Neural minimal distance methods

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

Abstract

Minimal distance methods are simple and in some circumstances highly accurate. In this paper relations between neural and minimal distance methods are investigated. Neural realization facilitates new versions of minimal distance methods. Parametrization of distance functions, distance-based weighting of neighbors, active selection of reference vectors from the training set and relations to the case-based reasoning are discussed.

I. Introduction

CLASSIFICATION is one of the most important applications of neural systems. Approximation or control problems may be presented as classification with infinite number of classes. Accuracy of 24 neural-based, pattern recognition and statistical classification systems has been recently compared on 11 large datasets by Rhower and Morciniec [1]. There is no consistent trend in the results of this large-scale study. Differences between error rates of many methods are within a few percent, which is statistically not significant [2]. For each classifier one may find a real-world dataset for which results will be excellent and another one for which results will be quite bad. Frequently simple methods, such as minimal distance methods or n-tuple methods, are among the best. Therefore it is important to investigate neural realizations of such simple methods.

One of the simplest classification algorithms used in pattern recognition is based on the k-nearest neighbor (k-NN) rule [3]. Each training data vector is labeled by the class it belongs to and is treated as a reference vector. During classification k nearest reference vectors to the new vector $\mathbf{X}$ are found and the class of $\mathbf{X}$ is determined by a majority rule. One can assign confidence measures to the classification decision, assuming that the probability of assigning the vector $\mathbf{X}$ to class C_i is $p(C_i|\mathbf{X}) = N_i/k$. To avoid ties k should be odd. In the simplest case $k = 1$ and the nearest neighbor determines the class of an unknown vector.

The asymptotic error rate of the k-NN classifier for large training sets is small and in the limit of large k becomes equal to the optimal Bayesian values [3]. Because the method is so simple it is frequently used as a standard reference for other classificators. One problem is the computational complexity of the actual classification, demanding for n reference vectors calculation of $\sim n^2$ distances and finding k smallest distances. Although Laaksonen and Oja [4] claim that "For realistic pattern space dimensions, it is hard to find any variation of the rule that would be significantly lighter than the brute force method" various hierarchical schemes of partitioning the data space or hierarchical clusterization is quite effective. The search for the nearest neighbors is easily paralelizable and training time (selection of optimal k) is relatively short. Minimal distance methods are especially suitable for complex applications, where large training datasets are available. They are also used in the case-based expert systems as an alternative to the rule-based systems (cf. [5]).

Only one neural model proposed so far is explicitly based on the nearest neighbor rule: the Hamming network [6], [7] computes the Hamming distances for the binary patterns and finds

the maximum overlap (minimum distance) with the prototype vectors, realizing the 1-NN rule. As a step towards integration of the neural and pattern recognition methods relations between minimal distance methods and neural-based classifiers are systematically investigated in this paper. Networks capable of realization of the classical k-NN rule and several new variants of the minimal distance methods are proposed. Various parametrizations of the minimal distance methods are discussed.

II. Neural networks and minimal distance methods

The problem is stated as follows: given a set of class-labeled training vectors $\{\mathbf{X}^k, \mathbf{C}^k\}$ and a vector $\mathbf{X}$ of unknown class use information provided in the distance $d(\mathbf{X}, \mathbf{X}_k)$ to estimate probability of classification $p(C_i|\mathbf{X}; M)$, where M is the model description, including the number of reference vectors in the neighborhood of $\mathbf{X}$, the size of the neighborhood considered, function used to compute distances, procedure to select reference vectors from the training set and the weighting scheme estimating contribution of reference vectors $G(d(\mathbf{X}, \mathbf{X}_k))$ to the classification probability. Various selections of parameters and procedures in the context of network computations lead to different versions of neural minimal distance methods.

What neural networks have to do with the minimal distance methods? Threshold neurons compute distances in a natural way. If the input signals $\{X_i = \pm 1\}$ and the weights $\{W_i = \pm 1\}$ are binary, neuron with N inputs and the threshold θ realizes the following function:

$$\Theta(\sum_{i}^{N} W_i X_i - \theta) = \begin{cases} 0 & \text{if } ||\mathbf{W} - \mathbf{X}|| > \theta \\ 1 & \text{if } ||\mathbf{W} - \mathbf{X}|| \leq \theta \end{cases} \tag{1}$$

where $||\cdot||$ norm is defined by the Hamming distance. One can interpret the weights of neurons in the first hidden layer as addresses of the reference vectors in the input space and the activity of threshold neuron as activation by inputs falling into a hard sphere of radius θ centered at $\mathbf{W}$. Changing binary into real values and threshold into sigmoidal neurons for inputs normalized to $||\mathbf{X}|| = ||\mathbf{W}|| = 1$ leads to a soft activation of neuron by input vector close to $\mathbf{W}$ on a unit sphere. In general the output function of a neuron:

$$\sigma(\mathbf{W} \cdot \mathbf{X}) = \sigma(\frac{1}{2}(||\mathbf{W}||^2 + ||\mathbf{X}||^2 - ||\mathbf{W} - \mathbf{X}||^2) = \sigma(I_{max} - d(\mathbf{W}, \mathbf{X})) \tag{2}$$

For normalized input vectors sigmoidal functions (or any other monotonically growing transfer functions) simply evaluate the influence of reference vectors $\mathbf{W}$, depending on their distance $d(\mathbf{W}, \mathbf{X})$, on classification probability $p(C_i|\mathbf{X}; \{\mathbf{W}, \theta\})$. Interpretation of neural classifiers as special minimal distance classifiers is thus feasible and worth of investigation. Several variants of the neural minimal distance methods are presented below.

k-NN networks.

In the simplest version of the k-NN method for fixed training dataset the number of neighbors k is the only parameter that should be optimized. For $k = 1$ there is no error on the training set, but already for $k = 3$ the training vector near the class border may have two nearest vectors from another class. Therefore the error on the training set, equal to zero for $k = 1$, grows for $k = 3$ and should decrease for larger values of k. To optimize k a validation

or a test dataset should be used, for example the leave-one-out test using the training set. Details of this procedure are rarely given and it is not always clear whether the k-NN classifier has actually not been optimized on the test data.

Neural realization of 1-NN rule is achieved by the Hamming network [6]. Except for the input and the output layers, there are two hidden layers, the first computing Hamming distances for binary patterns, the second (called Maxnet) working as the winner-takes-all recurrent layer, selecting the node with the strongest activation. The convergence of the Maxnet layer, despite the improvements in original algorithm [7], is unnecessarily slow. The Hamming network is significantly simplified if more complex output nodes are allowed. For normalized vectors the output unit should determine from which hidden node the maximum input is received and transfer the class label of this node to the output. In software simulations finding the node with maximum response is quite trivial. An alternative approach that may be implemented also in hardware is to build a network with hidden nodes realizing the **hard sphere** transfer functions, i.e. $\Theta(r - d(\mathbf{X}, \mathbf{D}))$, where Θ is the Heaviside threshold function, r is the radius of the sphere and $d(\mathbf{X}, \mathbf{D})$ is the distance between the vector $\mathbf{X}$ and the reference (training) vector $\mathbf{D}$. The output units for each class sum the incoming signals from all active hidden nodes belonging to that class. The number N_i of such units in the radius r from the new vector $\mathbf{X}$ allows to compute the probability of classification $p(C_i|\mathbf{X}) = N_i / \sum_j N_j$. From the geometrical point of view in the input space a hard sphere is assigned to each reference vector, labeled by the name of its class, and the output unit counts how many spheres of a given class reach the point $\mathbf{X}$.

Neural realization of k-NN method finds r for which the sum of all network outputs $\sum_j N_j = k$. Formally this can be done by introducing recurrent connections and stabilizing dynamics when the "superoutput" node achieves fixed value. Since most neural simulators are realized in software it is quite easy to implement this step by repeating classification a few times with different r values until the vector $\mathbf{X}$ finds itself in the range of exactly k neighbors.

NN-r algorithm.

Instead of enforcing exactly k neighbors the σ radius may be used as an adaptive parameter. The number of classification errors or the probability of classification $p(C_i|\mathbf{X}; r) = N_i / \sum_j N_j$ is optimized on the validation set. Again the hard sphere transfer functions are used in the network realization of this algorithm. k-NN for odd k always classifies the data, while NN-r may reject some vectors $\mathbf{X}$ if no reference vectors fall into the r radius of $\mathbf{X}$ or if equal probability of classification for several classes is obtained.

Introduction of variable radiuses r in different parts of the input space improves the method further. Development along this line leads to the Restricted Coulomb Energy (RCE) classifier introduced by Reilly, Cooper and Elbaum [8]. Network realization of this classifier uses hard sphere distance function. If no neighbors are found around training vector X new spheres (reference vectors) are added with largest radius such that the sphere does not overlap with spheres of other classes. If the new training vector falls into the range of a sphere of wrong class the radius of this sphere is shrinked to leave the vector outside of the sphere. Positions of the spheres are not optimized in the RCE algorithm (this would lead in the direction of LVQ algorithms), but voting methods for the committees of classifiers were used with success [9].

Soft weighting k-NN and NN-r algorithms.

Changing hard sphere transfer functions into softer function allows to include weights in-

fluencing classification decisions. Close reference vectors should influence probabilities of classification more than farther laying neighbors. The simplest suitable transfer function is **the conical radial function**: zero outside the radius r and $1 - d(\mathbf{X}, \mathbf{D})/r$ inside this radius. Classification probability is calculated by the output node using the formula:

$$p(C_i|\mathbf{X}; r) = \frac{\sum_{n \in C_i} G(\mathbf{X}; \mathbf{D}_n, r)}{\sum_n G(\mathbf{X}; \mathbf{D}_n, r)}; \qquad G(\mathbf{X}; \mathbf{D}, r) = \max\left(0, 1 - \frac{d(\mathbf{X}, \mathbf{D})}{r}\right) \tag{3}$$

Here $G(d) = \max(0, 1 - d), d \geq 0$ is the weight estimating contribution of reference vector at the distance d. Reference vectors outside of the r radius have no influence on the classification probability while those that are inside this radius have influence that is directly proportional to how close they are to the vector given for classification. In the soft NN-r algorithm the r parameter is optimized. Radial Basis Function (RBF) networks using Gaussian or inverse multiquadratic transfer functions are a particular example of soft weighting minimal distance algorithm. Other possibilities include optimization of shape of $G(||\mathbf{X} - \mathbf{D}_n||, r)$ transfer functions using additional parameters, for example by using a combination of two sigmoidal functions: $\sigma(||\mathbf{X} - \mathbf{D}_n|| - r) - \sigma(||\mathbf{X} - \mathbf{D}_n|| + r)$.

The cost function is either a classification error (as for the hard-distance case) or – since continuos output values are provided – minimization of risk for overall classification:

$$E(R, M) = \sum_{\mathbf{X}} \sum_i R(C_i, C(\mathbf{X})) \left[p(C_i|\mathbf{X}; M) - \delta(C_i, C(\mathbf{X}))\right]^2 \tag{4}$$

where $C(\mathbf{X})$ is the true class of vector $\mathbf{X}$, the elements of the cost matrix $R(C_i, C_j)$ are proportional to the risk of assigning the C_i class when the true class is C_j, and M specifies parameters of the classifier.

Parametrization of distance measures

Calculation of distance is most often based on Euclidean metric for continuos inputs and Hamming metric for binary inputs. Additional parameters that may be optimized are either global (for all data) or local (for each reference vector). Minkowski's metric involves one global parameter, exponent α. Scaling factors are useful global parameters – for Minkowski's distance:

$$d(A, B; g)^2 = \sum_i^N g_i (A_i - B_i)^\alpha \tag{5}$$

In particular if scaling factors $g_i \geq 0$ become small for some input features i these features may be eliminated. To facilitate elimination of features that are not useful for classification the cost function may include additional penalty term, such as the sum of all g_i^2.

In the simplest RBF version only one parameter – dispersion – is optimized. Independent optimization of all N components of dispersion vector is equivalent to optimization of scales g_i. General linear transformation applied to input vectors is equivalent to introduction of a metric tensor $G_{ij} = G_{ji}$, providing $N(N + 1)/2$ adaptive parameters:

$$d(A, B; \mathbf{G})^2 = \sum_{i,j}^N G_{ij} (A_i - B_i)(A_j - B_j) \tag{6}$$

Calculation of distances may also be parametrized in a different way around each reference vector. Local coordinate systems with their origin placed at the reference vectors may provide either local scaling factors or local metric tensors.

In memory-based reasoning the Modified Value Difference Metric (MVDM) has gained popularity [5]. The distance between two N-dimensional vectors A, B with discrete (for example symbolic) elements, in a K class problem, is computed using conditional probabilities:

$$d(A,B) = \sum_{j}^{N} \sum_{i}^{K} (p(C_i|A_j) - p(C_i|B_j)) \tag{7}$$

where $p(C_i|A_j)$ is estimated by calculating the number $N_i(A_j)$ of times feature A_j occurred in vectors belonging to class C_i and dividing it by the number of times feature A_j occurred for any class. We can also define a "value difference" for each feature j as $d_v(A_j, B_j) = \sum_i^K (p(C_i|A_j) - p(C_i|B_j))$ and compute $d(A,B)$ as a sum of value differences over all features. Metric is defined here via a data-dependent matrix with the number of rows equal to the number of classes and the number of columns equal to the number of all attributes. Generalization for continuos values requires a set of probability density functions $p_{ij}(x)$, with $i = 1..K, j = 1..N$.

Active selection of reference vectors.

Suppose that a large number of training vectors is available, divided into the reference and the remaining set of vectors. Clusterization techniques are used to select a relatively small number of initial reference vectors close to the cluster centers. Classification accuracy is checked on the remaining set (using k-NN or NN-r rule) and each time an error is made the vector is moved from the remaining to the reference set. In this way small number of reference vectors is selected. Variants of this approach may use validation set to determine best candidates for the reference set.

An alternative approach that does not require initial clusterization starts from the whole training set and removes those vectors that have all k nearest vectors from the same class. These vectors are far from cluster borders and all new vectors in their neighborhood will be anyway unambiguously classified. This approach leads to a "hollow" cluster representation. Here one may start with a large k' to remove vectors near the centers of clusters first and reduce it to k in a few steps.

Parametrization of reference vectors

Active selection of reference vectors may eliminate many training vectors from the reference set. Further optimization of their positions should decrease the training error. The reference vectors $\mathbf{D}_n$ in the neighborhood of training vector $\mathbf{X}$ are moved by:

$$\mathbf{D}_n^{new} = \mathbf{D}_n^{old} + \eta(2\delta(C(\mathbf{X}), C(\mathbf{D}_n^{old})) - 1)(\mathbf{X} - \mathbf{D}_n^{old}) \tag{8}$$

Here η is the learning rate that may decrease during training and the sign is $+$ if $\mathbf{X}$ and $\mathbf{D}_n^{old}$ belong to the same class or $-$ otherwise. Various rules for moving centers $\mathbf{D}_n$ are used:

moving only the nearest neighbor, moving all k neighbors by the same amount, using distance-dependent η etc. One can also optimize a subset of vectors, for example only those that are close to the center of clusters.

III. Summary and discussion

Models belonging to the neural minimal distance family estimate probability of classification $p(C_i|\mathbf{X}; k, G(d(\cdot;r)), \{\mathbf{D}_n\})$, where k is the number of neighbors taken into account, $G(d(\cdot,r))$ is the distance-dependent weighting function, $d(\cdot,r)$ is the distance function parametrized by the radius r and $\{\mathbf{D}_n\}$ is the reference set of vectors. These model differ by:
1. Treatment of the number of the nearest neighbors: integer k optimized on the validation set in classical k-NN, with hard sphere distance function $d(\cdot;r)$ and the set of reference vectors $\{\mathbf{D}_n\}$ equal to the training set; soft k if the sum of all activations of network nodes is restricted to k and conical or other soft weighting functions are used; variable k if the hard sphere radius r or other distance and weighting parameters are optimized without enforcing the fixed value of k.
2. Estimation of the influence of neighbors $G(d(\cdot,r))$: each neighbor counted with the same weight, as in the original k-NN, or counted using a distance-dependent function.
3. Parametrization of distances: hard sphere functions, conical functions, Gaussian and other localized functions, probabilistic distance measures.
4. Treatment of the reference set $\{\mathbf{D}_n\}$: training data taken as the reference set without changes; active selection of reference vectors (after initial clusterization) from the training set; optimization of reference vectors using learning vector quantization techniques.
5. Technical issues: speeding up calculations of distances using hierarchical clusterization, pre-processing of data (details of normalization and standardization procedures).

Both MLP and RBF networks are particular examples of neural minimal distance methods. In addition many possibilities to create fuzzy k-NN models exist. Performance of various methods described here depends unfortunately on the nature of the data given for classification and remains a subject of further empirical study (Duch and Grudziński, in preparation).

Acknowledgments: Support by the Polish Committee for Scientific Research, grant 8T11F 00308, is gratefully acknowledged.

References

[1] R. Rohwer and M. Morciniec, A Theoretical and Experimental Account of n-tuple Classifier Performance, *Neural Computation* 8 (1996) 657–670
[2] B.D.Ripley, Pattern Recognition and Neural Networks (Cambridge University Press 1996)
[3] P.R. Krishnaiah, L.N. Kanal, eds, Handbook of statistics 2: classification, pattern recognition and reduction of dimensionality (North Holland, Amsterdam 1982)
[4] J. Laaksonen, E. Oja, Classification with Learning k-Nearest Neighbors. In: *Proc. of ICNN'96*, Washington, D.C., June 1996, pp. 1480-1483.
[5] D.L. Waltz, Memory-based reasoning, in: M. A. Arbib, ed, *The Handbook of Brain Theory and Neural Networks* (MIT Press 1995), pp. 568–570
[6] R.P. Lippmann, An introduction to computing with neural nets, *IEEE Magazine on Acoustics, Signal and Speech Processing* 4 (1987) 4–22
[7] P. Floreen, The convergence of Hamming memory networks, *Trans. Neural Networks* 2 (1991) 449–457
[8] D.L. Reilly, L.N. Cooper, C. Elbaum, A neural model for category learning, *Biological Cybernetics* 45 (1982) 35–41
[9] P.D. Wasserman, Advanced methods in neural networks (van Nostrand Reinhold 1993)