数理解析研究所講究録
1127 巻 2000 年 27-35

Bootstrap Training for Neural Network Learning

Georges Dupret and Masato Koda
University of Tsukuba

Abstract

We analyse the impact of re-sampling on the ability of artificial neural networks to correctly learn a binary classification problem. We use the bootstrap expression of the prediction error to identify the optimal re-sampling proportions.

1 Introduction

Since a neural network minimizes an overall error, the proportion of types of data in the training set is critical. A network trained on a data set with 900 good and 100 bad cases will bias its decision towards good cases, as this allows the algorithm to lower the overall error (which is much more heavily influenced by the good cases). If the representation of good and bad cases is different from the real population, the network's decisions may be biased. A typical example would be disease diagnosis. Perhaps 90% of patients routinely examined are clear of a disease. A network is hence trained on an available data set with a 90/10 split. It is then used in diagnosis on patients complaining of specific physical problems, where the likelihood of disease might be 50/50. The network will naturally fail to recognize disease in some unhealthy patients. In contrast, if trained on the "complainants" data, and then tested on "routine" data, the network may raise a number of false positives. In such circumstances, the training set may need to be re-sampled to take account of the distribution of data (i.e. replication of the less numerous cases, or removal of some of the numerous cases). A common practice is to re-sample the training set so that there is the same number of patterns for each class. The question we address here is whether this is appropriate for empirical learning of neural networks.

This paper is organized as follows: First, we present the bootstrap formulation of the problem. Then we run a numerical experiment to asses empirically the impact of re-sampling on the network ability to learn.

2 Bootstrap

The *Bootstrap techniques* (see [1]) were introduced in 1979 as a computer-based method for estimating the standard error of empirical distributions. The method enjoys the advantage of being completely automatic and not requiring theoretical computations or assumptions on the original distributions. It was further extended to estimate prediction error.

2.1 Definitions

- Let $\mathbf{x}_i = (\mathbf{I}_i, O_i^d)$, $i = 1, ..., n$ be the i^{th} element (pattern) of the training set $\mathbf{x}$. $\mathbf{I}_i$ is an input vector and O_i^d is the desired output as opposed to the actual output of the network O_i.

- A bootstrap sample $\mathbf{x}^*$ has n elements, generated by sampling with replacement n times from the original data set $\mathbf{x}$. For example, if $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$, a possible bootstrap re-sampling may result in $\mathbf{x} = \{\mathbf{x}_3, \mathbf{x}_3, \mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_2\}$.

- Having observed a random sample of size n from a probability distribution F, the *empirical distribution function* $\widehat{F}$ is defined to be the discrete distribution that puts probability $1/n$ on each pattern $\mathbf{x}_i$.

- A *plug-in* estimate of a parameter $\theta = t(F)$ is defined to be $\widehat{\theta} = t(\widehat{F})$. In other words, we estimate the function $\theta = t(F)$ of the probability distribution F by the same function of the empirical distribution $\widehat{F}$, $\widehat{\theta} = t(\widehat{F})$. For example, if we consider the mean of the desired values, it is defined as $\theta = E_F(O^d) = \frac{1}{N}\sum_{i=1}^{N} O_i^d$. In the same manner, the plug-in estimate of the mean is $\widehat{\theta} = E_{\widehat{F}}(O^d) = \frac{1}{N}\sum_{i=1}^{N}(O_i^d)^*$. In the above example, $E_F(O^d) = \frac{1}{5}\sum_{i=1}^{5} O_i^d$, and its plug-in estimate is: $E_{\widehat{F}}(O^d) = \frac{1}{5}\sum_{i=1}^{5}(O_i^d)^* = \frac{1}{5}(2O_3^d + O_1^d + O_4^d + O_2^d)$.

- Suppose we train the network on the patterns contained in $\mathbf{x}$, producing a predicted value O_0 for the input $\mathbf{I} = \mathbf{I}_0$. We write: $O_0 = f_{\mathbf{x}}(\mathbf{I}_0)$, where

O_0 is the output of the network trained with the set $\mathbf{x}$ and presented with the input $\mathbf{I}_0$.

- $Q[O^d, O]$ denotes the measure of error between the desired output O^d and the prediction O. In the case of classification, a common measure of error is $Q[O^d, O] = 0$ if $O^d = O$ and 1 otherwise.

2.2 Prediction Error

Let $(\mathbf{I}_0, O_0^d)$ denote a new observation (i.e. a new pattern) from F, the complete population of patterns. The prediction error for $f_{\mathbf{x}}(\mathbf{I}_0)$ is defined by

$$err(\mathbf{x},F) \equiv E_F\{Q[O_0^d, f_{\mathbf{x}}(\mathbf{I}_0)]\} \tag{1}$$

where the notation E_F denotes the expectation over a new observation.

On the other hand, the plug-in estimate of $err(\mathbf{x},F)$ is given as:

$$err(\mathbf{x}^*,\hat{F}) = \frac{1}{n}\sum_{i=1}^{n} Q[O_i^d, f_{\mathbf{x}^*}(\mathbf{I}_i)] \tag{2}$$

In this expression, $f_{\mathbf{x}^*}(\mathbf{I}_i)$ is the predicted value at $\mathbf{I} = \mathbf{I}_i$, based on the network trained with the bootstrap data set $\mathbf{x}^*$.

We could use $err(\mathbf{x}^*,\hat{F})$ as an estimate of the prediction error, but it involves only a single bootstrap sample and hence is prone to be biased. Instead we focus on the average prediction error. The approximation to the prediction error is an average on B bootstrap samples and n observed patterns:

$$E_{\hat{F}}[err(\mathbf{x}^*,\hat{F})] = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{n} Q[O_i^d, f_{\mathbf{x}^{*b}}(\mathbf{I}_i)]/n \tag{3}$$

If the distribution F is known and finite, the n observed patterns are replaced by the complete population and the prediction error of the network trained on bootstrap samples is:

$$E_F[err(\mathbf{x}^*,F)] = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{n} Q[O_i^d, f_{\mathbf{x}^{*b}}(\mathbf{I}_i)]/n \tag{4}$$

In our bootstrap experiment, we will estimate the average prediction error for various schemes of re-sampling of the data set.

3 Numerical Experiments

3.1 Patterns

Each pattern is composed of six inputs, arranged as a vector, and one output.

3.1.1 Inputs

All elements in the input can take the values in $\{0, 1\}$ exclusively:

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \epsilon \\ \varepsilon \end{pmatrix} \text{ where } \alpha, \beta, \gamma, \delta, \epsilon, \varepsilon \in \{0, 1\} \tag{5}$$

The pattern is said to be symmetric if it has the form:

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \gamma \\ \beta \\ \alpha \end{pmatrix} \text{ where } \alpha, \beta, \gamma \in \{0, 1\} \tag{6}$$

and it is said to be asymmetric otherwise. There is $2^6 = 64$ different vectors in the complete sample space and $2^3 = 8$ symmetric vectors. There is $64-8 = 56$ asymmetric vectors.

3.1.2 Outputs

If the input vector of the pattern is symmetric, the output is 1. It is 0 otherwise. This is a symmetry detection problem for 6 bits code.

3.2 Architecture and Training Algorithms

We trained feed-forward neural network with one hidden layer of $3 \sim 10$ neurons as varying parameters, using back-propagation ([2]) and a conjugate gradient algorithm ([6] and [7]). A stochastic noise based algorithm proposed by the second author was also experimented (see [5]; [3] and [4]). The results presented in the next experiment is the 6 hidden neurons' case with conjugate gradient. The results did not differ significantly and were quite independent from the architecture and the learning algorithms.

3.3 Re-sampling Scheme

The complete patterns in the sample space consist of 8 symmetric vectors and 56 asymmetric vectors for a total of 64. The proportion of symmetric

vectors is 8/64. The re-sampling scheme will modify systematically this proportion to create different training sets on which the network learning ability is assessed. We will:

1. decide a proportion (for example 40/64), then
2. take randomly with replacement 64 vectors such that the proportion decided in step 1 is respected: if the proportion is 40/64, pick randomly 40 times a vector in the set of symmetric vectors and take randomly 24 (= 64 − 40) asymmetric vectors to complete the training set.

A second experiment is also undertaken where we apply duplicated bootstrapping: Instead of a re-sampling set of 64 vectors, we take 128 (= 64 × 2) vectors.

3.4 Experiment

We re-sample the sample space in order to assess the network learning ability. In the experiment,

1. The proportion ranges the values from 4 to 60 by step of 4.
2. For each proportion, we construct 100 re-sampling sets.
3. For each of these sets, we estimate the weights of the network. Each network must converge on the bootstrap set.
4. For each network, we compute the bootstrap prediction error on the original sample set as expressed in Eq. (4):

$$E_F[err(\mathbf{x}^*,F)] = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{n} Q[O_i^d, f_{\mathbf{x}^{*b}}(\mathbf{I}_i)]/n$$

This experiment allows us to address the following issue: if we know the true population F of the patterns, how should we sample in order to optimize the empirical learning of neural networks in general?

3.5 Results

The results are presented graphically in Figs. 1 and 2 for cases re-sampling 64 and 128 vectors, respectively. The vertical axis denotes the mean number of miss-classified patterns when the network that learned the training set

was tested on the complete sample space. If the error is 0, it means that the network ability to generalize is perfect. The standard deviation of this error is also presented under the form of a box surrounding the mean. The horizontal axis denotes the proportion of symmetric vectors in the training set. When the abscise is 48, it means that the 64 patterns of the training set consists of 48 symmetric and 16 asymmetric vectors.

The classification error attains a minimum - corresponding to a maximum in the generalization ability - around values for the proportion between 12/64 and 20/64. The same observation repeats when we re-sample 128 vectors. In that case, the optimal proportion seems to lie in the range between 24/128 and 40/128. Clearly the optimal proportion is neither the original distribution in the sample space nor the 50%-50% proportion often applied in practice for binary classification problems. Rather, it lies somewhere between these two values.

One might remark that when re-sampling *with replacement*, the number of different patterns in the training set varies. This certainly has an impact on the network performance on the sample space. We plotted in Figs. 3 and 4, the mean number of different patterns for each re-sampling proportion.

We observe that this mean number is higher when the original proportions in the sample space are respected. We may conjecture that even though there is fewer different patterns to learn from, the network still perform better where the proportion is optimal. This indicates the importance of identifying the optimal proportion when training a network.

4 Conclusions

We presented the bootstrap expression of the prediction error to base our work on sound statistical theory. Then, we set up numerical experiments meeting the following requirements:

1. all sample space is known,
2. it is finite and small so that training is fast and easy,
3. its distribution function is unbalanced so that the effects of re-sampling are easy to assess.

We assessed empirically the impact of re-sampling on the network ability to learn and the importance of the re-sampling proportion. In binary classification problems, it has been a common practice to present networks to be trained with an equal number of patterns in each class, irrelevant of the

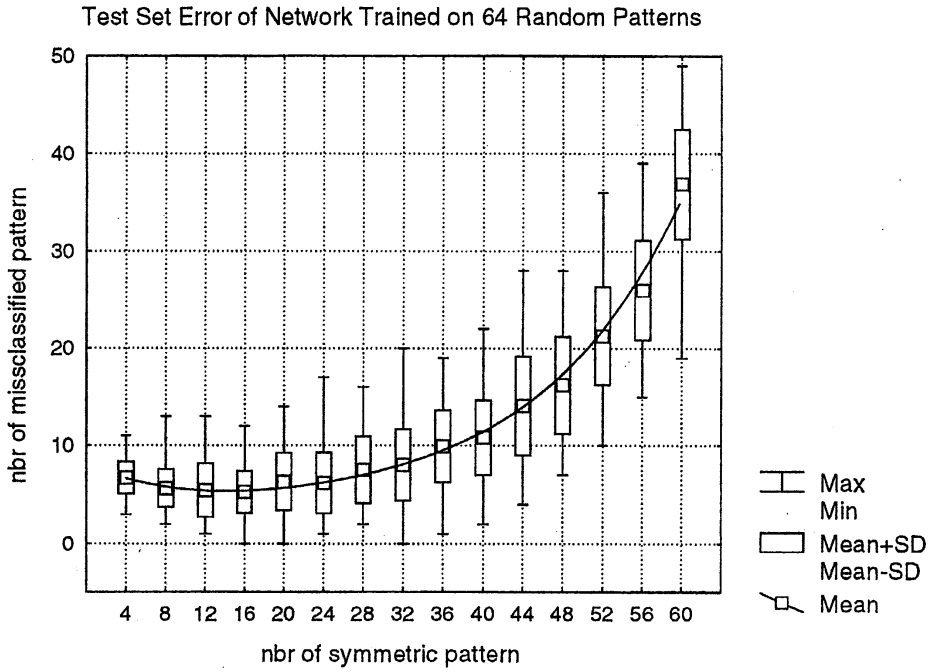

Figure 1: Miss-classification Error for 64 Patterns

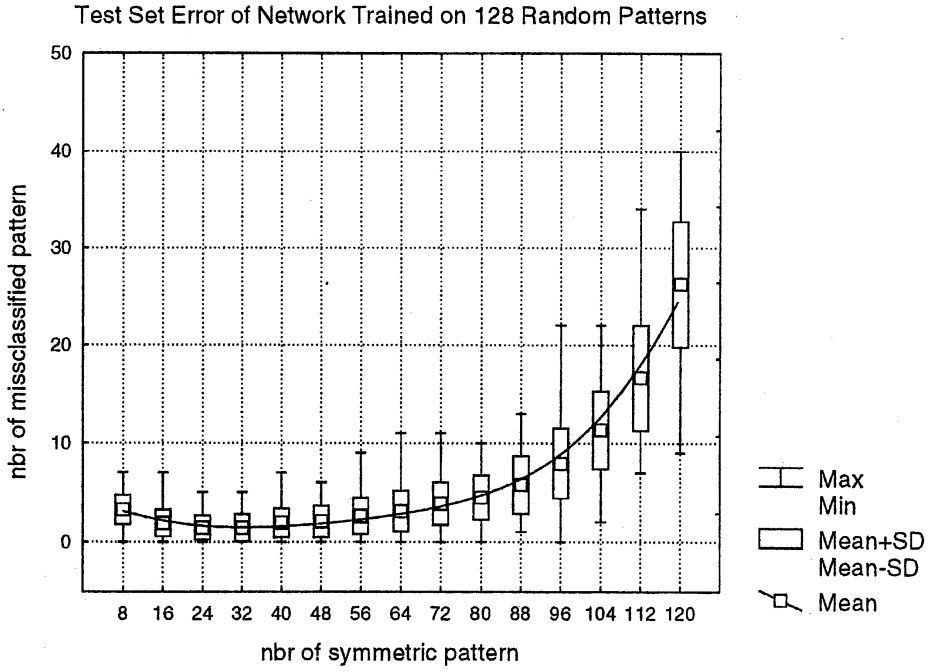

Figure 2: Miss-classification Error for 128 Patterns

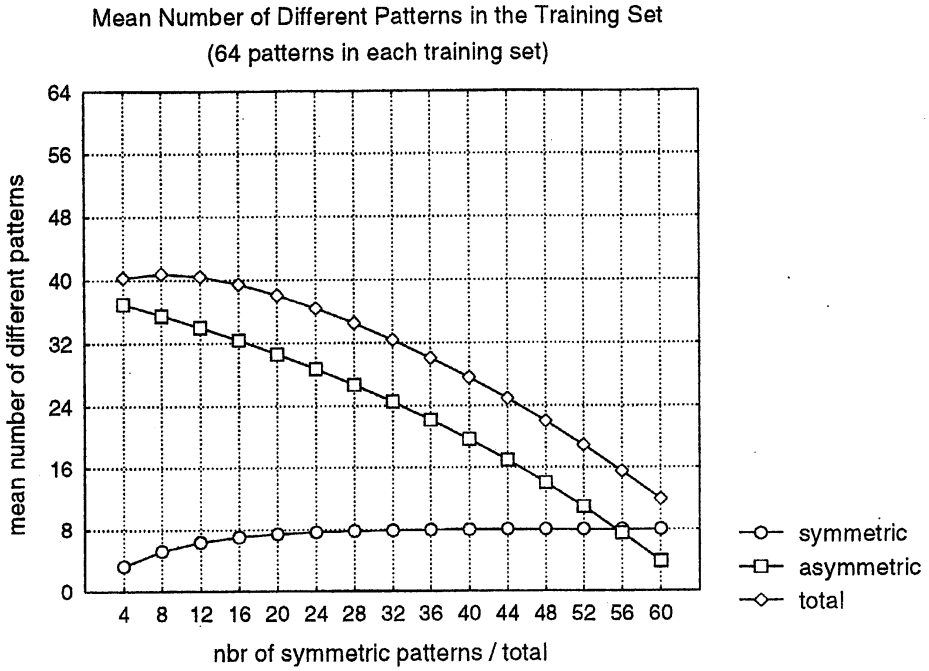

Figure 3: Mean Number of Different Patterns for 64 Patterns

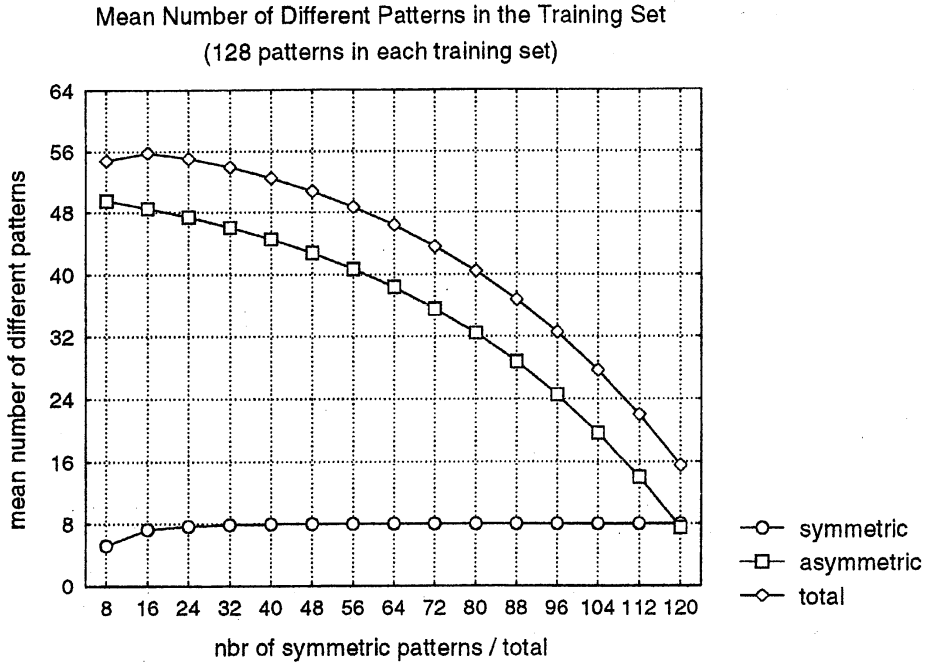

Figure 4: Mean Number of Different Patterns for 128 Patterns

original distribution. The numerical results of this paper indicate that the learning ability of the network is indeed enhanced by re-sampling, but only a few repetition of the scarce patterns may be a most efficient scheme.

References

[1] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap.* New York ; Tokyo : Chapman & Hall, 1993.

[2] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Redwood City, Calif. : Addison-Wesley Pub. Co., 1991.

[3] Masato Koda. Stochastic sensitivity analysis method for neural network learning. *Int. J. Systems., Vol. 26, No. 3*, pages 703–711, 1995.

[4] Masato Koda. Neural network learning based on stochastic sensitivity analysis. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 27,*, pages 132–135, 1997.

[5] Masato Koda. Stochastic sensitivity analysis and Langevin simulation for neural network learning. *Reliability Eng. and System Safety, Vol. 17*, pages 71–78, 1997.

[6] Timothy Masters. *Advanced Algorithms for Neural Networks.* John Wiley & Sons, New York., 1993.

[7] Timothy Masters. *Practical Neural Network Recipes in C++.* Academic Press, New York., 1993.