EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

European Journal of Operational Research 92 (1996) 230–243

Parametric distance functions vs. nonparametric neural networks for estimating road travel distances

Ethem Alpaydın [a,1], İ. Kuban Altınel [b,*], Necati Aras [b,1]

[a] Department of Computer Engineering, Boğaziçi University, TR-80815 Istanbul, Turkey

[b] Department of Industrial Engineering, Boğaziçi University, TR-80815 Istanbul, Turkey

Received July 1994; revised August 1995

Abstract

Measuring and storing actual road travel distances between the points of a region is often not feasible and it is a common practice to estimate them. The usual approach is to use distance estimators which are parameterized functions of the coordinates of the points. We propose to use nonparametric approaches using neural networks for estimating actual distances. We consider multi-layer perceptrons trained with the back-propagation rule and regression neural networks implementing nonparametric regression using Gaussian kernels. We also consider training multiple estimators and combining them using voting and stacking. On a real-world study using cities drawn from Turkey, we found out that these nonparametric approaches are more accurate than the parametric distance functions. Estimating actual distances has many applications in location and distribution theory.

Keywords: Artificial intelligence; Location; Neural networks; Regression; Road transportation

1. Introduction

The actual distance between any two points on the earth surface is the length of the shortest road connecting them travellable with the given means of transportation. Since it is often not feasible to measure the *actual road travel distances* (or *actual distances* in brief) for all pairs of points, it is a common practice to use distance estimators. The question is to choose a good estimator so that accurate distance approximations are obtained.

A good estimation of actual distances is critical in many applications. Almost all of the location problems, distribution problems such as the transportation problem, its generalization the transshipment problem, the travelling salesman problem, and the vehicle routing problem assume the knowledge of actual distances in their formulations. For example, in their recent simulation study to determine the number of fire-stations in Istanbul, Erkut and Polat multiply the Euclidean distance by an inflation factor, which they call the *road coefficient*, in order to estimate the actual distance between the fire-station and fire area [14].

We can define the problem of distance estimation formally as follows: Let us say that $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T$ where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are two points on the Cartesian plane with coordinates $\boldsymbol{x}_1 = (x_{11}, x_{12})^T$ and $\boldsymbol{x}_2 = (x_{21}, x_{22})^T$. The aim is to build an estimator $d(\boldsymbol{x} \mid \boldsymbol{\theta})$

* Corresponding author. Email: altinel@boun.edu.tr.

[1] Email: {alpaydin, arasn}@boun.edu.tr.

of the actual distance between x_1 and x_2. S is a sample of n pairs (x^i, r^i) where r^i is the actual distance between x_1^i and x_2^i with $i = 1 \dots n$. $\boldsymbol{\theta}$ is a vector of parameters estimated using S minimizing the following error measure:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left[E[\delta(d(x \mid \boldsymbol{\theta}, S), r)] \right] = \arg\min_{\boldsymbol{\theta}} \left[\frac{1}{n} \sum_{i=1}^{n} \delta\left(d(x^i \mid \boldsymbol{\theta}), r^i\right) \right], \quad (1)$$

$\delta(\cdot)$ is the difference measure. One possibility, originally proposed by Love and Morris [20], is the absolute value of the deviation:

$$\delta\left(d(x^i \mid \boldsymbol{\theta}), r^i\right) = \left| d(x^i \mid \boldsymbol{\theta}) - r^i \right|. \quad (2)$$

According to this criterion, a distance function must estimate greater actual distances relatively more accurately than shorter distances. This is a drawback if we are more interested in proportional deviations than absolute deviations. Another error measure, also proposed by Love and Morris [20], is normalized:

$$\delta\left(d(x^i \mid \boldsymbol{\theta}), r^i\right) = \left[\frac{d(x^i \mid \boldsymbol{\theta}) - r^i}{\sqrt{r^i}} \right]^2. \quad (3)$$

Although both criteria are very insightful on their own, the second one is superior not only because it gives importance to proportional errors but also because of the following three reasons. First, most of the experimental results in the literature use the second criterion, e.g., [4,5,11,20,24,35]. It has important statistical properties which leads to statistical tests for comparing the accuracy of distance functions under certain normality and independence assumptions and thus the results obtained can have further use. Finally, it is a continuous and differentiable function of the parameter vector and this enables the use of gradient descent methods for minimization which is important in neural network learning as we will see in Section 3.

The standard approach for distance estimation uses estimators that are parameterized functions of certain easy to obtain information, namely the coordinates of the points. This approach has been widely used ever since the first work by Love and Morris [20] because it provides simple analytical closed form expressions of the coordinates once the value of the parameters are determined. As any parametric method, it works with small samples but the accuracy may not be high if the assumed form of the function is not appropriate.

We view the problem of estimating distances in the context of function approximation or nonlinear regression and apply neural network based estimators for this task of estimating $d(x \mid \boldsymbol{\theta})$. These methods, being nonparametric, have the advantage that they do not assume any a priori model and are trained directly from a training sample. They of course necessitate larger training samples as we no longer have the comfort of a parametric model with just a few parameters.

One approach we use is the *multi-layer perceptron* trained with the back-propagation rule. The second approach is based on nonparametric regression using kernel density estimation; in the neural network literature, this is named a *regression neural network*. A recent idea in machine learning is to combine multiple estimators for improved performance. We test two methods for this. One trains multiple estimators and takes a *vote*; when vote weights are equal, this is simple averaging. The second *stacking* approach also trains a combiner network. We provide a comparison of all explained approaches based on a real-world sample collected by pairing a subset of the cities of Turkey.

This paper is organized into seven sections. The next section is a brief literature survey on the use of suitable distance functions to estimate actual distances. In Sections 3 and 4, we propose neural-network-based methods, i.e., a multi-layer perceptron and a regression neural network, for distance estimation. Section 5 explains the idea of combining multiple estimators. In Section 6, we give results obtained for Turkey using the previously explained approaches. Finally in the last section, we compare approaches using the criteria of estimation accuracy, memory requirement, and general applicability in optimization problems.

2. Distance functions

A generally used method for estimating actual road distances between any pair of points is to make approximations by means of a distance function,

Table 1
Distance functions used and their associated parameters

Distance function	Parameters ($\boldsymbol{\theta}$)
$d_1(x) = k(\lvert x_{11} - x_{21}\rvert + \lvert x_{12} - x_{22}\rvert)$	k
$d_2(x) = k(\lvert x_{11} - x_{21}\rvert^2 + \lvert x_{12} - x_{22}\rvert^2)^{1/2}$	k
$d_3(x) = k(\lvert x_{11} - x_{21}\rvert^p + \lvert x_{12} - x_{22}\rvert^p)^{1/p}$	k, p
$d_4(x) = k(\lvert x_{11} - x_{21}\rvert^p + \lvert x_{12} - x_{22}\rvert^p)^{1/s}$	k, p, s
$d_5(x) = k_1(\lvert x_{11} - x_{21}\rvert + \lvert x_{12} - x_{22}\rvert)$ $+ k_2(\lvert x_{11} - x_{21}\rvert^2 + \lvert x_{12} - x_{22}\rvert^2)^{1/2}$	k_1, k_2

which is a parameterized function of the planar coordinates of the two points. These functions can be classified in three major groups with respect to the type of coordinates they use. The members of the first group use spherical coordinates for the purpose of introducing the spherical effect of the earth surface into the distance estimation [20,21]. Although this idea provides certain additional accuracy, the contribution has been experimentally reported to be minor by Love and Morris [20]. The second group consists of functions which use polar coordinates [27,29]. The motivation is based on the observation that the roads in old cities are not usually planned according to a rectangular grid structure and therefore distances can be approximated better by a ring-radial measure. This approach seems to be very accurate especially for a spider's web-like road network structure. The third group contains some simple functions of the cartesian coordinates. These are mostly norms or norm-based functions. We list five of them which we adopt in this study in Table 1. The first four are the most important because of their wide usage in location and distribution problems [15,23].

The parameters, which should be nonnegative, k, p, s, k_1, and k_2 constitute $\boldsymbol{\theta}$ and are estimated over a sample to provide good approximations and as such, encode geographical characteristics of the region where they are used. There is a large literature on the determination of these parameters and the comparison of the distance metrics, sometimes with conflicting results [5–7,9,20–22].

For all practical purposes, the function chosen to estimate actual road distances should be as accurate as possible. In their early study, Love and Morris [20,21] compute the parameters k, p, and s of $d_1(\boldsymbol{x})$, $d_2(\boldsymbol{x})$, $d_3(\boldsymbol{x})$, and $d_4(\boldsymbol{x})$ for the United States and compare them with respect to the accuracy they provide. The important conclusion of this study is the superiority of $d_4(\boldsymbol{x})$ over the other three. The second best is $d_3(\boldsymbol{x})$.

At the end of their study on the road network of the former Federal Republic of Germany (FRG), Berens and Körling [6] and Berens [5] conclude that the accuracy provided by the weighted Euclidean norm $d_2(\boldsymbol{x})$ is sufficient and the use of $d_3(\boldsymbol{x})$ is not worth the extra computational effort necessary for calculations. However in a further study over the largest 25 cities of FRG, Love and Morris [22] report conflicting results which demonstrate that the accuracy of the weighted L_p norm, $d_3(\boldsymbol{x})$, is remarkably higher than the accuracy provided by $d_2(\boldsymbol{x})$. Although it supports the early findings of Berens and Körling [6] for FRG, the study by Berens [5] includes mixed results when it is enlarged to cover 11 other countries; the relative improvement introduced by $d_3(\boldsymbol{x})$ over $d_2(\boldsymbol{x})$ ranges within 0.00% and 11.27%. Finally, Berens and Körling [7], in their last comment, state that the empirical distance functions should be tailored for the regions in which they are to be used if the accuracy is the main interest and that there is not one general distance function which provides the same accuracy all over the world.

There are also distance measures which do not fit completely in any of the above mentioned three groups. They can be included in the last one but they are not always simple functions of the coordinates and require additional information such as a rotating angle for the coordinate axes [11,21] or vectors for possible directions on a typical road [35,36]. All of them are based on the idea that a travel has two major components; rectilinear and Euclidean, and the actual distance between any pair of points can be modelled as a positive linear combination of them. Ward and Wendell [35] initiate this hybrid idea by suggesting the weighted one-infinity norm and observe that the accuracy of this function is relatively close to the accuracy of the weighted L_p norm, $d_3(\boldsymbol{x})$, on the data set of Love and Morris [20]. In their later work, Ward and Wendell generalize the one-infinity norm to obtain the family of block norms in which the accuracy of the approximation depends on possible travel directions [36]. They report that the approximations obtained by the weighted L_p

norm are more accurate than those obtained by a two-parameter block norm, which is actually the weighted one-infinity norm, and the accuracy of the weighted L_p norm is slightly worse than the one-of-eight-parameter block norm's. Similar conclusions have been obtained also by Love and Walker [24] in their detailed empirical study on block and round norms. Block norms play an important role in location models because they lead to linear programming problems for certain objective functions, such as the minimax distance function; but the size of the linear program can become very large.

Another hybrid distance function, which is presented as $d_5(x)$ in Table 1, is due to Brimberg and Love [10]. It is called the weighted one-two norm since the rectilinear and Euclidean elements of the travel are presented respectively by the weighted L_1 and L_2 norms. The authors suggest its use to approximate $d_3(x)$ in estimating distances. The weighted one-two norm also provides good approximations for the probabilistic L_p norm [12]. Besides, its parameters can be calculated easily by simple linear regression [9], and it can perform very well when local information is also introduced through the rotation of coordinate axes.

Due to the statistical nature of distance functions, the unknown distance between the points may be overestimated or underestimated. Then, confidence intervals for unknown distances become important since they can be used to measure the accuracy of the estimated distance. In the recent work of Love et al. [25], this issue has been addressed. They have developed a procedure for calculating confidence intervals for unknown distances. Their procedure utilizes information provided by the sample Pearson coefficients.

3. Multi-layer perceptrons

A multi-layer perceptron is a feedforward type of neural network [18]. Each unit in a neural network performs a weighted sum of its inputs. In a feedforward network, input is fed to the input layer which propagates to the output layer through possibly a layer of nonlinear hidden units. Weights between consecutive layers encode interdependencies or constraints between the values that the units take simultaneously. It has been shown that such a network with one hidden layer is a universal approximator, i.e., can approximate any function with the desired accuracy [16,19]. A feedforward neural network with one hidden layer of H hidden units operates as follows:

$$F(\boldsymbol{u} \mid \boldsymbol{T}, \boldsymbol{W}) = \sum_{h=1}^{H} T_h g_h(\boldsymbol{W}_h, \boldsymbol{u}) + T_0, \tag{4}$$

$\boldsymbol{u}$ is the multi-dimensional input vector. T_h is the weight from hidden unit h to the output unit. T_0 is the "bias" weight. The output thus is simply a weighted sum of the hidden basis functions, $g_h(\boldsymbol{W}_h, \boldsymbol{u})$. Hidden units compute a weighted sum of the k-dimensional input $\boldsymbol{u}$ with their weights $\boldsymbol{W}_h$ and then filter this value through a nonlinear sigmoid activation function which is a smoothed version of thresholding:

$$\begin{aligned} H_h &= g_h(\boldsymbol{W}_h, \boldsymbol{u}) \\ &= 1 \Big/ \left[1 + \exp\left(- \sum_{i=1}^{k} u_i W_{ih} - W_{0h} \right) \right]. \end{aligned} \tag{5}$$

The most popular training method for multi-layer perceptrons is the back-propagation learning algorithm [31] which implements gradient descent over an error function. Given a differentiable error function E, we start with any set of weights and repeatedly change each parameter θ_p by an amount proportional to $\partial E / \partial \theta_p$:

$$\Delta \theta_p = -\eta \frac{\partial E}{\partial \theta_p}, \tag{6}$$

where η is the stepsize in descent.

In our application of distance estimation, we use the error function of Eqs. (1) and (3). The input to the neural network is a vector function of the coordinates of the two points: $\boldsymbol{u}^i = \phi(\boldsymbol{x}^i)$. Similarly the output of the neural network by a known transformation gives us the desired distance: $y^i = d(\boldsymbol{x}^i \mid \boldsymbol{\theta}) = \chi(F(\boldsymbol{u}^i \mid \boldsymbol{T}, \boldsymbol{W}))$. $\boldsymbol{\theta}$, the parameter vector of our estimator corresponds to the weights of the neural network, i.e., $\boldsymbol{W}$ and $\boldsymbol{T}$. The problem of choosing good *coding* or *representation* functions, $\phi(\cdot)$ and $\chi(\cdot)$, is critical in the application of neural networks.

Noting that $g'(a) = g(a)(1 - g(a))$, this leads to the following parameter update equations:

$$\begin{aligned}
\Delta T_0 &= -\eta\left(\frac{y^i - r^i}{r^i}\right),\\
\Delta T_h &= -\eta\left(\frac{y^i - r^i}{r^i}\right)H_h,\\
\Delta W_{0h} &= -\eta\left(\frac{y^i - r^i}{r^i}\right)T_h H_h(1 - H_h),\\
\Delta W_{ih} &= -\eta\left(\frac{y^i - r^i}{r^i}\right)T_h H_h(1 - H_h)u_i.
\end{aligned} \tag{7}$$

4. Regression neural networks

The nonparametric density estimate of probability at a certain value is a weighted sum of the effects of sample points. Given a sample of n real d-dimensional X_i values, the estimate of probability at x is [32]:

$$\hat{p}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{8}$$

K is the kernel density function and h is the window width of the kernel. We can use this approach for regression, i.e., for inference of a scalar y value for a given multi-dimensional vector x if we are given pairs of (X_i, Y_i). The kernel estimate of the joint probability density $f(x, y)$ is then written as:

$$\hat{f}(x, y) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \times \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[-\frac{(y - Y_i)^2}{2\sigma^2}\right]. \tag{9}$$

Using Bayes theorem:

$$\hat{f}(y \mid x) = \frac{\hat{f}(x, y)}{f(x)} = \frac{\hat{f}(x, y)}{\int \hat{f}(x, y)\, \mathrm{d}y}. \tag{10}$$

Our estimate of y is the expected value:

$$E[y \mid x] = \frac{\int y\hat{f}(x, y)\, \mathrm{d}y}{\int \hat{f}(x, y)\, \mathrm{d}y}. \tag{11}$$

Replacing the joint density by its kernel estimate, we get the *Nadaraya–Watson estimator*.

$$\hat{y}(x) = \frac{\sum_i Y_i K((x - X_i)/h)}{\sum_i K((x - X_i)/h)}. \tag{12}$$

Because of its attractive analytical properties [32], the kernel function is generally taken as the Gaussian, named a *Parzen window* [13]:

$$K(u) = \left(\frac{1}{\sqrt{2\pi}}\right)^d \exp\left[-\frac{\|u\|^2}{2}\right]. \tag{13}$$

This then leads to the following equation for our estimate $\hat{y}(x)$:

$$\hat{y}(x) = \frac{\sum_{i=1}^{n} Y_i \exp\left[-\|x - X_i\|^2/2h^2\right]}{\sum_{i=1}^{n} \exp\left[-\|x - X_i\|^2/2h^2\right]}. \tag{14}$$

The estimate $\hat{y}(x)$ is thus a weighted average of all the observed Y_i values where each weight is exponentially proportional to its Euclidean distance from x. Note that a good choice of the window width h is critical. If it is large then even distant neighbors affect the estimate at x leading to a very smooth estimate. In the extreme case when h goes to infinity, our estimate is the average of all Y_i, independent of the input. When h is small, only a few (if any) samples play a role, leading to a noisy estimate.

The "bias/variance dilemma" stated by Geman, Bienenstock, and Doursat [17], points out that when h is large, $\hat{y}(x)$ is an average of many samples and the variance contribution of our estimator is small as our estimator does not change from one training set to another (of course assuming that they are all drawn from the same distribution). The bias is large in this case as our estimate is biased toward the population response. When h is small, there is small bias but the estimate is dependent on the particular training set used. Therefore the variance contribution is high. Thus choosing h implies a trade-off between bias (systematic error) and variance (random error). h is determined by trial and error, cross-validating on a separate test set. A more detailed analysis of the approach and its properties are beyond the scope of this present paper; the interested reader is referred to [33].

Back-propagation rule is generally criticized on the grounds that learning takes many epochs over the training set and also that the performance depends on a good choice of H, the number of hidden units. The method just explained belongs to the class of *memory-based methods* where approximation is directly done from a table of stored values. These methods require only one epoch over the training set, i.e., to read in the table. Here $\boldsymbol{\theta}$, the parameter vector, includes the whole training set and h. Thus learning is fast but the memory requirement is large. There are also methods to choose a subset of the training set without decreasing performance; see [2] for a review.

As also noted by Specht who names this a *general regression neural network* [34], this approach can also be written in the framework of Eq. (4). Here H is equal to table size, i.e., the number of training patterns with $\boldsymbol{W}_h = \boldsymbol{u}^h$, $T_h = y^h$, the desired output for $\boldsymbol{u}^h$ and we have:

$$g_h(\boldsymbol{W}_h, \boldsymbol{u}) = \frac{\exp\left[-\|\boldsymbol{W}_h - \boldsymbol{u}\|^2/2h^2\right]}{\sum_{i=1}^{H} \exp\left[-\|\boldsymbol{W}_i - \boldsymbol{u}\|^2/2h^2\right]}. \tag{15}$$

The problems of finding good input and output codings apply also here.

5. Combining multiple estimators

There are many choices that should be made before an estimator can be used in an application. These include the model, method of estimating parameters, i.e., learning rule, training sample, input representation, error function minimized, etc. Each choice made is one additional sort of bias which may not be appropriate. The general approach is to try alternatives and choose the one that best performs on a test sample, distinct from the sample used for training, i.e., cross-validation. We advocate in this section that it is better not to discard alternatives but combine all to improve performance.

There are two ways in which multiple estimators can be combined: one is *voting* and the other is *stacking*. Note that regardless of the way we combine the models, there is no reason to expect improvement in performance through having multiple models if they are quite similar. One can only get fault tolerance through redundancy if the models fail under different circumstances. In neural networks, choosing parameters randomly like the initial weight values and hyperparameters like the network architecture or by having different learning rules, one guarantees this to a certain extent. Basically as we will see in the next subsection, when we have one model that has a certain success, we want to add a model that succeeds best for inputs on which the first one fails; we do not care about the new model's *overall* performance.

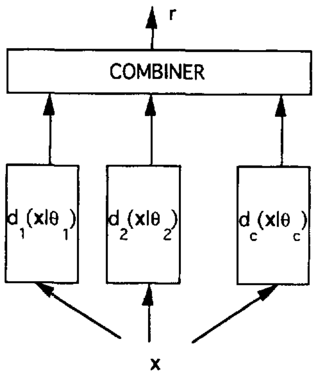

Fig. 1. Block diagram of combining multiple estimators. The combiner takes the predictions of the estimators as input and makes the final prediction. The combiner can be a fixed voting system or an estimator that is also trained.

As seen in Fig. 1, we have c estimators, $d_j(\boldsymbol{x} \mid \theta_j)$, each one trained separately. Then during estimation, each one gives an output and these outputs are combined by a *combiner* system to determine the final output r. We mention two types of combiners: The *voting* system computes a weighted sum where weights are fixed. In the *stacking* approach, the combiner is also an estimator that is trained.

In our application, we combined the three estimators – parametric, multilayer perceptron and the regression neural network – using both types of combining; parametric, multilayer perceptron, and the regression neural network. We believe them to be sufficiently different in our application of distance estimation.

5.1. Voting

In voting with c voters, each estimator $d_j(x \mid \theta_j)$ gives an output for the input x. The final output, r, is a weighted sum [1]:

$$r = \sum_{j=1}^{c} d_j(x \mid \theta_j)\omega_j, \tag{16}$$

where $\sum_{j=1}^{c} \omega_j = 1$. Unless there is a prior reason to favor one voter over another, weights are taken as equal: $\omega_j = 1/c$. If we look at the variance of r [26,30]:

$$\begin{aligned}\mathrm{Var}(r) &= \mathrm{Var}\left(\frac{1}{c}\sum_{j=1}^{c} d_j\right)\\ &= \frac{1}{c^2}\left[\sum_j \mathrm{Var}(d_j) + 2\sum_j \sum_{k \neq j} \mathrm{Cov}(d_j, d_k)\right].\end{aligned} \tag{17}$$

If the models are statistically independent, the second term cancels and variance (and mean square error) decreases with increasing c. However variance can further be decreased by choosing mutually "orthogonal" methods [37]. For example if we have two models, instead of having them independent, we do much better by having them negatively correlated.

It may be possible to have an a priori preference of one estimator over another then we would like to have the weights of those estimators in voting higher. We can for example assess them over a cross-validation set and use their performances to determine weights so as to assign larger weights to more successful voters. Using Occam's razor, we can also a priori assume that complex models with many free parameters tend to overfit on a small sample. Thus we may like to have the tendency to give them less weight unless they are more successful than simpler models, i.e., their additional complexity is justified [1]. We have not tested these two latter approaches in this present work.

5.2. Stacking

Combination of multiple estimators can also be considered as an estimation problem. Wolpert who initiated the idea calls it *stacked generalization* [37]. Breiman converted it to statistical language as *stacked regression* [8]. Here we have two levels of estimators. First on level 0, L_0, are our usual estimators chosen carefully to be as different as possible while maintaining individually acceptable levels of performances. On top of these, on level 1, L_1, is another estimator (the combiner) whose inputs are the outputs of the first level estimators.

L_0 estimators are trained separately on the same training sample. Then they are applied to a different set on which they make estimations. The L_1 estimator takes those estimations as input and maps them to the correct value. Thus L_0 and L_1 estimators are trained on separate training sets. This is because the performances of the L_0 estimators on the data that they are trained on can be very different from their behavior on the data that they are *not* trained on; this is exactly what L_1 should know about and be trained on. This is generally done by dividing the training set into two and using one half for training each level. When the training set is small, one can use leave-k-out (jackknife) to avoid losing precious data. Significant improvement in success has been achieved using this approach on the classification task of protein secondary structure prediction [38].

6. Implementation results

6.1. Database

We have collected two distinct samples by pairing respectively 28 and 23 cities of Turkey for training and test sets. The first is used for estimating the parameters and the second one to assess their performances. The training and test data sets contain planar coordinates and intercity distances for respectively 28 and 23 cities which make $28*27/2 = 378$ and $23*22/2 = 253$ data pairs. There are 74 cities in Turkey thus the complete data set contains $74*73/2 = 2701$ pairs and our training set constitutes 14% of it. The third dimension is ignored since previous empirical studies have shown that the effect of height in the accuracy of estimations in Turkey is almost null [4]. The values we report in the following sub-sections are average error per pair in kilometers on both the training and test sets. Recall that the error per pair is measured by the normalized error measure given in Eq. (3).

6.2. Distance functions

By looking at properties of the application, it is realistic to eliminate some of the distance functions a priori by judging them with the structural properties of the actual road network. First of all, the road structure in Turkey has been developed arbitrarily rather than rectilinearly or ring-radially. This arbitrariness makes the identification of a fixed pattern for possible travel directions within the country impossible; this is crucial for the use of block norms. Besides, the area is too small to require the consideration of the earth's roundness in estimating actual distances. Being convinced by these observations, it is rational to concentrate on functions $d_2(\boldsymbol{x})$, $d_3(\boldsymbol{x})$, $d_4(\boldsymbol{x})$, $d_5(\boldsymbol{x})$ and compute the best possible value of the parameters k, p, s, k_1, and k_2. In spite of this fact, we also computed the value of k for $d_1(\boldsymbol{x})$ since this function has been heavily considered in location literature [15].

The calculation of the parameters with respect to any of the estimation criteria introduced in Section 1 requires minimizing an error function like Eq. (1). Since we use the normalized error function given in Eq. (3), the minimization problems are continuous in the parameters. They are also well-behaved in the sense that any Karush–Kuhn–Tucker point with respect to the unknowns k, p, s, k_1, and k_2 is optimal. For $d_1(\boldsymbol{x})$, $d_2(\boldsymbol{x})$, and $d_5(\boldsymbol{x})$, we have analytical solutions and the values of k, k_1, k_2, which minimize error function (3), can be obtained easily by using the following equalities.
For $d_1(\boldsymbol{x})$:

$$k=\frac{\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|+|x_{12}^i-x_{22}^i|\right)}{\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|+|x_{12}^i-x_{22}^i|\right)^2/r^i}. \tag{18}$$

For $d_2(\boldsymbol{x})$:

$$k=\frac{\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|^2+|x_{12}^i-x_{22}^i|^2\right)^{1/2}}{\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|^2+|x_{12}^i-x_{22}^i|^2\right)/r^i} \tag{19}$$

and for $d_5(\boldsymbol{x})$:

$$k_1=\frac{a_{13}a_{22}-a_{12}a_{23}}{a_{21}^2-a_{22}a_{11}},\quad k_2=\frac{a_{21}a_{13}-a_{23}a_{11}}{a_{21}^2-a_{22}a_{11}}, \tag{20}$$

where

$$a_{11}=\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|+|x_{12}^i-x_{22}^i|\right)^2/r^i, \tag{21}$$

$$\begin{aligned}a_{21}&=a_{12}\\&=\sum_{i=1}^{n}\frac{1}{r^i}\Big[\left(|x_{11}^i-x_{21}^i|+|x_{12}^i-x_{22}^i|\right)\\&\qquad\times\left(|x_{11}^i-x_{21}^i|^2+|x_{12}^i-x_{22}^i|^2\right)^{1/2}\Big],\end{aligned}$$

$$a_{22}=\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|^2+|x_{12}^i-x_{22}^i|^2\right)/r^i,$$

$$a_{13}=\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|+|x_{12}^i-x_{22}^i|\right),$$

$$a_{23}=\sum_{i=1}^{n}\left(|x_{11}^i-x_{21}^i|^2+|x_{12}^i-x_{22}^i|^2\right)^{1/2}.$$

However, the calculations for k and p of $d_3(\boldsymbol{x})$, and k, p and s of $d_4(\boldsymbol{x})$ are slightly more complicated. They require the solution of the following unconstrained optimization problems.
For $d_3(\boldsymbol{x})$:

$$\min_{k,p}\sum_{i=1}^{n}\left[\frac{k\left(|x_{11}^i-x_{21}^i|^p+|x_{12}^i-x_{22}^i|^p\right)^{1/p}-r^i}{\sqrt{r^i}}\right]^2. \tag{22}$$

For $d_4(\boldsymbol{x})$:

$$\min_{k,p,s}\sum_{i=1}^{n}\left[\frac{k\left(|x_{11}^i-x_{21}^i|^p+|x_{12}^i-x_{22}^i|^p\right)^{1/s}-r^i}{\sqrt{r^i}}\right]^2. \tag{23}$$

Although they are quite simple with respect to the number of variables, which is two and three respectively, the number of nonlinearities induced by these problems can be very large depending on the size of the pairs of points within the training set. Their minimizations were carried out using MINOS 5.1 [28]. The results are given in Table 2. Observe that the rectilinear distance function has the worst performance. Meanwhile the accuracy of $d_4(\boldsymbol{x})$ is the highest. These facts definitely support our previous inference on the arbitrariness of the road structure in Turkey.

The parameters of the distance functions should be customized according to the geographical characteristics of the region they are to be used; this has

Table 2
Average error per pair using the five parametric distance functions

Data sets	Distance function				
	$d_1(x)$	$d_2(x)$	$d_3(x)$	$d_4(x)$	$d_5(x)$
Parameters	$k = 1.073$	$k = 1.310$	$k = 1.262$ $p = 1.603$	$k = 1.688$ $p = 1.686$ $s = 1.761$	$k_1 = 0.280$ $k_2 = 0.971$
Average error on training set	8.55	4.54	3.92	3.61	3.98
Average error on test set	12.49	9.05	8.48	8.10	8.49

been agreed as a consequence of many empirical studies previously mentioned. In other words, a distance function must be flexible enough to introduce also the effect of regional geography on distances into the estimations. An interesting attempt for modelling regional characteristics has been realized first by Love and Morris [21] who suggested the rotation of the original coordinate axes by an angle ψ, before using the weighted L_1 norm, $d_1(x)$. This approach results in the following two-parameter distance function:

$$d_1(\hat{x}) = k\left(| \hat{x}_{11} - \hat{x}_{21} | + | \hat{x}_{12} - \hat{x}_{22} | \right). \tag{24}$$

Here, $\hat{x}_{11}$, $\hat{x}_{12}$, $\hat{x}_{21}$, and $\hat{x}_{22}$ are the coordinates in the rotated system and they can be calculated with the following formula when ψ is given:

$$\begin{pmatrix} \hat{x}_{11} & \hat{x}_{12} \\ \hat{x}_{21} & \hat{x}_{22} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}. \tag{25}$$

Later on axes rotation has been applied to two more general distance functions: the weighted L_p norm, $d_3(x)$ [11,24], and the weighted one-two norm $d_5(x)$ [9,10]. Experiments show that rotation improves the accuracy of estimations.

We also studied the effect of rotation for distance functions $d_1(x)$ to $d_5(x)$ by using our data. Although rotation requires the addition of ψ as the new parameter to the original parameter vector $\boldsymbol{\theta}$, which is shown in the second column of Table 1, the method that we used to obtain Table 2 can be adopted easily. Once ψ is fixed, namely axes are rotated for a certain angle, then Eqs. (18) to (21) can be used to compute parameters k, k_1, and k_2 of the distance functions $d_1(x)$ and $d_5(x)$ and the unconstrained optimization problems (22) and (23) can be solved in order to compute the parameters k, p, and s of $d_3(x)$ and $d_4(x)$. Note that $d_2(x)$ is invariant under rotation, and therefore it is not considered in this part of the study. The results are given in Table 3. They support the findings of Table 2. The consideration of ψ as the new parameter increases the accuracy. However, the rectilinear distance function has again the worst performance and the accuracy of

Table 3
Average error per pair using the four parametric distance functions with rotation

Data sets	Distance function			
	$d_1(x)$	$d_3(x)$	$d_4(x)$	$d_5(x)$
Parameters	$\psi = 5°$ $k = 1.069$	$\psi = 48°$ $k = 1.395$ $p = 2.740$	$\psi = 50°$ $k = 1.807$ $p = 2.548$ $s = 1.761$	$\psi = 4°$ $k_1 = 0.290$ $k_2 = 0.958$
Average error on training set	8.28	3.82	3.52	3.94
Average error on test set	12.90	8.45	8.09	8.56

$d_4(x)$, which is closely followed by the accuracies of $d_3(x)$ and $d_5(x)$, is still the highest.

6.3. Multi-layer perceptrons

We employed a multi-layer perceptron with one hidden layer with the back-propagation learning rule. After several trials, the best input representation was determined as the four data values and the Euclidean distance in between as a hint:

$$u = \phi(x) = (x_{11}, x_{12}, x_{21}, x_{22}, \| x_1 - x_2 \|)^T.$$

In terms of output representation, we found out that learning the ratio of actual distance to Euclidean distance is better than learning the actual distance itself. This ratio is called the *directional bias* by Brimberg and Love [10] and Brimberg and Wesoloswky [12]:

$$y = \chi(r) = r / \| x_1 - x_2 \|.$$

This may be thought of as extending $d_2(x)$ in the parametric case in that, instead of computing one constant global k factor, it is as if the neural network computes a continuous function $k(x)$ by which it

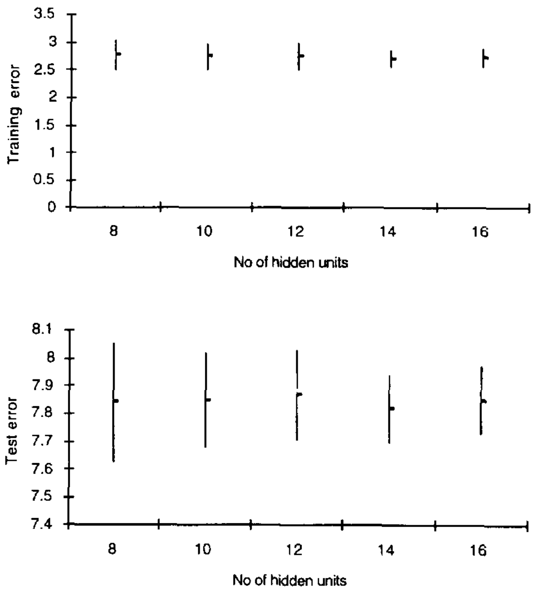

Fig. 2. Average error of multi-layer perceptrons on the training and test sets as a function of the number of hidden units after 10 independent runs. Squares denote means and error bars are one standard deviation long on either side.

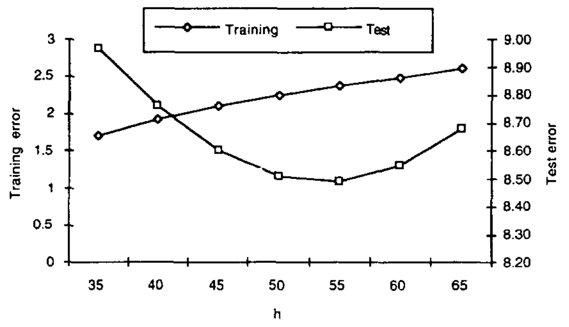

Fig. 3. Average error of regression neural network on the training and test sets as a function of the window width.

scales the Euclidean distance. Note that in these two cases, we can take advantage of our a priori knowledge that the distances are symmetric, namely, $d(x_1, x_2) = d(x_2, x_1)$, and double the training set. This can be done by adding data triples $(x_2, x_1, d(x_1, x_2))$ to the triples $(x_1, x_2, d(x_1, x_2))$ in the training set. In Fig. 2, average errors and standard deviations over the training and test sets are given over 10 independent runs as a function of the number of hidden units.

We used a momentum term and updated η dynamically for faster convergence. Training stops when no further improvements were made. Networks require around a hundred epochs for convergence after which slight improvements (less than 5%) are achieved if learning is continued.

6.4. Regression neural networks

By playing with the alternatives, we found out that the following input representation was the best:

$$u = \phi(x) = (|x_{11} - x_{21}|, |x_{12} - x_{22}|, \| x_1 - x_2 \|)^T.$$

Kernel-based estimation methods suffer from the "curse of dimensionality" [13] because the neighborhood information is lost when the dimensionality of the space increases. Thus decreasing the input dimensions from five to three helped. The output was the desired distance:

$$y = \chi(r) = r.$$

Results achieved are given in Fig. 3 as a function of the window width, h. Note that there is a large

Table 4
Average error of three estimators and the result of voting

Estimators	Training error	Test error
Parametric model	3.61	8.10
Multi-layer perceptron	2.63	7.91
Regression neural network	2.38	8.49
Voting	2.26	7.63

dependence on h. Overall, the success is slightly inferior to that of the multi-layer perceptron.

6.5. Combining estimators

6.5.1. Voting

Taking into account the fact that combining multiple estimators is a better idea when estimators are as different as possible, we used the following three estimators:

(1) Parameterized distance function, $d_4(\boldsymbol{x})$ with three parameters k, p, and s.
(2) Multi-layer perceptron with 12 hidden units.
(3) Regression neural network with $h = 54$.

The votes are taken as equal. Results achieved are given in Table 4. Note that the result of voting is better than the result of all voters. This indicates that estimators do fail under different circumstances.

6.5.2. Stacking

We used again three estimators as our L_0 estimators:

(1) Parameterized distance function, $d_2(\boldsymbol{x})$ with the parameter k.
(2) Multi-layer perceptron with 12 hidden units.
(3) Regression neural network with $h = 54$.

The combiner L_1 estimator is a multi-layer perceptron with 2 hidden units. We divided the training set into two parts performing "leave-$n/2$-out"; T_0 and T_1. We have trained all three L_0 estimators using T_0 only and saved their predictions on T_1 as R_1. Then we have trained them on T_1 and saved their predictions on T_0 as R_0. We have trained the combiner network L_1 on R_0 and R_1. After training the combiner network, we have trained all three on the complete training set. Stacking is superior to voting as is also shown in our study. Values reported are average errors on both the training and test sets (Table 5).

Table 5
Average error achieved through voting vs. stacking

Combiner	Training error	Test error
Voting	2.26	7.63
Stacking	3.81	7.41

7. Conclusions

We compare approaches based on the three criteria of accuracy, memory requirement, and time.

• *Accuracy.* We see that the multi-layer perceptron can be a better estimator than the parameterized distance functions in terms of estimation accuracy. This is because in the latter models, the parameters are fixed over the whole space. For example when $d_2(\boldsymbol{x})$ is used, k is constant over the whole country but we know that for example in Turkey, the eastern part is more mountainous than the western part and thus k there is higher than k of the west as a consequence of the extra curvature caused by these natural obstacles. The multi-layer perceptron may be seen as approximating an input dependent, continuous function $k(\boldsymbol{x})$. Being nonparametric, it does this without assuming any a priori form. An approach in distance function based estimation is to divide the space into multiple regions and make separate estimations in separate regions; this is a piecewise constant approximation of the $k(\boldsymbol{x})$ [3].

Though the regression neural network and the parameterized distance function do not perform as well as the multi-layer perceptron, when they are combined, their accuracy is higher. This indicates that on cases where the perceptron fails, the others make good guesses. The stacking approach as it is trained performs better than voting.

• *Memory Requirement.* Out of the three estimators we have implemented, the regression neural network has the highest memory requirement by storing $4n + 1$ parameters: 4 values for each pair for a training set of n pairs and the parameter h, here $4*378 + 1 = 1513$. When H is the number of hidden units, the multi-layer perceptron requires $6H$ (five inputs and one bias) values for $\boldsymbol{W}$ and $H + 1$ for $\boldsymbol{T}$, thus total of $7H + 1$ weights, here 85. Memory requirement increases when one has multiple models. In the stacking approach, there are also parameters of the combiner network. However, the parameterized distance functions considered in this work respectively have one, one, two, and three parameters in addition to the rotating angle ψ. Therefore, they use memory very efficiently.

• *Learning Time.* The regression neural network is trained in one epoch. Three of the first parameterized distance functions has an analytical solution where we compute k, k_1, and k_2 by making one pass over the training set. The computational effort spent for training the multi-layer perceptron is considerably higher than that is required by the other two parametric models given in Eqs. (22) and (23), as many trials should be made with the multi-layer perceptron to find out the good input and output representations and the number of hidden units. To train the stacking network, when one divides the set into two, the estimators are trained twice. Ideally, "leave-one-out" where at each step, the training set is divided into two parts of size one and $n-1$ is better for bias reduction but it requires n trainings. This is the approach Wolpert originally proposed for stacked generalization [37]. In short the effort required to compute the parameters of distance functions is very small compared to the training effort of neural networks.

It is generally accepted that using this or that learning technique by itself is not sufficient to train a neural network appropriately. A good knowledge of the application and the learning methods are necessary to guide the estimator to learn the important and ignore the irrelevant. As shown in the preceding sections, different methods may benefit from different input and output representations. Perhaps the quality of the training sample is the most important factor that affects the quality of estimation. For example, a large enough and representative sample is required for the approximation not to be biased by the idiosyncrasies of the particular sample used.

Viewing the results achieved, we can say that nonparametric approaches can be considered better than parameterized distance functions in terms of estimation accuracy. This is not just due to having more parameters. The fact that both training *and* test errors are lower implies that the nonparametric methods are able to extract some underlying structure which cannot be captured completely by the parametric approaches. The nonparametric method does not simply memorize the training set but generalizes accurately to unseen patterns in the test set. We believe that this is due to the fact that the nonparametric estimator can adjust itself to follow the change in the geographical characteristics of a region as opposed to the parameterized distance function estimating over the whole country. On the other hand, the advantage of the parameterized methods is that because they have a small number of parameters, they require much less memory and smaller training samples, making them suitable for applications where memory and learning time is limited.

Parametric distance functions are often incorporated into the objective functions of many optimization models, e.g. the objective function of continuous space facility location problems. This is also possible for neural network estimators. The optimization of the objective function calculated via a neural network can, in principle, be undertaken by standard optimization algorithms. Analytic formulae for the gradient and Hessian of the function on the network are available, for example for the case of multi-layer perceptrons. The major drawback of these functions compared to distance functions, is their being nonconvex, which is not the case for the parametric distance functions we study in this work – at least under certain restrictions on the parameters k, p, s, k_1, and k_2 [20,21]. Convexity with respect to the coordinates is an important property because the effort necessary to spend on the solution of the optimization models decreases drastically if the objective function to be minimized is a convex function and the solution space is a convex set.

The function calculated via a neural network is very suitable for parallel computations, which may provide considerable advantages especially when many different optimization calculations have to be undertaken.

In short there is not one method that is significantly superior to others in all four respects of accuracy, memory requirement, learning time, and the advantages it provides when incorporated into objective functions of optimization models. Thus when choosing a particular estimator, all these aspects should be taken into account according to the particular implementation constraints and not only accuracy as it is frequently done in the literature. Finally, we advocate the use of multiple estimators and combining them to get accurate estimators.

Acknowledgements

We thank the two anonymous referees for their construtive comments and informing us of recent work on parameterized distance functions. We thank Haluk Erkut and Seçkin Polat of the Department of Industrial Engineering, İstanbul Technical University for the first inspiration and continuous encouragement, D.H. Wolpert of Santa Fe Institute for giving pointers to related literature on combining estimators by stacking. This work is supported in part by Grants TBAG-1336 and EEEAG-143 from Tübitak, the Turkish Scientific and Technical Research Council.

References

[1] Alpaydın, E., "Multiple networks for function learning", *IEEE International Neural Network Conference*, San Francisco, CA, Vol. 1 (1993) 9–14.

[2] Alpaydın, E., "GAL: Networks that grow when they learn and shrink when they forget", *International Journal of Pattern Recognition and Artificial Intelligence* 8 (1994) 391–414.

[3] Altınel, İ.K., and Aras, N., "Estimating road distances in İstanbul with single and multi regional models", Research Paper Series No: FBE-IE-05/94-05, Department of Industrial Engineering, Boğaziçi University, İstanbul, 1994.

[4] Altınel, İ.K., Aras, N., Alie, A., Cangür, G., Özel, R., and Yücel, A., "Estimating road travel distances in Türkiye", Research Paper Series No: FBE-IE-03/94-03, Department of Industrial Engineering, Boğaziçi University, İstanbul, 1994.

[5] Berens, W., "The suitability of the weighted L_p norm in estimating actual road distances", *European Journal of Operational Research* 34 (1988) 39–43.

[6] Berens, W., and Körling, F., "Estimating road distances by mathematical functions", *European Journal of Operational Research* 21 (1985) 54–56.

[7] Berens, W., and Körling, F., "On estimating road distances by mathematical functions – A rejoinder", *European Journal of Operational Research* 36 (1988) 254–255.

[8] Breiman, L., "Stacked regression," TR-367, Department of Statistics, University of California, Berkeley (1992).

[9] Brimberg, J., Dowling, P.D., and Love, R.F., "The weighted one-two norm distance model: Empirical validation and confidence interval estimation", *Location Science* 2 (1994) 91–100.

[10] Brimberg, J., and Love, R.F., "A new distance function for modelling travel distances in a transportation network", *Transportation Science* 26 (1992) 129–137.

[11] Brimberg, J., Love, R.F., and Walker J.H. "The effect of axis rotation on distance estimation", *European Journal of Operational Research* 80 (1995) 357–364.

[12] Brimberg, J., and Wesolowsky, G.O., "Probabilistic L_p distances in location models", *Annals of Operations Research* 40 (1992) 67–75.

[13] Duda, R.O., Hart, P.E., *Pattern Classification and Scene Analysis*, John Wiley, New York, 1973.

[14] Erkut, H., and Polat, S., "A simulation model for a urban fire fighting system", *Omega* 20 (1992) 535–542.

[15] Francis, R.L., McGinnis, L.F. Jr., and White, J.A., *Facility Layout and Location: An Analytical Approach*, 2nd edition, Prentice Hall, Englewood Cliffs, NJ, 1992.

[16] Funahashi, K., "On the approximate realization of continuous mapping by neural networks", *Neural Networks* 2 (1989) 183–192.

[17] Geman, S., Bienenstock, E., and Doursat, R., "Neural networks and the bias/variance dilemma", *Neural Computation* 4 (1992) 1–58.

[18] Hertz, J., Krogh, A., and Palmer, R.G., *Introduction to the Theory of Neural Computation*, Addison Wesley, Reading, MA, 1991.

[19] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators", *Neural Networks* 2 (1989) 359–366.

[20] Love, R.F., and Morris, J.G., "Modelling inter-city road distances by mathematical functions", *Operational Research Quarterly* 23 (1972) 61–71.

[21] Love, R.F., and Morris, J.G., "Mathematical models of road travel distances", *Management Sciences* 25 (1979) 130–139.

[22] Love, R.F., and Morris, J.G., "On estimating road distances by mathematical functions", *European Journal of Operational Research* 36 (1988) 251–253.

[23] Love, R.F., Morris, J.G., and Wesolowsky, J., *Facilities Location: Models and Methods*, North-Holland, New York, 1988.

[24] Love, R.F., and Walker, J.H., "An empirical comparison of block and round norms for modelling actual distances", *Location Science* 2 (1994) 21–43.

[25] Love, R.F., Walker, J.H., and Tiku, M.L., "Confidence intervals for $l_{k,p,\theta}$ distances", *Transportation Science* 29 (1995) 93–100.

[26] Mani, G, "Lowering variance of decisions by using artificial neural network portfolios", *Neural Computation* 3 (1991) 484–486.

[27] Mittal, A.K., and Palsule, V., "Facilities location with ring radial distances", *Institute of Industrial Engineers Transactions* 16 (1984) 59–64.

[28] Murtagh, B.A., and Saunders, M.A., "MINOS 5.1 User's Guide", Technical Report No: SOL 83-20R, Stanford University, Stanford, CA, 1983 (revised 1987).

[29] Perreur, J., and Thisse, J., "Central metrics and optimal location", *Journal of Regional Science* 14 (1974) 411–421.

[30] Perrone, M.P., Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization, Ph.D. Thesis, Department of Physics, Brown University.

[31] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning internal representations by error propagation," in: D.E. Rumelhart, J.L. McClelland and the PDP Research Group (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol 1., MIT Press, New York, 1986, 318–362.

[32] Silverman, B.W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[33] Stone, C.J., "Consistent nonparametric regression", *The Annals of Statistics* 5 (1977) 595–645.

[34] Specht, D.F., "A general regression neural network", *IEEE Transactions on Neural Networks* 2 (1991) 568–576.

[35] Ward, J.E., and Wendell, R.E., "A new norm measuring distance which yields linear location problems", *Operations Research* 28 (1980) 836–844.

[36] Ward, J.E., and Wendell, R.E., "Using block norms for location modelling", *Operations Research* 33 (1985) 1074–1091.

[37] Wolpert, D.H., "Stacked generalization", *Neural Networks* 5 (1992) 241–259.

[38] Zhang, X., Mesirov, J.P., Waltz, D.L., "Hybrid system for protein secondary structure prediction", *Journal of Molecular Biology* 225 (1992) 1049–1063.