Multilayer Perceptrons Learning Control

Gilles Verley, Jean-Pierre Asselin de Beauville

Multilayer Perceptron Learning Control

Gilles VERLEY Jean Pierre ASSELIN de BEAUVILLE

Laboratoire d'informatique
Ecole d'Ingénieurs en Informatique pour l'Industrie
64, avenue Jean Portalis - Boîte 4
37913 - TOURS Cedex 9
verley@univ-tours.fr

Abstract. It has been shown that, when used for pattern recognition with supervised learning, a network with one hidden layer tends to the optimal Bayesian classifier provided that three parameters simultaneously tend to certain limiting values: the sample size and the number of cells in the hidden layer must both tend to infinity and some mean error function over the learning sample must tend to its absolute minimum. When at least one of the parameters is constant (in practice the size of the learning sample), then it is no longer justified mathematically to have the other two parameters tend to the values specified above in order to improve the solution. A lot of research has gone into determining the optimal value of the number of cells in the hidden layer. In this paper, we examine, in a more global manner, the joint determination of optimal values of the two free parameters: the number of hidden cells and the mean error. We exhibit an objective factor of problem complexity: the amount of overlap between classes in the representation space. Contrary to what is generally accepted, we show that networks usually regarded as oversized despite a learning phase of limited duration regularly yield better results than smaller networks designed to reach the absolute minimum of the square error during the learning phase. This phenomenon is all the more noticeable that class overlap is high. To control this latter factor, our experiments used an original pattern recognition problem generator, also described in this paper.

1. Theoretical introduction

We consider pattern recognition problems defined in probabilistic terms, i.e. where patterns are coded as vectors in Rn. Classes are defined by their prior probabilities and their class-conditional probability densities. In such a framework, the optimal classifier classifies any observation in the class having highest posterior probability. Let us sum up the conditions under which a multilayer network with one hidden layer tends to this optimal classifier.

1.1. Definitions

- Let A be a learning sample taken in a population of patterns according to some probability distribution. Each vector X_i is a realization of one of the classes defined by the problem's distributions.
- Let F be a set of functions mapping R^n to values in the interval [0, 1] in R and f be any function in this set.
- By convention, the value 1 is associated to the vectors in the learning sample that

are a realization of class C_k and the value 0 to all other vectors.

$$V(X_i) = 1 \text{ if } X_i \in C_k \quad \text{and} \quad V(X_i) = 0 \text{ if } X_i \notin C_k$$

- $D_A(f)$ denotes the sum, for all vectors in the learning sample, of the squared differences between the values associated to these vectors and the values given by function f.

$$D_A(f) = \sum ((V(X_i) - f(X_i))^2)$$

1.2. Known results

If $f_{optimal}$ denotes the function belonging to F that minimizes in an absolute sense $D_A(f)$, then it has been shown that $f_{optimal}$ can be interpreted as an estimate of the posterior probability of the class that it codes by convention. In addition, $f_{optimal}$ is the best estimate in set F of po7sterior probabilities when the size of the learning sample tends to infinity [HAMPSHIRE and PEARLMUTTER, 1991].

We then consider the sequence of sets G_k of functions g_k realized by a neural network having k hidden cells with a sigmoid transfer function and one output cell that codes g_k . It has been shown that, under such conditions, any continuous function can be approximated by a function g_k from G_k such that the approximation error tends to zero as k tends to infinity [WHITE 1990].

It immediately follows that the outputs of a multi-layer network tend to the posterior probabilities of the classes they code provided that both the size of the learning set and the number of hidden cells tend to infinity and that the error function tends to its absolute minimum. Then the multi-layer network tends to the optimal classifier.

1.3. The problem of overlearning

When the size of the learning set is finite, minimizing $D_A(f)$ no longer ensures the optimality of the Bayesian classifier, as Figure 1 illustrates. Figure 1 gathers results from a number of artificially constructed problems [VERLEY, 1994]. The various plots represent observations made during the learning phase of a 20 hidden cell network trained by the backpropagation algorithm. The y axis gives the mean error due to estimating posterior probabilities on a test sample; this quantity may be seen as a distance from the optimal classifier. The x axis represents the mean square error over the learning sample; as this quantity monotonically decreases during the learning phase, it may be seen as a qualitative measure of the duration of the learning phase or of the number of iterations of the learning algorithm. Thus Figure 1 shows that, during the learning phase, the network tends to get closer to the optimal classifier, up to a certain threshold after which the learning phase has the opposite effect, hence the expression *overlearning*. The overlearning threshold is all the more quickly reached that the sample size is small, which is remarkably consistent with theoretical results.

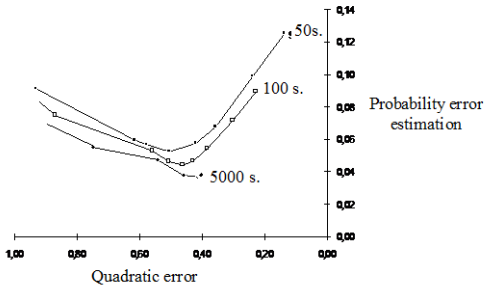

Fig 1. Posterior probability estimation error with the MLP for several sample sizes.

1.4. Solutions to a problem

To avoid the overlearning phenomenon, the usual solution is to give an upper bound to the number of hidden cells with respect to the sample size. That limits the solution space. It is estimated that the sample size should be at least ten times the number of weights in the network [BAUM et HAUSSLER, 1989].

It should be noted that the option of limiting the network size with a systematically maximum learning, i.e. such that it tends to the absolute minimum of the error, bears a symmetric option: that of limiting the duration of the learning phase for a network whose size is systematically maximum (as far as the available computing resources permit).

Theoretically this latter option qualifies as much as the first as a way of solving the overlearning problem. Our hypothesis is that this other option gives at least as good results as the first one for a relevant set of problems. It opens up interesting angles for the use of dedicated, highly parallel neural machines with a unique network of maximum size for which the duration of the learning phase would be adjusted to the size of the sample and to information on the intrinsic complexity of the problem to be solved.

2. A generator of samples

In order to investigate our hypothesis in a scientifically sound way, we propose a model, both mathematical and computer-based, that allows us, in theory, to construct any pattern recognition problem given in probabilistic terms with an accuracy as high as desired. The computer implementation of this model provides a generator that can produce all data necessary to the systematic study of the behavior of pattern recognition systems on sets of artificially constructed problems. Such sets of problems are approximations to the set of all mathematically specified pattern recognition problems.

Let f be a probability density function that is continuous on a bounded representation space. Consider also a finite sample having such density function. It has been demonstrated that it is possible to produce an estimate of f that is convergent and asymptotically unbiased from this sample by using Parzen's estimator [PARZEN, 1962].

Consider now the observations in the representation space. Formally they can be seen

at the center of adjacent, identical hypercubes. We call such a point a *prototype*. If we associate each prototype to a Parzen kernel, we naturally obtain the same estimator of f as above. Symmetrically, given a partition of R^n by adjacent hypercubes, a certain number of distinct prototypes can be arbitrarily assigned to a class. Each prototype is associated to a Parzen kernel. We then consider the function mapping any point in the representation space to the weighed sum of the contributions of the various prototypes assigned to a given class. It can be shown that this function is a probability density function. Thereby we obtain a generator capable of constructing diverse probability density functions by changing the combination of prototypes assigned to a class. Samples of any size can then be generated directly from these functions.

In the case of a pattern recognition system whose outputs can be seen as the estimates of density functions, one can compute a mean error over a testing sample, i.e. the square difference between the estimation and the real probability density function. This feature has been used to produce Figure 1, which exhibits the phenomenon of overlearning with a finite sample. Elsewhere, we demonstrated that any probability density function can be approximated as closely as desired by a function constructed by our generator. From a practical viewpoint, we have a generator for probability density functions that can be used to validate pattern recognition systems by experimentation. Each generated problem is thus explicitly defined by the probability density functions of the different classes constructed by the generator, as well as by the classes' prior probabilities. Given these specifications, it is possible to design an algorithm that generates samples of any size for the learning and testing phases of the problem at hand. Given a set of parameters, the generator can produce as many problems as desired together with the corresponding samples. Experimenting with a large number of such problems will increase the reliability of the average experimental results as well as their statistical soundness.

3. Problem construction and sample generation

In this section, we show how a problem can be constructed in a simple language and how samples of any size can be generated for this problem. Since the problem is fully defined in probabilistic terms, it is possible:

- to generate the Bayesian boundaries for any kind of system,
- to know the posterior probability of the different classes in any point.

We are therefore capable of comparing the hit rate of any classifier, in both the learning and testing phases, with the optimal rate of the Bayesian classifier.

When the classifier under scrutiny realizes a partition in a representation space of dimension less than 3, the boundaries generated by the classifier and the Bayesian boundaries can be visualized together.

We now show how the specification of the problem in a simple language can be used first to derive the analytical form of the problem in probabilistic terms and then to generate labelled samples.

- parameters of the representation space:
  number of dimensions: d
  bounds of the representation space for each dimension: $\{[\min_i, \max_i]\}\ \forall i = 1,..,d$
- parameters of the generator:
  number of hypercubes for each dimension $d_i \quad \forall i = 1,..,d$

variance of the Gaussian distribution associated with each prototype of the different classes: σ (we consider here the case of a diagonal matrix of variances-covariances).

The latter two parameters determine classes of problems, both from a mathematical and a computing viewpoints, and in fact characterize their complexity. For a given class of problems, the larger the number of hypercubes is, the more problems in this class have non-linear, thus complex, Bayesian boundaries. As the variance increases, so does the overlap between classes, i.e. the imperfection of the discrimination. This is why we vary this latter parameter in order to test the hypothesis stated above.

- problem structure:
  number of classes: C
  definition of the prototypes assigned to each class: lists $S_c \quad \forall c = 1,..,C$ in increasing order of the classes assigned to the different hypercubes.

The ratio of the number of hypercubes assigned to some class to the total number of hypercubes can be interpreted as a kind of measure of the homogeneity of the mixing distribution of the different classes.

The lists S_c may be either user-defined when the user wants to test a specific problem, or defined by a random sampling according to an experimentation plan when a more general hypothesis is tested on a set of random problems. We relied on this latter feature to produce some of our experimental results.

- parameters of the samples to be generated:
  sample cardinality: n
  number of samples to be generated: e

All these parameters are given in a text file in any order. A parser reads the file and extracts the parameters. Furthermore it is possible to give several values for any parameter to facilitate the generation of samples in an experimentation plan. When the problem constructed is two-dimensional, it is also possible to assign the hypercubes to the different classes via a graphical interface.

3.1. Analytical formulation of the problem

A problem specified along the lines described above can easily be reworded in probabilistic terms:

Let $E[\min_1, \max_1] \times ... \times [\min_d, \max_d]$ be a representation space.

N_p points $\underline{x}^{(j)}$ are defined as follows:

$$x_i^{(j)} = \min_i + \frac{1}{2} p_i + \left(\frac{j-1}{\prod_{k=1}^{i-1} d_k}\right) p_i \quad \forall i = 1,..,d \text{ et } \forall j = 1,..,n_p$$

with $p_i = \frac{\max_i - \min_i}{d_i} \quad \forall i = 1,..,d$

$n_p = \prod_{i=1}^{d} d_i$ where d_i is the number of hypercubes in dimension i

These points $\underline{x}^{(j)}$ are the centers of the hypercubes. A number of them are assigned to classes in lists S_c.

Let m_k denote these centers drawn from the $\underline{x}^{(j)}$. So k belongs to a subset of the

values taken by j. The points m_k are the class prototypes.
A prototype is then characterized by its center m_k and a distribution L of center m_k :

$$L_k(\underline{x}) = (2\pi)^{-\frac{d}{2}} \left|\sigma^2 \underline{I}\right|^{-\frac{1}{2}} e^{-\frac{1}{2}(\underline{x}-\underline{m}_k)^t (\sigma^2 \underline{I})^{-1} (\underline{x}-\underline{m}_k)} = N(\underline{m}_k; \sigma^2 \underline{I})$$

where $\underline{I}$ is the identity matrix of order d.
Therefore the density function for a class w_c is:

$$f^{(c)}(\underline{x}) = \frac{1}{\text{card}(k_c)} \sum_{i \in k_c} N(\underline{m}_i; \sigma^2 \underline{I})$$

where k_c is the set of prototype numbers assigned to class c classes by list S_c
In order to define the problem in probabilistic terms, we also need to know the prior probabilities of the classes. Let P_c be the prior probability of class w_c:

$$P_c = \frac{\text{card}(k_c)}{\sum_i \text{card}(k_i)}$$

Because we have a complete, probabilistic analytical form for problems that can be as general as desired, we can determine the theoretical Bayesian boundaries as well as the posterior probabilities of the different classes for any point of the representation space. The last step involves generating the samples for these problems.

3.2. Generation sample

Let T be the union of the lists S_c containing all the prototypes assigned to the various classes. Each element in T is a pair (C, $\underline{M}$) where C is the class assigned to the prototype and $\underline{M}$ is the coordinates of the prototype. For each element X in the sample, an element of T is drawn at random (uniform distribution) Call this element (c, $\underline{m}$) . The coordinates $\underline{x}$ of X are drawn according to a Gaussian distribution N($\underline{m}$, $\sigma\underline{I}$) .Consequently $\underline{x}$ determines an element conforming to the analytical model described in the previous section.
We illustrate the features of our generator by giving below two examples of two-dimensional problems (Figures 2 and 3). In the first example (Figure 2), we have a three class problem, a small variance for the Gaussian distributions and prototypes that define sharply separated, connected sets. We visualize the corresponding 1000 point sample and the optimal boundaries as computed in theory. These boundaries are piecewise linear and the sample is almost piecewise linearly separable given the small overlap. Therefore this is an objectively simple problem. In the second example (Figure 3), on the contrary, we have a two-class difficult problem such that the optimal boundaries are highly non-linear and such that the overlap is rather large. Note that, in order to obtain a complex problem of this kind, the generator must be set up with a higher number of hypercubes than in the previous problem. Furthermore, all hypercubes have been assigned to the two classes, which implies that the overall distribution of points is more homogenous than it previously was. Lastly it must be pointed out that the generator is not confined to two-dimensional problems. We are presently working toward visualization of three-dimensional problems. For higher dimensions, the results are necessarily in numerical forms.1

Commentaire [U1]: Il s'agit donc de définir une procédure qui, à chaque exécution, renvoie la valeur d'une classe et les coordonnées d'un point tel que l'on puisse affirmer que ce point classé a été tiré selon les lois de probabilité définies préalablement.
Soit **T** la liste qui est l'union des listes $\mathbf{S_c}$ de tous les prototypes affectés aux différentes classes. Chaque élément de cette liste est un doublet (classe affectée au prototype, coordonnées du prototype). On tire un élément de **T** selon une loi uniforme. Soit **(c,$\underline{\mathbf{m}}$)** cet élément. On tire alors les coordonnées d'un point selon une loi normale multivariée **L($\underline{\mathbf{m}}$, σ $\underline{\mathbf{I}}$)**. Soit $\underline{\mathbf{x}}$ le point ainsi déterminé. $\underline{\mathbf{x}}$ **et c** déterminent complétement un élément conforme au modéle analytique décrit au paragraphe précédent.

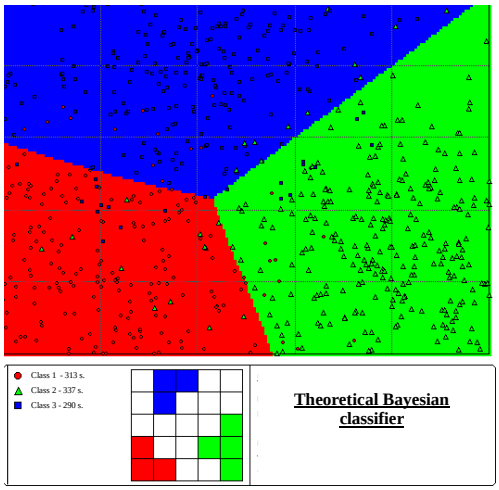

Fig.2 A three class, easy problem

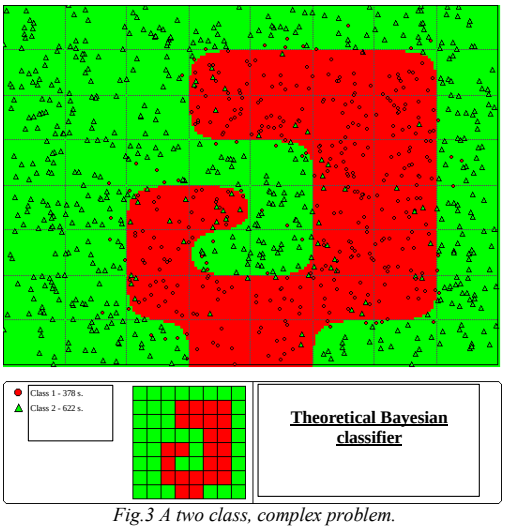

Fig.3 A two class, complex problem.

4. Experimental results

With the generator described above, it is possible to create classes of problems that have a variable complexity (in the sense of overlap between the classes) together with the corresponding learning and testing samples. 40 learning samples were thus generated from 4 problems, each problem having a distinct overlap value. These samples were systematically tested on networks with one hidden layer and a varying number of hidden cells (4 possible values). The maximum size we tested was 50 hidden cells. This value was deliberately very large with respect to the maximum size

of the learning samples, which is 500, and to the values usually found in the literature. The ratio of the size of the sample to the number of weighs in the network is here no higher than 2.5 whereas 10 is usually recommended. At regular intervals (5) during the learning phase of each network, the performance of the trained network was tested on the corresponding test sample. The systematic combination of the various factors yielded an experimentation plan with 3200 performance measures.

For each sample the network was trained with, a size was selected that had best generalization performance for **a learning phase of maximum duration** and, for each overlap value, the average was computed over all the samples of optimal size and of corresponding performance. In a similar way, for each problem, a learning duration was selected that provided the best generalization performance **for a network of maximum size**. For each overlap value, the average was computed on all problems having optimal duration and corresponding performance.

Our experimental results are summarized in three figures, each point in these figures merging several hundred experiments. The first figure (Figure 4) illustrates the effect of class overlap on the optimal number of hidden cells in a network with maximum learning duration. The average performance of these optimal networks is also represented. It is to be noted that the overlap value is inversely proportional to the optimal size. This is quite consistent with theory since overlearning is especially probable when both class overlap and the space of solutions learnable by a large-size network are large. In such a case, the network will comprise many weights that enable it to learn by rote the ambiguous examples ; the generalization performance will degrade above a certain threshold. If the classes are not ambiguous at all, the over-sizing of the network does not matter since there will exist no example to "trick" the network.

The second figure (Figure 5) illustrates the dual phenomenon for a deliberately too large network whose learning duration is constant. We see that the larger the overlap, the shorter the optimal duration of the learning phase. Again this is consistent with theory in the sense that overlearning is all the more probable when both class overlap and the space of effectively considered solutions during learning are large. In this case, the network will have enough time to learn by rote the ambiguous examples and the generalization performance will degrade.

The third figure (Figure 6) illustrates an unexpected phenomenon : the usual approach relying on network size regularly gives slightly worse performance than our new approach that relies on learning duration. And this is all the more noticeable that overlap is large, i.e. the risk of overlearning is high. We propose the following tentative theoretical explanation : it is easier to construct a network of very large size than it is to minimize the absolute mean square error by controlling the network size. As is well known in the community, it cannot be guaranteed that, even after a lengthy learning phase, the minimum of the mean square error has been correctly approximated.

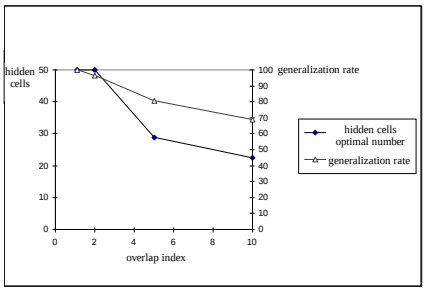

Fig. 4 Consequence of class overlap on the optimal number of hidden cells

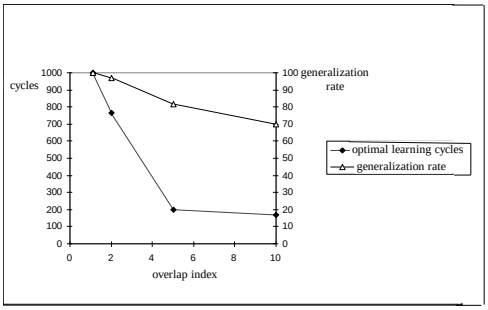

Fig.5 Consequence of class overlap on the optimal number of learning cycles

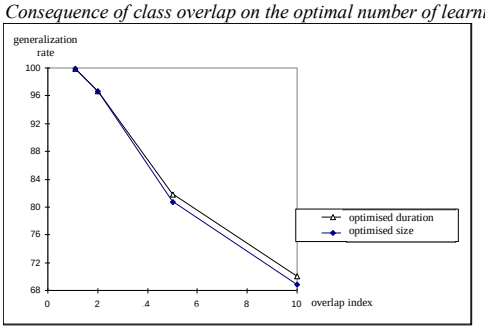

fig. 6 Compared performances in function of overlap

5. Conclusion

It is generally acknowledged that the optimal architecture of a multi-layer network is "context dependent". The experiments described in this paper reveal an objective factor of the context that determines the optimal size: the amount of overlap between classes. Even more interestingly, we note that, for a given network, the optimal duration of the learning phase depends, in the same way, on class overlap. In other words, optimizing the learning duration yields results that are at least as good as those due to optimizing the network architecture. We therefore reach the somewhat paradoxical conclusion that the most ambiguous problems are better solved by large,

poorly trained networks.

6. Bibliographic references

BAUM, E. On the capabilities of multilayer perceptrons. *Journal of complexity,* vol. 4, p. 193-215. 1988.

BAUM, E., HAUSSLER, D. What size net gives valid generalization?. *Neural Computation,* vol. 1, n° 1, p. 151-160. 1989.

BLUM, A., LI, L. K. Approximation theory and feedforward networks. *Neural networks*, vol. 4, n° 4, p. 511-516. sept. 1991.

FAHLMAN, S. E. An empirical study of learning speed in back-propagation netwworks. *Technical report n° CMU-CS-88-162*, Carnegie Mellon university, computer science department. juin 1988.

FUNAHASHI, K. I. On the approximate realization of continuous mappings by neural networks. *Neural networks,* Vol. 2, p. 183-192. 1989.

HAMPSHIRE, J.B., PEARLMUTTER, B.A. Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function. *Proc. 1990 Connectionnist Models Summer School*, Morgan Kaufmann, 1991.

HAUSSLER, D. Quantifying inductive bias: AI Learning algorithms and valiant's learning framework. *Artificial Intelligence 36*, p. 177-221. 1988.

HECHT-NIELSEN, R. *Neurocomputing*, Addison-Wesley. 1990

HORNIK, M., STINCHCOMBE, M., WHITE, H. Multilayer feedforward networks are universal approximators, *Neural networks*, vol. 2, p. 359-366. 1989.

KANAYA, F., SHIGEKI, M. Bayes statistical behavior and generalization of pattern classifying neural networks. *Cognitiva 90*, North-Holland: eds T. Kohonen and F. Fogelman-Soulié, p. 35-44. 1991.

MOODY, J.E. The effective number of parameters : an analysis of generalization in non linear learning systems. *NIPS 4*, p. 847-855. 1992.

PARZEN, E. An estimation of a probability density function and mode. *Ann. Math. Statist.,* Vol. 33, p. 1065-1076. 1962.

PAUGAM-MOISY, H. A Selecting and parallelizing neural networks for improving performances. *Artificial Neural Networks*, Kohonen & al. editors, Elsevier Sc. Pub., North-Holland, vol. I, p. 659-664. Juin 1991.

VALIANT, L. G. A theory of the learnable. *Communications of the ACM 27,* 1134-1142, 1984.

VAPNIK, V. N. *Estimation of dependences based on empirical data.* Springer series in statistics, Springer Verlag. 1982.

VERLEY, G., ASSELIN DE BEAUVILLE, J.P., RAMAT E., LECLERC N. Différences théoriques et expérimentales entre les réseaux de neurones multicouches et le classifieur bayésien optimal. *Actes du colloque sur le Neuromimétisme, Lyon*, p. 217-220. Juin 1994.

WHITE, H. Learning in artificial neural networks: a statistical perspective. *Neural comp. 1*, p. 425-464, 1989.