Computer Physics Communications 135 (2001) 167–175

Computer Physics Communications

www.elsevier.nl/locate/cpc

Probability density estimation using artificial neural networks

Aristidis Likas

Department of Computer Science, University of Ioannina, 45110 Ioannina, Greece

Received 18 July 2000; received in revised form 11 September 2000

Abstract

We present an approach for the estimation of probability density functions (pdf) given a set of observations. It is based on the use of feedforward multilayer neural networks with sigmoid hidden units. The particular characteristic of the method is that the output of the network is not a pdf, therefore, the computation of the network's integral is required. When this integral cannot be performed analytically, one is forced to resort to numerical integration techniques. It turns out that this is quite tricky when coupled with subsequent training procedures. Several modifications of the original approach (Modha and Fainman, 1994) are proposed, most of them related to the numerical treatment 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 very effective and in most cases superior to the method of Gaussian mixtures. © 2001 Elsevier Science B.V. All rights reserved.

PACS: 02.70.Lq; 07.05.Kf; 07.05.Mh

Keywords: Probability density estimation; Neural networks; Multilayer perceptron; Gaussian mixtures

1. Introduction

Probability density function estimation from a given set of observations constitutes an important computational problem with applications to many areas of experimental physics (high energy, spectroscopy, etc.). Parametric techniques for probability density function (pdf) estimation assume a model of suitable functional form with parameters to be adjusted so that it approximates 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 has been shown to exhibit the universal approximation property and efficient procedures for likelihood maximization have been developed for this model (EM algorithm) [2–4]. 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 [1] the use of feedforward neural networks with sigmoid hidden units called 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 [5], concerning the rate of convergence of such techniques. In this work we elaborate on the use of MLPs as models for pdf estimation. Our model pdf is written as $N(x,p)/\int_S N(z,p)\,\mathrm{d}z$, where

E-mail address: arly@cs.uoi.gr (A. Likas).

0010-4655/01/$ – see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0010-4655(00)00235-6

$N(x, p) \geqslant 0$ $(x \in S)$ is the output of an MLP with parameter vector p for a given input x.

It is well known that the Gaussian mixture approach encounters difficulties in approximating the uniform distribution. This is not the case for the MLP model. We conducted experiments that demonstrate the adaptability of the MLP model and its superiority in approximating the uniform distribution. However, MLPs do not integrate analytically and, hence, numerical quadrature has to be employed. This is a disadvantage and in addition, as it will be shown, incorporates salient difficulties in the optimization procedure that is used for training.

We propose first a preprocessing stage where supervised training is used to obtain a coarse approximation of the unknown pdf and force the network output in the domain of interest. For the quadrature we propose a moving grid approach. This is very important since, if a constant grid is used instead, it amounts to weighting the importance of the grid points in the optimization procedure and hence producing artifacts. We will see this point in detail, since to the best of our knowledge, it has not been discussed in the literature, most likely because it went by unobserved due to its salient character.

It must also be noted that except for the ordinary approaches to density estimation that are based on the maximization of the data likelihood, a very interesting different approach has been suggested that employs classification neural networks [6]. To approximate the unknown density $p_{\text{data}}(x)$ this approach uses artificially generated data drawn from a known (user specified) pdf $p_{\text{ref}}(x)$. Then a feedforward classification neural network is trained to determine whether a sample x has been drawn from the unknown density $p_{\text{data}}(x)$ or from the known density $p_{\text{ref}}(x)$. Finally, using the Bayes rule the outputs of the neural network can be used to compute the value of $p_{\text{data}}(x)$. From this brief description it is clear that this method is fundamentally different from the proposed one. The Garrido and Juste method transforms the density estimation problem to a classification problem, while our method treats the density estimation method in the regular way as a maximization problem of the data likelihood. Due to this crucial difference, many other differences between the two methods also arise that concern the network architecture, the objective function to be optimized and the numerical treatment and implementation of the training methods.

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 with the Gaussian mixture method 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 [7]. A training approach for multilayer perceptrons based on the minimization of the negative log-likelihood is described in [1]. For problems defined in R^d, the network architecture consisted of d input units, one hidden layer with H hidden units having the logistic activation function and of one output unit with exponential activation function:

$$N(x, p) = \exp\left(\sum_{i=1}^{H} v_i \sigma(o_i)\right), \tag{1}$$

where

$$o_i = \sum_{j=1}^{d} w_{ij} x_j + u_i \tag{2}$$

and

$$\sigma(z) = 1/\bigl(1 + \exp(-z)\bigr)$$

is the logistic (sigmoid) activation function. The model's adjustable parameters w_{ij}, u_i and v_i ($i = 1, \ldots, H$, $j = 1, \ldots, d$) are collectively denoted as a vector p. Training is performed through minimization of the negative log-likelihood of the data with respect to the network parameters p_i [1].

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 $g(x)$ that we want to approximate. We assume that the density function is defined on a compact subset $S \subset R^d$. The function:

$$p_N(x, p) = \frac{N(x, p)}{\int_S N(z, p)\,\mathrm{d}z} \tag{3}$$

constitutes the model pdf with parameter vector p that must be adjusted so as to minimize the negative log-likelihood of the data.

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

$$L(p) = -\sum_{k=1}^{n} \log p_N(x_k, p) \tag{4}$$

which using Eq. (3) is also written as

$$L(p) = -\sum_{k=1}^{n} \log N(x_k, p) + n \log \int_S N(x, p)\, \mathrm{d}x. \tag{5}$$

For notational convenience let us denote

$$I(p) = \int_S N(x, p)\, \mathrm{d}x.$$

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., backpropagation or any of its variants), quasi-Newton methods, conjugate gradient methods, etc. [8,9]. In [1] the on-line backpropagation algorithm is employed. The above minimization approaches require the computation of the gradient $\partial L/\partial p_i$ with respect to the network parameters p_i. This gradient computation naturally splits into two parts:

(i) the computation of the gradient 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 $\hat{I}(p)$:

$$\hat{I}(p) = \sum_{l=1}^{M} a_l N(y_l, p), \tag{6}$$

where the points $y_l \in S$ are the *integration points* and in general are distributed over the whole domain S, while the parameters a_l are the *integration coefficients*. 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 *equidistant point trapezoidal rule*, due to its simplicity.

Combining Eqs. (6) and (5) the negative log-likelihood to be minimized takes the form:

$$\hat{L}(p) = -\sum_{k=1}^{n} \log N(x_k, p) + n \log\left(\sum_{l=1}^{M} a_l N(y_l, p)\right). \tag{7}$$

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

$$\frac{\partial \hat{L}(p)}{\partial p_i} = -\sum_{k=1}^{n} \frac{1}{N(x_k, p)} \frac{\partial N(x_k, p)}{\partial p_i} + \frac{n}{\hat{I}(p)} \sum_{l=1}^{M} a_l \frac{\partial N(y_l, p)}{\partial p_i}. \tag{8}$$

Note that in Eq. (7) the choice of y_l should not affect the result (to within reasonable numerical accuracy), since these are points used to estimate the integral and are not part of the original problem. However, due to the numerical quadrature, the objective function $\hat{L}(p)$ does depend on the choice of y_l. We will next describe a procedure that essentially removes this dependency. The main idea is to use different sets of $\{y_l\}$ points each time we have to estimate the integral $I(p)$ and its gradient. Hence, the objective function will not depend on any particular set of points, but a to plethora of them, which amounts to essentially removing any particular dependence. We call this technique the *moving grid* procedure and it is taken up in Section 4.2.

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 training points x_k, it is natural to specify first the hyperrectangle R_X where these points lie, i.e. we compute the values $\min_i = \min_k x_{ki}$ and $\max_i = \max_k x_{ki}$ and define $R_X = \{x = (x_1, \dots, x_d)^\top \in R^d,\ \min_i \leqslant x_i \leqslant \max_i, \forall i\}$. Then we expand the space R_X in every dimension by a quantity δ, and define the domain S as

$$S = \bigl\{x = (x_1, \dots, x_d)^\top \in R^d, \\ \min_i - \delta \leqslant x_i \leqslant \max_i + \delta, \forall i\bigr\}.$$

Outside S we assume that the pdf is zero (since no training points exist) and the space between R_X and S gives the pdf the necessary room to fade out.

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 values 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 $(l = 1, \dots, M)$, we use the training 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 [8]:

$$\hat{p}(y_l) = \frac{1}{N}\sum_{i=1}^{n} \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(-\frac{|x_i - y_l|^2}{2\sigma^2}\right), \qquad (9)$$

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

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

$$E(p) = \sum_{l=1}^{M} \bigl(N(y_l, p) - \hat{p}(y_l)\bigr)^2. \qquad (10)$$

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.

4.2. Intelligent integration point selection

A thing that must be emphasized is the competition between the training 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 training points x_k and, at the same time, to be low (close to zero) at the integration points y_l. If many training points exist inside a cell of the integration grid, then high peaks may be formed inside this cell, due to the above mentioned tendency of the minimization procedure to maximize the network at training points and minimize it at the grid points. Such intermediate peaks do not contribute to the integral computation since the integral depends on the values of the network at the grid points only. Therefore the likelihood function is falsely computed and obtains large negative values. An example of such an artifact 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 will also happen even if the grid is dense, as long as there are training points inside the integration cell.

We conducted also the following one dimensional experiment in order to verify the source of the artifact. We considered as grid points all the distinct training points and used a non-equidistant trapezoidal rule of integration. Indeed no peaks were created since there were no training points in-between two successive grid points. However, extending this to more than one

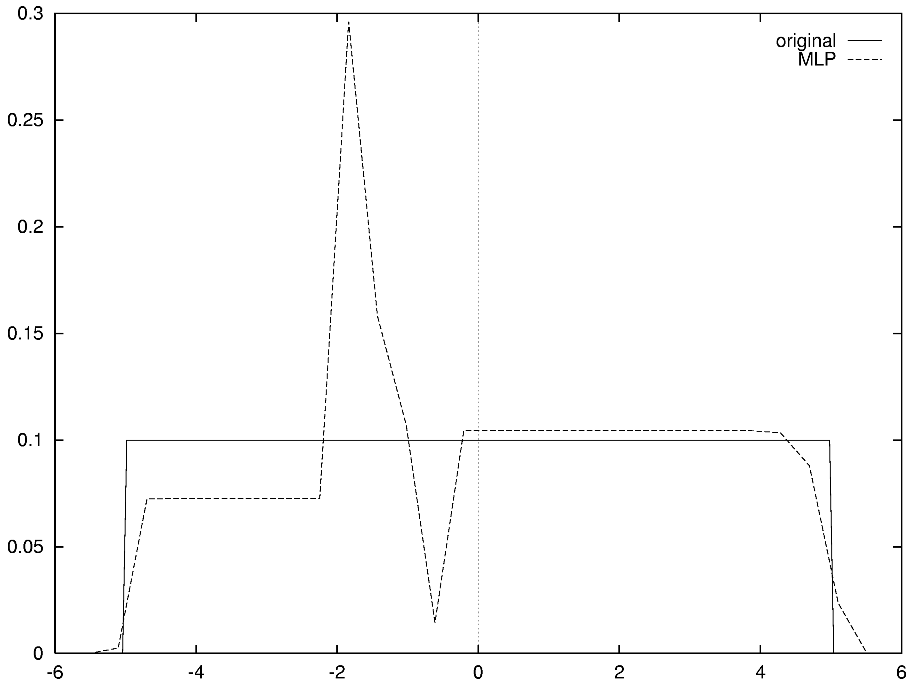

Fig. 1. An unacceptable approximation to the uniform pdf that is due to erroneous numerical integration.

dimension is troublesome and, hence, we proceeded by investigating alternative approaches.

A way to tackle the above problem that can be expanded in more than one dimension 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 we consider that the grid is moving by random offsets. This means that at each epoch t every integration point $y_l(t) = (y_{l1}(t), \ldots, y_{ld}(t))^\top$ is translated 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 $(-h_i/2, h_i/2)$ (h_i being the grid step in dimension i). It must be noted that, at each epoch t, the values of $d_i(t)$ are selected once for every i and are used to update all points $y_l(t)$. Therefore, for each t, the whole integration grid is offset by $d_i(t)$ along each dimension i. Consequently, the relative distances among integration points $y_l(t)$ remain fixed, thus the integration parameters a_l remain the same throughout the calculation.

The advantage of using the moving grid approach is that, since the integration points y_l are constantly moving around their original position, the formation of the undesirable peaks is avoided. Therefore, the proposed technique leads to correct solutions 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. (9).
   - Train the MLP to minimize the error function $E(p)$ (Eq. (10)).
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 integration 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 steepest descent, quasi-Newton or conjugate gradient methods.

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

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 with the proposed approach using an MLP with 10 sigmoid hidden units. After training, we tested the accuracy of the obtained solution with respect to the one used to generate the data. We have also compared our approach with results obtained from the use of the Gaussian mixture technique (with 10 kernels) trained using the Expectation-Maximization (EM) algorithm. The latter constitutes the most popular parametric technique for pdf estimation [2,4,8].

MLP training in both the preprocessing stage and the likelihood minimization phase was performed using the *quasi-Newton BFGS* method [9], which is provided by the MERLIN [10] optimization environment and has been found to be superior to gradient descent for MLP training [11].

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

In all experiments, since 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:

$$\tilde{L}(x_1, \ldots, x_n) = \sum_{k=1}^{n} \log\bigl(g(x_k)\bigr). \tag{11}$$

5.1. Example 1

In this experiment we have generated samples using the following pdf which is a mixture of two Gaussian and two uniform kernels:

$$g(x) = 0.25N(-7, 0.5) + 0.25U(-3, -1) + 0.25U(1, 3) + 0.25N(7, 0.5)$$

with $\tilde{L} = -10477$. We considered a one-dimensional integration grid with $M = 50$ equidistant points in $[-12, 12]$. As Fig. 2 indicates, the MLP approach provides very good approximation to $g(x)$ $(L = -10\,492)$ and is more effective compared to the Gaussian mixture model $(L = -10\,631)$.

5.2. Example 2

In this experiment the unknown pdf was the same as the one used in [1]:

$$g(x) = \begin{cases} 0 & \text{if } x < 0 \text{ or } x \geqslant 3 + \sqrt{2}, \\ (2 - x/2)/6.5523 & \text{if } 0 \leqslant x < 2, \\ \bigl(2 - (x-3)^2\bigr)/6.5523 & \text{if } 2 \leqslant x < 3 + \sqrt{2} \end{cases} \tag{12}$$

with $\tilde{L} = -7166$. We considered a one-dimensional integration grid with $M = 50$ equidistant points in $[-1, 6]$. Fig. 3 displays the solutions obtained using the Gaussian mixture and the MLP approach. It is clear that the MLP method is more effective

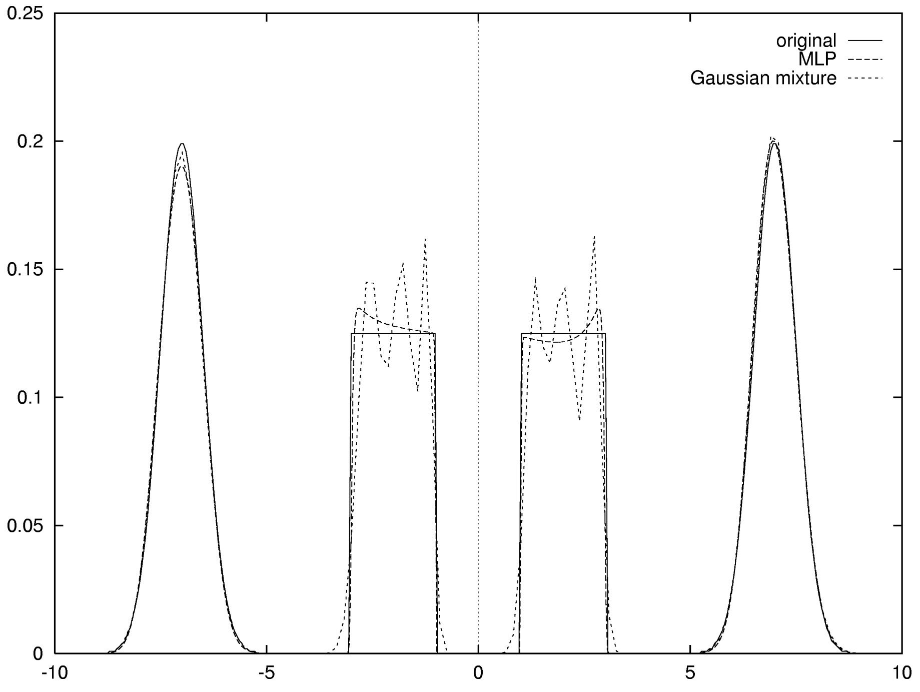

Fig. 2. The approximation of the mixture pdf of Example 1.

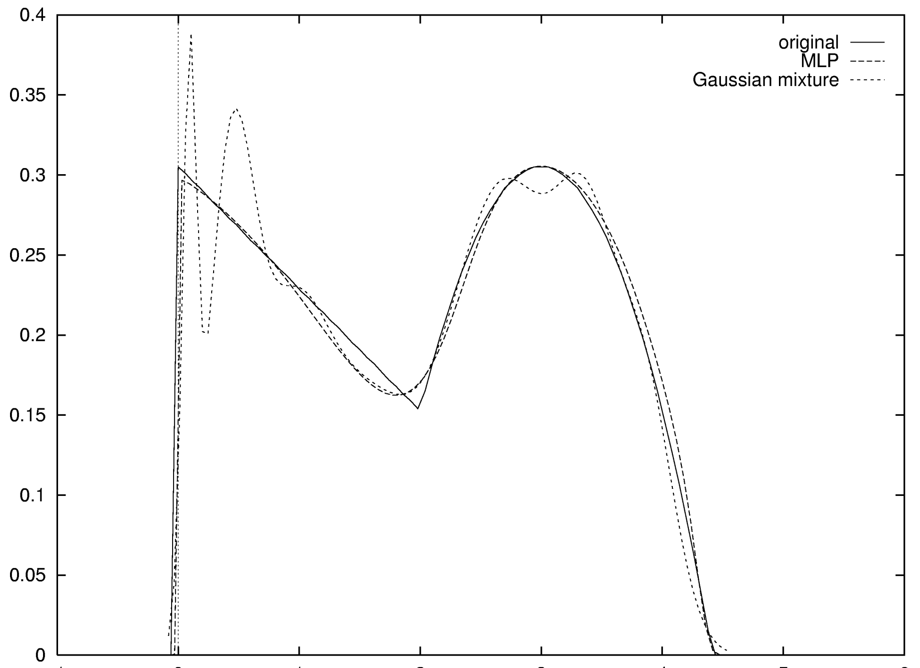

Fig. 3. The approximation of a non-smooth pdf.

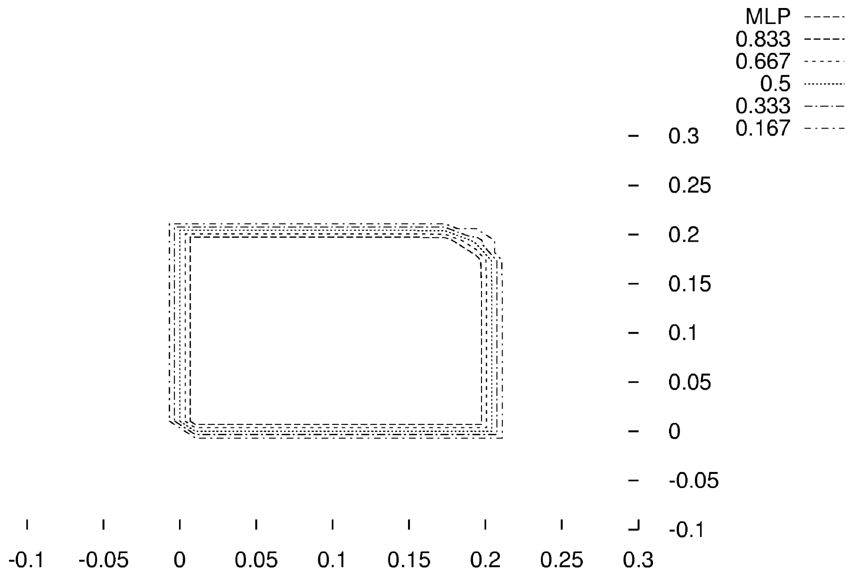

Fig. 4. Contour plots of the solution provided by the MLP model for a two-dimensional uniform pdf in $[0, 0.2] \times [0, 0.2]$.

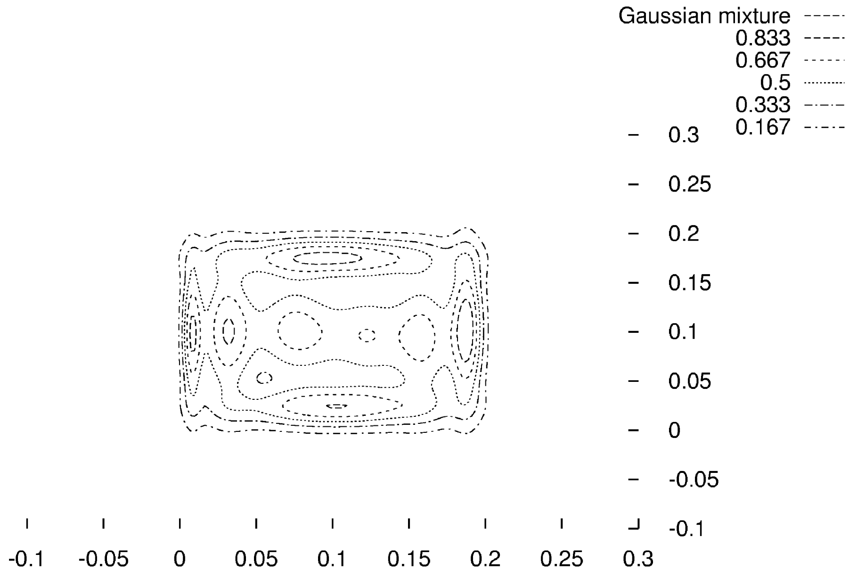

Fig. 5. Contour plots of the solution provided by the Gaussian mixture model for a two-dimensional uniform pdf in $[0, 0.2] \times [0, 0.2]$.

in approximating $g(x)$. The likelihood of the obtained solutions was -7195 for the Gaussian mixture model and -7171 (almost optimal) for the MLP model.

5.3. Example 3

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.

Figs. 4 and 5 display the contour plots of the obtained solutions using the MLP and the Gaussian mixture method respectively. The values corresponding to the contours are normalized by dividing with the maximum pdf value which is equal to 25. 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 $\tilde{L}$ was $-16\,094$ and the likelihood of the obtained solutions was $-15\,532$ for the Gaussian mixture model and $-15\,953$ for the MLP model.

6. Conclusions

We have presented an effective approach based on artificial neural networks to pdf estimation from a given a set of samples. We found that this method for low-dimensional problems has superior estimation capability compared to the widely used Gaussian mixtures approach. The disadvantage, that limits the method to low dimensions is that the normalizing integral cannot be computed analytically and one has to resort to numerical quadrature. However, up to three or four dimensions the method is a serious alternative tool for pdf estimation and problems of this dimensionality are frequently encountered in physics and other sciences.

For problems in higher dimensions, the only applicable quadrature schemes belong to the Monte-Carlo class [12], which raise the computational load excessively. Therefore, for high dimensional problems it might be preferable to employ Gaussian mixture models (where the integral is computed analytically), perhaps by sacrificing to some degree the accuracy of the model.

Future research may focus on the use of techniques for model selection, such as incremental construction of the neural network architecture or pruning for reducing the number of parameters. We also study techniques for constructing adaptive integration grids, based on the estimated density from the preprocessing stage. Such techniques may also prove useful in the case where Monte-Carlo integration is used. Another future research issue is 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 high computational load required by the numerical quadrature at higher dimensions.

Acknowledgements

The author would like to thank Dr. N. Vlassis of the University of Amsterdam, both for fruitful discussions and for providing the EM code for the Gaussian mixture technique, and also Prof. I.E. Lagaris of the University of Ioannina for illuminating discussions and encouragement.

References

[1] D.S. Modha, Y. Fainman, IEEE Trans. on Neural Networks 5 (1994) 519.

[2] A.P. Dempster, N.M. Laird, D.B. Rubin, J. Roy. Statist. Soc. B 39 (1977) 1.

[3] R. Redner, H. Walker, SIAM Rev. 26 (1984) 195.

[4] N.A. Vlassis, G. Papakonstantinou, P. Tsanakas, Neural Process. Lett. 9 (1999) 63.

[5] D.S. Modha, E. Masry, Neural Comput. 8 (1996) 1107.

[6] L. Garrido, A. Juste, Comput. Phys. Comm. 115 (1998) 25.

[7] H. White, Mathematical Perspectives on Neural Networks, P. Smolensky, M. Mozer, D. Rumelhart (Eds.), Erlbaum Associates, 1992.

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

[9] R. Fletcher, Practical Methods of Optimization, Wiley, New York, 1987.

[10] D.G. Papageorgiou, I.N. Demetropoulos, I.E. Lagaris, Comput. Phys. Comm. 109 (1998) 227.

[11] A. Likas, D.A. Karras, I.E. Lagaris, Int. J. Comput. Math. 67 (1998) 33.

[12] F. James, Rep. Prog. Phys. 43 (1980) 1145.