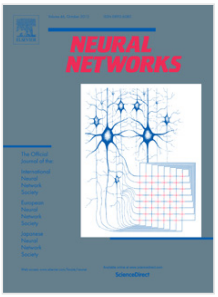
# Ordinal regression neural networks based on concentric hyperspheres

Gutiérrez, Pedro Antonio; Tino, Peter; Hervás-martínez, César

*DOI:*
10.1016/j.neunet.2014.07.001

# Accepted Manuscript




Ordinal regression neural networks based on concentric hyperspheres

Pedro Antonio Gutiérrez, Peter Tiňo, César Hervás-Martínez

PII: S0893-6080(14)00158-0
DOI: http://dx.doi.org/10.1016/j.neunet.2014.07.001
Reference: NN 3362

To appear in: *Neural Networks*

Received date: 7 February 2014
Revised date: 15 May 2014
Accepted date: 7 July 2014

Please cite this article as: Gutiérrez, P. A., Tiňo, P., & Hervás-Martínez, C. Ordinal regression neural networks based on concentric hyperspheres. *Neural Networks* (2014), http://dx.doi.org/10.1016/j.neunet.2014.07.001

# Ordinal regression neural networks based on concentric hyperspheres

Pedro Antonio Gutiérrez[*,a], Peter Tiňo[b], César Hervás-Martínez[a]

[a] *University of Córdoba, Dept. of Computer Science and Numerical Analysis*
*Rabanales Campus, Albert Einstein building, 14071 - Córdoba, Spain*
[b] *School of Computer Science, The University of Birmingham, Birmingham B15 2TT, United Kingdom*

**Abstract**

Threshold models are one of the most common approaches for ordinal regression, based on projecting patterns to the real line and dividing this real line in consecutive intervals, one interval for each class. However, finding such one-dimensional projection can be too harsh an imposition for some datasets. This paper proposes a multidimensional latent space representation with the purpose of relaxing this projection, where the different classes are arranged based on concentric hyperspheres, each class containing the previous classes in the ordinal scale. The proposal is implemented through a neural network model, each dimension being a linear combination of a common set of basis functions. The model is compared to a nominal neural network, a neural network based on the proportional odds model and to other state-of-the-art ordinal regression methods for a total of 12 datasets. The proposed latent space shows an improvement on the two performance metrics considered, and the model based on the three-dimensional latent space obtains competitive performance when compared to the other methods.

*Key words:* Ordinal regression; ordinal classification; neural networks; latent variable

## 1. Introduction

When we face an ordinal regression (OR) problem, the objective is to predict the label $y_i$ of an input vector $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^k$ and $y_i \in \mathcal{Y} \in \{C_1, C_2, \ldots, C_Q\}$. This is done by estimating a classification rule or function $L : \mathcal{X} \to \mathcal{Y}$ to predict the labels of new samples. In a supervised setting, we are given a training set of $N$ points, $D = \{(\mathbf{x}_i, y_i), 1 \leq i \leq N\}$. All these considerations can be also found in standard nominal classification, but, for OR, a natural label ordering is included, which is given by $C_1 \prec C_2 \prec \ldots \prec C_Q$. The symbol $\prec$ is an order relation representing the nature of the classification problem and expressing that a label is before another in the ordinal scale.

OR problems are very common in real settings, although the machine learning community has often treated them from a standard (nominal) perspective, ignoring the order relationship, $\prec$, between classes. Some examples of application fields where OR is found are credit rating [1], econometric modelling [2], medical research [3] or face recognition [4], to name a few. Considering the order relationship between classes can result in two significant benefits: 1) minimisation of specific classification errors, and 2) incorporation of the ordering into the classifier. With respect to the first benefit, it is clear that one should focus on predicting categories as close as possible to the real one when tackling an OR problem. Hence, OR methods are aimed to minimise those errors that involve large category gaps in the ordinal scale. As an example, consider a tumour classification problem where the categories are {*benign, dangerous, malign*}. Misclassification of *malign* tumours as *dangerous* is preferred to assign the label *benign* to a *malign* tumour and OR methods will generally minimise this second type of errors. The second benefit comes from the fact that label order is usually present, in a direct way in the input space or through a latent space representation [5]. Imbuing a classifier with this ordering will generally improve generalisation performance, as the classifier is better representing the nature of the task.

The field of OR has experienced significant development in the last decade, with many new methods adapted from traditional machine learning methodologies, from support vector machine (SVM) formulations [6] to Gaussian processes [7] or discriminant learning [8]. For all these methods, although classifier construction is motivated and undertaken from different points of view, the final models share a common structure or nature. They exploit the fact that it is natural to assume that an unobserved continuous variable underlies the ordinal response variable (e.g. the actual age of the person appearing in a picture for an age classification problem). This variable is called latent variable, and methods based on that assumption are known as threshold models [9]. Indeed, this structure can be found in one of the first models for OR, the proportional odds model (POM) [10], which is a probabilistic model estimating the cumulative probabilities of the different ordered categories and leading to linear decision boundaries. Threshold models methodologies estimate:

- A function $f(\mathbf{x})$ that tries to predict the values of the latent variable.
- A set of thresholds $\mathbf{b} = (b_1, b_2, \ldots, b_{Q-1}) \in \mathbb{R}^{Q-1}$ to represent intervals in the range of $f(\mathbf{x})$, which must satisfy the constraints $b_1 \leq b_2 \leq \ldots \leq b_{Q-1}$.

[*]Corresponding author: Pedro Antonio Gutiérrez
*Email address:* pagutierrez@uco.es (Pedro Antonio Gutiérrez)

# Ordinal regression neural networks based on concentric hyperspheres

**Abstract**

Threshold models are one of the most common approaches for ordinal regression, based on projecting patterns to the real line and dividing this real line in consecutive intervals, one interval for each class. However, finding such one-dimensional projection can be too harsh an imposition for some datasets. This paper proposes a multidimensional latent space representation with the purpose of relaxing this projection, where the different classes are arranged based on concentric hyperspheres, each class containing the previous classes in the ordinal scale. The proposal is implemented through a neural network model, each dimension being a linear combination of a common set of basis functions. The model is compared to a nominal neural network, a neural network based on the proportional odds model and to other state-of-the-art ordinal regression methods for a total of 12 datasets. The proposed latent space shows an improvement on the two performance metrics considered, and the model based on the three-dimensional latent space obtains competitive performance when compared to the other methods.

*Key words:* Ordinal regression; ordinal classification; neural networks; latent variable

## 1. Introduction

When we face an ordinal regression (OR) problem, the objective is to predict the label $y_i$ of an input vector $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^k$ and $y_i \in \mathcal{Y} \in \{C_1, C_2, \ldots, C_Q\}$. This is done by estimating a classification rule or function $L : \mathcal{X} \to \mathcal{Y}$ to predict the labels of new samples. In a supervised setting, we are given a training set of $N$ points, $D = \{(\mathbf{x}_i, y_i), 1 \leq i \leq N\}$. All these considerations can be also found in standard nominal classification, but, for OR, a natural label ordering is included, which is given by $C_1 \prec C_2 \prec \ldots \prec C_Q$. The symbol $\prec$ is an order relation representing the nature of the classification problem and expressing that a label is before another in the ordinal scale.

OR problems are very common in real settings, although the machine learning community has often treated them from a standard (nominal) perspective, ignoring the order relationship, $\prec$, between classes. Some examples of application fields where OR is found are credit rating [1], econometric modelling [2], medical research [3] or face recognition [4], to name a few. Considering the order relationship between classes can result in two significant benefits: 1) minimisation of specific classification errors, and 2) incorporation of the ordering into the classifier. With respect to the first benefit, it is clear that one should focus on predicting categories as close as possible to the real one when tackling an OR problem. Hence, OR methods are aimed to minimise those errors that involve large category gaps in the ordinal scale. As an example, consider a tumour classification problem where the categories are {*benign, dangerous, malign*}. Misclassification of *malign* tumours as *dangerous* is preferred to assign the label *benign* to a *malign* tumour and OR methods will generally minimise this second type of errors. The second benefit comes from the fact that label order is usually present, in a direct way in the input space or through a latent space representation [5]. Imbuing a classifier with this ordering will generally improve generalisation performance, as the classifier is better representing the nature of the task.

The field of OR has experienced significant development in the last decade, with many new methods adapted from traditional machine learning methodologies, from support vector machine (SVM) formulations [6] to Gaussian processes [7] or discriminant learning [8]. For all these methods, although classifier construction is motivated and undertaken from different points of view, the final models share a common structure or nature. They exploit the fact that it is natural to assume that an unobserved continuous variable underlies the ordinal response variable (e.g. the actual age of the person appearing in a picture for an age classification problem). This variable is called latent variable, and methods based on that assumption are known as threshold models [9]. Indeed, this structure can be found in one of the first models for OR, the proportional odds model (POM) [10], which is a probabilistic model estimating the cumulative probabilities of the different ordered categories and leading to linear decision boundaries. Threshold models methodologies estimate:

- A function $f(\mathbf{x})$ that tries to predict the values of the latent variable.
- A set of thresholds $\mathbf{b} = (b_1, b_2, \ldots, b_{Q-1}) \in \mathbb{R}^{Q-1}$ to represent intervals in the range of $f(\mathbf{x})$, which must satisfy the constraints $b_1 \leq b_2 \leq \ldots \leq b_{Q-1}$.

From a practical perspective, threshold models are basically trying to find a one-dimensional projection ($f(\mathbf{x})$) where patterns are ordered according to the class labels. Finding this projection can be a problem for real world datasets. If we consider linear models for $f(\mathbf{x})$, the chances that the patterns exhibit a linear ordering relationship are certainly very low. If we consider nonlinear models, the pressure to find this nonlinear projection can result in unnatural or too complex projections leading to poorer generalisation performance. This paper proposes

to relax this pressure by allowing a higher dimensional representation of the latent space. This is done by ordering patters in an $L$-dimensional space, where each class region is limited by concentric hyperspheres (centred in the origin). The ordering of the classes is imposed by assuring the radii of the hyperspheres are also ordered.

Another popular way to tackle OR problems is to decompose the original task into several binary tasks, where each binary task consists of predicting if the patterns belongs to a category higher (in the ordinal scale) than a given label $C_q$. One model is estimated for each class in the ordinal scale. The approach is presented in the work of Frank and Hall [11], also proposing a way to fuse probabilities given for all binary tasks. Later on, there have been two different lines of research where binary classification and OR were linked in a more direct way [12, 13, 14]. Instead of learning $Q$ different binary classifiers, a single binary classifier is learnt, where the category examined is included as an additional feature and training patterns are replicated and weighted. The framework in [13, 14] is more generic, in the sense that it can be applied to different cost matrices.

The Error-Correcting Output Codes (ECOC) methodology is a popular and effective coding method to learn complex class targets, which can be used also for OR. The main idea is to associate each class $C_q$ with a column of a binary coding matrix $\mathbf{M}_{R\times Q}$, where each entry of the matrix $\mathbf{M}(j,q) \in \{-1,+1\}$, $Q$ is the number of classes, $R$ is the number of binary classifiers, $1 \leq q \leq Q$ and $1 \leq j \leq R$. After training the binary classifiers, prediction is then accomplished by choosing the column of $\mathbf{M}$ closest to the set of decision values, where the distance function should be selected according to the error function minimised during learning [15, 16].

In the field of neural networks, there have been some proposals for OR problems. The first one dates back to 1996, when Mathieson proposed a non-linear version of the POM [2, 17] by setting the projection $f(\mathbf{x})$ to be the output of a neural network. Although the results were quite promising, the method was evaluated for a very specific dataset. A more extensive battery of experiments should be done to further validate the proposal.

Costa [18] derived another neural network architecture to exploit the ordinal nature of the data. It was based on a "partitive approach", where probabilities are assigned to the joint prediction of constrained concurrent events.

Other approach [19] applies the coding scheme of Frank and Hall and a decision rule based on examining output nodes with an order and selecting the first one whose output is higher than a predefined threshold $T$. The problem of this method is that inconsistencies can be found in the predictions (i.e. a sigmoid with value higher than $T$ after the index selected).

The ordinal neural network (oNN) of Cardoso and da Costa [12] adapts the previously discussed data replication method to neural networks (a single model for binary decomposition using an extended and replicated version of the dataset), allowing the derivation of nonlinear decision boundaries.

Additionally, extreme learning machines (ELMs) have been used as a very fast method to fit single layer neural networks, where the hidden neurons weights are random, and the output weights are analytically obtained [20]. They have been adapted to OR [21], considering again the Frank and Hall coding scheme and a prediction based on the ECOC loss-based decoding approach [15], i.e. the chosen label is that which minimises the exponential loss. Another recent paper by Riccardi et al. [22] introduces a cost-sensitive approach for adapting the stagewise additive modelling using a multiclass exponential boosting algorithm (SAMME, which is the multiclass version of the well-known AdaBoost) to OR problems. They consider ELMs as the base classifier and they introduce three different loss functions, affecting the update rule of the error estimation and/or of the pattern weights [22]. From the three variants introduced in the paper, the third one (which adapts the update rule of both the error estimation and the pattern weights) obtains the best results. The OR model proposed in [23] adapts ELM to OR problems by imposing monotonicity constraints in the weights connecting the hidden layer with the output layer. The optimum of the inequality constrained least squares problem is determined analytically according to the closed-form solution estimated from the Karush–Kuhn–Tucker conditions.

A conceptually different methodology is proposed by da Costa et al. [24, 25] for training OR models, with a special attention to neural networks. They assume that the random variable class of a pattern should follow a unimodal distribution. Two possible implementations are considered: a parametric one, where a specific discrete distribution is assumed and the associated free parameters are estimated by a neural network; and a non-parametric one, where no distribution is assumed but the error function is modified to avoid errors from distant classes. Finally, the approach in [26] is a distribution-independent methodology for OR based on pairwise preferences. The strength of dependency between two data instances (continuous preferences) is shown to improve algorithmic performance, obtaining competitive results.

In this paper, we extend the proposal of Mathieson [2, 17], deriving a nonlinear version of the POM based on neural networks. We present a common learning framework to fit the parameters of a nominal neural network (NNN) and the neural network based on the POM (POMNN). The framework is then used to fit an extended version of the POMNN, where, as previously discussed, the latent space is assumed to be $L$-dimensional and the patterns are ordered by considering $Q-1$ concentric hyperspheres. The underlying motivation is to relax the imposition of projecting all patterns in a real line.

With regards to the relationship between ECOC and the proposal of this paper, one single model is used for learning the ordinal target, and the problem is not decomposed in several binary ones. In this way, the latent space structure relates each pattern to the posterior probabilities without learning multiple binary classifiers.

This paper is organised as follows: Section 2 is devoted to a brief analysis of the POM model, closely related to the models proposed; the description of the different ordinal neural network models is carried out in Section 3; Section 4 contains the experimental results; and finally, Section 5 summarises the conclusions of our work.

## 2. Proportional Odds Model (POM)

This model is a direct extension of binary logistic regression for the case of OR. It was first presented by McCullagh [10] and dates back to 1980. POM can be grouped under a wider family of models, the cumulative link models (CLMs) [27], which predict probabilities of adjacent categories, taking the ordinal scale into account. The key idea of CLMs is to estimate cumulative probabilities as follows:

$$P(y \preceq C_q|\mathbf{x}) = P(y = C_1|\mathbf{x}) + \ldots + P(y = C_q|\mathbf{x}),$$
$$P(y = C_q|\mathbf{x}) = P(y \preceq C_q|\mathbf{x}) - P(y \preceq C_{q-1}|\mathbf{x}),$$

for $1 < q \leq Q$, considering by definition that $P(y \preceq C_Q|\mathbf{x}) = 1$ and $P(y = C_1|\mathbf{x}) = P(y \preceq C_1|\mathbf{x})$. CLMs relate a linear model of the input variables to these cumulative probabilities:

$$f(\mathbf{x}) = g^{-1}\left(P(y \preceq C_q|\mathbf{x})\right) = b_q - \mathbf{w}^T\mathbf{x},$$

where $g^{-1} : [0, 1] \rightarrow (-\infty, +\infty)$ is a monotonic transformation (the inverse link function), $b_q$ is the threshold defined for class $C_q$, and $\mathbf{w}$ is the coefficient vector of the linear model. The most common choice for the link function is the logistic function (which is indeed the one selected for the POM [10]), although probit, complementary log-log, negative log-log or cauchit functions could also be used [27]. The logit link function is the inverse of the standard logistic cumulative distribution function (cdf), with the following expression:

$$g^{-1}\left(P(y \preceq C_q|\mathbf{x})\right) = \ln\left(\frac{P(y \preceq C_q|\mathbf{x})}{(1 - P(y \preceq C_q|\mathbf{x})}\right), \quad (1)$$

while the probit link function is the inverse of the standard normal cdf:

$$g^{-1}\left(P(y \preceq C_q|\mathbf{x})\right) = \Phi^{-1}(P(y \preceq C_q|\mathbf{x})), \quad (2)$$

where $\Phi(x)$ is the normal distribution, and the cauchit link function is the inverse of the Cauchy cdf (characterised for having long thick tails):

$$g^{-1}\left(P(y \preceq C_q|\mathbf{x})\right) = \tan\{\pi \cdot [P(y \preceq C_q|\mathbf{x}) - 0.5]\}. \quad (3)$$

Under the assumption that $f(\mathbf{x})$ follows a logistic cdf and following the idea of the POM model [10], the cumulative likelihood of a pattern being associated with a class less than or equal to class $C_q$ is defined as:

$$P(y \preceq C_q|\mathbf{x}) = \frac{1}{1 + \exp(f(\mathbf{x}) - b_q)}, \quad (4)$$

where $q = 1, \ldots, Q$, and, by definition, $P(y \preceq C_Q|\mathbf{x}) = 1$.Therefore, this model approximates the posterior probability of a class $j$ as:

$$P(y = C_q|\mathbf{x}) = P(y \preceq C_q|\mathbf{x}) - P(y \preceq C_{q-1}|\mathbf{x}) =$$
$$= \frac{1}{1 + \exp(f(\mathbf{x}) - b_q)} - \frac{1}{1 + \exp(f(\mathbf{x}) - b_{q-1})}. \quad (5)$$

Thresholds must satisfy the constraint $b_1 < b_2 < \ldots < b_{Q-1}$ and their role is to divide the real line into $Q$ contiguous intervals; these intervals map the function value $f(\mathbf{x})$ into the discrete variable, while forcing a proper probability interpretation (increasing probability when a higher class in the ordinal scale is examined).

## 3. Neural network classification algorithms

This section explains the characteristics of the different neural network models considered for the present work, including the ones proposed. The first subsection will introduce the learning algorithm which will be based on a set of estimated probabilities $\mathbf{p}(\mathbf{x}) = \{P(y = C_1|\mathbf{x}), P(y = C_2|\mathbf{x}), \ldots, P(y = C_Q|\mathbf{x})\}$. The different ways of obtaining this probability vector $\mathbf{p}(\mathbf{x})$ constitute the different neural network models considered.

### 3.1. Learning algorithm

As previously stated, our aim is to estimate a classification rule $L$ based on a training set $D$. If we consider a "1-of-$Q$" encoding vector for representing the class labels, then we define a vector $\mathbf{y}_i = \left(y_i^{(1)}, y_i^{(2)}, ..., y_i^{(Q)}\right)$ for each training label $y_i$, such that $y_i^{(q)} = 1$ if $\mathbf{x}_i$ corresponds to an example belonging to class $C_q$ and $y^{(q)} = 0$ otherwise. We will denote $\theta$ to the vector of free parameters of the model to be learnt, which will be specified in following subsections. To perform the maximum likelihood estimation of the parameter vector $\theta$, we choose to minimise the cross-entropy error function:

$$L(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{q=1}^{Q} y_n^{(q)} \log P(y = C_q|\mathbf{x}_n, \theta). \quad (6)$$

An individual should be assigned to the class which has the maximum probability, given the measurement vector $\mathbf{x}$:

$$F(\mathbf{x}) = \hat{q}, \text{ where } \hat{q} = \arg\max_q P(y = C_q|\mathbf{x}_n, \theta),\ q = 1, ..., Q.$$

Considering the $L(\theta)$ error function, we optimise the model parameters by gradient descent using the $iRprop^+$ local improvement procedure [28]. This training scheme combines the local information (i.e. the sign of the partial derivative of the error with respect to a weight like $Rprop$) with more global information (i.e. the error value at each iteration) in order to decide whether to revert an update step for each weight individually, resulting in very robust performance [28].

For the sake of simplicity, let $S$ be the total number of parameters of the model. The gradient vector is given by:

$$\nabla L(\theta) = \left(\frac{\partial L(\theta)}{\partial \theta_1}, \frac{\partial L(\theta)}{\partial \theta_2}, \ldots, \frac{\partial L(\theta)}{\partial \theta_S}\right). \quad (7)$$

Considering (6), each of its component will be defined as:

$$\frac{\partial L}{\partial \theta_s} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{q=1}^{Q} \frac{y_n^{(q)}}{P(y = C_q|\mathbf{x}_n, \theta)} \cdot \frac{\partial P(y = C_q|\mathbf{x}_n, \theta)}{\partial \theta_s},$$

where $s = 1, \ldots, S$. These derivatives will depend on the actual model form and they will be specified in the following subsections.

### 3.2. Nominal neural network (NNN)

Even when an ordinal classification problem is considered, one could use as a baseline a nominal neural network (NNN) to estimate $\mathbf{p}(\mathbf{x})$. A feed-forward multilayer perceptron can be configured with $Q$ output nodes and one hidden layer. For a more robust model and learning process, the output of this NNN should be transformed by considering the softmax transformation:

$$P(y = C_q|\mathbf{x}_n, \theta) = \frac{\exp\left(h_q\left(\mathbf{x}, \theta_q\right)\right)}{\sum_{j=1}^{Q} \exp\left(h_j\left(\mathbf{x}, \theta_j\right)\right)}, \tag{8}$$

where $1 \leq q \leq Q$, $\theta = (\theta_1, \theta_2, ..., \theta_Q)$ and $h_q\left(\mathbf{x}, \theta_q\right)$ is the output of the $q$-th node of the output layer. A proper probability distribution should assure that $\sum_{l=1}^{Q} P(y = C_l|\mathbf{x}_n, \theta) = 1$, what implies that the probability for one of the classes could be expressed as a function of the others, reducing the degrees of freedom of the model. This can be done by setting one class as the reference class (in our case, the last class, $C_Q$), and dividing numerator and denominator by $\exp\left(g_Q\left(\mathbf{x}, \theta_Q\right)\right)$:

$$P(y = C_q|\mathbf{x}_n, \theta) = \frac{\exp\left(g_q(\mathbf{x}, \theta_q) - g_Q\left(\mathbf{x}, \theta_Q\right)\right)}{1 + \sum_{j=1}^{Q-1} \exp\left(g_j(\mathbf{x}, \theta_j) - g_Q\left(\mathbf{x}, \theta_Q\right)\right)}$$

for $q = 1, \ldots, Q-1$. Now we set $f_q(\mathbf{x}, \theta_q) = g_q(\mathbf{x}, \theta_q) - g_Q\left(\mathbf{x}, \theta_Q\right)$ and the final model reduces to:

$$P(y = C_q|\mathbf{x}_n, \theta) = \frac{\exp\left(f_q\left(\mathbf{x}, \theta_q\right)\right)}{1 + \sum_{j=1}^{Q-1} \exp\left(f_j\left(\mathbf{x}, \theta_j\right)\right)}, 1 \leq q \leq Q-1,$$

$$P(y = C_Q|\mathbf{x}_n, \theta) = \frac{1}{1 + \sum_{j=1}^{Q-1} \exp\left(f_j\left(\mathbf{x}, \theta_j\right)\right)},$$

which is equivalent to Eq. (8) when the last output is set to zero, $f_Q\left(\mathbf{x}, \theta_Q\right) = 0$. This way the number of model parameters is reduced. For the rest of classes, the outputs are defined by a linear combination of the hidden nodes, in the following way:

$$f_q(\mathbf{x}, \theta_q) = f(\mathbf{x}, \boldsymbol{\beta}_q, \mathbf{W}) = \beta_0^q + \sum_{j=1}^{M} \beta_j^q B_j(\mathbf{x}, \mathbf{w}_j),$$

where $1 \leq q \leq Q-1$, $\boldsymbol{\beta}_q = \{\beta_0^q, \beta_1^q, \ldots, \beta_M^q\}$, $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$, $\mathbf{w}_j = \{w_{j0}, w_{j1}, \ldots, w_{jk}\}$, and $B_j(\mathbf{x}, \mathbf{w}_j)$ can be any kind of basis function, in our case, sigmoidal units:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \left(1 + \exp\left(-w_{j0} - \sum_{i=1}^{k} w_{ji} \cdot x_i\right)\right)^{-1}. \tag{9}$$

With this configuration, the derivatives are given in the following way. For the sake of simplicity, let $\theta_s$ be any of the parameters of $\boldsymbol{\beta}_q$ or $\mathbf{W}$, $P(y = C_q|\mathbf{x}_n, \theta) = p_{nq}$ and $f_q(\mathbf{x}_n, \theta_q) = f_{nq}$:

$$\frac{\partial p_{nq}}{\partial \theta_s} = \sum_{j=1}^{Q} p_{nq} \cdot \left(I\left(j = q\right) - p_{nj}\right) \cdot \frac{\partial f_{nq}}{\partial \theta_s},$$

where $I(\cdot)$ is the indicator function. The derivatives of the parameters of the model output functions $f_{nq}$ can be expressed as:

$$\frac{\partial f_{nq}}{\partial \beta_0^k} = \begin{cases} 0 & \text{if } q \neq k, \\ 1 & \text{if } q = k. \end{cases}, \quad \frac{\partial f_{nq}}{\partial \beta_j^k} = \begin{cases} 0 & \text{if } q \neq k, \\ B_j(\mathbf{x}_n, \mathbf{w}_j) & \text{if } q = k. \end{cases}$$

and the gradient for the hidden layer depends on the kind of basis function used. For sigmoidal nodes:

$$\frac{\partial f_{nq}}{\partial w_{jt}} = \beta_j^q B_j(\mathbf{x}_n, \mathbf{w}_j)\left(1 - B_j(\mathbf{x}_n, \mathbf{w}_j)\right) x_{nt}, \quad 1 \leq t \leq k,$$

$$\frac{\partial f_{nq}}{\partial w_{j0}} = \beta_j^q B_j(\mathbf{x}_n, \mathbf{w}_j)\left(1 - B_j(\mathbf{x}_n, \mathbf{w}_j)\right).$$

### 3.3. Proportional odds model neural network (POMNN)

The fact that the POM is linear limits its applicability in real world datasets, given that the parallel linear decision boundaries are often unrealistic. A non-linear version of the POM model can be formed by setting the projection $f(\mathbf{x})$ to be the output of a neural network. While the POM model approximates $f(\mathbf{x})$ by a simple linear combination of the input variables, the POMNN considers a non-linear basis transformation of the inputs. For each class we will have:

$$f_q(\mathbf{x}, \theta_q) = f_q(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W}, b_q) = b_q - f(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W})$$

where $q = 1 \ldots Q-1$, $\theta_q = \{\boldsymbol{\beta}, \mathbf{W}, b_q\}$ and the projection function $f(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W})$ is estimated by:

$$f(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W}) = \sum_{j=1}^{M} \beta_j B_j(\mathbf{x}, \mathbf{w}_j),$$

where $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_M\}$, $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$, and $B_j(\mathbf{x}, \mathbf{w}_j)$ can be any kind of basis functions, in our case, sigmoidal units. Note that in this case the neural network model will have only one output node. As compared with the model of Mathieson [17], that model had skip-layer connections and the error function was endowed with a regularization term. Moreover, in order to ensure the constraints in the biases, $b_1 \leq \ldots \leq b_{Q-1}$, we propose the following definition: $b_q = b_{q-1} + \Delta_q^2$, $q = 1 \ldots Q-1$, with padding variables $\Delta_j$, which are squared to make them positive, and $b_1, \Delta_q \in \mathbb{R}$. Consequently, the parameter vector is defined as: $\theta = \{\boldsymbol{\beta}, \mathbf{W}, b_1, \Delta_2, \ldots, \Delta_{Q-1}\}$. In this way, we convert the constrained optimization problem into an unconstrained one, and gradient descent can be applied to all the parameters in $\theta$ without considering the constraints.

With this model structure, the derivatives can now be reformulated to perform gradient-descent optimization. Let $\theta_s$ be any of the parameters of $\boldsymbol{\beta}$ or $\mathbf{W}$, $P(y = C_q|\mathbf{x}_n, \theta) = p_{nq}$ and $f(\mathbf{x}_n, \boldsymbol{\beta}, \mathbf{W}) = f_n$:

$$\begin{aligned}
p_{nq|q>1} &= \frac{1}{1 + \exp(b_q - f_n)} - \frac{1}{1 + \exp(b_{q-1} - f_n)}, \\
p_{n1} &= \frac{1}{1 + \exp(b_1 - f_n)}, \\
\frac{\partial p_{nq|q>1}}{\partial \theta_s} &= \frac{\exp(b_q - f_n)}{(1 + \exp(b_q - f_n))^2} \frac{\partial f_n}{\partial \theta_s} \\
&- \frac{\exp(b_{q-1} - f_n)}{(1 + \exp(b_{q-1} - f_n))^2} \frac{\partial f_n}{\partial \theta_s}, \\
\frac{\partial p_{n1}}{\partial \theta_s} &= \frac{\exp(b_1 - f_n)}{(1 + \exp(b_1 - f_n))^2} \frac{\partial f_n}{\partial \theta_s},
\end{aligned}$$

where the derivatives for the projection parameters $\theta_s$ are:

$$\begin{aligned}
\frac{\partial f_n}{\partial \beta_j} &= B_j(\mathbf{x}_n, \mathbf{w}_j), \\
\frac{\partial f_n}{\partial w_{jt}} &= \beta_j B_j(\mathbf{x}_n, \mathbf{w}_j)\Big(1 - B_j(\mathbf{x}_n, \mathbf{w}_j)\Big)\, x_{nt}, 1 \le t \le k, \\
\frac{\partial f_n}{\partial w_{j0}} &= \beta_j B_j(\mathbf{x}_n, \mathbf{w}_j)\Big(1 - B_j(\mathbf{x}_n, \mathbf{w}_j)\Big).
\end{aligned}$$

The derivatives for the biases and the padding variables can be formulated in the following way:

$$\begin{aligned}
\frac{\partial p_{nq|q>1}}{\partial b_1} &= \frac{\exp(f_n - b_q)}{(1+\exp(f_n - b_q))^2} - \frac{\exp(f_n - b_{q-1})}{(1+\exp(f_n - b_{q-1}))^2}, \\
\frac{\partial p_{n1}}{\partial b_1} &= \frac{\exp(f_n - b_1)}{(1+\exp(f_n - b_1))^2}, \\
\frac{\partial p_{nq|q>1}}{\partial \Delta_j} &= 2\Delta_j \cdot \frac{\partial p_{nq|q>1}}{\partial b_1},\ j = 2 \ldots Q-1, \\
\frac{\partial p_{n1}}{\partial \Delta_j} &= 2\Delta_j \cdot \frac{\partial p_{n1}}{\partial b_1},\ j = 2 \ldots Q-1.
\end{aligned}$$

### 3.4. Concentric hypersphere neural network (CHNN)

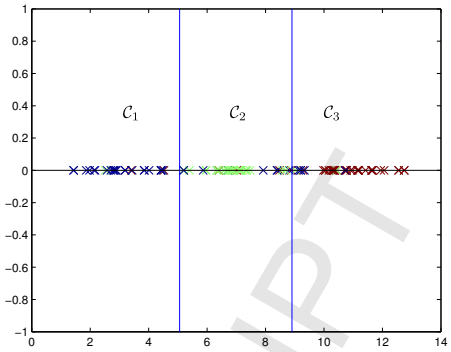
Both POM and POMNN project input patterns to the real line with the difference that POM impose a linear projection and POMNN a nonlinear one. After this, the projection is divided into intervals, each representing a different class. Finding such one-dimensional projection where patterns are ordered according to the class labels can be too harsh an imposition for some datasets. We propose to relax this requirement by allowing a multivariate representation. The class order will be represented by a natural order of nested concentric hyper-spheres (centered at the origin). The proposal tries to find a space made up by several different projections (coordinates) where the order of the class labels is presented in the form of concentric hyperspheres. The smallest hypersphere will contain the patterns of the first class in the ordinal scale.

An example showing the proposed idea is presented in Figure 1, where the one-dimensional projection of POMNN is compared against the two-dimensional one of the proposed CHNN approach. As can be seen, the patterns can be more easily positioned in their correct region, because they are projected into a two-dimensional space, where each coordinate is approximated separately.

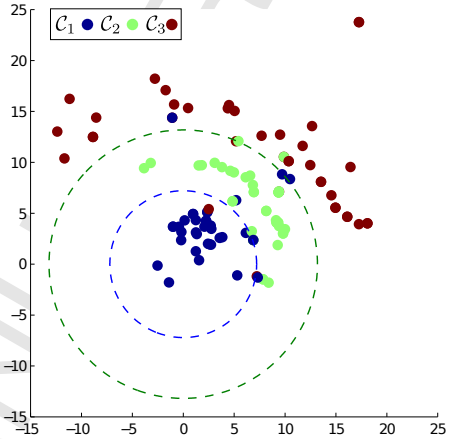
Non-linearity is achieved by letting these projections be the output of different linear combinations of the basis functions. Input conditional class probabilities are related to the distance of the pattern to the centre of the hyperspheres. Specifically, the latent space is defined in the following way:

$$f_l(\mathbf{x}, \theta_l) = f(\mathbf{x}, \boldsymbol{\beta}_l, \mathbf{W}) = \beta_0^l + \sum_{j=1}^{M} \beta_j^l B_j(\mathbf{x}, \mathbf{w}_j), 1 \le l \le L,$$

where $L$ is the latent space dimensionality (hence dimensionality of the hyperspheres), $\boldsymbol{\beta}_l = \{\beta_0^l, \beta_1^l, \ldots, \beta_M^l\}$ is the vector of coefficients of the linear combination for the $l$-th projection,




(a) One-dimensional projection of POMNN




(b) Two-dimensional projection of CHNN

Figure 1: POMNN projection for the `tae` dataset and proposed two-dimensional CHNN projection for the same dataset.

and $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$ is the matrix containing the parameters of the basis functions. The matrix $\mathbf{W}$ is common for all the projections, while the corresponding $\boldsymbol{\beta}_l$ vectors are specific for each one. In this way, the coordinates of the pattern in the latent space are decided by each of the $f_l(\mathbf{x}, \theta_l)$ functions. Consequently, each input pattern $\mathbf{x}$ is projected using the following mapping $\phi : \mathbb{R}^k \to \mathbb{R}^L$:

$$\phi(\mathbf{x}) = \{f_1(\mathbf{x}, \theta_1), f_2(\mathbf{x}, \theta_2), \ldots, f_L(\mathbf{x}, \theta_L)\}.$$

The dimensionality of the latent space is an additional parameter and different values will be considered in the experimental section.

The input conditional class probability is related to the distance to origin of the derived space. In order to simplify the calculus, we obtain the norm of the projection:

$$\|\phi(\mathbf{x})\| = \sqrt{f_1(\mathbf{x}, \theta_1)^2 + f_2(\mathbf{x}, \theta_2)^2 + \ldots + f_L(\mathbf{x}, \theta_L)^2},$$

and then the different classes are defined by a set of thresholds $\{b_1, \ldots, b_{Q-1}\}$, as in POM:

$$P(y \le C_q|\mathbf{x}) = \frac{1}{\Big(1 + \exp\Big(b_q - \|\phi(\mathbf{x})\|\Big)\Big)}, q = 1, \ldots, Q-1.$$

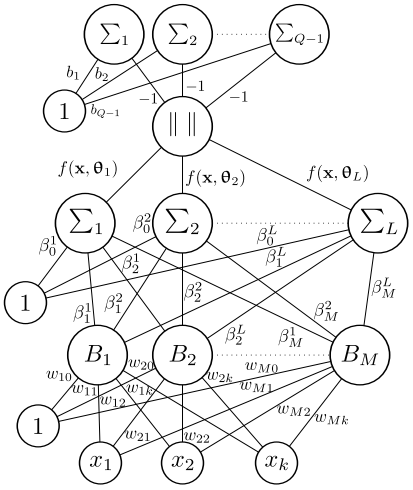


Figure 2: CHNN structure and topology

The scheme of the proposed neural network can be seen in Figure 2.

Again, we need a set of thresholds – radiuses of the concentric hyperspheres in the latent space. These thresholds have to respect the class order, $b_1 \leq b_2 \leq \ldots \leq b_{Q-1}$, so we again introduce padding variables (see previous subsection), $b_q = b_{q-1} + \Delta_q^2$, $1 < q < Q$, $b_1, \Delta_q \in \mathbb{R}$. Consequently, the parameter vector is now $\theta = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_L, \mathbf{W}, b_1, \Delta_2, \ldots, \Delta_{Q-1}\}$.

With respect to the derivatives, they are the same than for POMNN model, if we replace $f_n$ by $\|\phi(\mathbf{x})\|$. Then:

$$\frac{\partial f_n}{\partial \theta_s} = \frac{2 \cdot f(\mathbf{x}, \theta_l)}{\|\phi(\mathbf{x})\|^4} \cdot f_{nl} \cdot \frac{\partial f_{nl}}{\partial \theta_s},$$

where $f_{nl}$ is $f(\mathbf{x}_n, \theta_l)$ and the derivatives of the parameters of each latent dimension $f_{nl}$ can be expressed as:

$$\frac{\partial f_{nl}}{\partial \beta_0^k} = \begin{cases} 0 & \text{if } l \neq k, \\ 1 & \text{if } l = k. \end{cases}, \quad \frac{\partial f_{nl}}{\partial \beta_j^k} = \begin{cases} 0 & \text{if } l \neq k, \\ B_j(\mathbf{x}_n, \mathbf{w}_j) & \text{if } l = k. \end{cases}$$

The gradient for the hidden layer depends on the basis function. For sigmoidal nodes:

$$\begin{aligned} \frac{\partial f_{nl}}{\partial w_{jt}} &= \beta_j^l B_j(\mathbf{x}_n, \mathbf{w}_j)\left(1 - B_j(\mathbf{x}_n, \mathbf{w}_j)\right) x_{nt}, 1 \leq t \leq k, \\ \frac{\partial f_{nl}}{\partial w_{j0}} &= \beta_j^l B_j(\mathbf{x}_n, \mathbf{w}_j)\left(1 - B_j(\mathbf{x}_n, \mathbf{w}_j)\right). \end{aligned}$$

## 4. Experiments

### 4.1. Experimental design

The proposed methodology was applied to 12 different datasets taken from the UCI repository [29]. As can be seen in Table 1, the characteristics vary noticeably among the datasets. We included a more controlled dataset (the `toy` one), which is a 2D problem synthetically generated, using the instructions given in [25]. The performance of the different methods was evaluated by the following two metrics [30]:

Table 1: Characteristics of the 12 datasets used for the experiments: number of instances (Size), total number of inputs (#In.), number of classes (#Out.), and number of patterns per-class (NPPC)

| Dataset | Size | #In. | #Out. | NPPC |
|---|---|---|---|---|
| balance | 625 | 4 | 3 | (288,49,288) |
| car | 1728 | 21 | 4 | (1210,384,69,65) |
| ERA | 1000 | 4 | 9 | (92,142,181,172,158,118,88,31,18) |
| ESL | 488 | 4 | 9 | (2,12,38,100,116,135,62,19,4) |
| LEV | 1000 | 4 | 5 | (93,280,403,197,27) |
| newth. | 215 | 5 | 3 | (30,150,35) |
| pasture | 36 | 25 | 3 | (12,12,12) |
| sq.-st. | 52 | 51 | 3 | (23,21,8) |
| sq.-unst. | 52 | 52 | 3 | (24,24,4) |
| SWD | 1000 | 10 | 4 | (32,352,399,217) |
| tae | 151 | 54 | 3 | (49,50,52) |
| toy | 300 | 2 | 5 | (35,87,79,68,31) |

- The Correctly Classified Ratio ($CCR$) is the error rate of the classifier, expressed as a percentage:

$$CCR = \frac{100}{N} \sum_{i=1}^{N} I(y_i^* = y_i),$$

where $y_i$ is the true label and $y_i^*$ is the predicted label. This metric evaluates the global classification performance task without taking into account the different kinds of errors with regards to the category order.

- The Mean Absolute Error ($MAE$) is the average deviation in absolute value of the predicted rank from the true one [31]:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |O(y_i) - O(y_i^*)|,$$

where the position of a label in the ordinal scale is expressed by the function $O$ in the form $O(C_q) = q, 1 \leq q \leq Q$. $|O(y_i) - O(y_i^*)|$ is the distance between the true and predicted ranks. $MAE$ values range from 0 to $Q - 1$ (maximum deviation in number of categories).

All the compared algorithms have been run 30 times for each dataset, considering a random holdout partition with a 75% of patterns for the training set and the remaining 25% for testing generalization performance of the obtained classifier. The partitions were stratified in the sense that both partitions approximately presented the same class distribution of the complete dataset, and the same random partitions were considered for all the methods.

Apart from the methods previously presented in the paper (POM, NNN, POMNN and CHNN), different state-of-the-art methods were included in the experimentation for comparison purposes.

- Support vector ordinal regression (SVOR) by Chu et. al [7, 6], who optimized multiple thresholds in order to de-

fine parallel discriminant hyperplanes for the ordinal scales. In one first approach with explicit inequality constraints on the thresholds, they derive the optimality conditions for the dual problem, and adapt the SMO algorithm for the solution. We will refer it to as SVOREX. There is a second approach based on implicit constraints (SVORIM), but we consider SVOREX as we have found slightly better results for it in a previous work [32].

- Kernel discriminant learning for ordinal regression (KDLOR) [8], which is an adaption of kernel discriminant analysis to the field of ordinal regression. Order on classes is imposed by constraining the projection to be obtained from the optimization problem.

- Extreme learning machine with ordered partitions (ELMOP) [21], which is the ordinal regression version of ELMs. They are adapted by relabelling the dataset using the binary coding proposed in [11] and then fitting one multi-output model or several binary models based on the Error-Correcting Output Codes (ECOC) framework. The authors present the single model ELM as the one with the best performance [21], so this is the configuration chosen for ELMOP.

- AdaBoost for ordinal regression based on an ordinal cost model for both the error estimation and pattern weights (ABORC3). As proposed in [22], the base classifier is an ELM with Gaussian kernel and a regularization parameter and the weighted least squares closed-form solution of the error function was considered for estimating the linear parameters of the individuals in the final ensemble model.

Regarding hyper-parameter tuning, the following procedure has been applied. For kernel algorithms, i.e. SVOREX, KDLOR and ABORC3, the width of the Gaussian kernel, $\gamma$, was adjusted using a grid search with a 10-fold cross-validation, considering the following range: $\gamma \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$. An additional parameter $u$ was also used for KDLOR in order to avoid singularities in the covariance matrices, $u \in \{10^{-6}, 10^{-5}, \ldots, 10^{-2}\}$. For SVOREX and ABORC3, the additional cost parameter was adjusted by using the range $C \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$. Finally, the results of the ABORC3 algorithm were directly taken from [22], given that the authors used the same partitions than in this paper (as presented in [32]), and the same configuration for $\gamma$ and $C$ parameters. The number of members of the ensemble model was $M = 25$.

For the neural network algorithms, the hyper-parameters (number of hidden neuron, $H$, and number of iterations of the local search procedure, $N$), were adjusted using a grid search with a 5-fold cross-validation, considering the following ranges: $H \in \{1, 5, 15, 20, 30, 40\}$ and $N \in \{100, 200, 300, 400, 500\}$. For ELMOP, a higher numbers of hidden neurons are considered, $H \in \{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, given that it relies on sufficiently informative random projections [20]. With respect to $iRprop^{+}$, the parameter values are set to those specified in [28]. Although more advanced methods could have been considered for adjusting the structure of the neural networks [33, 34], we have selected cross-validation to ease the comparisons against other state-of-the-art methods.

The CHNN algorithm has been run with different options for the dimensionality of the latent space ($L \in \{1, 2, 3, 4\}$), leading to four versions of the algorithm: CHNN1D, CHNN2D, CHNN3D and CHNN4D. Indeed, the CHNN1D is very similar to POMNN (they both project the patterns into a one dimensional space), but the classes are arranged in nested intervals for CHNN1D, while the intervals are consecutive in the case of POMNN.

### 4.2. *Comparison of the different alternatives for the neural network algorithms*

In the first part of this section, we compare the different neural network algorithms in order to check 1) whether considering an ordinal regression configuration is able to improve the performance of the NNN classifier, and 2) if the additional dimensions for latent space helps to better separate the classes. The results are shown in Table 2. This table includes the average ranking ($\overline{R}$) obtained for each metric and all datasets ($R = 1$ for the best method, $R = 6$ for the worst one).

First of all, the attention should be drawn in the results of the nominal version (NNN) of the model presented in this paper. As can be seen, there are only two datasets (LEV and pasture), where the nominal version of the algorithm obtain a first or a second position (for $CCR$ and $MAE$) when compared to the ordinal versions. A further analysis should be done on those two datasets to check if the order scale of the target variable can be found (in a linear or non-linear form) in the input space distribution of the training patterns. Regarding the ordinal methods, the one dimensional projection of CHNN1D (where the intervals are nested) seems to be too restrictive, and the results are a bit worse than those of POMNN. However, when the dimensionality is increased (CHNN2D and CHNN3D), the results are considerably better, although a too high dimensionality seems to compromise the learning capability (CHNN4D).

The projections of the CHNN2D model for the `tae` dataset are included in Figure 1b, showing a higher degree of flexibility than the POMNN model (Figure 1a). The projection of the patterns in this space can be used to have better understanding about how they are organised with regards to the classification task, serving as a visualisation tool. In Figure 3, the result of CHNN3D has been included for one of the runs of `sq.-st.` dataset, where the model only confuses one green pattern, labelled as red.

An analysis of the significance of the differences found was also performed. We consider a procedure for comparing multiple classifiers over multiple datasets, following the guidelines of Demšar [35]. It begins with the Friedman test [36, 37], using the $CCR$ ranking as the test variable. This test is a non-parametric equivalent to the repeated-measures ANOVA test. We apply it because a previous evaluation of the $CCR$ ranking values results in rejecting the normality and the equality of variances' hypothesis. Applying this test to the average ranks at Table 2, the test shows that the effect of the method used for classification is statistically significant at a significance level of 5%,

Table 2: Statistical results obtained when comparing the different neural network algorithms considered for this work

| *CCR* (Mean$_{SD}$) | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | NNN | POMNN | CHNN1D | CHNN2D | CHNN3D | CHNN4D |
| balance | $95.75_{1.73}$ | $\mathbf{97.64_{1.51}}$ | $96.79_{1.45}$ | $97.24_{1.30}$ | $\mathit{97.28_{1.19}}$ | $97.11_{1.52}$ |
| car | $98.94_{0.53}$ | $98.98_{0.59}$ | $98.91_{0.71}$ | $\mathit{99.04_{0.35}}$ | $\mathbf{99.10_{0.54}}$ | $98.99_{0.57}$ |
| ERA | $26.49_{2.78}$ | $\mathbf{26.96_{2.47}}$ | $26.73_{2.17}$ | $\mathit{26.96_{2.51}}$ | $26.87_{2.46}$ | $26.37_{2.57}$ |
| ESL | $68.06_{3.00}$ | $70.08_{3.12}$ | $70.49_{2.54}$ | $\mathit{70.77_{3.11}}$ | $\mathbf{71.37_{2.88}}$ | $69.89_{3.51}$ |
| LEV | $\mathit{62.85_{2.58}}$ | $62.43_{2.97}$ | $62.67_{2.21}$ | $\mathbf{62.93_{2.20}}$ | $62.61_{2.71}$ | $62.85_{3.15}$ |
| newth. | $96.98_{2.15}$ | $97.04_{2.21}$ | $96.79_{2.48}$ | $\mathbf{97.28_{2.47}}$ | $\mathit{97.22_{2.32}}$ | $96.85_{2.24}$ |
| pasture | $\mathit{72.59_{14.51}}$ | $68.89_{13.81}$ | $64.07_{14.50}$ | $71.85_{14.22}$ | $\mathbf{74.07_{12.83}}$ | $69.63_{14.27}$ |
| sq. st. | $63.08_{12.35}$ | $\mathit{64.62_{10.99}}$ | $63.85_{10.91}$ | $63.85_{11.63}$ | $\mathbf{66.92_{12.48}}$ | $64.36_{13.17}$ |
| sq. unst. | $75.13_{13.20}$ | $\mathbf{76.92_{10.88}}$ | $66.15_{15.33}$ | $75.64_{10.71}$ | $\mathit{76.41_{13.69}}$ | $71.03_{12.40}$ |
| SWD | $56.81_{2.69}$ | $57.36_{3.23}$ | $57.84_{3.54}$ | $\mathit{57.92_{3.19}}$ | $\mathbf{57.96_{3.25}}$ | $57.55_{3.45}$ |
| tae | $57.19_{8.09}$ | $57.63_{7.29}$ | $55.61_{8.92}$ | $\mathit{59.04_{5.97}}$ | $\mathbf{59.91_{7.30}}$ | $56.93_{8.78}$ |
| toy | $92.93_{3.40}$ | $\mathbf{93.78_{2.63}}$ | $92.76_{3.26}$ | $93.11_{2.74}$ | $\mathit{93.42_{2.55}}$ | $92.44_{2.83}$ |
| $\overline{R}_{CCR}$ | 4.54 | 3.04 | 4.88 | *2.33* | **1.83** | 4.38 |
| *MAE* (Mean$_{SD}$) | | | | | | |
| Dataset | NNN | POMNN | CHNN1D | CHNN2D | CHNN3D | CHNN4D |
| balance | $0.055_{0.022}$ | $\mathbf{0.025_{0.018}}$ | $0.035_{0.016}$ | $0.030_{0.016}$ | $\mathit{0.029_{0.014}}$ | $0.031_{0.016}$ |
| car | $0.013_{0.007}$ | $0.010_{0.006}$ | $0.011_{0.007}$ | $\mathit{0.010_{0.004}}$ | $\mathbf{0.009_{0.005}}$ | $0.010_{0.006}$ |
| ERA | $1.301_{0.073}$ | $1.264_{0.049}$ | $1.263_{0.051}$ | $\mathit{1.258_{0.051}}$ | $\mathbf{1.257_{0.051}}$ | $1.262_{0.056}$ |
| ESL | $0.346_{0.035}$ | $0.315_{0.033}$ | $0.310_{0.029}$ | $\mathit{0.305_{0.034}}$ | $\mathbf{0.302_{0.032}}$ | $0.316_{0.037}$ |
| LEV | $\mathbf{0.406_{0.027}}$ | $0.412_{0.031}$ | $0.410_{0.023}$ | $0.407_{0.024}$ | $0.409_{0.029}$ | $\mathit{0.407_{0.033}}$ |
| newth. | $0.030_{0.021}$ | $0.030_{0.022}$ | $0.032_{0.025}$ | $\mathbf{0.027_{0.025}}$ | $\mathit{0.028_{0.023}}$ | $0.032_{0.022}$ |
| pasture | $\mathit{0.274_{0.145}}$ | $0.315_{0.143}$ | $0.367_{0.163}$ | $0.285_{0.151}$ | $\mathbf{0.259_{0.128}}$ | $0.304_{0.143}$ |
| sq. st. | $0.408_{0.143}$ | $\mathit{0.372_{0.116}}$ | $0.374_{0.112}$ | $0.372_{0.120}$ | $\mathbf{0.341_{0.129}}$ | $0.372_{0.143}$ |
| sq. unst. | $0.274_{0.156}$ | $\mathbf{0.233_{0.112}}$ | $0.344_{0.159}$ | $0.251_{0.118}$ | $\mathit{0.241_{0.140}}$ | $0.292_{0.125}$ |
| SWD | $0.464_{0.029}$ | $0.448_{0.035}$ | $0.443_{0.038}$ | $\mathit{0.443_{0.034}}$ | $\mathbf{0.441_{0.035}}$ | $0.447_{0.036}$ |
| tae | $0.550_{0.124}$ | $0.539_{0.096}$ | $0.541_{0.119}$ | $\mathit{0.514_{0.081}}$ | $\mathbf{0.497_{0.106}}$ | $0.517_{0.109}$ |
| toy | $0.072_{0.034}$ | $\mathbf{0.062_{0.026}}$ | $0.072_{0.033}$ | $0.069_{0.027}$ | $\mathit{0.066_{0.025}}$ | $0.076_{0.028}$ |
| $\overline{R}_{MAE}$ | 4.75 | 3.50 | 4.83 | *2.41* | **1.58** | 3.91 |

The best result is shown in bold face and the second one in italics.

as the confidence interval is $C_0 = (0, F_{(\alpha=0.05)} = 2.38)$ and the F-distribution statistical values are: 1) for *CCR*, $F^* = 9.48 \notin C_0$; and 2) for *MAE*, $F^* = 10.06 \notin C_0$. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking and a post-hoc test is needed. For the post-hoc test, the best performing method (CHNN3D) was considered as the control method, and it was compared to the remaining ones according to their rankings. It has been noted that the approach of comparing all classifiers to each other in a post-hoc test is not as sensitive as the approach of comparing all classifiers to a given classifier (a control method). One approach to this latter type of comparison is the Holm test. The test statistics for comparing the $i$-th and $j$-th method using this procedure is:

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}}, \tag{10}$$

where $k$ is the number of algorithms, $N$ is the number of datasets and $R_i$ is the mean ranking of the $i$-th method. The $z$ value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate level of significance $\alpha$. Holm test adjusts the value for $\alpha$ in order to compensate for multiple comparisons. This is done in a step-up procedure that sequentially tests the hypotheses ordered by their significance. We will denote the ordered p-values by $p_1, p_2, \ldots, p_k$ so that $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. Holm test compares each $p_i$ with $\alpha'_{\text{Holm}} = \alpha/(k-i)$, starting from the most significant $p$ value. If $p_1$ is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we allow to compare $p_2$ with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well.

Table 3: Holm test for the comparison of the different neural network algorithms: adjusted $p$-values using *CCR* and *MAE* as the test variables (CHNN3D is the control method).

| Variable Test: *CCR* | | | Variable Test: *MAE* | | |
|---|---|---|---|---|---|
| Algorithm | $p$-value | $\alpha'_{\text{Holm}}$ | Algorithm | $p$-value | $\alpha'_{\text{Holm}}$ |
| CHNN1D | $0.000_\star$ | 0.010 | CHNN1D | $0.000_\star$ | 0.010 |
| NNN | $0.000_\star$ | 0.013 | NNN | $0.000_\star$ | 0.013 |
| CHNN4D | $0.001_\star$ | 0.017 | CHNN4D | $0.002_\star$ | 0.017 |
| POMNN | 0.114 | 0.025 | POMNN | $0.012_\star$ | 0.025 |
| CHNN2D | 0.513 | 0.050 | CHNN2D | 0.275 | 0.050 |

$\star$: statistically signficant differences for $\alpha = 0.05$

The results of the Holm test can be seen in Table 3, using the corresponding $p$ and $\alpha'_{Holm}$ values. From the results of this test, it can be concluded that the CHNN3D methodology obtains a significantly higher ranking when compared to all methods except CHNN2D (for *CCR* and *MAE*) and POMNN (for *CCR*).

The reason why the proposed latent space helps to improve the classification for ordinal regression problems is that the ordering imposed by the projection helps to better locate new patterns in the ordinal scale, which is also truth for the POMNN model. However, as opposed to PONN, the CHNN model allows certain flexibility when constructing the multidimensional projection, which encourages more parsimonious models less prone to overfitting.

### 4.3. *Comparison of CHNN3D against other state-of-the-art algorithms*

This second part of the experiments compares the results obtained from CHNN3D to other state-of-the-art algorithms in

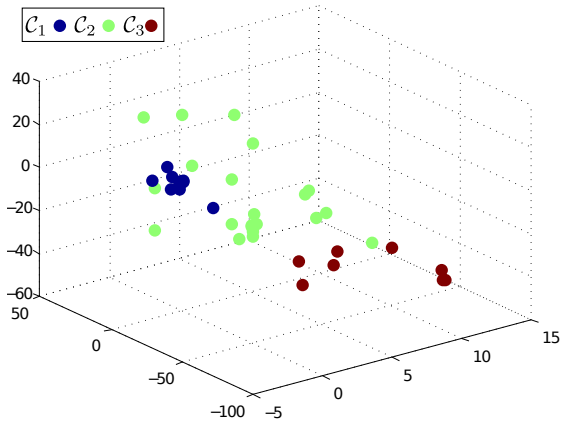



Figure 3: Three-dimensional CHNN projection for the `sq.-st.` dataset. Hyperspheres has been omitted for clarity.

Table 4: Statistical results obtained when comparing the CHNN3D approach against other state-of-the-art algorithms

| *CCR* (Mean$_{SD}$) | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | CHNN3D | ELMOP | POM | SVOREX | KDLOR | ABORC3 |
| balance | *97.28$_{1.19}$* | 91.85$_{2.30}$ | 90.55$_{1.86}$ | **99.79$_{0.56}$** | 83.69$_{2.40}$ | 95.12 |
| car | **99.10$_{0.54}$** | 84.85$_{1.10}$ | 15.75$_{30.63}$ | *98.74$_{0.53}$* | 95.35$_{0.83}$ | 98.30 |
| ERA | 26.87$_{2.46}$ | 24.55$_{1.69}$ | 25.61$_{2.11}$ | **28.60$_{2.62}$** | 20.51$_{3.46}$ | *27.24* |
| ESL | *71.37$_{2.88}$* | 69.51$_{3.30}$ | 70.55$_{3.36}$ | 71.23$_{3.36}$ | 65.38$_{3.68}$ | **71.88** |
| LEV | 62.61$_{2.71}$ | *62.85$_{2.58}$* | 62.33$_{2.80}$ | 62.55$_{1.97}$ | 54.93$_{3.46}$ | **65.01** |
| newth. | *97.22$_{2.32}$* | 94.88$_{2.51}$ | 97.22$_{2.22}$ | 96.85$_{2.39}$ | **97.53$_{2.02}$** | 96.96 |
| pasture | *74.07$_{12.83}$* | 61.11$_{13.29}$ | 49.63$_{15.37}$ | 66.67$_{11.67}$ | 66.30$_{12.54}$ | **74.44** |
| sq. st. | **66.92$_{12.48}$** | 56.41$_{15.16}$ | 38.21$_{15.18}$ | 64.36$_{14.21}$ | *64.87$_{11.90}$* | 59.49 |
| sq. unst. | **76.41$_{13.69}$** | 61.28$_{12.37}$ | 34.87$_{14.25}$ | 73.85$_{11.88}$ | *75.13$_{12.73}$* | 65.64 |
| SWD | *57.96$_{3.25}$* | 56.53$_{3.00}$ | 56.79$_{2.96}$ | 57.09$_{3.14}$ | 49.08$_{2.93}$ | **59.07** |
| tae | **59.91$_{7.30}$** | 54.39$_{7.52}$ | 50.44$_{7.73}$ | *57.89$_{5.82}$* | 57.63$_{5.76}$ | 57.89 |
| toy | 93.42$_{2.55}$ | 91.73$_{2.87}$ | 28.93$_{2.55}$ | **98.27$_{1.27}$** | 88.58$_{3.05}$ | *95.49* |
| $\bar{R}_{CCR}$ | **1.96** | 4.67 | 5.04 | 2.63 | 4.33 | *2.38* |

| *MAE* (Mean$_{SD}$) | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | CHNN3D | ELMOP | POM | SVOREX | KDLOR | ABORC3 |
| balance | *0.029$_{0.014}$* | 0.089$_{0.026}$ | 0.107$_{0.021}$ | **0.002$_{0.006}$** | 0.164$_{0.025}$ | 0.050 |
| car | **0.009$_{0.005}$** | 0.175$_{0.012}$ | 1.451$_{0.548}$ | *0.013$_{0.005}$* | 0.046$_{0.008}$ | 0.018 |
| ERA | 1.257$_{0.051}$ | 1.243$_{0.044}$ | 1.218$_{0.050}$ | **1.209$_{0.057}$** | 1.783$_{0.099}$ | *1.213* |
| ESL | *0.302$_{0.032}$* | 0.322$_{0.035}$ | 0.310$_{0.038}$ | 0.302$_{0.035}$ | 0.365$_{0.043}$ | **0.301** |
| LEV | 0.409$_{0.029}$ | *0.406$_{0.027}$* | 0.409$_{0.030}$ | 0.411$_{0.021}$ | 0.512$_{0.039}$ | **0.385** |
| newth. | *0.028$_{0.023}$* | 0.052$_{0.025}$ | 0.028$_{0.022}$ | 0.031$_{0.024}$ | **0.025$_{0.020}$** | 0.036 |
| pasture | *0.259$_{0.128}$* | 0.404$_{0.144}$ | 0.585$_{0.204}$ | 0.333$_{0.117}$ | 0.341$_{0.130}$ | **0.242** |
| sq. st. | **0.341$_{0.129}$** | 0.485$_{0.181}$ | 0.813$_{0.248}$ | *0.367$_{0.147}$* | 0.374$_{0.145}$ | 0.419 |
| sq. unst. | **0.241$_{0.140}$** | 0.423$_{0.137}$ | 0.826$_{0.230}$ | 0.262$_{0.119}$ | *0.251$_{0.132}$* | 0.354 |
| SWD | **0.441$_{0.035}$** | 0.452$_{0.031}$ | 0.450$_{0.030}$ | 0.446$_{0.031}$ | 0.579$_{0.035}$ | *0.443* |
| tae | 0.497$_{0.106}$ | 0.625$_{0.115}$ | 0.628$_{0.116}$ | *0.466$_{0.061}$* | **0.461$_{0.065}$** | 0.500 |
| toy | 0.066$_{0.025}$ | 0.083$_{0.029}$ | 0.981$_{0.039}$ | **0.017$_{0.013}$** | 0.114$_{0.030}$ | *0.043* |
| $\bar{R}_{MAE}$ | **2.29** | 4.58 | 4.83 | *2.45* | 4.17 | 2.67 |

The best result is shown in bold face and the second one in italics.

Table 5: Holm test for the comparison of the different neural network algorithms: adjusted $p$-values using *CCR* and *MAE* as the test variables (CHNN3D is the control method).

| Variable Test: *CCR*) | | | Variable Test: *MAE* | | |
|---|---|---|---|---|---|
| Algorithm | $p$-value | $\alpha'_{\text{Holm}}$ | Algorithm | $p$-value | $\alpha'_{\text{Holm}}$ |
| POM | 0.000$_\star$ | 0.010 | POM | 0.000$_\star$ | 0.010 |
| ELMOP | 0.000$_\star$ | 0.013 | ELMOP | 0.003$_\star$ | 0.013 |
| KDLOR | 0.002$_\star$ | 0.017 | KDLOR | 0.014$_\star$ | 0.017 |
| SVOREX | 0.383 | 0.025 | ABORC3 | 0.623 | 0.025 |
| ABORC3 | 0.546 | 0.050 | SVOREX | 0.827 | 0.050 |

$\star$: statistically signficant differences for $\alpha = 0.05$

order to check if the method can be considered competitive. The results of this comparison are included in Tables 4 and 5 and follow the same format that in the previous comparison. Again, the Friedman test shows that the effect of the method used for classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{(\alpha=0.05)} = 2.38)$ and the F-distribution statistical values are: 1) for *CCR*, $F^* = 11.23 \notin C_0$; and 2) for *MAE*, $F^* = 6.72 \notin C_0$. From Table 4, it can be checked that the best ranking is obtained by CHNN3D for *MAE* and *CCR*. The second methods are ABORC3 for *CCR* and SVOREX for *MAE*. Recall that ABORC3 is an ensemble of kernel models, while CHNN3D is a single model. SVOREX is known to be one of the most competitive methods for ordinal regression [6, 32], because it inherits the good properties of binary SVM. The statistical tests of Table 5 conclude that the differences are significant for all methods, except SVOREX and ABORC3. These results confirm that CHNN3D method is able to achieve a very competitive performance when compared to the state-of-the-art.

## 5. Discussion and Conclusion

This paper is motivated by the fact that the one dimensional projection of threshold models in ordinal regression can be too restrictive, resulting in too complex nonlinear models or unrealistic assumptions for linear models. We propose to relax this projection by extending the latent space and allowing multiple dimensions. The order of the classes is organised in this space by using concentric hyperspheres centred in the origin, in such a way that intermediate classes are bounded by consecutive hyperspheres.

In analogy with kernel methods, increased dimensionality of internal "feature space" representations allows one to use simpler, less complex functions to accomplish the task. In classification problems, one can always attempt to design potentially complex non-linear decision boundaries in the data space without the need to map to a higher dimensional feature space. However, the advantage of the feature space is that by (implicitly) representing important class structure properties in the higher dimensional space, usually much simpler (e.g. linear) functions are needed to accomplish the final task. These functions are easier to fit, the fit is more robust, and crucially, less prone to over-fitting.

By the very nature of binary classification, provided *linear* decision boundary is needed, we only need to specify its normal vector (and bias). In that case, the input points are eventually projected into a single dimension where the class membership is decided. The situation is different for ordinal regression. In one dimension, the class order can be naturally represented (as done in this paper) in a nested way, e.g. class $C_1$ would be represented by interval $[-a_1, a_1]$, class $C_2$ by $[-a_2, -a_1) \cup (a_1, a_2]$, class $C_3$ by $[-a_3, -a_2) \cup (a_2, a_3]$, etc. for some positive constants $a_1 < a_2 < \ldots < a_{Q-1}$.

Assume the input points $\mathbf{x}$ are projected onto the real line through function $\phi(\mathbf{x})$. By composing $\phi$ with a function $\psi(u) = |u|$, thus obtaining another projection function $\nu = \psi \circ \phi$, one can always transform the nested interval structure into the typical threshold structure used in ordinal regression, $[0, a_1]$, $(a_1, a_2]$, $(a_2, a_3]$, etc. However, if we did not fix $\psi$ a-priori and wanted to fit $\nu$ directly, the additional modelling burden would lead to more complex and hence more difficut to fit projection models.

We proposed to push this idea one step further: the nested interval structure can be naturally generalized to a series of nested $L$-dimensional hyper-spheres. Such structure still preserves the idea of class order, while eliminating the need to directly formulate and fit constrained 1-dimensional projections, as commonly done in ordinal regression. In our case we need

to learn a less restricted projection function $\phi = (f_1, f_2, \ldots, f_L)$ taking $\mathbf{x}$ into $\mathbb{R}^L$. This projection is then combined with a fixed function $\psi(u_1, u_2, \ldots, u_L) = \sqrt{u_1^2 + u_2^2 + \ldots + u_L^2}$ to form the final projection $\nu = \psi \circ \phi$. Crucially, increasing the latent space dimension from 1 to $L$ can allow us to put less strain on non-linear projection function $\phi$, taking us from the input space to the latent space that would be needed if we wanted to fit directly more complex projections to 1-dimensional structure of ordered intervals. In analogy with the feature space metaphor mentioned above, we suggest a "feature space" structure in our latent space that, at the price of increasing dimensions of the latent space (thus increasing its representational capacity), allows us to use a simple decision function $\psi$ on top of the latent space. In contrast with kernel methods, we do not fix the "feature mapping" $\phi$ while learning the decision function $\psi$ - instead, $\psi$ is fixed and $\phi$ is learnt.

The model is implemented by using a neural network approach, where each of these dimensions are set to a linear combination of basis functions (sigmoidal nodes, in our case). The proposed model shows better performance than a nominal neural network and a neural network based on the proportional odds model, and competitive performance when compared to the state-of-the-art in ordinal regression. A study of the number of dimensions of this extended latent space is also performed, where both two and three dimensional spaces seem to be a good option, at least for the 12 datasets considered. Finally, the projections learnt by the model are shown to be useful for studying additional characteristics of the dataset, acting as a visualisation tool.

As future research lines, the same latent space structure could be tested with other ordinal regression models, e.g. linear models or kernel models.

## References


[1] H. Dikkers, L. Rothkrantz, Support vector machines in ordinal classification: An application to corporate credit scoring, Neural Network World 15 (6) (2005) 491–507.

[2] M. J. Mathieson, Ordinal models for neural networks, in: J. M. A.-P. N. Refenes, Y. Abu-Mostafa, A. Weigend (Eds.), Proceedings of the Third International Conference on Neural Networks in the Capital Markets, Neural Networks in Financial Engineering, World Scientific, 1996, pp. 523–536.

[3] J. S. Cardoso, J. F. P. da Costa, M. Cardoso, Modelling ordinal relations with SVMs: an application to objective aesthetic evaluation of breast cancer conservative treatment, Neural Networks 18 (5-6) (2005) 808–817.

[4] M. Kim, V. Pavlovic, Structured output ordinal regression for dynamic facial emotion intensity prediction, in: K. Daniilidis, P. Maragos, N. Paragios (Eds.), Proceedings of the 11th European Conference on Computer Vision (ECCV 2010), Part III, Vol. 6313 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 649–662.

[5] J. Sánchez-Monedero, P. A. Gutiérrez, P. Tino, C. Hervás-Martínez, Exploitation of pairwise class distances for ordinal classification., Neural Comput. 25 (9) (2013) 2450–2485.

[6] W. Chu, S. S. Keerthi, Support Vector Ordinal Regression, Neural Comput. 19 (3) (2007) 792–815.

[7] W. Chu, Z. Ghahramani, Gaussian processes for ordinal regression, J. of Machine Learning Research 6 (2005) 1019–1041.

[8] B.-Y. Sun, J. Li, D. D. Wu, X.-M. Zhang, W.-B. Li, Kernel discriminant learning for ordinal regression, IEEE Trans. Knowl. Data Eng. 22 (6) (2010) 906–910.

[9] J. Verwaeren, W. Waegeman, B. De Baets, Learning partial ordinal class memberships with kernel-based proportional odds models, Computational Statistics & Data Analysis 56 (4) (2012) 928–942.

[10] P. McCullagh, Regression models for ordinal data, Journal of the Royal Statistical Society. Series B (Methodological) 42 (2) (1980) 109–142.

[11] E. Frank, M. Hall, A simple approach to ordinal classification, in: Proceedings of the 12th European Conference on Machine Learning, EMCL '01, Springer-Verlag, London, UK, 2001, pp. 145–156.

[12] J. S. Cardoso, J. F. P. da Costa, Learning to classify ordinal data: The data replication method, J. of Machine Learning Research 8 (2007) 1393–1429.

[13] L. Li, H.-T. Lin, Ordinal regression by extended binary classification, in: Advances in Neural Information Processing Systems, no. 19, 2007, pp. 865–872.

[14] H.-T. Lin, L. Li, Reduction from cost-sensitive ordinal ranking to weighted binary classification, Neural Comput. 24 (5) (2012) 1329–1367.

[15] E. L. Allwein, R. E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, J. of Machine Learning Research 1 (2001) 113–141.

[16] T. G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Int. Res. 2 (1) (1995) 263–286. URL `http://dl.acm.org/citation.cfm?id=1622826.1622834`

[17] M. J. Mathieson, Ordered classes and incomplete examples in classification, in: T. P. Michael C. Mozer, Michael I. Jordan (Ed.), Proceedings of the 1996 Conference on Neural Information Processing Systems (NIPS), Vol. 9 of Advances in Neural Information Processing Systems, 1999, pp. 550–556.

[18] M. Costa, Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities, Int. J. Neural Syst. 7 (5) (1996) 627–638.

[19] J. Cheng, Z. Wang, G. Pollastri, A neural network approach to ordinal regression, in: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN2008, IEEE World Congress on Computational Intelligence), IEEE Press, 2008, pp. 1279–1284.

[20] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst., Man, Cybern. B 42 (2) (2012) 513–529.

[21] W.-Y. Deng, Q.-H. Zheng, S. Lian, L. Chen, X. Wang, Ordinal extreme learning machine, Neurocomputing 74 (1–3) (2010) 447–456.

[22] A. Riccardi, F. Fernandez-Navarro, S. Carloni, Cost-sensitive adaboost algorithm for ordinal regression based on extreme learning machine, IEEE Transactions on Cybernetics In Press. doi:10.1109/TCYB.2014.2299291.

[23] F. Fernandez-Navarro, A. Riccardi, S. Carloni, Ordinal neural networks without iterative tuning, IEEE Trans. on Neural Networks and Learning Systems PP (99) (2014) 1–1. doi:10.1109/TNNLS.2014.2304976.

[24] J. F. P. da Costa, J. Cardoso, Classification of ordinal data using neural networks, in: J. Gama, R. Camacho, P. Brazdil, A. Jorge, L. Torgo (Eds.), Proceedings of the 16th European Conference on Machine Learning (ECML 2005), Vol. 3720 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 690–697.

[25] J. F. P. da Costa, H. Alonso, J. S. Cardoso, The unimodal model for the classification of ordinal data, Neural Networks 21 (2008) 78–91.

[26] M. Dobrska, H. Wang, W. Blackburn, Ordinal regression with continuous pairwise preferences, International Journal of Machine Learning and Cybernetics 3 (1) (2012) 59–70.

[27] A. Agresti, Categorical Data Analysis, 2nd Edition, John Wiley and Sons, 2002.

[28] C. Igel, M. Hüsken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (6) (2003) 105–123.

[29] A. Asuncion, D. Newman, UCI machine learning repository (2007). URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[30] M. Cruz-Ramírez, C. Hervás-Martínez, J. Sánchez-Monedero, P. A. Gutiérrez, Metrics to guide a multi-objective evolutionary algorithm for ordinal classification, Neurocomputing 135 (2014) 21–31. doi:10.1016/j.neucom.2013.05.058.

[31] S. Baccianella, A. Esuli, F. Sebastiani, Evaluation measures for ordinal regression, in: Proceedings of the Ninth International Conference on Intelligent Systems Design and Applications (ISDA'09), 2009, pp. 283–287.

[32] P. A. Gutiérrez, M. Pérez-Ortiz, F. Fernandez-Navarro, J. Sánchez-Monedero, C. Hervás-Martínez, An experimental study of different ordinal regression methods and measures, in: 7th International Conference

on Hybrid Artificial Intelligence Systems, 2012, pp. 296–307.

[33] A. Bueno-Crespo, P. J. García-Laencina, J.-L. Sancho-Gómez, Neural architecture design based on extreme learning machine, Neural Networks 48 (0) (2013) 19 – 24. doi:http://dx.doi.org/10.1016/j.neunet.2013.06.010.

[34] W. Xi-zhao, S. Qing-yan, M. Qing, Z. Jun-hai, Architecture selection for networks trained with extreme learning machine using localized generalization error model, Neurocomputing 102 (0) (2013) 3 – 9, advances in Extreme Learning Machines (ELM 2011). doi:http://dx.doi.org/10.1016/j.neucom.2011.12.053.

[35] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. of Machine Learning Research 7 (2006) 1–30.

[36] M. Friedman, A comparison of alternative tests of significance for the problem of *m* rankings, Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[37] J.-h. Zhai, H.-y. Xu, X.-z. Wang, Dynamic ensemble extreme learning machine based on sample entropy, Soft Computing 16 (9) (2012) 1493–1502. doi:10.1007/s00500-012-0824-6.