No. 853

BOOTSTRAP RE-SAMPLING AND CROSS-VALIDATION FOR NEURAL NETWORK LEARNING

by

Georges Dupret and Masato Koda

March 2000

BOOTSTRAP RE-SAMPLING AND CROSS-VALIDATION FOR NEURAL NETWORK LEARNING

Georges Dupret and Masato Koda

Institute of Policy and Planning Sciences
University of Tsukuba
1-1-1 Tennoudai, Tsukuba Science City, 305-8573 Japan
Tel: +81-298-53-5222
Fax: +81-298-55-3849
koda@shako.sk.tsukuba.ac.jp

Abstract

A technical framework to assess the impact of re-sampling on the ability of a neural network is presented to correctly learn a classification problem. We use the bootstrap expression of the prediction error to identify the optimal re-sampling proportions in a numerical experiment with binary classes and propose a new, simple method to estimate this optimal proportion. An upper and a lower bounds for the optimal proportion are derived based on Bayes decision rule. The analytical considerations to extend the present method to cross-validation are also illustrated.

1 Introduction

In data mining applications, the proportion of types of data in the training data set has a critical importance (for example, [8], [17], [18], etc.). Neural network or classification tree algorithm that is trained on a data set with 950 good and 50 bad cases, for example, will bias its decision towards good cases, as this would allow the algorithm to lower the overall error (which is much more heavily influenced by the good cases). By extracting and comparing the features which characterise these good and bad cases in the training sample, we can create a predictive model. Therefore, when the representation of good and bad cases in the training sample is unbalanced, the model's decisions may naturally be biased.

In order to deal with this sort of problems associated with unbalanced data records, recent advanced data mining tools are equipped with sophisticated sampling techniques for creating training and test data sets (see [8], [17], [18], etc.). These techniques are usually referred to as the *enriched sampling* or *balanced sampling*, and they create a training data set with an approximately equal number of good and bad cases. It is believed that a balanced training sample improves the generalisation ability of the tool because it helps the identification of the characteristics of the scarce class. The test data set for a balanced sample, however, is created randomly to maintain the validity of the test data set and, accordingly, to reflect the original distribution of the total records. The techniques are especially effective when using classification trees to create a predictive model, since the tree algorithm usually does not run if the data does not contain enough of the predicted behaviour in the data set.

This paper will address the effects the unbalanced data would have on supervised learning and, in particular, on binary classification problems. Our approach is based on the well-known bootstrap analysis techniques [4]. Hence, in this study, the training data set is re-sampled to take account of the unbalanced distribution of the original records, i.e., by replication with replacement of the less numerous cases and/or removal of some of the numerous cases to create the balanced training set so that there is about equal number of patterns for each class. We will study whether this balanced sampling is appropriate for empirical learning using neural networks.

This paper is organised as follows: In Section 2, we formulate the bootstrap expression of the prediction error to base our work on sound statistical theory. Then, in Section 3, numerical experiments are presented to assess empirically the impact of re-sampling on the network ability to learn. In Section 4, we identify analytically the optimal proportion for a network meeting simple and quite un-restricting conditions. We extend in Section 4.4 the re-

sults to networks behaving under weaker learning conditions. A lower and an upper bounds for the optimal re-sampling proportion for binary classification are derived in Section 4.5. In Section 5, an extension of the present method to mapping techniques for cross-validation is proposed. Conclusions are given in Section 6.

2 Bootstrap

The *bootstrap techniques* (see [4]) were introduced in 1979 as a computer-based method for estimating the standard error of empirical distributions. They allow for estimates of significant levels of arbitrary statistics when the form of the underlying distribution is unknown. 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.

There are related methods other than the bootstrap to estimate prediction errors. For examples, references for cross-validation are found in [19], [20], and [3]. References for the AIC (Akaike Information Criterion) can be found in [2], while references for the BIC (Bayesian Information Criterion), can be found in [16]. For more details, the reader is referred to [4] and [10].

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 $\mathbf{x}$, whereby 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:

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

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 $err(\mathbf{x}^*,\widehat{F})$ 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:

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

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

on bootstrap samples is given by:

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

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

3 Numerical Experiments

To assess the effects of re-sampling on the learning ability of the network, we study the symmetry detection problem for 6 bits code because:

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

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 are $2^6 = 64$ different vectors in the complete sample space and $2^3 = 8$ symmetric vectors. There are $64 - 8 = 56$ asymmetric vectors.

3.1.2 Outputs

If the input vector of the pattern is symmetric, the output is 1. It is 0 otherwise.

3.2 Network Architecture and Training Algorithms

We trained the feed-forward neural network with one hidden layer, which has a varying number of hidden units ($3 \sim 10$), by using back-propagation ([9]) and conjugate gradient algorithms ([14] and [15]). A stochastic noise based algorithm proposed by the second author was also experimented (see [13]; [11] and [12]). The result presented in this paper 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 (see [5]).

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 patterns. 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. The scheme takes the following steps:

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 re-sampling a set of 64 vectors, we take 128 ($= 64 \times 2$) 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 bootstrap re-sampling sets.
3. For each of these sets, the network learns the weight of neural connections. Each network must converge on the bootstrap training set.
4. For each network, we compute the 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 optimise the empirical learning of neural networks in general?

3.5 Results

The results are presented graphically in Figs. 1 and 2 for cases with 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 generalise 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, respectively.

The classification error attains a minimum - corresponding to a maximum in the generalisation 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

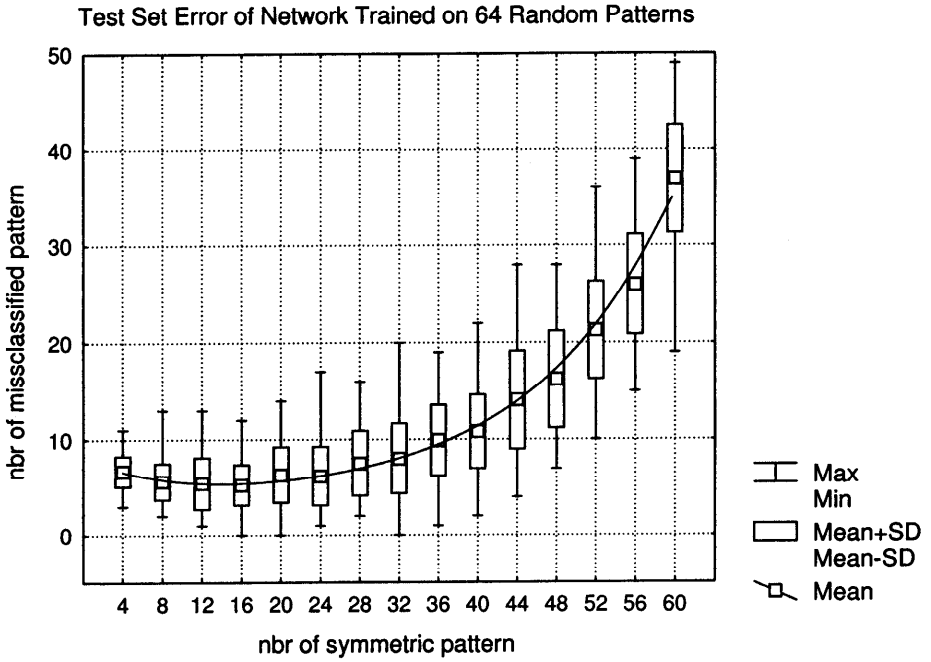

Figure 1: Miss-classification Error for 64 Patterns

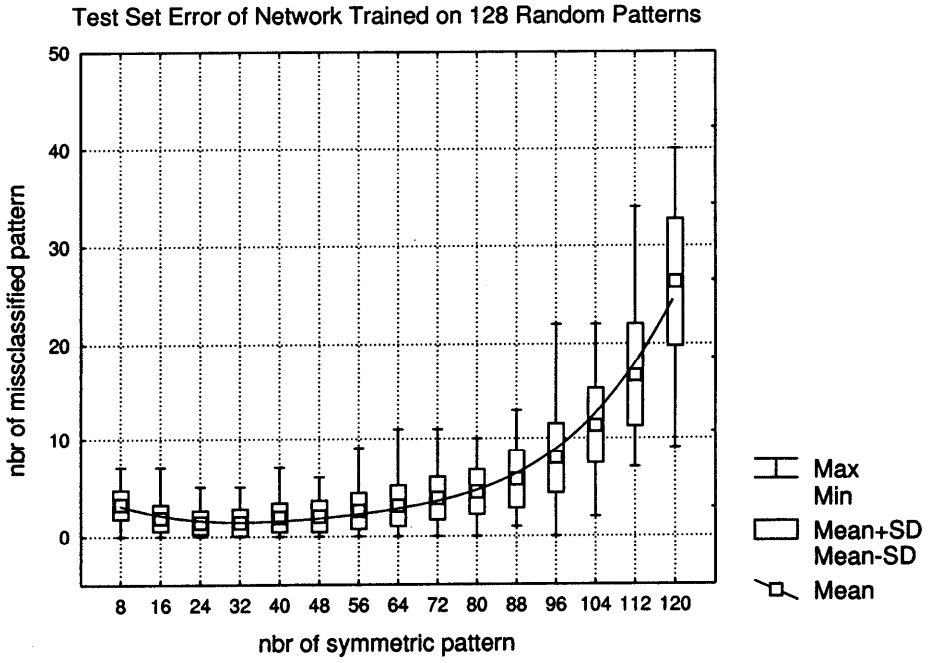

Figure 2: Miss-classification Error for 128 Patterns

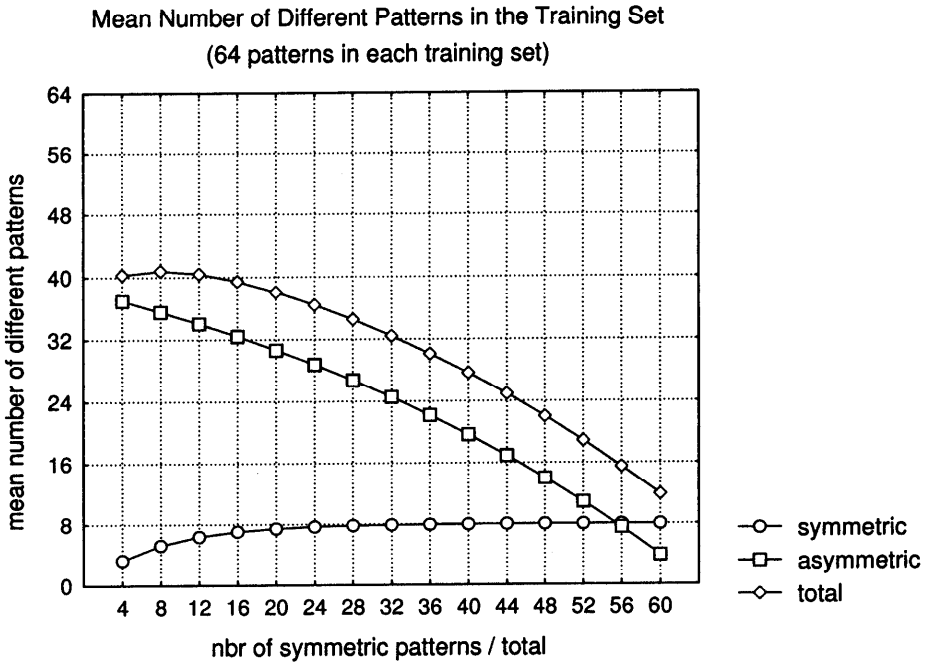

Figure 3: Mean Number of Different Patterns for 64 Patterns

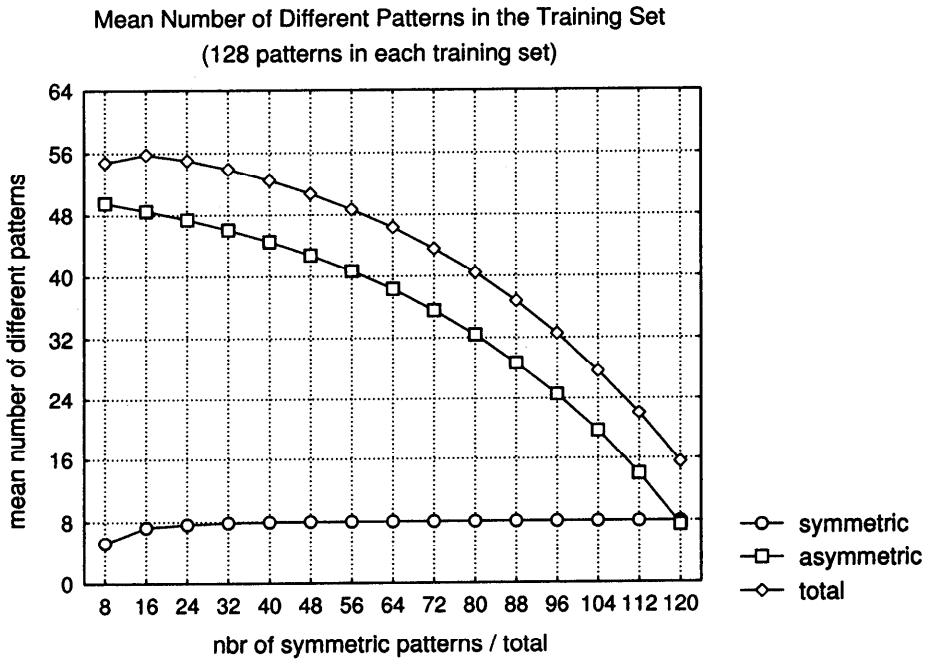

Figure 4: Mean Number of Different Patterns for 128 Patterns

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 are respected. We may conjecture that even though there are 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 Analytical Considerations

This section is organised as follows: First we propose a minimum requirement on the network ability to learn in the context of binary classification using Bayes decision rule. Then, we derive analytically the optimal re-sampling proportion. An extension to multiple classes is presented in Appendix.

4.1 Bayes Decision Rule

Bayes decision theory is a fundamental statistical approach to the problem of pattern classification. It shows that for every pattern classification problem, there is an optimal classifier that will, on a statistical basis, correctly determine the class of unknown patterns a higher percentage of the time than will any other classifier. This classifier is known as the *Bayes classifier*. This approach is based on the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known. Since we do not usually have this detailed level of knowledge, most classifiers are sub-optimal in their performance. Notice that, although a Bayes classifier has optimal performance, it may not be perfect. The performance of a Bayes classifier is determined by how much overlap exists between the classes (see [7] or [1]).

Given a new sample, represented by the vector $\mathbf{x}$ of its characteristics, the problem is to use the available information to classify it optimally following some criteria. In the numerical experiment of Section 3, $\mathbf{x}$ corresponds to the input vector.

The *state of nature* is defined as the class to which a new sample really belongs. Let $\Omega = \{\omega_1, \ldots, \omega_c\}$ be the finite set of c states of nature, and $D = \{\alpha_1, \ldots, \alpha_d\}$ be the finite set of d possible actions, i.e. the decision of classifying the new pattern to a given class. Let $\lambda(\alpha_i | \ \omega_j)$ be the loss incurred for taking action α_i when the state of nature is ω_j. Let the feature vector $\mathbf{x}$ be a d-dimensional vector-valued random variable, and let $p(\mathbf{x} | \ \omega_j)$ be the state conditional probability density function for $\mathbf{x}$ conditioned on

ω_j. Finally let $P(\omega_j)$ be the a priori probability that nature is in state ω_j. Then the a posteriori probability $P(\omega_j|\ \mathbf{x})$ can be computed from $p(\mathbf{x}|\ \omega_j)$ by Bayes rule:

$$P(\omega_j|\ \mathbf{x}) = \frac{p(\mathbf{x}|\ \omega_j)P(\omega_j)}{p(\mathbf{x})}, \tag{7}$$

where

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|\ \omega_j)P(\omega_j). \tag{8}$$

Suppose that we observe a particular $\mathbf{x}$ and that we contemplate taking action α_i. If the true state of nature is ω_j, a loss $\lambda(\alpha_i|\ \omega_j)$ will be incurred. Since $P(\omega_j|\ \mathbf{x})$ is the probability that the true state of nature is ω_j, the expected loss associated with taking action α_i is given by

$$R(\alpha_i|\ \mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\ \omega_j)P(\omega_j|\ \mathbf{x}). \tag{9}$$

In decision-theoretic terminology, an expected loss is called a *risk*, and $R(\alpha_i|\ \mathbf{x})$ is known as the *conditional risk*. Whenever we encounter a particular observation $\mathbf{x}$, we can minimise our expected loss by selecting the action that minimises the conditional risk. We shall now show that this is actually the optimal Bayes decision procedure.

Stated formally, our problem is to find a Bayes decision rule against $P(\omega_j)$ that minimises the overall risk. A *decision rule* is a function $\alpha(\mathbf{x})$ that tell us which action to take for every possible observation. To be more specific, for every $\mathbf{x}$ the *decision function* $\alpha(\mathbf{x})$ assumes one of the values $\alpha_1, \ldots, \alpha_d$. The overall risk R is the expected loss associated with a given decision rule. Since $R(\alpha_i|\ \mathbf{x})$ is the conditional risk associated with action α_i, and since the decision rule specifies the action, the overall risk is given by

$$R = \int R(\alpha(\mathbf{x})|\ \mathbf{x})p(\mathbf{x})d\mathbf{x}, \tag{10}$$

where $d\mathbf{x}$ denotes d-dimensional volume element, and the integral extends over the entire feature space. Clearly, if $\alpha(\mathbf{x})$ is chosen so that $R(\alpha(\mathbf{x})|\ \mathbf{x})$ is as small as possible for every $\mathbf{x}$, then the overall risk will be minimised. This justifies the following statement of the *Bayes decision rule*: To minimise the overall risk, compute the conditional risk expressed by Eq. (9) for $i = 1, \ldots, a$ and select the action α_i for which $R(\alpha_i|\ \mathbf{x})$ is minimum. (Note that if more than one action minimises $R(\alpha_i|\ \mathbf{x})$, it does not matter which of these actions is taken, and any convenient tie-breaking rule can be used.) The resulting minimum overall risk is called the Bayes risk and is the best performance that can be achieved.

Suppose that we generalise the model and replace the deterministic decision function $\alpha(\mathbf{x})$ with a *randomised rule*, i.e., the probability $P(\alpha_i|\ \mathbf{x})$ of taking action α_i upon observing $\mathbf{x}$. The resulting risk is given by

$$R = \int \left[\sum_{i=1}^{d} R(\alpha_i|\ \mathbf{x}) P(\alpha_i|\ \mathbf{x})\right] p(\mathbf{x}) d\mathbf{x}. \tag{11}$$

Clearly, R is minimised by choosing $P(\alpha_i|\ \mathbf{x}) = 1$ for the action α_i associated with the minimum conditional risk $R(\alpha_i(\mathbf{x})|\ \mathbf{x})$, thereby showing that the randomised classifier performs poorer than the deterministic one. Nevertheless, this classifier will be proven useful in the subsequent analysis.

4.2 Notations

To analyse the binary classification problem of Section 3 in the viewpoint of Bayesian decision theory, we define the following notation:

- Let N be the total number of patterns in the sample set.
- Let A be the total number of patterns in class $\mathcal{A}$, and S be the total number of patterns in class $\mathcal{S}$. We have: $N = A + S$.
- Let $q_a = \frac{A}{N}$ and $q_s = \frac{S}{N}$, the a priori probabilities. $q_a + q_s = 1$.
- Let m be the number of different patterns in the training set.
- Let a and s be the number of patterns of class $\mathcal{A}$ and $\mathcal{S}$ respectively, that are included in the training set. We have $m = a + s$.
- Let c_a be the cost of miss-classifying a pattern from class $\mathcal{A}$, and c_s the cost of miss-classifying a pattern from $\mathcal{S}$.

The set of possible actions D reduces to $D = \{\alpha_a, \alpha_s\}$, where α_a is the decision to classify a pattern to class $\mathcal{A}$ and α_s is the decision to classify it to $\mathcal{S}$. Eq. (11), then, can be rewritten as:

$$\begin{aligned} R = & \sum_{\mathbf{x}\in\mathcal{A}} \sum_{i=1}^{d} R(\alpha_i|\ \mathbf{x} \in \mathcal{A}) P(\alpha_i|\ \mathbf{x} \in \mathcal{A}) \\ & + \sum_{\mathbf{x}\in\mathcal{S}} \sum_{i=1}^{d} R(\alpha_i|\ \mathbf{x} \in \mathcal{S}) P(\alpha_i|\ \mathbf{x} \in \mathcal{S}) \end{aligned} \tag{12}$$

If the costs $R(\alpha_i|\ \mathbf{x})$ are fixed, the task of the network associated with this classifier will be to find $P(\alpha_i|\ \mathbf{x})$ that reduces the overall risk. We can make weaker assumptions on the performance of the network in order to estimate an upper bound to the overall risk.

4.3 Minimum Assumption on Network Learning

When the network has been presented with a patterns of class $\mathcal{A}$ and s patterns of $\mathcal{S}$, the network is assumed to behave as follows:

1. Classify correctly the $m = a + s$ patterns that are presented during the training phase.
2. Classify a pattern it has never seen with a probability a/m to class $\mathcal{A}$ (α_a) and with a probability s/m to class $\mathcal{S}$ (α_s).

In terms of the Bayes decision theory, this corresponds to:

$$\begin{aligned} R(\alpha_s|\ \mathbf{x} \in \mathcal{A}) &= c_a & \qquad R(\alpha_a|\ \mathbf{x} \in \mathcal{S}) &= c_s \\ R(\alpha_a|\ \mathbf{x} \in \mathcal{A}) &= 0 & \qquad R(\alpha_s|\ \mathbf{x} \in \mathcal{S}) &= 0 \\ P(\alpha_a|\ \mathbf{x}) &= a/s & \qquad P(\alpha_s|\ \mathbf{x}) &= m/s \end{aligned}$$

Note that the conditional probabilities $P(\alpha_a|\ \mathbf{x})$ and $P(\alpha_s|\ \mathbf{x})$ do not depend on $\mathbf{x}$. Because a real network extracts information from its input $\mathbf{x}$, it is expected to perform better than the classifier defined here. In this sense, the network showing performances equivalent to this classifier can be viewed as a lower limit on network learning ability.

4.4 Optimal Proportion

The minimum assumption enables us to compute a cost function of the network. We can assert as follows:

- The number of class $\mathcal{S}$ patterns unknown to the network is $(S - s)$,
- the probability of miss-classifying them is a/m, and
- the cost of miss-classifying one of these patterns is c_s.

The error associated with the other class is assessed in a similar manner. Consequently, Eq. (11) becomes:

$$Err = c_s(S - s)\frac{a}{m} + c_a(A - a)\frac{s}{m} \tag{13}$$

Setting $\alpha = a/m$, i.e., the proportion of vectors of class $\mathcal{A}$ in the training set, we have:

$$Err = c_s(S - m(1 - \alpha))\alpha + c_a(A - \alpha m)(1 - \alpha) \tag{14}$$

To determine the proportion of patterns that minimises the cost function, we differentiate Eq. (14) with respect to α:

$$\begin{aligned}\frac{\partial Err}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\{c_s(S - m(1-\alpha))\alpha + c_a(A - \alpha m)(1-\alpha)\} \\ &= \frac{\partial}{\partial \alpha}\{m\alpha^2(c_s + c_a) + \alpha(c_s S - c_a A - m(c_s + c_a)) + A\} \\ &= 2\alpha m(c_s + c_a) + c_s S - c_a A - m(c_s + c_a) \qquad (15)\end{aligned}$$

Therefore, we have:

$$\begin{aligned}\frac{\partial Err}{\partial \alpha} = 0 \quad &\Leftrightarrow \quad 2\alpha m(c_s + c_a) + c_s S - c_a A - m(c_s + c_a) = 0 \\ &\Leftrightarrow \quad \alpha = \frac{1}{2} + \frac{(c_a A - c_s S)}{2m(c_s + c_a)} \qquad (16)\end{aligned}$$

Note that α minimises Err because the second derivative is always positive:

$$\frac{\partial^2 Err}{\partial^2 \alpha} = 2m(c_s + c_a) > 0 \qquad (17)$$

The resulting optimal proportions are written as:

$$1 - \alpha^* = \frac{1}{2} + \frac{(c_s S - c_a A)}{2m(c_s + c_a)} \qquad (18)$$

$$\alpha^* = \frac{1}{2} + \frac{(c_a A - c_s S)}{2m(c_s + c_a)} \qquad (19)$$

This can be expressed in terms of a priori distributions q_a and q_s:

$$1 - \alpha^* = \frac{1}{2} + \frac{N}{2m}\frac{(q_s c_s - q_a c_a)}{c_s + c_a} \qquad (20)$$

$$\alpha^* = \frac{1}{2} + \frac{N}{2m}\frac{(q_a c_a - q_s c_s)}{c_s + c_a} \qquad (21)$$

Noting that $q_a + q_s = 1$, we can eliminate q_s from Eq. (21):

$$\alpha^* = \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s + c_a}\right) \qquad (22)$$

In the above analysis, the derivatives have been calculated without taking into account the natural conditions that α must meet. These are:

1. α is a proportion and, therefore, we have,

$$0 \leq \alpha \leq 1 \qquad (23)$$

2. There cannot be more patterns of a class in the training set than there are in the sample space, i.e.,

$$\begin{aligned} m\alpha &\leq Nq_a \\ m(1-\alpha) &\leq N(1-q_a) \end{aligned}$$

This can be rewritten like:

$$\frac{N}{m}q_a \geq \alpha \geq 1 - \frac{N}{m}(1-q_a) \tag{24}$$

Because the second derivative in Eq. (17) is always positive, there is no local minima and Err increases with the distance from the minimum point. Therefore, if Eqs. (19) and (21) give a solution outside of the feasible region defined by Eqs. (23) and (24), the minimum in that range will be the α which is closest from the solution of Eqs. (19) and (21).

Different cases may arise and are analyzed in the subsequent development.

4.4.1 Bounds on α

1. If $q_a < \frac{m}{N}$, conditions (23) and (24) reduce to

$$\alpha \leq \frac{N}{m}q_a < 1 \tag{25}$$

If the solution of Eq. (21) is greater than $\frac{N}{m}q_a$, then the feasible α is $\alpha^* = \frac{N}{m}q_a$. This happens when:

$$\begin{aligned} \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s + c_a}\right) &\geq \frac{N}{m}q_a \\ \Rightarrow\ q_a \leq \frac{m}{N} - \frac{c_s}{c_s + c_a} & \end{aligned} \tag{26}$$

2. If $q_a \geq \frac{m}{N}$, then condition (24) is irrelevant, i.e.,

$$\alpha \leq 1 \leq \frac{N}{m}q_a \tag{27}$$

and we have the condition $\alpha \leq 1$ to satisfy. If the solution of Eq. (21) is greater than 1, then the feasible α is $\alpha^* = 1$. This happens when:

$$\begin{aligned} \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s + c_a}\right) &> 1 \\ \Rightarrow\ q_a > \frac{c_s}{c_s + c_a} + \frac{m}{N} & \end{aligned} \tag{28}$$

3. If $q_a > 1 - \frac{m}{N}$, the conditions (23) and (24) become:

$$0 < 1 - \frac{N}{m}(1 - q_a) \leq \alpha \tag{29}$$

If this inequality is incompatible with the solution of Eq. (21), then the feasible α is $\alpha^* = 1 - \frac{N}{m}(1 - q_a)$. This happens when:

$$\begin{aligned} \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s + c_a}\right) &< 1 - \frac{N}{m}(1 - q_a) \\ \Rightarrow \; q_a &> 2 - \frac{c_s}{c_s + c_a} - \frac{m}{N} \end{aligned} \tag{30}$$

4. If $q_a \leq 1 - \frac{m}{N}$, the conditions (23) and (24) become:

$$1 + \frac{N}{m}(q_a - 1) \leq 0 < \alpha \tag{31}$$

If this inequality is incompatible with the solution of Eq. (21), then the feasible α is $\alpha^* = 0$. This happens when:

$$\begin{aligned} \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s + c_a}\right) &\leq 0 \\ \Rightarrow \; q_a &\leq \frac{c_s}{c_s + c_a} - \frac{m}{N} \end{aligned} \tag{32}$$

4.4.2 Summary

The conditions derived in the former section entail an exhaustive list of cases from which we determine α uniquely. In summary, we have the following upper bounds on q_a:

$$\text{if } 0 < q_a \leq \frac{c_s}{c_s + c_a} - \frac{m}{N} \quad \Rightarrow \quad \alpha^* = 0 \tag{33}$$

$$\text{if } 0 < q_a \leq \frac{m}{N} - \frac{c_s}{c_s + c_a} \quad \Rightarrow \quad \alpha^* = \frac{N}{m} q_a \tag{34}$$

It is important to note that above two inequality conditions cannot hold simultaneously.

There are also lower bounds:

$$\text{if } \frac{c_s}{c_s + c_a} + \frac{m}{N} < q_a < 1 \Rightarrow \alpha^* = 1 \tag{35}$$

$$\text{if } 2 - \frac{c_s}{c_s + c_a} - \frac{m}{N} < q_a < 1 \Rightarrow \alpha^* = 1 - \frac{N}{m}(1 - q_a) \tag{36}$$

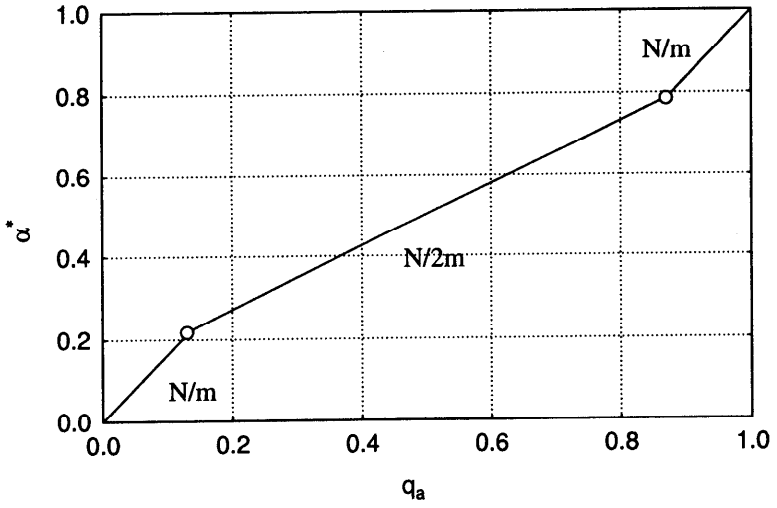

Figure 5: Optimal α as a Function of q_a

These two inequality conditions as well are mutually exclusive. In the rest of the cases, we have:

$$\text{if } q_a > \left|\frac{c_s}{c_s+c_a} - \frac{m}{N}\right| \quad \text{or} \quad (1-q_a) > \left|1 - \frac{c_s}{c_s+c_a} - \frac{m}{N}\right|$$
$$\Rightarrow \quad \alpha^* = \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s+c_a}\right) \qquad (37)$$

As a practical rule, first compute from Eq. (22) the optimal number of patterns in each class. If there is not enough patterns in one of the classes, then include in the training set all the patterns of this class, and complete the set with the other class to obtain a total of m patterns.

Distribution of samples between the two classes need not be the same in the available training set and the prospective set on which the network will be actually used. In term of the practical rule, this can be restated as follows. First compute the optimal number of patterns in each class, using q_a and q_s being the a priori frequencies in the prospective set. If it appears that one of the classes do not contain enough samples to respect the optimal proportion, then complete the training set with patterns of the other class.

A typical solution is presented in Fig. 5. Note that the slopes are indicated along the line segments they correspond to.

4.5 Bounds for the Optimal Proportion

Recall that the conditional probabilities $P(\alpha_a|\ \mathbf{x})$ and $P(\alpha_s|\ \mathbf{x})$ assumed in Section 4.3 are independent of $\mathbf{x}$. The classifier corresponding to this as-

sumption is expected to perform poorly because it only takes into account the distribution of outputs that it observed during the training phase, and ignores the input vector $\mathbf{x}$. A real network will actually classify better the patterns than the minimum assumption network given in the previous section. It will:

1. Classify correctly the $m = a + s$ patterns that were presented during the training phase.

2. Classify incorrectly a pattern it has never seen with a probability $P(\alpha_a | \mathbf{x} \in \mathcal{S}) = \alpha' = \alpha - \gamma_a = a/m - \gamma_a$ to class $\mathcal{A}$ and with a probability $P(\alpha_s | \mathbf{x} \in \mathcal{A}) = \beta' = \beta - \gamma_s = 1 - \alpha - \gamma_s = s/m - \gamma_s$ to class $\mathcal{S}$.

Clearly, γ_a and γ_s must be positive and functions of $\mathbf{x}$. Moreover, α' and β' being positive, it entails that $\gamma_a \leq \alpha$ and $\gamma_s \leq 1 - \alpha$

We can rewrite the expression of the cost function in Eq. (14) as

$$\begin{aligned} Err &= c_s(S - m(1-\alpha))\alpha' + c_a(A - \alpha m)\beta' \\ &= c_s(S - m(1-\alpha))(\alpha - \gamma_a) + c_a(A - \alpha m)(1 - \alpha - \gamma_s) \end{aligned} \quad (38)$$

The optimal proportion is the value of α that minimises this error:

$$\begin{aligned} \frac{\partial Err}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\{c_s(S - m(1-\alpha))(\alpha - \gamma_a) + c_a(A - \alpha m)(1 - \alpha - \gamma_s)\} \\ &= 2\alpha m(c_a + c_s) + (c_s S - c_a A) - m(c_a + c_s) - m(c_s\gamma_a - c_a\gamma_s) \end{aligned} \quad (39)$$

Equating this derivative to zero, we have:

$$\alpha^* = \frac{1}{2} + \frac{(c_a A - c_s S)}{2m(c_a + c_s)} + \frac{c_s\gamma_a - c_a\gamma_s}{2(c_a + c_s)} \quad (40)$$

or, in terms of the a priori distributions:

$$\alpha^* = \frac{1}{2} + \frac{N}{2m}\frac{(c_a q_a - c_s q_s)}{(c_a + c_s)} + \frac{c_s\gamma_a - c_a\gamma_s}{2(c_a + c_s)} \quad (41)$$

This corresponds to Eq. (21) considering the relative ability of the network to learn the patterns across binary classes. Because $0 \leq \gamma_a \leq \alpha$ and $0 \leq \gamma_s \leq 1 - \alpha$, we obtain a lower bound for α^* when $\gamma_a = 0$ and $\gamma_s = 1 - \alpha$:

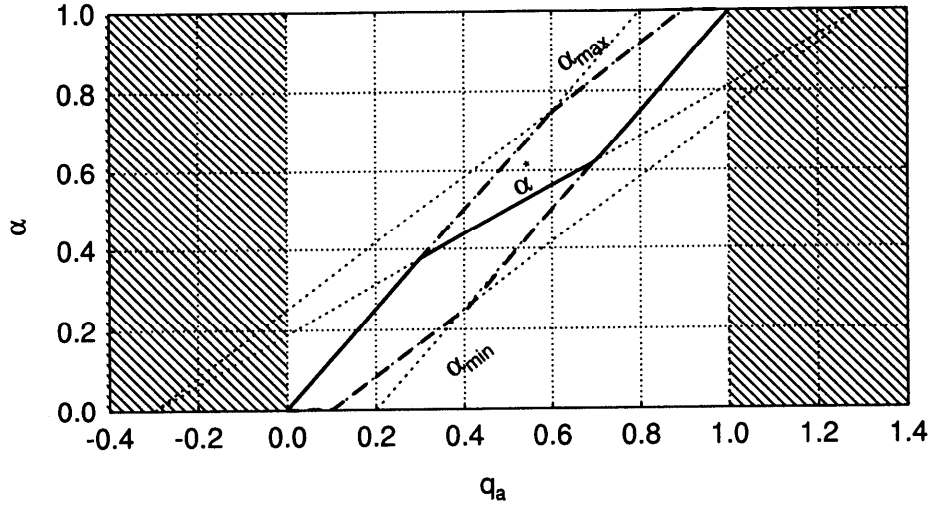

Figure 6: Optimal, Minimum and Maximum α as a Function of q_a

$$\begin{aligned} \alpha^*_{\min} &= \frac{1}{2} + \frac{N}{2m}\frac{c_a q_a - c_s q_s}{c_a + c_s} - \frac{c_a(1-\alpha^*_{\min})}{2(c_a+c_s)} \\ \Rightarrow \alpha^*_{\min} &= \frac{c_s}{c_a+2c_s} + \frac{N}{m}\frac{c_a q_a - c_s q_s}{c_a+2c_s} \end{aligned} \tag{42}$$

Similarly, the upper bound of α^* is attained when $\gamma_a = \alpha$ and $\gamma_s = 0$:

$$\begin{aligned} \alpha^*_{\max} &= \frac{1}{2} + \frac{N}{2m}\frac{c_a q_a - c_s q_s}{c_a + c_s} - \frac{c_s \alpha^*_{\max}}{2(c_a+c_s)} \\ \Rightarrow \alpha^*_{\max} &= \frac{c_a+c_s}{2c_a+c_s} + \frac{N}{m}\frac{c_a q_a - c_s q_s}{2c_a+c_s} \end{aligned} \tag{43}$$

The same conditions on α in Eqs. (23) and (24) apply to $\alpha_{\min}$ and $\alpha_{\max}$. This means that $\alpha_{\min}$ will be the maximum of the values defined by Eqs. (23), (24), and (42). Similarly, $\alpha_{\max}$ will be the minimum of 1, $\frac{N}{m}q_a$ and the α defined in Eq. (43). A typical situation has been represented in Fig. 6. The bold line represents α^*, the bold dotted lines correspond to $\alpha^*_{\max}$ and $\alpha^*_{\min}$.

4.6 Equal Costs

When the costs are equal, the optimal proportions for the minimum assumption network defined in Eqs. (20) and (21) simplify to:

$$1-\alpha^* = \frac{1}{2} + \frac{N}{4m}(q_s - q_a) = \frac{1}{2} + \frac{N}{4m}(2q_s - 1) \tag{44}$$

$$\alpha^* = \frac{1}{2} + \frac{N}{4m}(q_a - q_s) = \frac{1}{2} + \frac{N}{4m}(2q_a - 1) \tag{45}$$

If $q_a > 50\%$, then $\alpha^* > 50\%$. When the size of the training set m increases, the optimal proportion re-centres a-priori distribution according to Eqs. (44) and (45).

Following Eq. (41), the re-sampling proportion of a network with a higher learning capability will be optimal when:

$$\alpha^* = \frac{1}{2} + \frac{N}{4m}(q_a - q_s) + \frac{\gamma_a - \gamma_s}{4} \tag{46}$$

Comparing this result with Eq. (45), we see that the result obtained for the minimum requirement network is in fact valid for any network as long as the ability to learn both patterns is the same ($\gamma_a = \gamma_s$).

The lower and upper bounds for the optimal proportion are:

$$\alpha^*_{\min} = \frac{1}{3} + \frac{N}{3m}(q_a - q_s) \tag{47}$$

$$\alpha^*_{\max} = \frac{2}{3} + \frac{N}{3m}(q_a - q_s) \tag{48}$$

If both $\alpha_{\min}$ and $\alpha_{\max}$ defined in Eqs. (47) and (48) lie in the feasible region, the length of the interval is $\alpha^*_{\max} - \alpha^*_{\min} = \frac{1}{3}$. From the conditions expressed in Eqs. (23) and (24), we may conclude that the optimal proportion lies in an interval of length inferior or equal to $\frac{1}{3}$.

4.7 Numerical Experiment

In the numerical experiments of Subsection 3.4, there are 64 patterns, out of which 8 are symmetric. Taking costs to be equal, Eq. (19) becomes:

$$\alpha^* = \frac{1}{2} + \frac{(56 - 8)}{2m(1 + 1)} = \frac{1}{2} + \frac{12}{m} \tag{49}$$

As $\alpha \leq 1$, i.e., for $0 \leq m \leq 24$, the error is minimised for $\alpha = 1$. For $m > 24$, the proportion of asymmetric vectors decreases. When $m = 40$, the optimum proportion is 80% for asymmetric patterns and 20% for symmetric patterns. This corresponds in Fig. 1 to a number of symmetric patterns equal to roughly 13. The a priori proportion of 12.5% for symmetric patterns corresponds to 8 symmetric patterns in Fig. 1.

Following Eqs. (47) and (48), the lower and upper bounds are: $1 \geq \alpha \geq .73$ and $0 \leq 1 - \alpha \geq .27$. On Fig. 1, this corresponds to a number of symmetric patterns between 0 and 18.

4.8 Re-sampling for Binary Classification

In general, the proportion α^* is different from the proportion A/N in class 1. There is therefore an advantage in over-representing one of the patterns' set when all patterns cannot be used for training. There are different situations in which this can happen, for example,

- The distribution of patterns available for training is different from the distribution that will be used during the application.
- The training set is huge compared to the number of weights in the network. It may be a waste of computing power to train the network on all sample sets.
- When a part of the sample set is unknown, but we know the a priori distribution of patterns.
- When many patterns are available for one class and few for the other.

A simple and practical estimate for the optimal proportion is given by Eq. (21). If the solution is not in the feasible set, the closest feasible value must be selected. Unfortunately, the ratio m/N might be difficult to estimate if patterns are continuous. On the other hand, when cross-validation techniques are used (see [19], [20], and [3]), the total number of patterns N and the number m of patterns in the training set may be estimated. When the proportion in application data is known before-hand, it is useful to take advantage of the information during training.

An unbalanced training set is particulary critical when the number of patterns in one class is significantly fewer compared with the other, but the cost associated with miss-classification is high. This happens in many practical examples of disease diagnosis, car insurances, default predictions, credit assignments, etc. In these situations, q_a is small ($0 < q_a \ll q_s < 1$) and $c_a \gg c_s$.

Given these hypothesis, it is probable that in most of the cases the limit condition as expressed in Eq. (34) applies. To see it, we rewrite this equation:

$$0 < q_a \leq \frac{m}{N} - \frac{c_s}{c_s + c_a} = \frac{m}{N} - \frac{1}{1 + c_a/c_s} \tag{50}$$

As $c_a \gg c_s$, the second term of the right hand side is small, and unless m is very small compared to N, the condition expressed in Eq. (50) is met because $0 < q_a \ll q_s < 1$. Consequently, the expression of the optimal α becomes:

$$\alpha^* = \frac{N}{m} q_a$$

As we always have that $N \geq m$, the last expression entails that the optimal proportion will always correspond to an over representation of the less populated class.

If Eq. (50) doesn't hold, the optimal α is given by Eq. (22). This last can be rewritten as

$$\alpha^* = \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_s}{c_s + c_a}\right) = \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{1}{1 + c_a/c_s}\right) \tag{51}$$

In the limit case (considering N fixed), we have,

$$\frac{1}{2} + \lim_{\substack{q_a/q_s \to 0 \\ c_s/c_a \to 0}} \frac{N}{2m}\left(q_a - \frac{1}{1 + c_a/c_s}\right) = \frac{1}{2} \tag{52}$$

This may justify the re-sampling to 50%-50% proportion in most of the actual data mining practice. To examine further this justifications, we examine the lower and upper bounds of the optimal proportion, Eqs. (42) and (43) can be rewritten:

$$\alpha^*_{\min} = \frac{c_s}{c_a + 2c_s} + \frac{N}{m}\frac{c_a q_a - c_s q_s}{c_a + 2c_s} \tag{53}$$

$$= \frac{c_s/c_a}{1 + 2c_s/c_a} + \frac{N}{m}\frac{q_a/q_s - c_s/c_a}{1 + 2c_s/c_a} \tag{54}$$

and

$$\alpha^*_{\max} = \frac{c_a + c_s}{2c_a + c_s} + \frac{N}{m}\frac{c_a q_a - c_s q_s}{2c_a + c_s} \tag{55}$$

$$= \frac{1 + c_s/c_a}{2 + c_s/c_a} + \frac{N}{m}\frac{q_a/q_s - c_s/c_a}{2 + c_s/c_a} \tag{56}$$

Taking the limits for q_a/q_s and c_s/c_a tending to 0, we have:

$$\lim_{\substack{q_a/q_s \to 0 \\ c_s/c_a \to 0}} \alpha^*_{\min} = 0 \quad \text{and} \quad \lim_{\substack{q_a/q_s \to 0 \\ c_s/c_a \to 0}} \alpha^*_{\max} = \frac{1}{2} \tag{57}$$

We see that α^* lies between 0 and 1/2. We conclude that re-sampling to 50%-50% might be optimal, but is somehow an extreme case.

5 Cross-Validation

The essence of network learning is to encode an input-output mapping (represented by a set of labelled examples) into the synaptic weights and thresholds of a multilayer perceptron. The hope is that the network becomes well trained so that it learns enough about the past to generalise to the future. From such a perspective the learning process amounts to a choice of network parameterisation for this data set. More specifically, we may view the network selection problem as choosing, within a set of candidate model structures (parameterisations), the "best" one according to a certain criterion.

In this context, a standard tool in statistics known as cross-validation provides an appealing guiding principle (see [19], [20], [10], and [6]). First the available data set is randomly partitioned into a training set and a test set. The training set is further partitioned into two disjoint subsets:

- Estimation subset used to select the model.
- Validation subset, used to test or validate the model.

The motivation here is to validate the model on a data set different from the one used for parameter estimation. In this way we may use the training set to assess the performance of various candidate models, and thereby choose the "best" one. There is, however, a distinct possibility that the model with the best-performing parameter values so selected may end up overfitting the validation subset. To guard against this possibility, the generalisation performance of the selected model is measured on the test set, which is different from the validation subset.

The use of cross-validation is appealing particularly when we have to design large neural network with good generalisation capability. For example, we may use cross-validation to determine the multilayer perceptron with optimal number of hidden neurons, and when it is best to stop training.

The approach to cross-validation described is referred to as the *hold out method*. There are other variants of cross-validation that find their own uses in practice, particularly when there is a scarcity of labelled examples. In such a situation we may use *multifold cross-validation* by dividing the available set of N examples into K subsets, $K > 1$; assumes that K is divisible into N. The model is trained on all the subsets except one, and the validation error is measured by testing it on the subset left out. This procedure is repeated for a total of K trials, each time using a different subset for validation. The performance of the model is assessed by averaging the squared error under validation over all the trials of the experiment. There is a disadvantage to multifold cross-validation: it may require an excessive

amount of computation since the model has to be trained K times, where $1 < K \leq N$.

5.1 Function Mapping

Let us examine the standard case of training a network to map a continuous function and test it using the cross-validation technique. We consider that the available data set represent fairly enough the distribution of the sample in the prospective set. If this is not verified, the adaptation to the practical rule in Subsection 4.4.2 can be used.

Although the function takes continuous values, the data set contains typically a finite number of samples. We can artificially recreate the conditions for a binary classification by setting a threshold ϵ and dividing the data set at the sample whose output value is over a given threshold.

The optimal α for a given ϵ is obtained from Eq. (22). It implies that the training set must contain a^* samples of class $\mathcal{A}$, with a^* defined by

$$a^*(\epsilon) = m\alpha^*(\epsilon) = \frac{m}{2} + \frac{N}{4}(2q_a(\epsilon) - 1) \tag{58}$$

The similar division of the training set being possible for a value $\epsilon + h$ of the threshold, we thus have simultaneously that

$$a^*(\epsilon + h) = m\alpha^*(\epsilon + h) = \frac{m}{2} + \frac{N}{4}(2q_a(\epsilon + h) - 1) \tag{59}$$

Consequently, the number of sample with an output value between ϵ and $\epsilon + h$ that should be contained in the training set is

$$m(\alpha^*(\epsilon + h) - \alpha^*(\epsilon)) = \frac{N}{2}(q_a(\epsilon + h) - q_a(\epsilon)). \tag{60}$$

This relation provides a simple rule to create a training set that will lead to a network with an optimal generalisation ability on the data set. In particular, if we have $m = N/2$, the distribution in the training set and the data set becomes identical. This technique can be extended to cross-validation statistics of generic regression models.

6 Conclusions

We presented the bootstrap analysis approach to neural computations. Then, we set up numerical experiments to assess empirically the impact of resampling on the network ability to learn. The importance of the sample mixture in bootstrap training is investigated analytically.

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 original distribution. The numerical and theoretical results of this paper indicate that the learning ability of the network is indeed enhanced by re-sampling, but the proportion should be carefully assessed. In particular, the 50%-50% re-sampling scheme seems to be justified when one of the class contains fewer patterns but the associated cost of miss-classifying them is very expensive. A simple method to estimate the optimal proportion for binary classification problems has been proposed. The results have been presented in the context of neural computations, but the present methods apply to most of supervised learning systems, including decision trees.

Acknowledgements

The work of the second author is supported by a Grant-in-Aid of the Ministry of Education, Science, Sports, and Culture of Japan.

References

[1] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* New York : John Wiley, 1973.

[2] H. Akaike. Information theory and an extension of the maximum likelyhood principle. *Second International Symposium on Information Theory*, pages 267–281, 1973.

[3] D. M. Allen. The relationship between variable selection and data augmentation and a method of prediction. *Technometrics 16*, pages 125–7, 1974.

[4] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap.* New York ; Tokyo : Chapman & Hall, 1993.

[5] G. Dupret and M. Koda. Bootstrapping for neural network learning. *APORS' 2000 - Asia Pacific Operation Research Society (in press).*

[6] S. Haykin. *Neural networks : a comprehensive foundation.* New York : Macmillan College Publishing, 1994.

[7] R. Hecht-Nielsen. *Neurocomputing.* Reading, Mass. ; Tokyo : Addison-Wesley Pub. Co., c1990, 1990.

[8] IBM. Intelligent miner for relationship marketing. 1999.

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

[10] J. P. C. Kleijnen. Bootstrapping and cross-validation of metamodels in simulation. *Proceedings SAMO'98, European Commission and University of Venice*, pages 155–157, 1998.

[11] M. Koda. Stochastic sensitivity analysis method for neural network learning. *International Journal of Systems Science, Vol. 26, No. 3*, pages 703–711, 1995.

[12] M. 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.

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

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

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

[16] G. Schwartz. Estimating the dimension of a model. *Annals of Mathematical Statistics. 6*, pages 461–464, 1978.

[17] SPSS. Clementine, interactive data mining tool. 1999.

[18] StatSoft, Inc. *Electronic Statistics Textbook.* Tulsa, OK: StatSoft. WEB: http://www.statsoft.com/textbook/stathome.html, 1999.

[19] M. Stone. Cross-validation choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Ser.B : Methodological B 36*, pages 111–147, 1974.

[20] M. Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society. Ser.B : Methodological B 39*, pages 44–7, 1977.

Appendix

A Optimal Proportions for Multiple Classes

In this appendix, we outline the argument of Section 4 to many classes.

A.1 Notations

We assume that the sample space is composed of J classes, each of which is denoted I_j where $j = 1, ..., J$. We have the following definitions:

- q_j, $j = 1, ..., J$, denotes the a priori distribution of the classes in the sample space.
- p_j, $j = 1, ..., J$, denotes the distribution in the training set.
- $E(i, j)$ is the cost of the error associated with miss-classifying a pattern from class i to class j.
- N is the total number of patterns in the sample space.
- m is the number of patterns in the training set.

From these definitions, we can easily deduce:

- Nq_j: the number of patterns of class I_j in the sample set.
- mp_j: the number of patterns of class I_j in the training set.
- $Nq_j - mp_j$: the number of patterns of I_j not in the training set.

A.2 Optimal Proportions

Making the minimum assumption on the learning ability of the network as given in Section 4.4, the mean error of miss-classifying a pattern of I_i into I_j is:

$$Err_{ij} = (Nq_i - mp_i)p_j E(i, j) \tag{61}$$

The total error associated with the class I_i is given by:

$$Err_i = \sum_{j=1}^{J} (Nq_i - mp_i)p_j E(i, j) \tag{62}$$

where $E(i,j)$ is a measure of the error. The total average error is:

$$Err = \sum_{i=1}^{J} Err_i = \sum_{i=1}^{J}\sum_{j=1}^{J}(Nq_i - mp_i)p_j E(i,j) \tag{63}$$

We have the following constrains on the p_i:

1. The p_i are proportions. Therefore, we have:

$$\sum_{j=1}^{J} p_j = 1 \tag{64}$$

2. Proportions are always positive:

$$p_j \geq 0 \tag{65}$$

3. In the training set, there cannot be more patterns of a class than actually exist in the sample space:

$$mp_j \leq Nq_j \tag{66}$$

We define the stack variables y_j to transform the inequality $p_j \geq 0$ into a strict equality: $p_j - y_j^2 = 0$. Similarly, the stack variable z_j transforms the inequality $mp_j \leq Nq_j$ into the equality relation: $mp_j - Nq_j + z_j^2 = 0$.

The Lagrangian becomes for this problem:

$$\begin{aligned} L &= \sum_{i=1}^{J}\sum_{j=1}^{J} E(i,j)(Nq_i - mp_i)p_j \\ &\quad -\lambda(\sum_{j=1}^{J} p_j - 1) - \sum_{j=1}^{J}\mu_j(p_j - y_j^2) - \sum_{j=1}^{J}\nu_j(mp_j - Nq_j + z_j^2) \end{aligned} \tag{67}$$

To find the optimal proportions that minimise the upper bound for the error, we minimise Err with respect to the p_i and the Langrange multipliers λ, μ_j and ν_j.

A.2.1 Derivatives of the Lagrangian

1. Derivatives with respect to the proportions:

$$\frac{\partial L}{\partial p_k} = \frac{\partial}{\partial p_k}\{\sum_{i=1}^{J}\sum_{j=1}^{J} E(i,j)(Nq_i - mp_i)p_j$$

$$
\begin{aligned}
& -\lambda(\sum_{j=1}^{J} p_j - 1) - \sum_{j=1}^{J} \mu_j(p_j - y_j^2) - \sum_{j=1}^{J} \nu_j(mp_j - Nq_j + z_j^2)\} \\
= & -m\sum_{j=1}^{J} E(k,j)p_j + \sum_{i=1}^{J} E(i,k)(Nq_i - mp_i) - \lambda - \mu_k - m\nu_k \\
= & \sum_{i=1}^{J}\{E(i,k)Nq_i - mp_i(E(k,i) + E(i,k))\} - \lambda - \mu_k - m\nu_k
\end{aligned}
\tag{68}
$$

2. Derivatives with respect to the stack variables y_k:

$$
\frac{\partial L}{\partial y_k} = \frac{\partial}{\partial y_k}\{-\sum_{j=1}^{J} \mu_j(p_j - y_j^2)\} = 2\mu_k y_k \tag{69}
$$

3. Derivatives with respect to the stack variables z_k:

$$
\frac{\partial L}{\partial z_k} = \frac{\partial}{\partial z_k}\{-\sum_{j=1}^{J} \nu_j(mp_j - Nq_j + z_j^2)\} = -2\nu_k z_k \tag{70}
$$

A.2.2 Second Order Derivatives

We first relax the second and third conditions expressed in Eqs. (65) and (66), and postpone their treatment. Eq. (68) becomes:

$$
\frac{\partial L}{\partial p_k} \sum_{i=1}^{J}\{E(i,k)Nq_i - mp_i(E(k,i) + E(i,k))\} - \lambda \tag{71}
$$

We use the case $k = 1$ to eliminate λ:

$$
\lambda = \sum_{i=1}^{J}\{E(i,1)Nq_i - mp_i(E(1,i) + E(i,1))\} \tag{72}
$$

After substitution of Eq. (72) in Eq. (71), we obtain:

$$
\begin{aligned}
\frac{\partial L}{\partial p_k} = & \sum_{i=1}^{J}\{E(i,k)Nq_i - mp_i(E(k,i) + E(i,k))\} \\
& - \sum_{i=1}^{J}\{E(i,1)Nq_i - mp_i(E(1,i) + E(i,1))\}
\end{aligned}
\tag{73}
$$

$$= \sum_{i=1}^{J}\{(E(i,k) - E(i,1))Nq_i$$
$$-(E(k,i) + E(i,k) - E(1,i) - E(i,1))mp_i\} \quad (74)$$

The optimal proportions are therefore defined by the system of J equations:

$$\sum_{i=1}^{J}\{(E(i,k)-E(i,1))Nq_i-(E(k,i)+E(i,k)-E(1,i)-E(i,1))mp_i\} = 0 \quad (75)$$

Examining the second derivative with respect to p_k:

$$\frac{\partial^2 L}{\partial^2 p_k} = \frac{\partial}{\partial p_k}\sum_{i=1}^{J} -(E(k,i) + E(i,k) - E(1,i) - E(i,1))mp_i \quad (76)$$
$$= mp_k(E(1,k) + E(k,1)) > 0 \quad (77)$$

There is no local minimum and the solutions of Eq. (75) define the unique minimum in the sub-space where $\sum_{j=1}^{J} p_j = 1$. Similar to Section 4.4, the feasible optimal solution will correspond to the proportions closest from the solutions of Eq. (75) which lies in the feasible region. The rest of the arguments follows along the lines that are described in Section 4.