Application of the Preference Learning Model to a Human Resources Selection Task

Fabio Aiolli, Michele De Filippo, Alessandro Sperduti, IEEE senior member

Abstract— **In many applicative settings there is the interest in ranking a list of items arriving from a data stream. In a human resource application, for example, to help selecting people for a given job role, the person in charge of the selection may want to get a list of candidates sorted according to their profiles and how much they are suited for the target job role. Historical data about past decisions can be analyzed to try to discover rules to help in defining such ranking. Moreover, samples have a temporal dynamics. To exploit this possibly useful information, here we propose a method that incrementally builds a committee of classifiers (experts), each one trained on the newer chunks of samples. The prediction of the committee is obtained as a combination of the rankings proposed by the experts which are "closer" to the data to rank. The experts of the committee are generated using the Preference Learning Model, a recent method which can directly exploit supervision in the form of preferences (partial orders between instances) and thus particularly suitable for rankings. We test our approach on a large dataset coming from many years of human resource selections in a bank.**

I. Introduction

Nowadays the diffusion of IT systems, and specifically of data warehouses, for collecting up to date business information, allows the management of a company to have easily access a huge amount of potentially useful data. However, because of the large amount and variety of data, human exploitation of this resource is quite difficult. Thus, approaches that automatically attempt to extract meaningful knowledge from data are welcome.

In this paper, we are interested in instance ranking problems, i.e. the same kind of problems addressed in [3]. Specifically, we address a human resource management task where employees of a company are evaluated for promotion to a given target job role. From a mathematical point of view, we can understand the input of this task as the set of profiles of candidates, while the expected output is a list of profiles sorted according to their fitness to cover the job role. We assume that historical data about past decisions can be retrieved and analyzed to try to discover rules to help in defining such ranking. The problem, thus, can be cast as a machine learning instance ranking task.

Moreover, samples of historical data are ordered with respect to the time and this ordering can leads to a hidden but possibly crucial temporal relationship. Note however that this kind of task does not meet the typical temporal data mining task as temporal data mining treats the case where each data is a sequence and items of the sequences have temporal relations.

In this paper, we give the following original contributions: *i)* we argue that the Preference Learning Model [2] is particularly suited for this kind of task, since its adoption allows to specify only preferences over candidates, without imposing an absolute (and independent) reference score, as happens when casting the problem as a classification problem; we show how to model the problem using preferences and, in addition, we demonstrate how such formulation can lead to the definition of a kernel over preferences that brings to an efficient treatment of the problem; *ii)* to cope with temporal dynamics, we propose a new way to exploit 'local' experts trained on data in a temporal window; specifically, we suggest to dynamically select, among the set of available experts at time t, the ones that appear to be the most competent to rank a given set of candidates. The degree of competence of an expert is defined according to its maximum score over the set of current candidates. This definition is reasonable in our context since the number of candidates which are actually promoted ("positive" examples) are far much less than the number of candidates that are not selected. Thus, the score of an expert is very correlated with the fitness of the candidate to cover the target job role. The selected experts are then used as members of a committee.

We present the above ideas within a general framework where both standard a priori defined committees, such as a committee using the most recent generated experts, and committees exploiting our dynamical selection strategy, can be seen as specific instances.

We have tested the proposed approach, versus the one exploiting committees using the most recent generated experts, on a large dataset coming from many years of human resource selections in a bank, obtaining quite promising results.

The paper is organized as follows. We start by introducing the Preference Learning Model [1] in Section II. Then, in Section III we argue why the task of selecting some candidates from a pool for a job role can be cast as a preference learning task. In the same section, we also discuss why modeling the task as a simple binary classification task is not adequate. In Section III-B, we discuss how the preference model for the selection task can be efficiently implemented by an SVM. After that, in Section IV we introduce a general framework for defining committees of preference learning experts where the membership of experts created along time to the committee is dynamically computed. Experimental results on a quite large dataset are presented in Section V.

Fabio Aiolli and Alessandro Sperduti are with the Department of Pure and Applied Mathematics, Padua University, Italy (email: {aiolli, sperduti}@math.unipd.it). Michele De Filippo is a freelance computer science consultant.

978-1-4244-2765-9/09/$25.00 ©2009 IEEE

The section on conclusions closes the paper.

II. The Preference Learning Model

The General Preference Learning Model (GPLM for short) [1] is a quite flexible framework which gives a general solution to many different machine learning problems, including classification, instance and label ranking.

In our context, the GPLM assumes the existence of a real-valued relevance function that, given a role, for each candidate c, returns a score $R(c)$ (the *relevance value*) which measures the degree to which the role can be covered by the candidate c. Given a role, the relevance function induces a ranking among candidates. A *preference* is a constraint involving candidates, that should be satisfied by the relevance function. Specifically, GPLM can deal with two types of preferences: (i) *qualitative preferences* $c_i \rhd c_j$ ("the candidate c_i is suitable for the role more than c_j does"), which means $R(c_i) > R(c_j)$; and (ii) *quantitative* preferences of type $c \rhd \tau$ ("the degree to which the candidate c applies to a role is at least τ", where $\tau \in \mathbb{R}$), which means $R(c) > \tau$, and similarly $\tau \rhd c$ which means $R(c) < \tau$.

In this learning framework, supervision for a task is provided as a set of preferences (of either types). These preferences constitute constraints on the form of the relevance function which has to be learned. The aim of the learning process is to learn the parameter of the relevance function which is as consistent as possible with these constraints. From now on, we will consider only qualitative preferences.

It should be stressed that in GPLM any set of preferences can be associated to any pair of candidates, so if no information about the relative ranking of two candidates for a role is available, no preference involving these two candidates need to be stated. This allows us to impose on the learner only constraints which are really needed.

We may instantiate the GPLM by assuming that the relevance of a candidate to a given role can be expressed in linear form, i.e.

$$R(c_i) = \langle \mathbf{w}, \mathbf{c}_i \rangle \tag{1}$$

where $\mathbf{c}_i \in \mathbb{R}^D$ is a vectorial representation of candidate c_i in a feature space, e.g. the mapping of a kernel function, and D is the size of this vector. The vector $\mathbf{w} \in \mathbb{R}^D$ is the weight vector (containing parameters to be learnt) associated to the role under consideration. For this case, very effective algorithms are given which explicitly attempt to minimize the number of wrong predictions in the training set. In fact, following Eq. 1, qualitative preferences can be conveniently reformulated as linear constraints. Specifically, let us consider the qualitative preference $\lambda \equiv (c_i \rhd c_j)$. This preference imposes the constraint $R(c_i) > R(c_j)$ on the relevance function R, which using Eq. 1 can be rewritten as $\langle \mathbf{w}, \mathbf{c}_i \rangle > \langle \mathbf{w}, \mathbf{c}_j \rangle$, or $\langle \mathbf{w}, (\mathbf{c}_i - \mathbf{c}_j) \rangle > 0$.

In general, the training data, in the form of preferences between candidates, can then be reduced to a set of linear constraints of the form $\mathbf{w} \cdot \psi(\lambda) > 0$ where $\mathbf{w}$ is the vector of weights for a given role and $\psi(\lambda) = (\mathbf{c}_i - \mathbf{c}_j)$ is the representation of preference $\lambda \equiv c_i \rhd c_j$. It follows that, any preference learning problem can be seen as a (homogeneous) linear problem in $\mathbb{R}^D$. Specifically, any algorithm for linear optimization (e.g., perceptron or a linear programming package) can be used to solve it, provided the problem has a solution.

Unfortunately, the set of preferences may generate a set of linear constraints that have no solution (i.e., the set of the $\psi(\lambda)$'s is not linearly separable), i.e., such that there is no weight vector $\mathbf{w}$ able to fulfil all the constraints induced by the preferences in the training set. To deal with training errors we may minimize, consistently with the principles of Structural Risk Minimization (SRM) theory [13], an objective function which is increasing in the number of unfulfilled preferences (the training error) while maximizing the *margin* $1/\|\mathbf{w}\|_2$ (where $\|\cdot\|_2$ denotes the 2-norm of a vector).

To this end, consider the quantity $\rho(\lambda \mid \mathbf{w}) \equiv \langle \mathbf{w}, \psi(\lambda) \rangle$ as the degree of satisfaction of a preference λ given the hypothesis $\mathbf{w}$. This value is greater than zero when the hypothesis is *consistent* with the preference and smaller than zero otherwise. Now, let us assume a training set $Tr = \{(\Lambda_i)\}_{i=1,\ldots,N}$, where Λ_i is the set of preferences associated to the i-th job selection relative to a given role. We aim at minimizing the number of preferences which are unfulfilled (and thus the number of wrong predictions) on the training set, while trying to maximizing the margin $1/\|\mathbf{w}\|_2$. Let $L(\cdot)$ be a convex, always positive, and non-increasing function, such that $L(0) = 1$. It is not difficult to show that the function $L(\rho(\lambda \mid \mathbf{w}))$ is an upper bound to the error function on the preference λ. Thus, a fairly general approach is the one that attempts to minimize a (convex) function like

$$D(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} \sum_{\lambda \in \Lambda_i} L(\rho(\lambda \mid \mathbf{w})) \tag{2}$$

where γ is a parameter that determines the relative contributions of the regularization and the loss in the objective function. Note that, if we adopt the *hinge loss* $L(\rho) = [1 - \rho]_+ = \max(0, 1 - \rho)$, the optimization required is the same as required by a binary SVM with a training set consisting of all $\psi(\lambda) \in \mathbb{R}^D$, one for each preference, each one taken as a (positive) example (see [1] for details).

III. 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).

Our training set consists of past decisions about promotions to a given role. For any of these decisions, we know which candidates were in the 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 constraints 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 $s_i \rhd u_j$. 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.

A. 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.

B. GPLM with SVM

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 also be cast as a SVM problem where examples becomes $\psi(\lambda) = \mathbf{s}_i - \mathbf{u}_j$ for each $\lambda \equiv s_i \rhd u_j$.

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

$$\begin{aligned} \arg\min_\alpha \quad & || \textstyle\sum_t \sum_{s_i \in S_t} \sum_{u_j \in U_t} \alpha_{ij}^{(t)} \psi(s_i \rhd u_j)||^2 \\ & - \textstyle\sum_t \sum_{s_i \in S_t} \sum_{u_j \in U_t} \alpha_{ij}^{(t)} \\ \text{s.t. } 0 \leq \alpha_{ij}^{(t)} \leq \gamma, & \end{aligned} \tag{3}$$

and the (primal) SVM solution which solves Eq. 2 is 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:

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

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:

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

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

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

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.

IV. COMMITTEE OF GPLM MODELS

One standard approach for dealing with data whose distribution changes with time is to define classifiers using newer data instances only (what we refer as *temporal window* methods in the following). Unfortunately, these methods may produce unstable classifiers since they do not consider the history of the changing distribution.

Moreover, the nature of the distribution drift is not always strictly related to time. In some tasks, the drift phenomenon can also show more regular behaviors. For example, some trends observed in the past can also be observed again in the future, e.g. by cycling over a small set of different trends. In these cases, it would be useful to maintain previously created predictors which can be more suitable for the new distribution of data. The problem here is how to decide which predictors is better to use on never seen instances.

To deal with this problem, in our approach, new experts are added to a committee incrementally as new chunks of data arrive. We assume that the size of the chunk with which we create a new expert is fixed a priori. Clearly, the right size to apply is dependent on the changing rate of the input distribution and this also depends on the degree of concept drift. So the optimal value of the chunk size can only be defined by a validation phase or on the basis of background knowledge on the problem at hand. An interesting work addressing this point is presented in [9].

In the following, we describe a family of strategies which define how to select and how to combine the expert predictions in the committee. Let's consider a set of candidates $C_t = \{c_1, \ldots, c_{n_t}\}$ representing the pool for a new job role selection, and the set of experts $P_t = \{p_1, \ldots, p_{m_t}\}$ available a time t. Each individual expert, let say p_l, produces a score $f_l(c_i)$ for each candidate c_i, $i = 1, \ldots, n_t$. Thus, a matrix similar to the one given as an example in Table I, 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, $\text{RANK}(c_i|p_l) \in \{0, \ldots, n_t - 1\}$ 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 $\text{RANK}(p_l) \in \{0, \ldots, m_t - 1\}$ 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 distance of the expert p_l from the current pool.

The score matrix, as defined 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:

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

which returns the r experts which are 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 among one of the following:

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

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

Specifically, for the dynamical selection we may 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 $S_l^{(2)}$.

$f_l(c_i)$	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 I

RANK FUNCTION EXAMPLE WITH $P_t = 3$, $n_t = 3$. IN THIS CASE, WE WOULD HAVE $\text{RANK}(p_2) = 2$ AS p_2 IS THE THIRD RANKED IN THE SERIES {4.0,2.9,5.1}, WHILE $\text{RANK}(x_2|p_2) = 0$ AS x_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.

V. EXPERIMENTAL RESULTS

In this section we report the experimental results we have obtained on a quite large dataset involving a task in Human Resources Management. On this dataset, we have compared the performance of our proposed approach versus a local expert (i.e., a predictor trained on the most recent chunk of examples), and committees composed of the r most recent trained experts using the four alternative ways of combining the predictions described above.

Chunk size	Rule	$r=1$	$r=3$	$r=5$	$r=7$	$r=9$
4	max	75.45	74.73	72.53	72.31	71.51
	sum		67.11	65.79	66.01	66.27
	min pos		78.00	77.79	78.79	78.44
	sum pos		66.18	64.78	64.45	64.90
5	max	77.14	76.26	74.61	73.92	73.27
	sum		70.21	68.51	68.59	68.82
	min pos		80.11	80.32	80.39	80.56
	sum pos		67.81	67.27	66.96	67.01
6	max	78.31	77.38	75.85	75.32	74.81
	sum		71.56	69.35	69.21	69.47
	min pos		80.82	81.27	81.70	82.56
	sum pos		67.89	66.85	66.77	67.04

TABLE II

CUMULATIVE CROSS-VALIDATION PERFORMANCE OF TEMPORAL WINDOW COMMITTEES WITH $r \in \{1, 3, 5, 7, 9\}$ AND THE FOUR AGGREGATION RULES.

Chunk size	Rule	$r=1$	$r=3$	$r=5$	$r=7$	$r=9$
4	max	76.38	68.35	68.35	69.15	69.38
	sum		75.63	76.22	74.63	73.60
	min pos		79.40	81.19	81.17	80.96
	sum pos		71.58	69.94	67.88	66.69
5	max	78.36	68.95	69.78	70.31	70.70
	sum		79.43	79.61	77.16	75.23
	min pos		82.15	82.45	81.64	81.56
	sum pos		75.95	72.08	69.59	67.79
6	max	79.82	70.87	71.71	72.04	72.68
	sum		81.24	78.45	76.11	74.87
	min pos		83.58	83.91	84.06	83.71
	sum pos		75.37	71.46	68.79	67.58

TABLE III

CUMULATIVE CROSS-VALIDATION PERFORMANCE OF DYNAMICAL SELECTION COMMITTEES WITH $r \in \{1, 3, 5, 7, 9\}$ AND THE FOUR AGGREGATION RULES.

A. Human Resources Data

Our data is 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 typically involves k "positive" examples, i.e. the employees that were promoted, and $N \gg k$ "negative" examples, i.e. eligible employees that were not promoted. As already stated, k ranges from 1 to 7, while N ranges (approximately) from $3,700$ to $4,200$, for a total of 199 events, 267 positive examples, and $809,982$ negative examples[1]. 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.

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

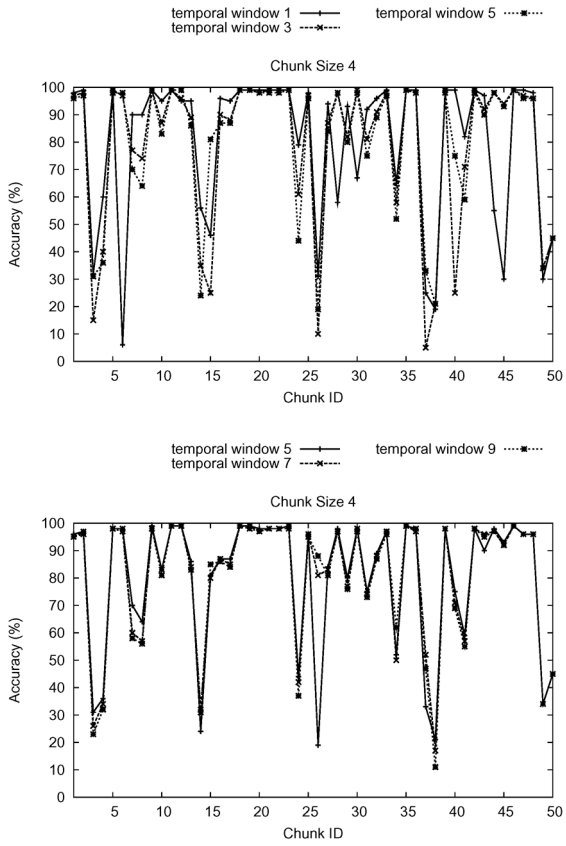

Fig. 1. Performance of *temporal window* committees with $r \in \{1, 3, 5, 7, 9\}$ on one split with chunk size 4. The committees, selected on the basis of the best cumulative performance on the validation set, use the minimal position aggregation rule.

The same employee can be represented by different profiles though time and events, as well as the same profile can be assigned to different employees. Not all the features are available at all times for all employees. When a feature is not available, the value 0 is used. The reason for using a specific value instead of a mark for a missing value is due to the fact that, in our case, a value for the feature was never generated, i.e. it does not exist.

B. 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 show that it is better to learn preferences, in fact, the SVM obtained a total accuracy of 61.88% versus an accuracy of $76,20\%$ obtained for the approach based on learning preferences. We recall that the accuracy measures how many ranking relations are correctly predicted. So, all the experiments we will describe in the following are based on the preference approach and use the whole dataset.

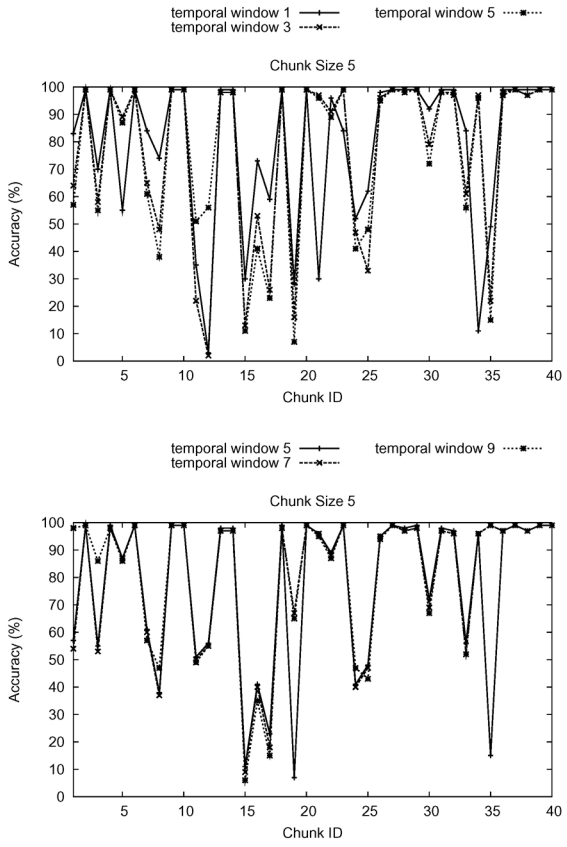

Fig. 2. Performance of *temporal window* committees with $r \in \{1,3,5,7,9\}$ on one split with chunk size 5. The committees, selected on the basis of the best cumulative performance on the validation set, use the minimal position aggregation rule.

To study the dependence of the concept drift on time, we have considered 3 alternative splits of the whole dataset, each composed of chunks of k consecutive events, with $k \in \{4,5,6\}$. To compare the different methods, given a split with a chunk size k, a k-fold cross-validation is performed. Specifically, at each cross-validation step, $k-1$ events are used for training and validation, and 1 for testing. When the "optimal" values for the hyperparameters are identified by using the validation set, a predictor is obtained using these values on the merge of the training and validation sets. This model is then evaluated on the test event. The same procedure is hence repeated k times, each time choosing a different event for testing. Finally, all the results are averaged.

As already mentioned, we have compared the performance of our approach, versus a committee using the r most recent experts, i.e. if the current chunk is the one with index i, the committee is composed of the experts $p_i,\ p_{i-1}, \ldots, p_{i-r+1}$ generated using chunks $i,\ i-1,\ \ldots, i-r+1$, respectively. We will refer to these committees with the name *temporal window* committees, since they are defined by using the most recent chunk of data in a temporal window of size r.

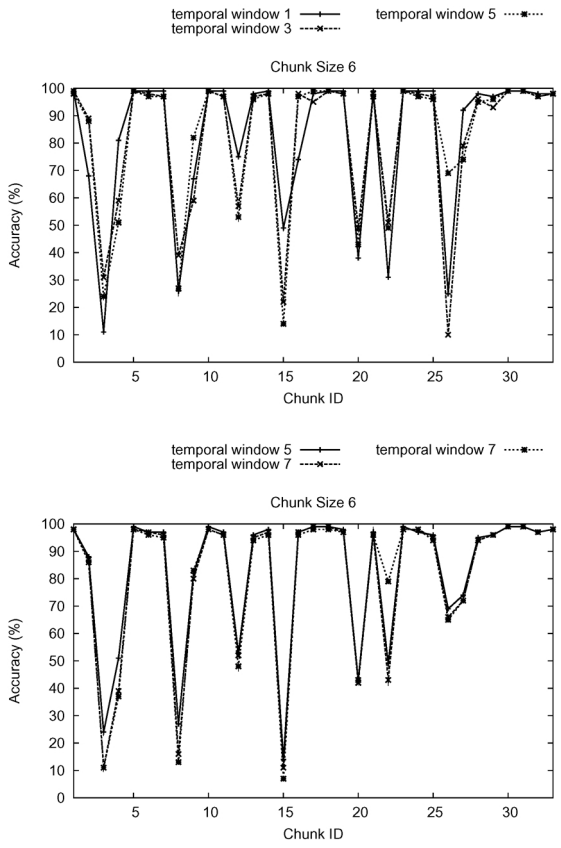

Fig. 3. Performance of *temporal window* committees with $r \in \{1,3,5,7,9\}$ on one split with chunk size 6. The committees, selected on the basis of the best cumulative performance on the validation set, use the minimal position aggregation rule.

In Table II we have reported, for each of the three splits defined above, the cumulative performance obtained by the cross-validation using the temporal window committees with $r \in \{1,3,5,7,9\}$ and the considered aggregation rules. It can be observed that, only if the *minimal position* aggregation rule is used, it helps to consider a committee, i.e. $r > 1$, instead of the most recent expert, i.e. $r = 1$.

In Table III we have reported, for each of the three splits defined above, the cumulative performance obtained by the cross-validation using the commitees with dynamical selection. These results confirm that the best aggregation rule is *minimal position*. Moreover, using this aggregation rule, the final results are consistently better independently from the chunk size and the value of r.

In Figures 1-3, for the three splits, we have reported the performance curves, obtained for a sample partition of the cross-validation, using temporal window committees with $r \in \{1, 3, 5, 7, 9\}$ and minimal position aggregation rule.

It can be observed that for all data splits, using a larger and larger window size (e.g., from $r = 5$ to $r = 9$) does not help too much, since there is no much difference in the performance curves.

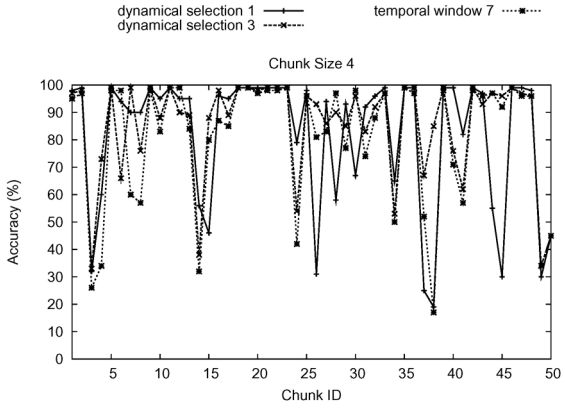

Fig. 4. Performance, on one cross-validation sample partition of the split with chunk size 4, of the best *temporal window* expert ($r = 7$), the best dynamically selected expert ($r = 1$), and the best committee with dynamical selection ($r = 3$).

In Figures 4-6, for the three splits, we have reported the performance curves, , obtained for a sample partition of the cross-validation, of the best temporal window committees versus the best dynamically selected expert ($r = 1$) and the best committee with dynamical selection ($r > 1$). All committees use the minimal position aggregation rule.

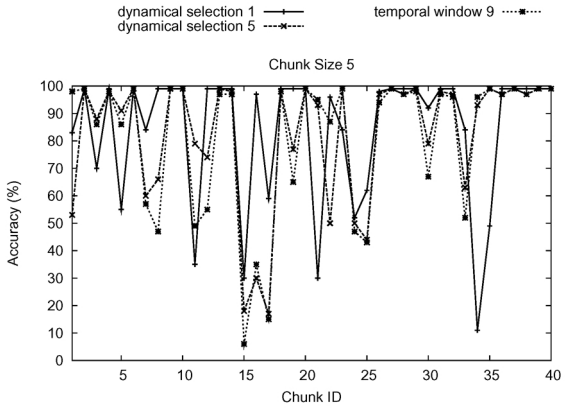

Fig. 5. Performance, on one cross-validation sample partition of the split with chunk size 5, of the best *temporal window* expert ($r = 9$), the best dynamically selected expert ($r = 1$), and the best committee with dynamical selection ($r = 5$).

In Figure 7, for the split of size 5 on one cross-validation sample partition, we show the first 6 experts selected by our approach when predicting the test data for each chunk of analyzed data. The oldest chunks have higher ID index. It should be recalled that the number of experts increases with the number of analyzed chunks. In fact, at the beginning (here

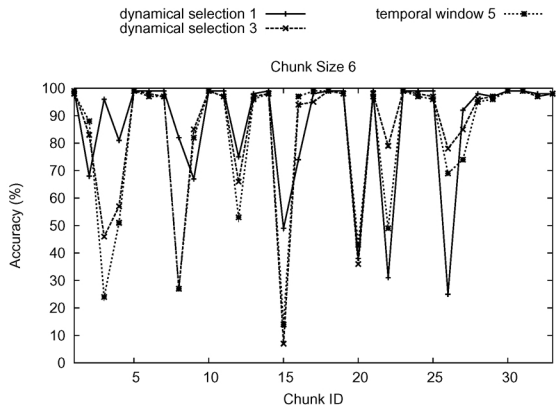

Fig. 6. Performance, on one cross-validation sample partition of the split with chunk size 6, of the best *temporal window* expert ($r = 5$), the best dynamically selected expert ($r = 1$), and the best committee with dynamical selection ($r = 3$).

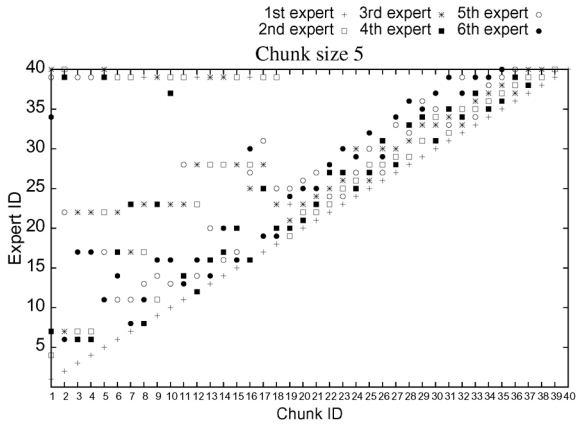

Fig. 7. Ranking of the first 6 experts for each chunks of size 5 on one cross-validation sample partition. The experts ID is reported on the ordinates for each chunk. The more recent chunks have lower ID number. At the beginning (chunk ID equal to 40), only one expert is available. As long as new chunks are analyzed, more and more experts are created and involved in the dynamical selection.

for chunk 33), only one expert is available, the current one. Then, for each new analyzed chunk of data, a new expert is generated and can be involved in the dynamical selection. The plot should be read by column, i.e. in correspondence of a chunk ID, the ID of the first 6 experts is reported on the ordinate. As expected, it can be observed, that the first expert is usually the current one. However, in some cases a different expert is selected. This, plus the fact that the dynamical selection of a single expert returns better results than using every time the current expert, confirms that the adopted selection rule is able to recognize when the current expert is not the best expert to use and a better expert, from the pool of available one, is selected. We can also observe that with the increase of the number of available experts, less and less recent experts are selected within the first 6. Finally, it is interesting to see that the selection of an expert for chunk i often implies that the same expert is selected also for chunks $i - 1$, $i - 2$, etc. In Figure 8, we report the same plot for the split of size 6. In this case, it can be observed

that the pattern of experts selection is more scattered. This could be due to the fact that the size of a single chunk is larger and thus there is higher probability to have concept drift within a single chunk.

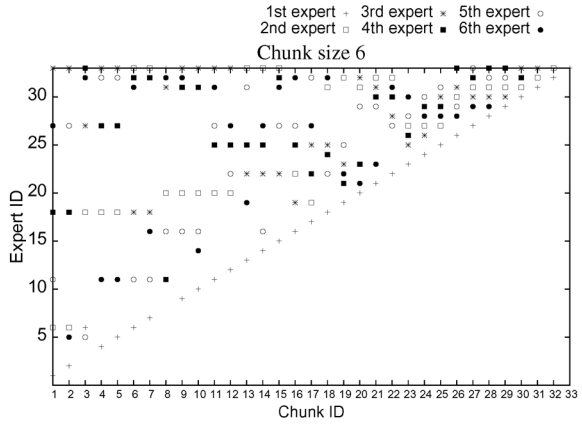

Fig. 8. Ranking of the first 6 experts for each chunks of size 6 on one cross-validation sample partition. The experts ID is reported on the ordinates for each chunk. The more recent chunks have lower ID number. At the beginning (chunk ID equal to 33), only one expert is available. As long as new chunks are analyzed, more and more experts are created and involved in the dynamical selection.

VI. Related Work

There are many works addressing the problem of non-stationary environments, especially when concerning binary classification tasks. Among these, quite related with our proposal are the approaches which fall under the *concept drift* umbrella. The most popular method to cope with drifting concepts in machine learning is by fixed or adaptive size temporal windows (see e.g. [6], [9]). In the case of fixed windows one has to make strong assumptions on the type of concept drift in the data. Adaptive window size based approaches try to overcome this problem by adapting the window size based on the extent of the current drift but they often use heuristics and parameter tuning which can cause a system to be unstable and generally difficult to tune. Other methods to deal with concept drift use example weighting [8], [7], [10]. A different kind of approaches to concept drift are the ones based on ensemble of classifiers. In [12] a boosting-like method is given to train a classifier ensemble from a data stream. The use of weighted learners is also suggested from the theoretical analysis given in [11].

Far less works has been done for the case of concept drift for tasks which are not binary classification tasks. For example, in [3] there is an application to instance ranking using a weighted ensemble. This method differs from ours in many aspects. First of all, the ranking algorithm of experts is done using a perceptron-like method and the way experts are combined is different since they perform a continuous reweighting of the experts as data arrives. Unfortunately, there is not public available implementation of this algorithm, which is quite involved and not so easy to implement from scratch.

VII. Conclusions

We have addressed a quite difficult ranking task involving a non-stationary data distribution. We suggested to approach the problem by using the preference learning framework for modeling the task. This decision was supported by experimental results showing that modeling the problem through a binary classification problem and then using the score of the classifier to rank the candidates was returning much worst performance than learning preferences. Moreover, we proposed to use a committee of experts where the members are dynamically selected from a set of experts generated on each new chunk of data. The dynamical selection is based on the "similarity" of an expert competence with the current data to be ranked. At our best knowledge, although dynamical selection of members in a committee has been studied in the past for classification tasks (e.g. [5], [4]), it is the first time that a ranking problem with non-stationary data distribution is addressed in this way, as well as it is the first time that our dynamical way of selecting the experts is proposed. It should be noted that our selection strategy seems to be particularly suited for ranking, since the score is highly correlated with the fitness of a candidate to cover the target job role.

We assessed our approach on a large dataset, comparing our proposal versus committees using the most recent generated experts. Our approach showed better performances.

VIII. Acknowledgments

We thank Maria Sole Franzin and Marco De Toni (Allos) for introducing us to the problem and providing the data.

References

[1] F. Aiolli. A preference model for structured supervised learning tasks. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, pages 557–560, Houston, US, 2005.

[2] F. Aiolli and A. Sperduti. Learning preferences for multiclass problems. In *NIPS*, 2004.

[3] H. Becker and M. Arias. Real-time ranking with concept drift using expert advice. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 86–94, New York, NY, USA, 2007. ACM.

[4] L. Didaci, G. Giacinto, F. Roli, and G. L. Marcialis. A study on the performances of dynamic classifier selection based on local accuracy estimation. *Pattern Recognition*, 38(11):2188–2191, 2005.

[5] G. Giacinto and F. Roli. A theoretical framework for dynamic classifier selection. In *ICPR*, pages 2008–2011, 2000.

[6] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pages 97–106, Madison, US, 2001.

[7] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3):281–300, 2004.

[8] R. Klinkenberg. Meta-learning, model selection, and example selection in machine learning domains with concept drift. In *LWA*, pages 164–171, 2005.

[9] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

[10] R. Klinkenberg and S. Rüping. Concept drift and the importance of example. In *Text Mining*, pages 55–78. 2003.

[11] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 449–456, New York, NY, USA, 2005. ACM.

[12] M. Scholz and R. Klinkenberg. Boosting classifiers for drifting concepts. *Intell. Data Anal.*, 11(1):3–28, 2007.

[13] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, US, 1998.