ELSEVIER

European Journal of Operational Research 134 (2001) 141–156

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

Bootstrap re-sampling for unbalanced data in supervised learning

Georges Dupret, Masato Koda *

Institute of Policy and Planning Sciences, University of Tsukuba, 1-1-1 Tennoudai, Tsukuba Science City, 305-8573 Japan

Received 3 April 2000; accepted 6 July 2000

Abstract

This paper presents a technical framework to assess the impact of re-sampling on the ability of a supervised learning to correctly learn a classification problem. We use the bootstrap expression of the prediction error to identify the optimal re-sampling proportions in binary classification experiments using artificial neural networks. Based on Bayes decision rule and the a priori distribution of the objective data, an estimate for the optimal re-sampling proportion is derived as well as upper and lower bounds for the exact optimal proportion. The analytical considerations to extend the present method to cross-validation and multiple classes are also illustrated. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Neural networks; Decision support systems; Simulation; Data mining

1. Introduction

In supervised learning, the proportion of types of data in the training data set has a critical importance (for e.g., [9,17,18]). A neural network or a 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 [9,17,18]). These techniques are usually referred to as *enriched sampling* or *balanced sampling*; 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

* Corresponding author. Tel.: +81-298-53-5222; fax: +81-298-55-3849.

E-mail address: koda@shako.sk.tsukuba.ac.jp (M. Koda).

PII: S0377-2217(00)00244-7

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 addresses 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 technique [5] and on the Bayesian optimal classifier methodology. We show that it is possible to derive from the distribution of the objective values in the prospective data set, upper and lower bounds on the re-sampling proportion that would lead to the best generalisation ability. The study is based on an example using a neural network, but is applicable to all methods of supervised learning.

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 an artificial neural network's ability to learn. In Section 4, we relate our study to the Bayesian formalism. We identify analytically the optimal proportion for a network that meets simple and quite un-restrictive conditions in Section 5. In Section 6, lower and upper bounds for the optimal re-sampling proportion for binary classification are derived. Sections 7 and 8 present extensions of the present method to function mapping and to multiple classes, respectively. The application to cross-validation is also illustrated in Section 7. Conclusions are given in Section 9.

2. Bootstrap

The *bootstrap techniques* (see [5]) were introduced in 1979 as a computer-based method for estimating the standard error of empirical distributions. They allow estimating 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 to the bootstrap to estimate prediction errors. For example, references for cross-validation are found in [2,19,20]. The jackknife method is also related to bootstrapping (see [5]). References for the AIC (Akaike information criterion) can be found in [1], while references for the BIC (Bayesian information criterion), can be found in [16]. For more details, the reader is referred to [5,10].

2.1. Definitions

- Let $\mathbf{x}_i = (\mathbf{I}_i, O_i^d)$, $i = 1, ..., n$, be the ith 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 $\theta = E_F(O^{\mathrm{d}}) = \frac{1}{N}\sum_{i=1}^{N} O_i^{\mathrm{d}}$. In the same manner, the plug-in estimate of the mean is

$$\widehat{\theta} = E_{\widehat{F}}(O^{\mathrm{d}}) = \frac{1}{N}\sum_{i=1}^{N}(O_i^{\mathrm{d}})^*.$$

In the above example, $E_F(O^{\mathrm{d}}) = \frac{1}{5}\sum_{i=1}^{5} O_i^{\mathrm{d}}$, and its plug-in estimate is: $E_{\widehat{F}}(O^{\mathrm{d}}) = \frac{1}{5}\sum_{i=1}^{5}(O_i^{\mathrm{d}})^* = \frac{1}{5}(2O_3^{\mathrm{d}} + O_1^{\mathrm{d}} + O_4^{\mathrm{d}} + O_2^{\mathrm{d}})$.

- Suppose, we train the artificial neural 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 $\mathbf{I}_0$.
- $Q[O^{\mathrm{d}}, O]$ 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 $Q[O^{\mathrm{d}}, O] = 0$ if $O^{\mathrm{d}} = O$ and 1 otherwise.

2.2. Prediction error

Let $(\mathbf{I}_0, O_0^{\mathrm{d}})$ 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

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

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

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

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

where $f_{\mathbf{x}^*}(\mathbf{I}_i)$ 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 $\mathrm{err}(\mathbf{x}^*, \widehat{F})$ 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

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

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

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

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 a given network, we need to set up an experiment where the complete sample space is finite, known, and small so that the training is fast and easy. Moreover, the distribution function must be unbalanced so that the effects of re-sampling can be assessed. The symmetry detection problem for a six-bit code (defined in Section 3.1) meet these requirements.

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 values in $\{0, 1\}$ only:

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

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

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

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; hence there are $64 - 8 = 56$ asymmetric vectors.

3.1.2. Outputs

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

The symmetry detection is a rather difficult problem since the nearest neighbours with Hamming distance 1 to a symmetric vector are all asymmetric. Most of the supervised learning techniques are not effective to recognise patterns by using six-dimensional hyper-planes.

3.2. Network architecture and training algorithms

We take a feed-forward neural network with one hidden layer, which has a varying number of hidden units (3–10). To train this neural network, we use back-propagation [8] and conjugate gradient algorithms (see [14,15]). A stochastic-noise based algorithm proposed by the second author was also experimented (see [11–13]). The results did not differ significantly and were quite independent from the architecture and the learning algorithms [4]. We present in this paper the results for a network with six hidden neurons trained by conjugate gradient algorithm.

3.3. Re-sampling scheme

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

1. Decide on a proportion (for example, 40/64).
2. Take randomly with replacement 64 vectors such that the proportion decided in step 1 is respected. For example, if the proportion is 40/64, pick randomly 40 vectors from the set of symmetric vectors, and 24 ($= 64 - 40$) among the asymmetric vectors.

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. The experiment runs as follows:

1. the numerator of the proportion ranges from 4 to 60 by step of 4;
2. for each proportion, we construct $B = 100$ bootstrap re-sampling sets;
3. for each of these sets, the network learns the weight of the neural connections. Each network should 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):

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

with $B = 100$ and $n = 100$.

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 (and supervised learning algorithms) 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 (see Section 3.3). The vertical axis denotes the mean number of misclassified 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 as a box surrounding the mean. The horizontal axis denotes the proportion of symmetric vectors in the training set; for example, when the abscissa is 48, 64 patterns of the training set consists of 48 symmetric and 16 asymmetric vectors, respectively.

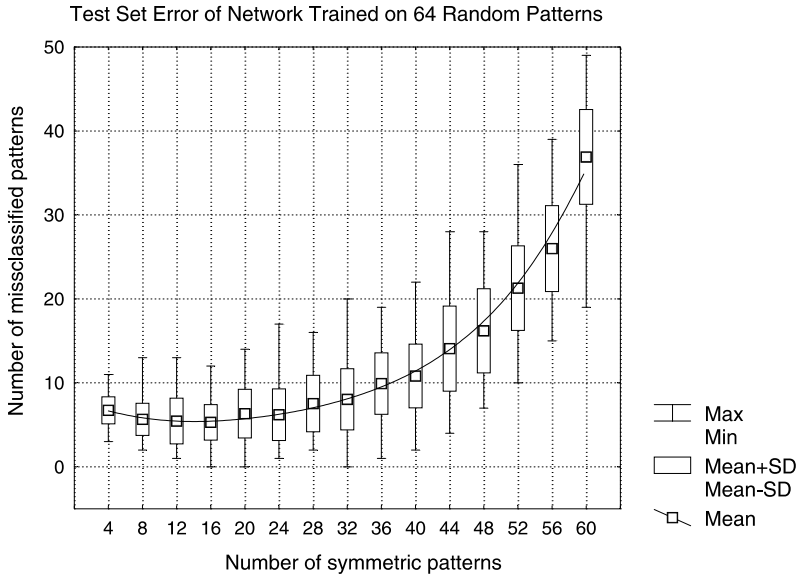

Fig. 1. Misclassification error for 64 patterns.

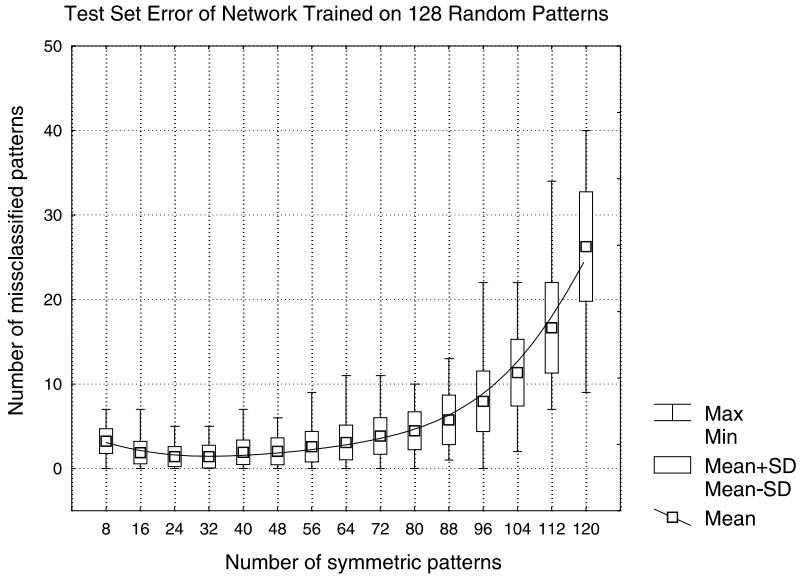

Fig. 2. Misclassification error for 128 patterns.

In Fig. 1 the classification error attains a minimum – corresponding to a maximum in generalisation ability – around values for the proportion between 12/64 and 20/64. The same observation repeats when we re-sample 128 vectors in Fig. 2. In that case, the optimal proportion seems to lie 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 have plotted in Figs. 3 and 4, the mean number of different patterns for each re-sampling proportion.

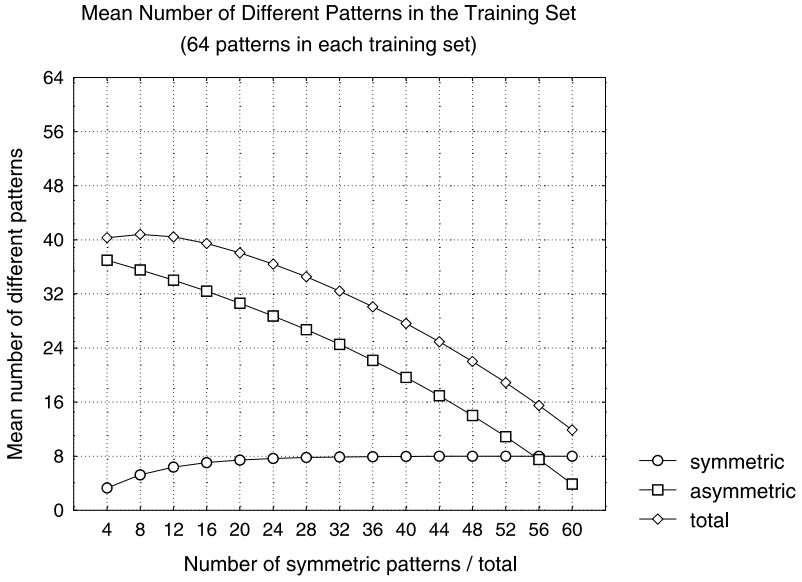

Fig. 3. Mean number of different patterns for 64 patterns.

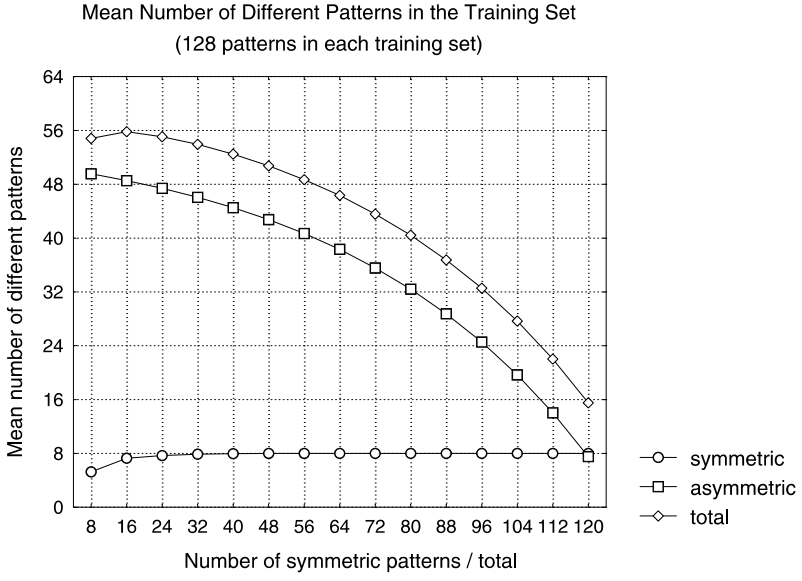

Fig. 4. Mean number of different patterns for 128 patterns.

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 performs better where the proportion is optimal. This indicates the importance of identifying the optimal proportion when training a network.

4. Bayesian classifier

This and subsequent sections are organised as follows: first we relate our study to the Bayesian formalism. Next we propose a minimum requirement on the network's ability to learn in the context of binary classification. Then we derive analytically the optimal re-sampling proportion. Lastly, an extension to multiple classes is presented.

4.1. Bayes decision rule

Bayesian 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 have sub-optimal 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 [3,7]).

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-component vector-valued random variable, and let $p(\mathbf{x} | \omega_j)$ be the probability density function for $\mathbf{x}$ conditioned on the state ω_j being the state of nature. Finally, let $P(\omega_j)$ be the a priori probability that nature is in state ω_j. Then, it is well known that the a posteriori probability $P(\omega_j | \mathbf{x})$ can be computed from $p(\mathbf{x} | \omega_j)$ by Bayes rule

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

where

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

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 $\lambda(\alpha_i \mid \omega_j)$ will then be incurred. Since $P(\omega_j \mid \mathbf{x})$ is the probability that the true state of nature is ω_j, the expected loss of action α_i is

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

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 assuming $P(\omega_j)$ that minimises the overall risk. A *decision rule* is a function $\alpha(\mathbf{x})$ that informs 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

$$R = \int_S R(\alpha(\mathbf{x}) \mid \mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{10}$$

where $\mathrm{d}\mathbf{x}$ denotes the d-dimensional volume element, and the integral extends over the entire feature space S. Clearly, if $\alpha(\mathbf{x})$ is chosen so that $R(\alpha(\mathbf{x}) \mid \mathbf{x})$ is as small as possible for every $\mathbf{x}$, then the overall risk is 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 $i = 1, \ldots, d$ and select the action α_i for which $R(\alpha_i \mid \mathbf{x})$ is minimum. (Note that if more than one action minimises $R(\alpha_i \mid \mathbf{x})$, 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 it is the best performance that can be achieved.

4.2. Binary classification

In the case of binary classification, the set of possible actions D reduces to $D = \{\alpha_1, \alpha_2\}$. Renaming the costs and benefits (i.e., negative costs) of classification to be

$$\lambda(\alpha_i \mid \omega_j) = c_{ij}, \; i, j = 1, 2, \tag{11}$$

the expected losses (or benefits) associated with the different actions α_i are written

$$\begin{aligned} R(\alpha_i \mid \mathbf{x}) &= \sum_{j=1}^{2} \lambda(\alpha_i \mid \omega_j) P(\omega_j \mid \mathbf{x}) \\ &= \sum_{j=1}^{2} c_{ij} \, P(\omega_j \mid \mathbf{x}). \end{aligned} \tag{12}$$

After substitution of Eq. (12) into Eq. (10), the overall risk can be written

$$\begin{aligned} R &= \int \sum_{i=1}^{2} R(\alpha_i \mid \mathbf{x}) p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &= \int \sum_{i=1}^{2} \sum_{j=1}^{2} c_{ij} \, P(\omega_j \mid \mathbf{x}) p(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \end{aligned} \tag{13}$$

In this expression, the decision of the networks is reflected by the term $P(\omega_j \mid \mathbf{x})$, while $p(\mathbf{x})\mathrm{d}\mathbf{x}$ depends on the data set on which the network will be used. We will see that if we make basic assumptions on $P(\omega_j \mid \mathbf{x})$ (i.e., the network behaviour in front of a new sample), we can deduce a re-sampling scheme that minimises the overall risk.

5. Minimum assumption on supervised learning

The sample space on which the network will be used is denoted by $\mathscr{S}$. It is partitioned into two subsets, $\mathscr{A}$ and $\mathscr{B}$ corresponding to the first and second classes, respectively. We will need the following definitions.

- Let q_a be the proportion of samples of class $\mathscr{A}$ in the whole sample space $\mathscr{S}$; i.e., $q_a = \int_{\mathscr{A}} p(\mathbf{x})\mathrm{d}\mathbf{x} / \int_{\mathscr{S}} p(\mathbf{x}) \, \mathrm{d}\mathbf{x}$. If we define q_b similarly, we have $q_a + q_b = 1$.
- The training set contains m different patterns, out of which a patterns belong to class $\mathscr{A}$ and b to class $\mathscr{B}$ ($m = a + b$). The subset of class $\mathscr{A}$ presented to the network is $\mathscr{A}_t$. We define similarly $\mathscr{B}_t$.
- c_{ab} is the cost of misclassifying a pattern from class $\mathscr{B}$, and c_{ba} the cost of misclassifying a pattern from $\mathscr{A}$. Similarly, c_{aa} and c_{bb} are the (negative) costs of classifying correctly.

5.1. Minimum assumption

When the network has been presented with a patterns of class $\mathscr{A}$ and b patterns of $\mathscr{B}$, it is assumed to behave as follows:

1. Classify correctly the $m = a + b$ patterns that are presented during the training phase.
2. Classify a pattern it has never seen with a probability a/m to class $\mathscr{A}$ and with a probability b/m to class $\mathscr{B}$.

Note that here the conditional probabilities $P(\alpha_a \mid \mathbf{x}) = a/m$ and $P(\alpha_b \mid \mathbf{x}) = b/m$, do not depend on $\mathbf{x}$. Because a real network extracts information from its input, it is expected to perform better than the classifier defined here. In this

sense, this classifier's performance can be understood as a lower limit on supervised learning ability.

5.2. Optimal proportion

The minimum assumptions in Section 5.1 allow us to rewrite the cost function of Eq. (13).

$$\begin{aligned} R &= \int \sum_{i=1}^{2}\sum_{j=1}^{2} c_{ij}\, P(\omega_j \,|\, \mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x}, \\ &= \int_{\mathscr{A}\setminus\mathscr{A}_t} \frac{a}{m}(c_{aa}+c_{ba})p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &\quad + \int_{\mathscr{B}\setminus\mathscr{B}_t} \frac{b}{m}(c_{bb}+c_{ab})p(\mathbf{x})\,\mathrm{d}\mathbf{x}, \\ &= \alpha c_a \int_{\mathscr{A}\setminus\mathscr{A}_t} p(\mathbf{x})\,\mathrm{d}\mathbf{x} + (1-\alpha)c_b \int_{\mathscr{B}\setminus\mathscr{B}_t} p(\mathbf{x})\,\mathrm{d}\mathbf{x} \end{aligned} \tag{14}$$

where $\alpha = a/m$, $c_a = c_{aa} + c_{ba}$, and $c_b = c_{bb} + c_{ab}$.

The sample space $\mathscr{M}$ on which the network will be applied is generally available under the form of a (hopefully) representative finite test set. Calling A and B the total number of samples in classes $\mathscr{A}$ and $\mathscr{B}$, respectively, and N their total, we have

$$\int_{\mathscr{A}\setminus\mathscr{A}_t} p(\mathbf{x})\mathrm{d}\mathbf{x} = A - a = q_a N - \alpha m, \tag{15}$$

$$\int_{\mathscr{B}\setminus\mathscr{B}_t} p(\mathbf{x})\mathrm{d}\mathbf{x} = B - b = q_b N - (1-\alpha)m. \tag{16}$$

Then, Eq. (14) becomes

$$\begin{aligned} R = {} & c_a\alpha(q_a N - \alpha m) + c_b(1-\alpha) \\ & \times (q_b N - (1-\alpha)m). \end{aligned} \tag{17}$$

To determine the proportion of patterns that minimises the cost function, we differentiate Eq. (17) with respect to α:

$$\begin{aligned} \frac{\partial R}{\partial \alpha} = {} & 2\alpha m(c_a + c_b) + c_b q_b N - c_a q_a N \\ & - m(c_a + c_b). \end{aligned} \tag{18}$$

Therefore, we have

$$\begin{aligned} \frac{\partial R}{\partial \alpha} = 0 &\iff 2\alpha^* m(c_a + c_b) + c_b q_b N - c_a q_a N \\ &\quad - m(c_a + c_b) = 0 \\ &\iff \alpha^* = \frac{1}{2} + \frac{N}{2m}\left(\frac{c_a q_a - c_b q_b}{c_a + c_b}\right). \end{aligned} \tag{19}$$

This α^* minimises R because the second derivative is always positive:

$$\frac{\partial^2 R}{\partial^2 \alpha} = 2m(c_a + c_b) > 0. \tag{20}$$

The resulting optimal proportions are:

$$1 - \alpha^* = \frac{1}{2} + \frac{N}{2m}\frac{(q_b c_b - q_a c_a)}{c_a + c_b}; \tag{21}$$

$$\alpha^* = \frac{1}{2} + \frac{N}{2m}\frac{(q_a c_a - q_b c_b)}{c_a + c_b}. \tag{22}$$

Because $q_a + q_b = 1$, we can eliminate q_b from Eq. (22):

$$\alpha^* = \frac{1}{2} + \frac{N}{2m}\left(q_a - \frac{c_b}{c_a + c_b}\right). \tag{23}$$

5.3. Natural conditions on α

In the above analysis, the derivatives have been calculated without taking into account the following natural conditions on α.

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

$$0 \leqslant \alpha \leqslant 1. \tag{24}$$

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

$$m\alpha \leqslant N q_a, \quad m(1-\alpha) \leqslant N(1-q_a).$$

This can be rewritten as

$$1 - (N/m)(1-q_a) \leqslant \alpha \leqslant N/m q_a. \tag{25}$$

If Eq. (23) gives a solution outside of the feasible region defined by Eqs. (24) and (25), the minimum in that range will be the α which is closest from the solution of Eq. (23). The reason is that the second derivative in Eq. (20) is always

positive, there is no local minima and R increases with the distance from the minimum point. Different cases may arise; they are analysed in the subsequent development.

Consider the value of α^* given in Eq. (23). We have 4 cases.

Case 1. If $1-(N/m)(1-q_a)\leqslant 0$ and $\alpha^*<0$, then α^* must be set to 0.

Case 2. If $1-(N/m)(1-q_a)>0$ and $\alpha^*<1-(N/m)(1-q_a)$, then α^* must be set to $1-(N/m)(1-q_a)$.

Case 3. If $(N/m)q_a<1$ and $\alpha^*>(N/m)q_a$, then α^* must be set to $(N/m)q_a$.

Case 4. If $(N/m)q_a\geqslant 1$ and $\alpha^*>1$, then α^* must be set to 1.

In all other cases, α^* remains to its original value given in Eq. (23). We can state these rules as follows. We first compute from Eq. (23) the optimal number of patterns in each class. If there are 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. A typical solution is presented in Fig. 5, where the slopes are indicated along the corresponding line segments.

The distribution of samples between the two classes need not be the same in the available training set and the prospective set to which the network will be actually used. In terms 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_b as a priori frequencies in the prospective set. If one of the classes does not contain enough samples to satisfy the optimal proportion, then complete the training set with patterns of the other class.

6. Bounds for the optimal proportion

The classifier defined in Section 5 corresponds to a minimum assumption on learning ability. It is

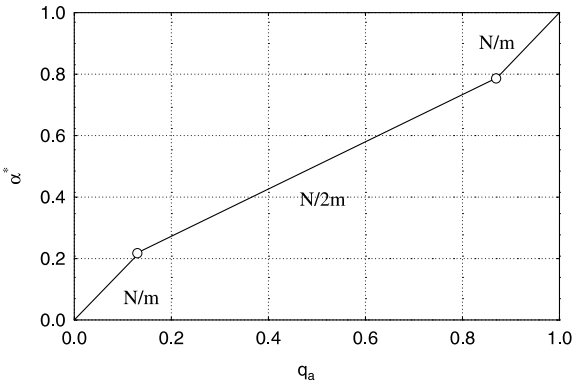

Fig. 5. Optimal α as a function of q_a.

expected to perform poorly because it takes into account only the distribution of outputs observed during the training phase, and ignores the input vector $\mathbf{x}$. Actually, a real network will better classify the patterns and we can relax the minimum assumptions of Section 5 as follows:

1. Classify correctly the $m=a+b$ patterns that were presented during the training phase.
2. Classify incorrectly a pattern it has never seen with a probability $\alpha'=\alpha-\gamma_a$ to class $\mathscr{A}$ and with a probability $\beta'=1-\alpha-\gamma_b$ to class $\mathscr{B}$.

Clearly, γ_a and γ_b must be positive. Moreover, α' and β' being positive, it entails that $\gamma_a\leqslant\alpha$ and $\gamma_b\leqslant 1-\alpha$.

Then, we can rewrite the expression of the cost function in Eq. (17) as

$$\begin{aligned}R&=c_a(q_aN-\alpha m)\beta'+c_b(q_bN-m(1-\alpha))\alpha'\\&=c_a(q_aN-\alpha m)(1-\alpha-\gamma_b)\\&\quad+c_b(q_bN-m(1-\alpha))(\alpha-\gamma_a).\end{aligned}\tag{26}$$

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

$$\begin{aligned}\frac{\partial R}{\partial\alpha}&=2\alpha m(c_a+c_b)+(c_bq_bN-c_aq_aN)\\&\quad-m(c_a+c_b)-m(c_b\gamma_a-c_a\gamma_b).\end{aligned}\tag{27}$$

Equating this expression to zero, and solving this equation gives

$$\alpha^*=\frac{1}{2}+\frac{N}{2m}\frac{(c_aq_a-c_bq_b)}{(c_a+c_b)}+\frac{c_b\gamma_a-c_a\gamma_b}{2(c_a+c_b)}.\tag{28}$$

This corresponds to Eq. (22) considering the relative ability of the network to learn patterns across binary classes. Because $0 \leqslant \gamma_a \leqslant \alpha$ and $0 \leqslant \gamma_b \leqslant 1-\alpha$, we obtain a lower bound for α^* when $\gamma_a = 0$ and $\gamma_b = 1-\alpha$:

$$\alpha^*_{\min} = \frac{1}{2} + \frac{N}{2m}\frac{c_a q_a - c_b q_b}{c_a + c_b} - \frac{c_a(1-\alpha^*_{\min})}{2(c_a+c_b)}$$
$$\Rightarrow \alpha^*_{\min} = \frac{c_b}{c_a+2c_b} + \frac{N}{m}\frac{c_a q_a - c_b q_b}{c_a+2c_b}. \quad (29)$$

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

$$\alpha^*_{\max} = \frac{1}{2} + \frac{N}{2m}\frac{c_a q_a - c_b q_b}{c_a + c_b} - \frac{c_b \alpha^*_{\max}}{2(c_a+c_b)}$$
$$\Rightarrow \alpha^*_{\max} = \frac{c_a+c_b}{2c_a+c_b} + \frac{N}{m}\frac{c_a q_a - c_b q_b}{2c_a+c_b}. \quad (30)$$

6.1. Natural conditions on $\alpha^*_{\min}$ and $\alpha^*_{\max}$

The same conditions on α^* in Eqs. (24) and (25) also apply to $\alpha^*_{\min}$ and $\alpha^*_{\max}$. This means that $\alpha^*_{\min}$ will be the maximum among the values defined by Eqs. (24), (25), and (29). Similarly, $\alpha^*_{\max}$ will be the minimum of 1, $(N/m)q_a$, and the α defined in Eq. (30).

Simultaneously, we must have $\alpha^*_{\min} < 1$ and $\alpha^*_{\max} > 0$. This is achieved if

$$\frac{m}{N} > \left| q_a - \frac{c_b}{c_a+c_b} \right|. \quad (31)$$

Below this number of samples in the training set, we cannot expect any reduction of the cost R. We can now restate the conditions on α^* of Section 5.3: Cases 2 and 3 remain valid, but the condition in Eq. (31) together with Eq. (23), replaces all the other cases.

A typical situation is depicted in Fig. 6 where the bold line represents α^*, whereas the bold dotted lines correspond to $\alpha^*_{\max}$ and $\alpha^*_{\min}$.

6.2. High risk misclassification

A very unbalanced training set ($q_a \ll q_b$) is particularly critical when the comparative cost associated with misclassification is high ($c_a \gg c_b$). This happens in many practical examples of disease diagnosis, car insurance (e.g., traffic accident predictions), default predictions, credit assignments, etc.

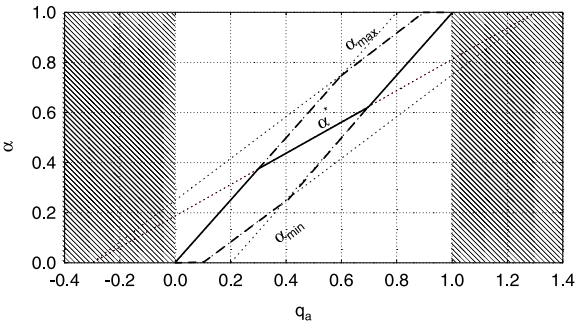

Fig. 6. Optimum, minimum and maximum α as a function of q_a.

To lighten notations we define $C = c_b/(c_a + c_b) \ll 1$. Similarly to the arguments in Section 5.3, the discussion is divided into different cases.

1. If $0 < \alpha^* < 1 - (N/m)(1-q_a)$, α^* should be set to $1-(N/m)(1-q_a)$. Substituting the expression of α^* in Eq. (23), this condition becomes
$$\frac{1}{2} + \frac{N}{2m}(q_a - C) < 1 - \frac{N}{m}(1-q_a)$$
$$\iff \frac{m}{N} > 2 - (q_a + C). \quad (32)$$
Since $0 < q_a \ll 1$ and $0 < C \ll 1$ entail that the right-hand side of the last relation exceeds 1. Therefore, this situation is impossible.
2. If $(N/m)q_a < \alpha^* < 1$, α^* is set to $(N/m)q_a$. This happens when
$$\frac{1}{2} + \frac{N}{2m}(q_a - C) > \frac{N}{m}q_a \iff \frac{m}{N} > q_a + C. \quad (33)$$
Given that $q_a \ll 1$ and $C \ll 1$, this last condition is easily met.
3. In all other cases, we have $m/N \leqslant q_a + C$. Moreover, we need $m/N > |\, q_a - C \,|$ following Eq. (31) to ensure that $\alpha^* \in (0,1)$. This means that m/N must lie on an interval of length $2C$ centred on q_a. This may be possible, but it is highly unlikely because $C \ll 1$.

Consequently, the expression of the optimal α becomes

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

As we have $N \geqslant m$, this implies that the optimal proportion corresponds to an over-representation of the less populated class.

A re-sampling to 50–50 would happen if $m = 2Nq_a$.

6.3. Equal costs

When the costs are equal, the optimal proportions for the minimum assumption network defined in Eq. (23) simplifies to

$$\alpha^* = \frac{1}{2} + \frac{N}{4m}(q_a - q_b) = \frac{1}{2} + \frac{N}{4m}(2q_a - 1). \tag{34}$$

If $q_a > 1/2$, then $\alpha^* > 1/2$. When the size of the training set m increases, the optimal proportion re-centres the a priori distribution.

Following Eq. (28), the re-sampling proportion of a network with a higher learning capability will be optimal, when

$$\alpha^* = \frac{1}{2} + \frac{N}{4m}(2q_a - 1) + \frac{\gamma_a - \gamma_b}{4}. \tag{35}$$

Comparing this result with Eq. (34), 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 ($\gamma_a = \gamma_b$).

The lower and the upper bounds for the optimal proportion are

$$\alpha^*_{\min} = \frac{1}{3} + \frac{N}{3m}(2q_a - 1) \tag{36}$$

and

$$\alpha^*_{\max} = \frac{2}{3} + \frac{N}{3m}(2q_a - 1). \tag{37}$$

If both $\alpha_{\min}$ and $\alpha_{\max}$ defined in Eqs. (36) and (37) lie in the feasible region, the length of the interval is $\alpha^*_{\max} - \alpha^*_{\min} = 1/3$. From the conditions expressed in Eqs. (24) and (25), we may conclude that the optimal proportion lies in an interval of length smaller than $1/3$.

6.4. Symmetry detection numerical experiment

In the numerical experiments of Section 3.4, there are 64 patterns with eight symmetric ones. Taking costs to be equal, Eq. (23) becomes

$$\alpha^* = \frac{1}{2} + \frac{64}{2m}\,\frac{56 - 8}{2 \times 64} = \frac{1}{2} + \frac{12}{m}. \tag{38}$$

For $0 \leqslant m \leqslant 24$, as $\alpha \leqslant 1$, 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. In Fig. 1, this corresponds to a number of symmetric patterns equal to roughly 13. The a priori proportion of 12.5% for symmetric patterns corresponds to eight symmetric patterns in Fig. 1.

Following Eqs. (36) and (37), the lower and upper bounds are: $1 \geqslant \alpha \geqslant 0.73$ and $0 \leqslant 1 - \alpha \leqslant 0.27$. In Fig. 1, this corresponds to a number of symmetric patterns between 0 and 18.

The benefit of re-sampling can be evaluated using Eq. (17). For $m = 40$ and $\alpha = q_a$, the total mean risk is $R = 5.25$, while using the optimal proportion $\alpha^* = 80\%$, we obtain a total mean risk of 4.8. The re-sampling to 50–50 is less advantageous: $R = 12$. The total mean risk is guaranteed to remain under eight because $R(\alpha^*_{\min}) = 6.48$ and $R(\alpha^*_{\max}) = 8$. This is reasonable because it corresponds to the classification of all patterns to the asymmetric majority class.

7. Cross-validation

The essence of neural learning is to encode an input–output mapping into the synaptic weights and thresholds of a multilayer perceptron. The hope is that the network becomes well trained so that it learns enough from the patterns in the training set to generalise to new patterns. 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 architectures (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 [6,10,19,20]). 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: an estimation subset used to select the model and a validation subset used to test or validate the model.

The motivation 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 may end up over-fitting 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 a large neural network with good generalisation ability. For example, we may use cross-validation to determine the multilayer perceptron with optimal number of hidden neurons, and to decide when it is best to stop training.

This type of cross-validation 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 patterns. In such a situation we may use *multifold cross-validation* by dividing the available set of N examples into K subsets with $K > 1$ and K divisible into N. The model is then 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. However, 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 $1 < K \leqslant N$.

Let us examine the standard case of training a network to map a continuous function, and test it using the cross-validation technique. We assume that the available data set represents fairly enough the distribution of the sample in the prospective set. If this is not verified, the adaptation to the practical rule in Section 5.3 can be used.

Although the function takes continuous values, the data set typically contains a finite number of samples. It is possible to artificially recreate the conditions for a binary classification by setting a threshold ϵ: we divide the data into two sets, one of them containing the sample whose output value is over the given threshold.

The optimal α for a given ϵ is obtained from Eq. (23). It implies that the training set must contain a^* samples of class $\mathscr{A}$, with a^* defined by

$$a^*(\epsilon) = m\alpha^*(\epsilon) = \frac{m}{2} + \frac{N}{4}(2q_a(\epsilon) - 1). \tag{39}$$

Because a similar division of the training set is possible for a value $\epsilon + h$ of the threshold, we also have

$$\begin{aligned} a^*(\epsilon + h) &= m\alpha^*(\epsilon + h) \\ &= \frac{m}{2} + \frac{N}{4}(2q_a(\epsilon + h) - 1). \end{aligned} \tag{40}$$

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

$$m(\alpha^*(\epsilon+h) - \alpha^*(\epsilon)) = \frac{N}{2}(q_a(\epsilon+h) - q_a(\epsilon)). \tag{41}$$

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.

8. Optimal proportions for multiple classes

In this section, we briefly outline the argument of Section 5 to many classes.

8.1. Notations

We assume that the sample space is composed of J classes, each of which is denoted by 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 misclassifying 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 the following:

- 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.

8.2. Optimal proportions

Making the minimum assumption on the learning ability of the network given in Section 5.2, we derive the mean error of misclassifying a pattern of I_i into I_j:

$$\mathrm{Err}_{ij} = (Nq_i - mp_i)p_j E(i, j). \tag{42}$$

The total error associated with the class I_i is given by

$$\mathrm{Err}_i = \sum_{j=1}^{J} (Nq_i - mp_i)p_j E(i, j), \tag{43}$$

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

$$\mathrm{Err} = \sum_{i=1}^{J} \mathrm{Err}_i = \sum_{i=1}^{J} \sum_{j=1}^{J} (Nq_i - mp_i)p_j E(i, j). \tag{44}$$

We have the following constrains on the p_i:

1. The p_i are proportions. Therefore,

$$\sum_{j=1}^{J} p_j = 1. \tag{45}$$

2. Proportions are always positive:

$$p_j \geqslant 0. \tag{46}$$

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

$$mp_j \leqslant Nq_j. \tag{47}$$

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

The Lagrangian for this problem becomes

$$\begin{aligned} L = & \sum_{i=1}^{J} \sum_{j=1}^{J} E(i, j)(Nq_i - mp_i)p_j \\ & - \lambda \left(\sum_{j=1}^{J} p_j - 1 \right) - \sum_{j=1}^{J} \mu_j (p_j - y_j^2) \\ & - \sum_{j=1}^{J} v_j (mp_j - Nq_j + z_j^2). \end{aligned} \tag{48}$$

To find the 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 v_j, as follows.

8.2.1. Derivatives of the Lagrangian

1. Derivatives with respect to the proportions:

$$\begin{aligned} \frac{\partial L}{\partial p_k} = & \frac{\partial}{\partial p_k} \left\{ \sum_{i=1}^{J} \sum_{j=1}^{J} E(i, j)(Nq_i - mp_i)p_j \right. \\ & - \lambda \left(\sum_{j=1}^{J} p_j - 1 \right) - \sum_{j=1}^{J} \mu_j (p_j - y_j^2) \\ & \left. - \sum_{j=1}^{J} v_j (mp_j - Nq_j + z_j^2) \right\} \\ = & -m \sum_{j=1}^{J} E(k, j)p_j + \sum_{i=1}^{J} E(i, k) \\ & \times (Nq_i - mp_i) - \lambda - \mu_k - mv_k \\ = & \sum_{i=1}^{J} \{E(i, k)Nq_i - mp_i(E(k, i) + E(i, k))\} \\ & - \lambda - \mu_k - mv_k. \end{aligned} \tag{49}$$

2. Derivatives with respect to the slack variables y_k:

$$\frac{\partial L}{\partial y_k} = \frac{\partial}{\partial y_k}\left\{ -\sum_{j=1}^{J} \mu_j (p_j - y_j^2) \right\} = 2\mu_k y_k. \quad (50)$$

3. Derivatives with respect to the slack variables z_k:

$$\frac{\partial L}{\partial z_k} = \frac{\partial}{\partial z_k}\left\{ -\sum_{j=1}^{J} v_j (mp_j - Nq_j + z_j^2) \right\} = -2v_k z_k. \quad (51)$$

8.2.2. Second-order derivatives

We first relax the second and third conditions expressed in Eqs. (46) and (47), and postpone their treatment. Eq. (49) becomes:

$$\frac{\partial L}{\partial p_k} = \sum_{i=1}^{J} \{E(i,k)Nq_i - mp_i(E(k,i) + E(i,k))\} - \lambda. \quad (52)$$

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

$$\lambda = \sum_{i=1}^{J} \{E(i,1)Nq_i - mp_i(E(1,i) + E(i,1))\}. \quad (53)$$

After substitution of Eq. (53) into Eq. (52), we obtain

$$\frac{\partial L}{\partial p_k} = \sum_{i=1}^{J} \{E(i,k)Nq_i - mp_i(E(k,i) + E(i,k))\} - \sum_{i=1}^{J} \{E(i,1)Nq_i - mp_i(E(1,i) + E(i,1))\} \quad (54)$$

$$= \sum_{i=1}^{J} \{(E(i,k) - E(i,1))Nq_i - (E(k,i) + E(i,k) - E(1,i) - E(i,1))mp_i\}. \quad (55)$$

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

$$\sum_{i=1}^{J} \{(E(i,k) - E(i,1))Nq_i - (E(k,i) + E(i,k) - E(1,i) - E(i,1))mp_i\} = 0. \quad (56)$$

The second derivative with respect to p_k is

$$\frac{\partial^2 L}{\partial^2 p_k} = \frac{\partial}{\partial p_k} \sum_{i=1}^{J} -(E(k,i) + E(i,k) - E(1,i) - E(i,1))mp_i \quad (57)$$

$$= mp_k(E(1,k) + E(k,1)) > 0. \quad (58)$$

There is no local minimum, and the solutions of Eq. (56) define the unique minimum in the subspace where $\sum_{j=1}^{J} p_j = 1$. Similar to Section 5.2, the feasible optimal solution will correspond to the proportions closest to the solutions of Eq. (56) which lies in the feasible region. The rest of the arguments follows along the lines described in Sections 5 and 6.

9. Conclusions

We presented a bootstrap approach to neural computations. We set up numerical experiments to assess empirically the impact of re-sampling on the network's ability to learn. The importance of the sample mixture in bootstrap training was 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, however, 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 classes contains fewer patterns and the associated cost of misclassifying them is very high. A simple method to estimate the optimal proportion for binary classification problems was proposed. The results were presented in the context of neural computations, but the methods apply to most of supervised learning systems, including decision trees.

Acknowledgements

The authors would like to thank anonymous referees for their detailed comments. The work of

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

References

[1] H. Akaike, Information theory and an extension of the maximum likelyhood principle, Second International Symposium on Information Theory (1973) 267–281.
[2] D.M. Allen, The relationship between variable selection and data augmentation and a method of prediction, Technometrics 16 (1974) 125–127.
[3] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[4] D. Dupret, M. Koda, Bootstrapping for neural network learning, APORS' 2000 – Asia Pacific Operations Research Societies, Conference CD-ROM, 2000.
[5] B. Efron, R.J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York, Tokyo, 1993.
[6] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing, New York, 1994.
[7] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley, Reading, MA, 1990.
[8] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA, 1991.
[9] IBM, Intelligent Miner for Relationship Marketing, Tokyo, IBM Japan Co. Ltd, 1999.
[10] J.P.C. Kleijnen, Bootstrapping and cross-validation of metamodels in simulation, in: Proceedings SAMO '98, European Commission and University of Venice, 1998, pp. 155–157.
[11] M. Koda, Stochastic sensitivity analysis method for neural network learning, International Journal of Systems Science 26 (1995) 703–711.
[12] M. Koda, Neural network learning based on stochastic sensitivity analysis, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 27 (1997) 132–135.
[13] M. Koda, Stochastic sensitivity analysis and Langevin simulation for neural network learning, Reliability Engeneering and System Safety 17 (1997) 71–78.
[14] T. Masters, Advanced Algorithms for Neural Networks, Wiley, 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 (1978) 461–464.
[17] SPSS, Clementine, Interactive Data Mining Tool, Tokyo, SPSS Japan Co. Ltd, 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. Series B: Methodological B 36 (1974) 111–147.
[20] M. Stone, An asymptotic equivalence of choice of model by cross-validation and akaike's criterion, Journal of the Royal Statistical Society. Series B: Methodological B 39 (1977) 44–47.