Random walk based web page ranking functions learning with gradient-free optimization methods

Pavel Dvurechensky* Alexander Gasnikov† Maxim Zhukovskii‡

December 2, 2014

Abstract

In this paper we consider a problem of web page relevance to a search query. We are working in the framework called Semi-Supervised PageRank which can account for some properties which are not considered by classical approaches such as PageRank and BrowseRank algorithms. We introduce a graphical parametric model for web pages ranking. The goal is to identify the unknown parameters using the information about page relevance to a number of queries given by some experts (assessors). The resulting problem is formulated as an optimization one. Due to hidden huge dimension of the last problem we use random gradient-free methods to solve it. We prove the convergence theorem and give the number of arithmetic operations which is needed to solve it with a given accuracy.

1 INTRODUCTION

Web page relevance to a search query scores are exploited for web ranking algorithms. There are many approaches to evaluation of this page importance. Among them there are those which are based on a structure of a graph which represents Web. Nowadays, various graphs such as link graph, user browsing graphs, query–flow graphs, and other graphs with a large amount of information on nodes are exploited for such algorithms.

The most acknowledged of these methods are based on random walk models. Particularly, PageRank (Page, Brin, Motwani, Winograd, 1999), HITS (Kleinberg, 1998), and their variants (see, e.g., Haveliwala, 2002, Richardson, Domingos, 2002), are classical link based algorithm and BrowseRank algorithm (Liu, Gao, Liu, Zhang, Ma, He, Li, 2008) is based on a user browsing history.

According to PageRank algorithm, the score of node \(p\) equals to its probability in the stationary distribution of a Markov process, which models a random walk on the graph. A certain random walk can be defined by its transition and restart probabilities. For instance, in the original PageRank algorithm these probabilities are equal for all nodes in the case of a restart and are equal for all destination nodes in the case of a transition.

Despite undeniable advantages of PageRank and BrowseRank, these algorithms miss the different aspects of the Web that are not described only by link structure or users’ behavior.

In contrast, in more recent approaches the restart and transition probabilities may essentially depend on the properties of nodes and edges between them. Nodes and edges may be supplied with some metadata including, e.g., the statistics about users interactions with the nodes (e.g., the

*Institute for Information Transmission Problems RAS, Moscow Institute of Physics and Technology
†Moscow Institute of Physics and Technology
‡Moscow Institute of Physics and Technology, Yandex
number or average duration of visits of documents by users or the number of transitions between them), document language models or histories of document changes. Such information defines certain properties of nodes and edges and can be used for a more accurate evaluation of nodes’ authorities as compared to the methods solely based on the geometric graph structure.

There is a number of studies addressing the problem of appropriate accounting for the above mentioned important properties of nodes and edges for graph ranking (see Dai, Davison, 2010, Eiron, McCurley, Tomlin, 2004, Haveliwala, 1999).

A ranking framework called Semi-Supervised PageRank (Gao, Liu, Huazhong, Wang, Li, 2011) account for such properties. Weights of nodes and edges in link graph in this framework are linear combinations of their features with coefficients as the model parameters. The authors learn the parameters by solving a particular optimization problem that cannot be extended to other quality measures of stationary distributions. Moreover, the objective function in SSP depends on a high-dimensional vector. Therefore, the optimization is extremely computationally expensive.

We introduce the general model of web page ranking which is based on the random walk on the user browsing graph and generalizes previous models. Our optimization algorithm can be applied to a wider class of optimization tasks and the objective function depends on a small number of parameters.

2 MODEL DESCRIPTION

Let us consider the graphical model for web page ranking score algorithm.

We define the sessions and the user’s browsing graph in the similar way to Liu et al (Liu, Gao, Liu, Zhang, Ma, He, Li, 2008). Let a query \( q \) be given. Let \( S_q = (i_1, i_2, ..., i_k) \) be a user session which is started from \( q \). For each \( j \in \{1, 2, ..., k - 1\} \) the element \( i_j \) is either a page or a query and there is a record \( i_j \rightarrow i_{j+1} \) which is made by toolbar. We call pages \( i_j, i_{j+1} \) the neighboring elements of the session \( S_q \). For given query \( q \) we define the user browsing graph \( G_q = (V_q, E_q) \) as follows. The set of vertices \( V_q = V_q^1 \sqcup V_q^2 \) consists of all the elements from all the \( S_q \), \( V_q^1 \) is the set of queries, \( V_q^2 \) is the set of pages. Denote by \( p_q \) the number of vertices in \( V_q^1 \) and by \( n_q \) the number of vertices in \( V_q^2 \). The set of directed edges \( E_q \) represents all the ordered pairs of neighboring elements \( i, \tilde{i} \) from the sessions.

Let \( \mathcal{F}_1 = \{f_q(\cdot, \cdot) : \mathbb{R}^{m_1} \times V_q^1 \rightarrow \mathbb{R}\} \), \( \mathcal{F}_2 = \{g_q(\cdot, \cdot) : \mathbb{R}^{m_2} \times E_q \rightarrow \mathbb{R}\} \) be two classes of functions parametrized by \( \varphi = (\varphi_1, \varphi_2)^T \in \mathbb{R}^{m_1 + m_2} \), where \( m_1 \) is the number of query features, \( m_2 \) is the number of link features or query–document features. We denote \( m = m_1 + m_2 \) and consider the case when \( m \) is of the order of \( 10^3 \). Let us describe the random walk on the graph \( G_q \). It starts from any query from \( V_q^1 \). Reset probability

\[
[p_q^0]_i = \frac{f_q(\varphi_1, i)}{\sum_{j \in V_q^1} f_q(\varphi_1, j)}
\]

is a probability of choosing a query with number \( i \) while a user types a new query (the probability equals 0 for all the pages). Transition probability

\[
g_q(\varphi_2, \tilde{i} \rightarrow i) \frac{1}{\sum_{j: \tilde{i} \rightarrow j} g_q(\varphi_2, \tilde{i} \rightarrow j)}
\]

is a probability of clicking a link \( \tilde{i} \rightarrow i \) or clicking the page \( i \) shown to the query \( \tilde{i} \). Finally, probability of moving to \( i \) from \( \tilde{i} \) equals

\[
\alpha \frac{f_q(\varphi_1, i)}{\sum_{\tilde{i} \in V_q^1} f_q(\varphi_1, \tilde{i})} + (1 - \alpha) \frac{g_q(\varphi_2, \tilde{i} \rightarrow i)}{\sum_{j: \tilde{i} \rightarrow j} g_q(\varphi_2, \tilde{i} \rightarrow j)}
\]
where the parameter $\alpha \in (0, 1)$ is called damping factor (it equals the probability of typing a query by a user and not clicking a linked page). Originally, $\alpha = 0.15$. Denote by $\pi_q \in \mathbb{R}^{n_q}$ the stationary distribution of the described Markov chain. The vector $\pi_q = (\pi^1_q, \pi^2_q)^T$ has two large components: first $\pi^1_q \in \mathbb{R}^{n_q}$ for query vertices from $V_q^1$ and second $\pi^2_q \in \mathbb{R}^{n_q}$ for pages vertices from $V_q^2$. The web page ranking score $s_q(p)$ is defined to be equal to the $p$-th component $[\pi_q]_p$ of this stationary distribution.

Our goal is to find the parameters vector $\varphi$ which minimizes the discrepancy of the web page ranking scores $\pi_q[p]$, $p \in V_q^2$ calculated as the stationary distribution in the above Markov chain from the web page ranking scores defined by assessors. The training data which is given by assessors $\pi_q$ is the collection of queries and sets of pages $P_q^1, P_q^2, ..., P_q^k$ for each query $q$ which are ordered from the most relevant to irrelevant pages. In other words, $P_q^+$ is the set of all pages with the highest score selected from among $k$ labels, pages from the set $P_q^k$ have the lowest score. For any two pages $p_1 \in P_q^i, p_2 \in P_q^j$ let $h(i, j, [\pi_q]_{p_2} - [\pi_q]_{p_1})$ be a value of a penalty we get if the position of the page $p_1$ according to our ranking algorithm is higher than the position of the page $p_2$ but $i > j$. The function $h$ is a loss function. We consider square loss with margins $b_{ij} > 0$, where $1 \leq i < j \leq k$: $h(i, j, x) = \max\{x + b_{ij}, 0\}^2$ as it was done in previous studies (see Backstrom, Leskovec, 2011, Zhukovskii, Khropov, Gusev, Serdyukov, 2013). We minimize

$$f(\varphi) = \frac{1}{Q} \sum_q \sum_{1 \leq i < j \leq k} \sum_{p_1 \in P_q^i, p_2 \in P_q^j} h(i, j, [\pi_q]_{p_2} - [\pi_q]_{p_1})$$

(2.1)

in order to learn our model using the data given by assessors. Here $Q$ is the total number of queries used for learning. In the next section we discuss this optimization problem.

### 3 OPTIMIZATION PROBLEM

As it was said above, finding web pages ranking score for the fixed query $q$ leads to the problem of finding the stationary distribution $\pi_q$ of the Markov chain as a solution of the equation.

$$\pi_q = \alpha \pi^0_q(\varphi) + (1 - \alpha) \frac{g_q(\varphi_2, \tilde{i} \rightarrow i)}{\sum_{j: i \rightarrow j} g_q(\varphi_2, \tilde{i} \rightarrow j)} \pi_q,$$

or equivalently

$$\pi_q = \alpha \pi^0_q(\varphi) + (1 - \alpha) P_q^T(\varphi) \pi_q.$$  

(3.1)

Here by $P_q(\varphi)$ we denote transition probabilities matrix. Note that $P_q(\varphi)e = e$, where $e$ is the vector of all ones. It is obvious that the solution $\pi^*_q(\varphi)$ of (3.1) can be found as

$$\pi^*_q(\varphi) = \alpha \left[ I - (1 - \alpha) P_q^T(\varphi) \right]^{-1} \pi^0_q(\varphi),$$

where $I$ is the identity matrix.

It is shown by Nemirovski, Nesterov (2012) that the vector

$$\tilde{\pi}_q^N(\varphi) = \alpha \frac{1}{1 - (1 - \alpha)^{N+1}} \sum_{i=0}^{N} (1 - \alpha)^i [P_q^T(\varphi)]^i \pi^0_q(\varphi)$$

(3.2)

satisfies

$$\|\tilde{\pi}_q^N(\varphi) - \pi^*_q(\varphi)\|_1 \leq 2(1 - \alpha)^{N+1}.$$
Hence to obtain vector $\tilde{\pi}_q^N(\varphi)$ satisfying $\|\tilde{\pi}_q^N(\varphi) - \pi_q^*(\varphi)\|_1 \leq \Delta$ we need $\frac{n_q(p_q+n_q)}{\alpha} \ln \frac{2}{\Delta}$ arithmetic operations. Here $s_q$ is the sparsity parameter denoting maximum over columns of the matrix $P_q(\varphi)$ number of non-zero elements in each column. Note that $s_q \ll n_q$.

Let us now consider the minimization of the function $f(\varphi)$ (2.1). We can rewrite this function as

$$f(\varphi) = \frac{1}{Q} \sum_q \| (A_q \pi_q^*(\varphi) + b_q)_+ \|_2^2,$$

where vector $x_+$ has components $[x_+]_i = \max\{x_i, 0\}$, the matrix $A_q \in \mathbb{R}^{q_r \times (p_q+n_q)}$ represents assessor’s view of the relevance of pages to the query $q$. Note that each row of the matrix $A_q$ contains one ”1” and one ”-1”, and all other elements of the row are equal to zero and hence $\|A_q\|_2 \leq \sqrt{2r_q}$.

Since $\varphi \in \mathbb{R}^m$ and $m$ is of the order of $10^3$ the first idea for minimization of the function $f(\varphi)$ is to use methods based on the gradient calculation. But in this case we have to calculate Jacobi matrix for the mapping $\pi_q^*(\varphi)$ which is a very large matrix with one dimension $p_q+n_q$ of the order $10^7$. This leads to calculation of the inverse matrix of dimension $(p_q+n_q) \times (p_q+n_q)$ on each iteration of the gradient method, which is too costly.

Another idea is to use Fast Automatic Differentiation which allows to calculate gradient of the function with not more than four times larger number of arithmetic operations than the number of arithmetic operations for calculating the value of the function. But it is also well known that the amount of memory to do that is very large. Another obstacle for using FAD is that we can’t calculate the exact value of the function $f(\varphi)$ since we can’t calculate the precise value of $\pi_q^*(\varphi)$ and only can calculate its approximation $\tilde{\pi}_q^N(\varphi)$ (3.2).

So we are going to use gradient-free methods for minimization of the function $f(\varphi)$. Such methods were introduced rather long ago, see e.g. (Matyas, 1965). Note that we have to work in the framework of non-exact zero-order oracle. The following result says how the error of the approximation of $\pi_q^*(\varphi)$ affects the error in the value of the function $f(\varphi)$.

**Lemma 1.** Assume that the vector $\tilde{\pi}_q^N(\varphi)$ satisfies $\|\tilde{\pi}_q^N(\varphi) - \pi_q^*(\varphi)\|_1 \leq \Delta$. Denote $r = \max_q r_q$, $b = \max_q \|b_q\|_2$. Then $f_\delta(\varphi) = \frac{1}{Q} \sum_q \| (A_q \tilde{\pi}_q^N(\varphi) + b_q)_+ \|_2^2$ satisfies $|f_\delta(\varphi) - f(\varphi)| \leq \Delta \sqrt{2r} (2\sqrt{2r} + 2b)$.

### 4 RANDOM GRADIENT-FREE METHODS

Let us describe the framework for random gradient-free methods (Nesterov, 2011) but use different randomization and extend it to the case of oracle error presence. Assume that the function $f(\cdot): \mathbb{R}^m \rightarrow \mathbb{R}$ is convex and has Lipschitz continuous gradient with constant $L$ (we write $f \in C_L^{1,1}$):

$$|f(x) - f(y) - \langle \nabla f(y), x-y \rangle| \leq \frac{L}{2} \|x-y\|_2^2, \quad x, y \in \mathbb{R}^m.$$  

Also we assume that the oracle returns the value $f_\delta(x) = f(x) + \tilde{\delta}(x)$, where $\tilde{\delta}(x)$ is the oracle error satisfying $|\tilde{\delta}(x)| \leq \delta$. Consider smoothed counterpart of the function $f(x)$:

$$f_\mu(x) = \mathbb{E}_b f(x + \mu b) = \frac{1}{V_B} \int_B f(x + \mu b) db,$$

where $b$ is a uniformly distributed over unit ball $B = \{x \in \mathbb{R}^m : \|x\|_2 \leq 1\}$ random vector, $V_B$ is the volume of the unit ball $B$, $\mu \geq 0$ is the smoothing parameter. It is easy to show that

- If $f$ is convex, then $f_\mu(x)$ is also convex
• If \( f \in C^{1,1}_L \) then \( f_\mu \in C^{1,1}_L \).
• If \( f \in C^{1,1}_L \) then \( f_\mu(x) \leq f(x) \leq f_\mu(x) + \frac{L\mu^2}{2} \), \( \forall x \in \mathbb{R}^m \).

We define random gradient-free oracle

\[
g_\mu(x) = \frac{m}{\mu} (f(x + \mu s) - f(x)) s,
\]

where \( s \) is uniformly distributed vector over the unit sphere \( S = \{ x \in \mathbb{R}^m : \|x\|_2 = 1 \} \). Since we can use only zeroth-order oracle with error we also define the counterpart of the above random gradient-free oracle which can be really computed. We will call it biased gradient-free oracle.

\[
g_{\mu, \delta}(x) = \frac{m}{\mu} (f_\mu(x + \mu s) - f_\mu(x)) s.
\]

The following estimates can be proved for the introduced inexact oracle.

**Lemma 1.** Let \( f \in C^{1,1}_L \). Then for any \( x, y \in \mathbb{R}^m \)

\[
\mathbb{E}_s \|g_{\mu, \delta}(x)\|_2^2 \leq m^2 \mu^2 L^2 + 8m \|\nabla f(x)\|_2^2 + \frac{8\delta^2 m^2}{\mu^2} + \frac{8\delta m}{\mu} \|x - y\|_2.
\]

(4.1)

Then the idea is to use gradient and fast gradient methods with oracle \( g_{\mu, \delta}(x) \) instead of real gradient in order to minimize \( f_\mu(x) \). Since it is uniformly close to \( f(x) \) we can obtain a good approximation to minimal value of \( f(x) \).

The first method (Algorithm 1) is the variation of gradient method. Here \( \Pi_G(x) \) denotes the Euclidean projection of the point \( x \) onto the set \( G \).

**ALGORITHM 1:** Gradient-type method

**Input:** The point \( x_0 \), number \( R \) such that \( \|x_0 - x^*\|_2 \leq R \), stepsize \( h > 0 \), number of steps \( N \).

**Output:** The point \( x_k \).

**Define** \( G = \{ x \in \mathbb{R}^m : \|x - x_0\|_2 \leq 2R \} \).

**repeat**

- Generate \( s_k \) and corresponding \( g_{\mu, \delta}(x_k) \).
- Calculate \( x_{k+1} = \Pi_G(x_k - h g_{\mu, \delta}(x_k)) \).

**until** \( k > N \);

Next theorem gives the convergence rate of the Algorithm 1. Denote \( U_k = (s_0, \ldots, s_k) \) the history of realizations of the vectors \( s_k \), generated on each iteration of the method, \( \psi_0 = f(x_0) \), and \( \psi_k = \mathbb{E}_{\psi_{k-1}}(f(x_{k-1})) \), \( k \geq 1 \).

We say that the smooth function is strongly convex with parameter \( \tau \geq 0 \) iff for any \( x, y \in \mathbb{R}^m \) we have

\[
f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\tau}{2} \|x - y\|^2.
\]

(4.3)

**Theorem 1.** Let \( f \in C^{1,1}_L \) and the sequence \( x_k \) be generated by the Algorithm 1 with \( h = \frac{1}{8mL} \). Then for any \( N \geq 0 \), we have

\[
\frac{1}{N+1} \sum_{i=0}^{N} (\psi_i - f^*) \leq \\
\frac{8mLR^2}{N+1} + \frac{\mu^2 L(m+8)}{8} + \frac{8\delta m R}{\mu} + \frac{\delta^2 m}{L \mu^2}.
\]

(4.4)
If additionally $f$ is strongly convex with constant $\tau$, then

$$\psi_N - f^* \leq \frac{1}{2} L \left( \delta_\mu + \left( 1 - \frac{\tau}{16mL} \right)^N \left( R^2 - \delta_\mu \right) \right), \tag{4.5}$$

where $\delta_\mu = \frac{\mu^2 L (m + 8)}{4\tau} + \frac{16m\delta R}{\tau \mu} + \frac{2m\delta^2}{\tau^2 L^2}$.

**Proof.**

Consider the point $x_k$, $k \geq 0$ generated by the method on the $k$-th iteration. Denote $r_k = \|x_k - x^*\|_2$. Note that $r_k \leq 4R$. We have:

$$r_{k+1}^2 = \|x_{k+1} - x^*\|^2_2 \leq \|x_k - x^* - h g_{\mu, \delta}(x_k)\|^2_2 = \|x_k - x^*\|^2_2 - 2h(g_{\mu, \delta}(x_k), x_k - x^*) + h^2\|g_{\mu, \delta}(x_k)\|^2_2.$$

Taking the expectation with respect to $s_k$ we get

$$\mathbb{E}_{s_k} r_{k+1}^2 \leq r_k^2 - 2h\langle \nabla f(\mu)(x_k), x_k - x^* \rangle + \frac{2\delta m h \mu}{\mu} r_k +$$

$$+ h^2 \left( m^2 \mu^2 L^2 + 4m\|\nabla f(x_k)\|^2_2 + \frac{8 \delta^2 m^2}{\mu^2} \right) \leq$$

$$\leq r_k^2 - 2h(f(x_k) - f(\mu)) + \frac{8 \delta m h R}{\mu} +$$

$$+ h^2 \left( m^2 \mu^2 L^2 + 8mL(f(x_k) - f^*) + \frac{8 \delta^2 m^2}{\mu^2} \right) \leq$$

$$\leq r_k^2 - 2h(1 - 4hmL)(f(x_k) - f^*) + \frac{8 \delta m h R}{\mu} +$$

$$+ m^2 h^2 \mu^2 L^2 + hL\mu^2 + \frac{8 \delta^2 m^2 h^2}{\mu^2} \leq$$

$$\leq r_k^2 + \frac{R\delta}{\mu L} - \frac{f(x_k) - f^*}{8mL} + \frac{\mu^2 (m + 8)}{64m} + \frac{\delta^2}{8\mu^2 L^2}. \tag{4.6}$$

Taking expectation with respect to $U_{k-1}$ and defining $\rho_{k+1} \overset{\text{def}}{=} \mathbb{E}_{s_k} r_{k+1}^2$ we obtain

$$\rho_{k+1} \leq \rho_k - \frac{\psi_k - f^*}{8mL} + \frac{\mu^2 (m + 8)}{64m} + \frac{R\delta}{\mu L} + \frac{\delta^2}{8\mu^2 L^2}.$$

Summing up these inequalities and dividing by $N + 1$ we obtain (4.4).

Now assume that the function $f(x)$ is strongly convex. From (4.6) we get

$$\mathbb{E}_{s_k} r_{k+1}^2 \leq \left( 1 - \frac{\tau}{16mL} \right) r_k^2 + \frac{R\delta}{\mu L} + \frac{\mu^2 (m + 8)}{64m} + \frac{\delta^2}{8\mu^2 L^2}.$$

Taking expectation with respect to $U_{k-1}$ we obtain

$$\rho_{k+1} \leq \left( 1 - \frac{\tau}{16mL} \right) \rho_k + \frac{R\delta}{\mu L} + \frac{\mu^2 (m + 8)}{64m} + \frac{\delta^2}{8\mu^2 L^2}$$

and

$$\rho_{k+1} - \rho_k \leq \left( 1 - \frac{\tau}{16mL} \right) (\rho_0 - \rho_k) \leq \left( 1 - \frac{\tau}{16mL} \right)^{k+1} (\rho_0 - \delta_\mu).$$
Using the fact that $\rho_0 = R^2$ and $\psi_k - f^* \leq \frac{1}{2}L\rho_k$ we obtain (4.5).

The estimate (4.4) also holds for $\hat{\psi}_N \overset{\text{def}}{=} \mathbb{E}_{\delta_k^{-1}} f(\hat{x}_N)$, where $\hat{x}_N = \arg\min_x \{ f(x) : x \in \{x_0, \ldots, x_N\} \}$. To make the right hand side of the inequality (4.4) less than desired accuracy $\varepsilon$ we need to choose

$$N = \left\lceil \frac{32mLR^2}{\varepsilon} \right\rceil, \quad \mu = \sqrt{\frac{2\varepsilon}{L(m+8)}},$$

$$\delta = \min \left\{ \frac{\varepsilon^{\frac{3}{2}} \sqrt{2}}{32mR\sqrt{L(m+8)}}, \frac{\varepsilon}{\sqrt{2m(m+8)}} \right\} = \frac{\varepsilon^{\frac{3}{2}} \sqrt{2}}{32mR\sqrt{L(m+8)}}.$$

Note that we choose the dependencies $\mu(\varepsilon)$ and $\delta(\varepsilon)$ in an optimal way. This means that $\delta(\varepsilon)$ is as much as it possible for the algorithm to converge at the same rate as in the case when $\delta = 0$.

Let’s note that from this theorem we can also estimate the probability of large deviations from the obtained rate of convergence. If $f(x)$ is strongly convex then we have a geometric rate of convergence (4.5). Consequently, from the Markov’s inequality we have (e.g. Nesterov, 2010) that after $O\left( m L \ln \left( \frac{LR^2}{\varepsilon \sigma} \right) \right)$ iterations we obtain $\psi_N - f^* \leq \varepsilon$ with probability greater than $1 - \sigma$. If the function $f(x)$ is not strongly convex then we can introduce regularization with parameter $\tau = \varepsilon/(2R^2)$ minimizing the function $f(x) + \frac{\tau}{2}\|x\|^2$ which is strongly convex. This will give us that after $O\left( m \frac{LR^2}{\varepsilon} \ln \left( \frac{LR^2}{\varepsilon \sigma} \right) \right)$ iterations we obtain $\psi_N - f^* \leq \varepsilon$ with probability greater than $1 - \sigma$.

5 SOLVING THE LEARNING PROBLEM

Now the idea for minimizing the function $f(\varphi)$ (3.3) is the following. We assume that we start from the small vicinity of the optimal value and hence the function $f(\varphi)$ is convex in this vicinity. Later we can consider some globalization techniques, e.g. multi-start. We choose the desired accuracy $\varepsilon$ for approximation of the optimal value of the function $f(\varphi)$. This value gives us the number of steps of the Algorithm 1, the value of the parameter $\mu$, the maximum value of the allowed error of the oracle $\delta$. Knowing the value $\delta$, using the Lemma 1 we choose the number of steps of the algorithm for approximate solution of the equation (3.1), i.e. the number $N$ in (3.2). This idea leads us to the Algorithm 2.

The most computationally consuming operation on each iteration of the main cycle of this method is the calculation of $2Q$ approximate solutions of the equation (3.1). Hence each iteration of the Algorithm 2 needs approximately $2Qs(p+q) \ln \frac{LR^2}{\alpha \varepsilon}$ arithmetic operations, where $s = \max_q s_q$, $p = \max_q p_q$, $n = \max_q n_q$. The total number of arithmetic operations for the accuracy $\varepsilon$ is given by

$$64m(n+p)sQ \frac{LR^2}{\alpha \varepsilon} \cdot \ln \left( 4(2r + b\sqrt{2r}) \frac{32mR\sqrt{L(m+8)}}{\varepsilon^{\frac{3}{2}} \sqrt{2}} \right).$$

Let us make some remarks. Note that each iteration of the main cycle of the algorithm above can be fully paralleled using $Q$ processors.

Without loss of generality we can assume that $R = 1$. This is because we can always make a proper rescaling. So instead of consideration of two unknown parameters $R$ and $L$ we can restrict
Algorithm 2: Method for model learning

**Input:** The point $\varphi_0$, $L$ – Lipschitz constant for the function $f(\varphi)$, number $R$ such that $\|\varphi_0 - \varphi^*\|_2 \leq R$, accuracy $\varepsilon > 0$, numbers $r, b$ defined in Lemma 1.

**Output:** The point $\hat{\varphi}_N = \arg \min_{\varphi} \{f(\varphi) : \varphi \in \{\varphi_0, \ldots, \varphi_N\}\}$.

Define $G = \{\varphi \in \mathbb{R}^m : \|\varphi - \varphi_0\|_2 \leq 2R\}$, $N = 32mL^2/\varepsilon^2$, $\delta = \varepsilon^2/2\sqrt{2}$, $\mu = \sqrt{2}\varepsilon L(m+8)$;

Set $k = 0$;

repeat

- Generate random vector $s_k$ uniformly distributed over a unit Euclidean sphere $S$ in $\mathbb{R}^m$;
- Set $\tilde{N} = 1/\alpha \ln \left(\frac{2\sqrt{2}(\sqrt{2} + b\sqrt{2})}{\delta}\right)$;
- For every $q$ calculate $\tilde{\pi}^N_q(\varphi_k)$, $\tilde{\pi}^N_q(\varphi_k + \mu s_k)$ defined in (3.2);
- Calculate $g_{\mu,\delta}(x_k) = \frac{1}{\mu} (f_\delta(\varphi_k + \mu s_k) - f_\delta(\varphi_k))s_k$;
- Calculate $\varphi_{k+1} = \Pi_G (\varphi_k - \frac{1}{8mL}g_{\mu,\delta}(\varphi_k))$;
- Set $k = k + 1$;

until $k > N$

ourselves only by one parameter $L$. The direct calculation of this parameter has many obstacles, which lead us to the overestimation. Another way is to use restart method. Since we know the exact number of required iteration for the fixed accuracy, confidence level and $L$, we can fulfill the following procedure. We start with $L = 1$. Calculate the optimal function value given by the algorithm. Then $L := 2L$ and repeat, i.e. calculate optimal function value given by the algorithm, working with new $L$, etc. The stopping rule here is stabilization (with the same accuracy as before) of this sequence of function values. The total number of such restarts will be of order $\log_2(2L)$.

Here we have omitted the full description of the application of fast-gradient-type scheme for the minimization of the function $f(\varphi)$ (Nesterov, 2004, 2011). The fast-gradient-type scheme is faster but requires the oracle to be more precise. The resulting value of the number of arithmetic operations to achieve the accuracy $\varepsilon$ for this method is

$$O\left(\frac{mnsQ}{\alpha^2 \varepsilon} \sqrt{\frac{LR^2}{\alpha^2 \varepsilon}}\ln\left(\frac{(r + b\sqrt{r})mRL}{\varepsilon}\right)\right).$$

Note that when $P_q(\varphi)$ does not depend on $\varphi$ (only $\pi^0_q(\varphi)$ depends on $\varphi$) or $P_q(\varphi)$ has elements which allow Fast Automatic Differentiation then we can eliminate the factor $m$ in the estimates above by using (fast) gradient methods with inexact oracle (Devolder, Glineur, Nesterov, 2011).

6 **CONCLUSION**

We have introduced new parametric graphical model for web page ranking problem. Ranking scores in this model is calculated using the stationary distribution of special Markov chain consisting of query and pages vertices. The probabilities of transition in this Markov chain depend on parameter vector $\varphi$. To learn the model and find parameters $\varphi$ we use ranking scores given by experts and adjust $\varphi$ to make scores given by random walks to be close to the expert’s scores. This leads to the problem of minimization of the function with expensive gradient and hence we use random gradient-free methods combined with simple-iteration-type method for finding stationary distribution. We provide an estimate for the total number of arithmetic operations to obtain the given accuracy.

In our opinion next steps are the following.

1. Try to use universal gradient method by Nesterov to make the algorithm flexibility of not knowing the constant $L$. 


2. Study which globalization techniques are the best for this problem.

3. Study more thoroughly the applicability of FAD-like techniques to this problem.

Acknowledgements

We are very thankful to prof. Yu. Nesterov, prof. A. Nemirovski, prof. B.T. Polyak, and prof. A.M. Raigorodskii for fruitful discussions. The research presented in this paper was partially supported by the Laboratory of Structural Methods of Data Analysis in Predictive Modeling, MIPT, through the RF government grant, ag.11.G34.31.0073, by RFBR, research project No. 14-01-00722 A and by Russian President grant No. MK-5285.2013.9.

References

L. Page, S. Brin, R. Motwani, and T. Winograd, The PageRank citation ranking: Bringing order to the web. http://dbpubs.stanford.edu/pub/1999-66, 1999.

J. M. Kleinberg, Authoritative sources in a hyperlinked environment, SODA’98.

Taher H. Haveliwala, Topic-Sensitive PageRank, WWW’2002.

M. Richardson, P. Domingos, The intelligent surfer: Probabilistic combination of link and content information in PageRank, Advances in Neural Information Processing Systems, 14, pp. 1441–1448, 2002.

Na Dai and Brian D. Davison, Freshness Matters: In Flowers, Food, and Web Authority, SIGIR’10.

Nadav Eiron, Kevin S. McCurley, John A. Tomlin, Ranking the web frontier, WWW’04.

Taher H. Haveliwala, Efficient computation of PageRank, Stanford University Technical Report, 1999.

Y. Liu, B. Gao, T.-Y. Liu, Y. Zhang, Z. Ma, S. He, H. Li, BrowseRank: Letting Web Users Vote for Page Importance. Proc. SIGIR’08, pp. 451–458, 2008.

B. Gao, Tie-Yan Liu, Wei Wei Huazhong, T. Wang, H. Li, Semi-supervised ranking on very large graphs with rich metadata, KDD’11.

L. Backstrom, J. Leskovec, Supervised random walks: predicting and recommending links in social networks, WSDM’11.

M. Zhukovskii, A. Khropov, G. Gusev, P. Serdyukov, Fresh BrowseRank, SIGIR’13.

J. Matyas (1965). Random optimization. Automation and Remote Control, 26, 246-253.

Yu. Nesterov, A. Nemirovski (2012). Finding the stationary states of Markov chains by iterative methods CORE Discussion Paper 2012/58, http://dial.academielouvain.be/handle/boreal:122163.

Yu. Nesterov (2011). Random gradient-free minimization of convex functions, CORE Discussion Paper 2011/16.

Yu. Nesterov (2010) Efficiency of coordinate descent methods on huge-scale optimization problems CORE Discussion paper. 2010/2.

Yu. Nesterov (2004) Introductory Lectures on Convex Optimization. Kluwer, Boston.

O. Devolder, F. Glineur and Yu. Nesterov (2011) First-order Methods of Smooth Convex Optimization with Inexact Oracle CORE Discussion Paper 2011/2, http://www.optimization-online.org/DB_FILE/2010/12/2865.pdf