ is eliminated here by the use of Runge's method of half step.

It was shown that the iterative learning process could be viewed as a string binding a gift box; it is straight on walls and abruptly changes direction on edges. During the learning process point moves in steps straight in the frame of wall and eventually changes the wall. The move, the iteration function is piece-wise linear, but has no derivative on edges. Theory based on results on nonsmooth contraction mapping [3] and generalized nonsmooth Newton method [21] shows that the nonsmooth error function has fixed point and iterative process converges quadratically to this point.

From the point of view of classification quality it was shown that learning weighted metrics approach with nonsmooth learning process outperforms the original LWM method [18] for some tasks from the machine Learning repository [13]. For Web Knowledge Base data the new approach does not seem to be the best approach but well comparable. From methodological point of view there is a question if data sets used in test were sufficiently similar even they were derived from the same set of web pages. Most interesting is finding that for synthetic data of three two dimensional normal distributions the approach presented here gets fast near Bayes limit with the size of data set. This convergence is much faster than any variant of the Learning Weighted Metrics method [18].

IX. Acnowledgements

This work was supported by the Ministry of Education of the Czech Republic under project Center of Applied Cybernetics No. 1M0567 (1M684077004), and project No. MSM6840770012 Transdisciplinary Research in the Field of Biomedical Engineering II. Authors are also grateful to Dr. R. Paredes for kind providing 4 Universities WebKb data.

X. Appendix- nonsmooth newton method

Suppose that $H: R^{nN} \to R^{nN}$ is almost everywhere differentiable. Let D_H be set of points x where H is differentiable. The generalized Jacobian of H at x is defined [21] by $\partial H(x) = \text{conv}\ \partial_B H(x)$ (conv means convex closure), where

$$\partial_B H(x) = \{ \lim_{\substack{x^j \to x \\ x^j \in D_H}} H'(x^j) \}$$

and where $H'(x^j)$ is a standard partial derivative wrt coordinate x^j; $\partial_B H(x)$ is a set of all partial derivatives wrt coordinates x^j on D_H.

The generalized Newton method [21] for solving equation $H(x) = 0$ is defined by formula $x_{k+1} = x_k - V_k^{-1} H(x_k)$, where $V_k \in \partial H(x_k)$.

Let H be directionally differentiable at x; the directioanal derivative in the direction d is denoted as $H'(x;d)$. H is said to be semi-smooth at x if[2]

$$Vd - H'(x;d) = o(\|d\|),\ d \to 0$$

and strongly semi-smooth at x if[3]

$$Vd - H'(x;d) = O(\|d\|^2),\ d \to 0,$$

where $V \in \partial H(x+d)$.

Theorem 1 [21]. Suppose that $H(x^*) = 0$ and that all $V \in \partial H(x^*)$ are nonsingular. Then the generalized Newton Method

$$x_{k+1} = x_k - V_k^{-1} H(x_k)$$

is Q-superlinearly convergent[4] in a neighborhood of x^* if H is semi-smooth at x^*, and quadratically convergent if H is strongly semi-smooth at x^*.

The condition "all $V \in \partial H(x^*)$ are nonsingular" can be changed to "all $V \in \partial_B H(x^*)$ are nonsingular", which sometimes may be less restrictive [21].

Note, lot of similar theorems can be found in literature, e.g. [28]. The method reminds also nonmonotone backtracking affine scaling inexact generalized Newton algorithm [30].

A broader variant of Newton method is an inexact Newton method; see e.g. [11], [7].

References

[1] T. Bailey, A.K. Jain: A note on distance-weighted k-nearest neighbor rules. IEEE Trans. Syst., Man, Cybern., vol. SMC=8, pp. 311-313, 1978.
[2] V. Castelli: Nearest Neighbor Classifiers. March 10, 2003. Available: http://www.ee.columbia.edu/~oblinger/e6880/L06-kNN.pdf. or https://skye.ee.columbia.edu/~vittorio/lecture8.pdf
[3] F.H. Clarke: Optimization and Nonsmooth Analysis. John Wiley & Sons, New York, 1983.
[4] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1):21–27, January 1967.
[5] C. Domeniconi, J. Peng, D. Gunopulos: Locally Adaptive nearest neighbor classification. IEEE Trans. Pattern Analysis and machine Intelligence, vol. 24, No. 9, pp. 1281-1285, Sept. 2002.
[6] S.A. Dudani: The distance-weighted k-nearest neighbor rule. IEEE Trans. Syst., Man, Cybern., vol. SMC-6, pp. 325-327, 1976.
[7] Y. Gao: Newton Methods for Quasidifferentiable Equations and Their Convergence1. Journal of Optimization Theory and Applications Vol. 131, No. 3, pp. 417–428, December 2006
[8] J. Goldberger, S Roweis, G. Hinton, R Salakhutdinov: Neighborhood component analysis. In: L.K. Saul, L. Bottou, editors, Advances in Neural Information Processing Systems vol. 17, pp. 513-520, MIT Press, Cambridge, MA, 2005
[9] G. Hall, J.M. Watt: Modern Numerical Methods for Ordinary Differential Equations. Clarenton Press, Oxford, 1976.
[10] T. Hastie, R. Tibshirani: Discriminant Adaptive Nearest Neighbor Classification. IEEE Trans. Pattern Analysis and machine Intelligence, vol.. 18, no. 6, pp. 607-616, June 1996
[11] J.M. Martínez, L. Qi: Inexact Newton Methods for Solving Nonsmooth Equations. Dept. of Applied Mathematics IMECC-UNICAMP, State University of Campias CP 6065, 13081 Campinas SP, Brazil., March 15, 1999, 29p.

[2] $f(x)=o(g(x))$ for $x \to a \ \Leftrightarrow \lim_{x \to a} f(x)/g(x) = 0$.
[3] $f(x)=O(g(x))$ for $x \to a \ \Leftrightarrow \limsup_{x \to a} |f(x)|/g(x) < \infty$.
[4] $\lim_{k \to \infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^q} = 0$, $q > 1$.

[12] J.E.S. MacLeod, A. Luk, D.M. Titterington: A re-examination of the distance-weighted k-nearest neighbor rule. IEEE Trans. Syst., Man, Cybern., vol. SMC-17, No. 4, pp. 689-696, Jyly/August 1987.
[13] C. J. Merz, P. M. Murphy, D. W. Aha. UCI Repository of Machine Learning Databases. Dept. of Information and Computer Science, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn/MLSummary.html (1997).
[14] M. Namba, H. Kamata, Y. Ishida,Neural Networks Learning with L1 Criteria and Its Efficiency in Linear Prediction of Speech Signals. Proc. ICSLP '96, vol. 2, Philadelphia, PA, pp. 1245-1248, 1996.
[15] H. Nyquist: Orthogional L1 norm Estimation. In: Statistical data analysis based on L1-norm and related methods, Y. Dodge, ed. Birkhauser Verlag, Basel, 2002.
[16] H.C. Ong, S.H. Quah: Error backpropagation using least absolute criterion. Int. J. Comput. Math. 82, No.3, 301-312 (2005). http://dx.doi.org/10.1080/0020716042000301743
[17] J.M. Ortega: Stability of difference equations and convergence of iterative ptocesses. Siam J. Numer. Anal. Vol. 10, No. 2, pp. 268-282, April 1973.
[18] R. Paredes, E. Vidal, Learning Weighted Metrics to Minimize Nearest-Neighbor Classification Error. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 7, July 2006, pp. 1100-1110.
[19] J. Peng, D.R. Heisterkamp, H. Dai: Adaptive quasiconformal kernel nearest neighbor classification. IEEE Trans. Pattern Analysis and machine Intelligence, vol. 26, no. 5, pp. 268-282, May 2004.
[20] P.K. Simpson: A Review of Artificial neural Systems. Pergamon Press, New York, 1990.
[21] L. Qi, D. Sun: Technical Report, School of Mathematics, University of New South Wales, Sydney, Australia, June 1998.
[22] D. Ridder, O. Kouropteva, O. Okun, M. Pietikainen, R.P.W. Duin: Supervised locally linear embedding. Proc of Joint Conf Artificial neural networks and neural Information processing, 2003.
[23] D. Ridder, M. Loog, M.J.T. Reinders: Local Fischer Embedding. Proc. 17th Int'l Conf. Pattern Recognition, vol. 2, pp. 295-298, 2004.
[24] S.T. Roweis, L.K. Saul: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, New Series, Vol. 290, No. 5500, pp. 2323-2326, (Dec. 22, 2000).
[25] D.W. Scott: Multivariate density estimtion: theory, practice, and visualization. John Wiley and Sons, Inc., New York, 1992, 317 pp.
[26] X. Tian, Y. Vardi, C-H. Zhang: L1-depth, Depth relative to a model, and robust regression. In: Statistical data analysis based on L1-norm and related methods, Y. Dodge, ed. Birkhauser Verlag, Basel, 2002.
[27] K.Q. Weinberger, J. Blitzer L. K. Saul: Distance Metric Learning for Large Margin Nearest Neighbor Classification. Advances in Neural Information Processing Systems, Department of Computer and Information Science, University of Pennsylvania, paper No 0265, 2005, available http://books.nips.cc/papers/files/nips18/NIPS2005_0265.pdf
[28] H. Xu, X. Chang: Approximate Newton methods for nonsmooth equations. Journal of Optimization Theory and Applications, Vol. 93 (1997), pp. 373--394, 1997.
[29] H. Zhang, A.C. Berg, M. Maire, J. Malik: SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 2126 – 2136, 2006.
[30] D. Zhu: Affine scaling inexact generalized Newton algorithm with interior backtracking technique for solving bound-constrained semi-smooth equations. Journal of Computational and Applied Mathematics, Vol. 187, no. 2 , pp. 227-252, 15 March 2006.
[31] C. Chopra, R. Hadsell, and Y. Lecun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. CVPR, pp. 539-546, 2005
[32] M. Craven et al.: Learning to Extract Symbolic Knowledge from the World Wide Web. Proc 15th National Conference on Artificial Intelligence, pp. 509-516, 1998.
[33] http://algoval.essex.ac.uk/data/vector/UCI/
[34] http://www.dsic.upv.es/~rparedes/ research/CPW/index.html