Journal of Machine Learning Research 8 (2007) 1393-1429 Submitted 2/06; Revised 9/06; Published 7/07

Learning to Classify Ordinal Data: The Data Replication Method

Jaime S. Cardoso JAIME.CARDOSO@INESCPORTO.PT
INESC Porto, Faculdade de Engenharia, Universidade do Porto
Campus da FEUP, Rua Dr. Roberto Frias, n 378
4200-465 Porto, Portugal

Joaquim F. Pinto da Costa JPCOSTA@FC.UP.PT
Faculdade Ciências Universidade Porto
Rua do Campo Alegre, 687
4169-007 Porto, Portugal

Editor: Ralf Herbrich

Abstract

Classification of ordinal data is one of the most important tasks of relation learning. This paper introduces a new machine learning paradigm specifically intended for classification problems where the classes have a natural order. The technique reduces the problem of classifying ordered classes to the standard two-class problem. The introduced method is then mapped into support vector machines and neural networks. Generalization bounds of the proposed ordinal classifier are also provided. An experimental study with artificial and real data sets, including an application to gene expression analysis, verifies the usefulness of the proposed approach.

Keywords: classification, ordinal data, support vector machines, neural networks

1. Introduction

Predictive learning has traditionally been a standard inductive learning, where different sub-problem formulations have been identified. One of the most representative is *classification*, consisting on the estimation of a mapping from the feature space into a finite class space. Depending on the cardinality of the finite class space we are left with binary or multiclass classification problems. Finally, the presence or absence of a “natural” order among classes will separate nominal from ordinal problems.

Although two-class and nominal data classification problems have been thoroughly analysed in the literature, the ordinal sibling has not received nearly as much attention yet. Nonetheless, many real life problems require the classification of items into naturally ordered classes. The scenarios involved range from information retrieval (Herbrich et al., 1999a) and collaborative filtering (Shashua and Levin, 2002) to econometric modeling (Mathieson, 1995) and medical sciences (Cardoso et al., 2005). It is worth pointing out that distinct tasks of relational learning, where an example is no longer associated with a class or rank, which include preference learning and reranking (Shen and Joshi, 2005), are topics of research on their own.

Conventional methods for nominal classes or for regression problems could be employed to solve ordinal data problems. However, the use of techniques designed specifically for ordered classes yields simpler and with better performance classifiers. Although the ordinal formulation seems conceptually simpler than the nominal one, difficulties to incorporate in the algorithms this piece of additional information—the order—may explain the widespread use of conventional methods to tackle the ordinal data problem.

This work addresses this void by introducing in Section 2 the data replication method, a nonparametric procedure for the classification of ordinal data. The underlying paradigm is the extension of the original data set with additional variables, reducing the classification task to the well known two-class problem. Starting with the simpler linear case, already established in Cardoso et al. (2005), the section develops the nonlinear case; from there the method is extended to incorporate the procedure of Frank and Hall (2001). Finally, the generic version of the data replication method is presented, allowing partial constraints on variables.

In section 3 the data replication method is instantiated in two important machine learning algorithms: support vector machines and neural networks. A comparison is made with a previous SVM approach introduced by Shashua and Levin (2002), the minimum margin principle, showing that the data replication method leads *essentially* to the same solution, but with some key advantages. The section is concluded with a reinterpretation of the neural network model as a generalization of the ordinal logistic regression model.

Section 4 describes the experimental methodology and the algorithms under comparison; results are reported and discussed in the succeeding sections. Finally, conclusions are drawn and future work is outlined in Section 8.

2. The Data Replication Method

Assume that examples in a classification problem come from one of K ordered classes, labeled from $\mathcal{C}_1$ to $\mathcal{C}_K$, corresponding to their natural order. Consider the training set $\{\mathbf{x}_i^{(k)}\}$, where $k = 1, \dots, K$ denotes the class number, $i = 1, \dots, \ell_k$ is the index within each class, and $\mathbf{x}_i^{(k)} \in \mathbb{R}^p$, with p the dimension of the feature space. Let $\ell = \sum_{k=1}^{K} \ell_k$ be the total number of training examples.

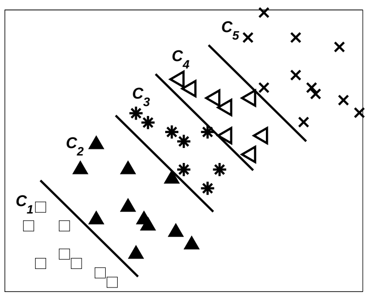
Suppose that a K-class classifier was forced, by design, to have $(K-1)$ *nonintersecting boundaries*, with boundary i discriminating classes $\mathcal{C}_1, \dots, \mathcal{C}_i$ against classes $\mathcal{C}_{i+1}, \dots, \mathcal{C}_K$. As the intersection point of two boundaries would indicate an example with three or more classes equally probable—not plausible with ordinal classes—this strategy imposes a sensible restriction. With this constraint emerges a monotonic model, where a better value in an attribute does not lead to a lower decision class. For the linear case, this translates into choosing the same weighted sum for all decisions—the classifier would be just a set of weights, one for each feature, and a set of thresholds, the *scale* in the weighted sum. By avoiding the intersection of any two boundaries, this model tries to capture the essence of the ordinal data problem. Additionally, we foresee a better generalization performance due to the reduced number of parameters to be estimated.

This rationale leads to a straightforward generalization of the two-class separating hyperplane (Shashua and Levin, 2002). Define $(K-1)$ hyperplanes that separate the training data into K ordered classes by modeling the ranks as intervals on the real line. The geometric interpretation of this approach is to look for $(K-1)$ parallel hyperplanes, represented by vector $\mathbf{w} \in \mathbb{R}^p$ and scalars $b_1, \dots, b_{K-1}$, such that the feature space is divided into K regions by the decision boundaries $\mathbf{w}^t\mathbf{x} + b_r = 0,\ r = 1, \dots, K-1$.

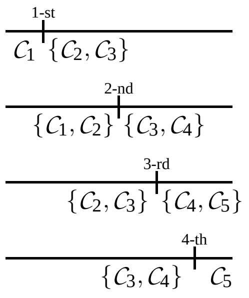
Given that there are many good two-class learning algorithms, it is tempting to reduce this ordinal formulation to a two-class problem. The data replication method allows us to do precisely that.

2.1 Data Replication Method—the Linear Case[1]

Before moving to the formal presentation of the data replication method, it is instructive to motivate the method by considering a hypothetical, simplified scenario, with five classes in $\mathbb{R}^2$. The plot of the data set is presented in Figure 1(a); superimposed is also depicted a reasonable set of four parallel hyperplanes separating the five classes.



(a) Plot of the data points. Also shown are reasonable class boundaries.


1-st
$\mathcal{C}_1$ $\{\mathcal{C}_2, \mathcal{C}_3\}$
2-nd
$\{\mathcal{C}_1, \mathcal{C}_2\}$ $\{\mathcal{C}_3, \mathcal{C}_4\}$
3-rd
$\{\mathcal{C}_2, \mathcal{C}_3\}$ $\{\mathcal{C}_4, \mathcal{C}_5\}$
4-th
$\{\mathcal{C}_3, \mathcal{C}_4\}$ $\mathcal{C}_5$

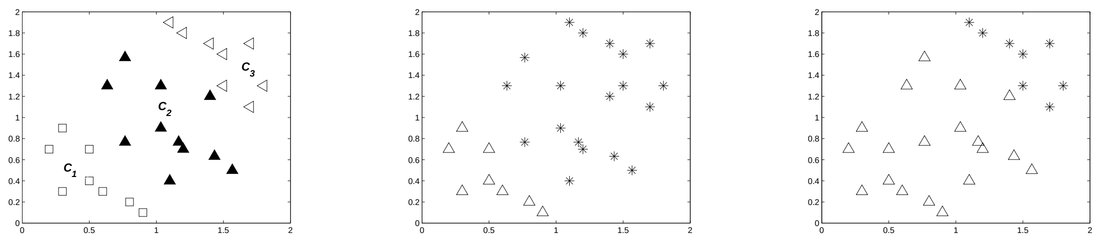
(b) Classes involved in the hyperplanes definition, for $K = 5$, $s = 2$.

Figure 1: Toy model with 5 classes in $\mathbb{R}^2$.

The i-th hyperplane discriminates classes $\mathcal{C}_1, \ldots, \mathcal{C}_i$ against classes $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_K$. Due to the order among classes, the i-th hyperplane is essentially determined by classes $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$. More generally, it can be said that the i-th hyperplane is determined by s classes to its 'left' and s classes to its 'right', with $1 \leq s \leq K-1$. Naturally, for some of the hyperplanes, there will be less than s classes to one (or both) of its sides. The i-th hyperplane has i classes on its left and $(K-i)$ on its right. Hence, for a chosen s value, we may say that classifier i depends on $\min(s,i)$ classes on its left—classes $\mathcal{C}_k, k = \max(i-s+1,1), \ldots, i$—and depends on $\min(s, K-i)$ classes on its right—classes $\mathcal{C}_k, k = i+1, \ldots, \min(i+1+s-1, i+1+K-i-1) = i+1, \ldots, \min(i+s, K)$. The classes involved in each hyperplane definition, for the five class data set with $s = 2$, are illustrated in Figure 1(b).

To start the presentation of the data replication method let us consider an even more simplified toy example with just three classes, as depicted in Figure 2(a). Here, the task is to find two parallel hyperplanes, the first one discriminating class $\mathcal{C}_1$ against classes $\{\mathcal{C}_2, \mathcal{C}_3\}$ (we are considering $s = K-1 = 2$ for the explanation) and the second discriminating classes $\{\mathcal{C}_1, \mathcal{C}_2\}$ against class $\mathcal{C}_3$. These hyperplanes will correspond to the solution of two binary classification problems but with the additional constraint of parallelism—see Figure 2. The data replication method suggests solving both problems simultaneously in an augmented feature space.

1. The linear version of the data replication method was already presented in Cardoso et al. (2005); here we provide, arguably, a cleaner presentation.



(a) Original data set in $\mathbb{R}^2$, $K = 3$.

(b) Binary problem $\mathcal{C}_1$ against classes $\{\mathcal{C}_2, \mathcal{C}_3\}$.

(c) Binary problem $\{\mathcal{C}_1, \mathcal{C}_2\}$ against class $\mathcal{C}_3$.

Figure 2: Binary problems to be solved simultaneously with the data replication method.

Using a transformation from the $\mathbb{R}^2$ initial feature-space to a $\mathbb{R}^3$ feature space, replicate each original point, according to the rule (see Figure 3(b)):

$$\begin{tikzcd} & \left[\begin{smallmatrix}\mathbf{x}\\ h\end{smallmatrix}\right] \in \mathbb{R}^3 \\ \mathbf{x} \in \mathbb{R}^2 \arrow[ur] \arrow[dr] & & , \text{ where } h = \text{const} \in \mathbb{R}^+. \\ & \left[\begin{smallmatrix}\mathbf{x}\\ 0\end{smallmatrix}\right] \in \mathbb{R}^3 \end{tikzcd}$$

Observe that any two points created from the same original point differ only in the new variable. Define now a binary training set in the new (higher dimensional) space according to (see Figure 3(c)):

$$\begin{bmatrix}\mathbf{x}_i^{(1)}\\ 0\end{bmatrix} \in \overline{\mathcal{C}}_1, \begin{bmatrix}\mathbf{x}_i^{(2)}\\ 0\end{bmatrix}, \begin{bmatrix}\mathbf{x}_i^{(3)}\\ 0\end{bmatrix} \in \overline{\mathcal{C}}_2 \qquad ; \begin{bmatrix}\mathbf{x}_i^{(1)}\\ h\end{bmatrix}, \begin{bmatrix}\mathbf{x}_i^{(2)}\\ h\end{bmatrix} \in \overline{\mathcal{C}}_1, \begin{bmatrix}\mathbf{x}_i^{(3)}\\ h\end{bmatrix} \in \overline{\mathcal{C}}_2\ . \qquad (1)$$

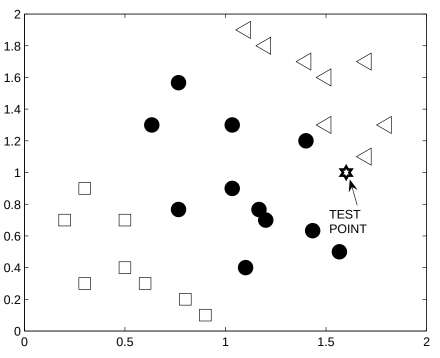
In this step we are defining the two binary problems as a single binary problem in the augmented feature space. A linear two-class classifier can now be applied on the extended data set, yielding a hyperplane separating the two classes, see Figure 3(d). The intersection of this hyperplane with each of the subspace replicas can be used to derive the boundaries in the original data set, as illustrated in Figure 3(e).

Although the foregoing analysis enables one to classify unseen examples in the original data set, classification can be done directly in the extended data set, using the binary classifier, without explicitly resorting to the original data set. For a given example $\in \mathbb{R}^2$, classify each of its two replicas $\in \mathbb{R}^3$, obtaining a sequence of two labels $\in \{\overline{\mathcal{C}}_1, \overline{\mathcal{C}}_2\}^2$. From this sequence infer the class according to the rule

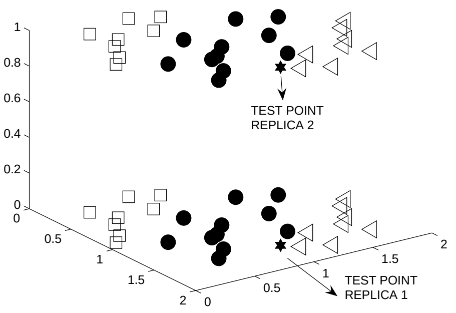
$$\overline{\mathcal{C}}_1\overline{\mathcal{C}}_1 \Longrightarrow \mathcal{C}_1, \qquad \overline{\mathcal{C}}_2\overline{\mathcal{C}}_1 \Longrightarrow \mathcal{C}_2, \qquad \overline{\mathcal{C}}_2\overline{\mathcal{C}}_2 \Longrightarrow \mathcal{C}_3.$$

To exemplify, from the test point in Figure 3(a), create two replicas and classify them in $\mathbb{R}^3$ (highlighted in Figure 3(d)). The first replica is classified as $\overline{\mathcal{C}}_2$ and the second as $\overline{\mathcal{C}}_1$. Hence, the class predicted for the test point is $\mathcal{C}_2$. This same class could have been obtained in the original feature space, as represented in Figure 3(e).

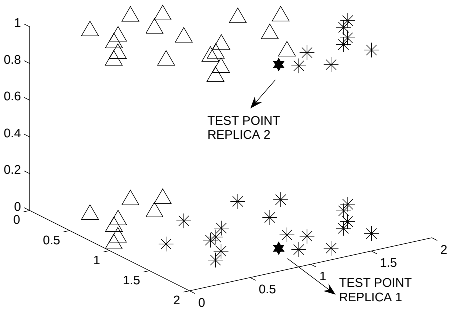
It is clear now that the principle behind the replication method is to have a replica of the original data set for each boundary. The replica i is binarized to discriminate classes $\mathcal{C}_1, \dots, \mathcal{C}_i$ against classes



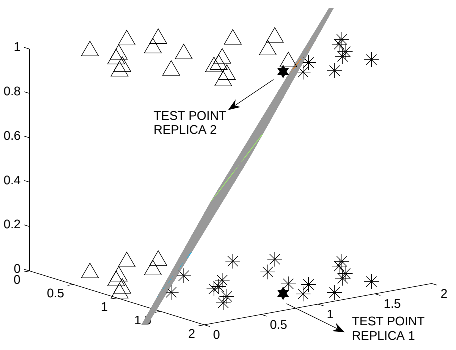
(a) Original data set in $\mathbb{R}^2$, $K = 3$.



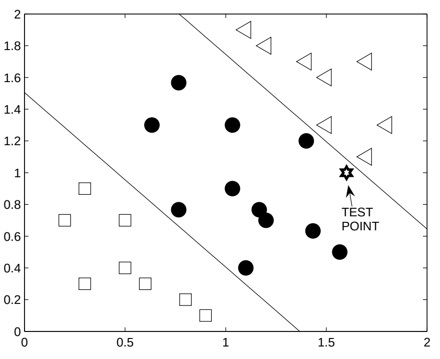
(b) Data set in $\mathbb{R}^3$, with samples replicated ($h = 1$).



(c) Transformation into a binary classification problem.



(d) Linear solution to the binary problem.



(e) Linear solution in the original data set.

Figure 3: Proposed data extension model in a toy example.

$\mathcal{C}_{i+1}, \ldots, \mathcal{C}_K$ (more generally, when parameterized by s, discriminating classes $\mathcal{C}_k, k = \max(i-s+1, 1), \ldots, i$ against classes $\mathcal{C}_k, k = i+1, \ldots, \min(i+s, K)$). The additional variables, in number of $(K-2)$, provide just the right amount of flexibility needed for having boundaries with the same direction but with different thresholds.

After the above exposition on a toy model, we will now formally describe a general K-class classifier for ordinal data classification. Define $\mathbf{e}_0$ as the sequence of $(K-2)$ zeros and $\mathbf{e}_q$ as the sequence of $(K-2)$ symbols $0, \ldots, 0, h, 0, \ldots, 0$, with $h > 0$ in the q-th position. Considering the problem of separating K ordered classes $\mathcal{C}_1, \ldots, \mathcal{C}_K$ with training set $\{\mathbf{x}_i^{(k)}\}$, define a new binary training data set in $\mathbb{R}^{p+K-2}$ as

$$\begin{array}{ll}
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{e}_0 \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = 1, \\ \overline{\mathcal{C}}_2 & k = 2, \ldots, \min(K, 1+s), \end{cases} \\
 & \vdots \\
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{e}_{q-1} \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, q-s+1), \ldots, q, \\ \overline{\mathcal{C}}_2 & k = q+1, \ldots, \min(K, q+s), \end{cases} \\
 & \vdots \\
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{e}_{K-2} \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, K-1-s+1), \ldots, K-1, \\ \overline{\mathcal{C}}_2 & k = K, \end{cases}
\end{array} \tag{2}$$

where parameter $s \in \{1, \ldots, K-1\}$ plays the role of bounding the number of classes defining each hyperplane. This setup allows controlling the increase of data points inherent to this method. The toy example in Figure 3(b) was illustrated with $s = K-1 = 2$; by setting $s = 1$ one would obtain the extended data set illustrated in Figure 4.

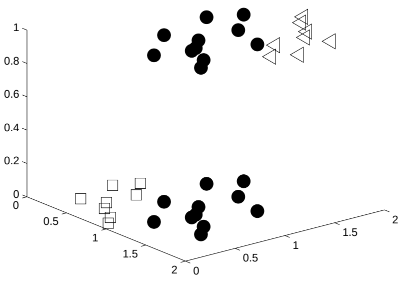


Figure 4: Toy data set replicated in $\mathbb{R}^3$, $h = 1$, $s = 1$.

Next, build a linear two-class classifier on this extended data set; to predict the class of an unseen example, obtain a sequence of $(K-1)$ labels $\in \{\overline{\mathcal{C}}_1, \overline{\mathcal{C}}_2\}^{(K-1)}$ by classifying each of the $(K-1)$ replicas in the extended data set with the binary classifier. Note that to make a prediction, all $(K-1)$ replicas of the test point are always classified, even if the classifier was trained with $s < K-1$. It is also worth noticing that the binary decision $\overline{\mathbf{w}}^t\overline{\mathbf{x}} + b = 0$, with $\overline{\mathbf{w}}, \overline{\mathbf{x}} \in \mathbb{R}^{p+K-2}$ when mapped into the original space gives rise to the $(K-1)$ boundaries, in the form $\mathbf{w}^t\mathbf{x}+b_i$, with

$$b_i = \begin{cases} b & \text{if } i=1, \\ hw_{p+i-1}+b & \text{if } i>1. \end{cases}$$

Now, what possible sequences can one obtain?

Assume for now that the thresholds are correctly ordered as $-b_1 \leq -b_2 \leq \ldots \leq -b_{K-1}$. Or, equivalently, that $0 \geq hw_{p+1} \geq hw_{p+2} \ldots \geq hw_{p+K-2}$. If the replica $\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_i\end{smallmatrix}\right]$ of a test point $\mathbf{x}$ is predicted as $\overline{\mathcal{C}}_1$, that is because $\bar{\mathbf{w}}^t\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_i\end{smallmatrix}\right]+b<0$. But then also the replica $\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_{i+1}\end{smallmatrix}\right]$ is predicted as $\overline{\mathcal{C}}_1$ because $0 \geq hw_{p+i} \geq hw_{p+i+1}$. One can conclude that the only K possible sequences and the corresponding predicted classes are

$$\begin{array}{ll} \overline{\mathcal{C}}_1,\overline{\mathcal{C}}_1,\ldots,\overline{\mathcal{C}}_1,\overline{\mathcal{C}}_1 & \Longrightarrow \mathcal{C}_1, \\ \overline{\mathcal{C}}_2,\overline{\mathcal{C}}_1,\ldots,\overline{\mathcal{C}}_1,\overline{\mathcal{C}}_1 & \Longrightarrow \mathcal{C}_2, \\ \vdots & \\ \overline{\mathcal{C}}_2,\overline{\mathcal{C}}_2,\ldots,\overline{\mathcal{C}}_2,\overline{\mathcal{C}}_1 & \Longrightarrow \mathcal{C}_{K-1}, \\ \overline{\mathcal{C}}_2,\overline{\mathcal{C}}_2,\ldots,\overline{\mathcal{C}}_2,\overline{\mathcal{C}}_2 & \Longrightarrow \mathcal{C}_K. \end{array}$$

Henceforth, the target class can be obtained by adding one to the number of $\overline{\mathcal{C}}_2$ labels in the sequence. To emphasize, the process is depicted in Figure 5 for a data set in $\mathbb{R}$ with four classes.

The reduction technique presented here uses a binary classifier to make multiclass ordinal predictions. Instead of resorting to multiple binary classifiers to make predictions in the ordinal problem (as is common in reduction techniques from multiclass to binary problems), the data replication method uses a single binary classifier to classify $(K-1)$ *dependent* replicas of a test point. A pertinent question is how the performance of the binary classifier translates into the performance on the ordinal problem. In Appendix A, a bound on the generalization error of the ordinal data classifier is expressed as a function of the error of the binary classifier.

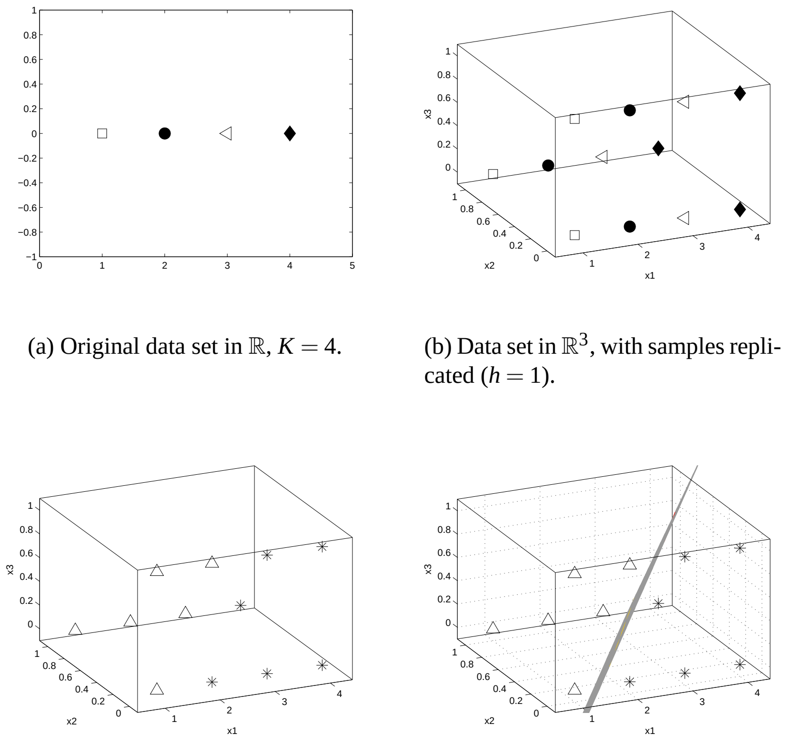
The thresholds b_i were assumed correctly ordered. It is not trivial to see how to keep them well ordered with this standard data replication method, for s general. We present next an alternative method of replicating the data, in which constraints on the thresholds are explicitly incorporated in the form of extra points added to the training set. This formulation enforces ordered boundaries for s values as low as 1, although compromising some of cleanliness in the interpretation as a binary classification problem in the extended space.

2.2 Homogeneous Data Replication Method

With the data replication method just presented, the boundary in the extended space $\bar{\mathbf{w}}^t\bar{\mathbf{x}}+b=0$ has correspondence in the original space to the $(K-1)$ boundaries $\mathbf{w}^t\mathbf{x}+b_i$, with $b_1=b$, $b_i = hw_{p+i-1}+b_1, i=2,\ldots,K-1$. It is notorious the asymmetry with respect to b_1.

One could attain a symmetric formulation by introducing the well-known homogenous coordinates. That would lead to the addition of a new variable and to the restriction of linear boundaries going through the origin. The same result can be obtained by starting with a slightly different extension of the data set.

Define $\mathbf{u}_q$ as the sequence of $(K-1)$ symbols $0,\ldots,0,h,0,\ldots,0$, with $h>0$ in the q-th position. Considering the problem of separating K ordered classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ with training set $\{\mathbf{x}_i^{(k)}\}$, define a new binary training data set in $\mathbb{R}^{p+K-1}$ as



(a) Original data set in $\mathbb{R}$, $K = 4$.

(b) Data set in $\mathbb{R}^3$, with samples replicated ($h = 1$).

(c) Transformation into a binary classification problem.

(d) Linear solution to the binary problem.

Figure 5: Proposed data extension model for a data set in $\mathbb{R}$, with $K = 4$.

$$\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{u}_1 \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k = 1, \\ \overline{\mathcal{C}}_2 & k = 2, \dots, \min(K, 1+s), \end{cases}$$

$$\vdots$$

$$\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{u}_q \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, q-s+1), \dots, q, \\ \overline{\mathcal{C}}_2 & k = q+1, \dots, \min(K, q+s), \end{cases}$$

$$\vdots$$

$$\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{u}_{K-1} \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, K-1-s+1), \dots, K-1, \\ \overline{\mathcal{C}}_2 & k = K. \end{cases}$$

Note that the homogeneous extended data set has dimension $p+K-1$, as opposed to $(p+K-2)$ in the standard formulation of the data replication method. It also comes that $b_i = h w_{p+i},\ i = 1, \dots, K-1$. Under the homogeneous approach, one has to look for a linear homogeneous boundary of the form $\overline{\mathbf{w}}^t \overline{\mathbf{x}} = 0$, as the bias of the boundary is incorporated as a new coordinate.

The main reason for preferring the standard over the homogeneous formulation of the data replication method is that most of the existing linear binary classifiers are formulated in terms of non-homogeneous boundaries having the form $\bar{\mathbf{w}}^t\bar{\mathbf{x}}+b=0$, instead of $\bar{\mathbf{w}}^t\bar{\mathbf{x}}=0$. Therefore, some adaptation is required before applying existing linear binary classifiers to the homogeneous data replication method.

Homogeneous Data Replication Method with Explicit Constrains on the Thresholds

Unless one sets $s=K-1$, the data replication method does not enforce ordered thresholds (we will return to this point later, when mapping to SVMs and neural networks). This is true for both the standard and the homogeneous formulations. Explicit constraints in the model's formulation can be introduced to enforce the correct order. With the homogeneous formulation, those explicit constraints can take the form of additional $(K-2)$ points in the training set.

Consider the relation $-b_i < -b_{i+1}$. This relation can be equivalently written as

$$-hw_{p+i} < -hw_{p+i+1} \Longleftrightarrow -\bar{\mathbf{w}}^t \begin{bmatrix} \mathbf{0}_p \\ \mathbf{u}_i \end{bmatrix} < -\bar{\mathbf{w}}^t \begin{bmatrix} \mathbf{0}_p \\ \mathbf{u}_{i+1} \end{bmatrix} \Longleftrightarrow \bar{\mathbf{w}}^t \begin{bmatrix} \mathbf{0}_p \\ \mathbf{u}_{i+1}-\mathbf{u}_i \end{bmatrix} < 0.$$

As a result, constraining $-b_i$ to be less than $-b_{i+1}$ is equivalent to correctly classify the point $\begin{bmatrix} \mathbf{0}_p \\ \mathbf{u}_{i+1}-\mathbf{u}_i \end{bmatrix}$ in the $\mathcal{C}_1$ class. It is interesting to note that this point is not in the subspace of any of the data replicas. To introduce the $(K-2)$ explicit constraints on the thresholds, just enforce that the $(K-2)$ points $\begin{bmatrix} \mathbf{0}_p \\ \mathbf{u}_{i+1}-\mathbf{u}_i \end{bmatrix}, i=1,\ldots,K-2$ are correctly classified in $\mathcal{C}_1$. Note that violations of these constraints can not be allowed.

2.3 Data Replication Method—the Nonlinear Case

Previously, the data replication method was considered as a design methodology of a linear classifier for ordinal data. This section addresses scenarios where data is not linearly separable and for which the design of a linear classifier does not lead to satisfactory results. Therefore, the design of nonlinear classifiers emerges as a necessity. The only constraint to enforce during the design process is that boundaries should not intersect.

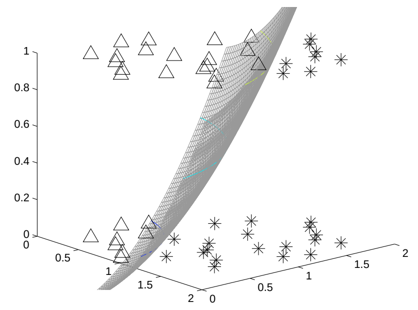
Inspired by the data replication method just presented, we now look for generic boundaries that are *level curves* of some nonlinear, real-valued function $G(\mathbf{x})$ defined in the feature space. The $(K-1)$ boundaries are defined as $G(\mathbf{x})=b_i, i=1,\ldots,K-1, b_i\in\mathbb{R}$. It is worth emphasizing that interpreting the decision boundaries as level curves of some (unknown) function does not result in loss of generality. For the linear version one take $G(\mathbf{x})=\mathbf{w}^t\mathbf{x}+b$.

Once again, the search for nonintersecting, nonlinear boundaries can be carried out in the extended space of the data replication method. First, extend and modify the feature space to a binary problem, as dictated by the data replication method. Next, search for a boundary $\overline{G}(\bar{\mathbf{x}})$ defined in the extended space that results on $(K-1)$ boundaries $G(\mathbf{x})=b_i$ when reverted to the original space. The simplest form for $\overline{G}(\bar{\mathbf{x}})$ is as a partially linear (nonlinear in the original variables but linear in the introduced variables) boundary $\overline{G}(\bar{\mathbf{x}})=G(\mathbf{x})+\underline{\mathbf{w}}^t\mathbf{e}_i=0$, with $\underline{\mathbf{w}}\in\mathbb{R}^{K-2}$, and $\bar{\mathbf{x}}=\left[\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_i\end{smallmatrix}\right]$. Notice that restricting the function $\overline{G}(\bar{\mathbf{x}})$ to be linear in the $(K-2)$ added variables imposes automatically nonintersecting boundaries in the original space: boundaries will have the form $G(\mathbf{x})+b_i=0$ with

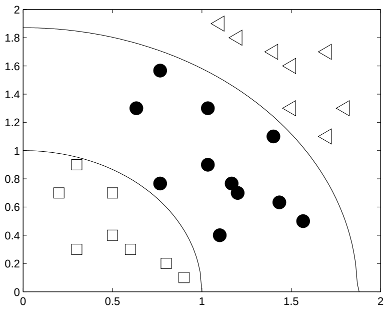
$$b_i = \begin{cases} 0 & \text{if } i=1, \\ hw_{p+i-1} & \text{if } 2\le i\le K-1. \end{cases}$$

Although a partially linear function $\overline{G}(\overline{\mathbf{x}})$ is the simplest to provide nonintersecting boundaries in the original space (level curves of some function $G(\mathbf{x})$), it is by no means the only type of function to provide them.

The intersection of the constructed high-dimensional boundary with each of the subspace replicas provides the desired $(K-1)$ boundaries. This approach is plotted in Figure 6 for the toy example. The $\#\overline{\mathcal{C}}_2+1$ rule can still be applied to predict the class of a test example directly in the extended feature space.



(a) Nonlinear solution to the binary problem. $\overline{G}(\overline{\mathbf{x}}) = 0.4(x_1^2+x_2^2-1)+x_3$



(b) Nonlinear solution in the original data set. $G(\mathbf{x}) = x_1^2+x_2^2-1$

Figure 6: Nonlinear data extension model in the toy example.

The nonlinear extension of the homogeneous data replication method follows the same rationale as the standard formulation. Now the $\overline{G}(\overline{\mathbf{x}}) = G(\mathbf{x})+\underline{\mathbf{w}}^t\mathbf{u}_i = 0$ boundary, with $\underline{\mathbf{w}} \in \mathbb{R}^{K-1}$, must be constrained such that $G(\mathbf{0}) = 0 \Longleftrightarrow \overline{G}(\overline{\mathbf{0}}) = 0$. Finally, the enforcement of ordered thresholds with the introduction of additional training points is still valid in the nonlinear case, as

$$\overline{G}\left(\begin{bmatrix}\mathbf{0}_p\\ \mathbf{u}_{i+1}-\mathbf{u}_i\end{bmatrix}\right) = G(\mathbf{0}_p)+\underline{\mathbf{w}}^t(\mathbf{u}_{i+1}-\mathbf{u}_i) = hw_{p+i+1}-hw_{p+i} = b_{i+1}-b_i.$$

2.4 A General Framework

As presented so far, the data replication method allows only searching for parallel hyperplanes (*level curves* in the nonlinear case) boundaries. That is, a single direction is specified for all boundaries. In the quest for an extension allowing more loosely coupled boundaries, let us start by reviewing the method for ordinal data by Frank and Hall (2001).

2.4.1 THE METHOD OF FRANK AND HALL

Frank and Hall (2001) proposed to use $(K-1)$ standard binary classifiers to address the K-class ordinal data problem. Toward that end, the training of the i-th classifier is performed by converting the ordinal data set with classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ into a binary data set, discriminating $\mathcal{C}_1,\ldots,\mathcal{C}_i$ against $\mathcal{C}_{i+1},\ldots,\mathcal{C}_K$. To predict the class value of an unseen instance, the $(K-1)$ outputs are combined to produce a single estimation. If the i-th classifier predicts $\mathcal{C}_X > \mathcal{C}_i$ with probability p_i, Frank and

Hall (2001) suggest to estimate the probability values of each of the K classes as

$$\begin{aligned} p_{\mathcal{C}_1} &= 1 - p_1, \\ p_{\mathcal{C}_j} &= p_{j-1} - p_j \quad j = 2, \cdots, K-1, \\ p_{\mathcal{C}_K} &= p_{K-1}. \end{aligned}$$

Note however that this approach may lead to negative estimates of probability values. A solution to that problem is to identify the output p_i of the i-th classifier with the conditional probability $p(\mathcal{C}_X > \mathcal{C}_i \,|\, \mathcal{C}_X > \mathcal{C}_{i-1})$. This meaning can be exploited to rank the classes according to the following formulas:

$$\begin{aligned} p(\mathcal{C}_X > \mathcal{C}_1) &= p_1, & p_{\mathcal{C}_1} &= 1 - p_1, \\ p(\mathcal{C}_X > \mathcal{C}_j) &= p_j\, p(\mathcal{C}_X > \mathcal{C}_{j-1}), & p_{\mathcal{C}_j} &= (1 - p_j)\, p(\mathcal{C}_X > \mathcal{C}_{j-1}) \quad j = 2, \cdots, K-1, \\ & & p_{\mathcal{C}_K} &= p(\mathcal{C}_X > \mathcal{C}_{K-1}). \end{aligned}$$

Any binary classifier can be used as the building block of this scheme. Observe that, under our approach, the i-th boundary is also discriminating $\mathcal{C}_1, \ldots, \mathcal{C}_i$ against $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_K$; the major difference lies in the *independence* of the boundaries found with Frank and Hall's method. This independence is likely to lead to intersecting boundaries.

2.4.2 A PARAMETERIZED FAMILY OF CLASSIFIERS

Thus far, nonintersecting boundaries have been motivated as the best way to capture ordinal relation among classes. That may be a too restrictive condition for problems where some features are not in relation with the ordinal property. Suppose then that the order of the classes is not totally reflected in a subset of the features. Without further information, it is unadvised to draw from them any ordinal information. It may be more advantageous to restrict the enforcing of the nonintersecting boundaries to the leftover features.

We suggest a generalization of the data replication method where the enforcement of nonintersecting boundaries is restricted only to the first j features, while the last $p - j$ features enjoy the independence as materialized in the Frank and Hall's method. Towards that end we start by showing how the independent boundaries approach of Frank and Hall can be subsumed in the data replication framework.

Instead of replicating the original train data set as expressed by Eq. (2), we indent to arrive at a strategy that still allows a single binary classifier to solve the $(K-1)$ classification problems simultaneously, but yielding independent boundaries. If the boundaries are expected to be independent, each of the $(K-1)$ data replicas of the original data should be made as 'independent' as possible.

Up to this point, when replicating the original data set, the original p variables were the first p variables of the $p + K - 2$ variables of the augmented data set, for each subspace replica, as seen in Eq. (2). Each of the $(K-2)$ extra variables accounts for a different threshold term, while the fact that the original variables are 'shared' among the different replicas results in a common direction for all of the boundaries. The argument is that if the set of the original p features is mapped into a different set of p variables for each data replica, while keeping the $(K-2)$ extra variables to account for different thresholds, the binary classifier will return (almost) independent boundaries. It is worth noticing that this procedure increases the number of variables in the extended space to $(K-1) \times p + (K-2)$.

Returning to the toy example, assume that the replication was done *not* according to Eq. (1) but instead using the following rule:

$$\begin{bmatrix} \mathbf{x}_i^{(1)} \\ \mathbf{0}_2 \\ 0 \end{bmatrix} \in \overline{\mathcal{C}}_1,\ \begin{bmatrix} \mathbf{x}_i^{(2)} \\ \mathbf{0}_2 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{x}_i^{(3)} \\ \mathbf{0}_2 \\ 0 \end{bmatrix} \in \overline{\mathcal{C}}_2, \qquad \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_i^{(1)} \\ h \end{bmatrix}, \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_i^{(2)} \\ h \end{bmatrix} \in \overline{\mathcal{C}}_1,\ \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_i^{(3)} \\ h \end{bmatrix} \in \overline{\mathcal{C}}_2$$

where $\mathbf{0}_2$ is the sequence of 2 zeros. Intuitively, by misaligning variables involved in the determination of different boundaries (variables in different subspaces), we are decoupling those same boundaries.

Proceeding this way, boundaries can be designed almost independently (the mapping on SVMs will clarify this issue). In the linear case we have now four parameters to estimate, the same as for two independent lines in $\mathbb{R}^2$. Intuitively, this new rule to replicate the data allows the estimation of the direction of each boundary in *essentially* an independent way.

The general formulation in Eq. (2) becomes

$$\begin{array}{ll}
\begin{bmatrix} \mathbf{x}_i^{(k)} \\ \mathbf{0}_{p(K-2)} \\ \mathbf{e}_0 \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = 1, \\ \overline{\mathcal{C}}_2 & k = 2, \dots, \min(K, 1+s), \end{cases} \\
 & \vdots \\
\begin{bmatrix} \mathbf{0}_{p(q-1)} \\ \mathbf{x}_i^{(k)} \\ \mathbf{0}_{p(K-q-1)} \\ \mathbf{e}_{q-1} \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, q-s+1), \dots, q, \\ \overline{\mathcal{C}}_2 & k = q+1, \dots, \min(K, q+s), \end{cases} \\
 & \vdots \\
\begin{bmatrix} \mathbf{0}_{p(K-2)} \\ \mathbf{x}_i^{(k)} \\ \mathbf{e}_{K-2} \end{bmatrix} & \in \begin{cases} \overline{\mathcal{C}}_1 & k = \max(1, K-1-s+1), \dots, K-1, \\ \overline{\mathcal{C}}_2 & k = K, \end{cases}
\end{array} \tag{3}$$

where $\mathbf{0}_*$ is the sequence of $*$ zeros.

While the basic linear data replication method requires the estimation of $(p-1)+(K-1)$ parameters, the new rule necessitates of $(p-1)(K-1)+(K-1) = p(K-1)$, the same as the Frank and Hall approach; this corresponds to the number of free parameters in $(K-1)$ independent p-dimensional hyperplanes. While this does not aim at being a practical alternative to Frank's method, it does pave the way for intermediate solutions, filling the gap between the totally coupled and totally independent boundaries.

To constrain only the first j variables of the p initial variables to have the same direction in all boundaries, while leaving the $(p-j)$ final variables unconstrained, we propose to extend the data according to

$$\begin{bmatrix} \mathbf{x}_i^{(k)}(1{:}j) \\ \mathbf{x}_i^{(k)}(j{+}1{:}p) \\ \mathbf{0}_{(p-j)(K-2)} \\ \mathbf{e}_0 \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k=1, \\ \overline{\mathcal{C}}_2 & k=2,\ldots,\min(K,1+s), \end{cases}$$

$$\vdots$$

$$\begin{bmatrix} \mathbf{x}_i^{(k)}(1{:}j) \\ \mathbf{0}_{(p-j)(q-1)} \\ \mathbf{x}_i^{(k)}(j{+}1{:}p) \\ \mathbf{0}_{(p-j)(K-q-1)} \\ \mathbf{e}_{q-1} \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k=\max(1,q-s+1),\ldots,q, \\ \overline{\mathcal{C}}_2 & k=q+1,\ldots,\min(K,q+s), \end{cases}$$

$$\vdots$$

$$\begin{bmatrix} \mathbf{x}_i^{(k)}(1{:}j) \\ \mathbf{0}_{(p-j)(K-2)} \\ \mathbf{x}_i^{(k)}(j{+}1{:}p) \\ \mathbf{e}_{K-2} \end{bmatrix} \in \begin{cases} \overline{\mathcal{C}}_1 & k=\max(1,K-1-s+1),\ldots,K-1, \\ \overline{\mathcal{C}}_2 & k=K. \end{cases}$$

With this rule $[p-1-(j-1)](K-1)+(K-1)+j-1,\ j\in\ \{1,\ldots,p\}$, parameters are to be estimated.

This general formulation of the data replication method allows the enforcement of only the amount of knowledge (constraints) that is effectively known *a priori*, building the right amount of parsimony into the model (see the pasture production experiment).

Now, in this general setting, we can no longer assume nonintersecting boundaries. Therefore, the space of features may be partitioned in more than K regions. To predict the class of an unseen instance we may estimate the probabilities of the K classes using the $(K-1)$ replicas, similarly to Frank and Hall (2001), or simply keep the $\#\overline{\mathcal{C}}_2+1$ rule.

3. Mapping the Data Replication Method to Learning Algorithms

In this section the data replication method just introduced is instantiated in two important machine learning algorithms: support vector machines and neural networks.

3.1 Mapping the Data Replication Method to SVMs

The learning task in a classification problem is to select a prediction function $f(\mathbf{x})$ from a family of possible functions that minimizes the expected *loss*.

In the absence of reliable information on relative costs, a natural approach for unordered classes is to treat every misclassification as equally likely. This translates into adopting the non-metric indicator function $l_{0-1}(f(\mathbf{x}),y)=0$ if $f(\mathbf{x})=y$ and $l_{0-1}(f(\mathbf{x}),y)=1$ if $f(\mathbf{x})\neq y$, where $f(\mathbf{x})$ and y are the predicted and true classes, respectively. Measuring the performance of a classifier using the l_{0-1} loss function is equivalent to simply considering the misclassification error rate. However, for ordered classes, losses that increase with the absolute difference between the class numbers are more natural choices in the absence of better information (Mathieson, 1995). This loss should be naturally incorporated during the training period of the learning algorithm.

A risk functional that takes into account the ordering of the classes can be defined as

$$R(f)=\mathbf{E}\left[l^s\left(f(\mathbf{x}^{(k)}),k\right)\right] \tag{4}$$

with

$$l^s\left(f(\mathbf{x}^{(k)}),k\right)=\min\left(|f(\mathbf{x}^{(k)})-k|,s\right).$$

The empirical risk is the average of the number of mistakes, where the magnitude of a mistake is related to the total ordering: $R^s_{emp}(f) = \frac{1}{\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell_k} l^s\left(f(\mathbf{x}_i^{(k)}), k\right)$.

Arguing as Herbrich et al. (1999a), we see that the role of parameter s (bounding the loss incurred in each example) is to allow for an incorporation of a priori knowledge about the probability of the classes, conditioned by $\mathbf{x}$, $P(\mathcal{C}_k|\mathbf{x})$. This can be treated as an assumption on the concentration of the probability around a "true" rank. Let us see how all this finds its place with the data replication method.

3.1.1 THE MINIMUM MARGIN PRINCIPLE

Let us formulate the problem of separating K ordered classes $\mathcal{C}_1, \dots, \mathcal{C}_K$ in the spirit of SVMs. Starting from the generalization of the two-class separating hyperplane presented in the beginning of previous section, let us look for $(K-1)$ parallel hyperplanes represented by vector $\mathbf{w} \in \mathbb{R}^p$ and scalars $b_1, \dots, b_{K-1}$, such that the feature space is divided into K regions by the decision boundaries $\mathbf{w}^t\mathbf{x} + b_r = 0,\ r = 1, \dots, K-1$.

Going for a strategy to maximize the margin of the closest pair of classes, the goal becomes to maximize $\min|\mathbf{w}^t\mathbf{x} + b_i|/||\mathbf{w}||$. Recalling that an algebraic measure of the distance of a point to the hyperplane $\mathbf{w}^t\mathbf{x} + b$ is given by $(\mathbf{w}^t\mathbf{x} + b)/\|\mathbf{w}\|$, we can scale $\mathbf{w}$ and b_i so that the value of the minimum margin is $2/\|\mathbf{w}\|$.

The constraints to consider result from the $(K-1)$ binary classifications related to each hyperplane; the number of classes involved in each binary classification can be made dependent on a parameter s, as detailed in Section 2.1. For the hyperplane $q \in \{1, \dots, K-1\}$, the constraints result as

$$\begin{array}{lll} -(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_q) & \geq +1 & k = \max(1, q-s+1), \dots, q, \\ +(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_q) & \geq +1 & k = q+1, \dots, \min(K, q+s). \end{array} \tag{5}$$

Reasoning as in the two-class SVM for the non-linearly separable data set, the inequalities can be relaxed using slack variables and the cost function modified to penalise any failure to meet the original (strict) inequalities. The model becomes (where sgn (x) returns $+1$ if x is greater than zero; 0 if x equals zero; -1 if x is less than zero)

$$\begin{array}{lllll} \min\limits_{\mathbf{w}, b_i, \xi_i} & \multicolumn{4}{l}{\frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum\limits_{q=1}^{K-1}\sum\limits_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum\limits_{i=1}^{\ell_k} \text{sgn}\ (\xi_{i,q}^{(k)})} \\ & -(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_1) & \geq +1 - \xi_{i,1}^{(k)} & k = 1, \\ & +(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_1) & \geq +1 - \xi_{i,1}^{(k)} & k = 2, \dots, \min(K, 1+s), \\ & \vdots & & \\ & -(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_q) & \geq +1 - \xi_{i,q}^{(k)} & k = \max(1, q-s+1), \dots, q, \\ s.t. & +(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_q) & \geq +1 - \xi_{i,q}^{(k)} & k = q+1, \dots, \min(K, q+s), \\ & \vdots & & \\ & -(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_{K-1}) & \geq +1 - \xi_{i,K-1}^{(k)} & k = \max(1, K-s), \dots, K-1, \\ & +(\mathbf{w}^t\mathbf{x}_i^{(k)} + b_{K-1}) & \geq +1 - \xi_{i,K-1}^{(k)} & k = K, \\ & \xi_{i,q}^{(k)} \geq 0. & & \end{array} \tag{6}$$

Since each point $\mathbf{x}_i^{(k)}$ is replicated $2s$ times, it is also involved in the definition of $2s$ boundaries; consequently, it can be shown to be misclassified $\min(|f(\mathbf{x}_i^{(k)})-k|,s) = l^s(f(\mathbf{x}_i^{(k)}),k)$ times, where $f(\mathbf{x}_i^{(k)})$ is the class estimated by the model. As with the two-class example, $\sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k} \operatorname{sgn}(\xi_{i,q}^{(k)})$ is an upper bound of $\sum_k\sum_i l^s(f(\mathbf{x}_i^{(k)}),k)$, proportional to the empirical risk.[2]

However, optimization of the above is difficult since it involves a discontinuous function $\operatorname{sgn}()$. As it is common in such cases, we choose to optimize a closely related cost function, and the goal becomes

$$\min_{\mathbf{w},b_i,\xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k}\xi_{i,q}^{(k)}$$

subject to the same constraints as Eq. (6).

In order to account for different misclassification costs or sampling bias, the model can be extended to penalise the slack variables according to different weights in the objective function (Lin et al., 2002):

$$\min_{\mathbf{w},b_i,\xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + \sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k}C_{i,q}^{(k)}\xi_{i,q}^{(k)}.$$

As easily seen, the proposed formulation resembles the fixed margin strategy in Shashua and Levin (2002). However, instead of using only the two closest classes in the constraints of an hyperplane, more appropriate for the loss function $l_{0-1}()$, we adopt a formulation that captures better the performance of a classifier for ordinal data.

Some problems were identified in the Shashua's approach. Firstly, it is an incompletely specified model. In fact, although the direction of the hyperplanes $\mathbf{w}$ is unique under the above formulation (proceeding as Vapnik (1998) for the binary case), the scalars $b_1,\dots,b_{K-1}$ are not uniquely defined, as illustrated in Figure 7.

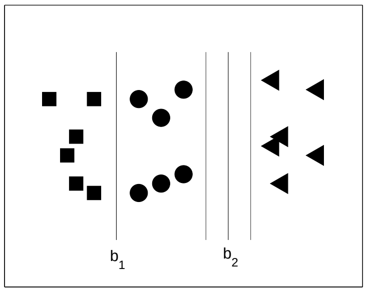


Figure 7: Scalar b_2 is undetermined over an interval under the fixed margin strategy.

Secondly, setting $s < (K-1)$ may produce awkward results on some unfortunate cases. In fact, as pointed out by Chu and Keerthi (2005), the ordinal inequalities on the thresholds $-b_1 \leq -b_2 \leq$

2. Two parameters named s have been introduced. In Section 2.1 the s parameter bounds the number of classes involved in the definition of each boundary, controlling the growth of the original data set. The parameter s introduced in Eq. (4) bounds the loss incurred in each example. Here we see that they are the same parameter.

$\ldots \leq -b_{K-1}$ are not guaranteed in this case. Only under the setting $s = K-1$ the ordinal inequalities on the thresholds are automatically satisfied (Chu and Keerthi, 2005).

Lastly, although the formulation was constructed from the two-class SVM, it is no longer solvable with the same algorithms. It would be interesting to accommodate this formulation under the two-class problem. That would allow the use of mature and optimized algorithms, developed for the training of support vector machines (Platt, 1998; Dong et al., 2005).

3.1.2 THE oSVM ALGORITHM

In order to get a better intuition of the general result, consider first the toy example previously presented. The binary SVM formulation for the extended and binarized training set can be described as $\left(\text{with } \overline{\mathbf{w}} = \left[\begin{smallmatrix}\mathbf{w}\\ w_3\end{smallmatrix}\right],\ \mathbf{w} \in \mathbb{R}^2\right)$

$$\begin{array}{ll} \min_{\overline{\mathbf{w}},b} & \frac{1}{2}\overline{\mathbf{w}}^t\overline{\mathbf{w}} \\ s.t. & \begin{array}{l} -(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(1)}\\ 0\end{bmatrix} + b) \geq +1, \\ +(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(2)}\\ 0\end{bmatrix} + b) \geq +1, \\ +(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(3)}\\ 0\end{bmatrix} + b) \geq +1, \\ -(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(1)}\\ h\end{bmatrix} + b) \geq +1, \\ -(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(2)}\\ h\end{bmatrix} + b) \geq +1, \\ +(\overline{\mathbf{w}}^t \begin{bmatrix}\mathbf{x}_i^{(3)}\\ h\end{bmatrix} + b) \geq +1. \end{array} \end{array}$$

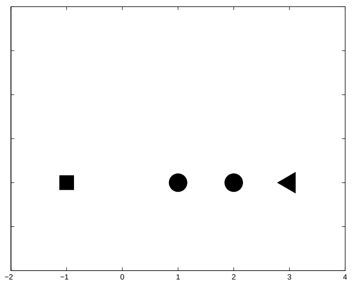
But because

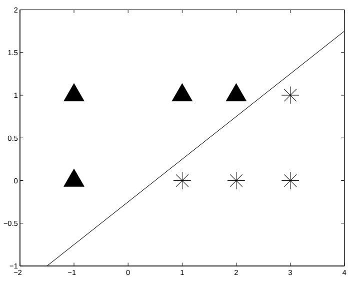
$$\begin{cases} \overline{\mathbf{w}}^t \left[\begin{smallmatrix}\mathbf{x}_i\\ 0\end{smallmatrix}\right] = \mathbf{w}^t\mathbf{x}_i \\ \overline{\mathbf{w}}^t \left[\begin{smallmatrix}\mathbf{x}_i\\ h\end{smallmatrix}\right] = \mathbf{w}^t\mathbf{x}_i + w_3 h \end{cases}$$

and renaming b to b_1 and $b + w_3 h$ to b_2 the formulation above simplifies to

$$\begin{array}{ll} \min_{\mathbf{w},b_1,b_2} & \frac{1}{2}\mathbf{w}^t\mathbf{w} + \frac{1}{2}\frac{(b_2-b_1)^2}{h^2} \\ s.t. & \begin{array}{l} -(\mathbf{w}^t\mathbf{x}_i^{(1)} + b_1) \geq +1, \\ +(\mathbf{w}^t\mathbf{x}_i^{(2)} + b_1) \geq +1, \\ +(\mathbf{w}^t\mathbf{x}_i^{(3)} + b_1) \geq +1, \\ -(\mathbf{w}^t\mathbf{x}_i^{(1)} + b_2) \geq +1, \\ -(\mathbf{w}^t\mathbf{x}_i^{(2)} + b_2) \geq +1, \\ +(\mathbf{w}^t\mathbf{x}_i^{(3)} + b_2) \geq +1. \end{array} \end{array}$$

Two points are worth mentioning: a) this formulation, being the result of a pure SVM method, has an unique solution (Vapnik, 1998); b) this formulation equals the formulation in Eq. (5) for ordinal data previously introduced, with $K = 3$, $s = K - 1 = 2$, and a slightly modified objective function by the introduction of a regularization member, proportional to the distance between the hyperplanes. The oSVM solution is the one that simultaneously minimizes the distance between boundaries and maximizes the minimum of the margins—see Figure 8. The h parameter controls the trade-off between the objectives of maximizing the margin of separation and minimizing the distance between the hyperplanes. To reiterate, the data replication method enabled us to formulate


(a) Original data set in $\mathbb{R}$.


(b) Data set in $\mathbb{R}^2$, with samples replicated ($s = 2$, $h = 1$) and oSVM solution to the binary problem.

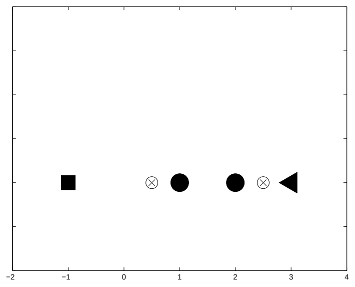

(c) oSVM solution in the original feature space.

Figure 8: Effect of the regularization member in the oSVM solution.

the classification of ordinal data as a standard SVM problem and to remove the ambiguity in the solution by the introduction of a regularization term in the objective function.

The insight gained from studying the toy example paves the way for the formal presentation of the instantiation of the data replication method in SVMs. Consider a general extended data set, as defined in Eq. (2). After the simplifications and change of variables suggested for the toy example ($b = b_1$, $b + hw_{p+i} = b_{i+1}$, $i = 1, \dots, K-2$), the binary SVM formulation for this extended data set yields

$$\min_{\mathbf{w}, b_i, \xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + \frac{1}{h^2}\sum_{i=2}^{K-1}\frac{(b_i - b_1)^2}{2} + C\sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k}\xi_{i,q}^{(k)}$$

with the same set of constraints as in Eq. (6).

This formulation for the high-dimensional data set matches the proposed formulation for ordinal data up to an additional regularization member in the objective function. This additional member is responsible for the unique determination of the thresholds.[3]

From the equivalence of the instantiation of the data replication method in SVMs and the model in Shashua and Levin (2002) and Chu and Keerthi (2005), the proof on the order of the thresholds is automatically valid for the oSVM algorithm with $s = K - 1$ (see the footnote on page 5 of Chu and Keerthi, 2005). The model parameter s cannot be chosen arbitrarily without additional constraints on the scalars $b_1, \dots, b_{K-1}$ or some additional information about classes' distribution.

To instantiate the homogeneous data replication method in support vector machines, some popular algorithm for binary SVMs, such as the SMO algorithm, must be adapted for outputting a so-

3. Different regulation members could be obtained by different extensions of the data set. For example, if $\mathbf{e}_q$ had been defined as the sequence $h, \dots, h, 0, \dots, 0$, with q h's and $(K-2-q)$ 0's, the regularization member would be $\frac{1}{2}\sum_{i=2}^{i=K-1}\frac{(b_i - b_{i-1})^2}{2}$.

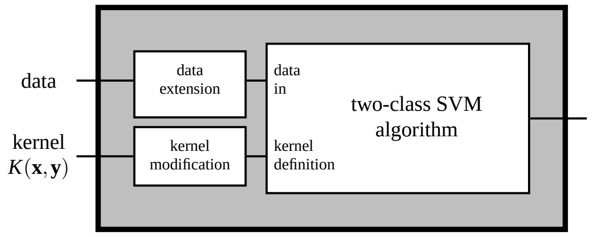


Figure 9: oSVM interpretation of an ordinal multiclass problem as a two-class problem.

lution without the bias term. Moreover, kernels have to be restricted to those satisfying $K(\mathbf{0},\mathbf{0}) = 0$, as for instance the linear kernel or the homogeneous polynomial kernel $K(x,y) = (\mathbf{x}^t\mathbf{y})^d$. To implement the enforcement on the thresholds with additional training points, one can use a different value for the C parameter in these points, sufficiently high to make certain a correct classification. Alternatively, these points must not be relaxed with a slack variable in the SVM formulation.

Nonlinear Boundaries As explained before, the search for nonlinear level curves can be pursued in the extended feature space by searching for a partially linear function $\overline{G}(\overline{\mathbf{x}}) = G(\mathbf{x}) + \underline{\mathbf{w}}^t\mathbf{e}_i$. Since nonlinear boundaries are handled in the SVM context making use of the well known kernel trick, a specified kernel $K(\mathbf{x}_i,\mathbf{x}_j)$ in the original feature space can be easily modified to $\overline{K}(\overline{\mathbf{x}}_i,\overline{\mathbf{x}}_j) = K(\mathbf{x}_i,\mathbf{x}_j) + \mathbf{e}^t_{\mathbf{x}_i}\mathbf{e}_{\mathbf{x}_j}$ in the extended space.

Summarizing, the nonlinear ordinal problem can be solved by extending the feature set and modifying the kernel function, as represented diagrammatically in Figure 9. Clearly, the extension to nonlinear decision boundaries follows the same reasoning as with the standard SVM (Vapnik, 1998).

It is true that the computational complexity of training a SVM model depends on the dimension of the input space, since the kernel functions contain the inner product of two input vectors for the linear or polynomial kernels or the distance of the two vectors for the Gaussian RBF kernel. However the matrix Q of inner products, with $(Q)_{ij} = K(x_i,x_j)$ can be computed once and kept in memory. Even on problems with many training examples, caching strategies can be developed to provide a trade-off between memory consumption and training time (Joachims, 1998). Therefore, most of the increase in the computational complexity of the problem is due to the duplication of the data; more generally, for a K-class problem, the data set is increased at most $(K-1)$ times, $O(\ell(K-1))$.

Independent Boundaries Considering now the setup for independent boundaries, as presented in Eq. (3), the linear, binary SVM formulation yields

$$\min_{\mathbf{w},b_i,\xi_i} \quad \sum_{k=1}^{K-1} \frac{1}{2}\mathbf{w}^t(kp-p+1:kp)\mathbf{w}(kp-p+1:kp) + \frac{1}{h^2}\sum_{i=2}^{K-1}\frac{(b_i-b_1)^2}{2} + C\sum_{q=1}^{K-1}\sum_{k=\max(1,q-s+1)}^{\min(K,q+s)}\sum_{i=1}^{\ell_k}\xi_{i,q}^{(k)}$$

$$s.t. \quad \begin{array}{rll}
-(\mathbf{w}^t(1:p)\mathbf{x}_i^{(k)}+b_1) & \geq +1-\xi_{i,1}^{(k)} & k=1,\\
+(\mathbf{w}^t(1:p)\mathbf{x}_i^{(k)}+b_1) & \geq +1-\xi_{i,1}^{(k)} & k=2,\ldots,\min(K,1+s),\\
\vdots & & \\
-(\mathbf{w}^t(qp-p+1:qp)\mathbf{x}_i^{(k)}+b_q) & \geq +1-\xi_{i,q}^{(k)} & k=\max(1,q-s+1),\ldots,q,\\
+(\mathbf{w}^t(qp-p+1:qp)\mathbf{x}_i^{(k)}+b_q) & \geq +1-\xi_{i,q}^{(k)} & k=q+1,\ldots,\min(K,q+s),\\
\vdots & & \\
-(\mathbf{w}^t((K-1)p-p+1:(K-1)p)\mathbf{x}_i^{(k)}+b_{K-1}) & \geq +1-\xi_{i,K-1}^{(k)} & k=\max(1,K-s),\ldots,K-1,\\
+(\mathbf{w}^t((K-1)p-p+1:(K-1)p)\mathbf{x}_i^{(k)}+b_{K-1}) & \geq +1-\xi_{i,K-1}^{(k)} & k=K,\\
\xi_{i,q}^{(k)} & \geq 0. &
\end{array}$$

If the regularization term $\frac{1}{h^2}\sum_{i=2}^{K-1}\frac{(b_i-b_1)^2}{2}$ is zero (in practice, small enough), the optimization problem could then be broken in $(K-1)$ *independent* optimization problems, reverting to the procedure of Frank and Hall (2001).

3.2 Mapping the Data Replication Method to NNs

When the nonlinear data replication method was formulated, the *real-valued* function $G(\mathbf{x})$ was defined arbitrarily. Nonintersecting boundaries were enforced by making use of a partially linear function $\overline{G}(\overline{\mathbf{x}}) = G(\mathbf{x}) + \underline{\mathbf{w}}^t\mathbf{e}_i$ defined in the extended space. Setting $G(\mathbf{x})$ as the output of a neural network, a flexible architecture for ordinal data can be devised, as represented diagrammatically in Figure 10. Because $G(\mathbf{x})$ an arbitrary real-valued function, it can be set as the output of a generic neural network with a single output. In Figure 10 $G(\mathbf{x})$ is represented as the output of a generic feedforward network. This value is then linearly combined with the added $(K-2)$ components to produce the desired $\overline{G}(\overline{\mathbf{x}})$ function.

For the simple case of searching for linear boundaries, the overall network simplifies to a single neuron with $p+K-2$ inputs. A less simplified model, also used in the conducted experiments, is to consider a single hidden layer, as depicted in Figure 11. Note that this architecture can be obtained from Figure 10 by collapsing layers 2 to $N-1$ into layer N, a valid operation when activation functions f_2 to f_{N-1} are all linear.

Similarly to the SVM mapping, it is possible to show that, if we allow the samples in all the classes to contribute errors for each threshold, by setting $s = K-1$, the order inequalities on the thresholds are satisfied automatically, in spite of the fact that such constraints on the thresholds are not explicitly included in the formulation. Refer to Appendix B for the detailed proof.

The mapping of the homogeneous data replication method to neural networks is easily realized. In order to obtain $G(\mathbf{0}) = 0$, just remove the biases inputs, represented in Figure 10, and restrict the activation functions $f_i()$ to those verifying $f_i(\mathbf{0}) = 0$. The constrains on the thresholds in form of additional training points can be realized in networks by adapting the performance function to penalise with a sufficiently high value any error in classifying these points.

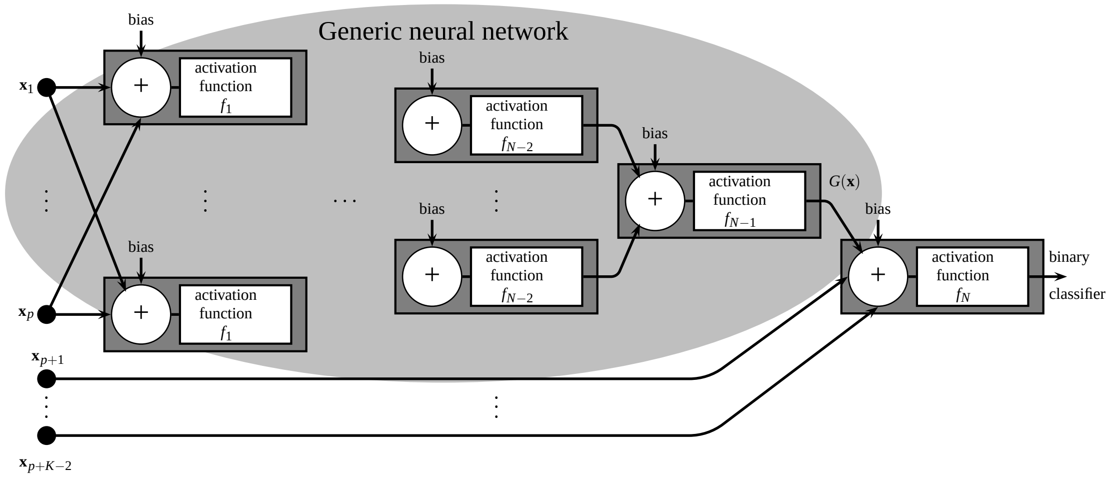


Figure 10: Data replication method for neural networks (oNN).

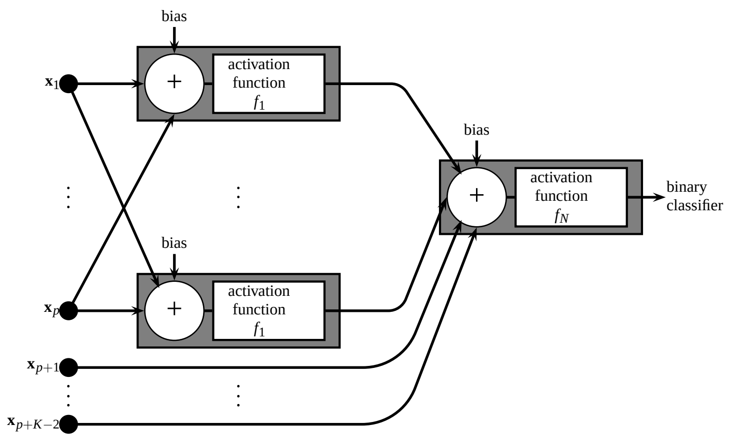


Figure 11: Simplified oNN model for neural networks.

3.2.1 ORDINAL LOGISTIC REGRESSION MODEL

Here we provide a probabilistic interpretation for the ordinal neural network model just introduced. The traditional statistical approach for ordinal classification models the cumulative class probability $P_k = p(C \leq k|\mathbf{x})$ by

$$\text{logit}(P_k) = \Phi_k - G(\mathbf{x}) \Leftrightarrow P_k = \text{logsig}(\Phi_k - G(\mathbf{x})), \quad k = 1, \ldots, K-1 \tag{7}$$

Remember that $\text{logit}(y) = \ln \frac{y}{1-y}$, $\text{logsig(y)} = \frac{1}{1+e^{-y}}$ and $\text{logsig}(\text{logit}(y)) = y$.

For the linear version (McCullagh, 1980; McCullagh and Nelder, 1989) we take $G(\mathbf{x}) = \mathbf{w}^t\mathbf{x}$. Mathieson (1995) presents a nonlinear version by letting $G(\mathbf{x})$ be the output of a neural network. However other setups can be devised. Start by observing that in Eq. (7) we can always assume $\Phi_1 = 0$ by incorporating an appropriate additive constant in $G(\mathbf{x})$. We are left with the estimation of

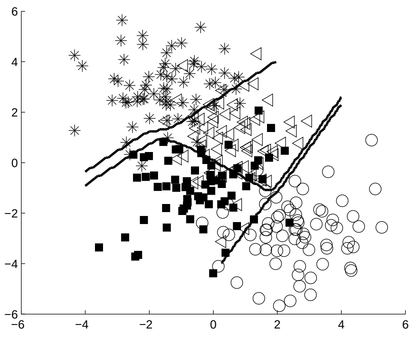


Figure 12: Decision boundaries for the oNN with 3 units in the hidden layer, for a synthetic data set from Mathieson (1995). $\mathcal{C}_1 = \circ$, $\mathcal{C}_2 = \blacksquare$, $\mathcal{C}_3 = \triangleleft$, $\mathcal{C}_4 = *$

$G(\mathbf{x})$ and $(K-2)$ cut points. By fixing $f_N() = \text{logsig}()$ as the activation function in the output layer of our oNN network, we can train the network to predict the values $P_k(\mathbf{x})$, when fed with $\bar{\mathbf{x}} = [\begin{smallmatrix}\mathbf{x}\\ \mathbf{e}_{k-1}\end{smallmatrix}]$, $k = 1, \ldots, K-1$. By setting $\overline{\mathcal{C}}_1 = 1$ and $\overline{\mathcal{C}}_2 = 0$ we see that the extended data set as defined in Eq. (2) can be used to train the oNN network. The predicted cut points are simply the weights of the connection of the added $K-2$ components, scaled by h.

Illustrating this model with the synthetic data set from Mathieson (1995), we attained the decision boundaries depicted in Figure 12.

3.3 Summation

The data replication method has some advantages over standard algorithms presented in the literature for the classification of ordinal data:

- It has an interesting and intuitive geometric interpretation. It provides a new conceptual framework integrating disparate algorithms for ordinal data classification: Chu and Keerthi (2005) algorithm, Frank and Hall (2001), ordinal logistic regression.

- While the algorithm presented in Shashua and Levin (2002); Chu and Keerthi (2005) is only formulated for SVMs, the data replication method is quite generic, with the possibility of being instantiated in different classes of learning algorithms, ranging from SVMs (or other kernel based approaches) to neural networks or something as simple as the Fisher method.

- Even the SVM instantiation of the data replication method possesses an advantage over the algorithm presented in Chu and Keerthi (2005): the latter misses the explicit inclusion of a regularization term in the objective function, leading to ambiguity in the solution. The data replication method incorporates naturally a regularization term; the unique regularization term allows interpreting the optimization problem as a single binary SVM in an extended space.

4. Experimental Methodology

In the following sections, experimental results are provided for several models based on SVMs and NNs, when applied to diverse data sets, ranging from synthetic to real ordinal data, and to a problem of feature selection. Here, the set of models under comparison is presented and different assessment criteria for ordinal data classifiers are examined.

4.1 Neural Network Based Algorithms

We compare the following algorithms:

- Conventional neural network (cNN). To test the hypothesis that methods specifically targeted for ordinal data improve the performance of a standard classifier, we tested a conventional feed forward network, fully connected, with a single hidden layer, trained with the special activation function *softmax*.
- Pairwise NN (pNN): Frank and Hall (2001) introduced a simple algorithm that enables standard classification algorithms to exploit the ordering information in ordinal prediction problems. First, the data is transformed from a K-class ordinal problem to $(K-1)$ binary problems. To predict the class value of an unseen instance the probabilities of the K original classes are estimated using the outputs from the $(K-1)$ binary classifiers.
- Costa (1996), following a probabilistic approach, proposes a neural network architecture (iNN) that exploits the ordinal nature of the data, by defining the classification task on a suitable space through a "partitive approach". It is proposed a feedforward neural network with $(K-1)$ outputs to solve a K-class ordinal problem. The probabilistic meaning assigned to the network outputs is exploited to rank the elements of the data set.
- Regression model (rNN): as stated in the introduction, regression models can be applied to solve the classification of ordinal data. A common technique for ordered classes is to estimate by regression any ordered *scores* $s_1 \leq \ldots \leq s_K$ by replacing the target class $\mathcal{C}_i$ by the score s_i. The simplest case would be setting $s_i = i, i = 1, \ldots, K$ (Mathieson, 1995; Moody and Utans, 1995; Agarwal et al., 2001). A neural network with a single output was trained to estimate the scores. The class variable $\mathcal{C}_\mathbf{x}$ was replaced by the score $s_\mathbf{x} = \frac{\mathcal{C}_\mathbf{x} - 0.5}{K}$ before applying the regression algorithm. These scores correspond to take as target the midvalues of K equal-sized intervals in the range $[0,1)$. The adopted scores are suitable for a sigmoid output transfer function, which always outputs a value in $(0,1)$. In the test phase, if a test query obtains the answer $\hat{s}_\mathbf{x}$ the corresponding class is predicted as $\hat{\mathcal{C}}_\mathbf{x} = \lfloor K\hat{s}_\mathbf{x} \rfloor + 1$.
- Proposed ordinal method (oNN), based on the standard data extension technique, as previously introduced.

Experiments with neural networks were carried out in Matlab 7.0 (R14), making use of the Neural Network Toolbox. All models were configured with a single hidden layer and trained with Levenberg-Marquardt back propagation method, over at most 2000 epochs.

4.2 SVM Based Algorithms

We compare the following algorithms:

- A conventional multiclass SVM formulation (cSVM), as provided by the software implementation LIBSVM 2.8., based on the one-against-one decomposition. The one-against-one decomposition transforms the multiclass problem into a series of $K(K-1)/2$ binary subtasks that can be trained by a binary SVM. Classification is carried out by a voting scheme.
- Pairwise SVM (pSVM): mapping in support vector machines the strategy of Frank and Hall (2001) above mentioned for the pNN model.
- Regression SVM (rSVM): The considered class of support vector regression was that of ν-SVR Scholkopf et al. (2000), as provided by the software implementation LIBSVM 2.8. Note that the model was trained to estimate the scores $s_{\mathbf{x}}$ as defined before in respect to the rNN model. Because ν-SVR does not guarantee outputs $\in [0,1)$, the predicted class $\hat{\mathcal{C}}_{\mathbf{x}} = \lfloor K\hat{s}_{\mathbf{x}} \rfloor + 1$ was properly cropped.
- Proposed ordinal method (oSVM), based on the standard data extension technique, as previously introduced.

All support vector machine models were implemented in C++, using as core the software implementation provided by LIBSVM 2.8.

4.3 Measuring the Performance of Ordinal Data Classifiers

Having built a classifier, the obvious question is "how good is it?". This begs the question of what we mean by good. A common approach is to treat every misclassification as equally costly, adopting the misclassification error rate (MER) criterion to measure the performance of the classifier. However, as already expressed, losses that increase with the absolute difference between the class numbers capture better the fundamental structure of the ordinal problem. The mean absolute deviation (MAD) criterion takes into account the degree of misclassification and is thus a richer criterion than MER. The loss function corresponding to this criterion is $l(f(\mathbf{x}),y) = |f(\mathbf{x}) - y|$. A variant of the above MAD measure is the mean square error (MSE), where the absolute difference is replaced by the square of the difference, $l(f(\mathbf{x}),y) = (f(\mathbf{x}) - y)^2$.

Still, all these measures depend on the number assigned to each class, which is somewhat arbitrary. In order to try to avoid the influence of the numbers chosen to represent the classes on the performance assessment, we can look only at the order relation between true and predicted class numbers. The use of Spearman (r_s) and Kendall's tau-b (τ_b) coefficients, nonparametric rank-order correlation coefficients well established in the literature (Press et al., 1992) are a step forward in that direction.

To get r_s start by ranking the two vectors of true and predicted classes. Ranking is achieved by giving the ranking '1' to the biggest number in a vector, '2' to the second biggest value and so on. Obviously, there will be many examples in the class vector with common values; when ranking, those examples are replaced by average ranks. If $\mathbf{R}$ and $\mathbf{Q}$ represent two rank vectors, then

$$r_s = \frac{\sum(\mathbf{R}_i - \bar{\mathbf{R}})(\mathbf{Q}_i - \bar{\mathbf{Q}})}{\sqrt{\sum(\mathbf{R}_i - \bar{\mathbf{R}})^2 \sum(\mathbf{Q}_i - \bar{\mathbf{Q}})^2}}.$$

To define τ_b, start with the N data points $(\mathcal{C}_{\mathbf{x}_i}, \hat{\mathcal{C}}_{\mathbf{x}_i}), i = 1, \ldots, N$, associated with the true and predicted classes, and consider all $\frac{1}{2}N(N-1)$ pairs of data points. Following the notation in Press et al. (1992), we call a pair (i, j) *concordant* if the relative ordering of the true classes $\mathcal{C}_{\mathbf{x}_i}$ and $\mathcal{C}_{\mathbf{x}_j}$ is the same as the relative ordering of the predicted classes $\hat{\mathcal{C}}_{\mathbf{x}_i}$ and $\hat{\mathcal{C}}_{\mathbf{x}_j}$. We call a pair *discordant* if the relative ordering of the true classes is opposite from the relative ordering of the predicted classes. If there is a tie in either the true or predicted classes, then we do not call the pair either concordant or discordant. If the tie is in the true classes, we will call the pair an "extra true pair", e_t. If the tie is in the predicted classes, we will call the pair an "extra predicted pair", e_p. If the tie is both on the true and the predicted classes, we ignore the pair. The τ_b coefficient can be computed as

$$\tau_b = \frac{concordant - discordant}{\sqrt{concordant + discordant + e_t}\sqrt{concordant + discordant + e_p}}.$$

Although insensitive to the number assigned to each class, both r_s and τ_b are in fact more appropriate for pairwise ranking rather than to ordinal regression, due to their failure to detect bias errors. In fact, if the predicted class is always a shift by a constant value of the true class, both indices will report perfect performance of the classifier.

Without a clear advantage of one criterion over the others, we decided on employing all the abovementioned assessment criteria in the conducted experiments.

5. Results for Synthetic Data

In a first comparative study we generated a synthetic data set in a similar way to Herbrich et al. (1999b). We generated 1000 example points $\mathbf{x} = [x_1\ x_2]^t$ uniformly at random in the unit square $[0,1] \times [0,1] \subset \mathbb{R}^2$. Each point was assigned a rank y from the set $\{1,2,3,4,5\}$, according to

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

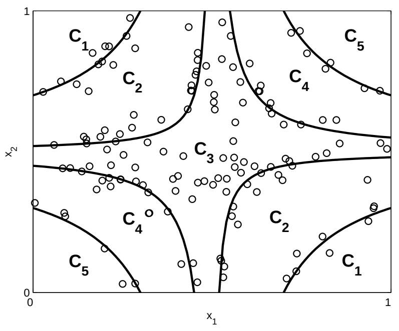
$$(b_0, b_1, b_2, b_3, b_4, b_5) = (-\infty, -1, -0.1, 0.25, 1, +\infty).$$

where $\varepsilon \sim N(0; 0.125^2)$ simulates the possible existence of error in the assignment of the true class to $\mathbf{x}$. Figure 13(a) depicts the 14.2% of examples which were assigned to a wrong class after the addition of ε. The unbalanced distribution of the random variable is shown is Figure 13(b).

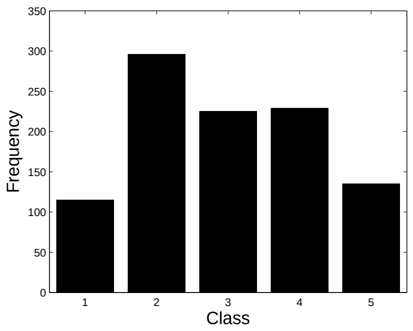
In order to compare the different algorithms, we randomly split 100 times the data set into training, validation and test sets. Each model parameterization, namely the C parameter for SVMs (we tried values of the form $C = 1.25^i$, where $i \in \{-8, \ldots, 40\}$) and the number of neurons in the hidden layer for networks (varied from 0 to 10), was selected in accordance with the best mean performance over the 100 setups of the validation set. This was repeated taking $\ell \in \{20, 40, 80\}$ for size of the training set, ℓ for the validation set and $1000 - 2 \times \ell$ for the test set. The test results for SVMs are shown in Table 1, for the MAD criterion.

We also investigated the other introduced criteria to assess models' relative performance. Results depicted in Figure 14 for this synthetic data set are representative of the agreement observed throughout the experimental study. All indices portrayed essentially the same relative models' performance. For this reason, we shall restrict in the following to present only the results for the MAD criterion, possibly the most meaningful criterion for the ordinal regression problem.

The results attained with neural networks based models are presented in Table 2. Notice that the size of the training set, ℓ, was taken in $\{40, 80, 120\}$. The training time of both SVM and NN models represents the total time to search on the parameter range and over the 100 setups.



(a) Scatter plot of the 14.2% data points wrongly classified. Also shown are the class boundaries.



(b) Class distribution.

Figure 13: Synthetic data set with 5 classes in $\mathbb{R}^2$.

Model		Training sets size $\ell = 20$	$\ell = 40$	$\ell = 80$	Training time for $\ell = 80$ (sec)
		MAD			
cSVM		0.47 (0.11)	0.30 (0.05)	0.22 (0.03)	25
pSVM		0.40 (0.10)	0.27 (0.04)	0.22 (0.02)	26
rSVM		0.32 (0.06)	0.26 (0.04)	0.24 (0.03)	2 246
	$s = 1$	0.29 (0.07)	0.20 (0.03)	0.17 (0.02)	508
oSVM	$s = 2$	0.28 (0.07)	0.20 (0.03)	0.17 (0.02)	488
	$s = 4$	0.28 (0.07)	0.20 (0.03)	0.17 (0.02)	449

Table 1: Mean (standard deviation) of MAD over 100 setups of the test set. Configuration: K $(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^t\mathbf{y})^2$, $h = 10$ for oSVM, $\nu = 0.5$ for rSVM.

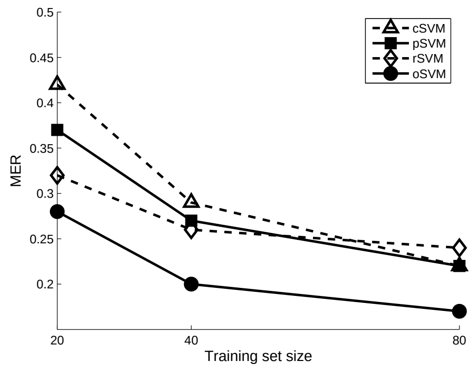
5.1 Accuracy Dependence on the Number of Classes and Data Dimension

To investigate the influence of the number of classes and data dimension on models' relative performance, the described experiment was repeated for a data set with 10 classes in $\mathbb{R}^4$. This time 2000 example points $\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^t$ were generated uniformly at random in the unit square in $\mathbb{R}^4$. The rank of each example was assigned according to the rule

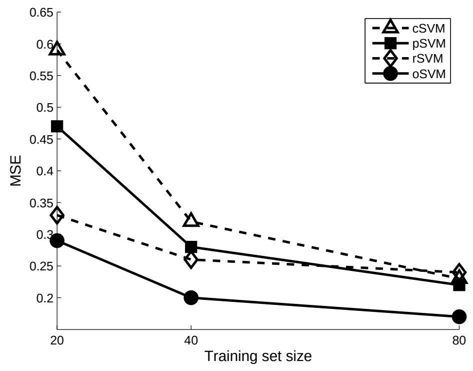
$$y = \min_{r\in\{1,2,3,4,5,6,7,8,9,10\}} \{r : b_{r-1} < 1000\prod_{i=1}^{4}(x_i - 0.5) + \varepsilon\ < b_r\},$$

$$(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}) \quad = \quad (-\infty, -5, -2.5, -1, -0.4, 0.1, 0.5, 1.1, 3, 6, +\infty).$$

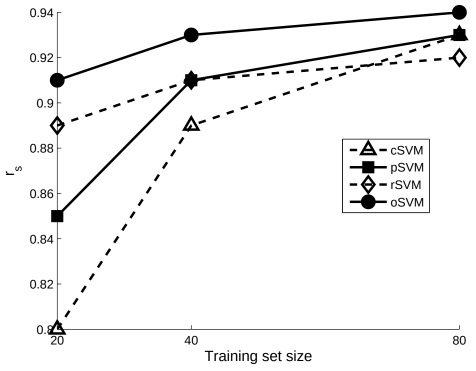
where $\varepsilon \sim N(0; 0.125^2)$. Class distributions are presented in Figure 15.



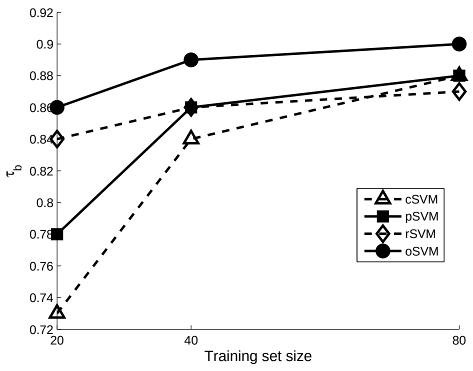
(a) MER coefficient.



(b) MSE coefficient.



(c) Spearman coefficient.



(d) Kendall's tau-b coefficient.

Figure 14: SVMs' results for 5 classes in $\mathbb{R}^2$. Configuration: K $(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^t\mathbf{y})^2$, $h = 10$ for oSVM, $\nu = 0.5$ for rSVM.

All models were trained following the same methodology as presented in the previous experiment, the only differences being those of a degree 4 for the polynomial kernel taken for the SVMs and the range of the hidden neurons for NNs, now from 0 to 20. Moreover, due to practical considerations, results were averaged only over twenty runs. Tables 3 and 4 show the test results.

5.2 Discussion

The main assertion concerns the superiority of all algorithms specific to ordinal data over conventional methods, both for support vector machines and neural networks. Additionally, neural networks exhibit a slower learning curve than support vector machine models. However, it is true that the nonlinearity selected for the data clearly favours the polynomial kernel of a support vector

Model		Training sets size $\ell = 40$	$\ell = 80$	$\ell = 120$	Training time for $\ell = 120$ (sec)
		MAD			
cNN		0.66 (0.21)	0.58 (0.18)	0.50 (0.18)	11 764
iNN		0.48 (0.13)	0.43 (0.18)	0.38 (0.16)	4 995
pNN		0.42 (0.10)	0.37 (0.09)	0.37 (0.11)	3 814
rNN		0.33 (0.15)	0.24 (0.04)	0.21 (0.03)	4 229
	$s = 1$	0.38 (0.14)	0.28 (0.05)	0.26 (0.08)	11 879
oNN	$s = 2$	0.37 (0.19)	0.30 (0.11)	0.25 (0.08)	11 079
	$s = 4$	0.38 (0.21)	0.30 (0.20)	0.26 (0.10)	11 477

Table 2: Mean (standard deviation) of MAD over 100 setups of the test set ($h = 10$ for oNN).

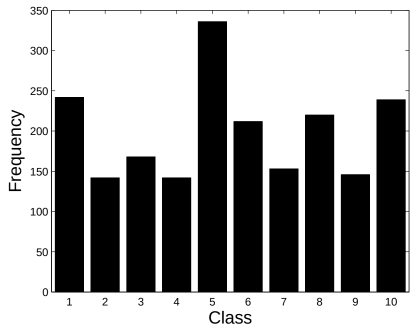


Figure 15: Class distribution for data set with 10 classes in $\mathbb{R}^4$.

Model		Training sets size $\ell = 80$	$\ell = 160$	$\ell = 240$	Training time for $\ell = 240$ (sec)
		MAD			
cSVM		2.26 (0.09)	2.06 (0.10)	1.88 (0.13)	536
pSVM		2.06 (0.13)	1.23 (0.11)	0.84 (0.07)	14 140
rSVM		1.73 (0.23)	1.29 (0.10)	1.20 (0.09)	382 867
	$s = 1$	1.36 (0.24)	0.51 (0.08)	0.33 (0.04)	97 691
oSVM	$s = 2$	1.35 (0.25)	0.50 (0.08)	0.31 (0.03)	113 383
	$s = 9$	1.35 (0.25)	0.50 (0.08)	0.31 (0.03)	115 917

Table 3: Mean (standard deviation) of MAD over twenty setups of the test set. Configuration: K $(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x}^t\mathbf{y})^4$, $h = 10$ for oSVM, $\nu = 0.5$ for rSVM.

machine algorithm over the possible models of a neural network with standard activation functions. The proposed data replication method, in spite of being the simplest model, exhibits the best performance among the support vector machine methods and the second best among networks. These

Model		Training sets size $\ell = 240$	$\ell = 320$	$\ell = 480$	Training time for $\ell = 480$
		MAD			(sec)
cNN		2.28 (0.15)	2.31 (0.39)	2.01 (0.23)	89 347
iNN		1.23 (0.41)	0.87 (0.33)	0.57 (0.34)	89 775
pNN		1.26 (0.13)	1.12 (0.14)	0.93 (0.09)	15 786
rNN		0.59 (0.07)	0.42 (0.03)	0.44 (0.24)	12 216
oNN	$s = 1$	1.06 (0.32)	0.64 (0.15)	0.44 (0.26)	14 618
	$s = 2$	0.80 (0.39)	0.54 (0.07)	0.39 (0.17)	26 057
	$s = 9$	0.79 (0.28)	0.52 (0.27)	0.40 (0.15)	58 381

Table 4: Mean (standard deviation) of MAD over twenty setups of the test set ($h = 10$ for oNN).

conclusions were reinforced with the increase of the dimension and the number of classes of the data, where differences were exacerbated.

Finally, it is worth pointing out that in these experiments no difference is visible among the performance of the different instantiations of the proposed method with SVMs, when trained with $s = 1$, $s = 2$ or $s = K-1$. Moreover, the difference is only visible for the neural network instantiation of the method when the number of classes is 10. This conclusion is likely to embody the fact that, while with neural networks all training points contribute to the boundary, with support vector machines only a small set of the same—the support vectors—have a say in the solution. Therefore, for well behaved data sets, the extra points intrinsic to $s > 1$ may not contribute to the set of support vectors, and the impact in the final solution be negligible.

6. Classifying Real Ordinal Data

In this section, we continue the experimental study by applying the algorithms considered to the classification of real ordinal data, namely to solving problems of prediction of pasture production and employee selection. The considered data sets are available at the WEKA website (`http://www.cs.waikato.ac.nz/ml/index.html`).

6.1 Pasture Production

The objective related to the pasture data set 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 (fertilizer application/stocking rate) histories (1973-1994) were measured and subdivided into 36 paddocks. Nineteen vegetation (including herbage production); soil chemical, physical and biological; and soil water variables were selected as potentially useful biophysical indicators, totaling 22 attributes. The target feature, the pasture production, has been categorized in three classes (Low, Medium, High), evenly distributed in the data set of 36 instances. Before training, the data was scaled to fall always within the range $[0,1]$, using the transformation $x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$. 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).

Continuing with the same experimental methodology, the C parameter was varied from 1.25^{-32} to 1.25^{32}; the number of neurons in the hidden layer for NNs was varied from 0 to 10. The results attained are summarized in Table 5.

Model		Training sets size $\ell = 3$	$\ell = 6$	$\ell = 16$	Training time for $\ell = 16$ (sec)
		MAD			
cSVM		0.44 (0.12)	0.42 (0.13)	0.28 (0.26)	1
pSVM		0.50 (0.12)	0.42 (0.12)	0.26 (0.12)	2
rSVM		0.49 (0.12)	0.38 (0.12)	0.29 (0.21)	5
oSVM	$s = 1$	0.40 (0.10)	0.40 (0.12)	0.34 (0.23)	3
	$s = 2$	0.40 (0.10)	0.40 (0.12)	0.34 (0.23)	3
	$s = 1$, $j = 21$	0.39 (0.37)	0.36 (0.16)	0.21 (0.13)	3
cNN		0.51 (0.19)	0.44 (0.15)	0.29 (0.26)	708
pNN		0.49 (0.17)	0.43 (0.13)	0.26 (0.26)	278
iNN		0.52 (0.18)	0.43 (0.12)	0.34 (0.27)	282
rNN		0.74 (0.24)	0.36 (0.23)	0.26 (0.28)	748
oNN	$s = 1$	0.43 (0.13)	0.41 (0.15)	0.34 (0.26)	195
	$s = 2$	0.45 (0.14)	0.43 (0.15)	0.35 (0.15)	192

Table 5: Mean (standard deviation) of MAD over 100 setups of the test set. Configuration: K $(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^t\mathbf{y})^2$ for SVMs, $h = 1$ for oSVM and oNN, $\nu = 0.5$ for rSVM.

We start by observing that conventional methods performed as well as ordinal methods. We were led to the suggestion that some of the features may not properly reflect the ordinal relation among classes. A likely exemplar is the fertiliser attribute. The lack of motivation to impose an ordered relation in the fertiliser attribute, suggests a good scenario to apply the general version of the data replication method, where only 21 attributes ($j = 21$) are constrained to have the same direction, with the fertiliser attribute (coded with four binary variables) left free. Using a linear kernel emerges a classifier with expected MAD of 21%. This way, a very simple classifier was obtained at the best performance.

6.2 Employee Selection: the ESL Data Set

The ESL data set contains 488 profiles of applicants for certain industrial jobs. Expert psychologists of a recruiting company, based upon psychometric test results and interviews with the candidates, determined the values of the input attributes (4 attributes, with integer values from 0 to 9). The output is an overall score (1..9) corresponding to the degree of fitness of the candidate to this type of job, distributed according to Figure 16.

Continuing with the same methodology and range of values for the C and the number of hidden neurons parameters from Pasture, one obtained the results reported in Table 6. In this experiment, as well as in all the previous ones, the SVM instantiation of the data replication method exhibits a performance nearly independent of the s parameter. The reasons for this behaviour were already invoked. The neural network mapping, on the other hand, seems to perform better with the increase of the s value, mainly when the number of classes involved is significant. Nonetheless, in this

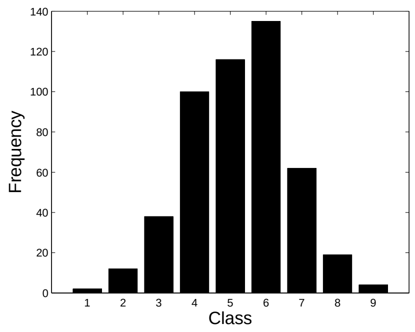


Figure 16: Class distribution for the 488 examples of the ESL data set.

Model		Training sets size $\ell = 25$	$\ell = 50$	$\ell = 100$	Training time for $\ell = 100$ (sec)
		MAD			
cSVM		0.52 (0.06)	0.47 (0.04)	0.39 (0.04)	3
pSVM		0.47 (0.05)	0.40 (0.03)	0.36 (0.03)	3
rSVM		0.36 (0.02)	0.34 (0.03)	0.33 (0.01)	32
oSVM	$s = 1$	0.46 (0.07)	0.42 (0.04)	0.35 (0.02)	47
	$s = 2$	0.45 (0.05)	0.39 (0.03)	0.34 (0.02)	64
	$s = 8$	0.45 (0.05)	0.39 (0.04)	0.34 (0.02)	85
cNN		0.88 (0.24)	0.80 (0.12)	0.49 (0.06)	8 611
pNN		0.56 (0.09)	0.52 (0.08)	0.39 (0.05)	3 498
iNN		0.55 (0.09)	0.46 (0.13)	0.39 (0.04)	7 398
rNN		0.43 (0.04)	0.38 (0.03)	0.34 (0.02)	1 622
oNN	$s = 1$	0.66 (0.28)	0.56 (0.24)	0.38 (0.16)	6 444
	$s = 2$	0.48 (0.08)	0.44 (0.08)	0.35 (0.03)	8 422
	$s = 8$	0.49 (0.14)	0.45 (0.13)	0.35 (0.05)	7 635

Table 6: Mean (standard deviation) of MAD over twenty setups of the test set. Configuration: K $(\mathbf{x},\mathbf{y}) = \mathbf{x}^t\mathbf{y}$ for SVMs, $h = 10$ for oSVM and oNN, $\nu = 0.5$ for rSVM.

experiment the instantiation with $s = 2$ already performed as well as the largest possible value of $s = 8$. Moreover, it is not clear a correlation between s and the training time. The overall results suggest that the data replication method learns faster than standard methods, although followed closely by the regression based models.

7. Gene Expression Analysis

Now we address the problem of selection of a small subset of genes from broad patterns of gene expression data, recorded on DNA micro-arrays. Singh et al. (2002) carried out microarray expression analysis on 12600 genes to identify genes that might anticipate the clinical behaviour of prostate cancer. Fifty-two samples of prostate tumour were investigated. For each sample, the degree of tumour cell differentiation or Gleason score (GS) was assessed by the pathologist; for tumour samples the GS ranged from 6 to 10. Predicting the Gleason score from the gene expression data is thus a typical ordinal classification problem, already addressed in Chu and Ghahramani (2005) using Gaussian processes. Following Chu and Ghahramani (2005), and since only 6 samples had a score greater than 7, we merged them as the top level, leading to three levels $\{=6, =7, \geq 8\}$, with 26, 20 and 6 samples respectively.

To evaluate the suitability of the oSVM algorithm for feature ranking and selection according to their relevance for the classification task, we applied the oSVM to the data set with the 12600 genes. A quality of SVMs is that the weights of the features in the borders $\mathbf{w}^t\mathbf{x} + b_r$ can be used for feature ranking and for the selection of subsets that are useful to build a good predictor (Guyon et al., 2002). Because the oSVM model computes the same set of weights for all borders, it provides an obvious feature ranking.

The application of the oSVM algorithm to the 12600 genes provided a first ranking of the genes given by the weights in the classifier. We then removed the irrelevant genes based on the rank list, assessing several subsets, as presented in Table 7(a). The best subset found had 500 genes. However, it is unlikely that the ranking of the 12600 genes have enough resolution to correctly rank genes in subsets much smaller than the original.

Therefore, we iterated the procedure starting now from the 500 genes selected by the first ranking: applying the oSVM algorithm to the 500 genes provided a second, refined ranking of these genes. Then, several subsets were assessed, Table 7(b). As visible, the ranking was in fact improved: the result for the first 100 genes was remarkably improved. Once again, starting with the best subset identified—100 genes—the oSVM algorithm was used to re-rank the genes, after which several subsets were again evaluated, Table 7(c). The procedure was successively repeated with 60, 40, 30, 26, 24, 19, 16 and 15 genes, when further reduction of the number of genes increased the misclassification error, Tables 7(d)-7(j).

During this process the leave one out method was used to estimate the MAD coefficient. The parameterization of the method was kept constant: $h = 1$, $s = 1$, $C = 5$. Before training, the data was scaled to fall always within the range $[-1, 1]$, using the transformation $x' = \frac{2x - x_{\max} - x_{\min}}{x_{\max} - x_{\min}}$. In this way the weight of each gene in the classifier conveys the relevance of the gene for the classification process.

Finally, the performance of the remaining algorithms was also determined in the considered subsets of genes.[4] Results are displayed in Table 8. We observe great and steady improvement of all classifiers using the subset of genes selected by the oSVM algorithm. The results attained agree with the results reported in Chu and Ghahramani (2005), although we were able to attain lower error rates. The best validation output was achieved with 15 genes, better than the 26 needed by the approach reported in Chu and Ghahramani (2005), revealing a better selection of genes for the classification task.

SVMs are particularly suited for the analysis of gene expression. They easily accommodate the high number of genes and perform well even with the small samples frequent in this area. Here the integration of the feature selection operation with the algorithm of classification in a consistent framework provided positive results.

4. Due to difficulties to handle such a high number of features with the Matlab `Neural Network Toolbox`, this experiment was only conducted with the SVM based algorithms.

n genes	MAD
12600	57.7
5000	67.3
500	**21.1**
100	38.5

(a) Ranking with 12600 genes.

n genes	MAD
500	21.1
300	13.5
100	**3.8**
50	13.5

(b) Ranking with 500 genes.

n genes	MAD
100	3.8
80	1.9
60	**0.0**
50	3.8

(c) Ranking with 100 genes.

n genes	MAD
60	**0.0**
50	**0.0**
40	**0.0**
30	7.7

(d) Ranking with 60 genes.

n genes	MAD
40	**0.0**
30	**0.0**
20	3.8
10	28.8

(e) Ranking with 40 genes.

n genes	MAD
30	**0.0**
28	**0.0**
26	**0.0**
25	1.9

(f) Ranking with 30 genes.

n genes	MAD
26	**0.0**
25	**0.0**
24	**0.0**
23	3.8

(g) Ranking with 26 genes.

n genes	MAD
24	**0.0**
21	**0.0**
19	**0.0**
17	5.8

(h) Ranking with 24 genes.

n genes	MAD
19	**0.0**
17	**0.0**
16	**0.0**
15	3.8

(i) Ranking with 19 genes.

n genes	MAD
16	**0.0**
15	**0.0**
14	1.9
13	7.7

(j) Ranking with 16 genes.

Table 7: MAD (%) for the oSVM algorithm, for the prostate cancer data set.

Model	MAD (%)										
	n genes										
	1	14	15	20	30	40	60	100	500	5000	12600
cSVM	81	6	10	6	2	2	6	8	23	58	48
pSVM	62	6	10	6	2	2	6	8	25	59	52
rSVM	62	15	11	2	10	19	4	6	21	69	60
oSVM	75	2	0	0	0	0	0	4	21	67	58

Table 8: Results using a linear kernel on the prostate cancer data of selected genes.

8. Conclusion

This study focuses on the application of machine learning methods, and in particular of neural networks and support vector machines, to the problem of classifying ordinal data. A novel approach to train learning algorithms for ordinal data was presented. The idea is to reduce the problem to the standard two-class setting, using the so called *data replication method,* a nonparametric procedure for the classification of ordinal categorical data. This method was mapped into neural networks and support vector machines. Two standard methods for the classification of ordinal categorical data were unified under this framework, the minimum margin principle (Shashua and Levin, 2002) and the generic approach by Frank and Hall (2001). Finally, a probabilistic interpretation for the neural network model was also presented.

The study compares the results of the proposed model with conventional learning algorithms for nominal classes and with models proposed in the literature specifically for ordinal data. Simple misclassification, mean absolute error, Spearman and Kendall's tau-b coefficients are used as measures of performance for all models and used for model comparison. The new methods are likely to produce simpler and more robust classifiers, and compare favourably with state-of-the-art methods. In spite of being usually assumed that learning in a higher dimension becomes a harder problem, the performance of the data replication method does not seem to be affected, probably due to the dependence among the data replicas.

The data replication method is parameterised by h (and C); because it may be difficult and time consuming to choose the best value for h, it would be interesting to study possible ways to automatically set this parameter, probably as a function of the data and C. It would also be interesting to study if this algorithm can be successfully applied to nominal data. Although the data replication method was designed for ordinal classes, nothing impedes its application to nominal classes. It is expected that the classifier should be evaluated for each possible permutation of the classes, choosing the one conducting to the best performance (feasible only when the number of classes is small).

Acknowledgments

The authors are grateful to Luís F. Teixeira and Vitor Cardoso for their help on preparing the manuscript and to the Reviewers and the JMLR Editor for their many stimulating and thoughtful comments.

Appendix A.

In this appendix we derive a margin-based bound on the generalization error of the proposed ordinal classifier. The following theorem shows that the fat-shattering dimension gives generalization error bounds for large margin classifiers (Bartlett and Shawe-Taylor, 1999).

Theorem Consider a class $\mathcal{F}$ of real-valued functions. With probability at least $1-\delta$ over ℓ independently generated examples $\mathbf{x}$, if a classifier $\text{sgn}(f) \in \text{sgn}(\mathcal{F})$ has margin at least γ on $\mathbf{x}$, then the error of $\text{sgn}(f)$ is bounded from above by

$$\frac{2}{\ell}\left(Z \log_2 \frac{8e\ell}{Z} \log_2(32\ell) + \log_2 \frac{8\ell}{\delta}\right),$$

where $Z = \text{fat}_{\mathcal{F}}(\gamma/16)$, with $\text{fat}_{\mathcal{F}}()$ the fat shattering dimension. Furthermore, with probability at least $1-\delta$, every classifier $\text{sgn}(f) \in \text{sgn}(\mathcal{F})$ has error no more than

$$R_{emp}^{0-1}(f) + \sqrt{\frac{2}{\ell}(Z \log_2(34e\ell/Z) \log_2(578\ell) + \log_2(4/\delta))},$$

where $R_{emp}^{0-1}(f)$ is the average of the number of training examples with margin less than γ.

It is not possible to apply the above equations directly to the extended data set because examples are not independently generated. We will proceed as follows. For each example $\mathbf{x}_i^{(k)} \in \mathbb{R}^p$ define $\mathbf{x'}_i^{(k)}$ as a single replica in $\mathbb{R}^{p+K-2} \times \{-1, 1\}$ chosen uniformly at random from all the replicas from

$\mathbf{x}_i^{(k)}$. Since an error is made on example $\mathbf{x}_i^{(k)}$ if any of its $(K-1)$ replicas is wrongly classified, it is necessary to guarantee that no error is made. It follows that the generalization error for the ordinal problem is bounded from above by

$$\frac{2(K-1)}{\ell}\left(Z\log_2\frac{8e\ell}{Z}\log_2(32\ell)+\log_2\frac{8\ell}{\delta}\right)$$

or by

$$R_{emp}(f)+(K-1)\sqrt{\frac{2}{\ell}(Z\log_2(34e\ell/Z)\log_2(578\ell)+\log_2(4/\delta))}$$

where $R_{emp}(f)$ is the average of the magnitude of 'mistakes' of training examples:$R_{emp}(f) = \frac{1}{\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell_k} l\left(f(\mathbf{x}_i^{(k)}),k\right) = \frac{1}{\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell_k}|f(\mathbf{x}_i^{(k)})-k|$.

Appendix B.

In this appendix we prove the following claim from Section 3.2:
Claim For the mapping of the data replication method in NNs, if we allow the samples in all the classes to contribute errors for each threshold, by setting $s = K-1$, the order inequalities on the thresholds are satisfied automatically, in spite of the fact that such constraints on the thresholds are not explicitly included in the formulation.
Proof To prove the inequalities on the thresholds at the optimal solution, let us consider the situation where $\mathbf{w}$ is fixed (the first p components of $\overline{\mathbf{w}}$) and only the b_i's are optimized. Note that $w_{p+j}h + b_1 = b_{j+1}, j = 1..K-2$.

The error contributed at the data replica with $\overline{\mathbf{x}}_i^{(k)} = \begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_{q-1}\end{bmatrix}$, where $k = 1,\dots,K$ denotes the class number, $i = 1,\dots,\ell_k$ is the index within each class and $q = 2..K-1$ comes as

$$\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\overline{\mathbf{w}}^t\overline{\mathbf{x}}_i^{(k)})-t_-\} + \sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\overline{\mathbf{w}}^t\overline{\mathbf{x}}_i^{(k)})-t_+\}$$
$$= \sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_-\} + \sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_+\}.$$

where err is the error performance function and t_- and t_+ are the target values for $\overline{\mathcal{C}}_1$ and $\overline{\mathcal{C}}_2$ respectively (f_N is the output transfer function).

Likewise, the error due to the data replica with $\overline{\mathbf{x}}_i^{(k)} = \begin{bmatrix}\mathbf{x}_i^{(k)}\\ \mathbf{e}_{q}\end{bmatrix}$ comes as

$$\sum_{k=1}^{q+1}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_-\} + \sum_{k=q+2}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_+\}.$$

The error contribution of the data replicas q and $q+1$ is just the sum of these two parcels:

$$\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_-\} + \sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q-1}h)-t_+\}+$$
$$\sum_{k=1}^{q+1}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q}h)-t_-\} + \sum_{k=q+2}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)}+w_{p+q}h)-t_+\}. \quad (8)$$

Suppose that, at the optimal solution, we have $w_{p+q-1} < w_{p+q}$. Exchanging the value of w_{p+q-1} with the value of w_{p+q}, we obtain a solution where the error contribution of the data replicas $q-1$ and q is given by:

$$\sum_{k=1}^{q}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_-\} + \sum_{k=q+1}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q}h) - t_+\} + \\ \sum_{k=1}^{q+1}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q-1}h) - t_-\} + \sum_{k=q+2}^{K}\sum_{i=1}^{\ell_k} err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(k)} + w_{p+q-1}h) - t_+\}. \quad (9)$$

Observing that the error contribution of the other data replicas do not get affected by this swap of values, the total error difference between the new solution and the optimal solution simplifies to (given by subtracting the value of Eq. (8) to the value of Eq. (9))

$$\sum_{i=1}^{\ell_{q+1}} \big\{ err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h) - t_-\} - err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) - t_-\} + \\ err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) - t_+\} - err\{f_N(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h) - t_+\}\big\}. \quad (10)$$

For clarity, assume now that f_N is the log-sigmoide transfer function logsig, $t_- = 0$, $t_+ = 1$, and err () is the absolute error performance function. Then, Eq. (10) simplifies to

$$\sum_{i=1}^{\ell_{q+1}} \big\{ \mathrm{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h) - \mathrm{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) + \\ - \mathrm{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q}h) + \mathrm{logsig}(\mathbf{w}^t\mathbf{x}_i^{(q+1)} + w_{p+q-1}h)\big\}.$$

Because this last value is clearly less than zero, we obtained a solution with lower error than the optimal solution. Then, at the optimal solution $w_{p+q-1} \geq w_{p+q} \iff -b_q \leq -b_{q+1}, q = 2, \ldots, K-2$. Note that this reasoning does not get affected by using other transfer functions such as the tan-sigmoid function, together with $t_- = -1$ and $t_+ = +1$, or other error performance functions such as the squared error performance function. For the nonlinear case, we just need to replace $\mathbf{w}^t\mathbf{x}_i^{(k)}$ by $G(\mathbf{x}_i^{(k)})$ in the proof. The proof of the order relation between b_1 and b_2 in the optimal solution is left to the reader. ■

References

A. Agarwal, J. T. Davis, and T. Ward. Supporting ordinal four-state classification decisions using neural networks. In *Information Technology and Management*, pages 5–26, 2001.

P. L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 43–54. MIT Press, Cambridge, MA, 1999.

J. S. Cardoso, J. F. Pinto da Costa, and M. J. Cardoso. Modelling ordinal relations with SVMs: an application to objective aesthetic evaluation of breast cancer conservative treatment. *Neural Networks*, 18:808–817, june-july 2005.

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

W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. In *Proceedings of International Conference on Machine Learning (ICML05)*, pages 145–152, 2005.

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

J. Dong, A. Krzyzak, and C. Y. Suen. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):603–618, 2005.

E. Frank and M. Hall. A simple approach to ordinal classification. In *Proceedings of the 12th European Conference on Machine Learning*, volume 1, pages 145–156, 2001.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

R. Herbrich, T. Graepel, and K. Obermayer. Regression models for ordinal data: a machine learning approach. Technical Report TR-99/03, TU Berlin, 1999a.

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Ninth International Conference on Artificial Neural Networks ICANN*, volume 1, pages 97–102, 1999b.

T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46:191–202, 2002.

M. J. Mathieson. Ordinal models for neural networks. In A.-P.N Refenes, Y. Abu-Mostafa, and J. Moody, editors, *Neural Networks for Financial Engineering*. World Scientific, Singapore, 1995.

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

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.

J. Moody and J. Utans. Architecture selection strategies for neural networks: application to corporate bond rating prediction. In A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, pages 277–300, Chichester, 1995. Wiley.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods-Support Vector Learning*, pages 185–208, 1998.

W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press, 1992.

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

A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *Neural Information and Processing Systems (NIPS)*, 2002.

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

D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:1019–1041, 2002.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley, 1998.