Supervised Learning as Preference Optimization: A General Framework and its Applications

Fabio Aiolli and Alessandro Sperduti

Dept. of Pure and Applied Mathematics - Padova - Italy
Via Trieste 63, 35131 Padova, Italy

Abstract. Supervised learning is characterized by a broad spectrum of learning problems, often involving structured predictions, including classification and regressions problems, ranking-based predictions (label and instance ranking), and ordinal regression in its various forms. All these different learning problems are typically addressed by specific algorithmic solutions. In this paper, we show that the general preference learning model (GPLM), which is based on a large-margin principled approach, gives a flexible way to codify cost functions for all the above problems as sets of linear preferences. Examples of how the proposed framework can be effectively used to address a variety of real-world applications are reported showing the flexibility and effectiveness of the approach.

1 Introduction

Supervised learning is probably the most commonly used learning paradigm. In fact, given "experience" under the form of examples of a target function, i.e. input-output pairs, it allows to devise practical solutions through a large spectrum of learning algorithms. The need for such large spectrum of learning algorithms is, in part, due to the many real-world learning problems which, falling under the supervised umbrella, are characterized by heterogeneous tasks and problem-specific learning algorithms for their solution. These include classification and regression problems (including multi-label and multi-class classification, and multivariate regression), as well as ranking-based (either label or instance ranking) and ordinal regression problems. The typical approach followed to cope with these complex problems is to map them into a series of simpler, well-known settings and then to combine the resulting predictions. Often, however, these solutions lack a principled theory and/or require too much computational resources to be practical for real-world applications.

In this paper we review a general framework encompassing all these supervised learning settings, and we show that many supervised learning problems can actually be modeled through a set of order preferences over the predictions of the learner. This is done by considering both the different type of predictions and type of supervision involved in the problem to be solved. Specifically, four characterizing problem dimensions are taken into account. Firstly, the type of prediction which is expected. Secondly, the feedback that the nature provides.

Thirdly, the hypothesis space used in the learning process. Finally, the evaluation which determines how accurate is a learning device.

From a practical point of view, we show how all these supervised tasks can be addressed in a linear setting, where any problem formulation can be transformed into a binary problem defined on an augmented space, thus allowing the exploitation of very simple optimization procedures available for the binary case. We also stress the flexibility of the preference model which allows a user to optimize the parameters on the basis of a proper evaluation function. In fact, while in general the goal of a problem in terms of its evaluation function is clear, a crucial issue in the design of a learning algorithm is how to get a theoretical guarantee that the defined learning procedure actually minimizes the target cost function. One advantage of the framework reviewed in this paper is that it defines a very natural and uniform way to devise and code a cost function into a learning algorithm.

Examples of real-world applications are then discussed. First of all, a quick overview of a range of problems already successfully addressed by the framework is recalled. Then, two new applications are discussed in more detail. Specifically, the first application concerns the problem to select the best candidate for a job role. This is an instance ranking problem, where, however, only binary supervision from the past history is available. In addition to that, concept drift is present. We show how to cope with it, within the preference model, by exploiting dynamical selected committees of classifiers. The second application concerns a patent classification task, where patent applications have to be associated to primary categories as well as secondary categories. This is an example of label ranking task which cannot be properly addressed by a ordinal regression approach.

Related Work Some efforts have already been made to give general algorithms for label ranking tasks. In particular, in [1] the authors show that different label ranking problems can be cast as a linear problem which is solvable by a perceptron in an augmented feature space. In [2] the authors propose a setting in which a label ranking problem is map into a set of preference graphs and a convex optimization problem is defined to solve it. The preference model proposed in [3] generalizes on these two previous approaches, proposes a more flexible way to model cost functions for these problems, and gives a kernel based large margin solution for these kind of tasks. More recently, in [4] a large margin method to solve single-label problems with structured output is proposed. Even if applicable in principle, this approach does not seem directly applicable to solve label ranking tasks as it would require an optimization problem with a different constraint for each possible (label) ranking. Unfortunately, we are not aware of works aiming at generalizing instance ranking algorithms.

In Section 2, we review the general preference learning model (GPLM). Specifically, we show how the preference model generalizes the supervised learning setting by considering supervision as a partial order of (soft) constraints over the learner predictions. In addition, we show (Section 2.1) how the suggested generalization can be instantiated to well-known supervised learning problems. In the same section, we also discuss a linear model for the learner (Section 2.2) and issues about evaluation (Section 2.3). Quite general optimization procedures for training models within the proposed framework are also presented (Section 2.4). In Section 3, different application scenarios are described and discussed. In particular, it is discussed how the GPLM apply to a job candidate selection task (Section 3.1) and to a patent classification task (Section 3.2). Finally, in Section 4, some future extensions to the preference framework and final conclusions are presented.

2 GPLM: A General Model for Supervised Learning

In supervised learning, supervision is assumed to be provided according to an unknown probability distribution $\mathcal{D}$. Commonly, it is assumed to consist of pairs, objects (instances) and corresponding expected predictions (labels).

In this paper, we generalize a supervised learning setting by considering supervision as (soft) constraints over the learner predictions, that is constraints whose violation entails a cost for the solution. Specifically, we assume a learner makes its predictions on the basis of a set of parameters Θ, characterizing its *hypothesis space*. Each supervision constraint S then makes the learner suffering a cost $c(S|\Theta)$. It is easy to notice that this generalizes the above-mentioned case of supervision as instance-label pairs. In fact, this is obtained when a unitary cost is given to hypotheses generating incorrect labelings.

Two main types of supervised learning can be identified and generalized. In the *on-line* paradigm the probability distribution $\mathcal{D}$ is not assumed to be stationary and learning takes place in rounds. At each step the learner receives supervision and updates its parameters with the aim to minimize future costs. In *batch* learning, the stationary assumption on the probability distribution $\mathcal{D}$ is made and a training set $\mathcal{S} = \{S_1, \ldots, S_n\}$ is available at the beginning. Supervisions S_i are also supposed to be drawn *i.i.d.* from the same $\mathcal{D}$ and a single training session is made with the explicit goal to minimize the expected cost on the true distribution $\mathcal{D}$.

Different learning problems are often characterized by different types of prediction and supervision. Nevertheless, we can show that a broad set of them can be studied in a common framework, whose general setting is as follows. We consider a space $\mathcal{X}$ of objects (or instances) and a space $\mathcal{Y}$ of classes (or labels). In principle, either sets can be infinite. Moreover, we assume the hypothesis space, based on which the learner makes its predictions, to consist of *relevance functions*

$$u : \mathcal{X} \times \mathcal{Y} \to \mathbb{R},$$

depending on some set of parameters Θ. The goal of the learner is then to select a function $\hat{u}$ from its hypothesis space, which is "consistent" with the supervision in a sense that will depend on the particular setting.

As for now, we can already notice that given a particular relevance function $\hat{u}$, the instances in $\mathcal{X}$ can be ordered basing on their relevance once fixed a label $y \in \mathcal{Y}$, and similarly, labels in $\mathcal{Y}$ can be ordered basing on their relevance once fixed an instance $x \in \mathcal{X}$.

2.1 Prediction and Supervision

In this section, we show that by using the setting given above, it is possible to give a quite detailed taxonomy of the main supervised learning tasks on the basis of their expected prediction and supervision feedback. To this end, let us first recall the definition of order relations.

Definition A *partial order* is a pair $(\mathcal{P}, \succeq)$ where $\mathcal{P}$ is a set and $\succeq$ is a reflexive, antisymmetric and transitive binary relation. A *partial ranking* of length r is a partial order where the set $\mathcal{P}$ can be partitioned in r sets $\mathcal{P}_1, \ldots, \mathcal{P}_r$ such that $z \in \mathcal{P}_i$, $z' \in \mathcal{P}_j$, $i < j$, implies $z \succeq z'$ and no further information is conveyed about the ordering within subsets $\mathcal{P}_k$. A *full order* on $\mathcal{P}$ is defined as a partial ranking of length $|\mathcal{P}|$. We denote by $PO(\mathcal{P})$, $PR(\mathcal{P})$, and $FO(\mathcal{P})$ the set of partial orders, partial rankings and full orders over the set $\mathcal{P}$, respectively.

Label Rankings A first important family of supervised learning tasks is related to the ordering of the classes on the basis of their relevance for an instance, thus they are characterized by the fact that predictions should be based on a full order over the labels. This family of problems is referred to as *label rankings*. Supervision is in the form of partial orders over the classes. In our notation we have supervision $S \in PO(\mathcal{Y})$ and predictions in $FO(\mathcal{Y})$. Different settings can be obtained corresponding to different types of supervision. A few well-known instances are listed in the following:

Category Ranking (CR) In this setting, the goal is to order categories on the basis of their relevance for an instance. As an example, in a collaborative filtering setting, users could correspond to our instances and the different movies to our classes. Then, one could be interested into the ordering (by relevance) of the set of movies based on user preferences. This is trivially a particular case of label ranking where supervision is given as full orders over $\mathcal{Y}$.

Bipartite Category Ranking (BCR) In this task, supervision is given as two groups of classes and it is required to predict full orders in which the first group of classes is ranked over the second. As a leading example, in information retrieval, given a document, one might have to rank the available topics with the aim to return the most relevant topics on the top of the list. This is again a specific case of label ranking where supervision is given as partial rankings of length two. This task has been also referred to as category ranking in literature [5]. Here a different terminology is adopted to avoid confusion between these two different tasks.[1]

We may also be interested in predictions consisting of the most relevant classes, that is, of a prefix of the full order induced by the relevance function $u(\mathbf{x}, y)$. This family of tasks is commonly referred to as *classification* problems. They can however be considered as subcases of the BCR ranking task. A few examples of this kind of problems, listed by increasing specificity, is given here:

q-label classification (QC) In this task, the goal is to select the q most appropriate classes for a given instance, with q fixed. The supervision here is a partial ranking of length two where a set of exactly q labels are preferred over the rest.

Single-label classification (SC) In this well-known classification task, the goal is to select exactly one class (the most relevant) for an instance. This is a trivial subcase of QC with $q = 1$.

Instance Rankings Another interesting family of tasks is *instance rankings* where the goal is to order instances on the basis of the relevance of a given class. In our notation, predictions are in $FO(\mathcal{X})$ and supervision is given in the form $S \in PO(\mathcal{X})$.

The duality with respect to label rankings is self-evident. In principle, a corresponding problem setting could be defined for each of the label ranking settings. We can easily see that the well-known *(Bipartite) Instance Ranking* (IR) task, corresponds to BCR and is the one to induce an order such that a given set of instances is top-ranked. A natural application of this kind of prediction is in information retrieval, e.g. when listing the results returned by a search engine. Another interesting application is the one presented in Section 3.1 for job role selections. Similarly to BCR, here supervision consists of partial rankings (this time over the set $\mathcal{X}$) of length two. Another task which can also be considered in this family is the one to learn preference relations from a given set of ranked instances. For example, in information retrieval the task to learn preference relations on the basis of basic preferences given as pairs of documents [6].

The two families of tasks above can be considered *qualitative tasks* since they are concerned with order relations between instance-class pairs. On the other side, *quantitative tasks* are the ones which are more concerned with the absolute values of the relevance of instance-class pairs.

Quantitative Predictions Sometimes there is the necessity to do quantitative predictions about data at hand. For example, in binary classification, one has to decide about the membership of an instance to a class as opposed to rank

[1] Note that this task and the two that follow, are conceptually different from the task to decide about the membership of an instance. Here, supervision only gives *qualitative* information about the fact that some classes are more relevant than others.

instances by relevance. These settings are not directly subsumed by the settings presented above. As we will see this can be overcome by adding a set of thresholds and doing predictions based on these thresholds.

Multivariate Ordinal Regression *(MOR)* There are many settings where it is natural to rank instances according to an ordinal scale, including collaborative filtering, where there is the need to predict people ratings on unseen items. Borrowing the movie-related application introduced above, suitable ranks for movies could be given as 'bad', 'fair', 'good', and 'recommended'. With no loss in generality, we can consider the target space as the integer set $\mathcal{Z} = \{0, \ldots, R-1\}$ of R available ranks. Following an approach similar to the one in [7], ranks are made corresponding to intervals of the real line. Specifically, a set of thresholds $T = \{\tau_0 = -\infty, \tau_1, \ldots, \tau_{R-1}, \tau_R = +\infty\}$ can be defined and the prediction based on the rule

$$\hat{z} = \{i : u(\mathbf{x}, y) \in (\tau_{i-1}, \tau_i)\}.$$

Given the target label z, a correct prediction will be consistent with the conditions: $u(\mathbf{x}, y) > \tau_i$ when $i < z$ and $u(\mathbf{x}, y) < \tau_i$ when $i \geq z$. Note that, a different threshold set could also be used for different labels. The well-known *(Univariate) Ordinal Regression*(OR) [8, 9] task is a trivial subcase of MOR when a single class is available.

Multi-label Classification *(MLC)* In this task, it is required to classify instances with a subset (the cardinality of which is not specified) of the available classes. For us, it is convenient to consider this task as a MOR problem where only two ranks are available, relevant and irrelevant, and $\mathcal{Z} = \{0, 1\}$.

The well-known *Binary Classification* (BC) can be considered a subcase of OR with two ranks $\mathcal{Z} = \{0, 1\}$. Note that this task is considered here conceptually different from SC with two classes.

Clearly, the taxonomy presented above is not exhaustive but well highlights how many different kinds of structured predictions can be seen as simple constraints over the predictions of a learner. Specifically, they consist of constraints in conjunctive form where each basic preference is defined over the scoring values and/or a set of threshold values.

In particular, we can differentiate between two types of order preferences: *qualitative* preferences in the form

$$(\mathbf{x}_i, y_r) \rhd (\mathbf{x}_j, y_s)$$

telling that the value of $u(\mathbf{x}_i, y_r)$ should be higher than the value of $u(\mathbf{x}_j, y_s)$, and *quantitative* preferences in the form

$$(\mathbf{x}, y) \rhd \tau \text{ or } \tau \rhd (\mathbf{x}, y), \tau \in \mathbb{R}$$

relating the value of $u(\mathbf{x}, y)$ to a given threshold τ.

In Table 1, a summary of supervision obtained for the most general settings are presented. Particular instantiations to more specific problems are immediate.

Setting	**Supervision P-sets**
LR	$\{(\mathbf{x}, y_r) \rhd (\mathbf{x}, y_s)\}_{(\mathbf{x},y_r) \succeq_S (\mathbf{x},y_s)}$
IR	$\{(\mathbf{x}_i, y) \rhd (\mathbf{x}_j, y)\}_{(\mathbf{x}_i,y) \succeq_S (\mathbf{x}_j,y)}$
MOR	$\{(\mathbf{x}, y) \rhd \tau_i\}_{i<z} \cup \{\tau_i \rhd (\mathbf{x}, y)\}_{i \geq z}$

Table 1. Supervision of problems in Section 2.1. Label and instance rankings (LR and IR respectively), have a preference for each order relation induced by the supervision S. In ordinal regression (MOR), a preference is associated to each threshold and $z \in \mathcal{Z}$ is the rank given by the supervision.

2.2 A Linear Model for the Learner

In this work, we focus on a simple form of the relevance function, that is

$$u(\mathbf{x}, y) = w \cdot \phi(\mathbf{x}, y)$$

where $\phi(\mathbf{x}, y) \in \mathbb{R}^d$ is a joint representation of instance-class pairs and $w \in \mathbb{R}^d$ is a weight vector [4]. Note that this form generalizes the more standard form $u(\mathbf{x}, y) = w_y \cdot \phi(\mathbf{x})$ where different weight vectors are associated to different labels. In fact, let $|\mathcal{Y}| = m$, we can write:

$$w = (w_1, \ldots, w_m) \text{ and } \phi(\mathbf{x}, y) = (\underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{y-1}, \phi(\mathbf{x}), \mathbf{0}, \ldots, \mathbf{0}).$$

With this assumption, it is possible to conveniently reformulate an order constraint as a linear constraint. Let $T = \{\tau_1, \ldots, \tau_{R-1}\}$ be the available thresholds, in the qualitative case, given $\lambda \equiv (\mathbf{x}_i, y_r) \rhd (\mathbf{x}_j, y_s)$, we obtain

$$u(\mathbf{x}_i, y_r) > u(\mathbf{x}_j, y_s) \Leftrightarrow (w, \tau_1, \ldots, \tau_{R-1}) \cdot \underbrace{(\phi(\mathbf{x}_i, y_r) - \phi(\mathbf{x}_j, y_s), \underbrace{0, \ldots, 0}_{R-1})}_{\psi(\lambda)} > 0$$

Viceversa, in the quantitative case, given $\delta \in \{-1, +1\}$, we have

$$\delta(u(\mathbf{x}, y) - \tau_r) > 0 \Leftrightarrow (w, \tau_1, \ldots, \tau_{R-1}) \cdot \underbrace{(\delta\phi(\mathbf{x}, y), \underbrace{0, \ldots, 0}_{r-1}, -\delta, \underbrace{0, \ldots, 0}_{R-r-1})}_{\psi(\lambda)} > 0.$$

In general, we can see that supervision constraints of all the above-mentioned problems, can be reduced to sets of linear preferences of the form $\mathbf{w} \cdot \psi(\lambda) > 0$ where $\mathbf{w} = (w, \tau_1, \ldots, \tau_{R-1})$ is the vector of weights augmented with the set of available thresholds and $\psi(\lambda)$ is an opportune representation of the preference under consideration. The quantity

$$\rho_A(\lambda|\mathbf{w}) = \mathbf{w} \cdot \psi(\lambda)$$

will be also referred to as the margin of the hypothesis w.r.t. the preference. Note that this value is greater than zero when the preference is satisfied and less than zero otherwise. We will say that a preference λ is *consistent* with an hypothesis when $\rho_A(\lambda|\mathbf{w}) > 0$. The margin of an hypothesis w.r.t. the whole supervision S (a partial order), can be consequently defined as the minimum of the margins of preferences contained in S, i.e.

$$\rho(S) = \min_{\lambda \in S} \rho_A(\lambda).$$

This definition turns out to be consistent with definitions of the margin commonly used in different problems. In particular, the margin is positive if and only if the prediction is consistent with the supervision.

Summarizing, all the problems defined in the taxonomy in Section 2.1 can be seen as an homogeneous linear binary problem in a opportune augmented space. Specifically, any algorithm for linear classification (e.g. perceptron or linear programming) can be used to solve it, provided the problem has a solution.

2.3 Evaluation and cost optimization

More flexible cost functions, measuring the disagreement between a prediction (a full order) and the target supervision (a partial order), are often preferred to the mere consistency of supervision constraints when evaluating a learning machine. For example, there are settings where the evaluation of a non-perfect label ranking result can be better described as the number of categories which are misordered instead of simply as an error. To obtain more flexible cost functions, the structure of the supervision can be used to map the supervision (a partial order) into sets of preferences where different costs can be possibly associated to different preferences.
Similarly to [2], we consider a cost mapping $\mathcal{G} : S \mapsto \{g_1, \ldots, g_{q_S}\}$ where each possible supervision S generates a set of preference graphs. These direct graphs represent order preferences and are defined on subsets of related pairs in S. To augment the flexibility of the model we also allow to have different costs $\gamma(\lambda)$ associated to different preferences and this can be represented by associating costs to the arcs of the graphs.

Now, given a preference graph g, the associated cost suffered by an hypothesis can be defined as the maximum among the costs of its unfulfilled preferences, i.e.

$$c(g|\mathbf{w}) = \max\{\gamma(\lambda) | \lambda \in g \text{ not fulfilled by } \mathbf{w}\}. \quad (1)$$

Finally, the total cost suffered by an hypothesis $\mathbf{w}$ for the supervision S is defined as the cumulative cost over all the preference graphs, i.e.

$$c(S|\mathbf{w}) = \sum_{j=1}^{q_S} c(\mathcal{G}_j(S)|\mathbf{w}). \quad (2)$$

Using cost mappings as defined above, we are able to reproduce many of the different cost functions used for ranking problems. The reader can see [10] for

Methods	$l(\rho)$
Perceptron	$\max(0, -\rho)$
β-margin	$\max(0, \beta - \rho)$
Exponential	$e^{-\rho}$
Sigmoidal	$(1 + e^{\lambda(\rho-\theta)})^{-1}$

Table 2. Approximation losses as a function of the margin. $\beta > 0, \lambda > 0, \theta \in \mathbb{R}$ are external parameters.

several examples on how to give a cost map on standard supervised problems. Moreover, this model can be easily adapted to extend the approach in [4] for single-label classification to general rankings of structured output.

2.4 Learning with preferences

In earlier sections we have discussed the structure behind the supervision and how it can be modelled using preference graphs. Now, we see how to give learning algorithms which are able to optimize the associated cost functions.

Specifically, the purpose of a GPLM based algorithm will be to minimize costs $c(S|\mathbf{w})$. Since these are not continuous w.r.t. the parameter vector $\mathbf{w}$, they are approximated by introducing a continuous non-increasing loss function $l : \mathbb{R} \to \mathbb{R}^+$ approximating the indicator function. The (approximate) cost will be then defined by

$$\tilde{c}(S|\mathbf{w}) = \sum_{g \in \mathcal{G}(S)} \max_{\lambda \in g} \gamma(\lambda) l(\rho_A(\lambda|\mathbf{w})).$$

Examples of losses one can use are presented in Table 2.

The goal in batch learning is to find the parameters $\mathbf{w}$ such to minimize the expected cost over $\mathcal{D}$, the actual distribution ruling the supervision feedback, which is defined by

$$R_t[\mathbf{w}] = E_{S \sim \mathcal{D}}[c(S|\mathbf{w})].$$

Although $\mathcal{D}$ is unknown, we can still try to minimize this function by exploiting the same structure of supervision and as much of the information we can gather from the training set $\mathcal{S}$. The general problem can be given as in the following:

- Given a set $\mathcal{V}(\mathcal{S}) = \bigcup_{S \in \mathcal{S}} \mathcal{G}(S)$ of preference graphs
- Find a set of parameters $\mathbf{w}$ in such a way to minimize the functional

  $$\mathcal{Q}(\mathbf{w}) = \mathcal{R}(\mathbf{w}) + \mu \mathcal{L}(\mathcal{V}(\mathcal{S})|\mathbf{w}) \quad (3)$$

  where $\mathcal{L}(\mathcal{V}(\mathcal{S})|\mathbf{w}) = \sum_{S \in \mathcal{S}} \tilde{c}(S|\mathbf{w})$ is related to the empirical cost and $\mathcal{R}(\mathbf{w})$ is a regularization term over the set of parameters. Note that, for the solution to be admissible when multiple thresholds are used and there are constraints defined over their values (as in the ordinal regression settings), these constraints should be explicitly enforced.

The use of a regularization term problems of this type has different motivations, including the theory on regularization networks (see e.g. [11]). Moreover, we can see that by choosing a convex loss function and a convex regularization term (let say the quadratic term $\mathcal{R}(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$) it warranties the convexity of the functional $\mathcal{Q}(\mathbf{w})$ in Eq. 3 and then the uniqueness of the solution. Indeed, current kernel-based approaches defined for basic supervised learning tasks can be seen in this form when using the β-margin with $\beta = 1$. This suggests a new *universal* kernel method which is able to solve many complex learning tasks [12].

3 GPLM Applications

In this section, we briefly discuss three different previous works which are related to the preference learning framework presented in this paper.

The first application of the preference learning model to a label ranking problem was presented in [3]. The proposed kernel method is able to optimize any conjunctive set of label-based preferences, thus being suitable to any label-ranking problem. Moreover, it is possible to specify a coding matrix which allows the embedding of the set of labels in a smaller vector space. Finally, it is shown how, by means of this matrix, it is possible to generalize different output-coding schemes, including ECOC, (One Vs All) OvA, and (One Vs One) OvO.

In [10] there is a comparison of different cost mappings and loss functions for the ordinal regression problem in an on-line setting. The setting on-line and the huge size of data, have motivated a stochastic version of the general algorithm we presented in Section 2.4 for the batch setting. Quite interestingly, experiments have shown that using loss functions which nicely approximate the true evaluation function (0-1 loss function), like the sigmoidal loss for example, gave far better results w.r.t. hinge loss and similar.

Finally, in [13] there is an application of the model to an interactive classification context. In such contexts, differently from *autonomous* Text Categorization (TC) systems [14], the system, rather than taking a final categorization decision, is required to support a human expert who is in charge of taking this decision. A real-world application that perfectly fits in this scenario is patent classification [15–17], where experts at international patent offices are presented with patent applications that they need to classify against a large set of classes of existing patents, in order to check the novelty of the proposed invention. A system that ranks the available classes in terms of their estimated suitability to the document to be classified is extremely useful to these experts since they can thus concentrate on the top-ranked categories, pretty much as a Web searcher concentrates on the top-ranked documents returned by a search engine following a query. Clearly, providing supervision in the form of rankings is more onerous than providing a *crisp* membership value to a document (i.e. being relevant or irrelevant); Then, a mapping which uses membership information to generate subsumed ranking preferences, is used. For example, we can consider a preference like $c_r \rhd_d c_s$ (category c_r is preferred to category c_s for the document d) whenever d belongs to c_r and d does not belong to c_s. Experimental comparison between GPLM and standard SVM has been performed showing a large improvement w.r.t. the precision at recall ranking evaluation measure on the Reuters-21578 benchmark dataset.

In the following sections, two recent unpublished applications of the GPLM to a job candidate selection task and to a patent classification task, are presented.

3.1 Job candidate selection as a preferential task

In a candidate selection task for filling a job role, one or more candidates have to be selected from a pool of candidates. Without loss of generality, let assume that the $k \geq 1$ most suited candidate for the job are selected. This decision is taken by looking at each candidate professional profile. Moreover, we may assume that the number k of candidates to select is already known from the beginning. This last point is very important to model the problem. In fact, a candidate will be selected on the basis of which other candidates are in the pool. In other words, no decisions can be taken for a candidate without knowing who else is competing for the same position(s).

Assume the training set consists of past decisions about promotions to a given role. Then, for any of these decisions, we know which candidates were in a selection pool and how many and which candidates were selected for the job. Thus, it seems natural to interpret any past decision as a set of preferences in which the k selected candidates were preferred to the others. More formally, we define $C_t = \{c_1, \ldots, c_{n_t}\}$ to be the set of candidates for the job role (the pool) at time t, $S_t = \{s_1^{(t)}, \ldots, s_{k_t}^{(t)}\}$ the set of candidates which got the promotion, and $U_t = \{u_1^{(t)}, \ldots, u_{n_t-k_t}^{(t)}\}$ the set of candidates which were not selected. Thus, there is evidence that s_i was preferred to u_j for each $i \in \{1, \ldots, k_t\}$ and $j \in \{1, \ldots, n_t - k_t\}$. Using our notation, we can write $s_i \triangleright u_j$. Note that a selection having a pool of cardinality n_t and k_t candidates selected for the job, will introduce exactly $k_t \times (n_t - k_t)$ preferences. However, since $k_t \ll n_t$, the order of magnitude is still linear in the number of candidates.

Why not a simple binary task? One could think of a job role selection as a setting where for each candidate an independent decision is taken. In this case, at any time t, we would have exactly n_t independent decisions (e.g. a +1 decision, representing that the candidate was selected for the job role, and a -1 decision representing that the candidate was not selected for the job role). This could be modeled as a typical binary task where any of the 2^{n_t} different outcomes are possible. However, a job role selection is competitive in its nature, i.e. the choice of one candidate instead of another is not independent on the other's candidates potentials and only a fixed number of candidates can get the promotion. For this reason, the binary task does not seem to be the best choice. This will be confirmed in the experimental section where we compare the GPLM model against a binary SVM implementation. Finally, it should be noted that the problem tends to be highly unbalanced when considered as a binary problem. In fact, the number of promoted candidates is a very small percentage of the number of candidates which compete for the promotion. On the other side, GPLM does not make any additional assumption on the sign of the relevance function for different candidates but only on the order it induces. This should make the problem easier and more balanced.

GPLM with SVM In Section 2.2 we have seen that the preferential problem, i.e. the task to find a linear function which is consistent with a set of preferences, can be cast as a binary problem. Examples in this case become $\psi(\lambda) = \mathbf{s}_i - \mathbf{u}_j$ for each $\lambda \equiv s_i \rhd u_j$. Thus, a standard SVM algorithm applied to this new set of examples can be used to find a solution to the preferential problem.

Specifically, let $\Lambda = \{(S_1, U_1), \ldots, (S_T, U_T)\}$ be the sets involved in past promotions given as a training set for a given role, thus the SVM dual problem will be posed as

$$\begin{array}{ll} \arg\max_\alpha \sum_t \sum_{s_i \in S_t} \sum_{u_j \in U_t} \alpha_{ij}^{(t)} - \frac{1}{2} || \sum_t \sum_{s_i \in S_t} \sum_{u_j \in U_t} \alpha_{ij}^{(t)} \psi(s_i \rhd u_j) ||^2 \\ \text{s.t.} \qquad 0 \leq \alpha_{ij}^{(t)} \leq \mu, \end{array} \tag{4}$$

and the (primal) SVM solution which solves Eq. 3 will be in the form

$$\mathbf{w}_{SVM} = \sum_t \sum_{s_i \in S_t} \sum_{u_j \in U_t} \alpha_{ij}^{(t)} \psi(s_i \rhd u_j).$$

Note that the kernel computation in this case consists in computing a kernel between preferences (i.e. dot product between their vectorial representations). Nevertheless, this kernel can be easily reduced to a combination of simpler kernels between candidate profiles in the following way:

$$\begin{aligned} \tilde{k}(c_i^1 \rhd c_j^1, c_i^2 \rhd c_j^2) &= \langle \mathbf{c}_i^1 - \mathbf{c}_j^1, \mathbf{c}_i^2 - \mathbf{c}_j^2 \rangle = \langle \mathbf{c}_i^1, \mathbf{c}_i^2 \rangle - \langle \mathbf{c}_i^1, \mathbf{c}_j^2 \rangle - \langle \mathbf{c}_j^1, \mathbf{c}_i^2 \rangle + \langle \mathbf{c}_j^1, \mathbf{c}_j^2 \rangle \\ &= k(c_i^1, c_i^2) - k(c_i^1, c_j^2) - k(c_j^1, c_i^2) + k(c_j^1, c_j^2) \end{aligned}$$

where $k(c_i, c_j) = \langle \mathbf{c}_i, \mathbf{c}_j \rangle$ is the kernel function associated to the mapping used for the candidate profiles. In this way, we have reduced a preferential task into a binary task which can be easily solved by a standard SVM by just redefining the kernel function suitably.

Furthermore, using the SVM decision function $f_{SVM}(\lambda) = sgn(\langle \mathbf{w}_{SVM}, \psi(\lambda) \rangle)$ it is possible to determine if a given order relation is verified between any two candidates. However, in order to decide which candidates should be selected for a new event t, $k_t \times (n_t - k_t)$ calculations of the above defined function should be computed to obtain the relative order of candidates.

In the following, we show that the selection can actually be computed in linear time. To this end, we can decompose the weight vector computed by the SVM in the following way:

$$\begin{aligned} \mathbf{w}_{SVM} &= \sum_t \sum_{c_i \in S_t} \sum_{c_j \in U_t} \alpha_{ij}^{(t)} \psi(\mathbf{c}_i \rhd \mathbf{c}_j) = \sum_t \sum_{c_i \in S_t} \sum_{c_j \in U_t} \alpha_{ij}^{(t)} (\mathbf{c}_i - \mathbf{c}_j) \\ &= \sum_t \sum_{c_i \in S_t} \sum_{c_j \in U_t} \alpha_{ij}^{(t)} \mathbf{c}_i - \sum_t \sum_{c_i \in S_t} \sum_{c_j \in U_t} \alpha_{ij}^{(t)} \mathbf{c}_j \end{aligned}$$

$f_l(c_i)$	c_1	c_2	c_3
p_1	2.1	3.4	*4.0*
p_2	1.2	*2.9*	−1.7
p_3	4.7	*5.1*	2.9

Table 3. RANK function example with $P_t = 3$, $n_t = 3$. In this case, we would have RANK$(p_2) = 2$ as p_2 is the third ranked in the series {4.0,2.9,5.1}, while RANK$(x_2|p_2) = 0$ as c_2 is the best ranked in the row p_2, i.e. in the series {1.2,2.9,-1.7}. Maximum score values obtained over different candidates by each predictor are emphasized.

This decomposition allows us to decouple, in the computation of the relevance function for a new candidate, the contribution of candidate profiles given in the training set

$$\begin{aligned} f(c) = \langle \mathbf{w}_{SVM}, \mathbf{c}\rangle &= \textstyle\sum_t \sum_{c_i \in S_t} (\sum_{c_j \in U_t} \alpha_{ij}^{(t)}) \langle \mathbf{c}_i, \mathbf{c}\rangle - \sum_t \sum_{c_j \in U_t} (\sum_{c_i \in S_t} \alpha_{ij}^{(t)}) \langle \mathbf{c}_j, \mathbf{c}\rangle \\ &= \textstyle\sum_t \sum_{c_i \in S_t} (\sum_{c_j \in U_t} \alpha_{ij}^{(t)}) k(c_i, c) - \sum_t \sum_{c_j \in U_t} (\sum_{c_i \in S_t} \alpha_{ij}^{(t)}) k(c_j, c) \\ &= \textstyle\sum_t \sum_{c_i \in S_t} \alpha_i^{(t)} k(c_i, c) - \sum_t \sum_{c_i \in S_t} \alpha_j^{(t)} k(c_j, c) \end{aligned}$$

Hence the relevance function can be directly computed by post-processing the output of the SVM (the α vector) and then building a new model as follows

$$f(c) = \textstyle\sum_{c_s} \beta_s k(c_s, c)$$

where $\beta_s = \sum_{t:c_s \in S_t} \alpha_s^{(t)} - \sum_{t:c_s \in U_t} \alpha_s^{(t)}$. The new model defined by the β's can directly be used by a SVM, and it returns the correct relevance for any candidate.

Committee of GPLM models The job role selection task is particularly prone to concept drift phenomena as there is not any guarantee that criteria used in previous selections will be used again in the future (e.g. because people in charge to decide can change over time). In this cases, it can be useful to add experts to a committee as new chunk of data arrive. We assume that the size of the chunk with which we create new experts is fixed a priori. The problem here is then how to decide which predictors is better to use on never seen instances. This problem gives us an interesting example of application where it is useful to combine multiple rankings.

In the following, we describe a family of strategies to selecting and combining the (preferential) expert predictions in the committee. At each time t, we consider a set of candidates $C_t = \{c_1, \dots, c_{n_t}\}$ which represents the pool for a new job role selection and we consider the set of experts $P_t = \{p_1, \dots, p_{m_t}\}$ available a time t. Each individual expert, let say p_l, produces a score $f_l(c_i)$ for each candidate c_i, $i = 1, \dots, n_t$. Thus, a matrix similar to the one given as an example in Table 3, can be defined, where rows contain the values of the score functions $f_l(c_i)$ over the candidates for a given expert, and columns are the values of the score function $f_l(c_i)$ given by different experts on the same candidate.

Based on this matrix, it is possible to define two ranking functions which will be used in the following. Specifically, $\text{RANK}(c_i|p_l) \in \{0, \dots, n_t - 1\}$ which gives the rank of candidate c_i induced by the ordered series of score values given by the expert p_l (lower ranks mean highest values), and $\text{RANK}(p_l) \in \{0, \dots, m_t - 1\}$ which gives the relative rank of the expert p_l relatively to the ordered series of values obtained by using the maximum score obtained over all candidates, i.e. $\max_i f_l(c_i)$. This last value may be interpreted as a measure of confidence (or closeness) of the expert p_l w.r.t. the current pool.

The score matrix above is then used to define a family of strategies to combine the experts in a committee which can be used to give the final rank of candidates in the pool. To this aim, we propose to use the following general formula:

$$\text{score(c)} = \sum_{l=1}^{m_t} I_l(r) S_l(c)$$

where $I_l(r)$ is the *selector function* which selects the r most "reliable" experts, $r \leq m_t$, and $S_l(c)$ is the *scoring function* which represents "how much" each expert contributes to the overall prediction. Specifically, a dynamical selection of experts can be obtained by defining I_l as follows:

$$I_l(r) = \begin{cases} 1 \text{ if } \text{RANK}(p_l) \leq r - 1 \\ 0 \text{ otherwise} \end{cases}$$

returning the r experts estimated to be *closer* to the pool under consideration.

Note that, the temporal window method with window size r, can be implemented by slightly changing the selector function as follows:

$$I_l^{(tw)}(r) = \begin{cases} 1 \text{ if } m_t - r < l \leq m_t \\ 0 \; otherwise \end{cases}$$

The scoring function S_l is then chosen between the following:

$$S_l^{(1)}(c) = f_l(c)$$

$$S_l^{(2)}(c) = \text{RANK}(c|p_l)$$

Specifically, for the dynamical selection we obtain the following four strategies:

- **Sum of scores:** the sum of the scores of the r-closest experts by using $I_l(r)$ and $S_l^{(1)}$.
- **Maximum of scores:** the score of the closest expert by using $I_l(1)$ and $S_l^{(1)}$.
- **Minimal position:** the minimal rank position of the closest predictor by using $I_l(1)$ and $S_l^{(1)}$.
- **Sum of position:** the sum of the rank positions of the r-closest experts by using $I_l(r)$ and $S_l^{(2)}$.

Experimental setting Our data was collected from the Human Resources data warehouse of a bank. Specifically, we have considered all the events related to the promotion of an employee to the job role of director of a branch office (target job role). The data used ranges from January 2002 to November 2007. Each event involves from a minimum of 1 promotion up to a maximum of 7 simultaneous promotions. Since for each event a short list of candidates was not available, we were forced to consider as candidates competing for the promotion(s) all the employees which at the time of the event were potentially eligible for promotion to the target job role. Because of that, each event t typically involves k_t "positive" examples, i.e. the employees that were promoted, and $n_t \gg k_t$ "negative" examples, i.e. eligible employees that were not promoted. As already stated, k_t ranges from 1 to 7, while n_t ranges (approximately) from $3,700$ to $4,200$, for a total of 199 events, 267 positive examples, and $809,982$ negative examples[2]. Each candidate is represented, at the time of the event, through a profile involving 102 features. Of these features, 29 involve personal data, such as age, sex, title of study, zone of residence, etc., while the remaining 73 features codify information about the status of service, such as current office, salary, hours of work per day, annual assessment, skills self-assessment, etc. The features, and the way they are numerically coded, were chosen in such a way that it is impossible to recognize the identity of an employee from a profile. Moreover, we were careful in preserving, for each numerical feature, its inherent metric if present, e.g. the ZIP codes where redefined so that the geographic degree of proximity of two areas is preserved in the numerical proximity of the new codes associated to these two areas.

Results In order to test whether learning preferences was better than using a binary classifier where binary supervision is used for training and the score of the resulting classifier used to rank the instances belonging to the same event, we have performed a set of preliminary experiments on a representative subset of the whole dataset. The binary classifier was an SVM with gaussian kernel and the values to use for the hyperparameters were decided through a validation set. The gaussian kernel was used also for learning preferences. The results showed that it is better to learn preferences as the SVM obtained a total accuracy of 61.88% versus an accuracy of $76,20\%$ obtained for the approach based on learning preferences. The accuracy measures how many ranking relations are correctly predicted. The cost mapping we used for the GPLM is the one described in Section 3.1 that is each training selection was mapped into the set of preferences obtained between any "selected" profile and any "not selected" profile. The SVMlight [18] implementation has been used for all the experiments.

In Table 4 we have reported the cumulative performance obtained on the test set by the committees with dynamical selection. Note that, with the exception of the first case (chunk size 4), the best results are obtained by a single expert. This could seem a surprising result but it is probably due to the strong concept drift

[2] Note that the same employee can play the role of negative example in several events. Moreover, it might also be a positive example.

present in our dataset. This implies that choosing the “right” expert pays more than using a committee of experts. However, it is possible to clearly observe that the best aggregation rule is the ranking-based *min pos* rule which seems reasonable given that predictors are built by considering this ranking information.

3.2 Three-layered patent classification as a preferential task

In many applicative contexts in which textual documents are labelled with thematic categories, a distinction is made between the primary and the secondary categories that are attached to a given document. The primary categories represent the topics that are central to the document, while the secondary categories represent topics that the document somehow touches upon, albeit peripherally. For instance, when a patent application is submitted to the European Patent Office (EPO), a primary category from the International Patent Classification (IPC) scheme[3] is attached to the application, and that category determines the expert examiner who will be in charge of evaluating the application. Secondary categories are instead attached for the only purpose of identifying related prior art, since the appointed expert examiner will need to determine the novelty of the proposed invention against existing patents classified under either the primary or any of the secondary categories. For the purposes of EPO, failing to recognize the true primary category of a document is thus a more serious mistake than failing to recognize a true secondary category. Another instance is represented by the ACM Computing Reviews magazine[4], which publishes reviews of articles and books related to computer science, each classified according to one primary and several secondary categories from the ACM Computing Classification System[5]. Here the primary category determines in which section of the magazine the review is going to be printed, while secondary categories, together with the primary category, are used for facilitating search (e.g., allowing a user to search only the reviews belonging to a particular category). Again, for the purposes of ACM Computing Reviews, getting the primary category of a document wrong is thus a more serious mistake than failing to recognize a true secondary category.

We now propose GPLM models for the principled solution of the three-layered classification task. Let d denote a document having the set $P(d) = \{c_p\}$ (a singleton) as the set of its primary categories, $S(d) = \{c_{s_1}, \dots, c_{s_k}\}$ as the (possibly empty) set of its secondary categories, and $N(d) = \{c_{n_1}, \dots, c_{n_l}\}$ as the set of its non-categories, such that $C = P(d) \cup S(d) \cup N(d)$.

GPLM: Ordinal Regression for three-layered classification. One could be tempted to interpret the three-layered classification problem as a (multi-variate) ordinal regression (MOR) problem, i.e. the problem to give a rank from the ordered set {primary, secondary, non-category} to each category for an instance.

[3] `http://www.wipo.int/classifications/en/`
[4] `http://www.reviews.com/`
[5] `http://www.acm.org/class/1998/`

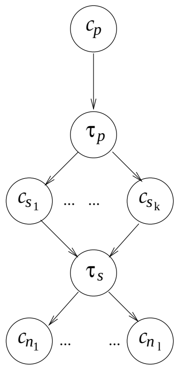

Fig. 1. GPLM mapping for ordinal-regression supervision.

In the following, we first give a GPLM mapping already presented in [19] which can be demonstrated to be equivalent to the ordinal regression method in [20]. Then, we discuss why, in our opinion, this setting does not exactly match the three-layered classification in the patent classification application. Our experiments, which will be reported in the following, will support this claim.

For ordinal regression, a GPLM model is built by considering two thresholds (see Fig. 1), let say τ_p and τ_s. For each training document, the relevance function of a primary category should be above the threshold τ_p, while the relevance function for any other category (either secondary or non-category) should be below the threshold τ_p. On the other hand, the relevance function of any secondary category should be above the threshold τ_s, while any non-category should be below the threshold τ_s. Summarizing, the preference graph for a given training document will be as in Figure 1. As a simple example, consider the set of categories $C = \{c_1, c_2, c_3, c_4, c_5\}$ and a training document d such that $P(d) = \{c_1\}$, $S(d) = \{c_2, c_3\}$, and $N(d) = \{c_4, c_5\}$. The set of preferences we generate is

$$\Lambda = \{(c_1 \rhd_d \tau_p), (\tau_p \rhd_d c_2), (\tau_p \rhd_d c_3), (c_2 \rhd_d \tau_s), (c_3 \rhd_d \tau_s), (\tau_s \rhd_d c_4), (\tau_s \rhd_d c_5)\}$$

Finally, three-layered classification will be performed by selecting the category reaching the highest relevance score as primary category, and among the others, all the categories reaching a relevance score above the threshold τ_s, as secondary categories.

At this point, we can discuss a little more about the OR-based preference model. In particular, in (multi-variate) ordinal regression, it is assumed that, for each document, the rank given to a category is independent from the rank given to other categories. This assumption would be reasonable when discriminating between relevant categories (primary and secondaries) and non-categories, since this is not a "competitive" decision, but is far less reasonable when one has to choose exactly one (the most relevant) among relevant categories as the primary category for a document, since in this case we actually have a "competitive" decision. Thus, in this last case, the choice of the primary category is strongly

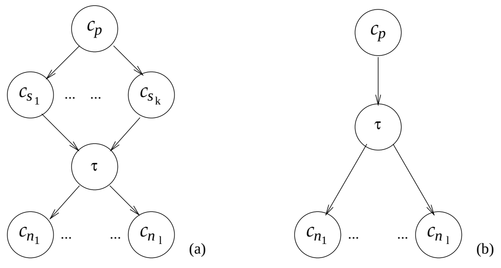

Fig. 2. GPLM mapping for supervision with (a) non-empty secondary category set and (b) empty secondary category set.

dependent on which are the relevant categories. This difference recalls the difference between single-label classification (which is competitive) and multi-label classification (which is not competitive) in multi-class classification tasks. In other words, requiring the relevance score for the primary category to be higher than a given threshold seems an unnecessary constraint which eventually could lead to a deteriorate overall performance.

GPLM: Ad-hoc mapping for three-layered classification. A variant of the ordinal regression scheme, which seems more suitable for the task of three-layered classification, can be built as follows. Let interpret the primary category as the most relevant among relevant categories. This constraint is introduced by the insertion of a set of qualitative preferences between the primary and all the secondary categories. Moreover, given the multi-label nature of the problem to discern the secondary categories with respect to the remaining categories, a single threshold τ on the relevance scores has to be added between the secondary categories and the non-categories. The categories reaching a relevance score above the threshold (apart from the one recognized as the primary category) will be predicted as secondary categories. See Figure 2(a) for a graphical representation of this kind of preference model. Note that whenever $S(d) = \emptyset$, this means that the relevance values for categories in $C \setminus P(d)$ are all below the threshold. To cope with this situation, the qualitative preferences can be collapsed into a direct quantitative preference between the primary category and the threshold. See Figure 2(b) for a graphical description of this kind of preference. As a simple example, consider the set of categories $C = \{c_1, c_2, c_3, c_4, c_5\}$ and a training document d such that $P(d) = \{c_1\}$, $S(d) = \{c_2, c_3\}$, and $N(d) = \{c_4, c_5\}$. The set of preferences we generate is

$$\Lambda = \{(c_1 \rhd_d c_2), (c_1 \rhd_d c_3), (c_2 \rhd_d \tau), (c_3 \rhd_d \tau), (\tau \rhd_d c_4), (\tau \rhd_d c_5)\}$$

Similarly, if d is instead such that $P(d) = \{c_1\}$, $S(d) = \emptyset$, $N(d) = \{c_2, c_3, c_4, c_5\}$, this will generate the set of preferences

$$\Lambda = \{(c_1 \rhd_d \tau), (\tau \rhd_d c_2), (\tau \rhd_d c_3), (\tau \rhd_d c_4), (\tau \rhd_d c_5)\}$$

Experimental setting We have evaluated our method on the WIPO-alpha dataset, a large (3GB) collection published by the World Intellectual Property Organization (WIPO) in 2003. The dataset consists of 75,250 patents classified according to version 8 of the International Patent Classification scheme (IPC). Each document d has one primary category (known as the *main IPC symbol* of d), and a variable (possibly null) number of secondary categories (the *secondary IPC symbols* of d). In order to avoid problems due to excessive sparsity, and consistently with previous literature [15], we only consider categories at the subclass level of the IPC scheme; each of the 630 IPC subclasses is thus viewed as containing the union of the documents contained in its subordinate groups.

WIPO-alpha comes partitioned into a training set Tr of 46,324 documents and a test set Te of 28,926 documents. Each category appears as *primary* category in at least 20 and at most 2,000 training documents, and of at least 10 and at most 1,000 test documents; each category appears as *secondary* category in at least 1 and at most 1,857 training documents, and of at least 1 and at most 1,621 test documents. The percentage of documents which are associated to at least one secondary category is 34% in the training set and 33% in the test set; most documents have thus no secondary categories attached. Categories that are attached as secondary categories to at least 10 documents are 62% of the total set of categories for the training set and 39,7% for the test set.

In our experiments we use the entire WIPO-alpha set of 75,250 documents. Each document includes a title, a list of inventors, a list of applicant companies or individuals, an abstract, a claims section, and a long description. Similarly to [15] we have only used the title, the abstract, and the first 300 words of the "long description" Preprocessing has been obtained by performing stop word removal, punctuation removal, down-casing, number removal, and Porter stemming. Vectorial representations have been generated for each document by the well-known "ltc" variant of cosine-normalized $tfidf$ weighting.

Two additional baseline methods have been defined. In the first baseline (dubbed "Baseline1'), a binary classifier is built for each $c \in C$ (by using as positive examples of category c_i all the documents that have c_i either as a primary or as a secondary category) and use the real-valued scores output by each classifier for d: the category for which the largest score has been obtained would be selected as the primary category, while the set of secondary categories could then be identified by optimizing a threshold for each individual category and selecting the categories whose associated classifier has returned a score above its associated threshold. We have indeed implemented this approach (by using standard binary SVMs). A slightly stronger approach (dubbed "Baseline2') consists in performing two different classification tasks, a first one (by means of a SVM-DDAG [21] single-label classifier h_P) aimed at identifying the primary category of d, and a second one (by means of a multi-label classifier h_S consisting of m SVM-based binary classifiers h_S^i, one for each category $c_i \in \{c_1, \ldots, c_m\}$) aimed at identifying, among the remaining categories, the secondary categories of d. The h_P classifier is trained by using, as positive examples of each c_i, only the training documents that have c_i as primary category. Each of the h_S^i is instead trained by using as positive examples only the training documents that have c_i as secondary category, and as negative examples only the training documents that have c_i as non-category (those that have c_i as primary category are discarded).

Results The results obtained for the different classifiers are summarized in Table 5. Ad-hoc evaluation measures have been used. In particular, the F_1 measure is computed for each pair of layers and then combined to form a single measure F_1^3. The first two rows report the performances of the two baseline classifiers. It can be observed that they have almost identical F_1^3 and are not so good in telling apart secondary categories from non-categories (F_1^{SN}). The third row reports the performance of the ordinal regression classifier, which turns out to have the best separation between primary and non-categories (F_1^{PN}) but a quite low performance on separating primary and secondary categories (F_1^{PS}). These results seem coherent with the analysis we have given in Section 3.2 as the separation between primary categories and non-categories is over-constrained by the ordinal regression model. The overall performance (F_1^3) slightly improves over the baseline classifiers. The fourth row reports the performance of the GPLM using an own implementation of the Kernel-Adatron [22] as optimizer. With respect to the baselines and the ordinal regression classifier, there is a clear improvement on F_1^{SN}, while F_1^{PS} decreases. Overall, however, there is a significant improvement in F_1^3.

4 Conclusion and future extensions

We have discussed a general preference model for supervised learning and applications to complex prediction problems, as job selection and patent classification. The first application is an instance-ranking problem in presence of concept drift. We suggested to use a committee of preferential models and proposed a set of strategies to select predictors and combine their predictions. The second application is a label-ranking problem where categories have to be associated to patents according to a three-layered structure (primary, secondary, non-category). A preference model ad hoc for this problem has been also proposed.

The interesting aspect of the proposed preference model is that it allows to codify cost functions as preferences and naturally plug them into the same training algorithm. In this view, the role of the cost functions resembles the role of kernels in kernel-machines. Moreover, the proposed method gives a tool for comparing different algorithms and cost functions on a same learning problem.

In the future, it would be interesting to explore extensions to the model, including: *(i)* Considering models with disjunctive preferences as it would increase the flexibility of the model. *(ii)* Studying new fast (approximate) algorithms when the number of examples/preferences are simply too large to be coped with standard learning algorithms. *(iii)* Extending the concept of preferences to preferences to a given degree, i.e. when a preference constraint have to be fulfilled with a given margin.

Chunk size	Rule	$r = 1$	$r = 3$	$r = 5$	$r = 7$	$r = 9$
4	max	82.15	72.40	71.68	68.83	69.18
	sum		83.50	80.53	80.04	77.71
	min pos		86.06	85.77	85.00	85.36
	sum pos		79.61	75.96	73.96	72.57
5	max	84.27	71.26	73.06	72.30	72.86
	sum		78.84	79.02	78.07	76.64
	min pos		80.73	82.73	81.82	81.22
	sum pos		76.97	73.08	70.79	70.73
6	max	86.83	67.70	66.11	65.17	67.72
	sum		79.02	72.48	67.90	67.87
	min pos		85.25	84.40	83.04	82.46
	sum pos		73.73	66.97	64.41	64.05

Table 4. Cumulative performance of dynamical selection committees with $r \in \{1, 3, 5, 7, 9\}$ and the four aggregation rules.

	F_1^{PS}	F_1^{SN}	F_1^{PN}	F_1^3
Baseline1	.851	.180	.482	.499
Baseline2	**.886**	.200	.464	.504
Ordinal Regression	.7847	.1774	**.5343**	.5077
GPLM Adatron	.8433	**.2138**	.5129	**.5206**

Table 5. Micro-averaged F_1^3 values obtained by the classifiers.

References

1. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: Advances in Neural Information Processing Systems. (2002)
2. Dekel, O., Manning, C., Singer, Y.: Log-linear models for label ranking. In: Advances in Neural Information Processing Systems. (2003)
3. Aiolli, F., Sperduti, A.: Preference learning for multiclass problems. In: Advances in Neural Information Processing Systems, MIT Press (2004)
4. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the Twenty-first international conference on Machine learning. (2004)
5. Crammer, K., Singer, Y.: A new family of online algorithms for category ranking. Journal of Machine Learning Research (2003)
6. Herbrich, R., Graepel, T., Bollmann-Sdorra, P., Ober-mayer, K.: Learning a preference relation for information retrieval. In: Proceedings of the AAAI Workshop Text Categorization and Machine Learning. (1998)
7. McCullagh, P., Nelder, J.: Generalized Linear Models. Chapman & Hall (1983)
8. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. In: Advances in Large Margin Classifiers. MIT Press (2000) 115–132,
9. Wu, H., Lu, H., Ma, S.: A practical svm-based algorithm for ordinal regression in image retrieval. In: Proceedings of the eleventh ACM international conference on Multimedia. (2003)
10. Aiolli, F.: A preference model for structured supervised learning tasks. In: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05), Houston, US (2005) 557–560
11. Evgeniou, T., Pontil, M., Poggio, T.: Regularization networks and support vector machines. Advances in Computational Mathematics **13** (2000) 1–50
12. Aiolli, F.: Large Margin Multiclass Learning: Models and Algorithms. PhD thesis, Dept. of Computer Science, University of Pisa (2004) http://www.di.unipi.it/˜ aiolli/thesis.ps.
13. Aiolli, F., Sebastiani, F., Sperduti, A.: Preference learning for category-ranking based interactive text categorization. In: Proceedings of the International Joint Conference on Neural Networks, Orlando, US (2007)
14. Lewis, D.D.: Evaluating and optmizing autonomous text classification systems. In: Proceedings of 18th ACM International Conference on Research and Development in Information Retrieval (SIGIR'95), Seattle, US (1995) 246–254
15. Fall, C.J., Törcsvári, A., Benzineb, K., Karetka, G.: Automated categorization in the International Patent Classification. SIGIR Forum **37** (2003) 10–25
16. Krier, M., Zaccà, F.: Automatic categorization applications at the european patent office. World Patent Information **24** (2002) 187–196
17. Larkey, L.S.: A patent search and classification system. In: Proceedings of DL-99, 4th ACM Conference on Digital Libraries, Berkeley, US (1999) 179–187
18. Joachims, T.: Making large-scale svm learning practical. In Schlkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. MIT-Press (1999)
19. Aiolli, F.: A preference model for structured supervised learning tasks. In: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05), Houston, US (2005) 557–560

20. Chu, W., Keerthi, S.S.: Support vector ordinal regression. Neural Comput. **19** (2007) 792–815
21. Platt, J.C., Cristianini, N., Shawe-Taylor, J.: Large margin DAGs for multiclass classification. In: Proceedings of the 11th International Conference on Neural Information Processing Systems (NIPS'99), Denver, US (1999) 547–533
22. Frie, T.T., Cristianini, N., Campbell, C.: The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers (1998)