Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 21 (2008) 78–91
Neural Networks
www.elsevier.com/locate/neunet

The unimodal model for the classification of ordinal data

Joaquim F. Pinto da Costa[a,*], Hugo Alonso[b], Jaime S. Cardoso[c]

[a] *Faculdade de Ciências da Universidade do Porto, Departamento de Matemática Aplicada/CMUP, Rua do Campo Alegre, 687, 4169-007 Porto, Portugal*
[b] *Faculdade de Ciências da Universidade do Porto, Departamento de Matemática Aplicada/UIMA, Rua do Campo Alegre, 687, 4169-007 Porto, Portugal*
[c] *Faculdade de Engenharia da Universidade do Porto/INESC Porto, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal*

Received 14 December 2006; accepted 9 October 2007

Abstract

Many real life problems require the classification of items into naturally ordered classes. These problems are traditionally handled by conventional methods intended for the classification of nominal classes where the order relation is ignored. This paper introduces a new machine learning paradigm intended for multi-class classification problems where the classes are ordered. The theoretical development of this paradigm is carried out under the key idea that the random variable class associated with a given query should follow a unimodal distribution. In this context, two approaches are considered: a parametric, where the random variable class is assumed to follow a specific discrete distribution; a nonparametric, where the random variable class is assumed to be distribution-free. In either case, the unimodal model can be implemented in practice by means of feedforward neural networks and support vector machines, for instance. Nevertheless, our main focus is on feedforward neural networks. We also introduce a new coefficient, r_{int}, to measure the performance of ordinal data classifiers. An experimental study with artificial and real datasets is presented in order to illustrate the performances of both parametric and nonparametric approaches and compare them with the performances of other methods. The superiority of the parametric approach is suggested, namely when flexible discrete distributions, a new concept introduced here, are considered.

Keywords: Classification; Ordinal data; Feedforward neural networks; Measures of association; Support vector machines

1. Introduction

Many supervised classification problems involve classifying examples (instances) into classes which have a natural ordering. Settings in which it is natural to rank instances arise in many fields, such as econometric modelling, information retrieval and collaborative filtering. Other applications include stock trading support systems, where one wants to predict, for instance, whether to buy, keep or sell a stock, and biomedical classification problems, where frequently the classes are ordered. In fact, we have found that in a great number of applications the classes are ordered, although that is almost never taken into account; most of the time conventional methods for nominal classes or regression are used.

The use of conventional methods of supervised classification is certainly possible, but first of all it is usually harder and slower to train with these methods and secondly the derived classifier might not be really appropriate (see Section 2). On the other hand, using regression methods introduces an arbitrary selection of numbers to represent the classes, which in turn influences both the prediction function and the usual measures of performance assessment that are used (see Section 4). However, the use of methods specifically designed for ordered classes results in simpler classifiers, making it easier to interpret the factors that are being used to discriminate among classes (Mathieson, 1995).

The works that take into account the information concerning the order relation between the classes date back at least to McCullagh (1980), where a general class of regression models for ordinal data, which eliminate the need for assigning scores to the classes, are presented. Herbrich, Graepel, and Obermayer (1999) applied the Principle of Structural Risk Minimization as used in support vector machines (Burges, 1998; Vapnik, 1998) to derive a learning algorithm based on large rank boundaries for the task of ordinal regression. Frank and Hall (2001) applied a simple method that, in conjunction with a decision tree learner, enables standard classification

* Corresponding author. Tel.: +351 220 402 270.
E-mail address: jpcosta@fc.up.pt (J.F. Pinto da Costa).

0893-6080/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2007.10.003

algorithms to make use of the ordering between the classes. More recently, Chu presented together with Ghahramani a probabilistic kernel approach to ordinal regression based on Gaussian processes (Chu & Ghahramani, 2005) and together with Keerthi two new support vector machine approaches for ordinal regression, which optimize multiple thresholds to define parallel discriminant hyperplanes for the ordinal scales (Chu & Keerthi, 2005). Other recent works are those by Shen and Joshi (2005), where the authors first studied the ranking, reranking, and ordinal regression algorithms proposed in the context of ranks and margins and then proposed a general framework, and by Li and Lin (2007), where the authors presented a reduction framework from ordinal regression to binary classification based on extended examples. In dos Santos Cardoso, Pinto da Costa, and Cardoso (2005) a novel framework for ordered classes, based on replicating the dataset, was introduced in the context of support vector machines. Finally, in a preliminary version of our work (Pinto da Costa & Cardoso, 2005), we proposed a new learning methodology which is extended and explored in various directions in this paper. Our approach to the problem of classification of ordinal data takes a very different perspective relative to that of other authors. It starts by analysing which characteristics a prediction function for this kind of data should possess. We were led to conclude that the distribution of the *a posteriori* class probabilities should be unimodal, so that the order relation between the different classes is respected. This new paradigm is introduced in Section 2. In Section 3.1, the parametric approach of this methodology is introduced and developed under the context of neural networks and then the nonparametric approach is introduced in Section 3.2. Section 3.3 is devoted to the exploration of this new paradigm in the context of regression using both neural networks and support vector machines.

The usual measures of performance comparison are, in our view, not adequate for assessing the quality of ordinal data classifiers. This is why we introduce a new measure of association in Section 4, which is then used to carry out a comparison between all proposed implementations and between these and some competing methods. In Section 5, we conduct an experimental study on artificial and real datasets. In Section 6 a new statistical concept, that of the flexibility of a discrete distribution, is presented in order to theoretically explain the practical results. Finally, we draw the main conclusions of our work in Section 7.

2. The unimodal paradigm

Let us start by discussing the supervised classification problem of separating K ordered classes $\mathcal{C}_1 < \cdots < \mathcal{C}_K$ in a feature space χ. In a supervised classification problem, the goal is to find a classifier represented by a map $f_T : \chi \to \{\mathcal{C}_i\}_{i=1}^K$ that minimises some cost functional relative to the ℓ examples in a given training set $T = \{(\mathbf{x}_i, \mathcal{C}_{\mathbf{x}_i})\}_{i=1}^{\ell} \subset \chi \times \{\mathcal{C}_i\}_{i=1}^K$. Bayes decision theory suggests classifying a new query point $\mathbf{x}$ in the class $\mathcal{C}_k$ maximising the *a posteriori* probability $P(\mathcal{C}_k|\mathbf{x})$, *i.e.*, a classifier f_T should be defined so that $f_T(\mathbf{x}) = \arg\max_{\mathcal{C}_k}\{P(\mathcal{C}_k|\mathbf{x})\}$. To that end, f_T must estimate the *a posteriori* probabilities. As an example, assume that we have a problem with $K = 5$ classes, namely that we are interested in classifying the weather temperature into five classes: {Very cold, Cold, Mild, Hot, Very hot}. There is clearly a natural ordering between these classes: Very cold < Cold < Mild < Hot < Very hot. Given a new query point $\mathbf{x}$ containing information about the weather, if the highest *a posteriori* probability is, for instance, $P(C_4|\mathbf{x})$, then we should have $f_T(\mathbf{x}) = \mathcal{C}_4$. Now, if we use a classifier which does not take into account the order relation between the classes, the second highest *a posteriori* probability can be, for instance, $P(\mathcal{C}_1|\mathbf{x})$. This does not make any sense of having most likely a Hot day and the second most likely a Very cold day. Given that there is an order relation between the classes, $\mathcal{C}_1 < \cdots < \mathcal{C}_5$, $\mathcal{C}_3$ and $\mathcal{C}_5$ are closer to $\mathcal{C}_4$ and therefore the second highest *a posteriori* probability should be attained in one of these classes. This means that if the most likely is a Hot day, then the second most likely should either be a Mild day or a Very hot day. More generally, the probabilities should decrease monotonically to the left and to the right of the class where the maximum probability is attained. This is the main idea behind the method we propose in this paper.

Our method assumes thus that in a supervised classification problem with ordered classes, the random variable class $\mathcal{C}_{\mathbf{x}}$ associated with a given query point $\mathbf{x}$ should follow a unimodal distribution. We impose unimodality in two ways. Firstly, in the so-called parametric approach, we assume a particular unimodal discrete distribution for $\mathcal{C}_{\mathbf{x}}$, and a classifier f_T estimates the *a posteriori* probabilities by estimating the parameters of the assumed distribution. Secondly, in the so-called nonparametric approach, we assume no distribution for $\mathcal{C}_{\mathbf{x}}$, hence for the *a posteriori* probabilities to be estimated by a classifier f_T; in this case, unimodality is imposed by new error measures which penalize nonunimodal distributions.

Before moving to the next section, a final remark should be made. Since we are dealing with ordered classes, we shall consider throughout the paper, without loss of generality, a bijective map $g : \{\mathcal{C}_i\}_{i=1}^K \to \{1, \ldots, K\}$ which assigns the number k to the class $\mathcal{C}_k$, *i.e.*, $g(\mathcal{C}_k) = k$.

3. Some unimodal classifiers

In this section we present three ways of implementing the unimodal paradigm in practice. First of all, in the parametric approach we can choose a unimodal statistical distribution, like the binomial or Poisson's, and force the output of a classifier to follow that distribution. Here, the classifier chosen by us is neural networks and also support vector machines, but other classifiers could be chosen as we plan to do in the future. Secondly, in the nonparametric approach, we show how to force the output of a neural network to be unimodal without assuming a statistical distribution. Finally, we show that the parametric approach can be implemented via regression; we just have to estimate the parameters of a statistical distribution, which are numeric, by a regression algorithm.

3.1. A network architecture for the classification of ordinal data: The unimodal parametric approach

In order to apply a neural network (Haykin, 1999) to solve a classification problem, we need first to design an equivalent function approximation problem by assigning a target value to each class. Then, for a feedforward architecture, the number of layers and neurons must be chosen: in fact, the number of input neurons corresponds to the number of attributes used to characterise each instance; the number of hidden layers and neurons is usually determined by model assessment and selection techniques, and is related to the problem complexity degree; finally, the number of output neurons depends on the number of classes in question. For instance, in a two-class problem, we can use binary target values of 1 for the first class, 0 for the other, and a network with a single output interpreted as an estimate of the probability that a given instance belongs to the first class. On the other hand, in a multi-class problem 1-of-K, with $K > 2$, we use target values of 1 for the correct class, 0 otherwise, and a network with K outputs, each one corresponding to a particular class and interpreted as the estimated probability that a given instance belongs to that class. Nevertheless, while in a two-class problem we can use the standard logistic function as the activation function of the output neuron, in a multi-class problem we need to use the softmax function in the output layer in order to satisfy the normalisation constraint on the total probability; moreover, mind that the targets in the multi-class case are not independent of each other. In either case, the training of the network is commonly performed using the popular mean square error. However, this approach does not take into account the order of the classes and is not, therefore, appropriate for ordinal multi-class classification problems. We have already introduced in Pinto da Costa and Cardoso (2005) a new method for classifying ordinal data. In this work we extend this new method in various directions.

As previously mentioned, the output layer in a network intended for classification usually has as many neurons as there are classes, namely K. Not only we adopt this, but also maintain the same order between the outputs as that existing between the classes. In order to impose a unimodal distribution at the network output, we start by assuming a unimodal probabilistic model whose unknown parameters are estimated by the network so that the *a posteriori* probabilities at the output neurons follow the assumed model. Many parametric models can be considered: binomial, truncated Poisson, hypergeometric, etc. We will explore the first two models, starting by detailing the binomial. Henceforth, we will refer to the network implementing the binomial model as the binomial network, etc (see Fig. 1).

In the binomial network, the output values follow the binomial distribution $B(K-1, p)$. This distribution is unimodal in most cases and when it has two modes these are for contiguous values, which makes sense in our case since we can have exactly the same probability for two classes. As this binomial distribution takes integer values in the set $\{0, 1, \ldots, K-1\}$, we will take value 0 to represent class $\mathcal{C}_1$, 1 to $\mathcal{C}_2$ and so on until value $K-1$ to represent class $\mathcal{C}_K$. As K is known, the only unknown parameter is the probability of success p. Hence, we consider a network architecture as in Fig. 1 and train it to adjust all connection weights from layer 1 to layer N. Note that the connections from layer N to layer $N+1$ have a fixed weight equal to one and serve only to forward the value of p to the output layer of the network where the probabilities from the binomial distribution are calculated. Therefore, there is no back-propagation during training from layer $N+1$ to layer N, only between layer N and its predecessors. For a given query $\mathbf{x}$, the output of layer N will be a single numerical value in the range $[0, 1]$, which we call $p_{\mathbf{x}}$. Then, the probabilities $P(\mathcal{C}_k|\mathbf{x})$ in layer $N+1$ are calculated from the binomial distribution:

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{(K-1)!\,p_{\mathbf{x}}^{k-1}(1-p_{\mathbf{x}})^{K-k}}{(k-1)!(K-k)!}, \quad k = 1, 2, \ldots, K.$$

In fact, these probabilities can be calculated recursively to save computing time, since

$$\frac{P(\mathcal{C}_k|\mathbf{x})}{P(\mathcal{C}_{k-1}|\mathbf{x})} = \frac{(K-k+1)\,p_{\mathbf{x}}}{(k-1)(1-p_{\mathbf{x}})} \quad \text{and so}$$
$$P(\mathcal{C}_k|\mathbf{x}) = P(\mathcal{C}_{k-1}|\mathbf{x})\frac{(K-k+1)\,p_{\mathbf{x}}}{(k-1)(1-p_{\mathbf{x}})}.$$

We start with $P(\mathcal{C}_1|\mathbf{x}) = (1-p_{\mathbf{x}})^{K-1}$ and compute the other probabilities, $P(\mathcal{C}_k|\mathbf{x})$, $k = 2, 3, \ldots, K$, using the above formula. For training, we have to choose a measure of error like, for instance, squared error or cross entropy (deviance) (Hastie, Tibshirani, & Friedman, 2001). Here, when the training instance $\mathbf{x}$ is presented, the error is defined by

$$\sum_{k=1}^{K}\left(P(\mathcal{C}_k|\mathbf{x}) - \delta(k - g(\mathcal{C}_{\mathbf{x}}))\right)^2 = \sum_{k=1}^{K}\left(\frac{(K-1)!\,p_{\mathbf{x}}^{k-1}(1-p_{\mathbf{x}})^{K-k}}{(k-1)!(K-k)!} - \delta(k - g(\mathcal{C}_{\mathbf{x}}))\right)^2, \quad (1)$$

where $\delta(k) = \begin{cases} 1 \text{ if } k = 0 \\ 0 \text{ otherwise} \end{cases}$ and $g(\mathcal{C}_{\mathbf{x}})$ is the number corresponding to the class $\mathcal{C}_{\mathbf{x}}$. Note that the only difference between the usual mean square error measure and (1) is the term $P(\mathcal{C}_k|\mathbf{x})$, which represents here the binomial probability distribution. The binomial network is trained to minimise the average value over all training cases of this error. Finally, in the test phase, we choose the class $\mathcal{C}_k$ which maximises the probability $P(\mathcal{C}_k|\mathbf{x})$.

We can assume other discrete distributions, apart from the binomial, for the final layer. Let us consider the use of the right truncated Poisson distribution, $P_t(\lambda)$, where the suffix t stands for truncated and λ is the distribution parameter. This distribution takes values in $\{0, 1, \ldots, K-1\}$, just as the binomial, but with probabilities $P(X_t = x) = c\frac{\mathrm{e}^{-\lambda}\lambda^x}{x!}$ $\forall x \in \{0, 1, \ldots, K-1\}$, where c is the constant such that the sum of the K probabilities equals 1. Given a query $\mathbf{x}$, it can be seen that the *a posteriori* probabilities can now be calculated recursively by

$$P(\mathcal{C}_k|\mathbf{x}) = P(\mathcal{C}_{k-1}|\mathbf{x})\frac{\lambda_{\mathbf{x}}}{k-1}, \quad k = 2, \ldots, K,$$

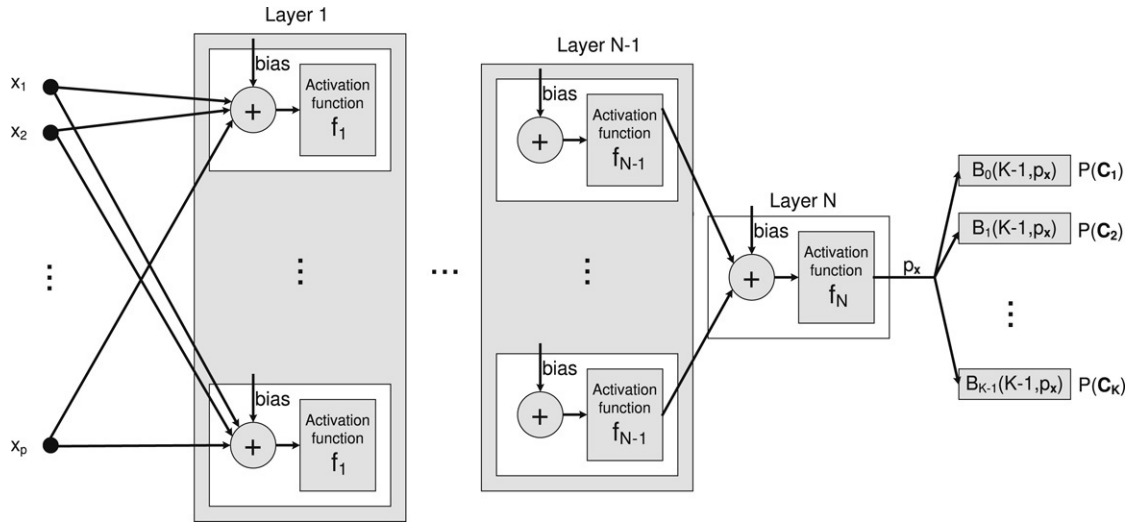

Fig. 1. Binomial network architecture.

and $P(\mathcal{C}_1|\mathbf{x}) = (\sum_{k=0}^{K-1} \frac{\lambda_{\mathbf{x}}^k}{k!})^{-1}$. Besides the way by which the *a posteriori* probabilities are calculated, the only difference between the binomial and the truncated Poisson networks is in the penultimate (Nth) layer: in the binomial network, the output of this layer corresponds to the value of p, whereas in the truncated Poisson network, it corresponds to the value of λ. In the usual Poisson distribution this parameter takes values in $]0, \infty[$. However, we are only interested in the range $]0, K[$, as the mode of this distribution is the integer part of λ, $\lfloor\lambda\rfloor$.

We will now introduce a nonparametric version of the unimodal model which, instead of choosing a statistical unimodal distribution for the outputs of the neural network, tries to oblige these outputs to be unimodal by imposing constraints on the error function.

3.2. A network architecture for the classification of ordinal data: The unimodal nonparametric approach

So far, the unimodal paradigm was implemented by assuming that the random variable class follows a particular unimodal discrete distribution that a neural network implements by estimating the distribution parameters. Now, we assume that there is no distribution for the class, and design learning schemes to train networks which try to fit the true class distribution by exploring the ordinal nature of the classification problem in the context of the unimodal paradigm.

To our knowledge, there is no way of imposing a unimodal output to a feedforward neural network without assuming a particular distribution. Therefore, we define error functions so that the network weights are adjusted during training in such a way that the network output $\mathbf{y} = (y_1, \ldots, y_K)$ tends to be unimodal. Mind that $\mathbf{y}$ corresponds to the estimated *a posteriori* probabilities, and recall that the error function of the conventional network is given by

$$E_c = \frac{1}{\ell} \sum_{j=1}^{\ell} e_c(\mathbf{x}_j),$$

where $\mathbf{x}_j \in \{\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_{\ell_1}^{(1)}, \ldots, \mathbf{x}_1^{(K)}, \ldots, \mathbf{x}_{\ell_K}^{(K)}\}$ is the jth training example belonging to one of the K classes with $\ell_1, \ldots, \ell_K$ examples, and where the error $e_c(\mathbf{x}_j)$ in classifying $\mathbf{x}_j$ is given by

$$e_c(\mathbf{x}_j) = \sum_{k=1}^{K} (y_k(\mathbf{x}_j) - \delta(k - g(\mathcal{C}_{\mathbf{x}_j})))^2, \tag{2}$$

$g(\mathcal{C}_{\mathbf{x}_j})$ being the number corresponding to class $\mathcal{C}_{\mathbf{x}_j}$. We start by defining a family of error functions $\{E_{u1}^{(\alpha)}\}$ which penalize classifying an example $\mathbf{x}$ from class $\mathcal{C}_k$ in class $\mathcal{C}_{k'}$ by considering the αth power of the distance between the numbers k and k' as

$$E_{u1}^{(\alpha)} = \frac{1}{\ell} \sum_{j=1}^{\ell} e_{u1}^{(\alpha)}(\mathbf{x}_j),$$

where

$$e_{u1}^{(\alpha)}(\mathbf{x}_j) = \sum_{k=1}^{K} (\delta(k - g(\mathcal{C}_{\mathbf{x}_j}))(y_k(\mathbf{x}_j) - 1)^2 + (1 - \delta(k - g(\mathcal{C}_{\mathbf{x}_j})))|k - g(\mathcal{C}_{\mathbf{x}_j})|^{\alpha} y_k^2(\mathbf{x}_j)). \tag{3}$$

This is a weighted sum of the K components of the vector containing the difference between the output vector $\mathbf{y}(\mathbf{x}_j) = (y_1(\mathbf{x}_j), \ldots, y_K(\mathbf{x}_j))$ and the vector representing the class of $\mathcal{C}_{\mathbf{x}_j}$, $(0, \ldots, 0, 1, 0, \ldots, 0)$. The weights take value 1 (when $k = g(\mathcal{C}_{\mathbf{x}_j})$), $2^{\alpha}, 3^{\alpha}, \ldots$, as k moves away from $g(\mathcal{C}_{\mathbf{x}_j})$. The implementation of the unimodal paradigm stems from the fact that a network trained with a member of $\{E_{u1}^{(\alpha)}\}$ tends to produce a unimodal output as the penalty term α tends to infinity. Nevertheless, we can only aim at getting in practice a near unimodal network output, since $\alpha < \infty$. Finally, it is straightforward to see that $e_c = e_{u1}^{(0)}$, hence $E_c = E_{u1}^{(0)}$; in this way, it can be seen that the conventional network does not penalize deviates between true and predicted classes, and therefore does not make the classification error an increasing function of such deviate.

Besides the proposed family of error functions, we also consider the possibility of taking the classification error to be proportional to the number of modes at the network output as

$$E_{u2} = \frac{1}{\ell}\sum_{j=1}^{\ell} e_{u2}(\mathbf{x}_j),$$

where

$$e_{u2}(\mathbf{x}_j) = M(\mathbf{x}_j)e_c(\mathbf{x}_j) \tag{4}$$

and $M(\cdot)$ denotes the number of modes at the network output and is defined by

$$M(\mathbf{x}_j) = I(y_1(\mathbf{x}_j) > y_2(\mathbf{x}_j)) + I(y_K(\mathbf{x}_j) > y_{K-1}(\mathbf{x}_j)) + \sum_{k=2}^{K-1} I(y_k(\mathbf{x}_j) > y_{k-1}(\mathbf{x}_j))I(y_k(\mathbf{x}_j) > y_{k+1}(\mathbf{x}_j)),$$

I being the indicator function. In order to apply back-propagation during training, it should be noted that the error function must be differentiable, so we take the approximation $I(x) \simeq \frac{1}{1+e^{-sx}}$ for a sufficiently large s ($s = 10\,000$ in all practical implementations).

3.3. The binomial ordinal classification model via regression

The binomial and truncated Poisson networks led to very promising results in the datasets considered by us as we shall see, although it is possible that a more flexible distribution (see Section 6) with more than one parameter to estimate is needed for other datasets. Nevertheless, and for the time being, we will look to the binomial ordinal classification model from a different perspective, namely that of usual regression. Mind that the same could be done for the truncated Poisson model.

As is obvious from Section 3.1, the binomial network estimates the parameter p, which is a real number in the range $[0, 1]$, and the final layer in the network is only intended to compute the *a posteriori* probabilities $P(\mathcal{C}_k|\mathbf{x}), k = 1, \ldots, K$, and thus the error function. From this perspective, the binomial network is applied as a regression model. However, there are two main differences between the binomial network and other models which treat ordinal classification problems as regression problems. Firstly, in the context of applying the latter models, arbitrary numbers are usually assigned to the classes and then usual regression is used, although certainly influenced by the arbitrary numbers chosen. Secondly, the error function is not the same we considered for the binomial network, *i.e.*, $\frac{1}{\ell}\sum_{\mathbf{x}}\sum_{k=1}^{K}(P(\mathcal{C}_k|\mathbf{x}) - \delta(k - g(\mathcal{C}_\mathbf{x})))^2$, where $g(\mathcal{C}_\mathbf{x})$ is the number corresponding to the class of $\mathbf{x}$ and $P(\mathcal{C}_k|\mathbf{x})$ depends on the estimated value of p; instead, the usual Mean Square Error (MSE), where $P(\mathcal{C}_k|\mathbf{x})$ is computed under no assumption on a probabilistic model, is quite often used. We will now drop this second difference and consider estimating the value of the parameter p, not $P(\mathcal{C}_k|\mathbf{x})$, using the usual MSE. Two algorithms will be used to suit this purpose: neural networks and support vector machines. To treat the problem as a regression one, the variable class needs to be replaced by a numerical variable corresponding to the value of p. It is easy to see that in the binomial distribution $B(K-1, p)$, if the value of p is in the range $[0, \frac{1}{K})$ then the distribution mode is attained at 0, which we assume to represent the first class. Generally, if p is in the range $[\frac{i}{K}, \frac{i+1}{K}), i = 0, 1, \ldots, K-1$, then the mode is attained at i and so the corresponding class is $\mathcal{C}_{i+1}$. We will take the mean value of these intervals to represent the value of p for each class, *i.e.*, if an observation belongs to class $\mathcal{C}_{i+1}$ then the corresponding value of p will be $\frac{i+0.5}{K}$. Having replaced the variable class by the numerical variable p, we can now use any regression algorithm. In the test phase, if for a test query $\mathbf{x}$ we obtain the answer $p_\mathbf{x}$, then the corresponding class will be $\lfloor Kp_\mathbf{x} \rfloor + 1$.

3.4. Practical implications of the unimodal idea

To implement the unimodal idea to solve a classification problem with $K > 2$ classes in neural networks, we have seen that we can choose an architecture either with one output neuron in the parametric approach (as the final layer of the network in Fig. 1 serves only to compute the *a posteriori* probabilities according to the chosen distribution) or with K output neurons in the nonparametric approach. The first option gave in general the best results in the study that we shall present in Section 5. Then, in order to train the parametric networks, with back-propagation, we can either choose the error function (1) or treat the problem as regression (Section 3.3) and use the usual MSE.

To use the error function (1), we have to compute the *a posteriori* probabilities $P(\mathcal{C}_k|\mathbf{x})$ using the binomial, Poisson, or other discrete distributions. Each class C_k is represented by a vector of size K containing 0 everywhere but in the kth position, whose value is 1; this is what is usually done in neural networks for classification, and therefore our method does not require additional work in the dataset preparation when compared to other common approaches.

To use the usual MSE, the class corresponding to an instance $\mathbf{x}$ will be represented by a numerical value $p_\mathbf{x} \in [0, 1]$ (see the last paragraph of the last subsection), and then the usual regression methods, like neural networks, SVMs and others, are used.

In the nonparametric networks, where there are K outputs, one of the error functions (2)–(4) should be used.

We can of course implement the unimodal idea with other classifiers. We have seen how to do it with SVMs in the regression context (Section 3.3). Actually, by approaching the problem from this perspective, any algorithm for regression can be used. In the classification perspective, we are now mapping the problem of using the unimodal paradigm into decision trees and other classification algorithms can be used.

As a final note, we would like to emphasize what we believe is a practical advantage of our method when compared to traditional methods. Our unimodal models, in particular the parametric ones, are less complex. For instance, the binomial or Poisson networks have only one output neuron regardless of the number of classes in the problem. This is in contrast with traditional networks, where the number of output neurons grows with the number of classes. Eventually, we might have to

use other discrete distributions with more than one parameter to estimate in highly complex problems (as we plan for our future work), but the corresponding number of output neurons will certainly be much smaller than the number of classes.

4. How to measure the performance of ordinal data classifiers?

In supervised classification problems with ordered classes, it is common to assess the performance of the classifier using measures which are not really appropriate. Very often, every misclassification is considered equally costly and the Misclassification Error Rate (MER) is used. Two other measures that are also usually used are the Mean Square Error (MSE) and the Mean Absolute Deviation (MAD). Both of these treat the problem as if it were a regression problem, *i.e.*, the performance of a classifier f_T is assessed in a dataset $\mathcal{O} \subset \mathcal{X}$ through

$$\text{MSE} = \frac{1}{\text{card}(\mathcal{O})} \sum_{\mathbf{x}\in\mathcal{O}} (g(\mathcal{C}_{\mathbf{x}}) - g(f_T(\mathbf{x})))^2,$$

and

$$\text{MAD} = \frac{1}{\text{card}(\mathcal{O})} \sum_{\mathbf{x}\in\mathcal{O}} |g(\mathcal{C}_{\mathbf{x}}) - g(f_T(\mathbf{x}))|,$$

where $g(\cdot)$ corresponds to the number assigned to a class. However, this assignment is arbitrary and the numbers chosen to represent the existing classes will evidently influence the performance measurement given by MSE or MAD. Still, these two measures are in a way somewhat better than MER, because they take values which increase with the absolute differences between "true" and "predicted" class numbers and so the misclassifications are not taken to be equally costly. In order to avoid the influence of the numbers chosen to represent the classes on the performance assessment, we should only look at the order relation between "true" and "predicted" class numbers. The use of Spearman's rank correlation coefficient, r_S, (Press, Flannery, Teukolsky, & Vetterling, 1992; Spearman, 1904) and specially Kendall's tau-b, τ_b, (Press et al., 1992) is a step forward in that direction. For instance, in order to get r_S, we start by defining two rank vectors of length n, corresponding to the size of the dataset, which are associated with the variables C and $\hat{C}$, corresponding respectively to the numbers representing the true and predicted classes. Obviously, there will be many examples in the dataset with common values for those variables and we do what is common, *i.e.*, use average ranks. If $\mathbf{R}$ and $\mathbf{Q}$ represent the two rank vectors, then $r_S = \frac{\sum(R_i-\bar{R})(Q_i-\bar{Q})}{\sqrt{\sum(R_i-\bar{R})^2\sum(Q_i-\bar{Q})^2}}$. As we can see, Spearman's coefficient is still dependent on the values chosen for the ranks representing the classes and so it is not completely appropriate to measure the performance of ordinal data classifiers. Kendal's coefficient is in our view better than the Spearman's one to compare ordinal variables, but it is still not completely adequate to evaluate the results of a classification algorithm. In fact, the variables C and $\hat{C}$ are two special ordinal variables because, as there are usually very few classes compared to the number of observations, these variables will take many tied values (most of them, in fact). As will be seen below, the coefficient introduced by us, r_{int}, takes this into account by defining a suitable order relation associated to each variable. Nevertheless, r_{int} is sufficiently general and, if there were no tied values, it could be applied as it is. It serves thus to compare any two ordinal variables, whether there are tied values or not.

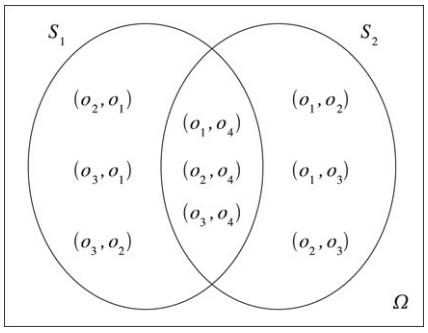

Fig. 2. Diagrammatic representation in Ω of v_1 and v_2 through S_1 and S_2, respectively.

Based on the work by Lerman (1992), we will now introduce r_{int}, a measure of performance that is not sensitive to the values that are chosen to represent the ordinal classes. Firstly, note that the only thing that matters is the order relation between such values, which is the same as the order relation between the classes. Let $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ represent the dataset where the performance is to be measured, *i.e.*, the test set, and take $\Omega = \mathcal{O} \times \mathcal{O} - \{(o_1, o_1), (o_2, o_2), \ldots, (o_n, o_n)\}$ to represent the set of pairs corresponding to different observations. Finally, define the order relation R_v as $o_i R_v o_j$ if and only if $v(o_i) \leq v(o_j)$, where v represents a qualitative ordinal variable on $\mathcal{O}$. Let v_1 and v_2 be two qualitative ordinal variables on $\mathcal{O}$, whose exact values are of no importance since the only information of interest to us is the order relation between those values. Under the previously exposed, each of the variables v_1 and v_2 can be fully represented by a subset of Ω. As a motivating example, suppose that $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$ and consider the following table:

$\mathcal{O}$	v_1	v_2
o_1	3	1
o_2	2	2
o_3	1	3
o_4	4	4

"According" to v_1, the subset of Ω (which in this case has cardinality twelve) whose elements verify the relation R_{v_1} is $S_1 = \{(o_1, o_4), (o_2, o_1), (o_2, o_4), (o_3, o_1), (o_3, o_2), (o_3, o_4)\}$. Similarly, v_2 is represented by $S_2 = \{(o_1, o_2), (o_1, o_3), (o_1, o_4), (o_2, o_3), (o_2, o_4), (o_3, o_4)\}$. The intersection $S_1 \cap S_2$, which in this case has three elements, is the key point for the comparison of the two variables v_1 and v_2 (see Fig. 2).

We define our measure of association between the two ordinal variables v_1 and v_2 to be

$$r_{\text{int}} = A + B\frac{\text{card}(S_1 \cap S_2)}{\sqrt{\text{card}(S_1)\text{card}(S_2)}},$$

where the denominator is considered to normalise the coefficient, and the constants A and B are such that r_{int} takes values in the range $[-1, 1]$: 1 when the two variables are identical ($S_1 = S_2$), and -1 when they are completely opposite ($S_1 \cap S_2 = \emptyset$). These two conditions imply that $A + B = 1$ and $A = -1$, respectively; thus, $A = -1$, $B = 2$ and therefore

$$r_{\text{int}} = -1 + 2\frac{\text{card}(S_1 \cap S_2)}{\sqrt{\text{card}(S_1)\text{card}(S_2)}}.$$

This coefficient allows to compare any two ordinal variables. Our purpose now is to apply it to the performance measurement of a classifier. As above, we will do that by comparing the two variables C and $\hat{C}$, corresponding to the true and predicted classes. As these two variables take values in the set $\{1, 2, \ldots, K\}$, there will be many observations with the same value in each variable. This fact led us to define a quicker way of computing r_{int}, which is based on the contingency table crossing C with $\hat{C}$:

	$\hat{C}$				Total
C	1	2	…	K	
1	n_{11}	n_{12}	…	n_{1K}	$n_{1\bullet}$
2	n_{21}	n_{22}	…	n_{2K}	$n_{2\bullet}$
⋮	⋮	⋮	⋮	⋮	⋮
K	n_{K1}	n_{K2}	…	n_{KK}	$n_{K\bullet}$
Total	$n_{\bullet 1}$	$n_{\bullet 2}$	…	$n_{\bullet K}$	n

In this table, n_{ij} represents the number of observations whose true class is $\mathcal{C}_i$ and whose predicted class is $\mathcal{C}_j$. The total number of observations whose true class is $\mathcal{C}_i$ is given by the sum of row i, $n_{i\bullet}$. The total number of observations whose predicted class is $\mathcal{C}_j$ is given by the sum of column j, $n_{\bullet j}$. Hence,

$$\text{card}(S_1) = \sum_{i=1}^{K} n_{i\bullet}\left(\sum_{j=i}^{K} n_{j\bullet} - 1\right) = \sum_{i=1}^{K}\sum_{j=1}^{K} n_{i\bullet}n_{j\bullet} - n,$$

$$\text{card}(S_2) = \sum_{i=1}^{K} n_{\bullet i}\left(\sum_{j=i}^{K} n_{\bullet j} - 1\right) = \sum_{i=1}^{K}\sum_{j=1}^{K} n_{\bullet i}n_{\bullet j} - n,$$

and

$$\begin{aligned}\text{card}(S_1 \cap S_2) &= \sum_{i=1}^{K}\sum_{j=1}^{K} n_{ij}\left(\sum_{i'=i}^{K}\sum_{j'=j}^{K} n_{i'j'} - 1\right)\\ &= \sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{i'=i}^{K}\sum_{j'=j}^{K} n_{ij}n_{i'j'} - n,\end{aligned}$$

which can now be considered in the definition of r_{int}.

5. Experimental study

5.1. Classifying artificial ordinal data

We start by conducting an empirical comparison in an artificial dataset between the various implementations of the unimodal paradigm and between these and other four classification methods. The first of the four methods is the conventional network which does not account for an order relation between classes and estimates directly the *a posteriori* probabilities without assuming a particular parametric probabilistic model at the output. The remaining three methods are specifically intended for the classification of ordinal data and they are the neural network (NN) based algorithms pNN by Frank and Hall (2001), which transform the problem with K classes into $K - 1$ binary problems, and iNN by Costa (1996), which proposes a NN with $K - 1$ outputs to solve the K-class ordinal problem, and the support vector machine (SVM) based algorithm pSVM also by Frank and Hall (2001), which maps into SVMs the strategy of the pNN algorithm. The comparison study is based on two performance measures: the Misclassification Error Rate (MER), which is the most commonly used, and r_{int}, the coefficient introduced in the previous section, which we think is the most appropriate one to assess the performance of classifiers applied to ordinal data.

As in Pinto da Costa and Cardoso (2005), we began by generating 1000 examples $\mathbf{x} = (x_1, x_2)^T$ in the unit square $[0, 1] \times [0, 1] \subset \mathbb{R}^2$ according to a uniform distribution. Then, we assigned to each example $\mathbf{x}$ a class corresponding to a number

$$y = \min\{r \in \{1, 2, 3, 4, 5\} : b_{r-1} < 10(x_1 - 0.5)(x_2 - 0.5) + \varepsilon < b_r\},$$

where $(b_0, b_1, b_2, b_3, b_4, b_5) = (-\infty, -1, -0.1, 0.25, 1, +\infty)$ determines the class boundaries and $\varepsilon \sim N(0, 0.125^2)$ simulates the possible existence of an error in the assignment of the true class to $\mathbf{x}$, $\mathcal{C}_{\mathbf{x}}$. Fig. 3(a) depicts the 16.5% of examples which are assigned to a different class after the addition of ε. The unbalanced distribution of the random variable class is shown in Fig. 3(b).

We randomly split the generated dataset into training, validation and test sets. In order to study the effect of varying the size of the training set, we considered three possibilities: 2%, 4% and 8% of all the data were used for training. The validation set had twice the size of the training set and the remaining data were used for testing. In each of the three possibilities, the splitting of the data into training, validation and test was repeated five times in order to obtain more stable results for MER and r_{int} by averaging and also to assess the variability of these measures. Each time, several pre-chosen models were trained. The models were neural networks with only one hidden layer and a number of hidden neurons ranging from 3 to 9, and support vector machines of the class ν-SVR (Scholkopf, Smola, Williamson, & Bartlett, 2000). We choose a SVM kernel given by a radial basis function, because this led us to best results, and a parameterization $\nu = 0.5$, $C = 2^i$, where i ranged from -10 to 11. The neural networks were not considered only in the pSVM method, whereas the support vector machines were only considered in the binomial regression case and in the pSVM method. The training of the networks was carried out under Matlab 7 R14 and was done using back-propagation together with the Levenberg–Marquardt algorithm in the parametric

Table 1
Mean (variation coefficient) of MER in the artificial dataset for the unimodal models

Misclassification Error Rate (MER)				
Model	Penalty term	(Training, Test) sets' percent size		
		(2%, 94%)	(4%, 88%)	(8%, 76%)
Binomial NN	–	0.40 (16%)	0.29 (8%)	0.27 (8%)
Truncated Poisson NN	–	0.43 (19%)	0.39 (20%)	0.33 (7%)
Nonparametric NN	$\alpha = 1$	0.55 (16%)	0.44 (16%)	0.35 (21%)
	$\alpha = 2$	0.60 (11%)	0.41 (28%)	0.50 (31%)
	$\alpha = 3$	0.55 (10%)	0.51 (13%)	0.45 (12%)
	Number of modes	0.57 (11%)	0.50 (23%)	0.41 (14%)
Binomial regression NN	–	0.42 (15%)	0.29 (30%)	0.23 (11%)
Binomial regression SVM	–	0.30 (21%)	0.24 (8%)	0.22 (9%)

Table 2
Mean (variation coefficient) of MER in the artificial dataset for four methods competing with the unimodal models

Misclassification Error Rate (MER)			
Model	(Training, Test) sets' percent size		
	(2%, 94%)	(4%, 88%)	(8%, 76%)
Conventional NN	0.55 (12%)	0.46 (12%)	0.41 (15%)
pNN	0.46 (17%)	0.38 (8%)	0.32 (17%)
iNN	0.51 (10%)	0.39 (4%)	0.37 (21%)
pSVM	0.43 (16%)	0.29 (9%)	0.23 (12%)

approach and with the Conjugate Gradient Method with the Polak–Ribiére formula in the nonparametric approach and for the conventional network. As for support vector machines, we used the software implementation provided by LIBSVM 2.8. After the five times training, the best model was chosen according to best mean performance in the validation set and the results from its performance assessment in the five setups of test set were used to obtain the mean and variation coefficient of MER and r_{int}.

Table 1 shows the test results regarding MER for the various implementations of the unimodal paradigm. It can be seen that the parametric models (rows 1, 2, 7, 8) exhibit a performance superior to the nonparametric models (rows 3–6) in two senses: not only their mean error is lower but also their generalization ability tends to be less variant with an increase in the number of training examples. Comparing the four models of the parametric approach, it can be seen that the best results are obtained with the binomial distribution. Moreover, it seems that there are no significant differences between using binomial networks in the context of classification and regression. On the other hand, it seems to be more advantageous to use SVMs rather than NNs in the regression approach, particularly when there are few training examples. Hence, the best implementation of the unimodal model for the current classification problem is the binomial regression SVM with an error of 0.22, a value close to the Bayes error of 0.165.

Table 2 shows the test results concerning MER for the four methods competing with the unimodal paradigm. It is possible to conclude that the performance of the conventional NN is worse than the performance of the other three methods, which account for the order relation between the classes contrary to the conventional NN. Moreover, the method exhibiting the best performance is pSVM. Finally, comparing Tables 1 and 2, it follows that our unimodal paradigm has several implementation strategies that outperform pSVM, thus all other three competing methods. For illustration purposes, Fig. 4 depicts the true class boundaries (black lines) and predicted class regions (from $\hat{\mathcal{C}}_1$ (white area) to $\hat{\mathcal{C}}_5$ (darker grey area)) for the five setups (columns) of the training set with 8% of all the data when the conventional (top row), binomial (middle row) and truncated Poisson (bottom row) networks are used. As expected from the results in the two tables previously analysed, it can be seen that the predicted class regions are much closer to the true class regions when they are predicted by both the binomial and the truncated Poisson networks.

The previous section introduced r_{int}, a measure of association between two ordinal variables, such as the variables' true and predicted classes in an ordinal data classification problem. Moreover, it was pointed out why r_{int} is better than MER and other usually used measures for performance assessment whenever there is an order relation between the classes. Hence, we now take r_{int} to carry out an empirical comparison between the generalization ability of the various approaches proposed under the context of the unimodal paradigm and between these and the other four competing methods. This comparison is made by considering Tables 3 and 4. It is important to start by stressing that the variation coefficient of r_{int} is always lower than the one of MER (see Tables 1 and 2); thus, we can look with more confidence at the mean value of r_{int}. In addition, although not shown, r_{int} tends to be more conservative and less variant than r_S, whose value is affected by the numbers $1, \ldots, 5$ used to order the classes. Looking at the tables under analysis, it is clear that our models, namely the parametric ones, not only learn better (higher values for r_{int}) but they also learn faster (with fewer training observations); in truth, they also learn faster in the sense that both the computational time and burden needed to estimate their weights are lower, as they have, in general, a less complex architecture with fewer weights to estimate. In the end, we are led to the same overall conclusion as before: our unimodal paradigm has several implementation strategies that outperform the other competing methods, most of which are also designed to solve ordinal data classification problems.

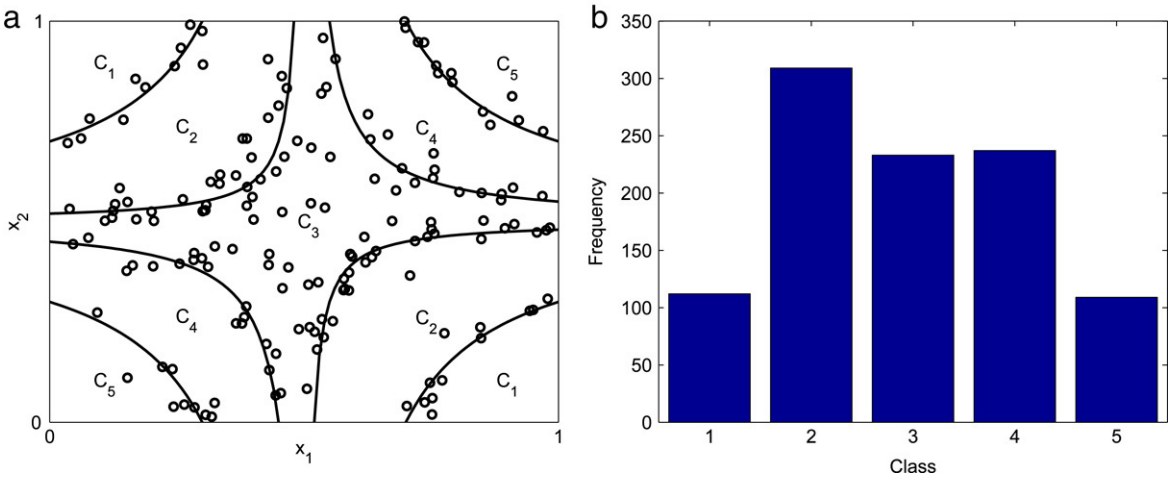

Fig. 3. Artificial dataset. (a) Scatter plot of the 16.5% of examples wrongly classified in the artificial dataset. Also shown are the class boundaries. (b) Class distribution in the artificial dataset.

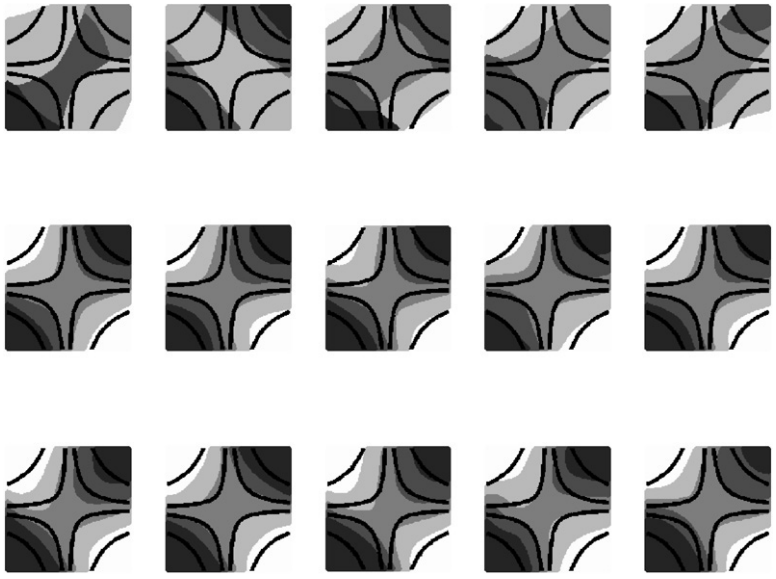

Fig. 4. True class boundaries (black lines) and predicted class regions (from $\hat{\mathcal{C}}_1$ (white area) to $\hat{\mathcal{C}}_5$ (darker grey area)) for the five setups (columns) of the training set with 8% of all the data in the artificial dataset when the conventional (top row), binomial (middle row) and truncated Poisson (bottom row) networks are used.

Table 3
Mean (variation coefficient) of r_{int} in the artificial dataset for the unimodal models

r_{int}				
Model	Penalty term	(Training, Test) sets' percent size		
		(2%, 94%)	(4%, 88%)	(8%, 76%)
Binomial NN	–	0.78 (6%)	0.83 (1%)	0.84 (1%)
Truncated Poisson NN	–	0.77 (7%)	0.81 (2%)	0.82 (1%)
Nonparametric NN	$\alpha = 1$	0.67 (6%)	0.74 (8%)	0.78 (6%)
	$\alpha = 2$	0.65 (9%)	0.76 (7%)	0.70 (11%)
	$\alpha = 3$	0.67 (8%)	0.70 (8%)	0.74 (4%)
	Number of modes	0.63 (9%)	0.69 (12%)	0.73 (9%)
Binomial regression NN	–	0.73 (10%)	0.82 (5%)	0.85 (2%)
Binomial regression SVM	–	0.81 (5%)	0.84 (2%)	0.85 (2%)

5.2. Classifying real ordinal data

In this subsection, we carry out a comparative empirical study between the approaches considered in the context of the unimodal model and the four competing methods described in the beginning of the last subsection. This time, we propose to solve real ordinal data classification problems of employee selection and prediction of pasture production

Table 4
Mean (variation coefficient) of r_{int} in the artificial dataset for four methods competing with the unimodal models

r_{int}			
Model	(Training, Test) sets' percent size		
	(2%, 94%)	(4%, 88%)	(8%, 76%)
Conventional NN	0.65 (11%)	0.71 (8%)	0.74 (5%)
pNN	0.71 (13%)	0.75 (4%)	0.80 (3%)
iNN	0.67 (10%)	0.75 (1%)	0.75 (6%)
pSVM	0.72 (12%)	0.82 (2%)	0.85 (2%)

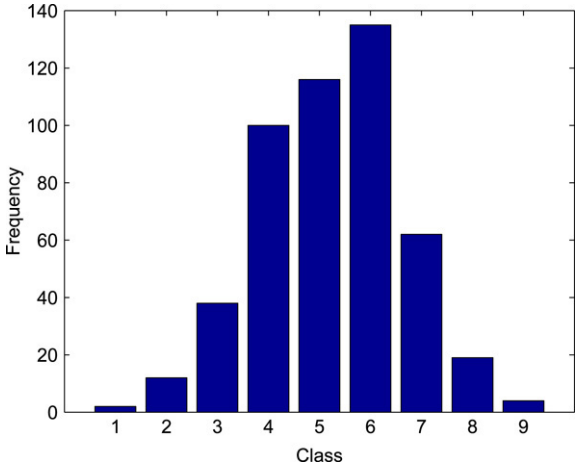

Fig. 5. Class distribution in the ESL dataset.

(the corresponding datasets are available at the WEKA website http://www.cs.waikato.ac.nz/~ml/index.html).

5.2.1. Employee selection: The ESL dataset

The ESL dataset contains 488 profiles of applicants for certain industrial jobs. Expert psychologists of a recruiting company determined the values of the input attributes (4 attributes, with integer values from 0 to 9) based upon psychometric test results and interviews with the candidates. The output is an overall score $(1, \ldots, 9)$, corresponding to the degree of fitness of the candidate to this type of jobs, which is nonevenly distributed as depicted in Fig. 5.

We randomly split the dataset into training, validation and test sets. In order to study the effect of varying the size of the training set, we considered three possibilities: 5%, 10% and 20% of all the data were used for training. The validation set had the same size of the training set and the remaining data were used for testing. The training, selection and testing of the models followed the same methodology presented in the problem of classifying artificial data. There is only one difference concerning support vector machines; the linear kernel was chosen because its results were the best.

The test results are shown in Tables 5 and 6 for the unimodal paradigm and in Tables 7 and 8 for the four competing methods. The performance measure MER suggests that nonparametric neural networks are slightly better than the conventional neural network and that parametric models are the best of all. As before, the differences between the performances of parametric models lie in the assumed distribution for the *a posteriori* probabilities (binomial over truncated Poisson), in the learning paradigm (support vector machines over neural networks), but not so much in the perspective (classification, regression). However, these latter conclusions are not so clearly drawn when r_{int} is the performance measure considered. In fact, previously pointed out differences, like the assumed distribution for the *a posteriori* probabilities, are not so clear now, particularly for higher training set sizes. Hence, it is only suggested that parametric models are better than nonparametric ones, which in turn are slightly better than the conventional neural network. Overall, the MER statistic suggests that some of our models are clearly superior to all the others, including the three competing methods that take into account the order relation between the classes. On the other hand, although r_{int} shows a similar suggestion, it is not so marked.

5.2.2. Pasture production: The pasture dataset

The objective related to the pasture dataset is to predict pasture production from a variety of biophysical factors. Vegetation and soil variables from areas of grazed North Island Hill Country with different management (fertiliser application/stocking rate) histories (1973–1994) were measured and subdivided into 36 paddocks. Nineteen vegetation variables (including herbage production), soil chemical, physical and biological, and water variables were selected as potentially useful biophysical indicators, totalling 22 attributes.The target feature, the pasture production, has been categorised in three classes (Low, Medium, High), evenly distributed in this dataset of 36 instances. Before training, the data was scaled to fall always within the range $[0, 1]$ given that the values of the attributes vary by orders of magnitude. The fertiliser attribute was represented using 4 variables: LL $=$ $(1, 0, 0, 0)$, LN $=$ $(0, 1, 0, 0)$, HL $=$ $(0, 0, 1, 0)$ and HH $=$ $(0, 0, 0, 1)$.

We randomly split the dataset into training, validation and test sets. In order to study the effect of varying the size of the training set, we considered three possibilities: 8%, 16% and 32% of all the data were used for training. The validation set had the same size of the training set and the remaining data were used for testing. The training, selection and testing of the models followed the same methodology presented in the problem of classifying artificial data; this time the kernel giving the best results in support vector machines was a polynomial of degree 3.

The test results are shown in Tables 9 and 10 for the unimodal paradigm and in Tables 11 and 12 for the four competing methods. We start by stressing that the variation coefficient of MER is high for all models and training set sizes; therefore, the mean value of MER is less informative and we cannot properly judge and compare the model's performance using it. If in turn we consider r_{int}, we are led to the suggestion that there are many cases where nonparametric models are better than parametric ones, but only the latter exhibit a consistent increasing in performance with an increasing training set size. In some situations, the competing methods are comparable to our best models, although there is

Table 5
Mean (variation coefficient) of MER in the ESL dataset for the unimodal models

Misclassification Error Rate (MER)				
Model	Penalty term	(Training, Test) sets' percent size		
		(5%, 90%)	(10%, 80%)	(20%, 60%)
Binomial NN	–	0.43 (25%)	0.43 (13%)	0.33 (7%)
Truncated Poisson NN	–	0.54 (15%)	0.47 (18%)	0.46 (19%)
Nonparametric NN	$\alpha = 1$	0.64 (25%)	0.51 (15%)	0.51 (12%)
	$\alpha = 2$	0.65 (13%)	0.52 (22%)	0.48 (14%)
	$\alpha = 3$	0.59 (13%)	0.57 (16%)	0.49 (17%)
	Number of modes	0.72 (10%)	0.52 (11%)	0.53 (18%)
Binomial regression NN	–	0.42 (16%)	0.36 (17%)	0.31 (8%)
Binomial regression SVM	–	0.35 (14%)	0.32 (5%)	0.30 (4%)

Table 6
Mean (variation coefficient) of r_{int} in the ESL dataset for the unimodal models

r_{int}				
Model	Penalty term	(Training, Test) sets' percent size		
		(5%, 90%)	(10%, 80%)	(20%, 60%)
Binomial NN	–	0.78 (8%)	0.76 (5%)	0.82 (3%)
Truncated Poisson NN	–	0.71 (13%)	0.77 (7%)	0.79 (2%)
Nonparametric NN	$\alpha = 1$	0.61 (23%)	0.70 (11%)	0.73 (4%)
	$\alpha = 2$	0.64 (13%)	0.70 (16%)	0.75 (4%)
	$\alpha = 3$	0.66 (11%)	0.66 (13%)	0.75 (6%)
	Number of modes	0.56 (5%)	0.72 (6%)	0.71 (12%)
Binomial regression NN	–	0.77 (7%)	0.81 (3%)	0.83 (1%)
Binomial regression SVM	–	0.82 (3%)	0.83 (1%)	0.84 (1%)

Table 7
Mean (variation coefficient) of MER in the ESL dataset for four methods competing with the unimodal models

Misclassification Error Rate (MER)			
Model	(Training, Test) sets' percent size		
	(5%, 90%)	(10%, 80%)	(20%, 60%)
Conventional NN	0.65 (16%)	0.57 (13%)	0.50 (18%)
pNN	0.55 (11%)	0.42 (8%)	0.35 (6%)
iNN	0.45 (18%)	0.39 (6%)	0.36 (7%)
pSVM	0.49 (9%)	0.39 (12%)	0.35 (10%)

Table 8
Mean (variation coefficient) of r_{int} in the ESL dataset for four methods competing with the unimodal models

r_{int}			
Model	(Training, Test) sets' percent size		
	(5%, 90%)	(10%, 80%)	(20%, 60%)
Conventional NN	0.63 (12%)	0.65 (17%)	0.73 (8%)
pNN	0.68 (11%)	0.78 (3%)	0.83 (1%)
iNN	0.76 (7%)	0.79 (1%)	0.81 (1%)
pSVM	0.76 (3%)	0.80 (2%)	0.82 (1%)

always one of our models which outperforms all models under analysis. As a final note, r_{int} suggests that this is the dataset where all considered models have more difficulty in solving the proposed problem.

5.3. General comments and conclusions

The results of our experimental study suggest that the unimodal model, specially designed to classify ordinal data, has a superior performance when compared both in terms of error measures like MER and r_{int} and also computing time. This was observed in an artificial and two real datasets and for several methods. In particular, the parametric approach was in general better than the nonparametric one; especially, when the binomial model is considered. This motivated us to find a theoretical explanation for its success, which we expose in the next section through a new concept that we introduce; that of the flexibility of a discrete distribution.

6. Flexibility of a discrete distribution

In the previous section, it was suggested that the binomial network exhibits a better performance; in particular than does the truncated Poisson. Given two classifiers, we can of course never say that one is always better than the other. As it is usually known, the performance of a classifier depends on many factors and sometimes simpler (or more rigid) classifiers such as linear discriminant functions outperform more elaborated ones. Nevertheless, we will show here that the binomial network is more flexible (less rigid) than the truncated Poisson network. This greater flexibility explains, in our view, the experimental findings observed above.

Table 9
Mean (variation coefficient) of MER in the pasture dataset for the unimodal models

Misclassification Error Rate (MER)				
Model	Penalty term	(Training, Test) sets' percent size		
		(8%, 84%)	(16%, 68%)	(32%, 36%)
Binomial NN	–	0.43 (31%)	0.39 (22%)	0.27 (26%)
Truncated Poisson NN	–	0.47 (25%)	0.37 (5%)	0.23 (39%)
Nonparametric NN	$\alpha = 1$	0.43 (37%)	0.37 (32%)	0.23 (39%)
	$\alpha = 2$	0.41 (39%)	0.34 (30%)	0.38 (33%)
	$\alpha = 3$	0.43 (41%)	0.33 (41%)	0.38 (33%)
	Number of modes	0.45 (42%)	0.35 (27%)	0.33 (31%)
Binomial regression NN	–	0.47 (29%)	0.38 (26%)	0.37 (55%)
Binomial regression SVM	–	0.43 (22%)	0.41 (22%)	0.37 (38%)

We will start by defining a new concept, which we call the flexibility of a discrete distribution. This concept has to do with the "dependence" between consecutive values of the distribution. We think that a distribution whose consecutive values are more "independent" might adapt better to certain problems and can therefore give better results. For instance, a distribution for which $P(X = x + 1) = cP(X = x)$, where c is a constant, is very "rigid", *i.e.*, has no flexibility at all. This distribution is monotonically increasing or decreasing (depending on the value of c), and so its mode is always attained in the extreme classes. The quotient between consecutive values in this distribution is always the same, the constant c, showing no variability. The more variable this quotient is, the more flexible the distribution will be. This was the reasoning which led us to the following definition:

Definition 1. Given a discrete distribution of probability function $f(x) = P(X = x) > 0$, we define its *flexibility* to be $\text{flexb}(f(X)) = \text{Var}\left(\frac{f(X)}{f(X-1)}\right)$.

In accordance with the definition, a distribution for which $P(X = x + 1) = cP(X = x)$ has zero flexibility. We will now find the flexibility of the binomial and right truncated Poisson distributions.

6.1. Flexibility of the binomial distribution

In the case of a binomial $B(M, p)$, we have

$$f(x) = P(X = x) = \frac{M!}{x!(M-x)!}p^x(1-p)^{M-x} \quad \forall x \in \{0, 1, \ldots, M\}.$$

Hence,

$$\frac{f(X)}{f(X-1)} = \frac{(M+1)p}{(1-p)}\frac{1}{X} - \frac{p}{(1-p)},$$

and so

$$\text{flexb}(f(X)) = \text{Var}\left(\frac{f(X)}{f(X-1)}\right) = \left(\frac{(M+1)p}{1-p}\right)^2 \text{Var}\left(\frac{1}{X}\right).$$

In order to find $\text{Var}\left(\frac{1}{X}\right)$ we will use the delta method (Rice, 1994). Briefly, these methods give us a way of approximating the variance of a function of a random variable: if we want to approximate the variance of $G(X)$, where X is a random variable, we start by using the approximation

$$G(X) \simeq G(\mu) + (X - \mu)G'(\mu),$$

where μ is the mean of X, so that we can get

$$\text{Var}(G(X)) \simeq \text{Var}(X)[G'(\mu)]^2.$$

In our case, $G(X) = \frac{1}{X}$ and $\mu = Mp$. Therefore,

$$\text{Var}(G(X)) \simeq \text{Var}(X)[G'(\mu)]^2 = Mp(1-p)\left(\frac{1}{(Mp)^2}\right)^2 = \frac{(1-p)}{M^3p^3},$$

and so

$$\text{flexb}(f(X)) \simeq \left(\frac{(M+1)p}{1-p}\right)^2 \frac{(1-p)}{M^3p^3} = \frac{(M+1)^2}{M^3}\frac{1}{p(1-p)}.$$

6.2. Flexibility of the right truncated Poisson distribution

In the case of the right truncated Poisson distribution $P_t(\lambda)$, taking the integer values between 0 and M, we have

$$f(x) = P(X_t = x) = c.\frac{e^{-\lambda}\lambda^x}{x!} \quad \forall x \in \{0, 1, \ldots, M\},$$

where c is the constant such that the sum of the $M + 1$ probabilities equals 1. Hence,

$$\frac{f(X_t)}{f(X_t - 1)} = \frac{\lambda}{X_t},$$

and so

$$\text{flexb}(f(X_t)) = \text{Var}\left(\frac{f(X_t)}{f(X_t - 1)}\right) = \lambda^2\text{Var}\left(\frac{1}{X_t}\right).$$

Table 10
Mean (variation coefficient) of r_{int} in the pasture dataset for the unimodal models

r_{int}				
Model	Penalty term	(Training, Test) sets' percent size		
		(8%, 84%)	(16%, 68%)	(32%, 36%)
Binomial NN	–	0.65 (14%)	0.68 (11%)	0.77 (10%)
Truncated Poisson NN	–	0.65 (14%)	0.73 (9%)	0.79 (10%)
Nonparametric NN	$\alpha = 1$	0.67 (10%)	0.72 (10%)	0.76 (11%)
	$\alpha = 2$	0.68 (10%)	0.74 (6%)	0.69 (6%)
	$\alpha = 3$	0.71 (10%)	0.76 (6%)	0.67 (7%)
	Number of modes	0.66 (7%)	0.72 (8%)	0.68 (8%)
Binomial regression NN	–	0.62 (12%)	0.69 (8%)	0.71 (18%)
Binomial regression SVM	–	0.68 (8%)	0.70 (9%)	0.73 (6%)

Table 11
Mean (variation coefficient) of MER in the pasture dataset for four methods competing with the unimodal models

Misclassification Error Rate (MER)			
Model	(Training, Test) sets' percent size		
	(8%, 84%)	(16%, 68%)	(32%, 36%)
Conventional NN	0.47 (29%)	0.40 (28%)	0.43 (44%)
pNN	0.62 (20%)	0.33 (15%)	0.35 (31%)
iNN	0.56 (13%)	0.41 (29%)	0.33 (18%)
pSVM	0.43 (35%)	0.39 (29%)	0.40 (27%)

Table 12
Mean (variation coefficient) of r_{int} in the pasture dataset for four methods competing with the unimodal models

r_{int}			
Model	(Training, Test) sets' percent size		
	(8%, 84%)	(16%, 68%)	(32%, 36%)
Conventional NN	0.56 (26%)	0.69 (15%)	0.63 (14%)
pNN	0.50 (21%)	0.73 (7%)	0.73 (7%)
iNN	0.54 (11%)	0.69 (9%)	0.70 (3%)
pSVM	0.65 (11%)	0.70 (8%)	0.67 (4%)

Now, the variable $\frac{1}{X_t}$, when X_t is a right truncated Poisson, has certainly smaller variance than its corresponding regular Poisson X. Thus, by applying the delta method to the regular Poisson, we conclude that $\text{Var}\left(\frac{1}{X_t}\right) < \text{Var}\left(\frac{1}{X}\right) \simeq \text{Var}(X)[G'(\mu)]^2$, where μ and $\text{Var}(X)$ are λ, the mean and variance of the regular Poisson distribution, and $G(X) = \frac{1}{X}$. We conclude therefore that, for the truncated Poisson distribution,

$$\text{flexb}(f(X_t)) = \text{Var}\left(\frac{\lambda}{X_t}\right) < \frac{1}{\lambda}.$$

In order to compare the flexibilities of the binomial and truncated Poisson distributions, we oblige their modes to be equal, *i.e.*, $\lfloor (M+1)p \rfloor = \lfloor \lambda \rfloor$, as this is the case in our problem. Thus, taking $p \simeq \frac{\lambda}{(M+1)}$, the flexibility of the binomial becomes

$$\frac{(M+1)^2}{M^3}\frac{1}{p(1-p)} \simeq \frac{(M+1)^2}{M^3}\frac{(M+1)^2}{\lambda(M+1-\lambda)} = \frac{(M+1)^3}{M^3}\frac{M+1}{(M+1-\lambda)}\frac{1}{\lambda}.$$

It is clear that this last expression is greater than the upper bound $\frac{1}{\lambda}$ for the flexibility of the right truncated Poisson distribution.

We think this notion of flexibility of a distribution explains the better results obtained by us with the binomial model. Nevertheless, as said above, more flexible classifiers are not always better, and so we suggest the use of both the binomial and the truncated Poisson networks in practice. Actually, as said before, other discrete distributions, more flexible, for instance with more than one parameter to estimate, might be needed for more difficult problems. We plan to consider other parametric networks in our future work.

7. Conclusions and final remarks

In this paper, we presented a novel model for the supervised classification of ordinal data. Our approach assumes that the variable class associated with a given query should follow a unimodal distribution. We pursued this goal using different ways of imposing unimodality, either parametrically or nonparametrically, and applying different learning methodologies, namely neural networks, as well as support vector machines. The parametric models have generally shown better performance, particularly the binomial one. In the future, we plan to use other discrete distributions and other learning methodologies in order to explore new approaches of the unimodal paradigm. The concept of flexibility of a discrete distribution was introduced in order to explain the superiority of some parametric models over others. We have also showed that there is always a unimodal model which outperforms other known methods for ordinal classification.

Some of the proposed models, namely the parametric ones, learn faster in two senses: first, the computational burden time required to train the learning models is lower than that expended during the training of conventional methods; second, they usually require less training patterns in order to reach higher generalization abilities.

A new coefficient to measure the performance of ordinal data classifiers has been introduced and its advantages over other usual measures, such as the misclassification error rate,

have been given. In particular, we found that in practical experiments the mean value for this new coefficient is much more informative since it has an associated variation coefficient which is relatively low regardless of the cardinality of the training, validation and test sets.

Acknowledgments

The authors would like to thank the anonymous reviewers for the comments and suggestions which helped in improving the presentation of the paper.

The second author would like to thank the Fundação para a Ciência e a Tecnologia under the Third Community Support Framework for the financial support during the course of this project. The second author has also been partially supported by Unidade de Investigação Matemática e Aplicações, Universidade de Aveiro, Portugal, through Programa Operacional "Ciência e Tecnologia e Inovação" of the FCT, co-financed by the European Union fund FEDER.

References

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining*, *2*, 121–167.

Chu, W., & Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, *6*, 1019–1041.

Chu, W., & Keerthi, S.S. (2005). New approaches to support vector ordinal regression. In *Proceedings of the 22nd international conference on machine learning* (pp. 145–152).

Costa, M. (1996). Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities. *International Journal of Neural Systems*, *7*(5), 627–638.

dos Santos Cardoso, J., Pinto da Costa, J., & Cardoso, M. J. (2005). Modelling ordinal relations with SVMs: An application to objective aesthetic evaluation of breast cancer conservative treatment. *Neural Networks*, *18*(5–6), 808–817.

Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. In *Proceedings of the 12th European conference on machine learning*: Vol. 1 (pp. 145–156).

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction* (1st ed.). New York: Springer-Verlag.

Haykin, S. (1999). *Neural networks, a comprehensive foundation* (2nd ed.). New Jersey: Prentice-Hall.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector learning for ordinal regression. In *Proceedings of the 9th international conference on artificial neural networks*: Vol. 1 (pp. 97–102).

Lerman, I. -C. (1992). Conception et analyse de la forme limite d'une famille de coefficients statistiques d'association entre variables relationnelles. II. *Mathématiques, Informatique et Sciences Humaines*, *29e année*(119), 75–100.

Li, L., & Lin, H. -T. (2007). Ordinal regression by extended binary classification. In B. Scholkopf, J. C. Platt, & T. Hofmann (Eds.), *Advances in neural information processing systems*: *Vol. 19*. Cambridge, MA: MIT Press.

Mathieson, M. J. (1995). Ordinal models for neural networks. In A. Refenes, Y. Abu-Mostafa, & J. Moody (Eds.), *Neural networks in financial engineering*. Singapore: World Scientific.

Pinto da Costa, J., & Cardoso, J. S. (2005). Classification of ordinal data using neural networks. In *LNAI*: *Vol. 3720*. *Proceedings of the 16th European conference on machine learning* (pp. 690–697). Springer-Verlag.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society B*, *42*, 109–142.

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (1992). *Numerical recipes in C: The art of scientific computing*. Cambridge University Press.

Rice, J. (1994). *Mathematical statistics and data analysis* (2nd ed.). Duxbury.

Scholkopf, B., Smola, A., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation*, *12*, 1207–1245.

Shen, L., & Joshi, A. K. (2005). Ranking and reranking with perceptron. *Machine Learning*, *60*, 73–96.

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, *15*, 72–101.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley.