Learning Web Page Scores by Error Back-Propagation

Michelangelo Diligenti, Marco Gori, Marco Maggini
Dipartimento di Ingegneria dell'Informazione
Università di Siena
Via Roma 56, I-53100 Siena (Italy)
{michi, marco, maggini}@dii.unisi.it

Abstract

In this paper we present a novel algorithm to learn a score distribution over the nodes of a labeled graph (directed or undirected). Markov Chain theory is used to define the model of a random walker that converges to a score distribution which depends both on the graph connectivity and on the node labels. A supervised learning task is defined on the given graph by assigning a target score for some nodes and a training algorithm based on error back-propagation through the graph is devised to learn the model parameters. The trained model can assign scores to the graph nodes generalizing the criteria provided by the supervisor in the examples. The proposed algorithm has been applied to learn a ranking function for Web pages. The experimental results show the effectiveness of the proposed technique in reorganizing the rank accordingly to the examples provided in the training set.

1 Introduction

In this paper we propose an algorithm able to learn a generic distribution of values over the nodes of a graph with symbolic labels. In many applications, the input data is naturally represented by a labeled graph. However, the actual lack of adequate algorithms to directly process this structured data commonly forces to convert the input patterns to vectors of real numbers, loosing significant information. In particular, "Web like" data structures are common in many pattern recognition tasks and possible applications range from object recognition in segmented images to molecular analysis in bioinformatics, and so on.

Like in the PageRank algorithm [Page *et al.*, 1998], the score of a node is assumed to correspond to the probability that a random walker will visit that node at any given time. The score distribution computed by the random walker depends both on the connectivity graph and the labels assigned to each node. Markov Chain theory allows to define a set of constrains on the parameters of the random walker in order to guarantee some desired features like convergence and independence of the final distribution on the initial state. A learning algorithm which is based on error back-propagation through the graph is used to optimize the parameters of the random walker minimizing the distance of the scores assigned by the walker from the target values provided by a supervisor for some nodes in the graph.

Previous work in the same field achieved very limited results. For example, in [Kondor and Lafferty, 2002] the authors present an algorithm that processes only non labeled graphs, while the technique presented in [Chang *et al.*, 2000] is very application specific and features very limited generalization capabilities. In [Tsoi *et al.*, 2003] a method to modify PageRank scores computed over a collection of pages on the basis of a supervisor's feedback is proposed. This algorithm has limited approximation capabilities, since it controls only a term of the PageRank equation and it does not optimize how the scores flow though the connectivity graph.

The algorithm presented in this paper allows to learn target scores assigned as real numbers, and this makes it more general than the one proposed in [Diligenti *et al.*, 2003] which allows only to specify a set of nodes whose scores should be increased or decreased.

The paper is organized as follows. In the next section the web surfer model is introduced. Then, in section 3 the learning algorithm based on error back-propagation is described, while the experimental results are presented in section 4. Finally, in section 5 the conclusions are drawn.

2 Random Walks

We consider the model of a graph walker that can perform one out of two atomic actions while visiting the Web graph: jumping to a node of the graph (action J) and following a hyperlink from the current node (action l).

In general, the action taken by the surfer will depend on the label and outlinks from the node where it is located. We can model the user's behavior by a set of probabilities which depend on the current node: $x(l|q)$ is the probability of following one hyperlink from node q, and $x(J|q) = 1 - x(l|q)$ is the probability of jumping from node q.

The two actions need to specify their targets: $x(p|q, J)$ is the probability of jumping from node q to node p, and $x(p|q, l)$ is the probability of selecting a hyperlink from node q to node p. $x(p|q, l)$ is not null only for the nodes p linked directly by node q, i.e. $p \in ch(q)$, being $ch(q)$ the set of the children of node q in the graph $\boldsymbol{G}$. These sets of values must satisfy the probability normalization constraints $\forall q \in \boldsymbol{G}$ $\sum_{p \in \boldsymbol{G}} x(p|q, J) = 1$ and $\sum_{p \in ch(q)} x(p|q, l) = 1$.

The model considers a temporal sequence of actions performed by the surfer and it can be used to compute the probability $x_p(t)$ that the surfer is located in node p at time t. The probability distribution on the nodes of the Web is updated by taking into account the actions at time $t+1$ using the following equation

$$\begin{aligned} x_p(t+1) &= \sum_{q \in \boldsymbol{G}} x(p|q,J) \cdot x(J|q) \cdot x_q(t) + \\ &+ \sum_{q \in pa(p)} x(p|q,l) \cdot x(l|q) \cdot x_q(t) \ , \end{aligned} \tag{1}$$

where $pa(p)$ is the set of the parents of node p. The score x_p of a node p is computed using the stationary distribution of the Markov Chain defined by equation (1) (i.e. $x_p = \lim_{t\to\infty} x_p(t)$). The probability distribution over all the nodes at time t is represented by the vector $\boldsymbol{x}(t) = [x_1(t), \ldots, x_N(t)]'$, being N the total number of nodes.

An interesting property of this model is that, under the assumption that $x(J|q) \neq 0$ and $x(p|q,J) \neq 0$, $\forall p, q \in \boldsymbol{G}$, the stationary distribution $\boldsymbol{x}^\star$ does not depend on the initial state vector $\boldsymbol{x}(0)$ (see [Diligenti *et al.*, 2002]). For example this is true for the PageRank equation for which $x(J|q) = 1-d$, being $0 < d < 1$, and $x(p|q,J) = \frac{1}{N}$.

The random walker model of equation (1) is a simplified version of the model proposed in [Diligenti *et al.*, 2002], where two additional actions are considered by allowing the surfer to follow backlinks or to remain in the same node. The learning algorithm proposed in the next section can be easily extended to this more general model.

3 Learning the score distribution

The random surfer assigns the scores according to the probabilities $x(p|q,J)$, $x(J|q)$, $x(p|q,l)$, and $x(l|q)$. In most of the approaches proposed in the literature, these parameters are predefined or computed from some features describing the node p and/or the node q. For example, setting the parameters to $x(l|q) = d$, $x(J|q) = 1-d$, $x(p|q,J) = \frac{1}{N}$, and $x(p|q,l) = \frac{1}{ch(q)}$, we obtain the popular PageRank random surfer [Page *et al.*, 1998].

In the general case, the model has a number of parameters quadratic in the number of nodes and it is infeasible to estimate these parameters without any assumption to reduce their number. In particular, we assume that the labels of nodes can assume n distinct values $C = \{c_1, \ldots, c_n\}$ and that the surfer's behavior for any node p is dependent only on its label $c^p \in C$ and the labels of the nodes pointed by p. Therefore, the parameters of the model are rewritten as:

$$\begin{aligned} x(p|q,l) &= \frac{x(c^p|c^q,l)}{\sum_{k \in ch(q)} x(c^k|c^q,l)} \\ x(p|q,J) &= \frac{x(c^p|c^q,J)}{|c^p|} \\ x(l|p) &= x(l|c^p) \qquad x(J|p) \ = \ x(J|c^p) \end{aligned}$$

where $c^q \in C$ is the label of node q and $|c^p|$ is the number of nodes with label c^p. The parameter $x(c^p|c^q,l)$ represents the tendency of the surfer to follow a hyperlink from a node with label c^q to a node with label c^p, the parameter $x(c^p|c^q,J)$ is the probability of the surfer to jump from a node with label c^q to a node with label c^p and $x(l|c_i) = 1 - x(J|c_i)$ represents the probability that the surfer decides to follow a link out of a node with label c_i. In the following, we indicate as $\mathcal{P}$ the set of model parameters. $\mathcal{T}$, $\mathcal{J}$, $\mathcal{B}$ indicate *transition*, *jump* and *behavior* parameters, respectively. In particular, it holds that: $\mathcal{P} = \{\mathcal{T}, \mathcal{J}, \mathcal{B}\}$, $\mathcal{T} = \{x(c_i|c_j,l)\ i,j = 1,\ldots,n\}$, $\mathcal{J} = \{x(c_i|c_j,J)\ i,j = 1,\ldots,n\}$, and $\mathcal{B} = \{x(l|c_i)\ i = 1,\ldots,n\}$. Under these assumptions, the random surfer equation becomes,

$$\begin{aligned} x_p(t+1) &= \sum_{q \in \boldsymbol{G}} \frac{x(c^p|c^q,J)}{|c^p|} \cdot x(J|c^q) \cdot x_q(t) + \\ &+ \sum_{q \in pa(p)} \frac{x(c^p|c^q,l)}{\sum_{k \in ch(q)} x(c^k|c^q,l)} \cdot x(l|c^q) \cdot x_q(t) \ , \end{aligned} \tag{2}$$

Typically, the values $x(c^p|c^q,l)$ are initialized to 1, such that all links are equally likely to be followed. Because of the model assumptions, the stronger is the tendency of the surfer to move to a node with a specific label, the higher are the scores associated to the nodes with that label. These assumptions reduce the number of free parameters to $2 \cdot (n^2 - n) + n = 2 \cdot n^2 - n$, being n the number of distinct labels.

3.1 Learning the parameters

We assume that a supervisor provides a set of examples $\mathcal{E}$ in the graph $\boldsymbol{G}$, where each example $s \in \mathcal{E}$ is a node associated to a target score x_s^d.

Consider the following cost function for the scores computed by a surfer model using the set of parameters $\mathcal{P}$,

$$E^{\mathcal{P}} = \frac{1}{|\mathcal{E}|} \sum_{s \in \mathcal{E}} e_s^{\mathcal{P}} = \frac{1}{|\mathcal{E}|} \sum_{s \in \mathcal{E}} \frac{1}{2} \cdot (x_s^{\mathcal{P}} - x_s^d)^2 \ , \tag{3}$$

where $e_s^{\mathcal{P}}$ is called instant error for the node s. Such error takes into account the distance between the target score and the value returned by the ranking function.

We want to minimize the cost function (3) by gradient descent with respect to a generic parameter $w \in \mathcal{P}$, i.e.,

$$w' = w - \eta \frac{\partial E^{\mathcal{P}}}{\partial w} = w - \frac{\eta}{|\mathcal{E}|} \sum_{s \in \mathcal{E}} \frac{\partial e_s^{\mathcal{P}}}{\partial w} \ . \tag{4}$$

Using a procedure similar to the *back-propagation through time* algorithm (BPTT) [Haykin, 1994], commonly used in training recurrent neural networks, for each node s in the training set we unfold the graph starting from s at time T and moving backward in time through the path followed by the random surfer. The graph unfolding is basically a backward breadth-first visit of the graph, starting from the target node at time T, where each new unfolded level corresponds to moving backward of one time step. Each unfolded node is associated to the variable $x_q(T-t)$ where q is the corresponding graph node, t is the number of backward steps from the target node. Similarly, the parameters of the model are split into an independent set of parameters corresponding to the

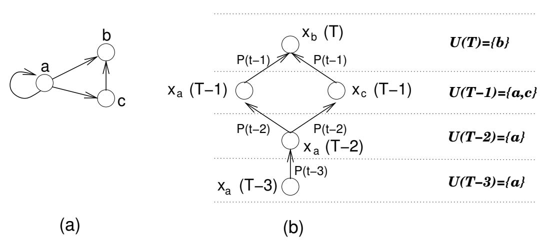


Figure 1: (a) A graph composed by three nodes. (b) The first four levels of the unfolding for the graph, starting from node b at time T. $\mathcal{U}(t)$ indicates the set of nodes at the level t of the unfolding (corresponding to the time step t). In the example, $\mathcal{U}(T) = \{b\}$, $\mathcal{U}(T-1) = \{a, c\}, \mathcal{U}(T-2) = \{a\}$ and $\mathcal{U}(T-3) = \{a\}$.

different time steps. We indicate with $\mathcal{P}(t)$ the model parameters at time t and with $w(t) \in \mathcal{P}(t)$ the value of any given parameter at time t. An infinite unfolding is perfectly equivalent to the original graph with respect to the computation of the score $x_s^{\mathcal{P}}(T)$ for a target node s. However, the unfolding takes into account only the score propagation due to the forward links, whereas the contribution of jumps (coming from any node in the graph) is neglected.

In the following, we indicate as $\mathcal{U}_s(t)$ the set of nodes contained in the unfolding centered on node s for the time step t. See figure 1 for an example of graph unfolding.

Thanks to the parameter decoupling in the unfolding, a parameter $w(t)$ influences only the scores at the following time step. Thus, the derivative of the instant error with respect to a parameter $w(t)$ can be written as,

$$\frac{\partial e_s^{\mathcal{P}}}{\partial w(t)} = \sum_{p \in \boldsymbol{G}} \frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)} . \tag{5}$$

Assuming that the parameters of the surfer are *stationary* (i.e. they do not depend on time: $w(t) = w \ \forall t$), the derivative with respect to a single parameter can be obtained summing all the contributions for the single time steps,

$$\frac{\partial e_s^{\mathcal{P}}}{\partial w} = \sum_{t=-\infty}^{T-1} \sum_{p \in \boldsymbol{G}} \frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)} . \tag{6}$$

The term $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t+1)}$ in equation (6) can not be directly computed, unless when $t = T-1, s = p$. In this latter case, it holds that,

$$\frac{\partial e_s^{\mathcal{P}}}{\partial x_s^{\mathcal{P}}(T)} = x_s^{\mathcal{P}}(T) - x_s^d . \tag{7}$$

Let $\delta_p(t)$ indicate $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)}$. If p is not a target node, then $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)}$ can be rewritten as,

$$\begin{aligned}
\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)} &= \delta_p(t) = \sum_{z \in \boldsymbol{G}} \frac{\partial e_s^{\mathcal{P}}}{\partial x_z^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_z^{\mathcal{P}}(t+1)}{\partial x_p^{\mathcal{P}}(t)} \approx \\
&\approx \sum_{z \in ch(p)} \frac{\partial e_s^{\mathcal{P}}}{\partial x_z^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_z^{\mathcal{P}}(t+1)}{\partial x_p^{\mathcal{P}}(t)} = \qquad (8) \\
&= \sum_{z \in ch(p)} \delta_z(t+1) \cdot \frac{x(c^z|c^p, l)(t)}{\sum_{k \in ch(p)} x(c^k|c^p, l)(t)} \cdot \\
&\cdot \ x(l|c^p)(t) = \\
&= \sum_{z \in ch(p)} \delta_z(t+1) \cdot P(p, t \to z, t+1) = \\
&= (x_s^{\mathcal{P}} - x_s^d) \cdot P(p, t \to s, T) ,
\end{aligned}$$

where $ch(p)$ is the set of children of node p in the unfolding and $P(p, t_1 \to z, t_2)$ indicates the probability of the surfer to follow a path in the unfolding connecting the node p at time t_1 to the node z at time t_2. Only the influence of transition parameters on the score of node p was taken into account in the propagation of the delta values.

Neglecting the influence of the jump parameters (i.e. taking into account only the influence due to link following) is an accurate approximation when the probability of following a link is much higher than the probability of jumping (like it is commonly assumed in PageRank). However, the experimental results show that such an approximation provides good results in the general case.

A generic node p at time t has influence over a supervised node s at time T proportional to the probability $P(p, t \to s, T)$ that the surfer will follow the path from the node p at time t to the central node s at time T. An implication of this result is that farther nodes will have less influence, given the exponential decrease of the probability of following a path with its length. In particular, a node p influences $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)}$ only if $P(p, t \to s, T) \neq 0$. Since $\mathcal{U}_s(T-t)$ is the set of nodes from which it is possible to reach node s at time T starting at time t, only for the nodes in this set it holds that $P(p, t \to s, T) \neq 0$. Thus, in equation (8) not all the score variables at time $t+1$ must be considered in the application of the chain rule. Inserting equation (8) into (6), and limiting the chain rule only to the nodes with a non-zero influence, yields,

$$\begin{aligned}
\frac{\partial e_s^{\mathcal{P}}}{\partial w} &= (x_s^{\mathcal{P}} - x_s^d) \sum_{t=-\infty}^{T-1} \sum_{p \in \mathcal{U}_s(t+1)} P(p, t \to s, T) \cdot \\
&\cdot \ \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)} .
\end{aligned}$$

Since the stable point of the system does not depend on the initial state, without loss of generality, the unfolding can be interrupted at time 1 (the graph is unfolded T times) assuming that T is large enough so that the state before time 1 has no effect on the final state (this is also expressed by the fact that the probability of following a path in the unfolding $P(p, t \to s, T)$ converges to 0 when the path length $T - t$ increases),

$$\begin{aligned}
\frac{\partial e_s^{\mathcal{P}}}{\partial w} &= (x_s^{\mathcal{P}} - x_s^d) \sum_{t=1}^{T-1} \sum_{p \in \mathcal{U}_s(t+1)} P(p, t \to s, T) \cdot \\
&\cdot \ \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)} . \qquad (9)
\end{aligned}$$

In the following sections, we compute the derivative $\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}$ to be inserted into equation (9) for a transition, jump and behavior parameter.

3.2 Learning the transition parameters

For a transition parameter $w = x(c_i|c_j, l)$, equation (2) shows that $\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial x(c_i|c_j,l)(t)} = 0$ if $c^p \neq c_i$. On the other hand if $c^p = c_i$, it holds that,

$$\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial x(c_i|c_j, l)(t)} = \sum_{q \in pa(p)} x(l|c^q)(t) x_q^{\mathcal{P}}(t) \cdot \cdot \frac{\partial \left[\frac{x(c^p|c^q,l)(t)}{\sum_{k \in ch(q)} x(c^k|c^q,l)(t)} \right]}{\partial x(c_i|c_j, l)(t)}$$

The application of the derivative rule for a fraction of functions, yields that when $c^p = c_i$,

$$\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial x(c_i|c_j,l)(t)} = \sum_{\substack{q \in pa(p) \\ q : c^q = c_j}} x(l|c^q)(t) \cdot x_q^{\mathcal{P}}(t) \cdot \qquad (10)$$
$$\cdot \frac{\sum_{k \in ch(q)} x(c^k|c^q,l)(t) - x(c^p|c^q,l)(t) \cdot |k \in ch(q) : c^k = c_i|}{\left[\sum_{k \in ch(q)} x(c^k|c^q,l)(t)\right]^2}$$

where $|k \in ch(q) : c^k = c_i|$ is the number of children nodes of q having label c_i.

Without loss of generality, it can be assumed that the system has already converged at time 0 (i.e. $x_q = x_q(t), t = 0, \ldots, T,\ \forall q \in \boldsymbol{G}$). Inserting equation (10) into (9) and, after removing all temporal dependencies from the parameters and from the node scores under the assumption of model stationarity and convergence at time 0, the derivative of the instant error for the considered transition parameter is obtained.

Averaging over all the examples, the model parameters can be updated by gradient descent according to equation (4). The new parameters can be used to compute a new score distribution over the nodes $x_p^{\mathcal{P}}$. Function optimization and score estimation can be iterated through multiple epochs, till a termination criterion is satisfied.

3.3 Learning the jump parameters

For a jump parameter $w = x(c_i|c_j, J)$, equation (2) shows that it holds that,

$$\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial x(c_i|c_j, J)(t)} = \begin{cases} 0 \ \ if \ \ c^p \neq c_i \\ \frac{x(J|c_j)(t)}{|c_i|} \cdot \sum_{\substack{q \in \boldsymbol{G} \\ q : c^q = c_j}} x_q^{\mathcal{P}}(t) \ \ if \ c^p = c_i \end{cases}$$

where $|c_i|$ indicates the number of nodes in the graph $\boldsymbol{G}$ with label c_i. The previous equation can be inserted into (9) and, removing all temporal dependencies under the assumption of model convergence and stationarity, yields the derivative of the instant error with respect to a jump parameter.

3.4 Learning the surfer action bias

Following a procedure similar to that shown for the transition and jump parameters, we start computing the derivative $\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial x(l|c_i)(t)}$ for a single parameter.

In particular, equation (2) shows that,

$$\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial x(l|c_i)(t)} = \sum_{\substack{q \in pa(p) \\ q : c^q = c_i}} \frac{x(c^p|c_i, l)(t)}{\sum_{k \in ch(q)} x(c^k|c_i, l)(t)} \cdot x_q^{\mathcal{P}} - - \sum_{\substack{q \in \boldsymbol{G} \\ q : c^q = c_i}} \frac{x(c^p|c_i, J)(t)}{|c^p|} \cdot x_q^{\mathcal{P}}(t)\ , \qquad (11)$$

where the surfer model is assumed to respect the probabilistic constrains at every time step, i.e. $x(J|c_i)(t) = 1 - x(l|c_i)(t)$.

Merging equation (9) and (11) under the assumption of model stationarity and convergence at time 0, yields the derivative of the instant error with respect to a behavior parameter. All parameters can be simultaneously updated as described in section 3.2, 3.3 and 3.4 to train the model. After the updating, the jump and behavior parameters must be normalized to respect the probabilistic constrains.

Finally, using an EM-style algorithm [Dempster *et al.*, 1977] the adapted surfer model can re-surf the Web graph yielding a new estimate of the scores. The procedure can be repeated till a stop criterion is satisfied. Since the parameters are estimated from a few levels deep unfoldings, only a small set of nodes must be considered at each iteration. The main overhead of the proposed learning algorithm is the full PageRank computation that must be completed at each step. Our experiments show that usually only very few iterations are needed to converge.

4 Experimental results

The experiments were performed on two datasets, which were collected by focus crawling the Web [Diligenti *et al.*, 2000], on the topics "wine" and "Playstation", respectively. The "wine" dataset contains $1.000.000$ documents, while the "Playstation" one 500.000. The documents were clustered using a hierarchical version of the k-means algorithm [Duda and Hart, 1973]. In particular, in two different runs, we clustered the pages from the dataset on topic "Playstation" into 25 sets, whereas the dataset on topic "wine" was clustered into 25 and 100 sets.

Two topic-specific search engines were built containing the downloaded documents. The rank of each page in the dataset was computed using a random walker producing the standard PageRank scores. We tested the quality of the ranking function by submitting a set of queries to the search engine whose results were sorted by the scores assigned by the random surfer. By browsing the results, we found pages which were authorities on the specific topic but were (incorrectly) not inserted in the top positions of the result list. Such pages were selected as examples of pages which should get a higher rank (positive examples). A target score equal to one (the maximum reachable score according to the model) was assigned to the positive examples. On the other hand, we found pages having a high rank which were not authorities for the topic. Such pages were selected as negative examples and a target score equal to zero was assigned to those pages. The learning algorithm, as previously described was applied to these datasets, using the selected examples. Only a small set of examples (3-15) was typically labeled as positive or negative in any single experiment.

4.1 Analysis of the effects of learning

This first group of experiments aims at demonstrating that the surfer model as learned by the proposed training algorithm, could not be generated by a non-adaptive schema as the focused versions of PageRank proposed in the literature.

In figure 2, we show the values of the transition, jump and behavior parameters for each cluster, resulting from the

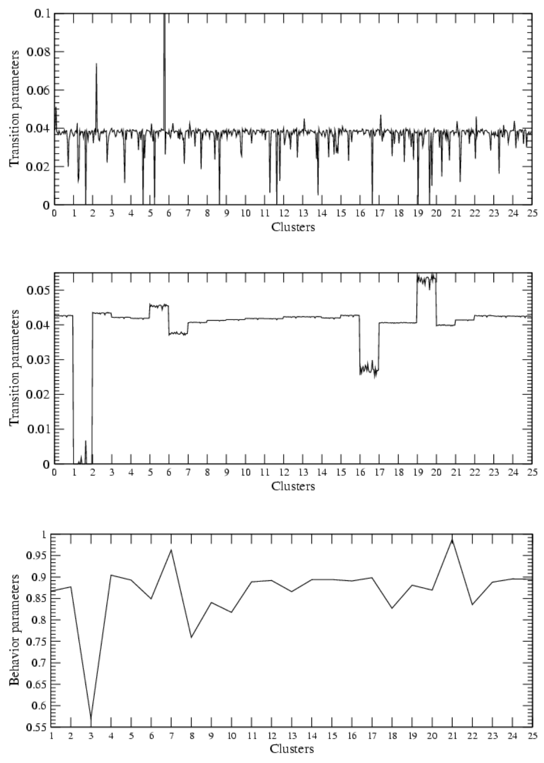
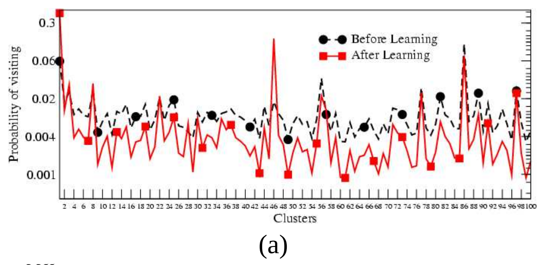


Figure 2: Values of the transition, jump and behavior parameters after learning for the "Playstation" dataset. For each cluster (x axis), all the n parameters to the other clusters are sequentially plotted.

training sessions for a surfer on the topic "Playstation". The learned parameters represent a complex interaction of the resulting surfer with the page clusters where a few clusters are strongly boosted and a few other strongly demoted. This behavior (expecially targeted demotions) can not be expressed by the focused versions of PageRank [Richardson and Domingos, 2002; Diligenti *et al.*, 2002], which simply boost all the parameters leading to the clusters of relevant pages (according to the simplistic assumption that relevant pages are more likely to point to other relevant pages). This demonstrates that the trained surfer is able to exploit hierarchies of topics to model the complex path leading to good/bad pages.

In figure 3-(a) and 3-(b), there are plotted the probabilities that the surfer is located in a page of a specific cluster. In particular we show the results for the dataset on topic "wine" clustered into 100 groups and a random graph with 200.000 nodes clustered into 25 groups. The learning algorithm is able to increase the likelihood of the random surfer to visit pages belonging to a set of clusters, while decreasing its probability of visiting pages belonging to other clusters.

Figure 4 shows the value of the cost function $E^{\mathcal{P}}$ at each iteration. Each plot reports the cost function when a specific set of parameters is optimized during training. The behavior parameters are less effective than the jump parameters in optimizing the surfer model. However, neither the behavior or the jump parameters are as effective as the transition parameters. As expected, when all parameters considered during the



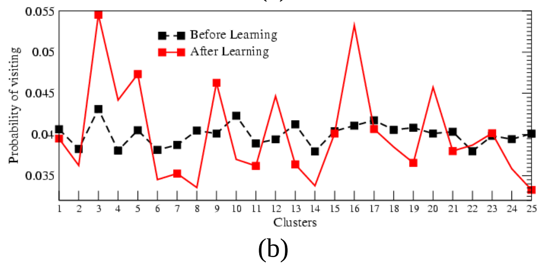


Figure 3: Plots of the normalized probability of the surfer of being located in a page of a given cluster before and after learning. (a) results on the crawl for the topic "wine" with 100 clusters. (b) results on a random graph with 200.000 nodes and 25 clusters.

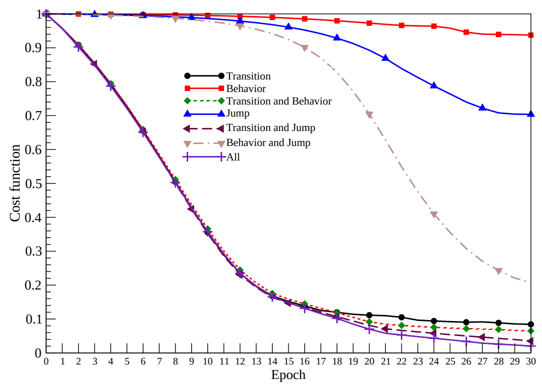


Figure 4: Values of cost function as a function of the iteration number. Each plot reports the cost function when a specific set of parameters is considered during training.

learning process, the cost function reaches its minimum.

In figure 5, there are shown the scores of the pages of cluster 8 of the dataset "Playstation" before and after learning. This cluster was manually detected as strongly on the topic "Playstation" and 3 pages belonging this cluster were explicitly inserted in learning set as pages that should get an higher score. As expected the rank of the other pages in this cluster was boosted up thanks to the capability of the algorithm to generalize from a small number of training examples.

4.2 Qualitative results

This experiment compares the quality of the page ranks before and after training. In particular, some authoritative pages on the topic "wine" were provided as positive examples, while some authoritative pages on other topics were provided as negative examples. Figure 6 reports the pages which obtained the largest variation in their rank. The pages that ob-

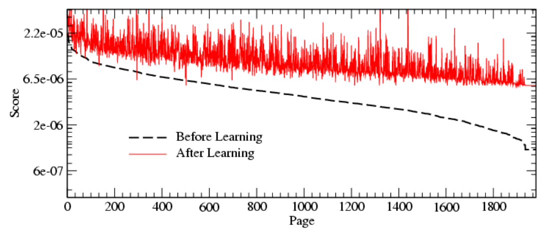


Figure 5: The scores of the pages of cluster 8 of the dataset "Playstation" before and after learning. In the learning process only 3 pages belonging to the cluster 8 were explicitly inserted in learning set as pages that should get an higher score.

Most descending pages on topic "wine"
http://www.macromedia.com/go/getflashplayer
http://www.netscape.com
http://www.macromedia.com/downloads
http://www.yahoo.com
http://www.adobe.com/products/acrobat/readstep.html
http://help.yahoo.com
http://www.nl.placestostay.com/index.html
http://home.netscape.com
http://www.forgottensoldier.com
http://www.inntravels.com

Most ascending pages on topic "wine"
http://www.winecountry.com
http://webwinery.com
http://www.ny-wine.com/wineinfo.asp
http://www.wine-searcher.com
http://www.insidewine.com
http://webwinery.com
http://www.viniesaporidipuglia.com/en/index.html
http://www.vinitaly.com/home_en.asp
http://www.wineinstitute.org
http://www.winespectator.com

Figure 6: URLs of the pages that yielded either the largest negative or positive rank changes among the pages that were initially in the top 1000 scoring documents.

tained a negative rank variation were either topic-generic authoritative pages (e.g. www.netscape.com) or pages relevant for different topics than "wine" (e.g. "www.forgottensoldier-.org"). Among the pages that got a negative rank variation, only the page "www.yahoo.com" was specified as a negative example. On the other hand, the pages that obtained a positive rank variation were effectively authoritative pages on the topic "wine" (e.g. "www.wine-searcher.com" or "webwinery.com"). Among the most ascending documents, only the page "www.winespectator.com" was explicitly provided to the system as a positive example, whereas other pages were pushed up thanks to the capability of the system to generalize from a small set of training examples.

5 Conclusions

In this paper we introduced a novel algorithm to learn a score distribution over the nodes of a labeled graph. The learning algorithm adapts the parameters of a random surfer in order to match the target scores assigned on a small subset of the nodes. Since the parameters are estimated from the unfoldings of the graph centered on the example nodes, the learning algorithm is efficient and can be applied to graphs with many nodes. The learning algorithm was applied to the problem of learning a generic ranking function on the portion of the Web graph.

References

[Chang *et al.*, 2000] H. Chang, D. Cohn, and A. K. McCallum. Learning to create customized authority lists. In *Proceedings of the 17th International Conference on Machine Learning*, pages 127–134. Morgan Kaufmann, 2000.

[Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:185–197, 1977.

[Diligenti *et al.*, 2000] M. Diligenti, F. Coetzee, S. Lawrence, L. Giles, and M. Gori. Focus crawling by context graphs. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000)*, pages 527–534, 2000.

[Diligenti *et al.*, 2002] M. Diligenti, M. Gori, and M. Maggini. Web page scoring systems for horizontal and vertical search. In *Proceedings of the World Wide Web Conference (WWW2002)*, pages 508–516, 2002.

[Diligenti *et al.*, 2003] M. Diligenti, M. Gori, and M. Maggini. A learning algorithm for web page scoring systems. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI2003)*, pages 575–580, 2003.

[Duda and Hart, 1973] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[Haykin, 1994] S. Haykin. *Neural networks*. Macmillan College Publishing Company, 1994.

[Kondor and Lafferty, 2002] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the International Conference on Machine Learning (ICML2002)*, pages 315–322, 2002.

[Page *et al.*, 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Computer Science Department, Stanford University, 1998.

[Richardson and Domingos, 2002] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In *Advances in Neural Information Processing Systems 14*, pages 1441–1448, Cambridge, MA, 2002.

[Tsoi *et al.*, 2003] A. C. Tsoi, G. Morini, F. Scarselli, M. Hagenbuchner, and M. Maggini. Adaptive ranking of web pages. In *Proceedings of the 12th International World Wide Web Conference (WWW12)*, pages 356–365, 2003.