Доклади на Българската академия на науките
Comptes rendus de l'Académie bulgare des Sciences

*Tome 75, No 10, 2022*

*MATHEMATICS*
*Computer science*

# LARGE SCALE RANKING USING STOCHASTIC GRADIENT DESCENT

**Engin Tas**

*Received on May 16, 2022*
*Presented by S. Margenov, Corresponding Member of BAS, on July 19, 2022*

**Abstract**

A system of linear equations can represent any ranking problem that minimizes a pairwise ranking loss. We utilize a fast version of gradient descent algorithm with a near-optimal learning rate and momentum factor to solve this linear equations system iteratively. Tikhonov regularization is also integrated into this framework to avoid overfitting problems where we have very large and high dimensional but sparse data.

**Key words:** ranking, stochastic gradient descent, least-squares, gradient methods, stochastic optimization

**2020 Mathematics Subject Classification:** 68T05, 68Q32, 90C05

**1. Introduction.** Search engines on the Internet and systems that offer specific suggestions in various fields have become very popular in recent years, placing the ranking procedures in these fields in a critical position. The availability of unprecedented size in these areas has become an indispensable feature of the scalability of machine learning algorithms used for sorting. A popular approach to modelling the ranking problem is to take into account pairwise preferences. In this context, a sample set is sorted according to the estimated utility scores, while the objective is to minimize the number of false misclassifications in the generated sequence.

Rankprop tries to learn a ranking function while training a model to predict target ranks simultaneously [1]. They also combined multi-task learning with

DOI:10.7546/CRABS.2022.10.03

the resulting Rankprop algorithm and achieved a decrease of more than 25% in the sum of squares error. Burges et al. [2] extended the backpropagation [3] to handle ordered pairs together with the probabilistic cost function. The resulting algorithm, using simple gradient descent, is called RankNet. Based on RankNet, LambdaMART, which is the boosted version of LambdaRank, won the track 1 of the 2010 Yahoo! Learning To Rank Challenge [4].

Learning a ranking function from the labelled data requires the loss function to be evaluated. In the case of a mean squared error function as a performance criterion, this requires squared errors to be summed over the training data many times. Gradient-based methods should also calculate the gradient at each step of the training process. Numerical optimization techniques such as quasi-Newton methods and conjugate gradient, on the other hand, require second-order information such as the Hessian matrix in addition to the gradient [5].

As available data sets become large and redundant with higher dimensionality, second-order methods are ineffective in almost all use cases. In contrast, gradient methods have a significant advantage in large and redundant data sets with higher dimensionality. Simple stochastic gradient descent generally outperforms sophisticated second-order batch methods as it processes one or a few samples at each step. Therefore, the computational requirements of stochastic methods are minimal compared to second-order optimization methods (e.g., [6–8]).

The inclusion of momentum speeds up the convergence but provides a second parameter whose value must be selected in addition to the learning rate. With the help of local quadratic approximation around a local minimum, recent work has focused on the analysis of quadratic error functions [9–11].

This paper first examines the momentum factor and learning rate intervals where gradient descent is stable for the minimization of a pairwise ranking loss. Consequently, near-optimal learning rate and a corresponding momentum factor are estimated from the largest and smallest eigenvalue of the Hessian matrix using *power iteration.*

**2. General framework.** Let us assume that the sequence $Z = ((x_1, y_1), \ldots, (x_m, y_m)) \in Z^m$ is drawn from a probability distribution defined on the sample space $Z = \mathbb{R}^n \times \mathbb{R}$. An example $z = (x, y) \in Z$ is a pair consisting of an $n$-dimensional column vector of real-valued features, and a corresponding real-valued utility score. $X \in \mathbb{R}^{n \times m}$ denotes the $n \times m$ input matrix whose columns include the feature representations of the training examples, and $y \in \mathbb{R}^m$ is a column vector of the corresponding utility scores.

The task is to learn a ranking function $f\colon \mathbb{R}^n \to \mathbb{R}$ from the data. In other words, given an example $x_i$ preferred over a second example $x_j$, we require from the ranking function that $f(x_i) > f(x_j)$. We are interested in the ordering of the inputs according to their relevance concerning the target class. This process is similar to web search engines and recommender systems where the task is to generate a proper order in which to present web pages or products to the end user. We consider the linear case where the ranking function is represented as $f(x) = x^T w$, where $w \in \mathbb{R}^n$ is a vector of parameters.

The empirical risk, which is the normalized pairwise least-squares loss over the training data, is formally defined as

$$\frac{1}{2m} \sum_{i,j \in Q} (y_i - y_j - x_i^T w + x_j^T w)^2, \tag{1}$$

where $Q$ denotes the index set of the training set. Let

$$L = I - p^T p, \tag{2}$$

where $I$ is the $m \times m$ identity matrix and $p$ is an $m$-dimensional column vector whose entries are equal to $1/\sqrt{m}$. The matrix $L$ is a symmetric idempotent matrix and multiplying it with a vector removes the mean of the vector from all elements of the vector.

The empirical risk (1) then can be re-written in matrix notation as

$$(X^T w - y)^T L (X^T w - y) = (LX^T w - Ly)^T (LX^T w - Ly), \tag{3}$$

where the equality is due to symmetry and idempotence of $L$. This forms the basis for transforming the empirical risk term to a standard least-squares form equivalently expressed as

$$\|LX^T w - Ly\|^2, \tag{4}$$

whose minimizer is the best fit, in the least-squares sense, to the equation

$$LX^T w = Ly. \tag{5}$$

Regularized least-squares estimation of this equation can be written in the following form

$$(XX^T - Xpp^T X^T + \gamma I) w = X(y - pp^T y), \tag{6}$$

where $\gamma$ is a regularization parameter which controls the trade off between the model complexity and data fit. This is a set of linear equations of the type $Hw = b$, where $H$ is symmetric positive definite. Solving this linear system is equivalent to minimizing the deterministic quadratic loss $F(w) = w^T H w - b$, where $H = XX^T - Xpp^T X^T + \gamma I$, that is, the Hessian and $b = X(y - pp^T y)$.

Standard GDM is a method for minimizing the quadratic loss $F(w)$ and is written as

$$w_{t+1} = [(1 + \mu) I - (1 - \mu)\eta H] w_t - \mu w_{t-1} + (1 - \mu)\eta b. \tag{7}$$

This method is also very convenient for sparse equations since it depends on repetitive multiplication of a vector by sparse matrices. Also, the application of GDM is straightforward and requires little memory. However, the main disadvantage is that it has a slower convergence rate than other more complex algorithms. In order to improve the convergence rate of GDM, we need a stability analysis on GDM.

***2.1. Convergence characteristics.*** Applying the orthogonal transformation $x^T = Q^T x$, (7) becomes in coordinates as

$$w_{i,t+1} = [1 + \mu - (1-\mu)\eta\kappa_i]w_{i,t} - \mu w_{i,t-1} + (1-\mu)\eta b, \tag{8}$$

where $\kappa_i$, $i = 1, 2, \dots, n$, are the eigenvalues of symmetric positive definite matrix $H$. Then the coordinates of vector $w$ are obtained by the linear combination of the coordinates of $w'$. Including the dummy equation $w'_{i,t} = w'_{i,t}$, we can write (8) in matrix form:

$$\widetilde{w}'_{i,t+1} = A_i \widetilde{w}'_{i,t} + d_i, \widetilde{w}'_{i,t} = \begin{pmatrix} \widetilde{w}_{i,t-1} \\ \widetilde{w}_{i,t} \end{pmatrix}, \quad i = 1, 2, \dots, n, \tag{9}$$

where $A_i = \begin{pmatrix} 0 & 1 \\ -\mu & 1 + \mu - (1-\mu)\eta\kappa_i \end{pmatrix}$, $2 \times 2$ matrix and $d_i = \begin{bmatrix} 0 \\ (1-\mu)\eta b_i \end{bmatrix}$, 2-dimensional vector ($i = 1, 2, \dots, n$).

The linear dynamic system given by (9) is stable if the magnitudes of eigenvalues of the matrix $A_i$ are smaller than one [12]. Thus a relation is set up between the stability problem of GDM (7) and the magnitudes of eigenvalues of $A_i$ matrix. From the characteristic equation, we have that the $\lambda$ eigenvalues are the roots of the following quadratic equations [10]:

$$\lambda^2 - [1 + \mu - (1+\mu)\eta\kappa_i]\lambda + \mu = 0, \quad i = 1, 2, \dots, n. \tag{10}$$

Consequently, learning rate and momentum factor for fast convergence of GDM are given in the following formulas:

$$\eta^0 = \frac{1}{\sqrt{\kappa_1 \kappa_n}}, \quad \mu^0 = \frac{\left(\sqrt{\dfrac{\kappa_1}{\kappa_n}} - 1\right)^2}{\left(\sqrt{\dfrac{\kappa_1}{\kappa_n}} + 1\right)^2}, \quad \kappa_n > 0, \tag{11}$$

where $\kappa_1$ and $\kappa_n$ are the largest and smallest eigenvalues of the Hessian correspondingly. For the proof, we refer the reader to [13].

In order to determine the optimal learning rate and momentum factor in eGDM, one needs the largest and smallest eigenvalues of the Hessian. However, direct computation of these eigenvalues costs $\mathcal{O}(n^3)$ in the pairwise ranking setup. Therefore, we used the Power iteration to estimate these eigenvalues.

Let $e_{\max}$ be the corresponding normalized eigenvector of the largest eigenvalue $\kappa_1$ of $H$. If we generate a uniform random vector $\upsilon$ which is non-orthogonal to $e_{\max}$, then iterating the procedure

$$\upsilon \leftarrow H\mathcal{N}(\upsilon), \tag{12}$$

where the notation $\mathcal{N}$ designates the normalized vector $\upsilon/\|\upsilon\|$. This iterative procedure will make $\mathcal{N}(\upsilon)$ converge to $e_{\max}$, and $\upsilon/\|\upsilon\|$ converge to $\|\kappa_1\|$. In a few iterations (typically about 10), a reasonable estimation of $\kappa_1$ can be obtained. On the other hand, this method can be slightly modified to yield the smallest eigenvalue $\kappa_n$ of $H$. First, the largest eigenvalue $\kappa_1$ must be computed. Then, since we know that the Hessian is symmetric positive definite, using the spectral shift $H - \kappa_1 I$ and iterating

$$\omega \leftarrow (H - \kappa_1 I)\mathcal{N}(\omega), \tag{13}$$

will make $\omega$ converge smallest eigenvalue $\kappa_n$ of $H$. The reason this shift works is that a positive-definite matrix has all positive eigenvalues. Therefore $H - \kappa_1 I$ has all non-positive eigenvalues, with the smallest eigenvalue of $H$ now the largest-magnitude (most negative) eigenvalue of $H - \kappa_1 I$. The power method will then find that eigenvalue. The resulting power iteration algorithm adapted for the pairwise ranking setup is given in Algorithm 1.

Figures 1 and 2 demonstrate the estimation of $\kappa_1\kappa_n$ and $\kappa_1/\kappa_n$, respectively, using Algorithm 1. Empirically, sufficiently accurate values are obtained in a very short time (10 iterations). Furthermore, the cost of estimating these values can be reduced with an on-line version of Algorithm 1 which exploits the stationarity of

**Algorithm 1:** Power iteration for estimation of $\kappa_1$ and $\kappa_n$

```
Data: Input matrix $X$ and regularization parameter $\gamma$
Output: Largest eigenvalue $\kappa_1$ and smallest eigenvalue $\kappa_n$
$k \leftarrow 0$;
Initialize $\upsilon$, $\omega$ to random vectors of size $n$;
$\widetilde{\upsilon} \leftarrow \upsilon/\|\upsilon\|$, $\widetilde{\omega} \leftarrow \omega/\|\omega\|$;
repeat
    $\upsilon \leftarrow XX'\widetilde{\upsilon} + \gamma\widetilde{\upsilon}$;
    $\kappa_1 \leftarrow \|\upsilon\|$;
    $\widetilde{\upsilon} \leftarrow \upsilon/\kappa_1$;
    $\omega \leftarrow XX'\widetilde{\omega} + \gamma\widetilde{\omega} - \kappa_1\widetilde{\omega}$;
    $\widetilde{\omega} \leftarrow \omega/\|\omega\|$;
until $k < 20$;
$\kappa_n = \dfrac{\omega'XX'\omega + \omega'\gamma\omega}{\omega'\omega}$;
```

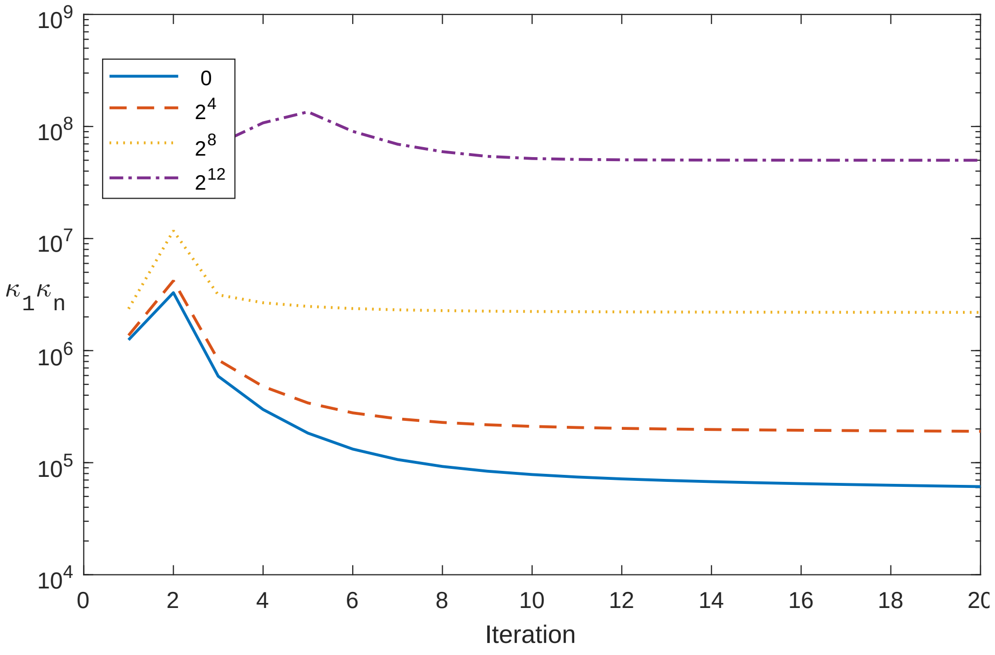

Fig. 1. Convergence of the eigenvalue estimation for $\kappa_1\kappa_n$

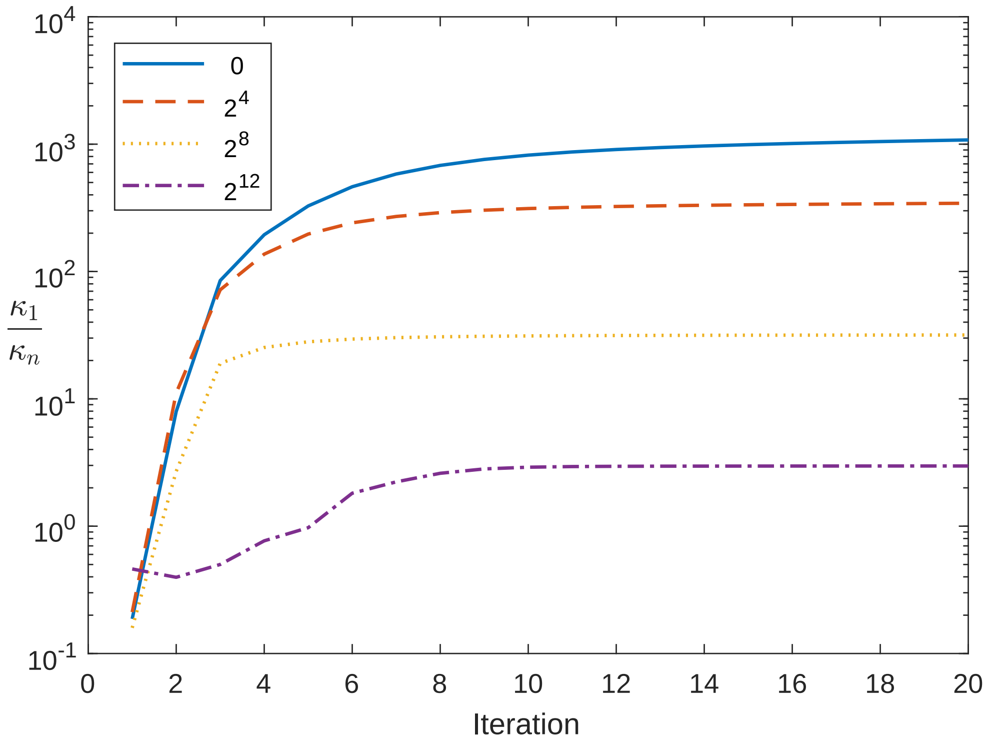

Fig. 2. Convergence of the eigenvalue estimation for $\kappa_1/\kappa_n$

**Algorithm 2:** eGDM with near-optimal learning rate and momentum factor

```
Data: Loss function F (1), input matrix $X$, outputs $y$, starting point
      $w_0$, regularization parameter $\gamma$ and convergence tolerance $\epsilon$
Output: The solution $w^*$
Compute $\kappa_1$ and $\kappa_n$ using Algorithm 2;
$\eta = \frac{1}{\sqrt{\kappa_1 \kappa_n}}, \quad \mu = \frac{\left(\sqrt{\frac{\kappa_1}{\kappa_n}} - 1\right)^2}{\left(\sqrt{\frac{\kappa_1}{\kappa_n}} + 1\right)^2}, \quad \kappa_n > 0;$
repeat
    $\nabla F(w_t) = Aw_t - b;$
    $w_t = -(1 - \mu)\eta \nabla F(w_t) + \mu(w_t - w_{t-1});$
until $\|\nabla F(w_t)\| < \epsilon;$
```

the second-order information over large and redundant data sets. Finally, GDM is adapted for pairwise ranking setup using these near-optimal parameters, called eGDM, and given in Algorithm 2.

**3. Experiments.** We have established an experimental setup on a document ranking task using the Reuters dataset constructed from the Reuters RCV1 collection [14] consisting of close to 800 000 examples and high dimensional (47 152 features) and sparse. Features are created by extracting the TF/IDF values of the documents. The task is to rank higher the documents in the CCAT category than other documents not belonging to this category.

Considering the ranking performance of eGDM with respect to regularization parameter $\gamma$, as $\gamma$ increases, the model complexity decreases and this leads to an increase in the ranking error of the test set. This is also the case for GDM, that is, for a given $\gamma$, both of the algorithms achieved the same ranking accuracy on the test set. However, when we compare the time of convergence to these accuracy levels, eGDM seems to perform much better than GDM, especially at low and moderate regularization levels. There is no statistically significant difference between the algorithms when the regularization is very high. Once again, we note that, in case of a high regularization, the model becomes so simple that it leads to poor data fit which can be seen by considering Fig. 1 and 2 simultaneously.

We know that the rate of convergence of GDM depends on the condition number of the Hessian ($H$) of (7). We can verify that the condition number of $H$ is $C = \frac{\kappa_1 + \gamma}{\kappa_n + \gamma}$. As the level of regularization increases $C$ approaches one from above. This is the case we observe in the experiments, with large enough values of $\gamma$, GDM converges very fast. On the other hand, for very small values of $\gamma$

the number of iterations required can be very large for GDM, whereas eGDM converges in few iterations. eGDM is unaffected by the difficulty of the problem and converges to the solution in a very short time. This indicates the robustness of eGDM regardless of the condition number of the Hessian.

**4. Conclusions.** Learning to rank has an essential role in machine learning since ranking associates queries with documents. Several ranking methods try to minimize a pairwise ranking error. However, directly minimizing a pairwise ranking error is computationally intractable. Therefore, a pairwise least-squares approximation is used to overcome this difficulty which leads to a linear equation system to be solved. Solving this linear system is equivalent to minimizing a quadratic loss. A fast version of gradient descent with near-optimal learning rate and momentum is proposed to minimize this quadratic loss. These near-optimal learning parameters are determined using the largest and smallest eigenvalues of the Hessian matrix. Power iteration is adapted to estimate these eigenvalues efficiently. Resulting algorithm eGDM outperforms standard GDM with respect to convergence times regardless of the difficulty of the problem and the level of regularization. Notably, for lower values of regularization eGDM converges much faster than GDM. Though we have mainly tried to focus our considerations on a deterministic quadratic loss, eGDM can be extended to work in the case of a stochastic quadratic loss. This is an open avenue for future research.

## REFERENCES

[1] Caruana R., S. Baluja, T. Mitchell (1996) Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In: Advances in Neural Information Processing Systems (eds D. Touretzky, M. C. Mozer, M. Hasselmo), **8** (NIPS 1995), 959–965.

[2] Burges C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds et al. (2005) Learning to rank using gradient descent. In: Proc. 22nd Int. Conf. on Machine Learning, ACM, 89–96.

[3] LeCun Y., L. Bottou, G. B. Orr, K.-R. Müller (1998) Efficient backprop. In: Neural Networks: Tricks of the Trade, Springer, 9–50.

[4] Chapelle O., Y. Chang (2011) Yahoo! learning to rank challenge overview. In: Proc. Learning to Rank Challenge, 1–24.

[5] Dennis J. E., Jr., R. B. Schnabel (1996) Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Classics in Applied Mathematics, **16**, SIAM, ISBN 0898713641.

[6] Bottou L. (2010) Large-scale machine learning with stochastic gradient descent. In: Proc. COMPSTAT'2010, Springer, 177–186.

[7] Polyak B. T., A. B. Juditsky (1992) Acceleration of stochastic approximation by averaging, Siam J. Control Optim., **30**, 838–855.

[8] Xu W. (2011) Towards optimal one pass large scale learning with averaged stochastic gradient descent, CoRR, abs/1107.2490, `http://arxiv.org/abs/1107.2490`.

[9] Bhaya A., E. Kaszkurewicz (2004) Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method, Neural Networks, **17**, 65–71.

[10] Qian N. (1999) On the momentum term in gradient descent learning algorithms, Neural Networks, **12**, 145–151.

[11] Torii M., M. T. Hagan (2002) Stability of steepest descent with momentum for quadratic functions, IEEE Trans. Neural Netw., **13**, 752–756.

[12] Brogan W. L. (1991) Modern Control Theory (3rd ed.), Upper Saddle River, NJ, USA, Prentice-Hall, Inc., ISBN 0-13-589763-7.

[13] Taş E., M. Memmedli (2017) Near optimal step size and momentum in gradient descent for quadratic functions, Turk. J. Math., **41**(1), 110–121.

[14] Lewis D. D., Y. Yang, T. G. Rose, F. Li (2004) RCV1: A new benchmark collection for text categorization research, J. Mach. Learn. Res., **5**, 361–397.

*Department of Statistics*
*Faculty of Science and Literature*
*Afyon Kocatepe University*
*Afyonkarahisar, 03200, Turkey*
*e-mail*: `engintas@aku.edu.tr`