A Machine Learning Algorithm for Solving Hidden Object-Ranking Problems

G. Mondonneix, Sébastien Chabrier, Jean-Martial Mari, Alban Gabillon

A Machine Learning Algorithm for Solving Hidden Object-Ranking Problems

Gaël Mondonneix, Sébastien Chabrier, Jean Martial Mari, and Alban Gabillon

University of French Polynesia
Géopôle du Pacifique Sud EA4238, LABEX CORAIL

Abstract. Hidden object-ranking problems (HORPs) are object-ranking problems stated in instance-ranking terms. To our knowledge, there is no algorithm able to process these problems with the appropriate bias. This lack is not significant as long as the size of the dataset makes it possible to capture enough information by mining more data; however, when the data are scarce, any information lying in the data is worth exploiting and such an algorithm would become useful. We explicit the appropriate bias for object-ranking problems and propose an algorithm able to apply this bias to cases where these problems arise in an instance-ranking form. The theoretical foundations of the algorithm are discussed and the algorithm is tested on scarce real data, yielding better results (94.4% accuracy) than traditional algorithms (92.6% accuracy for the best case).

Keywords: Object-Ranking, Instance-Ranking, Learning bias, Support Vector Machine

1 Introduction

1.1 Context

Supervised learning is usually split into two different types of tasks: classification tasks and regression tasks. However, a third type of supervised learning, borrowing to both classification (the labels to predict take on discrete values) and regression (the labels to predict convey order-related information), is getting more attention as recommender systems play an increasing role on the Web: ranking tasks.

Ranking Tasks. Fürnkranz and Hüllermeier [1] distinguish three subtypes of ranking tasks depending on the problem to solve: object-ranking, instance-ranking and label-ranking problems. In an object-ranking problem, we have to infer a total order from a set of partially ordered items (see [2]). In an instance-ranking problem, we are given a set of items belonging to ordered classes and have to infer for any new item the class it belongs to (see [3]). In a label-ranking problem, we are given a set of labels as well as a set of items, each one associated with a partial order on the set of labels; the task consists of inferring a total order on the set of labels for any new item (see [3], [4]).

Hidden Object-Ranking Problems (HORPs). We call hidden object-ranking problem (HORP) an object-ranking problem expressed in a form of an instance-ranking problem. A HORP has the appearance of an instance-ranking problem: the training data consist of a set of items distributed in some ordered classes and the task consists of distributing new items in these classes on a way that is consistent with the training data. However a HORP is not an instance-ranking problem in the sense that if two items are in a same class, it does not mean that they are tied (i.e. ordinally equal); it does only mean that we have no explicit information about their ordinal relation. In a HORP, the classes have no intrinsic meaning; they are external constraints meant to capture the trend of a total order by expressing it in a compact way, but the number of classes, as well as their boundaries, can change arbitrarily without having any impact on the ordinal information: ordinal relations between objects stay the same.

HORPs often arise in machine learning but they seem to get no particular attention and are solved as if they were simple instance-ranking problems. A typical example of HORP is when someone is asked to rate something by giving it from 1 to, say, 5 stars: if we ask this person to rate six items, then at least two of them will be given the same number of stars; the reason yet has less to do with preference than with external constraints we imposed on the way to express preference; this problem is not a true instance-ranking problem but a HORP.

Learning Bias. The bias is crucial in a learning process for being able to generalize to unseen data ([5] p.42), and the less the data, the most the predictions have to rely on the bias.

In a context of big data, losing some information on some data is not a problem because more information will be extracted from more data. The challenge is not to exploit all the information contained in a given piece of data but to process all the available data.

On the contrary, in a context of scarce data, if we do not exploit all the information lying in the available data, notably the possible ordinal information, there is no opportunity of recovering it through additional data. The choice of the bias becomes then a critical point (see Fig. 2. for an illustration): it has to be properly decided whether a problem stated in an instance-ranking form is a true instance-ranking problem (that is, there exists no ordinal difference between two items lying in a same class) or a HORP (ordinal differences exist between items inside a same class, even though they are unknown).

1.2 Related Work

Instance-Ranking Approach. Traditional classification algorithms do not take into account the possible ordinal information lying in the data. In multi-class classification, the problem comes from an undifferentiated cost function: if we have ordered classes C_i, $C_i \prec_p C_{i+1}$ for a given preference p, then a misclassification between C_i and C_{i+2} should be somehow more penalized than a misclassification between C_i and C_{i+1}.

Several methods have been proposed to address instance-ranking problems by extending or adapting traditional algorithms. For example, some authors propose a multiclass classification by SVM (support vector machine) with multiple parallel hyperplanes [6]. The set of hyperplanes is chosen in such a way that it maximizes the margin of the hyperplane with the thinnest margin. Since the hyperplanes are parallel, they preserve the ordinal information conveyed by the classes. Another method, called the data replication method [7], [8], is a setting that allows using binary classifiers to solve instance-ranking problems. The principle consists of learning a discrete cumulative distribution function: for all classes C_k but the last one, a binary classifier is trained to decide whether an item belongs to $\bigcup_{i=1}^{k} C_i$. At the prediction stage, an item is considered to belong to a class C_k if the classifiers put it in $\bigcup_{i=1}^{k} C_i$ but not in $\bigcup_{i=1}^{k-1} C_i$.

These methods however solve the true instance-ranking problem: they do not treat two items lying in a same class as two different objects but as two ordinally equal instances of the class; therefore their inductive bias seems not to be the most adapted for solving HORPs.

Object-Ranking Approach. Object-ranking consists of predicting a total order given only a set of sorted pairs. Solving object-ranking problems is usually done by learning a binary relation from the set of ordered pairs and choosing the closest total order [2]. Because this choice implies to solve the slaters problem [9], which is NP-equivalent [10], a way to approximate the right order is needed [2], [11].

Even though this method is tailored for object-ranking, the fact that it goes through a binary relation that is not necessarily a total order, as well as an approximation process, makes it tricky to have a control on the learning bias. Nonetheless, the pairwise approach is interesting in the perspective of solving HORPs since the training set of ordered pairs of the object-ranking problem can be derived from the interclass relations of the instance-ranking setting in which the object-ranking problem is stated.

1.3 Guideline

The two following sections present a machine learning algorithm for solving HORPs. The presentation follows the distinction between search space and search strategy [5]: in section 2 we describe the search space and prove its expressive power; in section 3 we discuss candidate algorithms with respect to their generalization power and their learning bias. The proper bias of object-ranking learning is explicited in subsection 3.2 and a setting is proposed in subsection 3.3 for enabling SVM solvers to deal with HORPs. In section 4, an experiment is reported, where the algorithm is tested on a real dataset.

2 Search Space

2.1 Definitions

Ordinal Equivalence. Let x and y be two sequences of n elements $(x_1, ..., x_n)$ and $(y_1, ..., y_n)$, $x_i \in X$, $y_i \in Y$, with order relations $\prec_x$ and $\prec_y$ on X and Y respectively. We define the ordinal equivalence between x and y, and we write $x \Leftrightarrow_{ord} y$, the property that $\forall i \leq n, \forall j \leq n, x_i \prec_x x_j \Leftrightarrow y_i \prec_y y_j$. By extension, we say that two derivations are ordinally equivalent if their left-hand members are ordinally equivalent *and* their right-hand members are ordinally equivalent.

For example, $\begin{pmatrix}0\\1\\3\end{pmatrix} \rightarrow \begin{pmatrix}3\\0\\1\end{pmatrix} \Leftrightarrow_{ord} \begin{pmatrix}1\\2\\3\end{pmatrix} \rightarrow \begin{pmatrix}3\\1\\2\end{pmatrix}$ since $\begin{pmatrix}0\\1\\3\end{pmatrix} \Leftrightarrow_{ord} \begin{pmatrix}1\\2\\3\end{pmatrix}$ and $\begin{pmatrix}3\\0\\1\end{pmatrix} \Leftrightarrow_{ord} \begin{pmatrix}3\\1\\2\end{pmatrix}$.

Ordinal equivalence is just a convenient way of ignoring scales: an ordinal equivalence between two sequences is the same as a Kendall tau of 1 between two statistical ordinal variables.

Dimensionality Extension. The rank of a matrix can be increased by adding rows or columns that are linearly independent of the rows and columns of the original matrix. Under the hypothesis that no row of X be null, we define the basis function φ_r such that $\varphi_r(X)$ extends X to rank r by adding columns that are nonlinear combinations of its original columns.

As an illustration, if $X = \begin{bmatrix}x_{1,1} & x_{1,2}\\x_{2,1} & x_{2,2}\\x_{3,1} & x_{3,2}\end{bmatrix}$ with $rank(X) = 2$, we can have for example $\varphi_3(X) = \begin{bmatrix}x_{1,1} & x_{1,2} & (x_{1,1}+x_{1,2})^2\\x_{2,1} & x_{2,2} & (x_{2,1}+x_{2,2})^2\\x_{3,1} & x_{3,2} & (x_{3,1}+x_{3,2})^2\end{bmatrix}$.

2.2 Formalization

Derscription of the Search Space. Let X be a set of n items represented by vectors x_i, $1 \leq i \leq n$. Let y be a permutation of the first n positive integers indicating an order between the items ($y_i = k$ means that the item represented by the vector x_i is in the k^{th} position in the sequence of items).

Without loss of generality, we suppose that any two items are distinct and that the mapping from the items to their representing vectors is injective ($i \neq j \Rightarrow x_i \neq x_j$). We search a vector w such that $\varphi_r(X)w \Leftrightarrow_{ord} y$.

Expressivity of the Search Space. Let write $\varphi_r(X)|y$ the matrix X extended to rank r, itself augmented with the column vector y.

If $rank(\varphi_r(X)|y) = rank(\varphi_r(X))$, then there necessarily exists a vector w such that $\varphi_r(X)w = y$; yet $y \Leftrightarrow_{ord} y$, so there necessarily exists a vector such that $\varphi_r(X)w \Leftrightarrow_{ord} y$. In other terms, the search space is expressive enough to contain any object-ranking target concept.

3 Search Strategy

3.1 Search by Extension

Derivation of a Greedy Algorithm. We can derive from the formalization of the search space an algorithm consisting of greedily computing w by iteratively increasing r until being able to solve $\varphi_r(X)w = y$.

Algorithm 1 Greedy Approach

```
r ← rank(X)
while r < rank(φ_r(X)|y) do
   r ← r + 1
end while
return w s.t. φ_r(X)w = y
```

For n items, $rank(\varphi_r(X)|y) \leq n$. Since n is finite, the algorithm is guaranteed to terminate (there are at most $n - 1$ iterations).

Limitations of the Greedy Algorithm. We can learn an order on a dataset as soon as we find a vector w such that $\varphi_r(X)w \Leftrightarrow_{ord} y$, and it is shown in the previous section that such a w exists in the search space for any X provided that r be high enough. Nonetheless, even though the greedy algorithm is guaranteed to find a vector w such that $\varphi_r(X)w \Leftrightarrow_{ord} y$, it is not guaranteed to find the one for which r is minimal.

We can illustrate this point with the following example: when we run the algorithm, we notice that it needs to go up to $r = 4$ for learning $\begin{pmatrix}1\\2\\3\\4\end{pmatrix} \rightarrow \begin{pmatrix}1\\2\\4\\3\end{pmatrix}$ but needs only to go up to $r = 2$ for learning $\begin{pmatrix}1\\2\\3\\4\end{pmatrix} \rightarrow \begin{pmatrix}27\\44\\51\\48\end{pmatrix}$; yet, $\begin{pmatrix}1\\2\\3\\4\end{pmatrix} \Leftrightarrow_{ord} \begin{pmatrix}27\\44\\51\\48\end{pmatrix}$.

In order to generalize well to unseen examples, the algorithm has to find the solution with the lowest r. The search for the lowest r seems justified from the

information theory point of view: the lowest the rank r, the highest the exploitation of patterns lying in the training set; if the training set is representative of the data on which predictions are made (which is a reasonable assumption), then these data contain the same regularities as the test set and in the same proportion; therefore, predictions based on this information will be correct with a probability equal to this proportion.

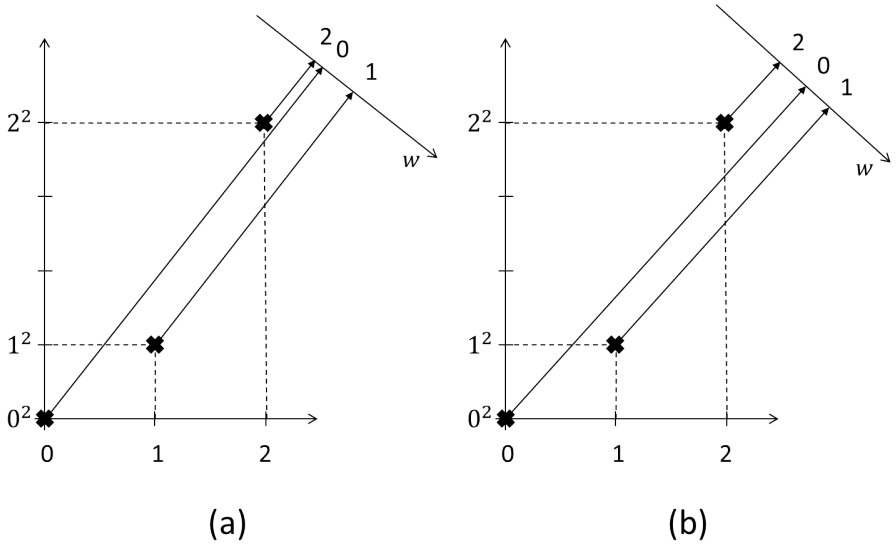

Fig. 1. Resolution of $\varphi_r\left(\begin{bmatrix}0\\1\\2\end{bmatrix}\right) w \Leftrightarrow_{ord} \begin{bmatrix}2\\0\\1\end{bmatrix}$ with the extension/adaptation algorithm. This algorithm ensures to find w with a minimal r (here $r = 2$) but does not ensure to find the most relevant solution for this value of r. **(a)** A candidate solution. **(b)** A relevant solution.

3.2 Search by Extension/Adaptation

Refinement of the Algorithm. The greedy algorithm takes a determined y and iterates on the dimensionality until it can solve the equation: it is able to extend the dimensionality of the search space but it does not adapt the target y.

We then modify the algorithm by adding an adapting phase at each stage. The extension/adaptation algorithm retains the properties of the greedy algorithm but ensures a minimal r (see Fig. 1.). Everything now depends on how to search $argmin_\gamma\{rank(\varphi_r(X)|\gamma)\}$ such that $\gamma \Leftrightarrow_{ord} y$.

Algorithm 2 Extension/Adaptation Approach

$r \leftarrow rank(X)$
while $r < rank(\varphi_r(X)|z), z = argmin_\gamma\{rank(\varphi_r(X)|\gamma)\} s.t. \gamma \Leftrightarrow_{ord} y$ **do**
  $r \leftarrow r + 1$
end while
return w s.t. $\varphi_r(X)w = z$

Choice of the Appropriate Vector. The improved algorithm allows expressing all the possible orders with the lowest r. The problem remaining now is that to each of these orders can correspond an infinity of sequences (due to the fact that the sequences are real valued). The right bias has to be found in order to select the most appropriate solution (see Fig. 1.).

In the case of SVM classification, it has been statistically proven that the most appropriate solution is the one that corresponds to the hyperplane with maximal margin [12]; In the case of instance-ranking, it has been proven that the most appropriate solution is the one that corresponds to a set of parallel hyperplanes among which the hyperplane lying between the closest classes has maximal margin [6]. Building on the existence of the latter proof, we can state that in the case of object-ranking, *the most appropriate solution is the vector on which the projection of all points results in a sequence in which the distance between the two closest points is maximal* (see Fig. 2. c.).

Indeed, we can consider object-ranking as a specific case of instance-ranking where there is only one instance per class. Then the most appropriate solution corresponds to the set of parallel hyperplanes among which the hyperplane lying between the closest classes, hence the closest points, has maximal margin. Since these hyperplanes are parallel, we can pass a vector through them at a normal angle: the intersections between the vector and the hyperplanes correspond to the projections of the points onto this vector, thus the distances between the hyperplanes correspond to the distances between the points of the resulting sequence.

3.3 Optimization Method

Herbrich and al. [6] extend the idea of large margin to ordinal learning and propose to solve instance-ranking problems by finding the hyperplane maximizing the margin between the two closest consecutive classes.

At first glance, it could be applied to object-ranking problems by setting each vector as a single instance of a class, as we did in the previous subsection for deriving the bias from instance-ranking problems to object-ranking problems. However, this method only allows finding hyperplanes with maximal margin between points that are explicitly defined as ordered pairs in the training set, whereas for solving our problem, we would need to find the hyperplane with maximal margin between the two closest points of the dataset even if the information about the order between them is not explicitly contained in the training

data. The solution we are looking for would then imply to consider every permutation of the intra-class elements to discover the permutation allowing finding a vector on which the distance between the two closest projected elements is maximized.

On the one hand, extending the idea of large margin to hidden object-ranking allows finding the appropriate solution but implies an exponential complexity with respect to the size of the biggest class due to the need for considering every permutation; on the other hand, considering only inter-class information is tractable but puts emphasis on the differences between classes, solving then the instance-ranking problem, not the object-ranking one. Instead, we look for a way to put emphasis on the order between elements, that is, to find the vector that lies at maximal distance from the closest vector yielding a wrong order.

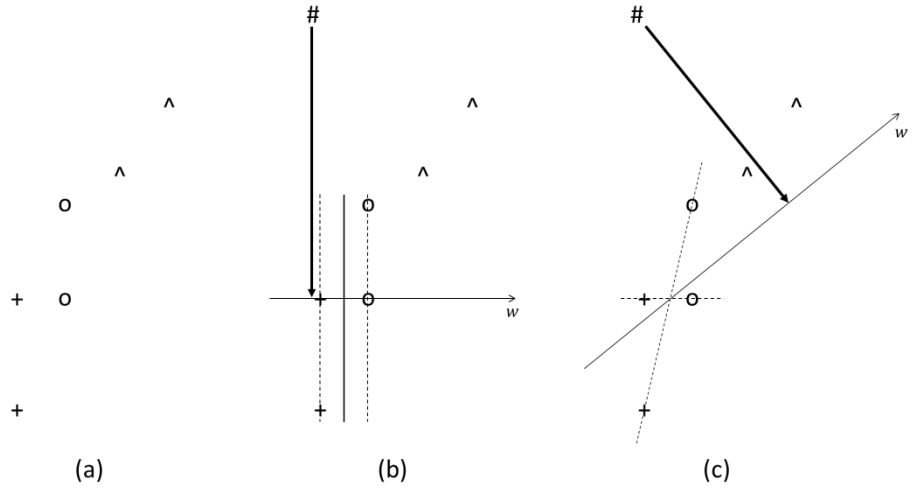

Fig. 2. Illustration of the importance of identifying a HORP for solving it with the right bias. **(a)** The training data. **(b)** The solution obeying the instance-ranking bias (traditional way of solving HORPs); **'#'** is predicted as the first element of the sequence and put in the class of **'+'**. **(c)** The solution obeying the object-ranking bias; **'#'** is predicted as the sixth element of the sequence and put in the class of **'^'**.

Resolution by Elimination. We derive a new training set from the original one by taking as new training examples the ordered pairs made explicit by the hinges between classes: let C_k and C_l be two different classes with $C_k \prec_p C_l$ for a given preference p; then for any $x_i \in C_k$ and any $x_j \in C_l$, our new training set will contain the vector $x_{ij} = \varphi_r(x_j) - \varphi_r(x_i)$.

In the case of a noise-free dataset, there exists a set of vectors whose dot product with any vector x_{ij} is positive. This set of possible solutions can be

found by elimination: each vector x_{ij} determines the direction of a hyperplane going through the origin and splitting the space into two half spaces; the possible solutions lie necessarily in the half space where the vector lies. As long as we suppose the dataset consistent, we can let each vector x_{ij} reduce the space of possible solutions: this space will never be empty.

Most of the vectors x_{ij} wont have any impact on the final space. Only vectors on the edges will. The goal is to find them so that the appropriate solution can be computed: it is the vector maximizing the angle with the hyperplanes normal to the edge vectors, or equivalently, minimizing the angle with these edge vectors (see Fig. 3. b.).

Setting for an SVM solver The resolution by elimination could be carried out analytically. However, it would be cumbersome as soon as the number of dimension increases. Moreover, it would need to be adapted in order to deal with noisy datasets.

Since solving a HORP boils down to finding the vector that minimizes the maximal angle, or, equivalently, that maximizes the minimal cosine, we propose to use SVM solvers instead. Indeed, the optimization problems implied by HORPs and SVMs are very close, and a HORP can be set in such way that an SVM solver can process it. Equations 1 and 2 express the HORP and SVM optimization problems respectively ('$*$' denotes the optimal arguments and y_n is the label associated to the training example x_n; $y_n \in \{-1, 1\}$).

$$w^*_{HORP} = argmax_w\{min_{i,j} \frac{w^t}{||w||} \frac{x_{ij}}{||x_{ij}||}\} \tag{1}$$

$$(w_{SVM}, w_0)^* = argmax_{w,w_0}\{min_n \frac{y_n(w^t x_n + w_0)}{||w||}\} \tag{2}$$

The setting is as follows: we normalize the size of any vector x_{ij}, label it as positive and add its opposite vector, normalized and negatively labelled, in the training set. Since the vectors have all the same magnitude and the positive and negative examples are symmetrical, the support vectors will coincide with the vectors lying on the edges, that is, the only vectors that have an impact on the final space. It follows that the weight vector yielded by the SVM solver will be w^*_{HORP} (Fig. 3. c.). Moreover, this setting allows taking advantage of the soft margin of the SVM to deal with noisy datasets. As well, as for the SVM, the basis functions can be replaced by properly designed kernel functions.

Time Complexity. The time complexity for solving an SVM optimization problem is $O(n^3)$ [13] with respect to the size of the input. In our setting, the size of the SVM input is $O(n^2)$ with respect to the size of the HORP input since it consists of pairs of training examples. Our method needs then $O(n^6)$ time for solving a HORP. This represents a serious drawback from a scalability point of view and has to be addressed in a future work.

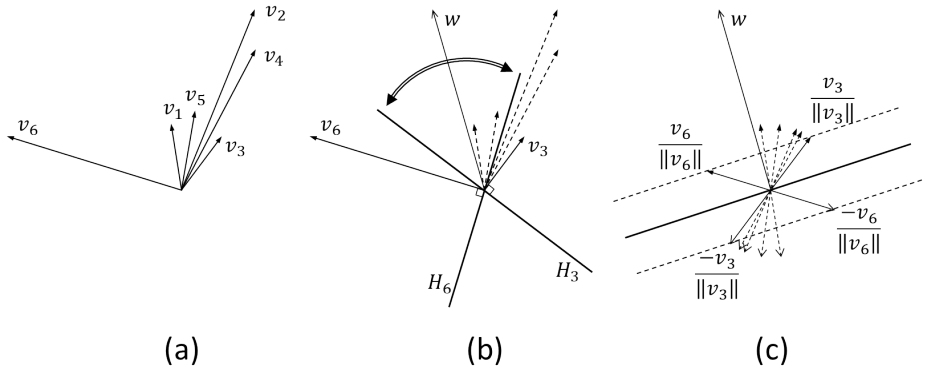

Fig. 3. (a) Six vectors v_h representing interclass relations $x_{ij} = \varphi_r(x_j) - \varphi_r(x_i)$ in a two-dimensional space. **(b)** Resolution by elimination: v_3 and v_6 are the edge vectors; the hyperplanes going through the origin and normal to them completely determine the set of possible solutions (shown by the double arrow); the appropriate solution, w, maximizes the angle to these hyperplanes, or equivalently, minimizes the angle with the edge vectors v_3 and v_6. **(c)** Resolution with an SVM solver; once normalized and symmetrized, the edge vectors correspond to support vectors.

4 Experiment

We conduct an experiment in order to test the extension/adaptation algorithm with an SVM solver for the adaptation step.

4.1 Methods

Dataset. The dataset consists of 54 instances, each describing a Tahitian pearl by 10 features corresponding to visual properties of their luster.

A description of these visual properties can be found in [14] and the way corresponding features are extracted from photographs of pearls is detailed in [15]. Briefly, three features quantify the appearance of specular reflectance and five features account for appearance of contrast between specular and diffuse reflectance, distinctness of reflected image, appearance of haze around the specular reflectance area, appearance of multiple reflectance on the successive layers of nacre the pearl is made of, and iridescence on the surface of the pearl. Two additional features, that do not *a priori* correspond to perceptual aspects of luster but are meant to test the impact of diffuse reflectance on luster determination, are mean saturation and chromaticity variance of the color on the surface of the pearl.

The instances are labeled by a human expert in 3 levels of luster quality. The classes are equally balanced.

Performance Evaluation. We use a 9-fold cross-validation method.

We run the experiment with 3 different settings. The first one is a traditional SVM multiclass classification; the second one implements the extension/adaptation algorithm with a genetic algorithm (population: 100; mutation rate: 0.2) for the adaptation step; the third one implements the extension/adaptation algorithm with a resolution by elimination carried out by an SVM solver for the adaptation step.

In all the settings, the experiment is run with and without feature selection. Feature selection is operated by picking the best combinations of features over the $2^{10} - 1$ possible non empty combinations for each setting. All the selected combinations turn out to involve specular reflectance, haze, deep reflectance, and chromaticity variance; none of them involves contrast or distinctness of reflected image (see [15] for a specification of the selected combinations).

Since the dataset is small, we test the statistical significance of the result; under the null hypothesis, the result would follow a binomial distribution of parameters $n = 54$ and $p = \frac{1}{3}$.

4.2 Results

Results are given in Table 1. The traditional multiclass classification reaches an accuracy of 77.7% with all features and 87% with selected features. The extension/adaptation algorithm with genetic algorithm reaches an accuracy of 90.7% with all features and 92.6% with selected features. The extension/adaptation algorithm with resolution by elimination reaches an accuracy of 94.4% with both all and selected features.

Let $Z \sim B(54, \frac{1}{3})$ under the null hypothesis; the p-value corresponding to an accuracy of 94.4%, which amounts to 51 correct predictions over 54, is equal to $Pr(Z \geq 51) < 10^{-19}$.

Table 1. Accuracy and 95% confidence interval estimate of different learners solving a HORP on a dataset of Tahitian pearls.

	All features	Selected features
SVM multiclass classification	$77.7\% \pm 11.1\%$	$87\% \pm 9\%$
Extension/adaptation with a genetic algorithm	$90.7\% \pm 7.7\%$	$92.6\% \pm 6.7\%$
Extension/adaptation with resolution by elimination	$94.4\% \pm 6.1\%$	$94.4\% \pm 6.1\%$

4.3 Discussion

The extension/adaptation algorithm with resolution by elimination reaches the highest accuracy. It confirms the theoretical insights. Moreover, with a p-value lower than 10^{-19}, this result is statistically significant, notwithstanding the small size of the dataset.

The extension/adaptation algorithm with genetic algorithm reaches a higher accuracy than the traditional multiclass algorithm even though the genetic algorithm returns the first solution it finds, which is not necessarily the best adapted; that is not surprising since the extension/adaptation makes it still ensure a minimal r.

Both SVM traditional multiclass classification and extension/adaptation with a genetic algorithm yield better results with selected features, while extension/adaptation with resolution by elimination makes no difference. Feature selection can be thought of as a way of experimentally finding the appropriate bias by letting the solution rely on the most relevant features, that is, on the features whose information they convey is the most representative of the information to capture for generalizing. From this point of view, we can interpret the results as a sign that the bias contained in our algorithm is better adapted to HORPs and makes therefore feature selection become less necessary.

5 Conclusion

Sum Up. In this paper, we propose an algorithm able to solve hidden object-ranking problems. The learning bias of object-ranking is explicited and the theoretical aspects of the algorithm are discussed. The algorithm is tested on scarce real data, yielding better results (94.4% of correct predictions) than traditional algorithms (92.6% of correct predictions for the best case) and suggesting that a feature selection step can be skipped without prejudicing accuracy.

Future Work. Even though the proposed algorithm is useful for scarce data because it allows exploiting the most information contained in object-ranking data, the optimization method we use makes it very sensitive to the size of the dataset. It would be worth finding an optimization method allowing the algorithm to scale.

References

1. Fürnkranz, J., Hüllermeier, E.: Preference learning: An introduction. [16] 1–17
2. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research (1999)
3. Fürnkranz, J., Hüllermeier, E.: Preference learning and ranking by pairwise comparison. [16] 65–82
4. Fürnkranz, J., Hullermeier, E.: Pairwise Preference Learning and Ranking. Machine Learning: ECML 2003 (2003) 145–156
5. Mitchell, T.M.: Machine learning, International Edition. McGraw-Hill Series in Computer Science. McGraw-Hill (1997)
6. Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: International Conference on Artificial Neural Networks. (1999) 97–102

7. Frank, E., Hall, M.A.: A simple approach to ordinal classification. In Raedt, L.D., Flach, P.A., eds.: ECML. Volume 2167 of Lecture Notes in Computer Science., Springer (2001) 145–156
8. Cardoso, J.S., da Costa, J.F.P.: Learning to classify ordinal data: the data replication method. Journal of Machine Learning Research **8** (2007) 1393–1429
9. Slater, P.: Inconsistencies in a Schedule of Paired Comparisons. Biometrika **48**(3/4) (1961) 303–312
10. Hudry, O.: On the complexity of Slater's problems. European Journal of Operational Research **203**(1) (2010) 216–221
11. Shmoys, D.B.: Approximation algorithms for np-hard problems. PWS Publishing Co., Boston, MA, USA (1997) 192–235
12. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (1998)
13. Abdiansah, A., Wardoyo, R.: Article: Time complexity analysis of support vector machines (svm) in libsvm. International Journal of Computer Applications **128**(3) (October 2015) 28–34 Published by Foundation of Computer Science (FCS), NY, USA.
14. Mondonneix, G., Chabrier, S., Mari, J.m., Gabillon, A., Barriot, J.P.: Tahitian Pearls' Luster Assessment. In McDonald, J., Markham, C., Winstanley, A., eds.: Proceedings of the 19th Irish Machine Vision and Image Processing conference, Maynooth, Irish Pattern Recognition & Classification Society (2017) 186–193
15. Mondonneix, G., Chabrier, S., Mari, J.m., Gabillon, A.: Tahitian Pearls' Luster Assessment Automation. In: Proceedings of the IEEE Applied Imagery Pattern Recognition Workshop : Big Data, Analytics, and Beyond. (2017)
16. Fürnkranz, J., Hüllermeier, E., eds.: Preference Learning. Springer-Verlag (2010)