Median Radial Basis Functions Neural Network

Adrian G. Borş and Ioannis Pitas

Department of Informatics
University of Thessaloniki
Thessaloniki, Greece

Abstract

Radial Basis Functions (RBF) consists of a two-layer neural network, where each hidden unit implements a kernel function. Each kernel is associated with an activation region from the input space and its output is fed to an output unit. In order to find the parameters of a neural network which embeds this structure we take into consideration two different statistical approaches. The first approach uses classical estimation in the learning stage and it is based on the learning vector quantization algorithm and its second order statistics extension. After the presentation of this approach, we introduce the Median Radial Basis Functions (MRBF) algorithm based on robust estimation of the hidden unit parameters. The proposed algorithm employs the marginal median for kernel location estimation and the median of the absolute deviations for the scale parameter estimation. A histogram-based fast implementation is provided for the MRBF algorithm. The theoretical performance of the two training algorithms is comparatively evaluated when estimating the network weights. The network is applied in pattern classification problems and in optical flow segmentation.

1 Introduction

Functional estimation is an important problem in data analysis and pattern recognition applications. The properties of Radial Basis Functions (RBF) make them suitable to be used as universal approximators [1]-[3]. A continuous function can be described as a weighted sum of kernels. A kernel decomposition is suitable to be embedded in a two-layer neural network structure where each kernel is implemented by a hidden unit. In supervised learning, the network is provided with input-output pairs of samples drawn from an observation set and the learning algorithm finds the rules that model the given mapping.

In [4] a window function was associated with each data sample. In other approaches, a distribution function is approximated by the superposition of a set of basis functions whose centers are situated on a regular grid [5]. The assignment of so many functions is not practical in most of the applications and further work was devoted to representing a set of data with one function, with the closest approximation possible. In RBF networks, an activation region with respect to the data sample local densities is assigned, after the learning stage, to each hidden unit. In [6] the RBF weights are found by solving the given system of equations and considering the desired values for the training set. The adaptive implementation of this approach [7] is related with clustering techniques such as adaptive k-means clustering algorithm [8] and learning vector quantization [9].

The RBF neural networks can be used to model the probability density functions (*pdf*) in nonparametric classification tasks [10]-[12]. The basis functions, when used as activation functions for hidden units, provide the network with the capability of forming complex separation boundaries between classes, which is equivalent to what perceptron networks can provide through an intermediate mapping. The main applications for the RBF have been so far in pattern classification where the network approximates the Bayesian classifier [12]-[14] and in system modeling [7, 15, 16]. In both areas, RBF networks gave better results when compared to other methods. The RBF network requires less computation time for the learning [7] and a more compact topology than other neural networks [17]. Various learning algorithms have been used in order to find the most appropriate parameters for the RBF decomposition. They can be classified in two major branches: batch learning, where the learning is done on groups of patterns [6, 13, 14, 18] and on-line learning, where the learning is adaptive, on a per pattern basis [7, 15, 16, 19]. These algorithms were employed for time series prediction [15], speech recognition [7], channel equalization [16] and image quantization [12].

The Gaussian function is usually chosen as kernel function and it will be considered in this study as well. The parameters to be estimated in this case are the center vector and the covariance matrix for each kernel as well as the weighting parameters corresponding to the output connections. The calculation of Gaussian centers [7] is similar to the Learning Vector Quantization (LVQ) algorithm [9, 20] and this approach can be extended for the calculation of the covariance matrices [16, 21]. The Gaussian centers correspond to the local estimates for the first order statistics and covariance matrices for the second order statistics. However, the estimators based on classical statistics produce bias in parameter estimation, if data are not normally distributed [22]. Robust statistics, which are extensively used in image processing algorithms [23] are known to give good results when data are contaminated with outliers or when the distributions are long-tailed. In this paper we introduce a learning algorithm for RBF networks based on robust estimation.

The RBF network is presented in detail in Section 2. The classical approach as well as its statistical interpretation are described in Section 3.1. In Section 3.2 we introduce a novel on-line learning algorithm called Median Radial Basis Functions (MRBF). For the estimation of the RBF centers we employ the Marginal Median estimator [23, 24] and for the estimation of the covariance matrix, the Median of the Absolute Deviations (MAD) algorithm [22]. The median algorithm relies on the ordering of the coming data samples in a running window with the decision to be assigned to that pattern situated in the middle of the window. The number of patterns to be taken into account by the algorithm depends on how fast the distribution of the data changes in time. A fast computing algorithm based on data sample histogram analysis is derived for the MRBF in Section 3.3. This implementation is very useful in the case when data have discrete values, e.g., in image processing and computer vision applications. In Section 4 the expected stationary values are derived in the case when we estimate the parameters from a mixture of one-dimensional (1-D) Gaussian functions. We provide the theoretical bounds for mean and variance estimators in the case when we use either classical or robust estimation. In the case where estimating the parameters of each function from a mixture of distributions we investigate the parameter convergence to the stationary values. In Section 5.1 both algorithms are applied in artificially generated data classification problems. In this application, the networks model the underlying probability for each class using the decomposition in RBF kernels. In order to decide the class for a new data sample, both Euclidean and Mahalanobis [8, 25] distances are used. The figures of merit are the classification error, the capability of functional approximation as well as the estimation of the optimal boundary between the classes. In Section 5.2 the proposed algorithm is applied for optical flow segmentation and in Section 6 we draw the conclusions of the present study.

2 RBF network and optimal classification

The RBF network has a feed-forward topology which models a mapping between a set of vector entries and a set of outputs, by using an intermediate level of representation implemented by the radial basis functions. The structure of this network is represented in Figure 1. Each network input is assigned to a vector entry (feature in a pattern recognition application) and the outputs correspond either to a set of functions to be modeled by the network or to various associated classes.

In supervised learning, the network is provided with a training set of patterns consisting of vectors and their corresponding classes. Each pattern is considered assigned only to one class C_k, according to an unknown mapping. After an efficient learning stage, the network implements the mapping rule and generalizes it for patterns which are not from the training set. According to Bayes theorem [25] we can express the relation among the a posteriori probabilities $P(C_k|\mathbf{X})$ of different classes by using their a priori probabilities $P(C_k)$:

$$P(C_k|\mathbf{X}) = \max_{j=1}^{M} P(C_j|\mathbf{X}) \tag{1}$$

$$p_k(\mathbf{X}) = P(C_k)\, p(\mathbf{X}|C_k) = \max_{j=1}^{M} [P(C_j)\, p(\mathbf{X}|C_j)] \tag{2}$$

where M is the number of classes and $\mathbf{X}$ is an N-dimensional vector denoting a pattern. Providing their capabilities of approximation [1]-[3], RBF networks can be used to describe the underlying probability as a sum of components with respect to a base (denoted by the function family ϕ) :

$$p_k(\mathbf{X}) = \sum_{j=1}^{L} \lambda_{k,j}\phi_j(\mathbf{X}) \tag{3}$$

where L is the number of kernel functions and $\lambda_{k,j}$ are the hidden unit to output weights.

Each hidden unit implements a Gaussian function

$$\phi_j(\mathbf{X}) = \exp\left[-(\mu_j - \mathbf{X})^T \, \mathbf{\Sigma}_j^{-1}(\mu_j - \mathbf{X})\right] \quad (4)$$

for $j = 1, \ldots, L$, where μ_j is the mean vector and $\mathbf{\Sigma}_j$ is the covariance matrix. Geometrically, μ_j represents the center and $\mathbf{\Sigma}_j$ the shape of the j-th basis function. A hidden unit function can be represented as a hyper-ellipsoid in the N-dimensional space. According to the function used (4), an activation region is defined around the mean vector. If a pattern falls inside the activation region of a hidden unit, that neuron will fire. The maximum activation for the hidden unit is obtained when the sample is identical to the center.

The activation region for a neuron is similar to the Voronoi neighborhood for a vector quantizer [26]. Let us denote by V_j the activation region of the j-th kernel with respect to a metric distance

$$V_j = \{\mathbf{X} \in I\!R^N \,|\, \|\mathbf{X} - \mu_j\| \leq \|\mathbf{X} - \mu_i\| \; i = 1, \ldots, L, i \neq j\} \quad (5)$$

where $\|\cdot\|$ represents a distance metric e.g., Euclidean. The separating boundary between two classes is the location of the vectors which have the same a posteriori probabilities for both classes

$$T_j = \{\mathbf{X} \in I\!R^N \,|\, P(C_j)\, p(\mathbf{X}|C_j) = P(C_{j+1})\, p(\mathbf{X}|C_{j+1})\}. \quad (6)$$

If we consider 1-D data, then we can express $V_j = [T_j, T_{j+1})$.

Each output implements a weighted sum of kernels as given by (3). Classes can be coded in different ways by the outputs. For a more accurate representation of the classes we choose the number of outputs as equal to the number of classes. In this case, the class decision is assigned to the maximal activated output unit (winner take all). The sigmoidal function is used in order to limit the output values in the interval $(0, 1]$:

$$Y_k(\mathbf{X}) = \frac{1}{1 + \exp[-p_k(\mathbf{X})]} \quad (7)$$

for $k = 1, \ldots, M$, where M represents the number of outputs and $p_k(\mathbf{X})$ is given by (3).

The sign of the weights $\lambda_{k,j}$ shows the activation or the inhibition of the hidden unit to output connection. If the sign of the weight $\lambda_{k,j}$ is positive, then the activation region of the kernel j corresponds to the class k; otherwise, it is not associated with the class k.

3 Learning in RBF Networks

3.1 Classical statistics approach

A combined unsupervised-supervised learning technique has been widely used [7, 16, 21] in order to estimate the RBF weights. This is an on-line technique which employs the LVQ algorithm [9, 20] in order to find the input to hidden unit weights in the unsupervised part and the Least Mean Squares [27] for finding the $\lambda_{k,j}$ weights in the supervised part. At each iteration we first update the kernel parameters and afterwards the output weights. The unsupervised part of the learning stage is based on classical statistics assumptions.

In the classical statistics approach, the estimation of the mean and of the covariance matrix for a given population of data samples is given by :

$$E_C[\hat{\mu}_j] = \frac{\sum_{i=1}^{n_j} \mathbf{X}_i}{n_j} \quad (8)$$

$$E_C[\hat{\mathbf{\Sigma}}_j] = \frac{\sum_{i=1}^{n_j}(\mathbf{X}_i - \hat{\mu}_j)^T(\mathbf{X}_i - \hat{\mu}_j)}{n_j - 1} \tag{9}$$

where n_j is the number of data samples from the given data population [25].

In order to decide which class center will be updated, in Learning Vector Quantization (LVQ) the Euclidean distance is computed between the data sample and each center :

$$\text{If } \| \mathbf{X}_i - \hat{\mu}_j \|^2 = \min_{k=1}^{L} \| \mathbf{X}_i - \hat{\mu}_k \|^2 \text{ then } \mathbf{X}_i \in C_j \tag{10}$$

where C_j is the winner class and $\| \cdot \|$ denotes the Euclidean distance. The LVQ algorithm is the adaptive version for (8), computed for patterns assigned to an activation region according to (10). In the original LVQ algorithm, used for RBF training [7, 16], only one center vector is updated. The centers are updated according to the rule

$$\hat{\mu}_j(t+1) = \hat{\mu}_j(t) + \eta\,(\mathbf{X}_i - \hat{\mu}_j(t)) \tag{11}$$

where η is the learning rate and $\hat{\mu}_j(t)$ is the center vector estimate at the moment t. Various decaying rules for the learning rate were tested for the LVQ algorithm [28]. The learning rate which achieves the minimum output variance [29] is

$$\eta = \frac{1}{n_j} \tag{12}$$

where n_j is the number of samples assigned to the cluster j. For the covariance matrix calculation we use the extension of the LVQ algorithm for second order statistics [16, 19] :

$$\hat{\mathbf{\Sigma}}_j(t+1) = \frac{n_j - 2}{n_j - 1}\hat{\mathbf{\Sigma}}_j(t) + \frac{(\mathbf{X}_i - \hat{\mu}_j(t+1))^T(\mathbf{X}_i - \hat{\mu}_j(t+1))}{n_j - 1} \tag{13}$$

where $\hat{\mathbf{\Sigma}}_j(t)$ is the covariance matrix estimate at the moment t. We can observe that the formulae (11,13) are the adaptive versions of (8,9).

In some applications it is worthwhile to use the Mahalanobis distance instead of the Euclidean one for the choice of the winner class. The Mahalanobis distance takes into consideration the covariance matrix for each basis function

$$\begin{aligned} \text{If } (\hat{\mu}_j - \mathbf{X}_i)^T\, \hat{\mathbf{\Sigma}}_j^{-1}(\hat{\mu}_j - \mathbf{X}_i) &= \min_{k=1}^{L} \left[(\hat{\mu}_k - \mathbf{X}_i)^T\, \hat{\mathbf{\Sigma}}_k^{-1}(\hat{\mu}_k - \mathbf{X}_i)\right] \\ &\text{then } \mathbf{X}_i \in C_j. \end{aligned} \tag{14}$$

However, at the start of the learning algorithm, an imprecision in estimating the covariance parameters may occur and this can lead to a singular covariance matrix. Thus, for the first few data samples we can use the Euclidean distance (10) and afterwards employ the Mahalanobis distance (14). The initial values for the centers $\hat{\mu}$ are randomly generated and the covariance matrices are initialized with $\mathbf{0}$.

3.2 Median based estimation for RBF parameters

In the training stage it is desirable to avoid using outliers which may cause bias in the estimation of the RBF network parameters. The patterns which are not consistent with data statistics (noisy patterns) should be rejected rather than used for training [30]. Robust methods are known to provide accurate estimates when data are contaminated with outliers or have long-tailed distributions [22, 23]. They are insensitive at extreme observations and this makes them attractive for parameter estimation. In Marginal Median LVQ algorithm [24, 31], the data samples are marginally ordered and the centroid is taken as the marginal median [23] :

$$\hat{\mu}_j = \text{med}\,\{\mathbf{X}_0, \mathbf{X}_2, \ldots, \mathbf{X}_{n-1}\} \tag{15}$$

where $\mathbf{X}_{n-1}$ is the last pattern assigned to the j-th neuron.

In order to avoid an excessive computational complexity, the median operation can be done on a finite set of data, extracted through a moving window that contains only the last W data samples assigned to the hidden unit j

$$\hat{\mu}_j = \begin{cases} \text{med}\,\{\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{n-1}\} & \text{if } n < W \\ \text{med}\,\{\mathbf{X}_{n-W}, \mathbf{X}_{n-W+1}, \ldots, \mathbf{X}_{n-1}\} & \text{if } n \geq W \end{cases} \tag{16}$$

where $\mathbf{X}_k$, $k = n - W, \ldots, n-1$ are the data samples assigned to the j-th neuron according to (10) or (14).

Window size W is small if the statistics of the sample population change rapidly in time and large if the data samples statistics is relatively unchanged in time and if a more accurate median estimate of the given data population is desired. Unlike in image filtering where the window is rather small, for a good median estimate of the data samples a rather big window should be employed.

For the dispersion vector associated with a kernel function, we use the Median of the Absolute Deviations (MAD) estimator

$$\hat{\sigma}_j = \frac{\text{med}\,\{|\mathbf{X}_0 - \hat{\mu}_j|, \ldots, |\mathbf{X}_{n-1} - \hat{\mu}_j|\}}{0.6745} \tag{17}$$

where 0.6745 is a scaling parameter in order to make the estimator, Fisher consistent for the normal distribution [22]. MAD calculation is performed along each data dimension, independently. The same set of data samples is taken into account in (17) as for the marginal median in (16).

The off-diagonal components of the covariance matrix can be calculated based on robust statistics as well [22]. We consider two arrays containing the difference and sum of each two different components for a data sample from the moving window

$$Z^+_{i,hl} = X_{i,h} + X_{i,l} \tag{18}$$

$$Z^-_{i,hl} = X_{i,h} - X_{i,l} \tag{19}$$

for $i = n - W, \ldots, n-1$. First, the median of these new data populations is calculated according to (16). The squares of the correspondent MAD estimates (17) for the arrays $\mathbf{Z}^+_{hl}$ and $\mathbf{Z}^-_{hl}$ represent their variances and they are denoted as $\hat{V}^+_{j,hl}$ and $\hat{V}^-_{j,hl}$. The off-diagonal components of the covariance matrix are derived as :

$$\hat{\sigma}^2_{j,hl} = \hat{\sigma}^2_{j,lh} = \frac{1}{4}\,(\hat{V}^+_{j,hl} \;-\; \hat{V}^-_{j,hl}\,). \tag{20}$$

In marginal median LVQ, both Euclidean (10) and Mahalanobis distances (14) can be used. In the case of Mahalanobis distance, a good estimation is desired for the covariance matrix in order to be appropriately used for winner class selection. By using a robust estimation of the covariance matrix as in (17)-(20) we can be confident in the evaluation of the Mahalanobis distance. The order of RBF network weights updating is well defined :

the kernel center, the covariance matrix which uses the previously estimated center and afterwards the hidden unit to output weights. The network found by means of the proposed training algorithm is called Median Radial Basis Functions (MRBF) neural network.

The second layer is used in order to group the clusters found in the unsupervised stage in classes. The output weights are updated as follows :

$$\lambda_{k,j}(t+1) = \lambda_{k,j}(t) + \eta_\lambda \left(F_k(\mathbf{X}) - Y_k(\mathbf{X})\right) Y_k(\mathbf{X})(1 - Y_k(\mathbf{X}))\phi_j(\mathbf{X}) \tag{21}$$

for $k = 1, \ldots, M$ and $j = 1, \ldots, L$, where the learning rate is $\eta_\lambda \in (0, 1]$. $F_k(\mathbf{X})$ is the desired output for the pattern vector $\mathbf{X}$ and it is coded as:

$$F_k(\mathbf{X}) = \begin{cases} 1 & \text{if } \mathbf{X} \in C_k \\ 0 & \text{if } \mathbf{X} \notin C_k \end{cases} \tag{22}$$

for $k = 1, \ldots, M$. The formula (21) corresponds to the backpropagation [32, 33] for the output weights of a RBF network with respect to the mean square error cost function [14].

The network topology represents the number of neurons on each layer. The number of inputs and outputs can be set up from the given supervised problem. For evaluating the number of hidden units we can use various approaches : growing architecture, decreasing number of hidden units, or a combination of these two. When the performance of the network is poor, the number of hidden units should be increased [14, 15, 17]. If some hidden units are not relevant for the classification, or their activation fields are overlapping, the network should be pruned [13]. The relevance of the hidden units is calculated based on the ratio between the number of data samples contained in their activation field and the total number of data samples. The overlapping of the activation fields can be evaluated by clustering similarity measures [8].

3.3 Fast training in Median RBF based on data sample histograms

When the data samples are distributed in a discrete range of values we can find solutions for a fast MRBF training stage. A fast implementation for the median algorithm based on histogram updating, used in image filtering, was proposed in [34]. The first data sample assigned to a unit becomes the starting point in finding the median. In the updating stage we take into consideration pairs of data samples $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$, assigned to the same unit according to either (10) or (14). We build up the marginal histogram associated with each activation region, denoted here as $H_{jh}[k]$, where j is the hidden unit, h is the data sample entry and k represents the histogram level. We denote by $\hat{\mu}_{j,h}(t)$ the center estimate at instant t, and let us assume $X_{i,h} < X_{i+1,h}$. Median updating can be performed according to the rank of the incoming data samples :

$$\begin{gathered} \hat{\mu}_{j,h}(t+1) = \hat{\mu}_{j,h}(t) + K \\ \text{If } \hat{\mu}_{j,h}(t) \in (X_{i,h}, X_{i+1,h}) \text{ then } K = 0 \\ \text{If } \hat{\mu}_{j,h}(t) < X_{i,h} \text{ then } K > 0 \\ \text{If } \hat{\mu}_{j,h}(t) > X_{i+1,h} \text{ then } K < 0 \end{gathered} \tag{23}$$

where K is the number of histogram levels necessary to add/subtract in order to obtain the new location for the median. K is evaluated on the condition that median is located where the data marginal histogram splits in two sides containing equal number of samples :

$$\sum_{k=0}^{\hat{\mu}_{j,h}+K} H_{jh}[k] = \sum_{k=\hat{\mu}_{j,h}+K}^{l} H_{jh}[k] \tag{24}$$

where l represents the total number of levels in the updated histogram $H_{jh}[k]$. In some histograms it is not possible to obtain exact equality in the relation (24). The median is chosen such that (24) is best approximated.

We implement a fast calculation of MAD (17) by using the histogram of data samples. In order to estimate the dispersion parameter we use the histograms obtained during median calculation by (23,24). We build new histograms denoted as $H_{jh,MAD}[k]$ and representing the distributions for $|\mathbf{X}_i - \hat{\mu}_j|$

$$H_{jh,MAD}[0] = H_{jh}[\hat{\mu}_{j,h}] \tag{25}$$

$$H_{jh,MAD}[k] = H_{jh}[\hat{\mu}_{j,h} + k] + H_{jh}[\hat{\mu}_{j,h} - k] \tag{26}$$

where k represents the histogram level. MAD represents the weighted median of the $H_{jh,MAD}[k]$ histogram and should fulfill a similar condition as in (23,24). Let us denote by ψ_{jh} the location where the histogram $H_{jh,MAD}[k]$ splits in two equal sides. MAD estimator of the distribution can be derived from the median of this histogram, taking into account the quantization error as well :

$$\hat{\sigma}_{j,h} = \frac{1}{0.6745}\left(\psi_{jh} + \frac{\frac{n_j}{2} - \sum_{i=0}^{\psi_{jh}} H_{jh,MAD}[i]}{H_{jh,MAD}[\psi_{jh}+1]} - 0.5\right) \tag{27}$$

where n_j is the total number of samples assigned to the j-th hidden unit and where 0.5 should be subtracted in order to compensate for the first term of the folded histogram $H_{jh,MAD}[0]$ from (25). The second term represents the quantization error and can be omitted if a faster implementation is desired. It can be easily seen that this term is zero for a properly balanced histogram associated with MAD estimation.

For the off-diagonal components of the covariance matrix we can employ a similar approach to that used for calculating the MAD estimator (25,26). We calculate the joint histograms of Z_{hl}^{+} and Z_{hl}^{-} based on (18,19) and we evaluate the median and MAD estimators similarly with (27). Afterwards, the covariance matrix components are derived as in (20).

4 The estimation of the network weights

The marginal median estimator operates independently on each data axis. Therefore, the performance analysis can be done for the 1-D case without loss of generality. In this Section we estimate the parameters, i.e., the centers and variances of 1-D normal distribution mixtures. The proposed robust estimation techniques are compared against the classical estimators. A mixture of Gaussians is a very general model, used in many applications. We shall perform an asymptotic analysis of performance when we have a sufficiently large number of observations for each class.

Let us take a *pdf* function $f(X)$ equal to a mixture made up of L 1-D normal distributions $N(\mu_j, \sigma_j)$, each of them with a priori probability ε_j

$$f(X) = \sum_{j=1}^{L} \frac{\varepsilon_j}{\sqrt{2\pi}\sigma_j} \exp\left[-\frac{(X-\mu_j)^2}{2\sigma_j^2}\right] \tag{28}$$

$$\sum_{j=1}^{L} \varepsilon_j = 1. \tag{29}$$

The second equation represents the normalization relation for the a priori probabilities. We assume that each distribution in the mixture (28) corresponds to one data class. Our aim is to separate each class j by choosing appropriate thresholds T_j, T_{j+1}. An example of three overlapping Gaussian distributions separated by thresholds T_1 and T_2 is provided in Figure 2. When estimating the components of (28) we have to consider the overlapping among different distributions. The normalized distribution is given by :

$$g(X) = \frac{f(X)}{\int_{T_j}^{T_{j+1}} f(X)} \tag{30}$$

where T_j and T_{j+1} are the optimal boundaries of the j-th function with its neighboring functions.

The expected value of the center can be obtained from :

$$E[\hat{\mu}_j] = E[X|X \in [\hat{T}_j, \hat{T}_{j+1})] = \frac{\int_{\hat{T}_j}^{\hat{T}_{j+1}} X f(X) dX}{\int_{\hat{T}_j}^{\hat{T}_{j+1}} f(X) dX} \tag{31}$$

where $\hat{T}_j$ and $\hat{T}_{j+1}$ are the estimates of the separating boundaries for the j-th Gaussian kernel and $f(X)$ is given by (28).

In an estimation problem, the bias represents the difference between the estimated value and the optimum one [22]. It is desirable to obtain as small bias as possible. The bias of the boundary between two classes is directly related to the estimation of the class probabilities (6). If the class probabilities are well estimated, then the bias is small. If not, then the bias is large.

When $\hat{\mu}_j$ is evaluated as in classical LVQ (11), the stationary estimate for the j-th Gaussian kernel center is given by :

$$\begin{aligned} E_C[\hat{\mu}_j] = \frac{1}{\delta} \sum_{i=1}^{L} \varepsilon_i \Bigg\{ \mu_i \left[\mathrm{erf}\left(\frac{\hat{T}_{j+1} - \mu_i}{\sigma_i} \right) - \mathrm{erf}\left(\frac{\hat{T}_j - \mu_i}{\sigma_i} \right) \right] + \\ + \frac{\sigma_i}{\sqrt{2\pi}} \left[\exp\left(- \frac{(\hat{T}_j - \mu_i)^2}{2\sigma_i^2} \right) - \exp\left(- \frac{(\hat{T}_{j+1} - \mu_i)^2}{2\sigma_i^2} \right) \right] \Bigg\} \end{aligned} \tag{32}$$

where δ is

$$\delta = \sum_{i=1}^{L} \varepsilon_i \left[\mathrm{erf}\left(\frac{\hat{T}_{j+1} - \mu_i}{\sigma_i} \right) - \mathrm{erf}\left(\frac{\hat{T}_j - \mu_i}{\sigma_i} \right) \right] \tag{33}$$

and the function erf(y) is [25] :

$$\mathrm{erf}(y) = \frac{1}{\sqrt{2\pi}} \int_0^y \exp\left(-\frac{t^2}{2} \right) dt. \tag{34}$$

When employing the median estimator (15), the *pdf* for $n = 2i + 1$ independent and identically distributed data is given by [23] :

$$f_{i+1}(X) = n \begin{pmatrix} n-1 \\ i \end{pmatrix} F^i(X)[1 - F(X)]^i f(X) \tag{35}$$

where $F(X)$ is the cumulative distribution function for the data whose *pdf* is (28). If we insert (35) in (31) we obtain the expected value of the median estimator assuming n data samples.

The median is located where the *pdf* of the given data samples splits in two equal areas [23]

$$\int_{\hat{T}_j}^{E_R[\hat{\mu}_j]} f(X) dX = \int_{E_R[\hat{\mu}_j]}^{\hat{T}_{j+1}} f(X) dX \tag{36}$$

where $E_R[\hat{\mu}_j]$ is the median stationary estimate of the given data. The stationary value of j-th distribution center estimate using the median estimator is obtained after inserting (28) in (36) :

$$\sum_{i=1}^{L} \varepsilon_i \operatorname{erf}\left(\frac{E_R[\hat{\mu}_j] - \mu_i}{\sigma_i}\right) = \frac{\delta}{2} \tag{37}$$

where δ is provided in (33).

In the case where we want to find the expectation for the variance estimator $\hat{\sigma}_j^2$ we use a similar approach. The expected stationary estimate for $\hat{\sigma}_j^2$ based on classical estimation (13) is :

$$E_C[\hat{\sigma}_j^2] = E[(X - \hat{\mu}_j)^2 | x \in [\hat{T}_j, \hat{T}_{j+1})] = \frac{\int_{\hat{T}_j}^{\hat{T}_{j+1}} (X - E_C[\hat{\mu}_j])^2 f(X) dX}{\int_{\hat{T}_j}^{\hat{T}_{j+1}} f(X) dX} \tag{38}$$

where $f(X)$ is given by (28) and $E_C[\hat{\mu}_j]$ is evaluated in (32).

When MAD is used as dispersion estimator (17), its expected value when assigning n data samples is :

$$E_{R,n}[\hat{\sigma}_j] = \frac{\int_{\hat{T}_j}^{\hat{T}_{j+1}} \mid X - E_{R,n}[\hat{\mu}_j] \mid f_{i+1}(X) dX}{c \int_{\hat{T}_j}^{\hat{T}_{j+1}} f_{i+1}(X) dX} \tag{39}$$

where $f_{i+1}(X)$ is given in (35), $E_{R,n}[\hat{\mu}_j]$ is evaluated after inserting (35) in (31) and $c = 0.6745$.

By taking into account the median property of splitting the data distribution into two equal areas as in (36), we obtain the expected stationary state of the MAD estimator, denoted as $E_R[\hat{\sigma}_j]$:

$$\int_{E_R[\hat{\mu}_j] - cE_R[\hat{\sigma}_j]}^{E_R[\hat{\mu}_j] + cE_R[\hat{\sigma}_j]} f(X) dX = \frac{1}{2} \int_{\hat{T}_j}^{\hat{T}_{j+1}} f(X) dX \tag{40}$$

where $E_R[\hat{\mu}_j]$ can be calculated from (37). $E_R[\hat{\sigma}_j]$ can be derived from :

$$\sum_{i=1}^{L} \varepsilon_i \left[\operatorname{erf}\left(\frac{E_R[\hat{\mu}_j] - \mu_i + cE_R[\hat{\sigma}_j]}{\sigma_i}\right) - \operatorname{erf}\left(\frac{E_R[\hat{\mu}_j] - \mu_i - cE_R[\hat{\sigma}_j]}{\sigma_i}\right)\right] = \frac{\delta}{2} \tag{41}$$

where δ is (33).

In order to evaluate the parameters for the Gaussian kernel we must also estimate the activation domains $[\hat{T}_j, \hat{T}_{j+1})$ for each Gaussian function. If Euclidean distance is used for deciding the corresponding activation region for a new data sample (10), we estimate the boundary $\hat{T}_j$ between two activation regions j and $j+1$ as :

$$\hat{T}_j = \frac{\hat{\mu}_j + \hat{\mu}_{j+1}}{2} \tag{42}$$

for $j = 1, \ldots, L-1$. When the Euclidean distance is replaced with the Mahalanobis distance (14), the boundary condition can be found by solving the equations :

$$\left(\frac{\hat{T}_j - \hat{\mu}_j}{\hat{\sigma}_j}\right)^2 = \left(\frac{\hat{T}_j - \hat{\mu}_{j+1}}{\hat{\sigma}_{j+1}}\right)^2 \tag{43}$$

for $j = 1, \ldots, L-1$. The first and last boundaries are defined as $T_0 = -\infty$ and $T_L = \infty$. The $2L-2$ parameters (Gaussian centers and boundaries) for the case described by (42) and $3L-2$ parameters (including the variances) for (43) have to be evaluated. In order to do this, analytical methods can be employed by calculating iteratively the centers of the Gaussian functions and the class boundaries.

From the condition (6), the relationship which gives the optimal boundary T_1 between two classes, each of them modeled by a Gaussian *pdf*, can be derived as :

$$\left(\frac{T_1-\mu_1}{\sigma_1}\right)^2 - \left(\frac{T_1-\mu_2}{\sigma_2}\right)^2 = 2\ln\frac{\sigma_2\varepsilon_1}{\sigma_1\varepsilon_2}. \tag{44}$$

Two Gaussian *pdf* functions are shown in Figure 3: $p_1(X)$ with $\mu_1=3$, $\sigma_1=2$, $\varepsilon_1=0.7$ and $p_2(X)$ with $\mu_2=12$, $\sigma_2=4$, $\varepsilon_2=0.3$. The optimal boundary (44) is compared with the boundary obtained based on the marginal median estimator and with the boundary given by classical LVQ. The bias $|\,T_1-\hat{T}_1\,|$ provided by the robust estimator is smaller than that of the classical LVQ.

We make the assumption that the a priori probabilities and the variances of the Gaussian functions are equal to each other. In the two-distribution estimation case, the expected values of the median estimator for the two centers can be derived from (37)

$$\mathrm{erf}\left(\frac{E_R[\hat{\mu}_1]-\mu_1}{\sigma}\right) + \mathrm{erf}\left(\frac{E_R[\hat{\mu}_1]-\mu_2}{\sigma}\right) = \frac{1}{2}, \tag{45}$$

$$\mathrm{erf}\left(\frac{E_R[\hat{\mu}_2]-\mu_1}{\sigma}\right) + \mathrm{erf}\left(\frac{E_R[\hat{\mu}_2]-\mu_2}{\sigma}\right) = -\frac{1}{2}. \tag{46}$$

In the case where estimating the dispersion based on the MAD estimator for two overlapping Gaussian distributions we obtain from (41) :

$$\begin{aligned}\mathrm{erf}\left(\frac{E_R[\hat{\mu}_1]-\mu_1+cE_R[\hat{\sigma}]}{\sigma}\right) - \mathrm{erf}\left(\frac{E_R[\hat{\mu}_1]-\mu_1-cE_R[\hat{\sigma}]}{\sigma}\right) + \\ +\mathrm{erf}\left(\frac{E_R[\hat{\mu}_1]-\mu_2+cE_R[\hat{\sigma}]}{\sigma}\right) - \mathrm{erf}\left(\frac{E_R[\hat{\mu}_1]-\mu_2-cE_R[\hat{\sigma}]}{\sigma}\right) = \frac{1}{2}.\end{aligned} \tag{47}$$

The particular examples considered here are :

$$f(X) = \frac{1}{2}N(5,\sigma) + \frac{1}{2}N(10,\sigma), \tag{48}$$

$$f(X) = \frac{1}{3}N(3,\sigma) + \frac{1}{3}N(5,\sigma) + \frac{1}{3}N(10,\sigma), \tag{49}$$

where $N(\mu,\sigma)$ denotes a Gaussian distribution.

The convergence is the property of a neural network to achieve a stable state after a finite number of iterations. The convergence can be defined individually for a weight or globally, expressing the state of the network by a cost function. In the following we analyze the capacity of various weights to achieve a stable state. In the example (48) let us assume $\sigma=2$. We use MRBF to estimate the parameters of the distribution $N(5,2)$. We find the expectation for the median by replacing the formula (35) in (31) and computing the integral numerically. In Figure 4 (a) we compare the expected bias for the marginal median against the bias of the stationary estimate of the mean, when estimating the Gaussian center. In Figure 4 (b) we provide a comparison between the expected bias of the MAD estimator (41) and that of the stationary estimate of the classical estimator for scale, which can be derived from (38). The expectation for scale parameter using the MAD estimator is obtained after evaluating (39). From these plots it is clear that median and MAD algorithms provide better parameter estimation than arithmetic mean and classical sample dispersion estimator, in the case when estimating overlapping normal functions.

We estimate the center and the scale parameter for the distribution $N(5, \sigma)$ using both classical and robust type learning in the case of the distribution mixtures (48,49). The class which corresponds to $N(5, \sigma)$ is bounded in the case of (49) and unbounded to the left in the case of (48). Thus, in the case of (48), the data samples used for training are drawn from a "medium-tailed" distribution and in the case of (49) from a "short-tailed" one. The stationary state of the bias $E[\hat{\mu}] - \mu$ is depicted in Figure 5 (a) for the distribution (48) and in Figure 5 (b) for the distribution (49), both with respect to the assumed dispersion (scale parameter) σ. The comparison results for estimating the stationary state of the bias for the scale parameter $E[\hat{\sigma}] - \sigma$, are given in Figure 5 (c) and in Figure 5 (d). From these plots it is evident that if certain overlaps occur among various Gaussian functions from the mixture, the respective amount of data samples contains outliers, while median and MAD estimators provide lesser bias than mean and classical sample deviation estimators. If the Gaussian functions are far away from each other with respect to their dispersions, the amount of outliers decreases and both algorithms provide similar results. However, if the isolated Gaussian functions are truncated, e.g., due to the decision (10), the robust estimators are more accurate than those based on classical statistics.

5 Simulation results

5.1 Estimating probability density functions

In the previous Section we have evaluated the theoretical performance in parameter estimation for both algorithms described in Section 3. In this Section we test these algorithms for the estimation of mixed bivariate normal and contaminated normal distributions.

The problem of finding the parameters for the Gaussian functions is seen as a supervised learning task. We consider four artificially generated distributions, each containing 2-D clusters. This problem can be considered as a 2-D extension of the mixture of Gaussians model analyzed in Section 4. A 2-D Gaussian distribution is denoted by $N(\mu_1, \mu_2; \sigma_1, \sigma_2)$. The Gaussian clusters are grouped in two classes in order to form more complex distributions :
Distribution I:

$$\begin{aligned} P_1^I(X) &= N(2,1;3,1) + N(8,7;3,1) \\ P_2^I(X) &= N(8,2;1,3) + N(2,6;1,3), \end{aligned} \tag{50}$$

Distribution II:

$$\begin{aligned} P_1^{II}(X) &= N(6,0;4,1) + N(0,6;1,4) \\ P_2^{II}(X) &= N(6,6;2,2). \end{aligned} \tag{51}$$

Two more distributions are obtained from the first two by adding uniform distributed data samples :
Distribution III

$$P_k^{III}(X) = \epsilon\, P_k^I + (1-\epsilon)\, U([-5,15],[-5,15]), \tag{52}$$

Distribution IV

$$P_k^{IV}(X) = \epsilon\, P_k^{II} + (1-\epsilon)\, U([-5,15],[-5,15]), \tag{53}$$

where $k \in \{1,2\}$ and $\epsilon = 0.9$. We denote by $U([-5,15],[-5,15])$ a uniform distribution having the domain $[-5,15] \times [-5,15]$.

For MRBF we consider a running window of W=401 samples when evaluating the median estimator according to (16). Both Euclidean and Mahalanobis distances were considered in order to decide which neuron to be updated for a new data sample. The same data were used for training both algorithms. We have tested the ability of classification for both networks after the learning stage was concluded. The misclassification error compares the true output $F_k(\mathbf{X})$ with the output $Y_k(\mathbf{X})$ given by the network and is represented as a fraction of the total number of samples. The second comparison criterion is the approximation of the *pdf* functions by the networks. The optimal network is obtained when the network weights are equal with the parameters of the Distribution I or II (50, 51). The mean square error calculated between the ideal function and the estimated one is defined as:

$$MSE = \frac{1}{M}\sum_{k=1}^{M} \int (p_k(X) - \hat{p}_k(X))^2 \, dX \tag{54}$$

where $\hat{p}_k(X)$ is the hypersurface modeled by the k-th output unit. This consists of a global performance estimation measure.

The comparison results provided by the networks are provided in Table 1. Each experiment was repeated many times for different data, drawn from the same distributions. Patterns from the first two distributions are represented in Figures 6 and 7. The two figures display also the boundaries found by means of neural networks as well as the optimal boundaries. The same number of hidden units are assumed for each network. From these figures it is evident that MRBF approximates better the boundaries between the classes than RBF based on classical statistics estimators. The advantage is clear for MRBF in all the cases considered in Table 1. However when the mixture of bivariate normal distributions is contaminated with uniform distributed patterns (e.g., in Distributions III and IV) the difference becomes very large because robust type learning is insensitive at extreme observations. By using the Mahalanobis distance (14) instead of the Euclidean one we obtain better results for both algorithms, excepting for the case when we use the classical estimators for the uniform contaminated model (52,53). In this case, because of the noise corruption, the estimation of the covariance matrices is poor. The MRBF algorithm based on the Mahalanobis data assignment rule gives the best results in all the assumed cases, as it can be seen from Table 1.

In Figure 8 we evaluate the global convergence of the algorithms in the case when the data are drawn from Distribution I. The learning curves represent the estimation of the *pdf* functions given by MSE (54), with respect to the number of drawn samples. From this plot it is clear that MRBF network provides a smaller MSE when compared to the classical RBF network. The improvement produced when using Mahalanobis distance is evident from this plot as well.

5.2 Optical flow segmentation

Motion representation and modeling is an important step towards dynamic image understanding. The optical flow field consists of the distribution of the velocities associated with the image element displacement. A variety of motion estimation techniques exists [35]. Block matching motion estimation techniques are widely used in video coding. A block matching algorithm assumes that the object displacement is constant within a small block of pixel elements. The block displacement is estimated by using correlation or matching techniques. The 2-D vector is chosen such that it minimizes the mean absolute error or mean square error between two blocks from a certain neighborhood from two different frames [35]. The best results are obtained when a full search is employed. This method takes into consideration all the possible pixel blocks within a region around the original block. The search region is chosen according to the expected maximal speed in the sequence. By employing block matching techniques, good results can be obtained in the regions having many details. However, in regions with almost constant pixel intensity, this algorithm usually gives a certain number of erroneous decisions.

Optical flow segmentation algorithms identify the regions having similar motion vectors. Various algorithms based on clustering were proposed to be used in optical flow segmentation [36, 37]. When applying RBF networks for optical flow segmentation, the centers of the hidden units represent groups of motion vectors [38]. Each set of vectors corresponds to an object or to a part from an object, moving with a certain velocity.

We have applied the algorithms presented in Section 3 in the "Hamburg taxi" sequence. The first and third frames of this sequence are shown in Figures 9 (a) and (b). Their frame size is 256×190 and they contain three important moving objects: a taxi turning around the corner, a car in the lower left moving from left to right and a van in the lower right moving from right to left. In the first processing stage we have estimated the optical flow by using the full search block matching algorithm, when assuming blocks of 4×4 pixels. The block matching search region is taken $[-8, 8] \times [-8, 8]$ pixels wide. The motion field provided by the block matching is shown in Figure 10 (a). The optical flow histogram is represented in Figure 11 (a). The four moving objects (including the background) can be easily identified as histogram peaks (a concentration of motion vectors with similar velocity) in Figure 11 (a).

We have employed both RBF and MRBF neural networks for optical flow segmentation. The input to hidden unit weights are calculated in an unsupervised manner as it was presented in Section 3. We have considered a second level clustering algorithm for finding the output weights $\lambda_{k,j}$. Each cluster of motion vectors which activates a hidden unit is assigned to an output unit based on the Euclidean distance between each two hidden unit centers.

We have evaluated the performance of the algorithms in terms of mean square error (MSE) as well as mean absolute error (MAE) of the optical flow with respect to the center selection

$$MSE = \sum_{j=1}^{M} \left(\frac{1}{n_j} \sum_{i=1}^{n_j} \|\mathbf{X}_i - \mu_j\|^2 \right), \tag{55}$$

$$MAE = \sum_{j=1}^{M} \left(\frac{1}{n_j} \sum_{i=1}^{n_j} \sum_{h=1}^{N} |X_{i,h} - \mu_{j,h}| \right), \tag{56}$$

where $\mathbf{X}_i$, $i = 1, \ldots, n_j$ are data samples assigned to the j-th unit. Image sequence processing needs fast algorithms in most applications. Thus, we have implemented the histogram-based algorithm for MRBF as presented in Section 3.3. Both algorithms were tested in the same conditions. The hidden units which have assigned only a very small amount of motion vectors, are pruned out. Only one pass through data is enough in order to achieve a good motion segmentation. The comparison results in terms of MAE, MSE and training time are shown in Table 2. The time for the first layer updating corresponds to the calculation of the hidden unit weights. The total time includes also the output weights calculation. All these times correspond to an implementation on a Silicon Graphics Indigo Workstation. The implementation algorithm proposed in Section 3.3 for MRBF parameter evaluation proved to be fast.

We have also evaluated the speed (in pixels/frames) for the corresponding moving objects. The optimal velocity was calculated as the average of the clear feature displacements from each moving object, obtained independently, in a semiautomatic way. The comparison results between the real speed of the objects and the speed obtained by means of the RBF and MRBF algorithms are given in Table 3. The velocity vectors found in the "Hamburg taxi" sequence by MRBF and RBF algorithms are displayed in Figure 10 (b) and Figure 10 (c). The smoothing obtained after the optical flow processing by using either MRBF or RBF algorithm is clear from these figures. A more complex criterion taking into account the block average graylevel and the position of the blocks would provide better moving object segmentation results [38].

The histograms representing the optical flow modeled by means of MRBF and RBF networks as $\sum_{j=1}^{L} |\lambda_{k,j}| \phi_j(\mathbf{X})$ are shown in Figure 11 (b) and Figure 11 (c), respectively. Comparing these histograms to the initial optical flow histogram from Figure 11 (a) we can observe that MRBF network approximates it better than RBF network using classical statistics based training. From Figure 10 (c) and Figure 11 (c), as well as from Table 3 we can see that the algorithm based on classical training was not able to identify correctly the movement of the "taxi" moving object. According to these experiments, the proposed learning algorithm MRBF, provides better estimation for the desired parameters when compared to the classical statistics based training for the RBF network.

6 Conclusions

In this paper we present a novel algorithm for estimating the RBF weights based on robust estimation and called Median RBF (MRBF). This algorithm is presented in comparison with a classical approach for training an RBF network. We have employed the marginal median estimator for evaluating the basis function centers and the median of the absolute deviations for estimating the dispersion parameters. We propose an implementation for MRBF network based on data histogram updating which proved to be fast. We provide the theoretical evaluation of the bias for both algorithms in the case when estimating overlapping Gaussian distributions. The MRBF-based training is less biased by the presence of the outliers in the training set and was proved to provide an accurate estimation of the implied probabilities. Both RBF and MRBF algorithms were compared in *pdf* estimation of artificially generated data as well as in motion segmentation of a real-life image sequence. In both cases, the MRBF gave better estimation of the implied *pdf*s and has shown better classification capabilities.

References

[1] T. Poggio, F. Girosi, "Networks for approximation and learning," *Proc. of the IEEE,* vol. 78, no. 9, pp. 1481-1497, Sep. 1990.

[2] E. J. Hartman, J. D. Keeler, J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Computation,* vol. 2, pp. 210-215, 1990.

[3] J. Park, J. W. Sandberg, "Universal approximation using radial basis functions network," *Neural Computation,* vol. 3, pp. 246-257, 1991.

[4] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.,* vol. 33, pp. 1065-1076, 1962.

[5] R. M. Sanner, J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. on Neural Networks,* vol. 3, no. 6, pp. 837-863, Nov. 1992.

[6] D. S. Broomhead, D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems,* vol. 2, pp. 321-355, 1988.

[7] J. Moody, C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation,* vol. 1, no. 2, pp. 281-294, 1989.

[8] R. J. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches.* New York: John Wiley, 1992.

[9] T. K. Kohonen, *Self-organization and associative memory.* Berlin, Heidelberg, Germany: Springer-Verlag, 1989.

[10] D. F. Specht, "Probabilistic neural networks and the polynomial adaline as complementary techniques for classification," *IEEE Trans. on Neural Networks,* vol. 1, no.1, Mar. 1990.

[11] D. F. Specht, "A general regression neural network," *IEEE Trans. on Neural Networks,* vol. 2, no. 6, pp. 568-576, Nov. 1991.

[12] H. G. C. Tråvén, "A neural network approach to statistical pattern classification by semiparametric estimation of probability density functions," *IEEE Trans. on Neural Networks,* vol. 2, no. 3, pp. 366-377, May 1991.

[13] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, D. M. Hummels, "On the training of radial basis functions classifiers," *Neural Networks,* vol. 5, pp. 595-603, 1992.

[14] A. G. Borş, M. Gabbouj, "Minimal topology for a radial basis functions neural network for pattern classification," *Digital Signal Processing:* A review journal, vol. 4, no. 3, pp. 173-188, July 1994.

[15] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation,* vol. 3, no. 2, pp. 213-225, 1991.

[16] S. Chen, B. Mulgrew, P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. on Neural Networks,* vol. 4, no. 4, pp. 570-579, July 1993.

[17] S. Lee, R. M. Kil, "A Gaussian potential function network with hierarchically self-organizing learning," *Neural Networks,* vol. 4, pp. 207-224, 1991.

[18] S. Chen, C. F. N. Cowan and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis functions networks," *IEEE Tran. on Neural Networks,* vol. 2, no. 2, pp. 302-309, Mar. 1991.

[19] S. Chen, B. Mulgrew, "Overcoming co-channel interference using an adaptive radial basis functions equalizer," *Signal Processing,* vol. 28, pp. 91-107, July 1992.

[20] T. K. Kohonen, "The self-organizing map," *Proc. of the IEEE,* vol. 78, no. 9, pp. 1464-1480, Sep. 1990.

[21] L. Xu, A. Krzyżak, E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. on Neural Networks,* vol. 4, no. 4, pp. 636-649, July 1993.

[22] G. Seber, *Multivariate Observations.* New York: John Wiley, 1986.

[23] I. Pitas, A. N. Venetsanopoulos, *Nonlinear Digital Filters: principles and applications.* Norwell, MA: Kluwer Academic, 1990.

[24] I. Pitas, P. Kiniklis, "Median learning vector quantizer," *Proc. SPIE, vol. 2180, Nonlinear Image Processing,* San Jose, CA, pp. 23-34, 7-9 Feb. 1994.

[25] A. Papoulis, *Probability, Random Variables and Stochastic Processes.* New York: McGraw-Hill, 1984.

[26] A. Gersho, R. M. Gray, *Vector quantization and signal compression.* Norwell, MA: Kluwer Academic, 1992.

[27] B. Widrow, S.D. Stearns, *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1985.

[28] J. A. Kangas, T. K. Kohonen, J. T. Laaksonen, "Variants of self-organizing maps," *IEEE Trans. on Neural Networks,* vol. 1, no. 1, pp. 93-99, Mar. 1990.

[29] E. Yair, K. Zeger, A. Gersho, "Competitive learning and soft competition for vector quantizer design," *IEEE Trans. on Signal Processing,* vol. 40, no. 2, pp 294-309, Feb. 1992.

[30] A. G. Borş, I. Pitas, "Robust estimation of radial basis functions," *Proc. of IEEE Workshop on Neural Networks for Signal Processing,* Ermioni, Greece, pp. 105-114, 6-8 Sep. 1994.

[31] I. Pitas, C. Kotropoulos, N. Nikolaidis, R. Yang, M. Gabbouj, "A class of order statistics Learning Vector Quantizers," *Proc. of IEEE Int. Symposium on Circuits and Systems,* London, pp. VI-387 - VI-390, 1994.

[32] D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing.* Cambridge: MIT Press, 1986.

[33] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to theory of neural computation.* New York: Addison-Wesley, 1991.

[34] T. S. Huang, G. J. Yang, G. Y. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. on Acoustics, Speech and Signal Processing,* vol. 27, no. 1, pp 13-18, 1979.

[35] A. N. Netravali, B. G. Haskell, *Digital Pictures: representation and compression.* New York: Plenum Press, 1988.

[36] M. M. Chang, A. M. Tekalp, M. I. Sezan, "Motion-field segmentation using an adaptive MAP criterion," *Proc. of IEEE Int. Conf. Acoust. Speech and Signal Proc.,* Minneapolis, MN, pp. V-33 - V-36, Apr. 1993.

[37] D. P. Kottke, Y. Sun, "Motion estimation via clustering matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 16, no. 11, pp. 1128-1132, Nov. 1994.

[38] A. G. Borş, I. Pitas, "Median radial basis functions network for optical flow processing," *Proc. of IEEE Workshop on Nonlinear Signal and Image Processing,* Neos Marmaras, Greece, pp. 702-705, 20-22 June 1995.

Table 1: COMPARISON BETWEEN MRBF AND RBF NETWORKS IN PATTERN CLASSIFICATION AND DISTRIBUTION MODELING.

Distribution	Method	Distance Measures			
		Euclidean		Mahalanobis	
		Error (%)	MSE	Error (%)	MSE
I	RBF	21.26	13.69	17.17	6.90
	MRBF	17.58	8.65	13.75	2.75
	Optimal	12.13	0.00	12.13	0.00
II	RBF	3.89	3.69	2.95	1.24
	MRBF	2.90	1.20	2.61	0.82
	Optimal	2.52	0.00	2.52	0.00
III	RBF	26.63	34.22	35.05	48.59
	MRBF	21.11	10.11	18.82	5.74
	Optimal	15.78	0.00	15.78	0.00
IV	RBF	15.28	32.36	22.21	39.61
	MRBF	8.78	5.50	7.24	2.49
	Optimal	7.18	0.00	7.18	0.00

Table 2: OPTICAL FLOW SEGMENTATION RESULTS.

Network Topology N-L-M	MRBF				RBF			
	MAE	MSE	TIME (s)		MAE	MSE	TIME (s)	
			First layer	Total			First layer	Total
2-6-4	1.70	10.13	0.12	0.20	2.40	12.94	0.19	0.27
2-8-4	1.64	10.52	0.15	0.26	2.15	13.03	0.22	0.31
2-10-4	1.37	5.92	0.18	0.33	2.20	14.43	0.26	0.40
2-12-4	1.53	8.57	0.21	0.37	2.39	17.60	0.30	0.46

Table 3: EVALUATION OF THE SPEED (IN PIXELS/FRAMES) FOR VARIOUS MOVING OBJECTS.

Method	Network Topology	taxi		van		car		background	
		X	Y	X	Y	X	Y	X	Y
RBF	2-4-4	-5.08	-1.62	-5.08	-1.62	5.27	-0.39	-0.02	-0.05
	2-8-4	-5.25	-1.45	-5.25	-1.45	5.57	1.11	-0.08	-0.01
MRBF	2-4-4	-2	-1	-6	0	5	1	0	0
	2-8-4	-2	-1	-6	0	6	1	0	0
Optimal		-2	-1	-5	0	6	1	0	0

List of figure captions

Figure 1. The Radial Basis Functions neural network structure.

Figure 2. Overlapping Gaussian functions.

Figure 3. The separation boundary between two Gaussian probability density functions where '-' denotes the optimal boundary, '- -' the boundary found by using marginal median estimator and '- ·' the boundary found by using classical LVQ.

Figure 4. The convergence to the stationary estimate in 1-D data.
(a) first order statistics estimators;
(b) second order statistics estimators.

Figure 5. Theoretical analysis of the bias for median and classical statistics estimators in evaluating the RBF parameters:
(a) center for $N(5, \sigma)$ in the distribution (48) ;
(b) center for $N(5, \sigma)$ in the distribution (49) ;
(c) scale parameter for $N(5, \sigma)$ in (48) ;
(d) scale parameter for $N(5, \sigma)$ in (49) .

Figure 6. Samples from Distribution I. The boundaries between classes are marked with: '-' for optimal classifier, '- -' for MRBF and '- ·' for RBF.

Figure 7. Samples from Distribution II. The boundaries between classes are marked with: '-' for optimal classifier, '- -' for MRBF and '- ·' for RBF.

Figure 8. The learning curves in the case when the samples are drawn from Distribution I.

Figure 9. Frames from the "Hamburg taxi" sequence.
(a) the first frame ;
(b) the third frame .

Figure 10. The pindiagrams representing the optical flow corresponding to the movement between the first and third frames.
(a) the optical flow provided by the block matching algorithm ;
(b) the optical flow after it was smoothed by the MRBF network ;
(c) the optical flow after it was smoothed by the RBF network .

Figure 11. The optical flow histograms corresponding to the movement between the first and third frames. From left to right the peaks in histograms represent the van, the taxi, the background and the left car.
(a) the histogram of the optical flow when using the full search block matching algorithm ;
(b) the histogram represented by means of MRBF network ;
(c) the histogram represented by means of RBF network .

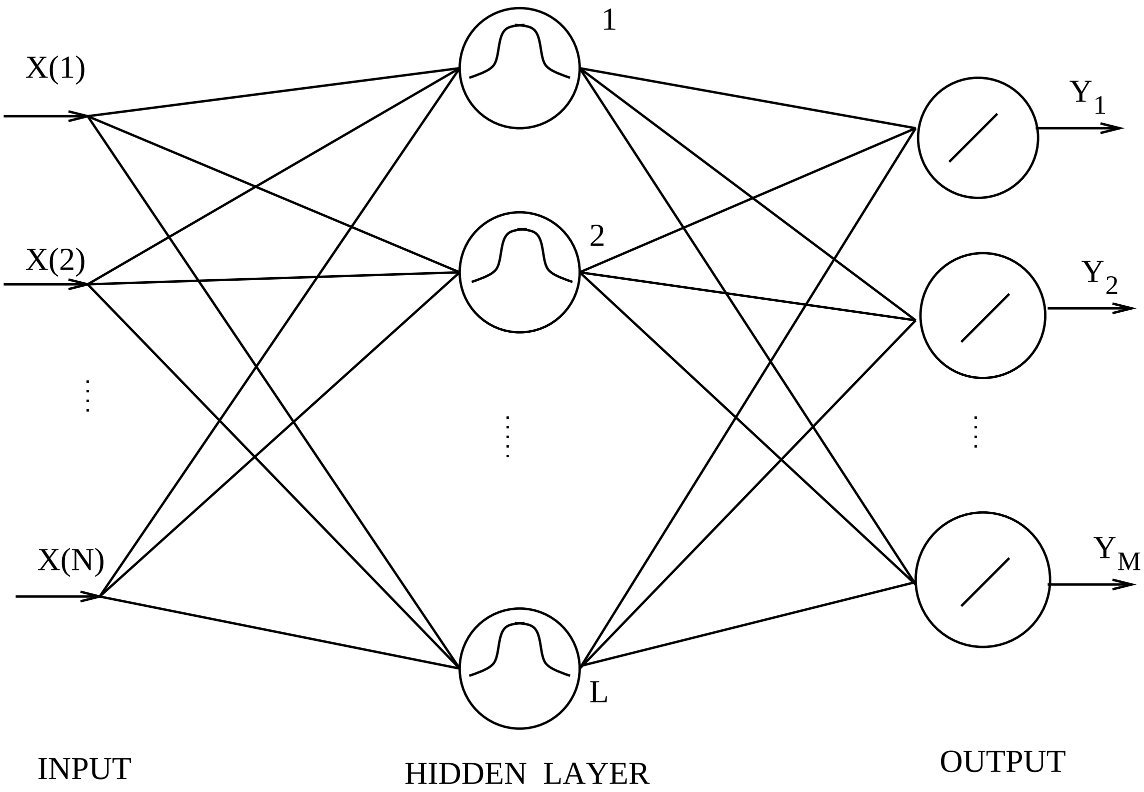


Figure 1:

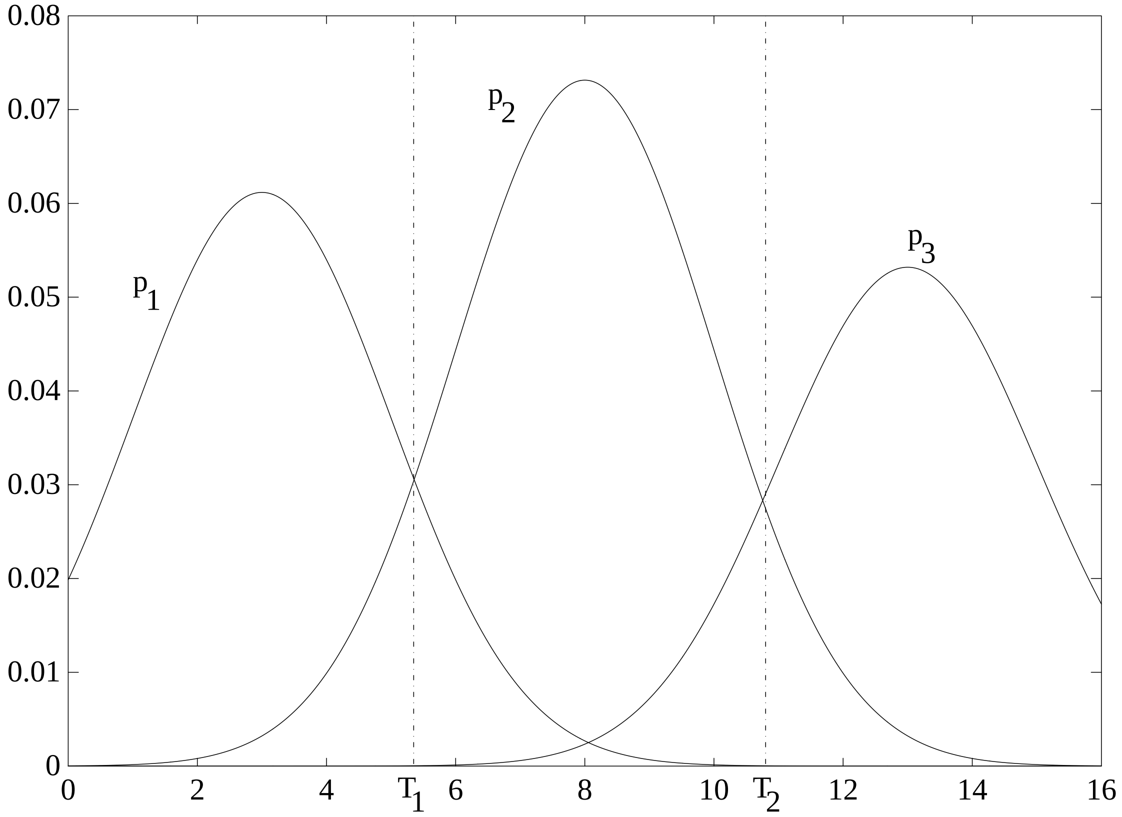


Figure 2:

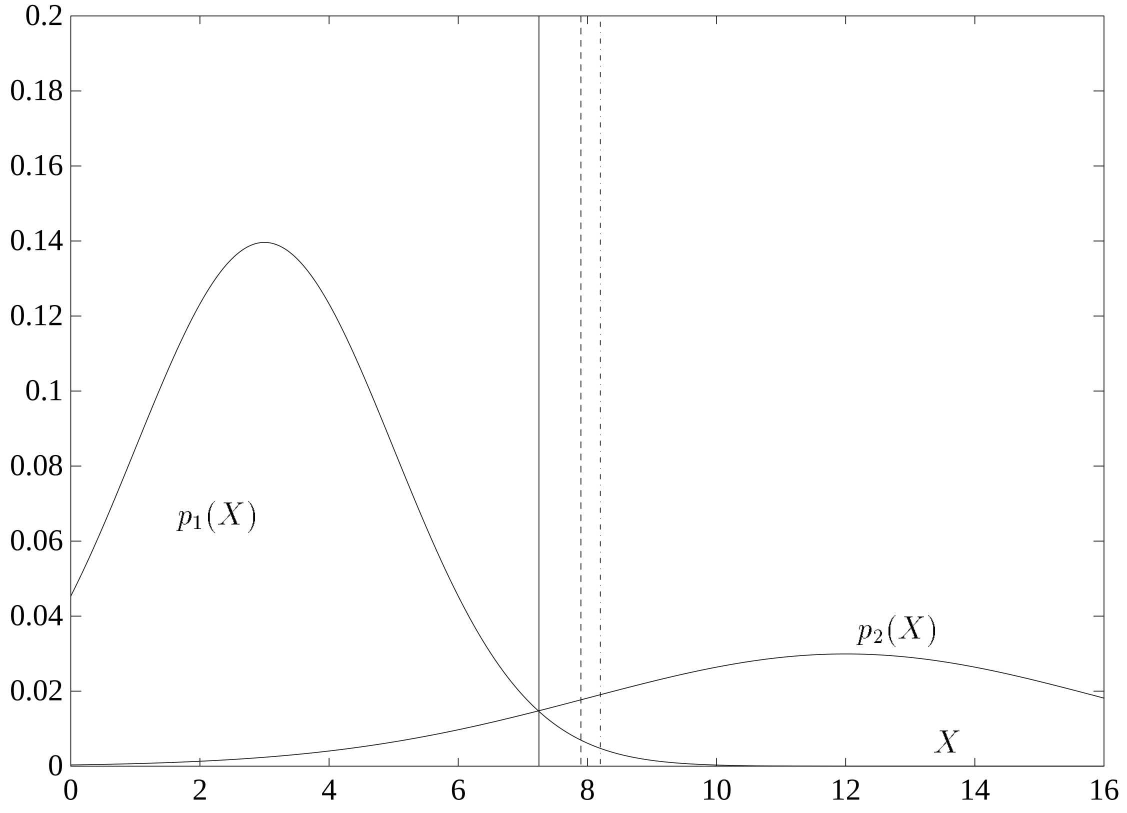


Figure 3:

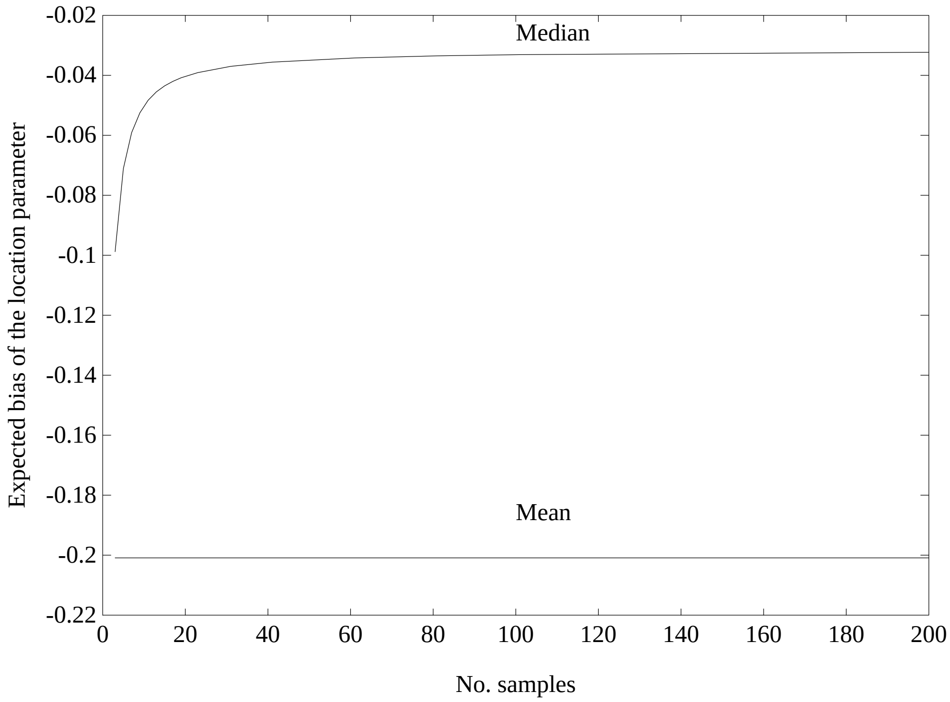


Figure 4 (a)

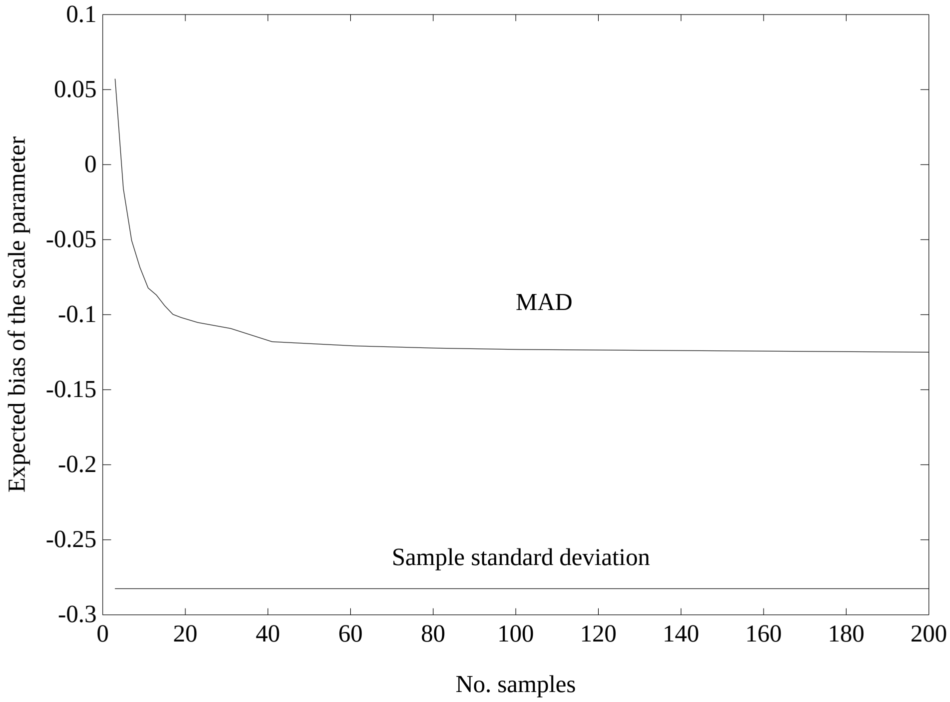
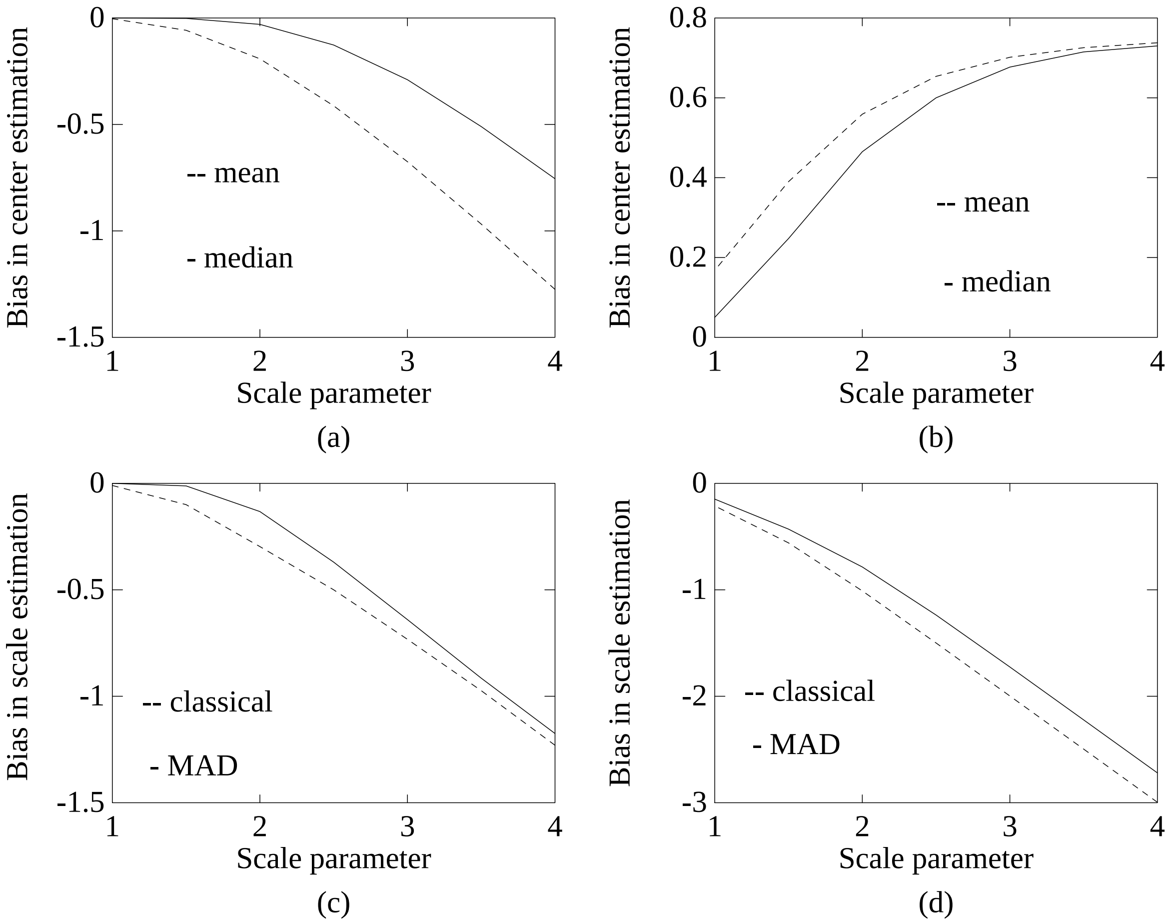


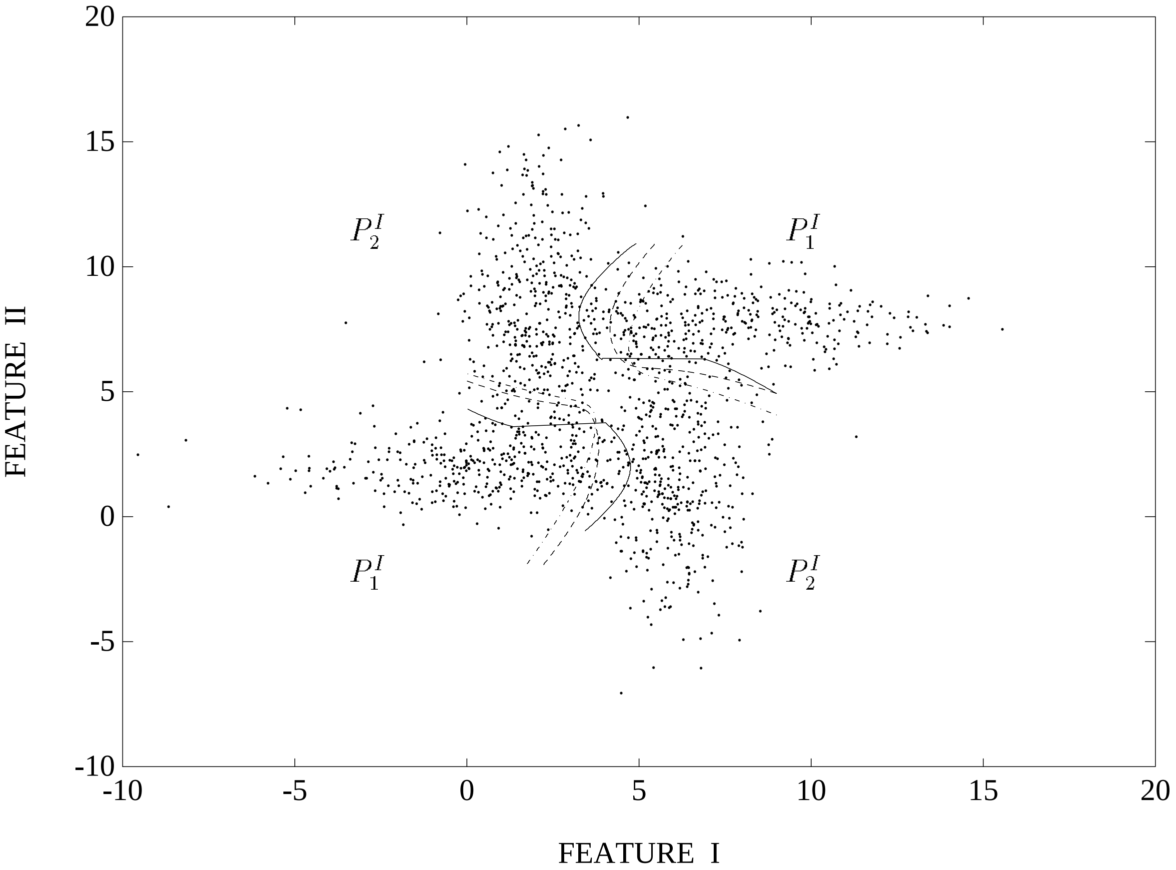
Figure 4 (b)

Figure 4:



0
-0.5
-1
-1.5
Bias in center estimation
-- mean
- median
1
2
3
4
Scale parameter
(a)
0.8
0.6
0.4
0.2
0
Bias in center estimation
-- mean
- median
1
2
3
4
Scale parameter
(b)
0
-0.5
-1
-1.5
Bias in scale estimation
-- classical
- MAD
1
2
3
4
Scale parameter
(c)
0
-1
-2
-3
Bias in scale estimation
-- classical
- MAD
1
2
3
4
Scale parameter
(d)

Figure 5:



20
15
10
5
0
-5
-10
-10
-5
0
5
10
15
20
FEATURE II
FEATURE I
P_2^I
P_1^I
P_1^I
P_2^I

Figure 6:

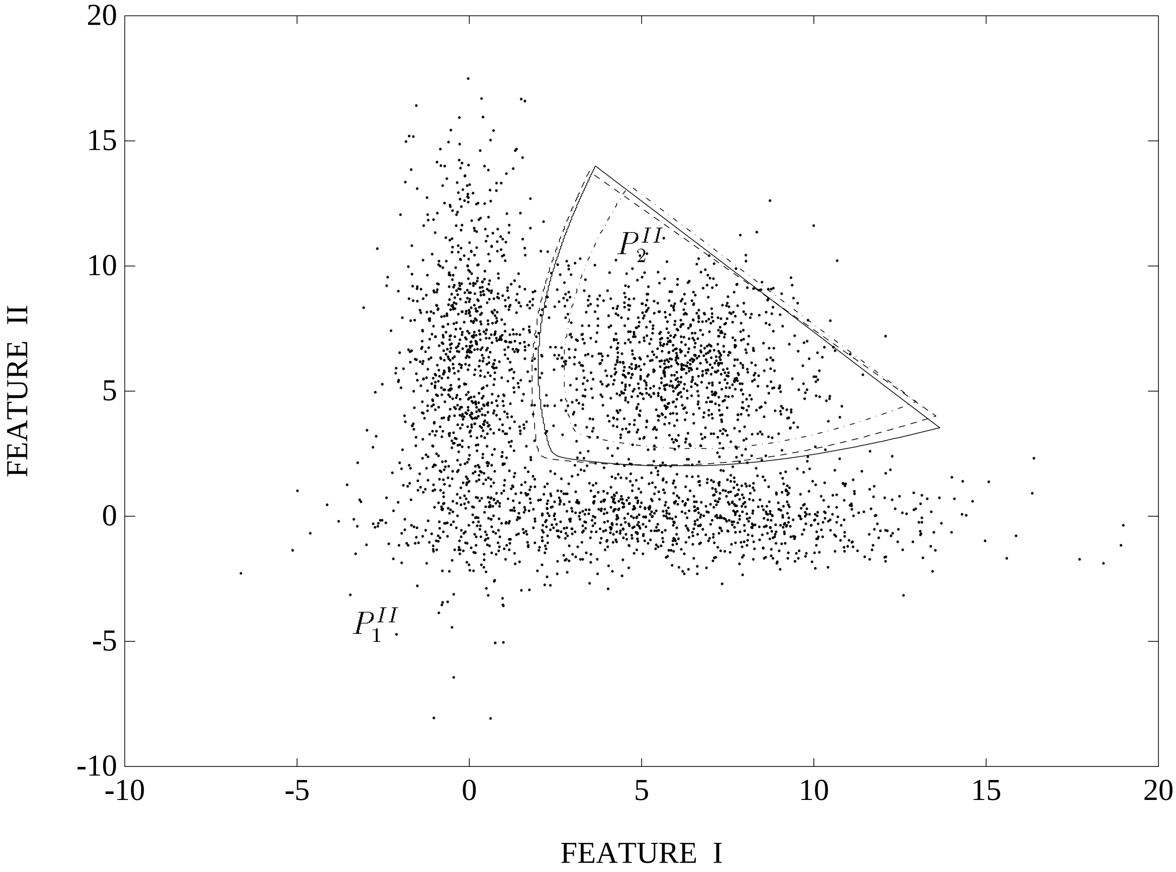
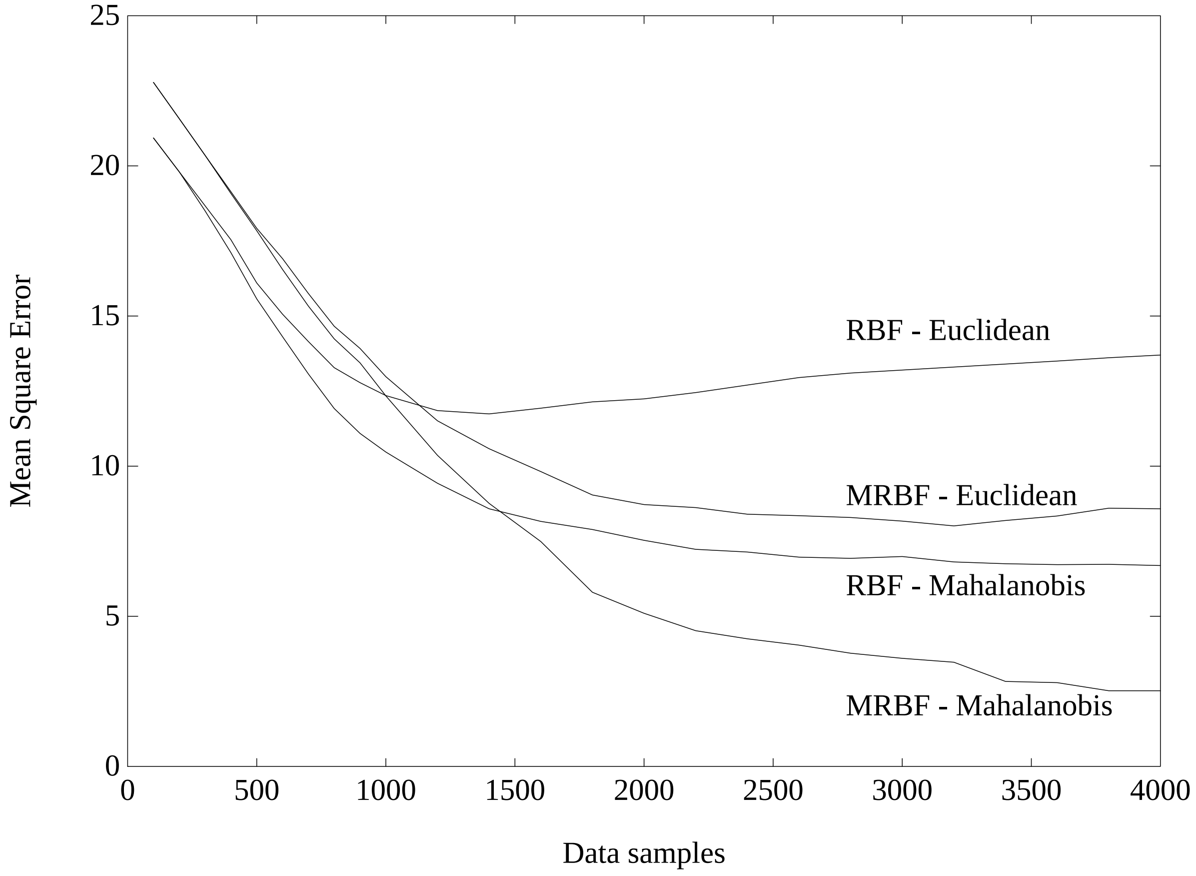


Figure 7:



RBF - Euclidean
MRBF - Euclidean
RBF - Mahalanobis
MRBF - Mahalanobis
Mean Square Error
Data samples
0
5
10
15
20
25
0
500
1000
1500
2000
2500
3000
3500
4000

Figure 8:

Figure 9 (a)

Figure 9 (b)

Figure 9:

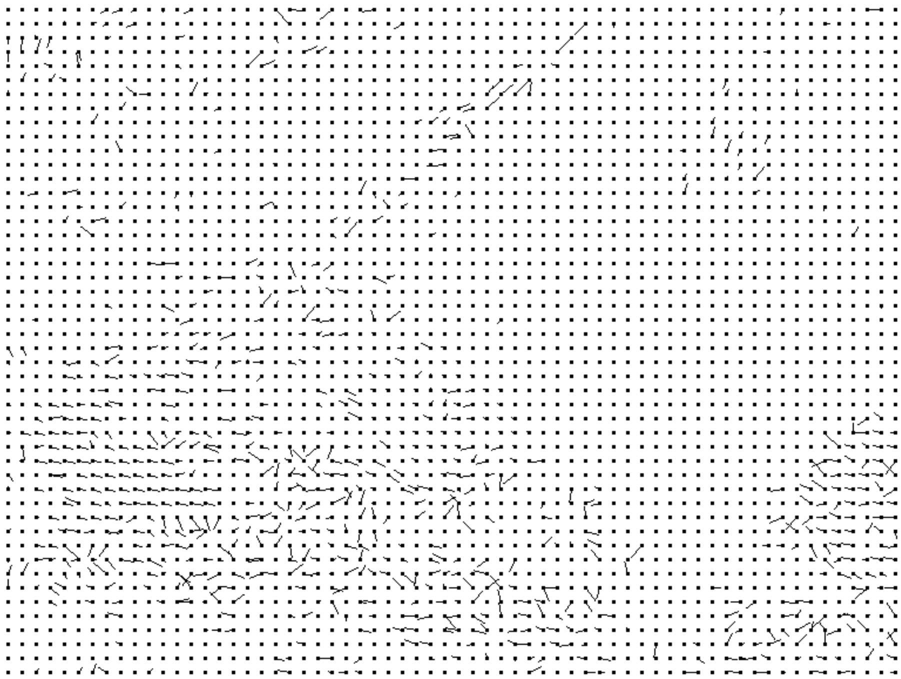


Figure 10 (a)

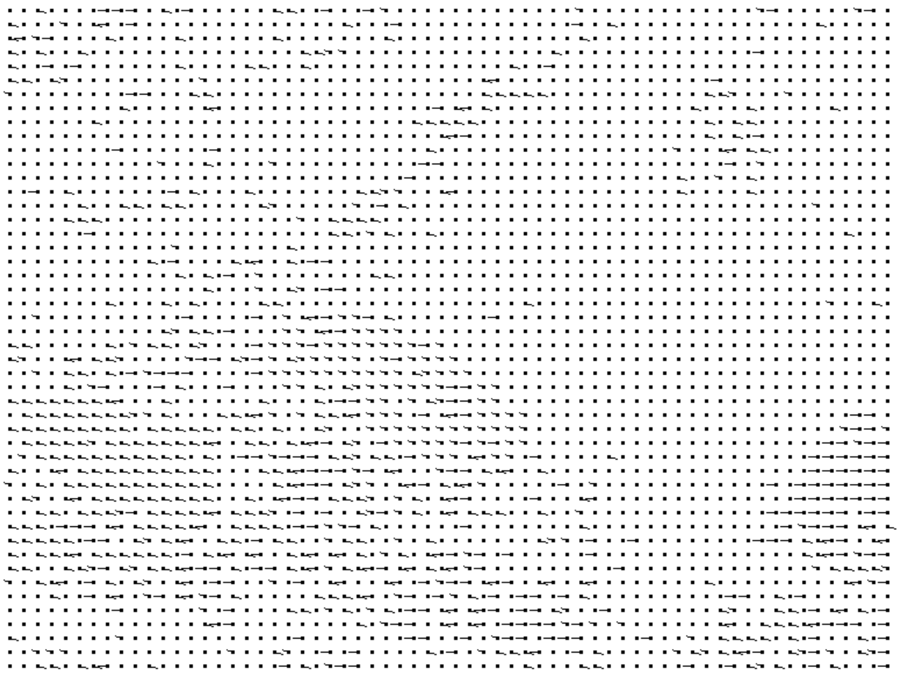


Figure 10 (b)

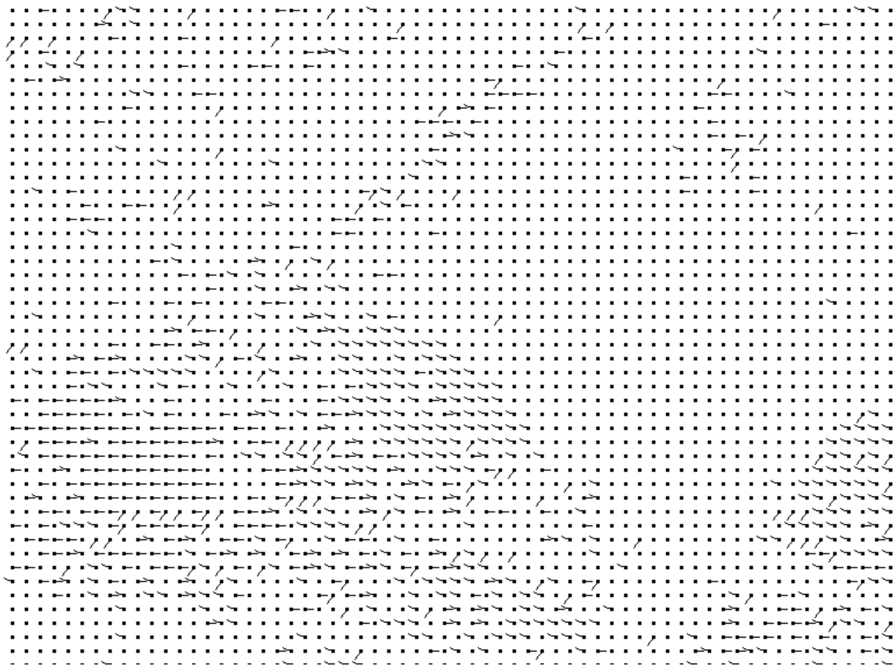


Figure 10 (c)

Figure 10:

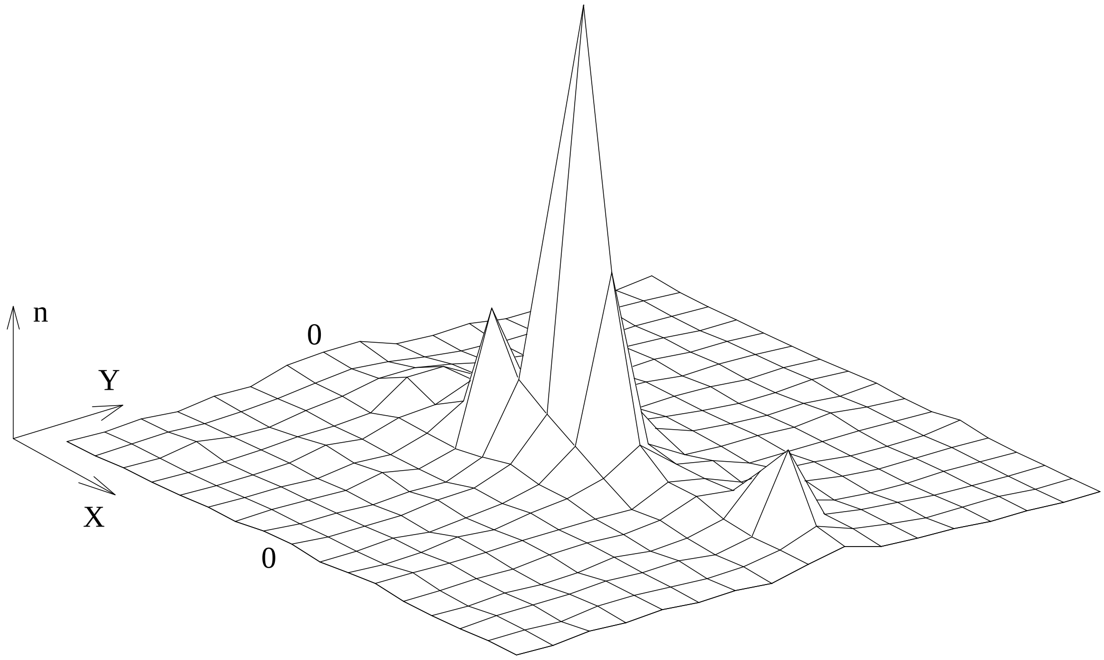


Figure 11 (a)

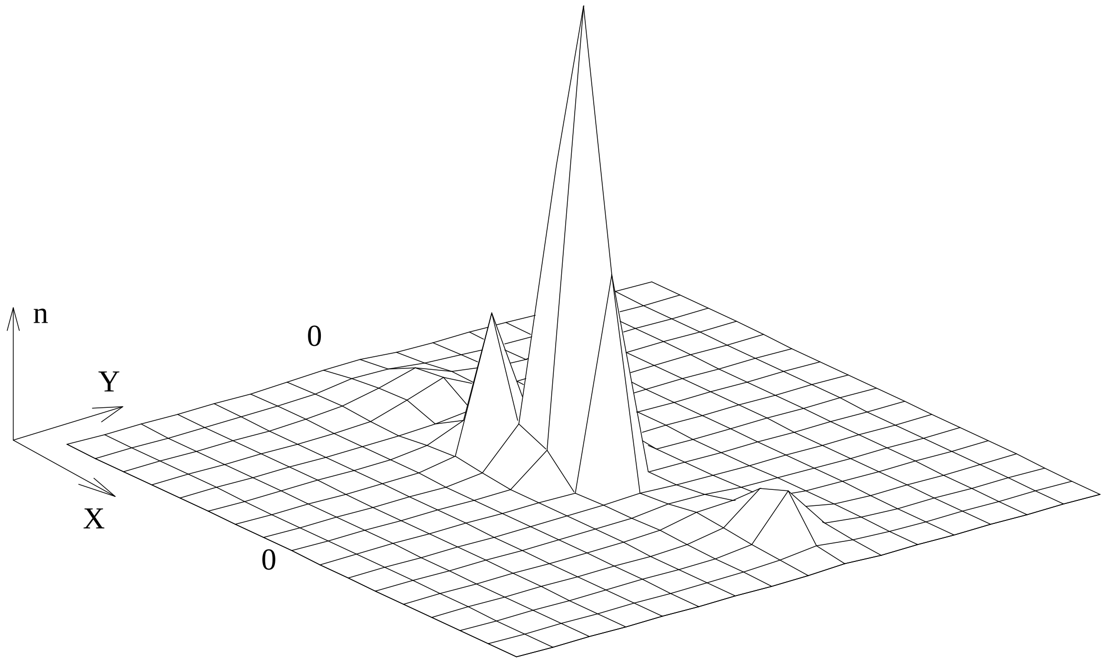


Figure 11 (b)

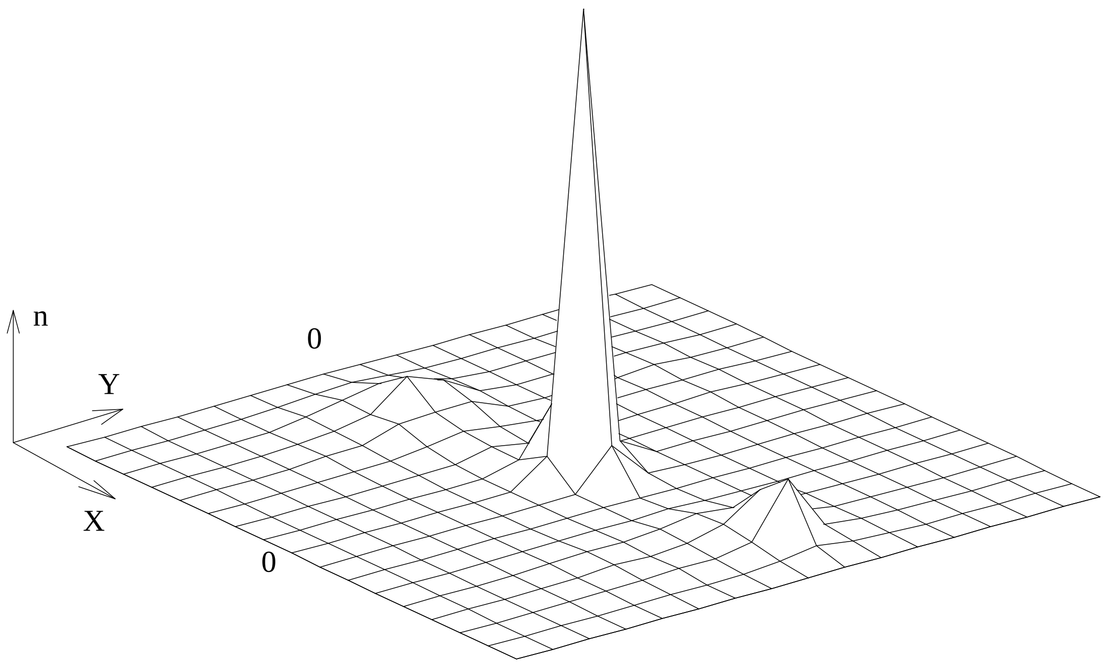


Figure 11 (c)

Figure 11: