Volume 6 Issue 11, 2020
DOI: 10.6919/ICJE.202011_6(11).0046

Different Loss Functions Used in the Low-rank Approximation

Chenyu Shi[1]; Ge Mou[2]; Haoyuan Wang[3]; Tianyu Liu[4]; Yifan Yang[5]; Zhengyu Li[6]

[1]School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, 710049, China;

[2]College of Computer Science and Technology, Northeastern University, Shenyang, Liaoning, 110819, China;

[3]College of mathematic, Jilin University, Changchun, Jilin, 130012, China;

[4]Guangzhou Foreign Language School, Guangzhou, Guangdong, 511458, China;

[5]School of computer and Cyberspace Security, Hainan University, Haikou, Hainan, 570228, China;

[6]College of Electrical Engineering, Shanghai DianJi University, Pudong, Shanghai, 201306, China.

Abstract

With the rapid development of science and technology, modern mechanical learning is not only limited to artificial intelligence, it has been involved in many other fields. A given algorithm will have many parameters, some are known, and some are unknown. But those unknown parameters are difficult to be solved by classical mathematical methods. Because these parameters often need a lot of data, which are obtained through a series of statistical operations. Therefore, a learning based algorithm is concerned in recent years. The advantage of this algorithm is that it can adjust the parameters according to the input data and improve the efficiency of some common algorithm tasks. Through the optimization of loss function (1905.11528), this work can improve the training speed, accuracy and data utilization rate, and use neural network to solve the problem of delayed feedback of continuous training in CTR prediction (1907.06558), which also inspires us to think about new loss function and neural network training. The former discusses the use of loss function to improve the efficiency of the algorithm, and the latter discusses the method of neural network. Weighted low rank approximation is mainly about the application of singular value decomposition (SVD) in low rank approximation. SVD is the key method to understand the properties of matrix. In the algorithm mentioned in this work, it is used as well. In this paper, the work will use different loss functions to solve the low order problem. The work is in order to find the best solution of matrix low order decomposition.

Keywords

Loss function, Low-rank approximation, SVD.

1. Introduction

1.1 Background

The success of modern machine learning makes it suitable for problems beyond the range of “classic artificial intelligence”. Particularly, people have more and more interests to improve the property of “standard “algorithms by adjusting their behavior, and it can fit the properties of the input distribution through using machine learning. [1-6] This learning-based algorithmic design approach has attracted considerable attention in the past few years, because it has the potential to improve the workpiece ratio of some widely used algorithmic tasks significantly. Many applications concern dealing with the data streams (videos, data logs, the customer activities, etc.) by performing the same algorithms per hour. These datasets are usually not “randomly” or not the “worst case”; instead, they are coming from certain distributions and can not change rapidly through the executions. This can make it more possible to design a better algorithm which is suitable for a given data distribution that are trained based on the examples in the past problems. This kind of method is quite successful in the context of compressive sense.

1.2 Related work

Just as overviewed in the introduction, machine learning has been applied in a number of related-papers in order to ameliorate the higher cost of ordinary used algorithms. Learning-Based Low-Rank [7-9] Approximations is highly connected with the topic of our paper. In this paper, the authors discuss the use of machine learning in Low-Rank Approximations [10-11] and give a loss function. In our paper, we come up with a new loss function and make a comparison to the previous one so that we can make this algorithm even more efficient.

Improved training speed, accuracy, and data utilization through loss function optimization (1905.11528) [12] and Addressing Delayed feedback for continuous training with neural networks in CTR prediction (1907.06558) [13] also enlighten us to think up the new loss functions and the training for neural networks. The former discusses about the use of loss function to improve the efficiency of an algorithm and the latter talks about the method for neural networks.

Weighted Low-Rank Approximations is mainly about the use of SVD (Singular Value Decomposition) in Low-Rank Approximations. SVD is the key method for us to learn about the character of a matrix and it is used in our algorithm as well.

1.3 Our work

In our paper, we will use different loss functions to solve the low-rank problem. we do this to find the best solution for the low rank decomposition of the matrix.

2. Loss Function

In order to better understand our research ideas, we divide the whole section into four subsections. What are the loss functions respectively, the loss functions in the article of learning based low rank approximation, the two new loss functions we replaced, and the relations and differences between these loss functions.

2.1 What is Loss Function?

In this part, we will explain the loss function in four steps. The first is the addition and subtraction of matrix. The second is what the norm is. The third is the most important part, what is the loss function. Finally, I would like to give some examples.

2.1.1. Addition and Subtraction of Matrix

First of all, when calculating the loss function, we will inevitably encounter the problem of matrix addition and subtraction. Matrix addition and subtraction is a mathematical term. In mathematics, matrix

addition generally refers to the operation of adding the corresponding elements of two matrices together, while subtraction generally refers to the operation of subtracting the corresponding elements of two matrices.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1m}+b_{1m} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2m}+b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \cdots & a_{nm}+b_{nm} \end{bmatrix} (1)$$

Add the first item $a11$ of the first matrix and the second item $b11$ of the second matrix together to get the $a11$ plus $b11$ in the first item of the sum matrix. By analogy, all elements of the matrix are added to the corresponding elements and the resulting elements are placed in the corresponding positions.

2.1.2. What is Norm?

Norm is a basic concept in mathematics. In functional analysis, it is defined in normed linear space and satisfies certain conditions, that is, the first nonnegativity, the second homogeneity and the third trigonometric inequality. It is often used to measure the length or size of each vector in a vector space or matrix. What we take here is the measure of norm to matrix. There are some examples of norm.

(1) F-norm: It is to square every element in the matrix, then add the sum of the elements after the square, and finally open the root sign to get the f norm.

$$\|A_{n\times m}\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2\right)^{\frac{1}{2}} \tag{2}$$

(2) V-norm: It is to first separate each row in the matrix. First, we make two norms for each row vector, that is, we square each element in the row vector, then add it, and then open the root sign. This is the value of the 2-norm we get for each row of vectors. We add together the values of the 2-norm for each row we get.   Finally, the v-norm of a matrix is obtained.

$$\|A_{n\times m}\|_v = \sum_{i=1}^{n}\left(\sum_{j=1}^{m} a_{ij}^2\right)^{\frac{1}{2}} \tag{3}$$

(3) W, F-norm: Similar to the F-norm, we first square each element in the matrix. But there is a different operation from the F-norm here, which is to multiply the corresponding weight before the number we want to arrive after the square and then add the sum. Finally, we get the W, F-norm of the matrix.

$$\|A_{n\times m}\|_{\{W,F\}} = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}a_{ij}^2\right)^{\frac{1}{2}} \tag{4}$$

It is easy to verify that these three norms are all in accordance with the above three norm axioms, that is, nonnegativity; homogeneity; satisfaction of trigonometric inequality. That is to say, the three definitions are well defined. After we understand the basic definition of norm, we will learn more about what is loss function.

2.1.3. What is Loss Function?

In mathematical optimization and decision theory, loss function is a function that maps one or more values of one or more variables to real numbers, intuitively represents some “costs” associated with events, and an optimization problem seeks to minimize loss function. This definition seems very abstract. Loss function is used to evaluate the difference between the predicted value and the real value of the model. The better the loss function is, the better the performance of the model is. The loss functions used in different models are generally different. In the same model, the smaller the value of loss function is, the closer the value of integer parameter is to the value we want. In statistics, the loss function is usually used for parameter estimation, and the event in question is a function of the difference between the estimated value and the real value of the data instance. In the same model, the smaller the value of loss function is, the closer the value of integer parameter is to the value we want. Our work formally uses the property of loss function to estimate the parameters of the whole experiment.

2.1.4. Examples

Here are three loss functions constructed using the norms just introduced. We call them F-loss function, v-loss function and W, F-loss function respectively. Because of the regularity of norm, the whole loss function is a positive value. The greater the difference between A and A', the greater the value of the norm. So we can reduce the error between A' and A by adjusting the approximation value of a', and then reduce the value of loss function. When the value of the loss function reaches the minimum, it means that some "costs" associated with the event are minimum. That's exactly what loss function defines.

(1) F-no$||A - A'||_v$rm:$||A - A'||_F$

(2) V-norm:

(3) {W,F}-norm:$||A - A'||_{\{W,F\}}$

2.2 Function in the Learning-Based Low-Rank Approximations

In this section, we'll explain what the loss function is in the learning-based low-rank approximation article. And how to apply it.

2.2.1. Old function

The loss function in this paper is defined by F-norm. Subtract the original matrix from the approximation matrix and make the F-norm. The SCW algorithm here is the approximation algorithm in this paper. Take the matrix calculated by SCW algorithm into A'. Do the above for each test matrix, and then add all the resulting norms. Then we get the formula

$$\sum_i ||A_i - SCW(s^*, A_i)||_F \quad (5)$$

2.2.2. Interpretion

This formula also plays an important role in the article. In this paper, the loss function is used as parameter adjustment. This Ai is the a of the above formula, and SCW is the A of the above formula. This loss function represents the sum of the distances between all samples Ai and the matrix approximated by SCW algorithm. The smaller the function value of this function, the better the approximation effect of SCW is. The SCW algorithm is affected by adjusting the sketch matrix S, and the value of loss function is also affected. When we adjust the sketch matrix to minimize the loss function, S is the matrix we want to get.

2.3 New loss function

Having said the loss function the artical used, it's our turn to use the new loss function. V-norm and W, F-norm are used in our experiment. These two loss functions are defined in exactly the same way as the loss functions mentioned before, except that we change the original norm to a new one.

2.3.1. V-loss function

Let me start with the derivation of the v-loss function. First, the approximation matrix of SCW algorithm is brought in like the previous function, and then all the matrices are added. For the sake of illustration, we note B as the matrix approximated by SCW algorithm. So every matrix can be written like this

$$\|A_i - B_i\|_v = \begin{Vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{Vmatrix} (6)$$

Expand the V norm to get this formula. And we get the loss function of the whole.

$$\sum_{i}\|A_i - SCW(s^*,A_i)\|_{v} = \sum_{i}[\sum_{i=1}^{n}(\sum_{j=1}^{m}(a_{ij}-b_{ij})^2)^{\frac{1}{2}}] \tag{7}$$

What we need to do in our research is to use SCW algorithm to bring the approximation of matrix into these two equations. Then determine sketch matrix.

2.4 Comparison

In the article of learning based low rank approximation, we use three directions: learned, sparse random and dense random to compare the function. In our experiment, we will also compare these three aspects. The specific comparison process and results will be explained in the experience.

3. Preliminaies

3.1 SVD

Singular value decomposition. We first show the basic method of eigenvalue decomposition of a matrix.

Theorem 1. Consider a square matrix A ∈ Rmxm. A can be decomposed into $A = U\Lambda U^{-1}$. Particularly, if A is a symmetric matrix, it can be decomposed into $A = U\Lambda U^T$. And each column of U is the eigenvector of A. And Λ∈Rmxm, and the elements in the diagonal of $\Lambda$are the eigenvalues of A. Eigenvalue decomposition gives the method to decompose a square matrix. However, if the matrix is not square, how can we decompose it? According to Learning-Based Low-Rank Approximations, SVD (Singular Value Decomposition) can be used to decompose general matrix.

Theorem 2. Consider a matrix A ∈ Rmxn, if $AA^T = P\Lambda_1 P^T, A^T A = Q\Lambda_2 Q^T$, then A can be decomposed into $A = P\Sigma Q^T$. Each column of P is the eigenvector ofAA^T, and each ceigenvalues of AA^T, and the elements in the diagonal of $\Lambda_2 \in R^{n\times n}$are the eigenvalues of $A^T A$. Particularly, these two matrix share the same nonzero eigenvalues.

And they are defined as λ_i (i=1…k)

Suppose the rank of A is k. The elements in the diagonal of matrix $\Sigma \in R^{m\times n}$ are defined as the “Singular value”, and for $\sigma_i = \sqrt{\lambda_i}$ each σ_i (i=1…k),

Through SVD, we can decompose general matrixes into their singular values. Then how do we use it in low-rank approximations?

3.2 Previous Works

For the past few years, there has been a lot of papers using the methods which is machine learning to improve the performance of a kind of algorithm we called “standard” algorithms. Our paper’s main topic is closest to the papers worked on the compressive sensing based on the machine learning, and the streaming algorithms based on machine learning. Because we use the computer matrix spectra, these topics are really different from ours.

We concentrate on the optimization of low-rank approximation algorithms based on machine learning which use linear sketches. There are other algorithms about the sketching are not linear. The advantages of the linear is that the matrix can get linear change easily. Our paper is mainly about the low-rank approximation that use the different loss functions.

Notations We supposed there is a matrix A$\epsilon\mathbb{R}^{n\times d}$, and there is a distribution D on matrix A. We define a training sampled from the distribution D. The SVD of matrix A can be written as A = UΣVT such that both U$\epsilon\mathbb{R}^{n\times n}$and V$\epsilon\mathbb{R}^{d\times n}$have orthonormal columns and Σ = diag {λ1,…,λd } is a diagonal matrix and it has nonnegative entries. Moreover, if we say rank (A) = n, then the first n columns of U in the column place of A will be an orthonormal basis, the first n columns of V in the row place of A will be an orthonormal basis for the and λi = 0 for i > n. In many applications, the advantages of this algorithm is faster and more economical to compute the compact SVD, this kind of SVD only contains the rows and columns corresponding to the non-zero singular values of Σ :$A = U^{C}\Sigma^{C}(V^{C})^{T}$. How sketching works. At the start, We describe the SCW algorithm for low-rank matrix approximation. This algorithm computes the SVD of $SA = U\ \Sigma V$, and compute the optimal rank-k approximation of AV. At last it will output [AV]kV T as a rank- k approximation of A.. In this section, we follow the framework (Clarkson and Woodruff, 2009) but use learned matrices.

Algorithm 1 Rank-k approximation of a matrix A using a sketch matrix S. [14]

```
1.Input: $A \in \mathbb{R}^{n\times d}, S \in \mathbb{R}^{m\times n}$
2.$U, \sum, V^{\perp} \leftarrow COMPACTSVD(SA)$    $\triangleright$    $\{r = rank(SA), U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{d\times r}\}$
3.Return: $[AV]_k V^{\perp}$
```

Sketching matrix. The identify of matrix will not be introduced in this paper. This part is defined in *Learning-Based Low-Rank Approximations (Indyk and Vakilian)*.

Training Algorithm. In this part, we will briefly summarize the algorithm of learning based calculation data and related sketch matrix s, whose main idea is to use learning feedback algorithm to calculate the random gradient of S. Once we have a random gradient, we can run a random gradient descent (SGD) algorithm to optimize s to minimize the loss. The advantage of this algorithm is that it keeps the sparse structure of S and only optimizes the values of N non-zero entries (initially 1 or -1).

Algorithm 2 Differentiable SVD implementation

```
1:Input: $A_1 \in \mathbb{R}^{m\times d}$ where m<d
2:$U, \sum, V \leftarrow \{\}, \{\}, \{\}$
3:for i←1...m do
4:    $v_1$ ←random initialization in $\mathbb{R}^d$
5:    for t←1...T do
6:        $v_{t+1} \leftarrow \frac{A_i^T A_i v_t}{\|A_i^T A_i v_t\|_2}$ $\triangleright$ {power method}
7:    end for
8:    V[i]←$v_{T+1}$
9:    $\sum$ [i]←$\|A_i V[i]\|_2$
10:   U[i]←$\frac{A_i V[i]}{\sum[i]}$
11:   $A_{i+1} \leftarrow A_i - \sum[i] U[i] V[i]^T$
12:end for
13:Return: $U, \sum, V$
```

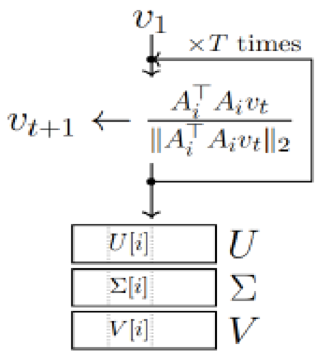

v_1
$\times T$ times
$v_{t+1} \leftarrow \frac{A_i^\top A_i v_t}{\|A_i^\top A_i v_t\|_2}$
$U[i]$
$\Sigma[i]$
$V[i]$
U
Σ
V

Figure 1: The expression of algorithm 2

We use singular value decomposition to make SVD differentiable. The specific algorithm is shown in the figure, which makes the gradient flow easily.

In order to solve the problem that it is impossible to write the explicit form or gradient of loss function because of the long calculation chain. We use the automatic gradient feature in py torch to calculate the gradient related to sketch matrix s.

This method only optimizes the training phase of S. After s complete training, we still call algorithm 1 for low rank approximation, which has the same running time as SCW algorithm and improves the quality of the returned rank-k matrix.

Finally, we refer to these methods and algorithms, then use different loss functions to deal with the low-rank approximations.

4. Experiment

4.1 Result

The main problem discussed in this paper is the utility of v-norm and {W,F} -norm in the low rank decomposition problem of sketch algorithm. To solve this problem, we use python programming software to change the rank k, of matrix and compare their utility through some known algorithms, using the v norm and the {W,F} -norm respectively in the sketch algorithm. The final result is by comparing the results we run after we find that in low rank decomposition {W,F} -norm and v-norm are very approximate in error, time and error rate, and the effect of both is very similar.

4.2 Estimate

Consider a-[*a*]*k* ***f*** as the best k-rank approximation of the matrix, and consider a-scw(s,a) ***f*** - a-[*a*]*k* ***f***

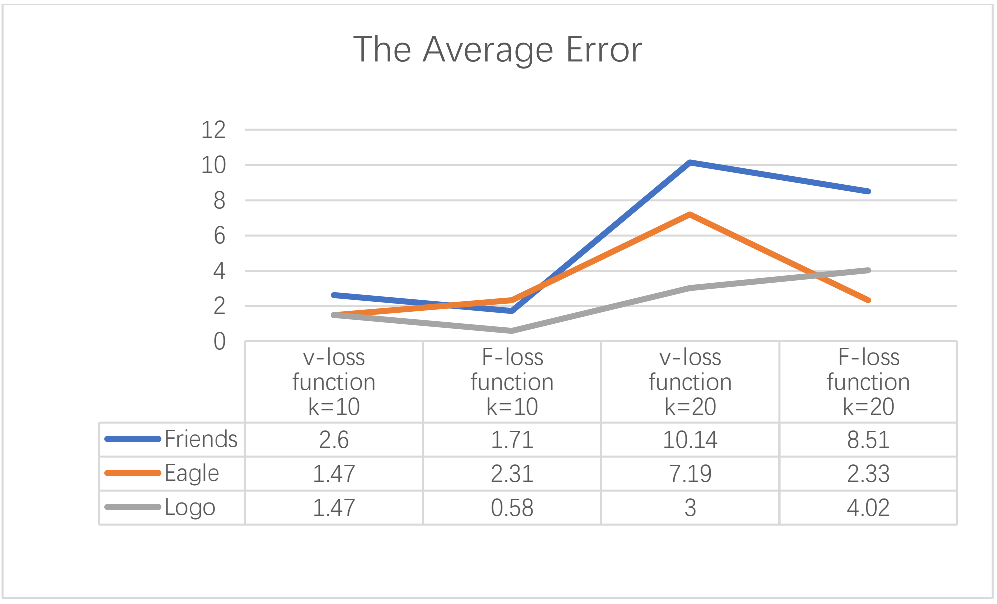
as the error of s. Because we consider s as the weight matrix w, the error is also for the weight matrix w.

Table 1: The Average error

	v-loss function k=10	F-loss function k=10	v-loss function k=20	F-loss function k=20
Friends	2.60	1.71	10.14	8.51
Eagle	1.47	2.31	7.19	2.33
Logo	1.47	0.58	3	4.02

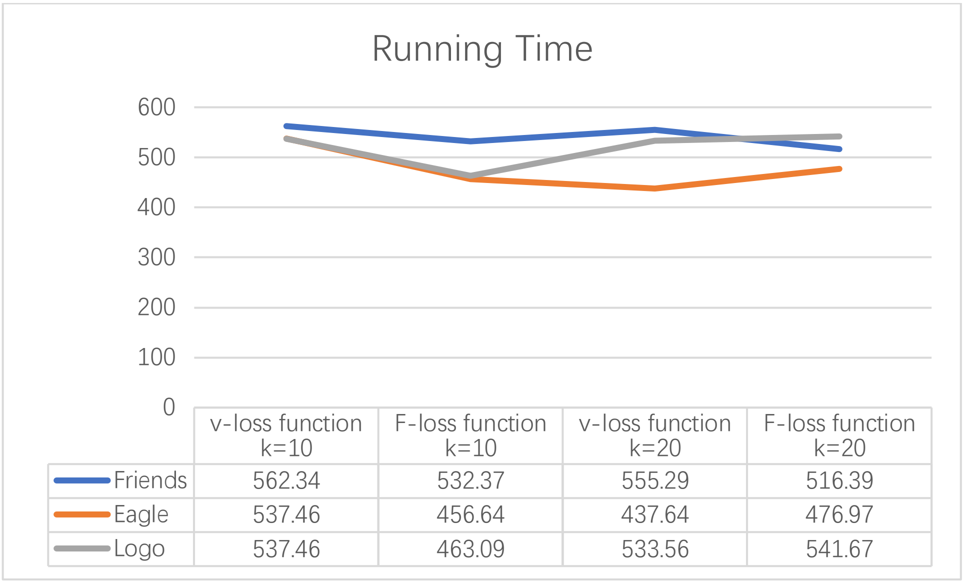
Table 2: Running time

	v-loss function k=10	F-loss function k=10	v-loss function k=20	F-loss function k=20
Friends	562.34	532.37	555.29	516.39
Eagle	537.46	456.64	437.64	476.97
Logo	537.46	463.09	533.56	541.67



	v-loss function k=10	F-loss function k=10	v-loss function k=20	F-loss function k=20
Friends	2.6	1.71	10.14	8.51
Eagle	1.47	2.31	7.19	2.33
Logo	1.47	0.58	3	4.02

Figure 2:The figure of table 1



	v-loss function k=10	F-loss function k=10	v-loss function k=20	F-loss function k=20
Friends	562.34	532.37	555.29	516.39
Eagle	537.46	456.64	437.64	476.97
Logo	537.46	463.09	533.56	541.67

Figure 3:The figure of table 2

5. Conclusion

A comparison of two loss functions, based on an existing method developed by A. Mousavi, A. B. Patel, and R. G. Baraniuk. New loss function can not significantly improve training speed, accuracy, and data utilization.

In this paper we compare the utility of {W,F} norm and V norm in sketch algorithms. The weight matrix of {W,F} norm is replaced by a learning-based Matrix S. Using several different data sets, We compare the average errors and running time of these two norms by change the rank of the matrixes. They show similar effects.

Future research could extend our proposed approach and can explain the similar effect of loss functions. More efforts are needed to create a new function to achieve better process performance.

Acknowledgements

We thank CIS reviewers for their useful comments and language editing which have greatly improved the manuscript. The authors are in alphabetic order.

References

[1] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. Proceedings of the IEEE, 104(1):34–57, 2016.

[2] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In Advances in Neural Information Processing Systems, pages 6348–6358, 2017.

[3] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. In Proceedings of the 2018 International Conference on Management of Data, pages 489–504, 2018

[4] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. Learning to branch. In International Conference on Machine Learning, pages 353–362, 2018.

[5] M. Purohit, Z. Svitkina, and R. Kumar. Improving online algorithms via ml predictions. In Advances in Neural Information Processing Systems, pages 9661–9670, 2018.

[6] T. Lykouris and S. Vassilvitskii. Competitive caching with machine learned advice. In International Conference on Machine Learning, pages 3302–3311, 2018.

[7] A. Mousavi, A. B. Patel, and R. G. Baraniuk. A deep learning approach to structured signal recovery. In Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on, pages 1336–1343. IEEE, 2015.

[8] L. Baldassarre, Y.-H. Li, J. Scarlett, B. G¨ozcu¨, I. Bogunovic, and V. Cevher. Learning-based compressive subsampling. IEEE Journal of Selected Topics in Signal Processing, 10(4):809–822, 2016.

[9] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed sensing using generative models. In International Conference on Machine Learning, pages 537–546, 2017.

[10] Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-Based Low-Rank Approxima- tions. 2019.

[11] Nathan Srebro and Tommi Jaakkola. Weighted Low-Rank Approximations. 2003

[12] Santiago Gonzalez and Risto Miikkulainen. Improved Training Speed, Accuracy, and Data Utilization Through Loss Function Optimization. Proceedings of the 2020 IEEE,2019.

[13] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilip- kumar, and Ferenc Huszar. Addressing Delayed Feedback for Continuous Training with Neural Net- works in CTR prediction. 2019.

[14] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In Proceedings of the forty-first annual symposium on Theory of computing (STOC), pages 205–214, 2009.