Probability Density Estimation Using Multilayer Perceptrons

Aristidis Likas
Department of Computer Science
University of Ioannina
45110, Ioannina, Greece
e-mail: arly@cs.uoi.gr

Preprint no. 22-98/1998

Abstract

We present an approach for the estimation of probability density functions (pdf) given a set of observations that is based on the use of feedforward multilayer neural networks. The particular characteristic of the method is that the output of the network is not a pdf, therefore, the computation of the integral of the output over the input domain is required. Several modifications of the original approach [7] are proposed, most of them related to the numerical computation of the integral and the employment of a preprocessing phase where the network parameters are initialized using supervised training. Experimental results using several test problems indicate that the proposed method is numerically stable and very effective, providing in most cases more accurate estimations of the unknown densities compared to the method of gaussian mixtures.

1 Introduction

Probability density function (pdf) estimation of a given set of observations constitutes a fundamental problem in pattern recognition data analysis and more recently data mining. Many approaches have been developed that can be distinguished in parametric and nonparametric techniques. Nonparametric techniques are model free and, in general, make use of the whole set of observations to estimate the pdf at given point of the input space. Such techniques include histograms, Parzen estimators and nearest neighbor estimators. Disadvantages of such techniques are that they require the whole training set for each estimation and that they are very sensitive to the values of some crucial parameters (eg. the window width in Parzen techniques).

Parametric techniques assume that the unknown pdf follows a certain functional form (model) with parameters that must be adjusted in order for the model to approximate the distribution of the data. This adjustment is usually achieved through the maximization of the *likelihood* of the data with respect to the parameters of the model. The most widely used

parametric approach for density estimation is based on *gaussian mixtures*, which assumes that the model pdf is the weighted sum of gaussian distributions, with the model parameters being the parameters (mean and covariance matrix) of each gaussian and the coefficients of the weighted sum. The gaussian mixture model has been shown to exhibit the universal approximation property (ie as the number of gaussian components increases, it can approximate any pdf to arbitrary accuracy) and efficient procedures for likelihood maximization have been developed for this model (EM algorithm) [2, 11, 13]. It must be noted that, in analogy with the case of gaussian mixtures, most parametric approaches assume that the model function is a pdf by construction.

In [7] the use of multilayer Perceptrons (MLPs) as models for pdf estimation is proposed, along with a training procedure for adjusting the parameters of the MLP (weights and biases) so that the likelihood is maximized. Moreover, theoretical results are presented in [8], concerning the rate of convergence of such techniques.

The main difficulty with the employment of MLPs as a model for pdf estimation lies in the fact that *the output of an MLP is not a pdf*, since its integral over the whole input space does not sum to unity). Therefore, if $N(x) \geq 0$ is the output of the MLP for a given point x of the input space, the actual pdf $p_N(x)$ is given by: $p_N(x) = N(x)/\int N(x)dx$. The numerical computation of the integral raises several difficulties that are not explicitly addressed in [7], where no details are provided concerning numerical integration issues. In many cases numerical problems appear that lead to unacceptable solutions as will be shown in the paper.

In this work we elaborate on the use of MLPs as models for pdf estimation. The reason lies on the wish to exploit the widely acknowledged function approximation capabilities of MLPs and the possibility of fast implementation (either on parallel machines or on specialized neuroprocessors). The use of a model function that is not a pdf gives the method greater flexibility (as experiments indicate) in approximating unknown densities. For example, it is widely known that the gaussian mixture approach encounters difficulties in approximating the uniform distribution. On the other hand, the price paid by the MLP approach for the increased flexibility is the incorporation of numerical integral computations in the likelihood maximization procedure. We propose two basic mechanisms for making the method numerically stable and effective. The first is a preprocessing stage where supervised learning is used to obtain a coarse approximation of the unknown pdf and "constrain" the network output to the domain of interest. The second mechanism is the use of a moving grid for integral computations, that is necessary to ensure the accuracy of the integration and that the minimization procedure will not provide unacceptable solutions. Moreover, we have conducted

comparative experiments with the gaussian mixture technique (where the EM algorithm is used for training [2, 11]), which assure the flexibility of the MLP approach in approximating pdfs of arbitrary shape.

The remainder of the paper is organized as follows: Section 2 describes the basic approach to pdf estimation using MLPs, while Section 3 deals with issues of numerical integration. The proposed approach is described in detail in Section 4. Section 5 provides comparative experimental results on several test problems and, finally, Section 6 provides conclusions and future research directions.

2 The basic MLP approach to pdf estimation

The probability density function approximation capabilities of general multilayer feedforward neural networks have been established by White [14]. The universal approximation property is ensured by selecting a continuous, bounded nonconstant activation function for hidden units and an increasing, nonnegative, locally Lipschitz continuous activation function for the output unit. Moreover, it is argued that the normalization condition can be ensured by appropriately selecting the bias of the output unit. A training approach for multilayer perceptrons based on the minimization of the negative log-likelihood is described in [7].

The latter approach relies on the universal approximation capabilities of multilayer perceptrons to estimate the logarithm of the density function. The network architecture consisted of one hidden layer with logistic activation function and of one output unit with exponential activation function. Training is performed through minimization of the negative log-likelihood of the data with respect to the network parameters [7].

More specifically, let $x_k \in R^d$, $(k = 1, \ldots, n)$ be a set of n data points drawn independently according to an unknown density $g(x)$ we want to approximate. We assume that the density function is defined on a compact subset $S \subset R^d$.

Let also $N(x, p)$ the output of a multilayer perceptron with d input units, arbitrary number of hidden layers with sigmoidal units and an output unit with *exponential* activation function. In addition let p denote the vector of network parameters (weights and biases). Then the function:

$$p_N(x, p) = \frac{N(x, p)}{\int_S N(x, p)dx} \tag{1}$$

constitutes the model pdf with parameter vector p that must be adjusted in order to minimize the negative log-likelihood of the data. It is obvious that $p_N(x, p) \geq 0$ and $\int_S p_N(x, p)dx = 1$. It must be stressed that the computation of the integral is by no means a trivial task and requires the employment of numerical integration techniques. This constitutes the main

burden for the application of MLPs to density estimation problems. It must also be noted that in [7] no details are given concerning the methodology used for computing the integral.

For a given parameter vector p the negative log-likelihood of the set of n observations x_k is given by

$$L(p) = -\sum_{k=1}^{n} \log p_N(x_k, p) \tag{2}$$

which using eq. (1) is also written as

$$L(p) = -\sum_{k=1}^{n} \log N(x_k, p) + n \log \int_S N(x, p) dx \tag{3}$$

For notation convenience let denote $I(p) = \log \int_S N(x, p) dx$.

Training the neural network $N(x, p)$ means finding the optimum parameter vector $p^\star$ so that $L(p^\star)$ is minimum. This can be accomplished using gradient descent methods (e.g. back-propagation or any of its variants), quasi-Newton methods, conjugate gradient methods etc [1, 3, 5]. In [7] the on-line backpropagation algorithm is employed. All the above minimization approaches require the computation of the gradient $\partial L / \partial p_i$ with respect to any network parameter p_i. This gradient computation naturally splits into two parts: i) the computation of the first term in the $L(p)$ formula, which requires the well-known and easily computed gradients $\partial N(x_k, p)/\partial p_i$ and ii) the computation of the gradient $\partial I(p)/\partial p_i$, which depends on the numerical method used for computing the integral and will be described next. It must be noted that neither $I(p)$ nor $\partial I(p)/\partial p_i$ depend on the data points x_k.

3 Numerical Integration Issues

In numerical integration techniques, the value of the integral $I(p)$ is approximated by a weighted sum $\hat{I}(p)$:

$$\hat{I}(p) = \sum_{l=1}^{M} a_l N(y_l, p) \tag{4}$$

where the points $y_l \in S$ are called *integration points* and in general are distributed over the whole domain S, while the parameters a_l are called *integration coefficients* and remain fixed, as long as the integration points remain the same. The integration points can be considered as forming a multidimensional *grid* covering the domain S. Both y_l and a_l depend on S, the grid density (specified by the user) and the selected numerical integration technique. In the following we assume that S is a hyperrectangle in R^d with minimum and maximum value along each dimension i $s_{\min_i}$ and $s_{\max_i}$ respectively.

Among the various techniques for numerical integration we have selected the *trapezoidal rule*, since it is the simplest and computationally less expensive technique both for obtaining

the integration points and computing the integration coefficients [4]. This rule suggests that the points y_l correspond to the nodes of a grid obtained by dividing the interval $[s_{\min_i}, s_{\max_i}]$ along each dimension i using equidistant points (with distance h_i between successive points). The h_i value is user defined and specifies the density of the grid along dimension i. The trapezoidal rule, along with the integration points, specifies the integration coefficients through a very simple formula that *depends only* on h_i $(i = 1, \ldots, d)$ [4].

Let $y_l \in S$ the integration points and a_l $(l = 1, \ldots, M)$ the corresponding integration coefficients. Using eq. (4), the negative log-likelihood to be minimized (eq. (3)) takes the form:

$$\hat{L}(p) = -\sum_{k=1}^{n} \log N(x_k, p) + n \log \hat{I}(p) \tag{5}$$

which gives

$$\hat{L}(p) = -\sum_{k=1}^{n} \log N(x_k, p) + n \log(\sum_{l=1}^{M} a_l N(y_l, p)) \tag{6}$$

Therefore the gradient of $\hat{L}(p)$ with respect to every parameter p_i is given by:

$$\frac{\partial \hat{L}(p)}{\partial p_i} = -\sum_{k=1}^{n} \frac{1}{N(x_k, p)} \frac{\partial N(x_k, p)}{\partial p_i} + \frac{n}{\hat{I}(p)} \sum_{l=1}^{M} a_l \frac{\partial N(y_l, p)}{\partial p_i} \tag{7}$$

which is readily computed.

From the above observation it becomes obvious that the problem of pdf estimation using MLPs (or any other neural architecture whose output is not a pdf per se) can be considered as a training (ie minimization) problem with *two sets* of training points:

- the *actual* training points x_k $k = 1, \ldots, n$ (specified by the problem)
- the *integration* training points y_l $l = 1, \ldots, M$, that are determined by the integration method of choice and depend on the domain S.

The freedom in selecting the integration points makes the problem particularly interesting as it will be described below.

4 The proposed technique

4.1 Preprocessing

In order to apply the pdf estimation method, the input domain S must be specified first. Since the only available information is the set of actual training points x_k, it is natural to specify first the hyperrectangle R_X where these points lie, ie. we compute the values $min_i = \min_k x_{ki}$ and

$max_i = \max_k x_{ki}$ and define $R_X = \{x = (x_1, \ldots, x_d)^\top \in R^d, min_i \leq x_i \leq max_i, \forall i\}$. Then we expand the space R_X in every dimension by a quantity δ, and define the domain S as

$$S = \{x = (x_1, \ldots, x_d)^\top \in R^d, min_i - \delta \leq x_i \leq max_i + \delta, \forall i\}$$

Outside S we assume that the pdf is zero (since no actual points exist) and the space between R_X and S gives the pdf the necessary room for fading to the zero value.

A fundamental problem related with the use of MLPs is that, at the beginning of training (where the parameters p_i take small random values in $(-1, 1)$), the output of an MLP is not a *local function*. This means that the network output is not local and may have nonzero value everywhere in R^d. As a result, it is very difficult for the minimization procedure to force the output of the network to be zero at the boundary of S. To overcome this problem and to locate a good starting point (in terms of parameter values p_i) for the minimization of the likelihood $\hat{L}(p)$, we first perform *supervised training* of the MLP by constructing a training set using some *non-parametric* technique for pdf estimation. This means that, for each integration point y_l $(l = 1, \ldots, M)$, we use the actual points x_k to estimate the pdf $\hat{p}(y_l)$ by a non-parametric technique. More specifically we have selected the Parzen estimation method based on Gaussians [1]:

$$\hat{p}(y_l) = \frac{1}{N} \sum_{i=1}^{n} \frac{1}{(2\pi)^{1/2}\sigma} \exp(-\frac{|x_i - y_l|^2}{2\sigma^2}) \tag{8}$$

where the only parameter to be adjusted is the value of σ.

Using the above non-parametric specification, a training set with M pairs $(y_l, \hat{p}(y_l))$ $(l = 1, \ldots, M)$ is constructed and is subsequently used to train the MLP through the normal minimization procedure of the error function:

$$E(p) = \sum_{l=1}^{M} (N(y_l, p) - \hat{p}(y_l))^2 \tag{9}$$

This preprocessing stage offers two benefits: i) the network output is constrained to the input domain S since $\hat{p}(y_l) \simeq 0$ for y_l near the boundary of S and ii) a good initial point is provided for the minimization of the negative log-likelihood, since the network output after this training phase resembles the unknown pdf to a degree depending on the accuracy of the employed non-parametric pdf estimation method.

The choice for selecting the integration points y_l for supervised training, instead of using the actual points x_k, was based on the fact that the points y_l are uniformly distributed across the whole domain of interest S, while points x_k appear only inside the domain R_X. Moreover, in cases where there are regions inside R_X with no points x_k (ie with zero pdf),

this information would not be incorporated into the training set, if points x_k were used for supervised preprocessing. An additional reason for preferring integration points y_l is that, in general, they are fewer than the actual points. For example, in our experiments the number of actual points was $n = 5000$, while the number of integration ranged from 50 to 225, depending on the domain.

It must be noted that the accuracy of the approximation is not crucial for the success of the method, since the estimated probabilities $\hat{p}(y_l)$ are not accurate. In addition, this is used only as a preprocessing stage providing initial values for the network parameters that will be improved further through the likelihood minimization which constitutes the main training task. Moreover, this preprocessing offers the additional benefit that, from the effectiveness of the training, one may draw conclusions concerning the architecture of the network (for example the number of hidden units). If a specific number of hidden units is not adequate to coarsely perform the above fit (ie to adequately minimize the error function $E(p)$ (eq. (9)), this is an indication that a higher number of hidden units is needed to approximate the unknown pdf.

4.2 Intelligent integration point selection

As stated in the previous section, the accuracy with which the integral $I(p)$ is computed depends critically on the number of the integration points M and the smoothness of the integrated function. If the integration grid is sparse and narrow peaks occur in a space with no grid points, then the peak is not taken into account in the numerical computation of the integral, giving rise to erroneous integral computations. In general many training points are required to cover the input space S, especially in cases where the input dimension d is high. Since integration points are used as training points, the employment of many integration points introduces a computational overhead to the method. Therefore efforts must be made so that M remains as small as possible.

A thing that must be emphasized is the opposite contribution of the actual points and the integration points in the computation of the objective function $\hat{L}$. The minimization of $\hat{L}$ drives the network output $N(x, p)$ to be high at the actual training points x_k and, at the same time, to be low (close to zero) at the integration points y_l. This means that the minimization of the first sum in $\hat{L}$ guides the network to maximize its output for every actual training point, while the presence of the second term ($\hat{I}$) prevents the network output from seeking the obvious maximum at ∞ [7].

An ill-conditioned solution that is very often obtained from the minimization procedure results from the case of inaccurate numerical integration mentioned previously: if many actual

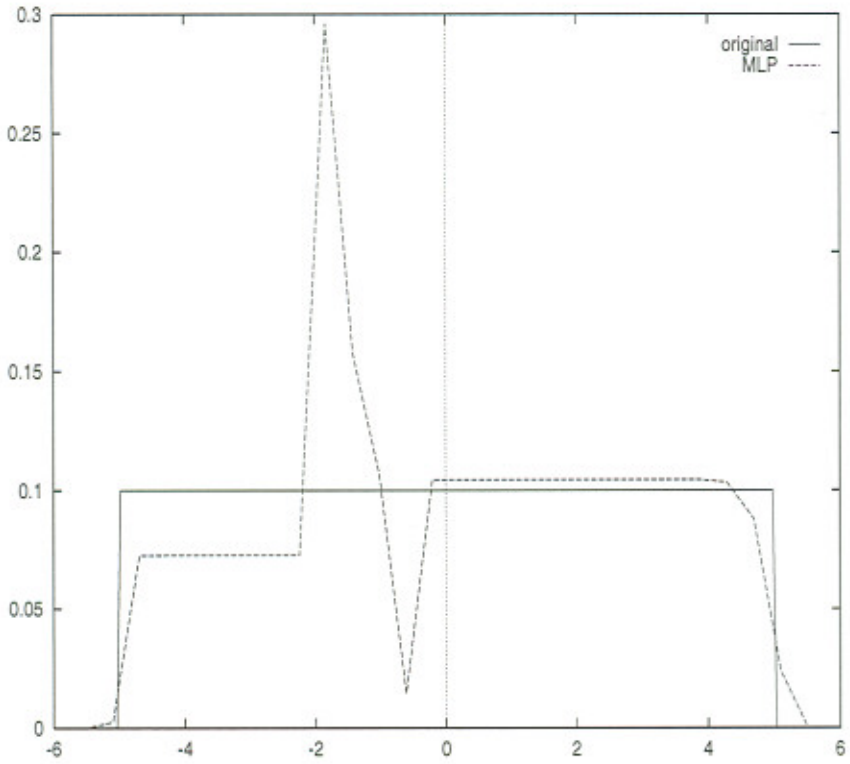

Figure 1: An unacceptable approximation to the uniform pdf that is due to erroneous numerical integration

.

training points exist inside a cell of the integration grid, then high peaks may be formed inside this cell (where no integration points exist), due to the above mentioned force of the minimization procedure to provide high values for the actual training points. Such peaks do not contribute to the integral computation, therefore the likelihood function is falsely computed and obtains large negative values. An example of such phenomenon is depicted in Fig. 1 for the case where a uniform one-dimensional pdf is approximated. This behavior is encountered frequently in the case where the integration grid is sparse (low density), but may also happen even if the grid is dense, since it is always possible for more narrow peaks to be constructed.

A way to tackle the above problem is to use an intelligent selection scheme for the integration points. We call this selection scheme the *moving grid method*, since at each epoch t of the minimization algorithm (which may be either gradient descent or any other method) we consider that the grid is *randomly* moving. This means that at each epoch t every integration point $y_l(t) = (y_{l1}(t), \ldots, y_{ld}(t))^\top$ is perturbed from its original position $y_l(0)$ by setting: $y_{li}(t) = y_{li}(0) + d_i(t)$, where for each i the displacements $d_i(t)$ are randomly selected in $(-h_i/2, h_i/2)$ (using the uniform distribution). It must be noted that, at each epoch t, the

values of $d_i(t)$ are selected once for every i and are used for the update of all points $y_l(t)$. Therefore, for each t, the whole integration grid is moving by $d_i(t)$ along each dimension i. Consequently, the relative distances among integration points $y_l(t)$ remain fixed, thus the integral parameters a_l remain fixed and need not be computed at each epoch.

If $N(x,p)$ is relatively smooth (does not form strange narrow peaks inside a grid cell), the random displacement of the integration grid by a small amount at each epoch, affects the value of the sum $\hat{I}(p)$ only at the boundary of S, since at the interior points the relative distances between points do not change. But, as a result of the preprocessing stage, the network output $N(x,p)$ is close to zero near the boundary of S, therefore the integral value seem to be insensitive to the perturbation values d_i. The advantage of using the moving grid approach is that we obtain a significant benefit: it is not possible for the peaks to be constructed, since the integration points y_l are constantly moving around their original position, leaving no space for the formation of undesirable peaks. Therefore, the proposed method leads to accurate numerical integration and yields accurate solutions. Of course, accuracy depends on the density of the integration grid, but we have found experimentally that using the above technique, it is possible to obtain solutions of acceptable accuracy even with sparse integration grids, in cases where the pdf to be approximated is relatively smooth.

Summarizing all the above issues, the proposed method for pdf estimation using MLPs is specified as follows:

1. Initialization: set $t := 0$, specify the domain S, the density of the integration grid (h_i values) and compute the integration points $y_l(0)$ and the integration coefficients a_l ($l = 1, \ldots, M$).

2. Preprocessing:

   - Compute $\hat{p}(y_l)$ using eq. (8).
   - Train the MLP to minimize the error function $E(p)$ (eq. (9)).

3. **Repeat**

   - Set $t := t + 1$
   - Compute displacements $d_i(t)$ selected uniformly in $(-h_i/2, h_i/2)$ $i = 1, \ldots, d$.
   - Compute the new integral points: $y_{li}(t) = y_{li}(0) + d_i(t)$ for $l = 1, \ldots, M$, $i = 1, \ldots, d$.
   - Compute the integral $\hat{I}(p) = \sum_{l=1}^{M} a_l N(y_l(t), p)$.
   - Compute the gradients for each parameter p_i: $\partial N(x_k, p)/\partial p_i$ $(k = 1, \ldots, n)$ and $\partial N(y_l(t), p)/\partial p_i$ $(l = 1, \ldots, M)$.

- For each parameter p_i set:

$$\frac{\partial \hat{L}}{\partial p_i} = -\sum_{k=1}^{n} \frac{1}{N(x_k, p)} \frac{\partial N(x_k, p)}{\partial p_i} + \frac{n}{\hat{I}(p)} \sum_{l=1}^{M} a_l \frac{\partial N(y_l(t), p)}{\partial p_i}$$

- Using the gradient information, update the parameter vector p using either gradient descent or quasi-Newton methods or conjugate gradient methods.

4. **Until** some termination criterion is satisfied

As in the case with ordinary supervised MLP training, several termination criteria may be specified, for example, maximum number of epochs or convergence of the parameter vector to a local minimum.

5 Examples

To assess the effectiveness of our approach we have conducted experiments with data drawn independently from known distributions, which in turn we tried to approximate using the described approach. After training, we tested the accuracy of the obtained solution with respect to the one used to generate the data. For visualization purposes we have considered one dimensional and two dimensional problems and for each problem we have also compared our approach with results obtained from the use of the gaussian mixture technique trained using the Expectation-Maximization (EM) algorithm [2, 11]. The latter constitutes the most popular parametric technique for pdf estimation [1, 2, 13].

In all experiments we have considered an MLP with d input units, one hidden layer with 10 sigmoid logistic units and one output unit with exponential activation function (as in [7]). Training in both the preprocessing stage and the likelihood minimization phase was performed using the *quasi-Newton BFGS* method [3, 5], which is provided by the MERLIN optimization environment [9] and has been found to be superior to gradient descent for MLP training [6]. Training was stopped when convergence was achieved, ie. the infinite norm of the difference in the value of $\hat{L}$ between two successive epochs was less than 0.01. Moreover, after training, an additional integral computation is performed with many integration points, to ensure accurate computation of the integral I. Then, for each point x, the corresponding density value is $p_N(x) = N(x)/I$.

In all problems we have considered a training set with $n = 5000$ data points drawn independently from the corresponding pdf to be approximated. Moreover, the value of parameter σ used in nonparametric estimation was set equal to 0.1 in all cases. It must be noted that we have also tested the Monte-Carlo method for numerical integration [10] using points se-

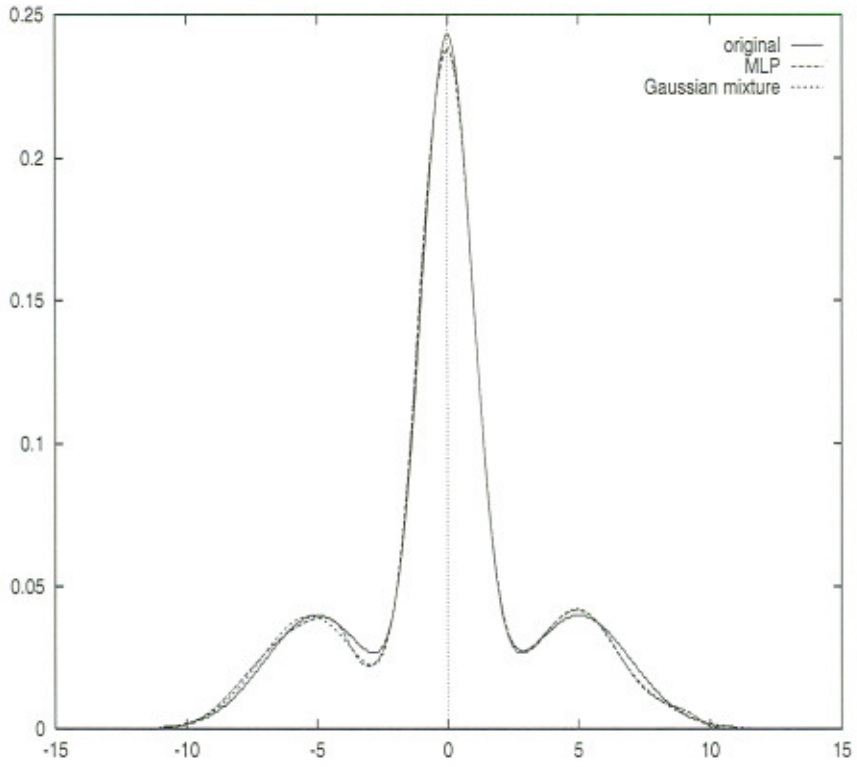

Figure 2: The approximation of a gaussian mixture pdf with four kernels

lected from the initial pdf obtained after the preprocessing stage, but the number of required integration points to achieve satisfactory accuracy was extremely high.

We have considered three one dimensional problems: i) a gaussian mixture pdf with four kernels, ii) a uniform pdf and iii) a non-smooth pdf tested in [7]. In addition, we have considered a two dimensional uniform pdf.

In all experiments, since we the original pdf $g(x)$ is known, we computed the theoretically optimal log-likelihood $\tilde{L}(x_1, \ldots, x_n)$ for the specific set of samples x_k $(k = 1, \ldots, n)$ that we have used in every problem:

$$\tilde{L}(x_1, \ldots, x_n) = \sum_{k=1}^{n} log(g(x_k)) \tag{10}$$

Moreover, in all experiments with the gaussian mixture method, we have used a mixture with 10 kernels (same as the number of hidden MLP units).

5.1 Gaussian mixture pdf

In this experiment we have generated samples using the following pdf:

$$g(x) = 0.2N(-5, 2) + 0.4N(0, 1) + 0.2N(1, 0.5) + 0.2N(5, 2)$$

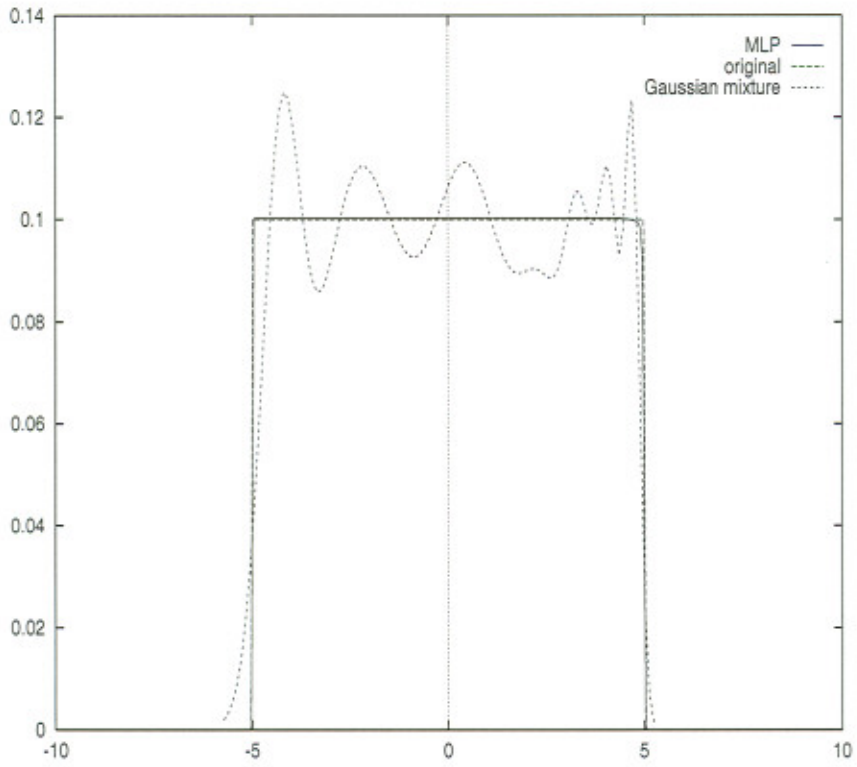

Figure 3: The approximation of a one-dimensional uniform pdf.

We considered a one-dimensional integration grid with $M = 50$ equidistant points in $[-15, 15]$. It is obvious that in this case the gaussian mixture method is expected to exhibit accurate approximation behavior. As Fig. 2 indicates, the MLP approach also provides a very good approximation to $g(x)$. The value of $\tilde{L}(x_1, \ldots, x_n)$ was -12445 and the likelihood of the obtained solutions was -12452 for the gaussian mixture model and -12461 for the MLP model, ie very close to the optimal.

5.2 1-d Uniform pdf

In this case the pdf to be approximated was the one-dimensional uniform pdf in the interval $[-5, 5]$. We considered a one-dimensional integration grid with $M = 50$ equidistant points in $[-7, 7]$.

It is known that the gaussian mixture method encounters difficulties in approximating uniform pdfs. This difficulty was confirmed by our experiments (Fig. 3). On the contrary, the solution provided by the MLP approach was an excellent approximation to $g(x)$. The value of $\tilde{L}(x_1, \ldots, x_n)$ was -11512 and the likelihood of the obtained solutions was -11652 (far from the optimal) for the gaussian mixture model and -11518 (almost optimal) for the MLP model.

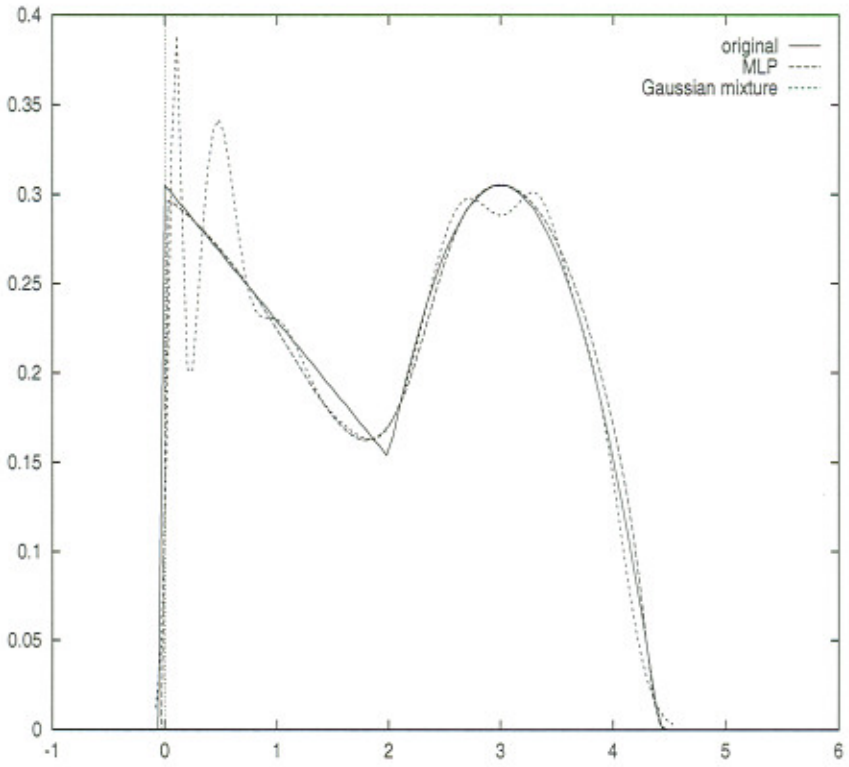

Figure 4: The approximation of a non-smooth pdf.

5.3 A non-smooth pdf

In this experiment the unknown pdf was the same as the one used in [7]:

$$g(x) = \begin{cases} 0 & \text{if } x > 0 \text{ or } x \geq 3 + \sqrt{2} \\ (2 - x/2)/6.5523 & \text{if } 0 \leq x < 2 \\ (2 - (x-3)^2)/6.5523 & \text{if } 2 \leq x < 3 + \sqrt{2} \end{cases} \tag{11}$$

We considered a one-dimensional integration grid with $M = 50$ equidistant points in $[-1, 6]$. Fig. 4 displays the solutions obtained using the gaussian mixture and the MLP approach. It is clear that the MLP method is more effective in approximating $g(x)$. The value of $\tilde{L}(x_1, \ldots, x_n)$ was -7166 and the likelihood of the obtained solutions was -7195 for the gaussian mixture model and -7171 (almost optimal) for the MLP model.

5.4 2-d Uniform pdf

The pdf to be approximated was a two-dimensional uniform pdf defined on the domain $[0, 0.2] \times [0, 0.2]$. We considered a two-dimensional integration grid with $M = 225$ grid points obtained by dividing the interval $[-0.1, 0.3]$ on the x-axis and the interval $[-0.1, 0.3]$ on the y-axis using 15 equidistant points for each axis.

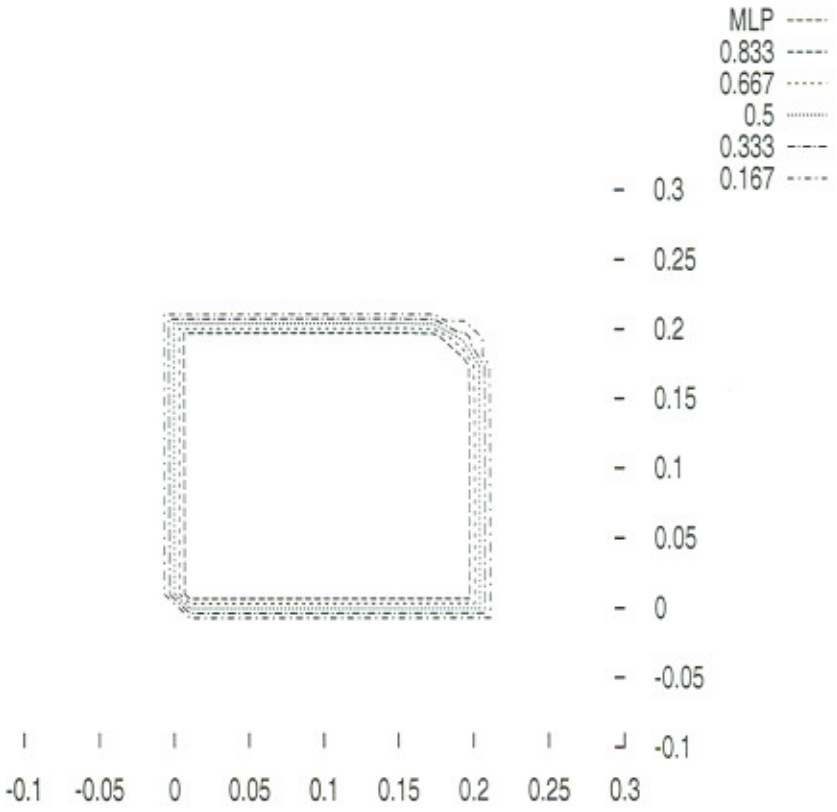

Figure 5: Contour plots of the solution provided by the MLP model for a two-dimensional uniform pdf in $[0, 0.2] \times [0, 0.2]$.

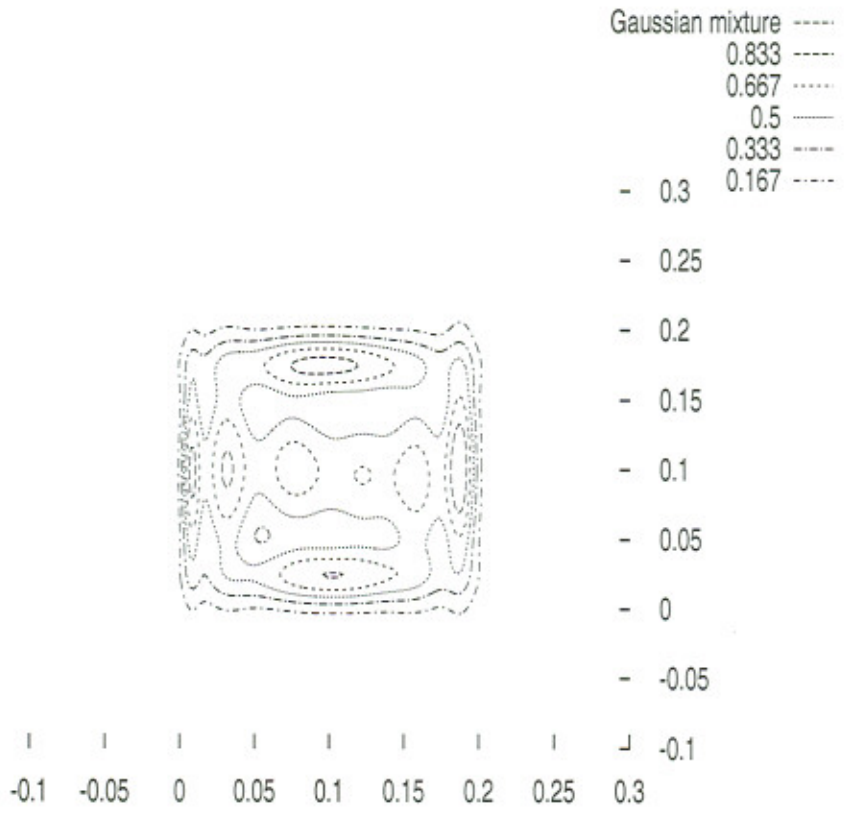

Figure 6: Contour plots of the solution provided by the gaussian mixture model for a two-dimensional uniform pdf in $[0, 0.2] \times [0, 0.2]$.

Figures 5 and 6 display the contour plots of the obtained solutions using the MLP and the gaussian mixture method respectively. It is obvious that the gaussian mixture method provides a solution of poor quality. On the other hand, the MLP approach is very effective and is able to approximate the uniform pdf with very good accuracy. The value of $\tilde{L}(x_1, \ldots, x_n)$ was -16094 and the likelihood of the obtained solutions was -15532 for the gaussian mixture model and -15953 for the MLP model.

From the above experiments, it is clear that the proposed MLP method is much more effective in approximating pdfs with linear structure compared with the gaussian mixture approach. In, addition, as indicated by the first experiment, it is almost equally effective in approximating pdfs with gaussian shape. In what concerns execution time, the results are comparable for both methods.

6 Conclusions

An effective approach has been presented for pdf estimation given a set of samples that is based on the use of multilayer perceptrons. It must be noted that the approach is general and any feedforward neural architecture (with universal approximation properties) can be used in place of MLPs (for example RBF networks). As experiments indicate, the estimation method is very flexible and exhibits in many cases superior approximation capabilities compared with gaussian mixtures.

Current research focuses on the use of techniques for model selection, such as incremental constructive techniques for adjusting the network architecture and pruning techniques for reducing the number of parameters. We also study techniques for constructing integration grids with adaptive density, based on density estimation results obtained from the preprocessing stage. Another important issue of research is related with the hardware implementation of the method, either on parallel machines or on neuroprocessors. Such an implementation is particularly interesting in our case due to the computational overhead imposed by the integral calculations.

Finally, we aim at testing the effectiveness of the proposed method on several real problems where pdf estimation is required. In addition, we will focus on the application of the technique to classification problems. This can be carried out in two ways: the straightforward one is to construct one pdf for each class (using only the data belonging to the particular class) and then perform decisions on new data by applying the Bayes rule to select the class with maximum density. The second (and more challenging) is to define an error function based on the pdf of each class and, instead of training the network through likelihood maximization, to perform training by minimizing the probability of misclassification.

Acknowledgments

The author would like to thank Dr. N. Vlassis, Dept. of Electrical and Computer Engineering, National Technical University of Athens, Greece, for fruitful discussions and for providing the EM code for the gaussian mixture technique.

References

[1] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society B*, vol. 39, pp. 1-38, 1977.

[3] R. Fletcher, *Practical Methods of Optimization*, Wiley, 1987.

[4] E. Isaacson and H. Bishop Keller, *Analysis of Numerical Methods*, Wiley, 1994.

[5] P. Gill, W. Murray and M. Wright, *Practical Optimization*, Academic Press, 1997.

[6] A. Likas, D.A. Karras and I.E. Lagaris, "Neural Network Training and Simulation Using a Multidimensional Optimization System", *Int. J. of Computer Mathematics*, vol. 67, pp. 33-46, 1998.

[7] D.S. Modha and Y. Fainman, "A Learning Law for Density Estimation", *IEEE Trans. on Neural Networks*, vol. 5, no. 3, pp. 519-523, 1994.

[8] D.S. Modha and E. Masry, "Rate of convergence in density estimation using neural networks", *Neural Computation*, vol 8, no 5, pp. 1107-22, 1996.

[9] D.G. Papageorgiou, I.N. Demetropoulos and I.E. Lagaris, "Merlin 3.0, A multidimensional optimization environment", *Computer Physics Communications*, vol. 109, pp. 227–249, 1998.

[10] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, *Numerical Recipes in C*, Cambridge, MA: Cambridge Univ. Press, 1988.

[11] R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm", *SIAM Review*, vol. 26, no. 2, pp. 195-239, 1984.

[12] H. Traven, "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, 1991.

[13] N. A. Vlassis, G. Papakonstantinou and P. Tsanakas, "Mixture Density Estimation based on Maximum Likelihood and Test Statistics", *Neural Processing Letters*, to appear, vol. 9, no. 1, 1999.

[14] H. White, "Parametric Statistical Estimation with Artificial Neural Networks", in *Mathematical Perspectives on Neural Networks*, P. Smolensky, M. Mozer and D. Rumelhart, (eds), Hilldale, NJ: Erlbaum Associates, 1992.