Density Estimation as a Preprocessing Step for Constructive Algorithms

J.C. Lemm[*], V. Beiu[†], and J.G. Taylor
Department of Mathematics, King's College London
Strand, London WC2R 2LS, UK

Abstract

The aim of the paper is to combine the generalization power of classical density estimation techniques with the efficiency of VLSI–friendly, constructive algorithms. This is important for VLSI–implementations of classification networks in the case of noisy data to avoid the well–known "overfitting" effects. The method consists of three steps 1) estimation of the probability densities for each class, 2) discretization of the input space and 3) describing the resulting classification regions using only easy implementable boolean AND and OR gates and comparisons. The "noisy spirals" classification problem, a noisy variant of the "two spirals" benchmark, is used for demonstration.

Introduction: Classical neural network architectures with McCulloch–Pitts–like neurons are not well suited for VLSI–implementations. They require multiplications which are quite complex operations in terms of boolean gates, and they usually have a very high connectivity, compared to a limited fan–in VLSI–implementation.

To enable effective hardware realizations of classifiers or function approximators there exist especially designed algorithms which build networks with simple nodes, limited fan–in, and a local connectivity structure [3, 2]. These algorithms are constructive, i.e. they explicitly build up a network architecture adapted to the problem [4, 8]. In contrast, classical neural net algorithms like backpropagation fix the architecture and only train the weights.

Many of the VLSI–friendly algorithms have been developed with regard to their efficiency, but not with regard to their generalization performance. Especially in the case of noisy data the generalization behaviour of a network should be controlled to avoid "overfitting" of the data (or "training of the noise") by too complex networks.

To increase the generalization behaviour of VLSI–friendly algorithms we show how to combine them with a density estimating preprocessing step. We assume a situation where building the net ("learning") is allowed to take some time, but execution of the net should be very fast, so it has to be realized as VLSI–circuit.

Density estimation: We formulate the strategy for classification problems. In this case the optimal strategy is to choose the class C_i with maximal probability

[*]on leave from Institut für Theoretische Physik I, Universität Münster. J.L. was supported by the British–German ARC–programme.

[†]on leave of absence from the "Politehnica" University of Bucharest. V.B. is a HC&M Research Fellow of the EC and is financed by the Commission of the European Communities under contract ERBCHBICT941741.

$P(C_i|x) \propto P(x|C_i)P(C_i)$, where x stands for the data point which has to be classified. To apply this criteria, besides the apriori probabilities $P(C_i)$, the likelihoods $P(x|C_i)$ must be known. In the case of a finite number of possible values for x, the likelihoods can be estimated by sampling during the training phase. In the case of a continuous variable x one can never sample all the values of x. Estimating $P(x|C_i)$ from a set of examples $\{x_i, C(x_i)\}$, where $C(x_i)$ denotes the class of x_i, can be done with a variety of methods known under names like "density estimation", "nonparametric regression", or "smoothing techniques". They include kernel methods, k–nearest neighbors, splines and orthogonal series estimators [7]. We use here a gaussian kernel or radial basis function [6] to approximate the likelihood on the basis of n_i d–dimensional data x_j for each class C_i by

$$\widehat{P}(x|C_i) = \frac{1}{n_i \widehat{\sigma}^d (2\pi)^{d/2}} \sum_{j=1}^{n_i} \exp\left(-\frac{1}{2\widehat{\sigma}^2}(x - x_j)^T (x - x_j)\right). \tag{1}$$

Having the estimation $\widehat{P}(x|C_i)$ for the "true" density $P(x|C_i)$ we can map to every point x of the input space the corresponding class $\widehat{C}(x)$ with maximal probability $P(C_i|x)$. The resulting classification regions can now be described by a VLSI–friendly algorithm.

A Constructive Algorithm: The principle of the constructive algorithm we use here is to divide the classification regions in as few as possible d–dimensional boxes with borders parallel to the coordinate lines [2]. Each such box is characterized by $2d$ numbers, i.e. its lower and upper bounds in each of the d dimensions. For the sake of simplicity we explain the algorithm for finding the boxes in the two–dimensional case. A number of starting boxes (=rectangles) for the algorithm are generated by discretising $\widehat{P}(x|C_i)$ on a rectangular mesh. This defines at the same time a minimal length scale and limits the maximal possible number of rectangles and therefore the complexity of the resulting net. The algorithm tries to find a small number of rectangles in the following way: It starts by enlarging a small rectangle given by the discretization step in all directions simultaneously, without leaving the region belonging to the same class as the starting point. If a rectangle cannot be enlarged further in one dimension, the border in this direction is frozen and it is enlarged only in the remaining dimension as far as possible. After the borders of a rectangle are fixed the process continues with the next small rectangle not yet processed.

This allows the construction of a VLSI–friendly net, describing the classification regions using only comparisons, which can be efficiently implemented [1], and boolean AND and OR gates. Every rectangle is represented by a group of four comparisons and three AND gates

$$\left((x^{(1)} > A) \text{ AND } (x^{(1)} < B) \text{ AND } (x^{(2)} > C) \text{ AND } (x^{(2)} < D)\right), \tag{2}$$

where $x^{(i)}, i = 1, 2$ are the components of the vector x. The total region belonging to a specific class is obtained by combining all the rectangles by OR gates. To obtain a limited fan–in the OR gates can be cascaded.

An Example: To demonstrate the method we use a noisy variant of the well–known "two spirals" problem [5]. Two probability densities were generated by superimposing Gaussians with variance $\sigma^2 = 0.1$ centered at 485 points on each

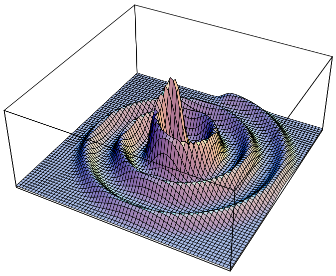
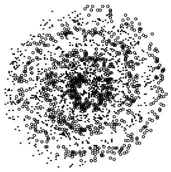

Figure 1: (left) The noisy spiral: $P(x|C_1)$ generated by a superposition of Gaussians with variance $\sigma^2 = 0.1$.
Figure 2: (right) 1000 examples drawn from $P(x|C_1)$ of Fig.1 (circles) and 1000 examples drawn from $P(x|C_2)$ (points).

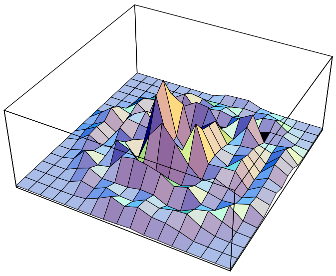
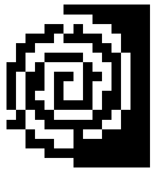

Figure 3a: (left) Reconstruction: $\widehat{P}(x|C_1)$ of the initial probability distribution $P(x|C_1)$ of Fig.1 with $\widehat{\sigma}^2 = 0.05$ after the 1000 examples of Fig.2 on a 16×16 mesh.
Figure 3b: (right) Classification regions obtained from $\widehat{P}(x|C_1)$ of Fig.3a and the corresponding $\widehat{P}(x|C_2)$.

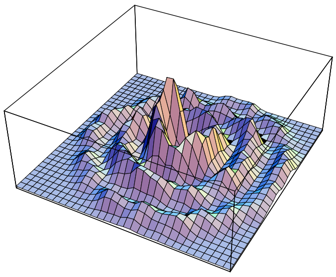
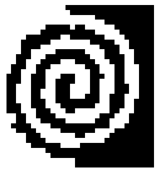

Figure 4a: (left) Same as Fig.3a, but on a 32×32 mesh.
Figure 4b: (right) Same as Fig.3a, but on a 32×32 mesh.

of two interlocking spirals. (See Fig.1.) Figs.3a and 4a show the discretized probability estimation on a 16×16 and 64×64 mesh constructed from 1000 randomly drawn examples, and Figs.3b and 4b the corresponding classification areas. The algorithm has been systematically tested for 1.) varying values of the reconstruction variance $\widehat{\sigma}^2$, 2.) varying number of training examples, and 3.) for mesh sizes of 16×16, 32×32, and 64×64. Two variables have been measured for every network: 1.) The generalization error E as the mean number of wrong classifications for 1000 newly generated test examples. 2.) The complexity of the resulting net as the number N of rectangles needed to describe the classification regions. In Figs.5a and 5b we present the results for the 64×64 mesh because there the effects of "overfitting" can be seen most clearly. The Fig.5a shows that about 100–200 examples are enough to obtain reasonable low error measures. In the region above 200 examples three regions can easily be classified. 1.) A large reconstruction width (standard deviation) $\widehat{\sigma}$ naturally leads to small networks (small N) but to large errors. In this "underfitting" area an *increase* in complexity N is coupled to a *decrease* of the error E. 2.) There is an optimal range for the reconstruction width of about $\widehat{\sigma}$=0.5 to 0.3 where the error is *minimal* and both, the error and the complexity are nearly *independent* from the number of examples and the reconstruction width. 3.) The "overfitting" area is at small reconstruction widths $\widehat{\sigma} < 0.2$ where the large 64×64 mesh already allows for "fitting of the noise". Here an *increase* of the complexity N is linked to an *increase* of the generalization error. The complexity N is proportional to the number of examples, because the network tends to memorize every example individually.

We should mention that for practical purposes a 16×16 mesh is more

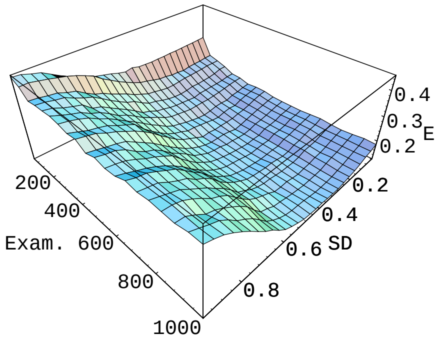

Figure 5a: Generalization error measured with 1000 new input data after n(=Exam.) training data using $\widehat{\sigma}$(=SD) as standard deviation for reconstruction on a 64×64 mesh.

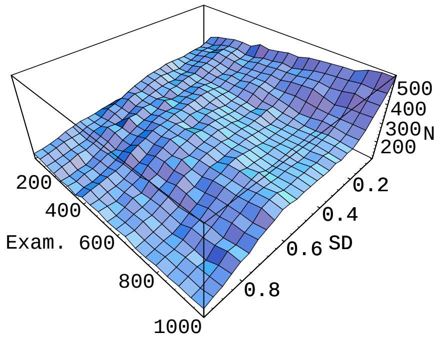

Figure 5b: Same as Fig.5a, but plotted against the number of rectangles(= N) needed to describe the classification regions.

appropriate for the "noisy spirals" if one allows slightly larger errors, because it produces much smaller nets ($N < 80$) with still reasonable minimal error ($E_{Min}(16 \times 16) = 0.201$ vs. $E_{Min}(64 \times 64) = 0.1295$) and shows nearly no "overfitting" phenomena.

References

[1] Beiu, V., Peperstraete, J.A., Vandewalle, J. and Lauwereins, R. (1993). Efficient Decomposition of Comparison and Its Applications. In Verleysen, M.(ed.): ESANN'93, D facto, Brussels, Belgium, 1993, 45-50.

[2] Beiu, V., Peperstraete, J.A., Vandewalle J., and Lauwereins, R. (1994) Learning from Examples and VLSI Implementation of Neural Networks. In R. Trappl (ed.): Cybernetics and System Research '94, World Scientific Publishing, Singapore, 1994, 1767-1774.

[3] Beiu, V. and Taylor, J.G. (1995). Optimal VLSI Implementations of Neural Networks: VLSI-Friendly Learning Algorithms. *The Applied Decision Technologies ADT'95*, London, UK, 3-5 April, 1995.

[4] Fiesler, E. (1994) Comparative Bibliography of Ontogenic Neural Networks. ICANN94, Proceedings, Springer, London, 793–796.

[5] Lang K.J., and Witbrock M.J. (1988). Learning to tell two spirals apart. *Proc. Connectionists Models Summer School*, Morgan Kauffmann, San Mateo, 52–59.

[6] Poggio, T., and Girosi, F.A. (1990). Networks for Approximation and Learning. *Proceedings of the IEEE* **78**, September 1990, 1481–1497.

[7] Silverman, B.W. (1986) *Density Estimation.* Chapman&Hall, London.

[8] Śmieja, F.J. (1993). Neural Network Constructive Algorithm: Trading Generalisation for Learning Efficiency? *Circuits, Systems, Signal Processing*,**12**(2), 331–374.