Skolkovo Institute of Science and Technology

MATRIX FACTORIZATION METHODS FOR TRAINING EMBEDDINGS IN SELECTED MACHINE LEARNING PROBLEMS

Doctoral Thesis

by

Alexander Fonarev

Doctoral Program in Computational and Data Science and Engineering

Supervisor

Prof. Ivan Oseledets

Moscow

2018

Abstract

One of the essential problems in machine learning that appears in many practical applications is the problem of representing complex objects as real-valued low-dimensional vectors, so-called embeddings, using collected statistical data. Methods that build and learn embeddings have a significant amount of various applications — from information retrieval and recommender systems to natural language processing and computer vision.

While most of popular embedding methods are based on neural network approaches, in this thesis, we introduce and thoroughly examine a low-rank matrix factorization framework for training embeddings, which often is more suitable because of a large number of effective and efficient computational techniques that are developed in the numerical linear algebra field. This framework generalizes existing embedding approaches based on low-rank approximations, and allowed us to develop several new methods for embedding learning that outperform state-of-the-art approaches.

Firstly, we propose the novel unsupervised method to learn embeddings of categorical features' values both for supervised and unsupervised machine learning problems. This approach allows us to encode the categorical features into real values in order to efficiently perform data analysis in various applications, such as security access management.

Moreover, we introduced and systematically investigated the general formulation of the word embeddings training problem based on Skip-Gram Negative Sampling that consists of two steps with clear objectives. These steps and objectives were not formulated in the literature before.

We also developed the new method to learn word embeddings based on the Riemannian optimization technique that outperforms the existing state-of-the-art approaches

dealing with approximate objectives of the embeddings training problem. The developed method has been implemented, tested and shared on the Web as open source.

Furthermore, we developed the method to obtain embeddings of cold users and items from recommender system data. The developed method outperforms the existing state-of-the-art approaches in terms of quality and computational speed and also does not have their limitations by design. This methods was also implemented in Python and publicly shared on the Web. Moreover, the developed method has one more indicator of its practical significance — it is used as one of the sources of features for the industrial recommender system at Yandex.Music.

Publications and Talks

The results were published in the papers:

1. **Fonarev, A.** (2014, August). Transformation of Categorical Features into Real Using Low-Rank Approximations. In Russian Summer School in Information Retrieval (pp. 253-262). Springer, Cham. *The whole work was done by the author.*

2. **Fonarev, A.**, Mikhalev, A., Serdyukov, P., Gusev, G., & Oseledets, I. (2016, December). Efficient Rectangular Maximal-Volume Algorithm for Rating Elicitation in Collaborative Filtering. In 16th International Conference on Data Mining (pp. 141-150). IEEE. *Top rank A* computer science conference. Contribution of the authors: the first author — the problem formulation, the existing approaches analysis, designing the experiments, writing the text (partially), conducting the experiments (partially), the method developement (partially); the second author — writing the text (partially), conducting the experiments (partially), the method developement (partially); the third author — writing the text (partially); the fourth author — writing the text (partially); the fifth author — initial idea proposition, writing the text (partially).*

3. **Fonarev, A.**, Hrinchuk, O., Gusev, G., Serdyukov, P., & Oseledets, I. (2017, August). Riemannian Optimization for Skip-Gram Negative Sampling. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers, pp. 2028-2036). Association for Computational Linguistics. *Top rank A* computer science conference. Contribution of the authors: the first author — initial idea proposition, the problem formulation, the exist-*

ing approaches analysis, designing the experiments, the method developement, writing the text (partially), conducting the experiments (partially); the second author — writing the text (partially), conducting the experiments (partially); the third author — writing the text (partially); the fourth author — writing the text (partially); the fifth author — writing the text (partially).

The results introduced in the thesis were presented at top international conferences:

1. Annual Meeting of Association of Computational Linguistics 2017, August (top rank A* computer science conference), Vancouver, Canada;
2. International Conference on Data Mining 2016, December (top rank A* computer science conference), Barcelona, Spain;
3. International Conference on Matrix Methods and Applications 2015, August, Skolkovo, Russia.
4. Russian Summer School in Information Retrieval 2014, August, Nizhniy Novgorod, Russia.

The results were also presented at scientific seminars:

1. The joint seminar of Intellectual Systems department and Machine Intelligence Laboratory of Moscow Institute of Physics and Technology, April 2018, Moscow, Russia;
2. The seminar of the 72nd department of Federal Research Center "Informatics and Management" of Russian Academy of Sciences, May 2018, Moscow, Russia.

Contents and Structure of Thesis

The thesis consists of 7 chapters. Chapter 1 introduces the embeddings training problem and overviews state-of-the-art and important applications of embeddings. Chapter 2 overviews some of the existing methods for embeddings learning and provides the background that is needed for understanding the methods described in the work. Chapter 3 introduces the relative simple but efficient low-rank matrix factorization framework to train embeddings that is a foundation for the new algorithms proposed in this work. Chapter 4 introduces a method for unsupervised transformation of categorical features' values into embeddings. Chapter 5 introduces a novel approach to train word embeddings based on Riemannian optimization. Chapter 6 presents an algorithm to learn embeddings of cold users and items in recommender systems. Chapter 7 concludes and summarizes the results of the thesis.

Acknowledgements

I am grateful to Alexander Mikhalev, Alexey Grinchuk, Gleb Gusev, Pavel Serdyukov, Denis Kolesnikov for productive joint work and research, without your help this thesis could not be possible. I also wanted to thank people who helped me with proofreading the thesis text: Alexey Grinchuk, Anna Potapenko, Ilya Solomatin, Alexander Katrutsa, Valentin Khrulkov. Special thanks to Victor Lempitskiy, Andrzej Cichocki, Evgeny Burnaev, Konstantin Vorontsov, Alexander Bernshtein for deep dive into the thesis ideas and very helpful comments. Moreover, I wanted to thank people who significantly helped me with the PhD process challenges that I encountered: Maxim Fedorov, Maxim Panov, Dmitry Artamonov, Evgeny Frolov, Daniil Merkulov, Andrei Chertkov, Ilya Sochenkov. A very special gratitude goes out to all PhD jury members that found time in their tough schedules to participate in my PhD defense: Andrzej Cichocki, Victor Lempitskiy, Dmitry Ignatov, Alexander Tuzhilin, Andre Uschmajew.

Especially, I wanted to acknowledge my first teachers who gave me the motivation to continue to work on math, information technology and data science: Alexander Spivak, Victor Matyukhin and Alexander Dyakonov. And also, of course, huge thanks to my family for support and giving me a chance of getting a good education. Furthermore, thanks for fruitful discussions on various topics that substantially formed my understanding of the data science field: Mikhail Gavrikov, Peter Romov, Konstantin Vorontsov, Anna Potapenko, Alexander Dyakonov.

To my scientific advisor, Ivan Oseledets for patience, flexibility, trust, openness, mentoring and support along the whole PhD path. Many thanks!

Thanks to everyone above and everyone who is not listed for your encouragement!

Contents

1 Introduction

In the modern world, more and more technological processes collect data about the behavior of various systems. For example, the spread of online social networks led to collecting huge amounts of data about users and spread of mobile phones led to obtaining a big amount of data about phone owners gathered via phones' sensors. Often, such data contains a lot of useful information that could be exploited for improving business processes, developing new products, providing better client service or gaining more revenue.

Usually, the collected data has a complicated structure and contains complex regularities, so it is expensive or even impossible to perform in-depth manual analysis of this data in order to make valuable decisions from it. Moreover, often data-driven decisions should be made instantly and frequently, so the manual decision-making process is not appropriate. That is why, data science and, more specifically, machine learning approaches become more and more demanded today. They focus on extracting hidden features and structures from the data automatically without any human intervention. Nowadays, machine learning solutions become a core part of a lot of businesses — online search engines, online stores, online social networks, online advertisement platforms, for banks, factories, automobile industry, etc.

1.1 Introduction to Embeddings

One of the essential directions within the machine learning field today is to represent complex objects or data by vectors. Such process is called embedding and allows us to store semantics of an object. As an example, let us look at a natural language processing (NLP). In the NLP, a word is just a sequence of letters, so processing this infor-

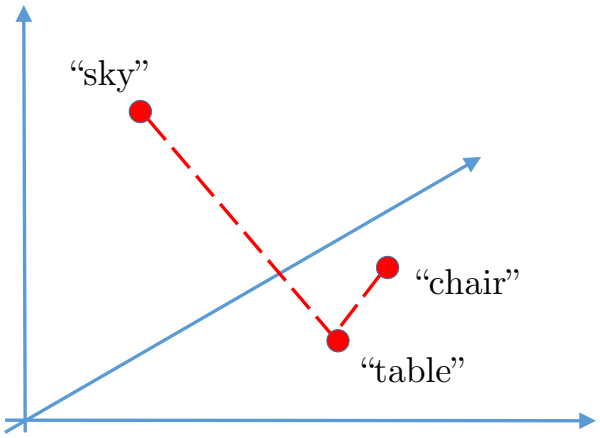

Figure 1: An intuition behind word embeddings. The 3-dimensional embeddings of word ”table” and ”chair” are closer to each other than embeddings of words ”table” and ”sky”, which is aligned with our understading of semantics of these words.

mation on a computer in order to make decisions based on the semantics of this word is difficult. However, having a big amount of collected natural language texts (e.g., all English Wikipedia texts) and analyzing co-occurrences of various words close to each other within these texts, it is possible to build a low-dimensional semantic representation of the word that would contain some aspects of the meaning. For example, having such word embeddings, a computer can understand that the word “table” is semantically more similar or related to the word “chair” than to the word “sky” (Figure 1 illustrates this idea). In general, embedding learning procedures could be applied to graphs, images, user preferences, product specifications, websites and other complex objects.

Let us formalize the concept descriped above. Let $\mathcal{A}$ be a set of objects that we want to build d-dimensional embeddings for. Then the embedding function f is the following

mapping:

$$f : \mathcal{A} \to \mathbb{R}^d. \tag{1}$$

There are three main reasons to train embeddings of complex objects in practical tasks:

1. **Aligned dimensionality of representations.** Most of statistical tools and systems (e.g. most of supervised learning methods) are able to handle objects which descriptions have the same dimensionality and the dimensions are aligned (have the same meaning).

2. **Real-valued components of representations.** Most of statistical tools and systems can handle objects which descriptions consists of real values.

3. **Low dimensionality of representations.** Usually, both for automated and manual object analysis, it is very useful to keep their descriptions low-dimensional because it either simplifies manual work or reduces computational complexity of algorithms. Moreover, having low-dimensional embeddings often provide an automatical de-noising effect that helps to avoid further overfitting and perform more precise data analysis.

In this work, we do not focus on the problem of embedding dimensionality d selection — it is a hyperparameter that could be selected based on performance metric of a task where the embeddings are used.

1.2 Using Embeddings in Practical Machine Learning Tasks

As mentioned above, embeddings of complex objects are helpful in many practical problems within various industrial processes. As it is illustrated in Figure 2, there are two main steps where embeddings are involved in solving business problems:

Step 1: Embeddings are learned from collected data. Building embeddings with a good performance is often a challenging problem. This work focuses on developing new methods to perform Step 1.

Step 2: The embeddings obtained on Step 1 are used in practical applications. Important to note that usually the same embeddings can be used in various related problems. For example, same word embeddings could be used in machine translation, speech recognition, news articles categorization, etc. So once an efficient embeddings learning method is developed, it can be applied for many tasks. That is one more reason, why the focus of the thesis is to improve the quality of existing approaches to build embeddings.

1.3 Measuring Embeddings Performance

As mentioned above, this thesis focuses on developing new embedding algorithms that outperform existing approaches in terms of performance. There are two main ways to measure the performance of embeddings.

1. **Using the performance measure of a target application.** As it was mentioned above, embeddings are used as a part of solutions for various tasks. Usually, a task has its own performance measures, so it is very natural to use them to estimate the performance of embeddings that are used to solve it. Despite straightforwardness, this approach's important limitation is the fact that there are many factors that influence on the target performance, so sometimes it is hard to distinguish gain obtained by embeddings from other factors. Moreover, in complex industrial

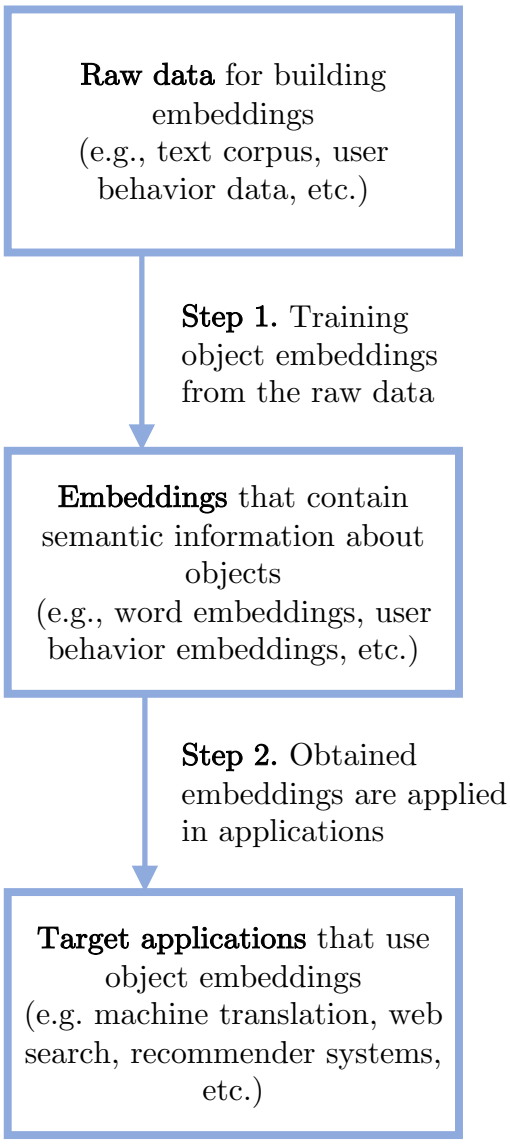

Figure 2: A common pipeline of using embeddings in practical problems. The embeddings are usually used on the second step illustrated on the scheme. The thesis focuses on developing new embedding methods for the first step.

processes, it might be costly to perform experiments over the whole system in order to estimate the quality of one part of it.

2. **Using an internal embeddings performance measure.** This approach implies constructing a separate performance measure for embeddings that should correlate with the performance in target applications where the embeddings are being used. For example, tasks often require to have embeddings of similar objects placed close to each other in the representation space. The level of fulfillment of this requirement could be measured using a separate dataset which contains pairs of objects with assessed levels of similarity between them. Moreover, for example, embeddings should satisfy additional properties, such as interpretability [115], which also could be measured and used as an embedding performance measure.

Experiments described in the thesis use both evaluation approaches.

1.4 Embeddings Application Examples

This section describes several real-life examples of embeddings in order to illustrate a wide range of applications and the fact that improving embeddings once will lead to improvements of several applications that use these embeddings.

1.4.1 Word Embeddings Applications

In most of the problems where natural language words are used for automated analysis, it is very convenient to have them in a real-valued vector format, such as one-hot-encoding or embeddings (which is usually more preferable), because such representations could be easily used by various machine learning and NLP algorithms. Thus, many

NLP tasks take advantage of learning high-quality embeddings. This could be language modeling tasks (predict next words given the previous), sequence tagging tasks (predict some tag for each word in a sequence, e.g., Named Entity Recognition), sequence to sequence tasks (predict a sequence of tags given a sequence of words, e.g., machine translation), and many more. A detailed survey of all possible applications goes beyond the scope of this thesis, but a few specific applications are listed below.

Named Entity Recognition. NER is an essential problem in natural language processing. An NER algorithm gets a text as an input and classifies all relevant named nouns that are mentioned in the text. For example, an NER algorithm could recognize names of people, cities, brands, etc. within the text. It has a lot of applications in various systems — from internet search engines and news articles analysis to automated lawyers and content recommenders.

A significant input for an NER algorithm is a context of a word that is being classified, so word embeddings store this information in a compact real-valued low-dimensional format. These embeddings could be used, for example, as features for a machine learning classifier which implements a NER logic. There are a large number of papers that discuss how embeddings can be applied in NER. For example, [26] combines word embeddings with categories data from Wikipedia, [102] uses word embeddings as features for learning algorithms, [105] adapts popular Word2Vec embedding learning systems to NER, etc.

Coreference resolution. In natural language processing, coreference resolution algorithms are widely used in applications related to text meaning understanding, e.g., in automated dialog systems. There are a lot of paper that use embeddings for the coref-

erence resolution problems. For example, authors of [106] use word embeddings as features in a machine learning classifier, [69] presents an end-to-end system based on word embeddings, [109] explores usage of word embeddings for coreference resolution in Russian language, [23] uses embeddings as features to detect coreference pairs of clusters and so on.

Textual Entailment. The problem of textual entailment focuses on understanding complex structures and entailments in natural language, which also is widely used in automated dialog systems. In fact, this is another field where word embeddings are successfully applied. For example, [5] does this for Arabic language, [107] uses this for analysis based in Twitter data, [129] uses embeddings in addition to attention and some composition improvements.

Question Answering. The problem of automated question answering becomes more and more demanded with a growth of automated personal assistants. Solutions of this task also widely use embeddings. For example, [104] proposes to merge embeddings and models inspired from machine translation field, [103] exploits causal information into account, [84] uses embeddings for questions retrieval.

1.4.2 User Behavior Embeddings Applications

Embeddings that correspond to user behaviours or preferences and describe their previous behaviour also are widely used in various applications. For example, they could be used for unsupervised analysis of users by clustering their embeddings, [116] describes methods to personalize web search using embeddings, [4] describes personalization of product search, [89] proposes methods to perform personalized lessons based on stu-

dents' embeddings, [61] explores methods to personalize automatic conversations using users' embeddings, while [60] overviews user embeddings methods to build recommender systems.

1.4.3 Other Embeddings Applications

There are many other types of embeddings that are used in various applications. For example, [54] develops methods to build embeddings of images, [124, 42] explore embeddings for queries in search engines, embeddings are used as an initialization of neural nets in speech recognition [65], sentence embeddings are used in machine translation [80], [60] explores items embeddings in recommender systems, product embeddings for searching close products to recommend are described in [43].

1.5 Chapter Summary

In this chapter, we introduced the embedding concept, explained how embeddings are usually applied to practical problems, overviewed some embedding method examples and formulated the objectives of the thesis. While most of embedding methods are based on neural network approaches, in this thesis, we focus on embedding algorithms based on low-rank matrix factorizations since often they have better properties and outperform state-of-the-art approaches. The next chapter describes how popular embedding methods work internally and gives the required background for understanding the following sections.

2 Background

2.1 Matrix Factorization Background

2.1.1 Low-Rank Matrix Factorization

Let $\mathbf{A}$ be a matrix from $\mathbb{R}^{n \times m}$. Then given a number $d \in \mathbb{N}$, called a factorization rank, $\mathbf{A}$ could be approximated as a product of two matrices $\mathbf{P} \in \mathbb{R}^{n \times d}$ and $\mathbf{Q} \in \mathbb{R}^{m \times d}$:

$$\mathbf{A} \approx \mathbf{P} \cdot \mathbf{Q}^{\top} = \mathbf{B}. \tag{2}$$

This approximation is called low-rank matrix factorization or low-rank matrix decomposition.

2.1.2 Matrix Similarity Measures

The factorization is usually performed based on a distance measure $\rho(\cdot, \cdot)$ between two matrices, which is used to calculate the similarity between $\mathbf{A}$ and its approximation $\mathbf{B}$, and additional constraints on $\mathbf{P}$ and $\mathbf{Q}$ that depend on a problem being solved:

$$\rho(\mathbf{A}, \mathbf{B}) \rightarrow \min_{\mathbf{P} \in \mathbb{R}^{n \times d}, \mathbf{Q} \in \mathbb{R}^{m \times d}} \tag{3}$$

The choice of a particular measure ρ significantly influences the result of matrix low-rank approximation process. Let us overview the most popular types of these measures.

The most popular way is to use the Frobenius norm to measure the distance:

$$\| \mathbf{A} - \mathbf{B} \|_F^2 = \sum_{i,j} (a_{i,j} - b_{i,j})^2. \tag{4}$$

In some cases, when the original matrix $\mathbf{A}$ contains probability values that describe a probability distribution, it becomes more natural to use the generalized Kullback-Leibler divergence or I-divergence [21]:

$$D_{\mathrm{KL}}(\mathbf{A} \parallel \mathbf{B}) = \sum_{i,j} (a_{i,j} \cdot \log \frac{a_{i,j}}{b_{i,j}} - a_{i,j} + b_{i,j}), \tag{5}$$

which represents a distance between two probability distributions. These measures are the particular cases of so called β-divergence [21]:

$$D^{\beta}_{\mathrm{Beta}}(\mathbf{A} \parallel \mathbf{B}) = \sum_{i,j} (a_{i,j} \cdot \frac{a_{i,j}^{\beta-1} - b_{i,j}^{\beta-1}}{\beta - 1} - \frac{a_{i,j}^{\beta} - b_{i,j}^{\beta}}{\beta}), \tag{6}$$

where β is a hyperparameter of the metric. Indeed, here are the cases when $\beta \to 1$:

$$D^{\beta}_{\mathrm{Beta}}(\mathbf{A} \parallel \mathbf{B}) \to D_{\mathrm{KL}}(\mathbf{A} \parallel \mathbf{B}), \tag{7}$$

and when $\beta = 2$:

$$D^{\beta}_{\mathrm{Beta}}(\mathbf{A} \parallel \mathbf{B}) = \frac{1}{2} \parallel \mathbf{A} - \mathbf{B} \parallel_F^2 . \tag{8}$$

The optimization problem (3) can be solved using gradient descent based optimization approaches that optimize all elements of $\mathbf{P}$ and $\mathbf{Q}$ independently or using alternating approaches that iteratively optimize factor $\mathbf{P}$ having factor $\mathbf{Q}$ fixed and conversely. The details about the optimization using β-divergence could in found in [28]. Computational and algorithmic details related to sparse nonnegative orthogonal matrix approximations can be found in [8].

Note that in case of the optimization problem (3), the factorization $\mathbf{P}\mathbf{Q}^{\top}$ is not

unique. Indeed, for every square non-singular matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ we have

$$\mathbf{P}\mathbf{Q}^\top = \mathbf{P}\mathbf{S}\mathbf{S}^{-1}\mathbf{Q}^\top = (\mathbf{P}\mathbf{S})(\mathbf{S}^{-1}\mathbf{Q}^\top) = \tilde{\mathbf{P}}\tilde{\mathbf{Q}}^\top, \tag{9}$$

where matrices $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{Q}}$ represent another factorization with the same value of ρ.

2.1.3 Singular Value Decomposition

Let us explore the following approximation of matrix $\mathbf{A}$ with three factors:

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{A}', \tag{10}$$

where columns of $\mathbf{U} \in \mathbb{R}^{n \times d}$ are dominant orthonormal eigenvectors of $\mathbf{A}\mathbf{A}^\top$, rows of $\mathbf{V}^\top \in \mathbb{R}^{d \times m}$ are dominant orthonormal eigenvectors of $\mathbf{A}^\top\mathbf{A}$, and matrix $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is diagonal and contains d maximal singular values σ_i of $\mathbf{A}$ in the decreasing order on the diagonal. This factorization is called truncated Singular Value Decomposition (SVD).

Denoting $\mathbf{P} = \mathbf{U}\sqrt{\mathbf{\Sigma}}$ and $\mathbf{Q} = \mathbf{V}\sqrt{\mathbf{\Sigma}}$, we get the familiar low-rank approximation of matrix $\mathbf{A}$:

$$\mathbf{A} \approx \mathbf{P}\mathbf{Q}^\top. \tag{11}$$

According to the Eckart-Young theorem [36], this solution is the global optimum of the optimization problem (3) with Frobenuis norm of matrices difference as a distance measure ρ:

$$||\mathbf{A} - \mathbf{P}\mathbf{Q}^\top||_2 \to \min_{\mathbf{P},\mathbf{Q}}. \tag{12}$$

SVD can be efficiently performed using the block power numerical method [13], which consists of a series of QR decompositions. The computational complexity of the

SVD algorithm strongly depends of the properties of the matrix $\mathbf{A}$. Also, several SVD algorithms are more efficient in case of sparse and thin-and-toll matrices [67].

2.1.4 Riemannian Optimization

Problem Setting. The Riemannian optimization framework is a powerful tool to solve low-rank optimization problems. It allows to solve the following maximization problem:

$$\begin{aligned} &\text{maximize} \quad F(\mathbf{X}), \\ &\text{subject to} \quad \mathbf{X} \in \mathcal{M}_d, \end{aligned} \tag{13}$$

where $\mathcal{M}_d$ is the manifold (see Section 1.1 in [113] for details) of all matrices in $\mathbb{R}^{n\times m}$ with rank d:

$$\mathcal{M}_d = \{\mathbf{X} \in \mathbb{R}^{n\times m} : \text{rank}(\mathbf{X}) = d\}. \tag{14}$$

An introduction to optimization over Riemannian manifolds can be found in [112].

In case of minimizing the distance ρ between matrices A and B, the problem could be rewritten in the following form:

$$\begin{aligned} &\text{minimize} \quad \rho(\mathbf{A}, \mathbf{B}), \\ &\text{subject to} \quad \mathbf{B} \in \mathcal{M}_d. \end{aligned} \tag{15}$$

The key difference between the problems (15) and (3) is the non-requirement of explicit search for factors matrices $\mathbf{P}$ and $\mathbf{Q}$. As we present in Chapter 5, we developed the novel approach without explicit search for these factors which provides a significant performance boost in some applications. Moreover, Riemannian optimization can

be successfully applied to various data science problems: for example, matrix completion [113], large-scale recommender systems [110], and tensor completion [62].

Riemannian Optimization Framework. Assume we have an approximated solution $\mathbf{X}_i$ on a current step of the optimization process of the problem given by (13), where i is the iteration step number. In order to improve $\mathbf{X}_i$, the next step of the standard gradient ascent outputs the point

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \nabla F(\mathbf{X}_i), \tag{16}$$

where $\nabla F(\mathbf{X}_i)$ is the gradient of objective F at the point $\mathbf{X}_i$. Note that the gradient $\nabla F(\mathbf{X}_i)$ can be naturally considered as a matrix in $\mathbb{R}^{n\times m}$. Point $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ leaves the manifold $\mathcal{M}_d$, because its rank is generally greater than d. That is why Riemannian optimization methods need to map point $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ back to manifold $\mathcal{M}_d$. The simplest Riemannian gradient method first projects the gradient step onto the tangent space at the current point $\mathbf{X}_i$ and then *retracts* it back to the manifold:

$$\mathbf{X}_{i+1} = R\left(P_{\mathcal{T}_M}(\mathbf{X}_i + \nabla F(\mathbf{X}_i))\right), \tag{17}$$

where R is the *retraction* operator, and $P_{\mathcal{T}_M}$ is the projection onto the tangent space.

Although the optimization problem is non-convex, Riemannian optimization methods show good performance on it. The overview of retractions of high-rank matrices to low-rank manifolds is provided in [1].

Projector-Splitting Algorithm. In this thesis, we use a simplified version of such approach that retracts point $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ directly to the manifold and does not require

projection onto the tangent space $P_{\mathcal{T}_M}$ as illustrated in Figure 3:

$$\mathbf{X}_{i+1} = R(\mathbf{X}_i + \nabla F(\mathbf{X}_i)). \tag{18}$$

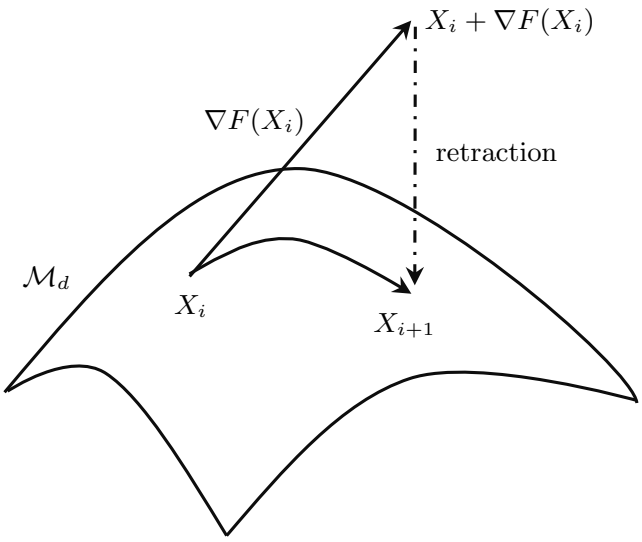

Figure 3: Geometric interpretation of one step of projector-splitting optimization procedure: the gradient step and the retraction of the high-rank matrix $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ to the manifold of low-rank matrices $\mathcal{M}_d$.

Intuitively, retractor R finds a rank-d matrix on the manifold $\mathcal{M}_d$ that is similar to high-rank matrix $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ in terms of Frobenius norm. How can we do it? The most straightforward way to reduce the rank of $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ is to perform the SVD by keeping d largest singular values of it:

$$\begin{aligned} &1\text{: } \mathbf{U}_{i+1}, \mathbf{\Sigma}_{i+1}, \mathbf{V}_{i+1}^\top \leftarrow \text{SVD}(\mathbf{X}_i + \nabla F(\mathbf{X}_i)), \\ &2\text{: } \mathbf{X}_{i+1} \leftarrow \mathbf{U}_{i+1}\mathbf{\Sigma}_{i+1}\mathbf{V}_{i+1}^\top. \end{aligned} \tag{19}$$

However, such approach is computationally expensive.

Therefore, we propose to apply the projector-splitting method proposed in [73], which is a second-order retraction onto the manifold. The projector-splitting algorithm also was mentioned in [1] as "Lie-Trotter retraction". The theoretical properties and convergence guarantees of such method are discussed in [57].

Its practical implementation is quite intuitive: instead of computing the full SVD of $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ according to the gradient projection method, we use just one step of the block power numerical method [13] which computes the SVD in order to reduce the computational complexity.

Let us present the current point in the following factorized form:

$$\mathbf{X}_i = \mathbf{U}_i \mathbf{S}_i \mathbf{V}_i^\top, \tag{20}$$

where matrices $\mathbf{U}_i \in \mathbb{R}^{n \times d}$ and $\mathbf{V}_i \in \mathbb{R}^{m \times d}$ have d orthogonal columns and $\mathbf{S}_i \in \mathbb{R}^{d \times d}$ is upper triangle matrix. Then we need to perform two QR-decompositions to retract point $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ back to the manifold to point $\mathbf{X}_{i+1} = \mathbf{U}_{i+1}\mathbf{S}_{i+1}\mathbf{V}_{i+1}^\top$ as follows:

$$\begin{aligned}
&1\text{: } \mathbf{U}_{i+1}, \mathbf{S}_{i+1} \leftarrow \text{QR}\left((\mathbf{X}_i + \nabla F(\mathbf{X}_i))\mathbf{V}_i\right), \\
&2\text{: } \mathbf{V}_{i+1}, \mathbf{S}_{i+1}^\top \leftarrow \text{QR}\left((\mathbf{X}_i + \nabla F(\mathbf{X}_i))^\top \mathbf{U}_{i+1}\right), \\
&3\text{: } \mathbf{X}_{i+1} \leftarrow \mathbf{U}_{i+1}\mathbf{S}_{i+1}\mathbf{V}_{i+1}^\top.
\end{aligned}$$

Using this trick, we always keep the solution $\mathbf{X}_{i+1} = \mathbf{U}_{i+1}\mathbf{S}_{i+1}\mathbf{V}_{i+1}^\top$ on the manifold $\mathcal{M}_d$.

What is important, we only need to compute $\nabla F(\mathbf{X}_i)$, so the gradients with respect to $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$ do not need to be calculated explicitly. Due to this we avoid the subtle case where $\mathbf{S}$ is close to singular (so-called singular (critical) point on the manifold).

Indeed, the gradient with respect to $\mathbf{U}$ (while keeping the orthogonality constraints) can be written [56] as:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{U}} = \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \mathbf{V} \mathbf{S}^{-1}, \tag{21}$$

which means that the gradient will be large if $\mathbf{S}$ is close to singular. The proposed projector-splitting scheme is free from this problem.

2.1.5 Pseudo-Skeleton Factorization

Pseudo-Skeleton Factorization. In some practical applications, it is needed to decompose matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ into factors that consist of original elements of $\mathbf{A}$. Let us have a look at the following decomposition:

$$\mathbf{A} \approx \mathbf{P} \hat{\mathbf{A}}^{-1} \mathbf{H}, \tag{22}$$

where $\mathbf{P}$ consists of a subset of columns of matrix $\mathbf{A}$, $\mathbf{H}$ consists of rows of $\mathbf{A}$ and $\hat{\mathbf{A}}$ is the intersection of columns $\mathbf{P}$ and rows $\mathbf{R}$. This low-rank factorization is called pseudo-skeleton [39] and can effectively approximate initial matrix $\mathbf{A}$ in various problems. In this work, we use the following simplification of the approximation above:

$$\mathbf{A} \approx \mathbf{A}[:, \mathbf{k}] \mathbf{C}, \tag{23}$$

where $\mathbf{k} \in \mathbb{N}^d$ represents a set of column indices of $\mathbf{A}$, $\mathbf{A}[:, \mathbf{k}]$ consists of columns of matrix $\mathbf{A}$ with indices $\mathbf{k}$ and $\mathbf{C} \in \mathbb{R}^{d \times m}$ is the matrix of coefficients. This work uses the Numpy Python indexing notation[1]: $\mathbf{A}[\mathbf{k}, :]$ is the matrix whose column j coincides

[1] https://docs.scipy.org/doc/numpy-1.13.0/reference/arrays.indexing.html

with the row k_j of $\mathbf{A}$, where k_j is the j-th component of vector $\mathbf{k}$.

The following sections describe the simple algorithm to perform such decomposition.

Maximal Volume Concept. The maximal-volume concept [38] provides an approach for a matrix approximation in a pseudo-skeleton form, which is a product of matrices formed by columns or rows of the source matrix.

The algorithm, called Maxvol [37], allows to efficiently find a well-conditioned submatrix with a high enough volume for building such an approximation. Maximal volume submatrices are useful for various applications, such as wireless communications [117], preconditioning of overdetermined systems [7], tensor decompositions [83], and recommender systems [72]. The generalization of the maximal-volume concept to the rectangular case presented in this work offers additional degrees of freedom, which is potentially useful in any of these areas.

Suppose we want to select d columns with indices $\mathbf{k} \in \mathbb{N}^d$. First of all, the algorithm requires to compute the rank-d SVD factorization of $\mathbf{A}$ given by Equation (11). After this, searching for a column set is equivalent to searching for a square submatrix

$$\mathbf{S} = \mathbf{Q}[\mathbf{k}, :]^\top \in \mathbb{R}^{d \times d} \tag{24}$$

in the factor matrix $\mathbf{Q}$. As it was shown in [37], it is an efficient (from the computational time point of view) approach to search for such submatrix $\mathbf{S}$ that has a big volume (the modulus of a determinant).

Overall, we have the following optimization task:

$$\mathbf{k} \leftarrow \underset{\mathbf{k}}{\operatorname{argmax}} \operatorname{Vol} \mathbf{S} = \underset{\mathbf{k}}{\operatorname{argmax}} |\det \mathbf{S}|, \quad \mathbf{S} = \mathbf{Q}[\mathbf{k}, :]. \tag{25}$$

The problem is NP-hard in the general case [22] and, therefore, suboptimal greedy procedures are usually applied.

Maximal Volume Algorithm. One of the most popular greedy procedures is called Maxvol algorithm [37] and is based on searching for a *dominant* submatrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ of $\mathbf{Q}^\top$. The dominant property of $\mathbf{S}$ means that all columns $q_i \in \mathbb{R}^d$ of $\mathbf{Q}^\top$ can be represented via a linear combination of columns from $\mathbf{S}$ with the coefficients not greater than 1 in modulus. Although this property does not imply that $\mathbf{S}$ has the maximal volume, it guarantees that $\mathbf{S}$ is *locally optimal*, which means that replacing any column of $\mathbf{S}$ with a column of $\mathbf{Q}^\top$, does not increase the volume [37].

At the initialization step, Maxvol usually takes d linearly independent latent vectors that are the pivots from LU-decomposition [36] of matrix $\mathbf{Q}^\top$. Practice shows that this initialization provides a good initial approximation $\mathbf{S}$ to maximal volume matrix [37]. After that, the algorithm iteratively performs the following procedure. At each step, it computes the coefficient matrix $\mathbf{C} = \mathbf{S}^{-1}\mathbf{Q}^\top \in \mathbb{R}^{d \times m}$. Each column of $\mathbf{C}$ contains the coefficients of the representation of a row in $\mathbf{Q}$ via the vectors from $\mathbf{S}$. Thus, i-th entry in j-th column of $\mathbf{C}$ that is larger than 1 in modulus provides us with a hypothesis that j-th latent vector is "larger" than i-th. Therefore, the algorithm seeks for the maximal in modulus coefficient in order to swap a "smaller" latent vector inside the seed set with a "larger" one out of it. The procedure repeats until convergence. In the work, we also call this algorithm *Square Maxvol*, because it seeks for a square submatrix (since

determinant is defined only for square $\mathbf{S}$). Furthermore, it is important to note that the original algorithm presented in [37] has crucial speed optimizations for avoiding the expensive matrix multiplications and inversions, which are not presented in the work. We recommend to go to the original literature to for a more rigorous explanation of the Maxvol algorithm.

Algorithm 1 presents a high-level pseudo-code of the Maxvol based approach.

Algorithm 1 Searching good columns using Maxvol

Require: Matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, number of columns d
Ensure: Indices $\mathbf{k} \in \mathbb{N}^d$ of d representative items
1: Perform rank-d SVD of the matrix $\mathbf{A} \approx \mathbf{P}\mathbf{Q}^\top$
2: $\mathbf{k} \leftarrow d$ pivot indices from LU-decomposition of $\mathbf{Q}$
3: **repeat**
4: $\quad \mathbf{S} \leftarrow \mathbf{Q}[\mathbf{k}, :]^\top$
5: $\quad$ Find coefficients for representing row of $\mathbf{Q}$ via the seed set: $\mathbf{C} \leftarrow \mathbf{S}^{-1}\mathbf{Q}^\top$
6: $\quad$ Find the maximal coefficient in $\mathbf{C}$: $(i, j) \leftarrow \operatorname{argmax}_{i,j \notin \mathbf{k}} |c_{ij}|$
7: $\quad$ Swap i-th element from the seed set and j-th column out of it: $k_i \leftarrow j$
8: **until** $\forall (i, j) : |c_{ij}| \leq 1$
9: **return** $\mathbf{k}$

Computational Complexity of Maxvol. Let us analyze the computational complexity of Maxvol. The LU-decomposition with pivoting takes $O(md^2)$ operations. The iterative updates take $O(\alpha md)$ operations, where α is the number of iterations. Typically, $\alpha \leq d$ iterations are needed. The overall complexity of Square Maxvol can be estimated as $O(md^2)$. A more detailed complexity analysis of Square Maxvol is given in [37].

2.2 Supervised Learning Background

2.2.1 Introduction

Supervised learning methods form a foundation of a lot of embedding methods, so further in this section, we overview most widely used existing supervised learning approaches.

Let us have a dataset $\{\mathbf{x}_i\}_{i=1}^n$, where every sample (also called an object) $\mathbf{x}_i = (x_{i,1}, ..., x_{i,m}) \in \mathcal{X}$, where $\mathcal{X}$ is the set of all possible objects. The element $x_{i,j}$ is called the j-th feature of the object $\mathbf{x}_i$ [30]. The whole dataset can be represented as a matrix $\mathbf{X}$ with size $n \times m$. Unsupervised problems involve finding regularities within this data. In supervised learning problems every element $\mathbf{x}_i$ of the training set has an object label $y_i \in \mathcal{Y}$, and the goal is to approximate $\mathcal{X} \to \mathcal{Y}$ function.

There are several popular choices of $\mathcal{Y}$. If $\mathcal{Y} = \mathbb{R}$ then the problem is called *regression* problem. If $\mathcal{Y} = \{0, 1, ..., K-1\}$ the problem is called *classification* problem. The most widely used and important case of classification problems is binary classification problem when $K = 2$. Cases of $K > 2$ could be solved using classifiers for the binary classification case, so further we overview only the binary classification case.

2.2.2 Decision Trees Based Methods

Decision tree based methods [86] often show a good performance in problems where the number of features m is not large compared to the number of objects n. These methods work exceptionally well in problems where features have different nature, which happens in such applications as web search, credit scoring, churn prediction, etc. The decision-based approaches can handle various types of supervised learning problems, including classification, regression, and others.

Although single decision trees [17] usually do not show a good performance, ensembled decision trees often provide state-of-the-art results. The most popular ensembling approaches are Random Forest [16], which learns a set of independent decision trees using bagging [15] and Random Supspace Method [9] techniques, and Gradient Boosted Decision Trees [31, 32], which learns decision trees one by one and each new decision tree training focuses on compensating the error of the all previous trees combination.

2.2.3 Linear Methods

Historically, the most popular methods to solve supervised learning problems are linear methods [30], which decision function is based on the scalar product $\langle \mathbf{w}, \mathbf{x}_i \rangle$ of an object's real-valued feature vector $\mathbf{x}_i = (x_{i,1}, ..., x_{i,m})$ and a vector of weights $\mathbf{w} = (w_1, ..., w_m)$ that are parameters of the algorithm.

In case of regression problems, a predicted label $\tilde{y}_i$ of i-th object is set to this scalar product and the weights w are usually found based on the following optimization problem:

$$||\mathbf{X}\mathbf{w} - \mathbf{y}||_2 \to \min_{\mathbf{w} \in \mathbb{R}^m}. \tag{26}$$

In case of a binary classification, a predicted probability $\tilde{y}_i$ of i-th object to belong to the class 1 is calculated in the following way:

$$p(y_i = 1 | \mathbf{x}_i) = \sigma\left(\langle \mathbf{w}, \mathbf{x}_i \rangle\right) = \frac{1}{1 + \exp\left(-\langle \mathbf{w}, \mathbf{x}_i \rangle\right)},$$

and the weights are tuned by the optimization

$$\sum_{i=1}^{n} \mathcal{L}(y_i, \sigma(\langle \mathbf{w}, \mathbf{x}_i \rangle)) \to \min_{\mathbf{w} \in \mathbb{R}^m},$$

where $\mathcal{L}$ is the negative logarithm of the likelihood (the so-called log-loss) to observe the training set's target labels in accordance to the predicted probabilities $\tilde{\mathbf{y}}$:

$$\mathcal{L}(y_i, \tilde{y}_i) = -(y_i \log \tilde{y}_i + (1 - y_i) \log(1 - \tilde{y}_i)).$$

2.2.4 Neural Networks

Neural networks [30, 120] naturally generalize the linear supervised learning approach to the non-linear case. A neural network decision-making process consists of several steps (layers), and each of these steps takes outputs (neurons) from the previous step, applies a non-linear transformation to them and multiplies them by some weight matrix in order to generate outputs. The first step takes object's features as an input, and the last step's output is interpreted as a prediction of the neural network.

Usually, neural network training procedure finds optimal weight matrices in accordance to the task being solved. The most popular optimization approach is based on gradient descent procedure and is called backpropagation [30, 20]. The number of layers, the number of neurons in each of them, non-linear transformation functions are hyperparameters of the algorithm.

Nowadays, neural networks are successfully applied in various applications where the data has some spatial structure — images, time series, texts, audio, etc. The main reason for this is inventions of particular types of neural networks that successfully handle such data [12, 63, 68, 75, 119].

2.3 Background on Categorical Features in Machine Learning

2.3.1 Categorical Features

Most of existing machine learning methods suppose that the features $\{x_{i,j}\}_{j=1}^{m}$ of object $\mathbf{x}_i$ are in $\mathbb{R}$. But there are many problems, where the feature values come from a finite unordered set, not from $\mathbb{R}$. These features are called categorical, nominal or factoral. For example, the categorical feature *City* may have values from set

$$\{ \quad \textit{Moscow}, \quad \textit{New York}, \quad \textit{Paris}, \quad \ldots \quad \} \tag{27}$$

in some tasks. Such tasks become even more difficult with the increase of the size q of this set, because of huge data sparsity [30]. There are very few methods that are directly suitable for categorical data analysis, e.g., naive Bayes based methods for supervised problems. It means that many widely used and very powerful real-valued techniques cannot be efficiently applied to these tasks. That is why Section 4 aims at finding a method of meaningful transformation of categorical features' values into real-valued vectors in order to avoid this issue.

2.3.2 Examples of Problems Involving Categorical Features

There are a lot of tasks where objects are described using categorical features. We point out some examples in this section.

Collaborative Filtering. Collaborative filtering is one of the most popular examples [95]. Each object $\mathbf{x}_i$ in the training set is a corresponding user rating description. In the simplest case, each object has two categorical features *User* and *Item* with target

label y_i which represents the rating of *Item* set by *User*. The task is to predict rating scores for objects unseen before. See Section 2.5 for details.

Web Search. A lot of existing information retrieval techniques can be improved using the categorical features. For example, the learning to rank task can be generalized to the personalized learning to rank just by adding a single categorical feature *User* to the features vectors that describe *Query-Document* pairs [123].

Natural Language Modelling. Categorical features are also applied in the natural language processing [50]. For example, the categorical feature *Word* appears in tasks such as the highly demanded language modelling problem, which requires an estimation of probability $p(w_{t-n+1}, ..., w_t)$, where w_i is the word in i-th position of a sequence of words with length n, which, in turn, is the size of the language model. This task can be viewed as the task with n categorical features [12], and such approach shows high performance in applications.

2.3.3 Existing Approaches to Handling Categorical Features

Naive Encoding. This work proposes a method that transforms a categorical data matrix $\mathbf{X}$ with size $n \times m$ (each column is categorical) into a real matrix $\mathbf{Z} \in \mathbb{R}^{n \times m'}$, where m' is a number of features after the transformation. The real-valued matrix $\mathbf{Z}$ can be further used with classical machine learning methods.

Let us enumerate categorical feature's values. There is an example for feature *City*:

$$\begin{aligned} \textit{Moscow} &\rightarrow 1, \\ \textit{New York} &\rightarrow 2, \\ \textit{Paris} &\rightarrow 3, \\ \textit{London} &\rightarrow 4, \\ \ldots &\rightarrow \ldots . \end{aligned}$$

So we use $\tilde{\mathbf{X}} \in \mathbb{N}^{n \times m}$ instead of original $\mathbf{X}$.

Indeed, the categorical feature's values set is unordered, so such simple transformation of categorical features cannot be effectively used with most real-valued machine learning techniques, because a non-existing order of feature values is significantly considered this way. E.g., there is no sense in the inequality *New York* $<$ *London*, but most real-valued machine learning algorithms would take into an account this wrong knowledge.

Decision Tree Based Methods. Decision Trees [17], Random Forests [16] and other Decision Tree based methods can handle categorical features out-of-the-box. They work in the following way. The training process tries to find the optimal split of the dataset into two parts in each node. The best split is the one that increases the purity of the subsets. But it is very computationally expensive to find the optimal split in case of categorical features with a large number of unique values. Actually there are $O(2^{q_j})$ different splits for the j-th feature with q_j distinct values. So even if $q_j = 100$, it becomes impossible to find the optimal split because of computational complexity.

One-Hot Encoding. A very popular method of transformation of categorical matrix X into real-valued matrix $\mathbf{Z}$ is the so-called one-hot encoding [82], also known as Dummy Encoding. Let some j-th feature has q_j unique values $\{a_1, ..., a_{q_j}\}$. Then this feature is expanded into q_j new binary features in this way:

$$z_{i,(j,a_k)} = \mathbf{1}[x_{i,j} = a_k], \quad \mathbf{X} \in \mathbb{N}^{n \times m}, \quad \mathbf{Z} \in \mathbb{R}^{n \times m'}, \quad k \in \{1, ..., q_j\}, \tag{28}$$

where $m' = \sum_{j=1}^{m} q_j$. Here $\mathbf{1}[a]$ is an indicator of a, i.e.

$$\mathbf{1}[a] = \begin{cases} 1, & \text{if } a \text{ is true,} \\ 0, & \text{else.} \end{cases}$$

We get new transformed matrix $\mathbf{Z}$ by applying this procedure for every feature of $\mathbf{X}$.

Although one-hot transformation is natural and easy to use, it has many drawbacks. The major one is a high increase of object space dimensionality m'. Because most of the binary feature values are zeros, it's especially necessary to use sparse data representation if m' is large. Only a few methods can efficiently work with such representations, e.g., sparse linear methods. Thus, using one-hot encoding leads to significant limitations in choosing further tools of data analysis.

There are many other categorical encoding approaches that are very similar to the one-hot encoding, e.g., dummy encoding, effects encoding and other [45]. They are primarily designed for further using with linear methods. All these approaches have the same drawbacks as the previously discussed one-hot encoding.

Naive Bayes. To solve the problem of binary classification with categorical features, a naive multinomial Bayesian classifier can be used. Its main idea is the assumption of probabilistic conditional independence of features. In case of classification, the following class will be predicted:

$$\underset{y_i}{\operatorname{argmax}}\, p(y_i|\mathbf{x}_i) = \underset{y_i}{\operatorname{argmax}}\, p(y_i) \prod_{j=1}^{m} p(x_{i,j}|y_i). \tag{29}$$

Since all the multipliers are statistics over discrete and unordered sets (categorical features' values) then the principle of maximum likelihood provides the following estimation of such multiplier:

$$p(x_{i,j}|y_i) = \frac{\sum_{k=1}^{n} \mathbf{1}[y_i = y_k] \cdot \mathbf{1}[x_{i,j} = x_{k,j}]}{\sum_{k=1}^{n} \mathbf{1}[y_k = y_i]}. \tag{30}$$

In this situation, the maximum likelihood principle may have several disadvantages. For example, if some categorical value of j was not found in the training sample, the algorithm will not be able to calculate $p(x_{i,j}|y_i)$. In addition, this approach does not take into account the variance of the maximum likelihood estimate. In order to avoid this in practice, an additive smoothing of probabilities [50] is used. It imposes a priori Dirichlet distribution on the parameters of the probabilistic model:

$$p(x_{i,j}|y_i) = \frac{\sum_{k=1}^{n} (\mathbf{1}[y_i = y_k] \cdot \mathbf{1}[x_{i,j} = x_{k,j}] + \alpha)}{\sum_{k=1}^{n} (\mathbf{1}[y_k = y_i] + \alpha)}, \tag{31}$$

where parameter α is a smoothing parameter. The greater is the value α, the closer the probability estimates are to the prior probabilities of each class. And conversely, the smaller is α, the closer the probability estimations are to the maximum likelihood

estimations.

Factorization Machines. Factorization Machines (FMs) [90] is the state-of-the-art technique for a modeling recommender data. FMs use polynomial approximation methods and the one-hot encoding. The advantages of FMs are in their scalability and the fact, that they can mimic the most successful approaches for the task of collaborative filtering, including SVD++ [58], PITF [93] and FPMC [92]. Note that FMs is a complete supervised prediction model that can handle categorial features, but it is not a categorical feature transformation method.

2.4 Word Embeddings

A short overview of the embeddings methods and their applications was presented in Chapter 1. Two further sections focus on two specific cases: word embeddings and embeddings of users and items in recommender systems.

2.4.1 Overview of Word Embeddings Methods

Word embedding training methods became an essential part of the natural language processing field after the publication of [76], which described the implementation of popular "word2vec" software and gave a significant impact for the whole industry. This work introduced the Skip-Gram and Skip-Gram Negative Sampling embedding models, which could be implemented efficiently and can process big amounts of data. After this, other popular approaches were published, such as GloVe [85]. Moreover, non-negative matrix factorizations are also used to build embeddings [79].

Apart the word-level embeddings, a lot of subword-level embeddings that could analyze the meaning of words by analyzing their parts were introduced, e.g., [14, 19,

41, 46, 101, 114, 121].

In this work, we focus on the classical and one of the most popular approaches — Skip-Gram Negative Sampling.

2.4.2 Skip-Gram Negative Sampling

Skip-Gram Negative Sampling (SGNS) approach to train word embeddings was introduced in [76]. The "negative sampling" approach is thoroughly described in [35], and the learning method is explained in [96]. There are several open-source implementations of SGNS neural network, which is widely known as "word2vec". [1][2]

Assume we have a text corpus given as a sequence of words $w_1, \dots, w_n$, where n may be larger than 10^{12} and w_i belongs to a vocabulary of words V_W. A *context* $c \in V_C$ of the word w_i is a word from set $\{w_{i-L}, ..., w_{i-1}, w_{i+1}, ..., w_{i+L}\}$ for some fixed window size L. Usually, V_C and V_W are equal, however in some application they are different (e.g. rare words or contexts are replaced with a special token). Let $\mathbf{w}, \mathbf{c} \in \mathbb{R}^d$ be the *word embeddings* of word w and context c, respectively. Assume they are specified by the following mappings:

$$\mathcal{W} : V_W \to \mathbb{R}^d, \quad \mathcal{C} : V_C \to \mathbb{R}^d, \tag{32}$$

where d is the dimensionality of the embedding space. The ultimate goal of SGNS word embedding training is to fit good mappings $\mathcal{W}$ and $\mathcal{C}$ or, in other words, to find good embeddings $\mathbf{w}$ and $\mathbf{c}$ for each word w and context c. Important to note that in a general case, vocabulary of contexts can differ from the vocabulary of words, e.g. we might

[1] Original Google word2vec: https://code.google.com/archive/p/word2vec/
[2] Gensim word2vec: https://radimrehurek.com/gensim/models/word2vec.html

want to consider nouns only. However, in the majority of cases, and in the SGNS case in particular, these two sets are just equal.

Let D be a multiset of all word-context pairs observed in the corpus. In the SGNS model, the probability that word-context pair (w, c) is observed in the corpus is modelled as a following distribution:

$$\begin{aligned} P\left(\#(w,c) \neq 0 | w, c\right) = \\ = \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) = \frac{1}{1 + \exp(-\langle \mathbf{w}, \mathbf{c} \rangle)}, \end{aligned} \tag{33}$$

where $\#(w, c)$ is the number of times the pair (w, c) appears in D and $\langle \mathbf{x}, \mathbf{y} \rangle$ is the scalar product of vectors $\mathbf{x}$ and $\mathbf{y}$. Number d is a hyperparameter that adjusts the flexibility of the model. It usually takes values from tens to hundreds.

In order to collect a training set, we take all pairs (w, c) from D as positive examples and k randomly generated pairs (w, c) as negative ones. The number of times the word w and the context c appear in D can be computed as

$$\#(w) = \sum_{c \in V_c} \#(w, c), \tag{34}$$

$$\#(c) = \sum_{w \in V_w} \#(w, c) \tag{35}$$

accordingly. Then negative examples are generated from the distribution defined by $\#(c)$ counters:

$$P_D(c) = \frac{\#(c)}{|D|}. \tag{36}$$

In this way, we have a model maximizing the following logarithmic likelihood objective

for all word-context pairs (w, c):

$$\begin{aligned} l_{wc}(\mathbf{w}, \mathbf{c}) =& \#(w, c)(\log \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) + \\ &+ k \cdot \mathbb{E}_{c' \sim P_D} \log \sigma(-\langle \mathbf{w}, \mathbf{c}' \rangle)), \end{aligned} \tag{37}$$

where k is a hyperparameter that controls the weight of the negative class in the model. In order to maximize the objective over all observations for each pair (w, c), we arrive at the following SGNS optimization problem over all possible mappings $\mathcal{W}$ and $\mathcal{C}$:

$$\begin{aligned} l = \sum_{w \in V_W} \sum_{c \in V_C} (\#(w, c)(\log \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) + \\ + k \cdot \mathbb{E}_{c' \sim P_D} \log \sigma(-\langle \mathbf{w}, \mathbf{c}' \rangle))) \to \max_{\mathcal{W}, \mathcal{C}}. \end{aligned} \tag{38}$$

Usually, this optimization is done via the stochastic gradient descent procedure that is performed during passing through the corpus [76, 96].

2.4.3 SGNS Optimization as Matrix Factorization

As shown in [70], the logarithmic likelihood (38) can be represented as the sum of $\tilde{l}_{w,c}(\mathbf{w}, \mathbf{c})$ over all pairs (w, c), where $\tilde{l}_{w,c}(\mathbf{w}, \mathbf{c})$ has the following form:

$$\begin{aligned} \tilde{l}_{w,c}(\mathbf{w}, \mathbf{c}) =& \#(w, c) \log \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) + \\ &+ k \frac{\#(w)\#(c)}{|D|} \log \sigma(-\langle \mathbf{w}, \mathbf{c} \rangle). \end{aligned} \tag{39}$$

A crucial observation is that this loss function depends only on the scalar product $\langle \mathbf{w}, \mathbf{c} \rangle$ but not on embeddings $\mathbf{w}$ and $\mathbf{c}$ separately:

$$\tilde{l}_{w,c}(\mathbf{w}, \mathbf{c}) = f_{w,c}(x_{w,c}), \tag{40}$$

where

$$f_{w,c}(x_{w,c}) = a_{w,c} \log \sigma(x_{w,c}) + b_{w,c} \log \sigma(-x_{w,c}), \tag{41}$$

and $x_{w,c}$ is the scalar product $\langle \mathbf{w}, \mathbf{c} \rangle$, and

$$a_{w,c} = \#(w, c), \quad b_{w,c} = k \frac{\#(w)\#(c)}{|D|} \tag{42}$$

are constants. This means that (38) represents the low-rank factorization problem.

Indeed, denote the size of the words vocabulary $|V_W|$ as n and the size of the context words vocabulary $|V_C|$ as m. Let $\mathbf{W} \in \mathbb{R}^{n \times d}$ and $\mathbf{C} \in \mathbb{R}^{m \times d}$ be low-rank matrices, where each row $\mathbf{w} \in \mathbb{R}^d$ of matrix $\mathbf{W}$ is the word embedding of the corresponding word w and each row $\mathbf{c} \in \mathbb{R}^d$ of matrix $\mathbf{C}$ is the context embedding of the corresponding context c. Then the optimization problem can be written in the following form:

$$\sum_{w \in V_W} \sum_{c \in V_C} \tilde{l}_{w,c}(\mathbf{w}, \mathbf{c}) \to \max_{\mathbf{W}, \mathbf{C}}. \tag{43}$$

2.4.4 SVD over SPPMI matrix

In order to be able to use out-of-the-box SVD for this factorization task, the authors of [70] proposed the surrogate version of SGNS as the objective function based on a Shifted Positive Pointwise Mutual Information (SPPMI) matrix. There are two general assumptions made in their algorithm that distinguish it from the SGNS optimization:

1. SVD optimizes Mean Squared Error (MSE) objective instead of SGNS loss function.

2. In order to avoid infinite elements in Shifted Pointwise Mutual Information (SPMI) matrix, which results from the SGNS optimizaton, it is transformed in the ad-hoc

manner (SPPMI matrix is used) before applying SVD.

This makes the objective not interpretable in terms of the original task (38). As mentioned in [70], SGNS objective weighs different (w, c) pairs differently, unlike the SVD, which works with the same weight for all pairs and may entail the performance fall. The comprehensive explanation of the relation between SGNS and SVD-SPPMI methods is provided in [53]. Papers [64, 71] give a good overview of highly practical methods to improve these word embedding models.

2.5 Embeddings in Recommender Systems

2.5.1 Collaborative Filtering

Collaborative Filtering (CF) [94] is one of the most widely used approaches to recommender systems. It is based on the analysis of users' previous activity (likes, watches, skips, etc. of items) and discovering hidden relations between users and items. Compared to other recommender system approaches, conventional Collaborative Filtering methods do not analyze any domain-specific context of users/items [98], such as explicit user and item profiles, items' text descriptions or social relations between users. Therefore, they are domain- and data-independent and can be applied to a wide range of tasks, which is their major advantage.

2.5.2 Implicit and Explicit Feedback

There are different types of user feedback [60]: explicit and implicit. Explicit feedback supposes every single user rates only a small set of items and ratings for other items are unknown. That means that the rating matrix is sparse. For example, explicit feedback may be convenient in cases when there are both positive and negative feedback, e.g.,

likes and dislikes. Implicit feedback implies that any user action can be viewed as feedback: e.g., skips, likes, purchases, views, playbacks, etc. [81, 49]. Actually, an absence of any actions related to a particular item can be considered as feedback as well, and therefore the rating matrix may be dense.

2.5.3 Latent Factor Models

Among CF methods, matrix factorization techniques [59, 60] offer the most competitive performance [25] and usually outperforms popular neighborhood methods [59]. Latent Factor Models map users and items into a latent factor space of d-dimensional embeddings which contains information about preferences of users w.r.t. items. The mapping is provided by a low-rank approximation of the rating matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ where one dimension represents n users, another dimension represents m items of interest and entries represent user feedback on items. An unknown relevance of a particular item that would be assigned by a particular user can be estimated by the scalar product of their embeddings (latent vectors).

The question is how to find a good factorization of a rating matrix. In case of explicit feedback, the most popular approach for sparse matrix factorization is Alternating Least Squares (ALS) method [11]. Its principal idea is based on componentwise optimization through factors that can be reduced to the least squares estimation. Sometimes, the absence of any actions can be viewed as negative feedback with a small weight and, in this case, the method for low-rank approximation is called Implicit Alternating Least Squares (IALS) [49].

2.5.4 PureSVD

A very popular and natural way to extract latent vectors from the data is to use classic truncated sparse Singular Value Decomposition (SVD) of the rating matrix, which is also called PureSVD in the recommender literature [25]:

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top, \quad \mathbf{P} \in \mathbb{R}^{n \times d}, \quad \mathbf{Q} \in \mathbb{R}^{m \times d}. \tag{44}$$

This approach assumes that all values in the rating matrix are known, but a lot of them are equal to zero (actually, all entries with unknown ratings). In terms of popular ranking measures, this approach performs better than other popular methods such as SVD++ [25].

This factorization can be interpreted as follows. Every user u has a low dimensional embedding $\mathbf{p}_u \in \mathbb{R}^d$, a row in the matrix $\mathbf{P}$, and every item i has an embedding $\mathbf{q}_i \in \mathbb{R}^d$, a row of the matrix $\mathbf{Q}$. As a result, PureSVD method provides an approximation $\tilde{r}_{ui}$ of the unknown rating for a pair (u, i), which is computed as the scalar product of the embeddings: $\tilde{r}_{ui} = \langle \mathbf{p}_u, \mathbf{q}_i \rangle$.

2.5.5 Cold-Start Problem

Since Collaborative Filtering approaches use only user behavioral data for predictions, but not any domain-specific context of users/items, they cannot generate recommendations for new *cold* users or *cold* items which have no ratings so far.

There are various approaches to cold start problem proposed in the literature. Additional context information (e.g., category labels [126] or all available metadata [10]) may be used. Moreover, there is a class of methods that use adaptive tree-based ques-

tionnaires to acquire the initial information about new users [34, 40, 51, 52, 108, 131]. Moreover, the cold start problem can be viewed from the exploration-exploitation trade-off point of view [3, 130]. The methods from [24, 55] analyze the performance of CF methods w.r.t. the number of known ratings for a user. This work, in turn, focuses on the most popular type of approaches called Rating Elicitation.

2.5.6 Cold Objects Embeddings with Rating Elicitation

A very common approach to solve this *cold-start problem* [99], called *rating elicitation*, is to explicitly independently ask cold users to rate a small representative *seed set* of items or to ask a representative *seed set* of users to rate a cold item [33, 34, 72]. These obtained ratings, in turn, could be used as an embedding of a cold user or a cold item.

The rating elicitation methods, such as [6, 33, 72, 87], are based on the same common scheme, which is introduced in this section. Suppose we have a system that contains a history of users' ratings for items, where only a few items may be rated by a particular user. Denote the rating matrix by $\mathbf{R} \in \mathbb{R}^{n \times m}$, where n is the number of users and m is the number of items, and the value of its entry r_{ui} describes the feedback of user u on item i. If the rating for pair (u, i) is unknown, then r_{ui} is set to 0. Without loss of generality and due to the space limit, the following description of the methods is provided only for the user cold start problem. Without any modifications, these methods for the user cold start problem can be used to solve the item cold start problem after the transposition of matrix R.

Algorithm 2 presents the general scheme of a rating elicitation method. Such procedures ask a cold user to rate a *seed set* of representative items with indices $\mathbf{k} \in \mathbb{N}^{L_0}$ for modeling his preference characteristics, where L_0, called *budget*, is a parameter of the

rating elicitation system.

Algorithm 2 Rating elicitation for user cold start problem

Require: Warm rating matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, cold user, budget L_0
Ensure: Predicted ratings of the cold user for all items
1: Compute indices $\mathbf{k} \in \mathbb{N}^{L_0}$ of representative items that form a seed set
2: Elicit ratings $\mathbf{z}' \in \mathbb{R}^{1 \times L_0}$ of the cold user on items with indices $\mathbf{k}$
3: Predict ratings of the cold user for all items $\mathbf{z} \in \mathbb{R}^{1 \times m}$ using $\mathbf{z}'$
4: **return** $\mathbf{z}$

The performance of a rating elicitation procedure and the performance of the obtained embeddings should be measured using a quality of predictions $\mathbf{z}$. For this purpose, we use ranking measures (such as Precision@k), which are well suitable for CF task (see Section 6.6 for details).

2.5.7 Rating Elicitation Methods

Scoring Methods. The simplest method to find a good seed set is to rank users or items by some ad-hoc score which shows how representative they are and take the top ranked entities as a seed set [87, 88, 125, 127]. An obvious drawback of such methods that is avoided in the approach is that these elements are taken from the seed set independently, and diversity of the selected elements is limited [33].

GreedyExtend. Among scoring methods, the most straightforward method is the GreedyExtend approach [33]. Unfortunately, the brute force manner of GreedyExtend implies very high computational costs. Hence, it is hardly scalable, in contrast to the approaches that are empirically compared in this work. This method greedily adds the item i to the current seed set of indices $\mathbf{k} \in \mathbb{N}^L$ that maximizes the target quality measure. The search of the best i is computed in a brute force manner, i.e. the algorithm iteratively adds the best item into the seed set: $\mathbf{k} \leftarrow [\mathbf{k}, i]$, where $i = \operatorname{argmin}_{i' \notin \mathbf{k}} \mathcal{F}([\mathbf{k}, i'])$ and

$\mathcal{F}([\mathbf{k}, i'])$ is the quality measure of recommendations generated using the seed set indices $[\mathbf{k}, i']$. The authors of this method reported the results only for an approach that uses similarities of items to predict the ratings via the seed set ratings. More effective [6] linear approach described in Section 6.2 costs $O(Lnm)$, where L is a current length of $\mathbf{k}$. At each step, the least squares solution is computed for almost all items, i.e., $O(m)$ times. Since the algorithm has L_0 such steps, the total complexity is $O(L_0^2 nm^2)$ (more than 10^{16} operations for the Netflix dataset and the seed set size $L_0 = 10$). Therefore, we do not use this method in the experiments.

Further, in this section, we overview the methods that aim at a selection of a diverse seed set and that have better performance.

Backward Greedy Selection. Another class of methods of searching for diverse representatives is based on the factorization of the rating matrix. Since the selection of user or item representatives is equivalent to selecting a submatrix of the corresponding factor, these algorithms seek for the submatrix that maximizes some criterion. One such approach, called Backward Greedy Selection [6], solves only the item cold start problem, but not the user one. This method is based on the techniques for transductive experimental design introduced in [122]. To get the seed set, it greedily removes users from a source user set in order to get a good seed set minimizing the value Trace $\left((\mathbf{S}\mathbf{S}^\top)^{-1}\right)$, where $\mathbf{S} \in \mathbb{R}^{d \times L}$ is a submatrix in the items' factor $\mathbf{Q} \in \mathbb{R}^{d \times m}$ of a rank-d decomposition. Each deletion of an item requires an iterative lookup of all the items in the data, where each iteration costs $O(d^2 L)$. So, one deletion takes $O(d^2 Lm)$ operations. Assuming that $L_0 \ll m$, the whole procedure takes $O(d^2 m^3)$ operations, which is too expensive to be computed on real-world datasets (the authors have selected a small subset of users to perform their evaluation).

2.5.8 Representative Based Matrix Factorization

In this section, we focus on a one more rating elicitation algorithm of seeking for a set of representative items that relies on the following intuitions. First, such algorithm should not select items, if they are not popular and thus cover preferences of only a small non-representative group of users. That means that the latent vectors from the seed set should have large norms. Second, the algorithm has to select diverse items that are relevant to different users with different tastes. This can be formalized as choosing latent vectors that are far from being collinear. The requirements can be met by searching for a subset of columns of $\mathbf{Q}$ that maximizes the volume of the parallelepiped spanned by them. This intuition is demonstrated in Figure 4, which captures a two-dimensional latent space and three seed sets. The volume of each seed set is proportional to the area of the triangle built on the corresponding latent vectors. The dark grey triangles have small volumes (because they contain not diverse vectors or vectors with small length) and hence correspond to bad seed sets. Contrariwise, the light gray triangle has a large volume and represents a better seed set.

The method presented in [72], called Representative Based Matrix Factorization (RBMF), uses this intuition. It uses maximal-volume concept and the Maxvol algorithm [37] for searching the most representative rows or columns in the factors of a CF factorization. This approach is highly efficient and more accurate than all other competitors, but it also has one significant limitation. It must use the same rank of factorization as the desired number of representative users or items for the seed set. This work proposes a generalization of Maxvol that allows using different rank values. It often leads to a better recommendation accuracy, as shown in Section 6.6.

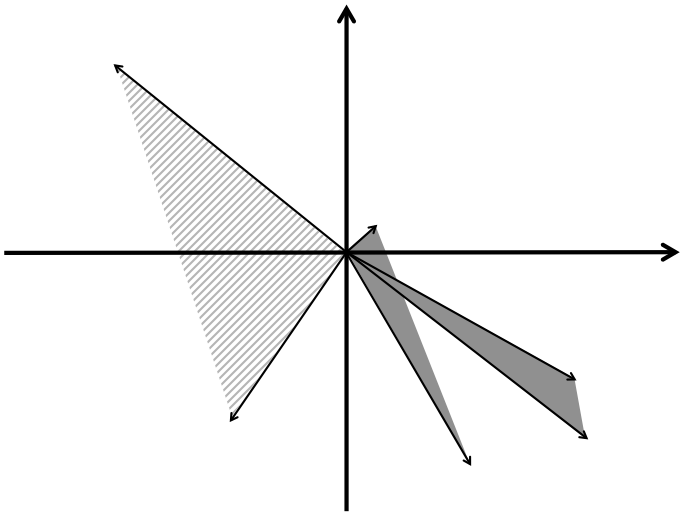

Figure 4: An illustration of the intuition behind Maxvol for searching the seed set: correlated or short embeddings form a triangles (dark grey) with a smaller volume

2.6 Chapter Summary

This chapter overviewed existing approaches to train word embeddings and embeddings of cold users and items from recommender data. Also, this chapter introduced all background required to understand the existing and new embedding methods described in the work. In the following chapter, we are going to discuss a general framework to train embeddings using low-rank matrix factorizations.

3 Matrix Factorization Framework to Train Embeddings

3.1 Framework

Low-rank matrix factorization techniques are widely used to build embeddings, even though sometimes the obtained low-dimensional object representations are not called "embeddings". We formalized the common ideas behind this folklore knowledge [111] and call this formalization a *framework*. Using this framework, we also develop new methods that are proposed in the thesis. The framework procedure consists of 2 steps:

Step 1. Let say, we want to build embeddings of some objects (e.g. words, users, etc). Denote the number of these objects as n. And then we need to construct descriptions (features) of these objects. As a result, matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ should be obtained, where m is a number of features.

Step 2. After this, we need to define an algorithm to factorize matrix $\mathbf{X}$ into two factors $\mathbf{P}$ and $\mathbf{Q}$. In the most popular case, if some matrix loss-function $\rho(\cdot, \cdot)$ is used, the factorization algorithm is equal to solving the following optimization problem:

$$\begin{aligned} &\rho(\mathbf{X}, \mathbf{P}\mathbf{Q}^\top) \to \min_{\mathbf{P},\mathbf{Q}}, \\ &\text{s.t. } \mathbf{P} \in \mathbb{R}^{n \times d},\ \mathbf{Q} \in \mathbb{R}^{m \times d}. \end{aligned} \tag{45}$$

The obtained matrix $\mathbf{P} \in \mathbb{R}^{n \times d}$ represents d-dimensional embeddings of each of n objects. After obtaining these embeddings, it becomes possible to measure their performance (see Section 1.3 for details).

Further sections show how the existing approaches and methods developed within

the thesis suit to this framework.

3.2 How Existing Embedding Approaches Fit to Framework

First of all, let us demonstrate how some of the existing methods fit to the framework described above.

3.2.1 Principal Component Analysis

Well-known Pricipal Component Analysis (PCA, also called Karhunen-Loeve transformation) [30] dimensionality reduction procedure natually fits to the framework above.

Step 1. Objects' descriptions are defined by the PCA task.

Step 2. Matrix factorization solves the optimization problem defined by Equation (45), where loss function $\rho(\mathbf{A}, \mathbf{B}) = ||\mathbf{A} - \mathbf{B}||_F$ is used. Since SVD provides the optimal solution of this problem, it is usually used to perform PCA.

3.2.2 Text Embeddings via Latent Semantic Analysis

Latent Semantic Analysis (LSA) [66] is a particular case of PCA:

Step 1. Objects are texts and they are described by Bag of Words representation [128] with TF-IDF statistics.

Step 2. Loss function $\rho(\mathbf{A}, \mathbf{B}) = ||\mathbf{A} - \mathbf{B}||_F$ is used for factorization.

3.2.3 Text Embeddings via Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis (PLSA) [48] solves the problem similar to LSA. The only difference is the loss function ρ used in the optimization.

3.2.4 Word Embeddings via SVD over SPPMI Matrix

The work [70] describes the method to train word embeddings using low-rank factorization of SSPMI matrix. Here is how it fit to the framework:

Step 1. Objects are natural language words. Their features are Shifted Positive Pointwise Mutual Information [70] of co-occurrences frequencies with other words within a same context.

Step 2. Loss function $\rho(\mathbf{A}, \mathbf{B}) = ||\mathbf{A} - \mathbf{B}||_F$ is used to perform the matrix factorization.

3.2.5 Users or Items Embeddings via PureSVD Recommender

PureSVD recommender approach learns users' and items' embedding via low-rank factorization of the rating matrix (see section 2.5 for details). This is how PureSVD fit the framework in case of user embeddings:

Step 1. Each user is described by ratings of films that he/she rated, and unknown ratings are filled with zeros.

Step 2. Loss function $\rho(\mathbf{A}, \mathbf{B}) = ||\mathbf{A} - \mathbf{B}||_F$ is used to perform the factorization.

3.2.6 Representative Based Matrix Factorization

As shown in Section 2.5.6, the cold start problem in recommenders could be solved by the rating elicitation procedure that obtains an embedding of a cold user (or, symmetrically, a cold item). This is how the Representative Based Matrix Factorization (RBMF) fits to the framework:

Step 1. Each user is described by ratings of films that he/she rated, and unknown ratings are filled with zeros.

Step 2. The rating matrix is factorized using the pseudo-skeleton approach, which means that users' embeddings consists of ratings for a small seed set of representative items.

3.3 How Developed Methods Fit Framework

This work focuses on developing new methods to train embeddings in accordance with the framework described above. In this section, you can find how the developed methods suit to the framework.

3.3.1 Building Embeddings of Categorical Features' Values

Chapter 4 is introduces a method to transform categorical (discrete) feature values into real value embeddings that could be trained in the unsupervised fashion. This is how the developed method fits the framework described above:

Step 1. Embedded objects are the values of categorical features. Each categorical feature's value is described by a vector of co-occurrence frequencies with another categorical feature's values. So each categorical feature's value has as many embeddings as is the number of the rest categorical features in the dataset.

Step 2. The factorization algorithm (45) uses various β-divergency loss functions. The performance of the embeddings is evaluated as a performance of simple supervised learning tasks where these embeddings are used instead of original categorical features' values.

3.3.2 Riemannian Optimization for Training Skip-Gram Negative Sampling Word Embeddings

Skip-Gram Negative Sampling (SGNS) word embedding model, well known by its implementation in "word2vec" software, is usually optimized by stochastic gradient descent. However, the optimization of SGNS objective can be viewed as a problem of searching for a good matrix with the low-rank constraint. The most straightforward way to solve this type of problems is to apply Riemannian optimization framework to optimize the SGNS objective over the manifold of required low-rank matrices. Chapter 5 proposes an algorithm that optimizes SGNS objective using Riemannian optimization and demonstrates its superiority over popular competitors.

This is how the developed method fits the framework described above:

Step 1. Objects are natural language words. Each word is described by a vector of co-occurrence frequencies of encountering this word within the same context with any other word from the dictionary.

Step 2. This work reformulated the original algorithm training Skip-gram Negative Sampling embeddings as a two-steps procedure, where the first step naturally fits to the Riemannian optimization approach, which we successfully applied. The second step, in turn, decomposes an obtained low-rank solution into two factors $\mathbf{W}$ and $\mathbf{C}^\top$. We used standard datasets to evaluate the performance of such embeddings.

3.3.3 Obtaining Cold User and Item Embeddings in Recommender Systems

The cold start problem in Collaborative Filtering can be solved by asking new users to rate a small seed set of representative items or by asking representative users to

rate a new item (see Section 2.5 for details). The question is how to build a seed set that can give enough preference information for making good recommendations. Chapter 6 introduces a fast algorithm for an analytical generalization of the existing approach (RBMF) based on Maxvol algorithm, which we call Rectangular Maxvol.

This is how the developed method fits the framework described above:

Step 1. Objects are users (or items) including cold ones. Each user (item) is described by a vector of ratings that he left for all items (that were left by all users). Unknown ratings are filled with zeros according to the PureSVD approach.

Step 2. The novel pseudo-skeleton factorization algorithm is applied to select a seed set, and ratings from the seed set form embeddings of cold objects. The performance of these embeddings is evaluated as a performance of recommendations based on them.

As you can see, the approaches introduced in this thesis also naturally fit the framework described above.

3.4 Chapter Summary

This chapter formalized a general framework that allows to train embeddings using low-rank matrix factorizations. It was shown that this framework comprises existing embedding approaches and the new methods developed in this thesis. Three following chapters introduce these novel methods.

4 Building Embeddings of Categorical Features' Values

4.1 Section Overview

Most of existing machine learning techniques can handle objects described by real but not categorical features. This chapter introduces a simple unsupervised method to transform categorical features' values into real-valued embeddings. It is based on low-rank approximations of co-occurrence matrices of categorical features' values. Once object's categorical features are transformed into real-valued, any common real-valued machine learning technique can be applied for further data analysis. We tested our approach in several supervised learning problems with categorical features. The experiments show that the combination of the proposed features transformation method with a common real-valued supervised algorithm (classical Random Forest predictor) leads to the results that are comparable to the state-of-the-art approaches that are able to handle categorical features out-of-box, such as Factorization Machines.

4.2 Proposed Methods

4.2.1 Transformation Using Direct Feature Value Frequencies

Suppose we have training data matrix $\mathbf{X}$ with size $n \times m$ that describes n objects with m features, and all features are categorical. In this section, we construct the method that transforms each pair of features into a new real-valued feature using categorical features' values co-occurrence frequencies, which contain essential information about the internal dataset structure.

Let us denote a transformed real-valued matrix as $\mathbf{Z} \in \mathbb{R}^{n \times m'}$, where each row corresponds to an object from $\mathbf{X}$, all categorical features' values of $\mathbf{X}$ are replaced with

real-valued ones and m' is the dimensionality of the new feature space. Then we can compute every new feature j_3 of dataset $\mathbf{Z}$ that corresponds to every pair of categorical features j_1 and j_2 of initial dataset $\mathbf{X}$ as their co-occurrence value frequency:

$$z_{i,j_3} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{1}[x_{i,j_1} = x_{k,j_1}] \cdot \mathbf{1}[x_{i,j_2} = x_{k,j_2}], \quad j_3 \in \{1, ..., m'\}, \tag{46}$$

where

$$m' = \frac{m(m-1)}{2}. \tag{47}$$

This new matrix $\mathbf{Z}$ contains only real-valued features and such transformation keeps the statistical information about "interactions" [91] between pairs of the categorical features.

Note that this transformation is unsupervised because it does not require target object labels $\mathbf{y}$. In other words, this transformation considers a generative structure of the object space and can be used for a wide range of machine learning tasks.

4.2.2 Low-Rank Frequency Approximations

In this section, we propose another categorical features transformation approach that is a modification of the previously discussed method. Instead of using the co-occurence frequencies of feature values explicitly, we propose to use low-rank approximations of these values.

Let us denote the unique values of features j_1 and j_2 as $\{a_k\}_{k=1}^{q_{j_1}}$ and $\{b_l\}_{l=1}^{q_{j_2}}$ respectively. Then we can construct matrix $\mathbf{G} \in \mathbb{R}^{q_{j_1} \times q_{j_2}}$ of co-occurence features' values frequencies:

$$g_{k,l} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[x_{i,j_1} = a_k] \cdot \mathbf{1}[x_{i,j_2} = b_l]. \tag{48}$$

Then, we can approximate matrix $\mathbf{G}$ as a product of two matrices with low rank d, where $d \leq q_{j_1}, d \leq q_{j_2}$:

$$\mathbf{G} \approx \mathbf{P}\mathbf{Q}^\top, \quad \mathbf{P} \in \mathbb{R}^{q_{j_1} \times d}, \quad \mathbf{Q} \in \mathbb{R}^{q_{j_2} \times d}. \tag{49}$$

Let us denote rows of $\mathbf{P}$ and $\mathbf{Q}$ corresponding to the values a and b of the j_1-th and j_2-th features of $\mathbf{X}$ respectively as $\mathbf{P}[a,:] \in \mathbb{R}^d$ and $\mathbf{Q}[b,:] \in \mathbb{R}^d$. So the scalar product of vectors $\mathbf{P}[a,:]$ and $\mathbf{Q}[b,:]$ is an approximated co-occurence frequency of pair (a,b). Using this transformation for every pair of the features, we can get new transformed matrix $\mathbf{Z} \in \mathbb{R}^{n \times m'}$:

$$z_{i,j_3} = \langle \mathbf{P}[x_{i,j_1},:], \mathbf{Q}[x_{i,j_2},:] \rangle, \quad j_3 \in \{1,...,m'\}, \tag{50}$$

where

$$m' = \frac{m(m-1)}{2}. \tag{51}$$

The scheme of this transformation process is illustrated in Figure 5. The usage of low-rank factorizations provides the reduction of the transformation model parameters number from quadratic $O(q_{j_1} \cdot q_{j_2})$ to linear $O(d \cdot (q_{j_1} + q_{j_2}))$, that is why this approach is helpful in avoiding extra overfitting. Note that since many elements of $\mathbf{G}$ equal to zero, usually it is computationally efficient to use low-rank factorization algorithms that work with sparse matrices [8].

4.2.3 Embeddings Based on Low-Rank Approximations

Vectors $\mathbf{P}[a,:]$ and $\mathbf{Q}[b,:]$ can be used as embeddings of categorical features' values a and b, which are based on the information about their co-occurence with other categorical features' values. They may be used explicitly for feature encoding instead of scalar

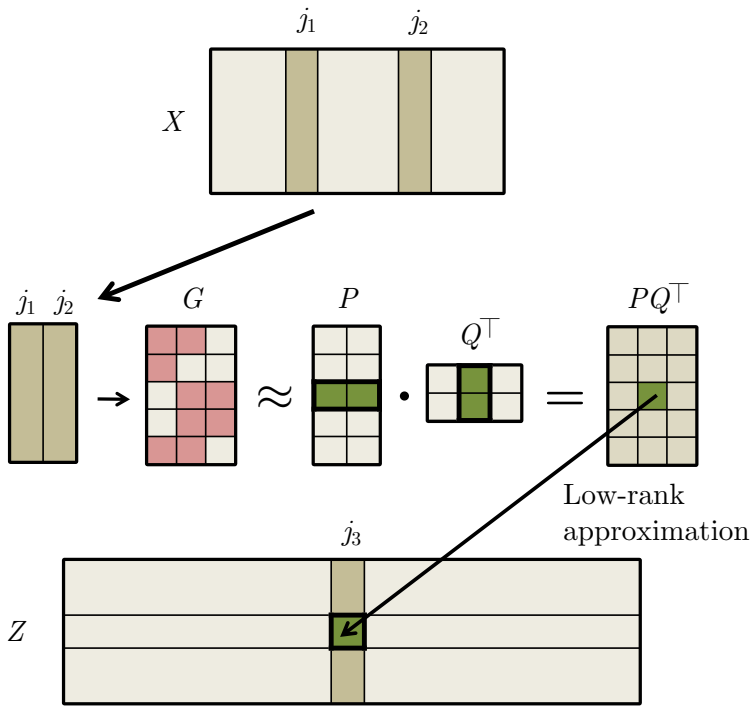

Figure 5: Scheme of using low-rank feature value frequency approximations for transformation categorical features of matrix X into real-valued features of Z.

multiplication, as it was done in the previous section. In this case, $2d$-dimensional set $\mathbf{j}_3$ of new transformed features' values of $\mathbf{Z} \in \mathbb{R}^{n \times m'}$ consists of the following embeddings concatenation:

$$\mathbf{Z}[i, \mathbf{j}_3] = \text{concat}\left(\mathbf{P}[x_{i,j_1}, :], \mathbf{Q}[x_{i,j_2}, :]\right), \quad j_3 \in \{1, ..., m'\}, \tag{52}$$

where

$$m' = 2d \cdot \frac{m(m-1)}{2} = dm(m-1). \tag{53}$$

Thus, every pair of features is transformed into new $2d$ real features, or, in other words, each feature value is replaced with a $d(m-1)$-dimensional embedding. This approach is unsupervised as well. The scheme of this process for all features is illustrated in Figure 6.

4.3 Experiments

We implemented three categorical feature transformation methods proposed in Section 4.2 using [8] and [28]. In order to measure the performance of any categorical features' values embedding approach, we use it in several supervised learning tasks for categorical features transformation and train the Random Forest real-valued supervised learning method using the transformed real-valued dataset. Its performance indicates the performance of the embedding approach. Some of the existing algorithms that can handle categorical features out-of-the-box (without transforming them into real-valued) were used in the experiments as baselines: Factorization Machines, sparse logistic regression, Naive Bayes.

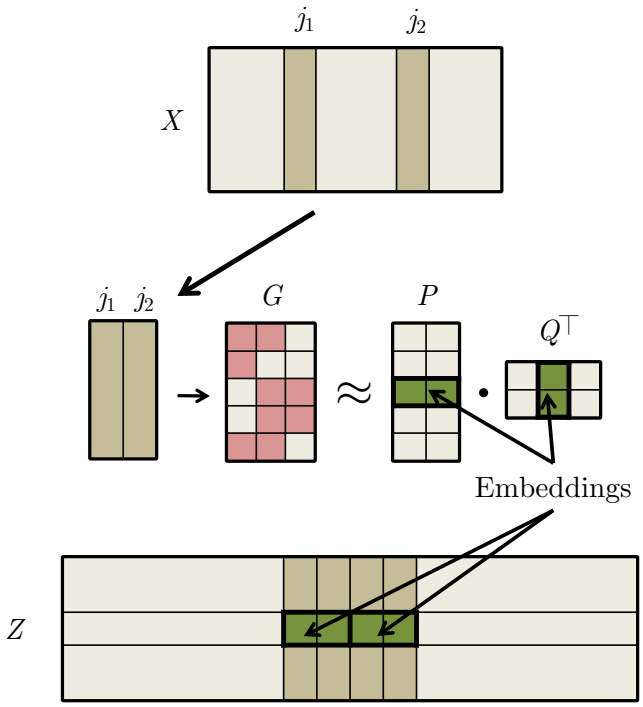

Figure 6: Scheme of using low-rank features' value embeddings for transformation of categorical features.

4.3.1 Datasets

We use two supervised datasets in the experiments. The first dataset was published at the international data mining competition *Amazon.com – Employee Access Challenge*[2], which was held in 2013. We call this dataset shortly *Amazon* in the work. It represents a binary classification task. The training set has 32769 objects and approximately 94% of them belong to class 1, other objects belong to class 0. Each object corresponds to an employee access request to some resource. The target label shows whether the corresponding request was approved by a supervisor or not. The task is to provide a model that would automatically predict the probability of approvement.

Every object has a categorical feature description: *Employee ID*, *Resource ID*, *Department ID*, etc (9 features in total). The number of unique values of each feature is shown in Table 1.

Number of the feature	1	2	3	4	5	6	7	8	9
Number of unique values	7518	4243	128	177	449	343	2358	67	343

Table 1: Number of unique values for every feature in *Amazon* dataset

The second dataset is *Movie Lens 100K*[3]. We call it shortly *Movie Lens*. The dataset provides 100000 user-film ratings, and each rating is described by the following categorical features: *User ID*, *Item ID*, *User social information* (several features), *Genre information*. The original dataset includes many binary indicators corresponding to the films genres (each film can have several genres). We merged these genre indicators together into a new categorical feature, where each unique value corresponds to a unique genre combination. The target object label is binary and represents whether a corresponding rating equals to 5 or not. Approximately 64% of the objects belong to the

[2] https://www.kaggle.com/c/amazon-employee-access-challenge
[3] https://grouplens.org/datasets/movielens/100k/

class 1. The number of unique values of every feature is shown in Table 2.

Number of the feature	1	2	3	4	5	6
Number of unique values	943	1682	2	21	795	216

Table 2: Number of unique values for every feature in *Movie Lens* dataset

4.3.2 Prediction Quality Estimation

Let us have true label vector $\mathbf{y} \in \{0, 1\}^n$ and vector of predictions $\tilde{\mathbf{y}} \in [0, 1]^n$ (e.g. it can be estimated probabilities of belonging to the class 1). The performance of predictions is computed as the area under the ROC-curve (AUC-ROC) [30].

We use the cross-validation technique to estimate the performance. We set 7-fold cross-validation for the *Amazon* dataset and 5-fold for the *Movie Lens* dataset. The cross-validated performance results of different algorithms are compared using Mann-Whitney U statistical test [74] with p-value bounded by 0.05 for statistical significance.

4.3.3 Results of the Experiments

The experiments show that the transformation based on low-rank embeddings (Section 4.2.3) is the most effective approach among the proposed ones. The performance of rank-10 transformations combined the Random Forest classifier is presented in Table 3. Also, you can see the results for the baselines. All the results are statistically significantly different, which is proved by Mann-Whitney U test.

First of all, from the table, we can see that the simple enumeration of categorical values provides very poor results. Also, we can note that the transformation approach in combination with Random Forest (see Section 2.2 for details) leads on the *Amazon* dataset and inferiors on *Movie Lens*. However, this advantage is expected because Fac-

torization Machines (see Section 2.2 for details) are designed as a recommender method, so it is natural that they show better results on the specific domain dataset *Movie Lens*. Summing it up, the proposed transformation approach combined with the popular Random Forest predictor allows us to get the performance comparable to the state-of-the-art.

Figure 7 shows the dependency of the prediction performance on the rank d of matrix approximation on the *Amazon* dataset. According to the results, it is not necessary to use rank much more than 10. The situation with the *Movie Lens* dataset is quite similar.

Another question is the choice of the matrix similarity measure used in low-rank matrix fatorizations. All results above use the Frobenius norm as this measure. Figure 8 shows the dependence of AUC-ROC score on the parameter β in β-divergency. We see that the optimization of Kullback-Leibler divergence ($\beta \to 1$) gives more precise results than the Frobenius norm optimization ($\beta = 2$). But the optimization of Kullback-Leibler divergence works much slower (more than 10^4 times slower) than the Frobenius norm optimization, because of efficient implementations of sparse SVD. That is why, we were not able to compute results for the *Amazon* dataset in case of the KL-divergence. Anyway, the obtained results are promising even in case of using Frobenius norm.

We also did not compare our approach to the industrial algorithm CatBoost[4] by Yandex, which focuses on processing categorical data, because it was developed much later that the current research was conducted and published.

[4] https://github.com/catboost/catboost

Method	Amazon	Movie Lens
Direct frequency transformation + RF	0.8503	0.7597
Low-rank approximated transformation + RF	0.8472	0.7539
Low-rank latent transformation + RF	**0.8817**	0.7702
Low-rank latent transformation + SVM	0.8442	0.7174
Simple numeration + RF	0.5703	0.5311
Sparse logistic regression	0.8691	0.7958
Smoothed naive Bayes	0.8776	0.7744
Factorization machines	0.8765	**0.8116**

Table 3: Comparison of proposed and existing approaches in terms of AUC-ROC performance measure. All performance differences are statistically significant.

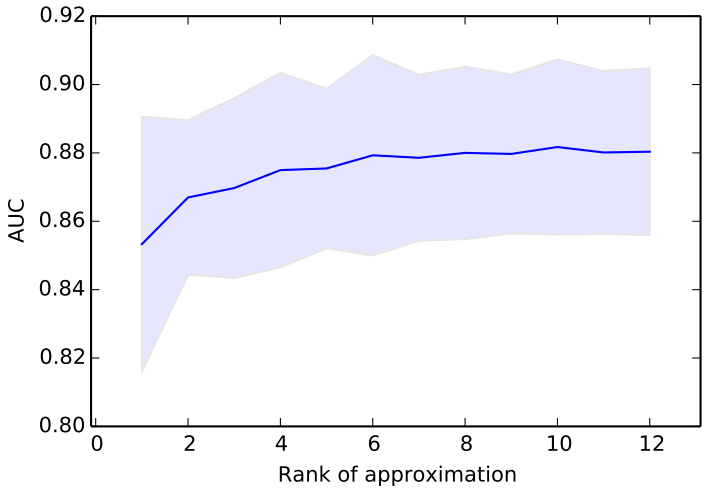

Figure 7: Dependence of AUC-ROC on parameter d with 2σ errorbar on *Amazon* dataset.

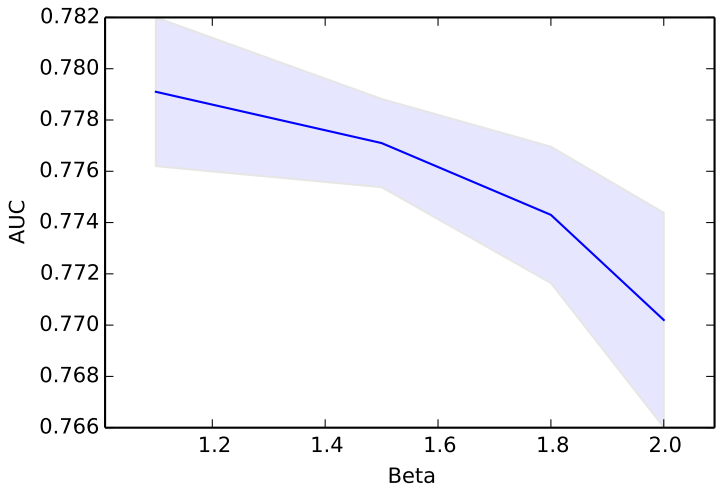

Figure 8: Dependence of AUC-ROC on parameter β with 2σ errorbar on *Movie Lens* dataset.

4.4 Chapter Summary

This chapter introduced a domain-independent method of unsupervised transformation of categorical features values into real-valued embeddings that are obtained from low-rank factors. Categorical data transformed in this way can be efficiently analyzed by common real-valued machine learning techniques. In order to demonstrate the performance of this embedding approach, it was shown that the categorical features transformation method based on the proposed embeddings and combined with widely used Random Forest binary classifier leads to the results comparable to the state-of-art approaches that can handle categorical features, such as Factorization Machines. Besides the binary classification problem, the proposed unsupervised embeddings can also be used for regression, multiclass classification, cluster analysis and many other machine learning problems.

5 Riemannian Optimization for Training Skip-Gram Negative Sampling Word Embeddings

5.1 Section Overview

In this chapter, we consider the problem of embedding words into a low-dimensional space in order to measure the semantic similarity between them. As an example, how to find whether the word "table" is semantically more similar to the word "chair" than to the word "sky"? That is achieved by constructing a low-dimensional vector representation for each word and measuring the similarity between the words as the similarity between the corresponding vectors. The dimensionality of this embedding space is a hyperparameter of the embedding training problem.

One of the most popular word embedding models [76], described in Section 2.4.2, is a discriminative neural network that optimizes Skip-Gram Negative Sampling (SGNS) objective (see Equation (38)). It aims at predicting whether two words can be found close to each other within a text. As shown in Section 5.2, the process of word embeddings training using SGNS can be divided into two general steps with clear objectives (in this chapter, we use the matrix factorization notation that is standard in the literature and different from the rest of the thesis):

Step 1. Search for a low-rank matrix $\mathbf{X}$ that provides a good SGNS objective value;

Step 2. Search for a good low-rank representation $\mathbf{X} = \mathbf{W}\mathbf{C}^\top$ in terms of linguistic metrics, where $\mathbf{W}$ is a matrix of word embeddings and $\mathbf{C}$ is a matrix of so-called context embeddings.

Unfortunately, most previous approaches mixed these two steps into a single one, which

entails a not completely correct formulation of the optimization problem. For example, popular approaches to train embeddings (including the original "word2vec" implementation) do not take into account that the objective from Step 1 depends only on the product $\mathbf{X} = \mathbf{W}\mathbf{C}^\top$: instead of straightforward computing of the derivative w.r.t. $\mathbf{X}$, these methods are explicitly based on the derivatives w.r.t. $\mathbf{W}$ and $\mathbf{C}$, which complicates the optimization procedure. Moreover, such approaches do not take into account that parametrization $\mathbf{W}\mathbf{C}^\top$ of matrix $\mathbf{X}$ is non-unique and Step 2 is required. Indeed, for any invertible matrix S, we have

$$\mathbf{X} = \mathbf{W}_1\mathbf{C}_1^\top = \mathbf{W}_1\mathbf{S}\mathbf{S}^{-1}\mathbf{C}_1^\top = \mathbf{W}_2\mathbf{C}_2^\top, \tag{54}$$

therefore, solutions $\mathbf{W}_1\mathbf{C}_1^\top$ and $\mathbf{W}_2\mathbf{C}_2^\top$ are equally good in terms of the SGNS objective but entail different cosine similarities between embeddings and, as a result, different performance in terms of linguistic metrics (see Section 5.4.2 for details).

A successful attempt to follow the described above steps, which outperforms the original SGNS optimization approach in terms of various linguistic tasks, was proposed in [70]. In order to obtain a low-rank matrix $\mathbf{X}$ on Step 1, this method reduces the dimensionality of Shifted Positive Pointwise Mutual Information (SPPMI) matrix via Singular Value Decomposition (SVD). On Step 2, it computes embeddings $\mathbf{W}$ and $\mathbf{C}$ via a simple formula that depends on the factors obtained by SVD. However, this method has one important limitation: SVD provides a solution to a surrogate optimization problem, which has no direct relation to the SGNS objective. In fact, SVD minimizes the Mean Squared Error (MSE) between $\mathbf{X}$ and SPPMI matrix, which does not lead to minimization of SGNS objective in general (see Section 2.4.4 and Section 4.2 in [70] for details).

These issues bring us to the main idea of the work: while keeping the low-rank matrix search setup on Step 1, optimize the original SGNS objective directly. This leads to an optimization problem over matrix $\mathbf{X}$ with the low-rank constraint, which is often [77] solved by applying *Riemannian optimization* framework [112]. In the work, we use the projector-splitting algorithm [73] (see Section 2.1.4 for details), which is easy to implement and has low computational complexity. Of course, Step 2 may be improved as well, but we regard this as a direction of future work.

As a result, the approach achieves the significant improvement in terms of SGNS optimization on Step 1 and, moreover, the improvement on Step 1 entails the improvement on Step 2 in terms of linguistic metrics. That is why the proposed two-step decomposition of the problem makes sense, which, most importantly, opens the way to applying even more advanced approaches based on it (e.g., more advanced Riemannian optimization techniques for Step 1 or a more sophisticated treatment of Step 2).

To summarize, the main contributions of this chapter are:

- we reformulated the problem of SGNS word embedding learning as a two-step procedure with clear objectives;
- For Step 1, we developed an algorithm based on Riemannian optimization framework that optimizes SGNS objective over low-rank matrix $\mathbf{X}$ directly;
- The algorithm outperforms state-of-the-art competitors in terms of SGNS objective and the semantic similarity linguistic metric [70, 76, 100].

5.2 Problem Setting

5.2.1 Matrix Notation of the Problem

Relying on the prospect proposed in [70], let us show that the SGNS optimization problem given by (38) from Section 2.4.2 can be considered as a problem of searching for a matrix that maximizes a certain objective function and has the rank-d constraint (Step 1 in the scheme described in Section 5.1). In the current section, we use the same notation as introduced in Section 2.4.2.

Denote the size of the words vocabulary $|V_W|$ as n and the size of the context words vocabulary $|V_C|$ as m. Let $\mathbf{W} \in \mathbb{R}^{n \times d}$ and $\mathbf{C} \in \mathbb{R}^{m \times d}$ be matrices, where each row $\mathbf{w} \in \mathbb{R}^d$ of matrix $\mathbf{W}$ is the word embedding of the corresponding word w and each row $\mathbf{c} \in \mathbb{R}^d$ of matrix $\mathbf{C}$ is the context embedding of the corresponding context c. Then the elements of the product of these matrices

$$\mathbf{X} = \mathbf{W}\mathbf{C}^\top \tag{55}$$

are the scalar products $x_{w,c}$ of all pairs (w, c):

$$\mathbf{X} = (x_{w,c}), \quad w \in V_W, c \in V_C. \tag{56}$$

Note that this matrix has rank d, because $\mathbf{X}$ equals to the product of two matrices with sizes $(n \times d)$ and $(d \times m)$. Now we can write SGNS objective given by (38) as a function of $\mathbf{X}$:

$$F(\mathbf{X}) = \sum_{w \in V_W} \sum_{c \in V_C} f_{w,c}(x_{w,c}), \quad F : \mathbb{R}^{n \times m} \to \mathbb{R}, \tag{57}$$

where

$$f_{w,c}(x_{w,c}) = l_{w,c}(\mathbf{w}, \mathbf{c}) = \#(w,c)\log\sigma(\langle \mathbf{w}, \mathbf{c}\rangle) + \\ + k\frac{\#(w)\#(c)}{|D|}\log\sigma(-\langle \mathbf{w}, \mathbf{c}\rangle). \tag{58}$$

This arrives us at the following proposition:

Proposition 1. *SGNS optimization problem given by (38) can be rewritten in the following constrained form:*

$$\begin{aligned} &\textit{maximize} \quad F(\mathbf{X}), \\ &\textit{subject to} \quad \mathbf{X} \in \mathcal{M}_d, \end{aligned} \tag{59}$$

where $\mathcal{M}_d$ is the manifold [112] of all matrices in $\mathbb{R}^{n\times m}$ with rank d:

$$\mathcal{M}_d = \{\mathbf{X} \in \mathbb{R}^{n\times m} : \textit{rank}(\mathbf{X}) = d\}. \tag{60}$$

The key idea of this chapter is to solve the optimization problem given by (59) via the framework of Riemannian optimization, which we introduce in Section 2.1.4.

Important to note that this prospect does not suppose the optimization over parameters $\mathbf{W}$ and $\mathbf{C}$ directly. This entails the optimization in the space with $((n+m-d)\cdot d)$ degrees of freedom [78] instead of $((n+m)\cdot d)$, which simplifies the optimization process (see Section 5.5 for the experimental results).

5.2.2 Computing Embeddings from a Low-Rank Solution

Once $\mathbf{X}$ is found, we need to recover $\mathbf{W}$ and $\mathbf{C}$ such that $\mathbf{X} = \mathbf{W}\mathbf{C}^\top$ (Step 2 in the scheme described in Section 5.1). This problem does not have a unique solution, since if $(\mathbf{W}, \mathbf{C})$ satisfy this equation, then $\mathbf{W}\mathbf{S}^{-1}$ and $\mathbf{C}\mathbf{S}^\top$ satisfy it as well for any non-

singular matrix $\mathbf{S}$. Moreover, different solutions may achieve different values of the linguistic metrics (see Section 5.4.2 for details). While the work focuses on Step 1, we use, for Step 2, a heuristic approach that was proposed in [71] and it shows good results in practice. We compute SVD of $\mathbf{X}$ in the form

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{61}$$

where $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, and $\mathbf{\Sigma}$ is the diagonal matrix, and use

$$\mathbf{W} = \mathbf{U}\sqrt{\mathbf{\Sigma}}, \quad \mathbf{C} = \mathbf{V}\sqrt{\mathbf{\Sigma}} \tag{62}$$

as matrices of embeddings. Since $\mathbf{X}$ is stored in the low-rank format, the SVD could be done efficiently.

A simple justification of this solution is the following: we need to map words into vectors in a way that similar words would have similar embeddings in terms of cosine similarities:

$$\cos(\mathbf{w}_1, \mathbf{w}_2) = \frac{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\|\mathbf{w}_1\| \cdot \|\mathbf{w}_2\|}. \tag{63}$$

It is reasonable to assume that two words are similar, if they share contexts. Therefore, we can estimate the similarity of two words w_1, w_2 as

$$s(w_1, w_2) = \sum_{c \in V_C} x_{w_1,c} \cdot x_{w_2,c}, \tag{64}$$

which is the element of the matrix $\mathbf{X}\mathbf{X}^\top$ with indices (w_1, w_2). Note that

$$\mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top. \tag{65}$$

If we choose $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}$, we exactly obtain $\langle \mathbf{w_1}, \mathbf{w_2} \rangle = s(w_1, w_2)$, since $\mathbf{W}\mathbf{W}^\top = \mathbf{X}\mathbf{X}^\top$ in this case. That is, the cosine similarity of the embeddings $\mathbf{w_1}, \mathbf{w_2}$ coincides with the intuitive similarity $s(w_1, w_2)$. However, scaling by $\sqrt{\mathbf{\Sigma}}$ instead of $\mathbf{\Sigma}$ was shown in [71] to be a better solution in experiments.

5.3 Algorithm

In order to solve the optimization problem given by (59), we use the Projector-Splitting algorithm [1], described in Section 2.1.4. In case of SGNS objective given by (57), an element of gradient ∇F required for the optimization method has the form:

$$\begin{aligned}(\nabla F(\mathbf{X}))_{w,c} &= \frac{\partial f_{w,c}(x_{w,c})}{\partial x_{w,c}} = \\ &= \#(w,c) \cdot \sigma\left(-x_{w,c}\right) - k\frac{\#(w)\#(c)}{|D|} \cdot \sigma\left(x_{w,c}\right).\end{aligned} \tag{66}$$

To make the method more flexible in terms of convergence properties, we additionally use $\lambda \in \mathbb{R}$, which is a step size parameter. In this case, retractor R returns $\mathbf{X}_i + \lambda \nabla F(\mathbf{X}_i)$ instead of $\mathbf{X}_i + \nabla F(\mathbf{X}_i)$ onto the manifold. The whole optimization procedure is summarized in Algorithm 3.

5.4 Experimental Setup

5.4.1 Training Models

We compare the proposed method ("RO-SGNS" in the tables) performance to two baselines: SGNS embeddings optimized via Stochastic Gradient Descent, implemented in the original "word2vec", ("SGD-SGNS" in the tables) [76] and embeddings obtained by SVD over SPPMI matrix ("SVD-SPPMI" in the tables) [70]. we have also experi-

Algorithm 3 Riemannian Optimization for SGNS

Require: Dimentionality d, initialization $\mathbf{W}_0$ and $\mathbf{C}_0$, step size λ, gradient function $\nabla F : \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times m}$, number of iterations K

Ensure: Factor $\mathbf{W} \in \mathbb{R}^{n \times d}$

$\mathbf{X}_0 \leftarrow \mathbf{W}_0 \mathbf{C}_0^\top$ # get an initial point at the manifold

$\mathbf{U}_0, \mathbf{S}_0, \mathbf{V}_0^\top \leftarrow \text{SVD}(\mathbf{X}_0)$ # the first point satisfying the low-rank constraint

for $i \leftarrow 1, \ldots, K$ **do**

  $\mathbf{U}_i, \mathbf{S}_i \leftarrow \text{QR}\left((\mathbf{X}_{i-1} + \lambda \nabla F(\mathbf{X}_{i-1}))\mathbf{V}_{i-1}\right)$ # step of the block power method

  $\mathbf{V}_i, \mathbf{S}_i^\top \leftarrow \text{QR}\left((\mathbf{X}_{i-1} + \lambda \nabla F(\mathbf{X}_{i-1}))^\top \mathbf{U}_i\right)$

  $\mathbf{X}_i \leftarrow \mathbf{U}_i \mathbf{S}_i \mathbf{V}_i^\top$ # update the point at the manifold

end for

$\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^\top \leftarrow \text{SVD}(\mathbf{X}_K)$

$\mathbf{W} \leftarrow \mathbf{U}\sqrt{\mathbf{\Sigma}}$ # compute word embeddings

return $\mathbf{W}$

mented with the blockwise alternating optimization over factors W and C, but the results are almost the same as SGD results, that is why we do not include them in the work. The source code of the experiments is available online[1].

The models were trained on English Wikipedia "enwik9" corpus[2], which was previously used in most work on this topic. Like in previous studies, we counted only the words which occur more than 200 times in the training corpus [70, 76]. As a result, we obtained a vocabulary of 24292 unique tokens (set of words V_W and set of contexts V_C are equal). The size of the context window was set to 5 for all experiments, as it was done in [70, 76]. We conduct three series of experiments: for dimensionality $d = 100$, $d = 200$, and $d = 500$.

Optimization step size is chosen to be small enough to avoid huge gradient values. However, thorough choice of λ does not result in a significant difference in performance (this parameter was tuned on the training data only, the exact values used in experiments are reported below).

[1] https://github.com/AlexGrinch/ro_sgns

[2] http://mattmahoney.net/dc/textdata

5.4.2 Evaluation

We evaluate word embeddings via the word similarity task. We use the following popular datasets for this purpose: "wordsim-353" ([29]; 3 datasets), "simlex-999" [47] and "men" [18]. Original "wordsim-353" dataset is a mixture of the word pairs for both word similarity and word relatedness tasks. This dataset was split [2] into two intersecting parts: "wordsim-sim" ("ws-sim" in the tables) and "wordsim-rel" ("ws-rel" in the tables) to separate the words from different tasks. In the experiments, we use both of them on a par with the full version of "wordsim-353" ("ws-full" in the tables). Each dataset contains word pairs together with assessor-assigned similarity scores for each pair. As a quality measure, we use Spearman's correlation between these human ratings and cosine similarities for each pair. We call this quality metric *linguistic* in the work.

5.5 Results of Experiments

First of all, we compare the value of SGNS objective obtained by the methods. The comparison is demonstrated in Table 4.

	$d = 100$	$d = 200$	$d = 500$
SGD-SGNS	-1.68	-1.67	-1.63
SVD-SPPMI	-1.65	-1.65	-1.62
RO-SGNS	$\mathbf{-1.44}$	$\mathbf{-1.43}$	$\mathbf{-1.41}$

Table 4: Comparison of SGNS values (multiplied by 10^{-9}) obtained by the models. Larger is better.

We see that SGD-SGNS and SVD-SPPMI methods provide quite similar results; however, the proposed method obtains significantly better SGNS values, which proves the feasibility of using Riemannian optimization framework in SGNS optimization problem. It is interesting to note that SVD-SPPMI method, which does not optimize SGNS

Dim. d	Algorithm	ws-sim	ws-rel	ws-full	simlex	men
$d = 100$	SGD-SGNS	0.719	0.570	0.662	0.288	0.645
	SVD-SPPMI	0.722	0.585	0.669	0.317	**0.686**
	RO-SGNS	**0.729**	**0.597**	**0.677**	**0.322**	0.683
$d = 200$	SGD-SGNS	0.733	0.584	0.677	0.317	0.664
	SVD-SPPMI	0.747	0.625	0.694	0.347	**0.710**
	RO-SGNS	**0.757**	**0.647**	**0.708**	**0.353**	0.701
$d = 500$	SGD-SGNS	0.738	0.600	0.688	0.350	0.712
	SVD-SPPMI	0.765	0.639	0.707	0.380	**0.737**
	RO-SGNS	**0.767**	**0.654**	**0.715**	**0.383**	0.732

Table 5: Comparison of the methods in terms of the semantic similarity task. Each entry represents the Spearman's correlation between predicted similarities and the manually assessed ones.

five				he				main			
SVD-SPPMI		RO-SGNS		SVD-SPPMI		RO-SGNS		SVD-SPPMI		RO-SGNS	
Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.
lb	0.748	**four**	0.999	**she**	0.918	when	0.904	**major**	0.631	**major**	0.689
kg	0.731	**three**	0.999	was	0.797	had	0.903	busiest	0.621	**important**	0.661
mm	0.670	**six**	0.997	promptly	0.742	was	0.901	**principal**	0.607	line	0.631
mk	0.651	**seven**	0.997	having	0.731	who	0.892	nearest	0.607	external	0.624
lbf	0.650	**eight**	0.996	dumbledore	0.731	**she**	0.884	connecting	0.591	**principal**	0.618
per	0.644	and	0.985	**him**	0.730	by	0.880	linking	0.588	**primary**	0.612

Table 6: Examples of the semantic neighbors obtained for words "five", "he" and "main".

objective directly, obtains better results than SGD-SGNS method, which aims at optimizing SGNS. This fact additionally confirms the idea described in Section 5.2.1 that the independent optimization over parameters $\mathbf{W}$ and $\mathbf{C}$ may decrease the performance.

However, the target performance measure of embedding models is the correlation between semantic similarity and human assessment (Section 5.4.2). Table 5 presents the comparison of the methods in terms of it. We see that the proposed method outperforms the competitors on all datasets except for "men" dataset where it obtains slightly worse results. Moreover, it is essential that the higher dimension entails higher performance gain of the method in comparison to the competitors.

To understand how the model improves or degrades the performance in comparison

usa					
SGD-SGNS		SVD-SPPMI		RO-SGNS	
Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.
akron	0.536	**wisconsin**	0.700	**georgia**	0.707
midwest	0.535	**delaware**	0.693	**delaware**	0.706
burbank	0.534	**ohio**	0.691	**maryland**	0.705
nevada	0.534	northeast	0.690	**illinois**	0.704
arizona	0.533	cities	0.688	madison	0.703
uk	0.532	southwest	0.684	**arkansas**	0.699
youngstown	0.532	places	0.684	**dakota**	0.690
utah	0.530	counties	0.681	**tennessee**	0.689
milwaukee	0.530	**maryland**	0.680	northeast	0.687
headquartered	0.527	**dakota**	0.674	**nebraska**	0.686

Table 7: Examples of the semantic neighbors from 11th to 20th obtained for the word "usa" by all three methods. Top-10 neighbors for all three methods are exact names of states.

to the baseline, we found several words, whose neighbors in terms of cosine distance change significantly. Table 6 demonstrates neighbors of the words "five", "he" and "main" for both SVD-SPPMI and RO-SGNS models. A neighbor is marked bold if we suppose that it has similar semantic meaning to the source word. First of all, we notice that the model produces much better neighbors of the words describing digits or numbers (see word "five" as an example). A similar situation happens for many other words, e.g., in case of "main" — the nearest neighbors contain 4 similar words for the model instead of 2 in case of SVD-SPPMI. The neighborhood of "he" contains less semantically similar words in case of the model. However, it filters out irrelevant words, such as "promptly" and "dumbledore".

Table 7 contains the nearest words to the word "usa" from 11th to 20th. We marked names of USA states bold and did not represent top-10 nearest words as they are exactly names of states for all three models. Some non-bold words are arguably relevant as they present large USA cities ("akron", "burbank", "madison") or geographical regions of

several states (“midwest”, “northeast”, “southwest”), but there are also some completely irrelevant words (“uk”, “cities”, “places”) presented by first two models.

The experiments show that the optimal number of iterations K in the optimization procedure and step size λ depend on the particular value of d. For $d = 100$, we have $K = 7, \lambda = 5 \cdot 10^{-5}$, for $d = 200$, we have $K = 8, \lambda = 5 \cdot 10^{-5}$, and for $d = 500$, we have $K = 2, \lambda = 10^{-4}$. Moreover, the best results were obtained when SVD-SPPMI embeddings were used as an initialization of Riemannian optimization process.

Figure 9 illustrates how the correlation between semantic similarity and human assessment scores changes through iterations of the method. The optimal value of K is the same for both whole testing set and its 10-fold subsets chosen for cross-validation. The idea to stop optimization procedure on some iteration is also discussed in [64].

Training of the same dimensional models ($d = 500$) on English Wikipedia corpus using SGD-SGNS, SVD-SPPMI, RO-SGNS took 20 minutes, 10 minutes and 70 minutes respectively. The proposed method works slower, but not significantly. Moreover, since we were not focused on the code efficiency optimization, this time can be reduced.

5.6 Chapter Summary

In this part of the work, we proposed the general two-step scheme of training SGNS word embedding model that is successfully used by other researchers [132] and introduced the algorithm that performs the search of a solution in the low-rank form via Riemannian optimization framework. We also demonstrated the superiority of the method by providing an experimental comparison to existing state-of-the-art approaches.

Possible direction of future work is to apply more advanced optimization techniques to the Step 1 of the scheme proposed in Section 5.1 and to explore the Step 2 — obtain-

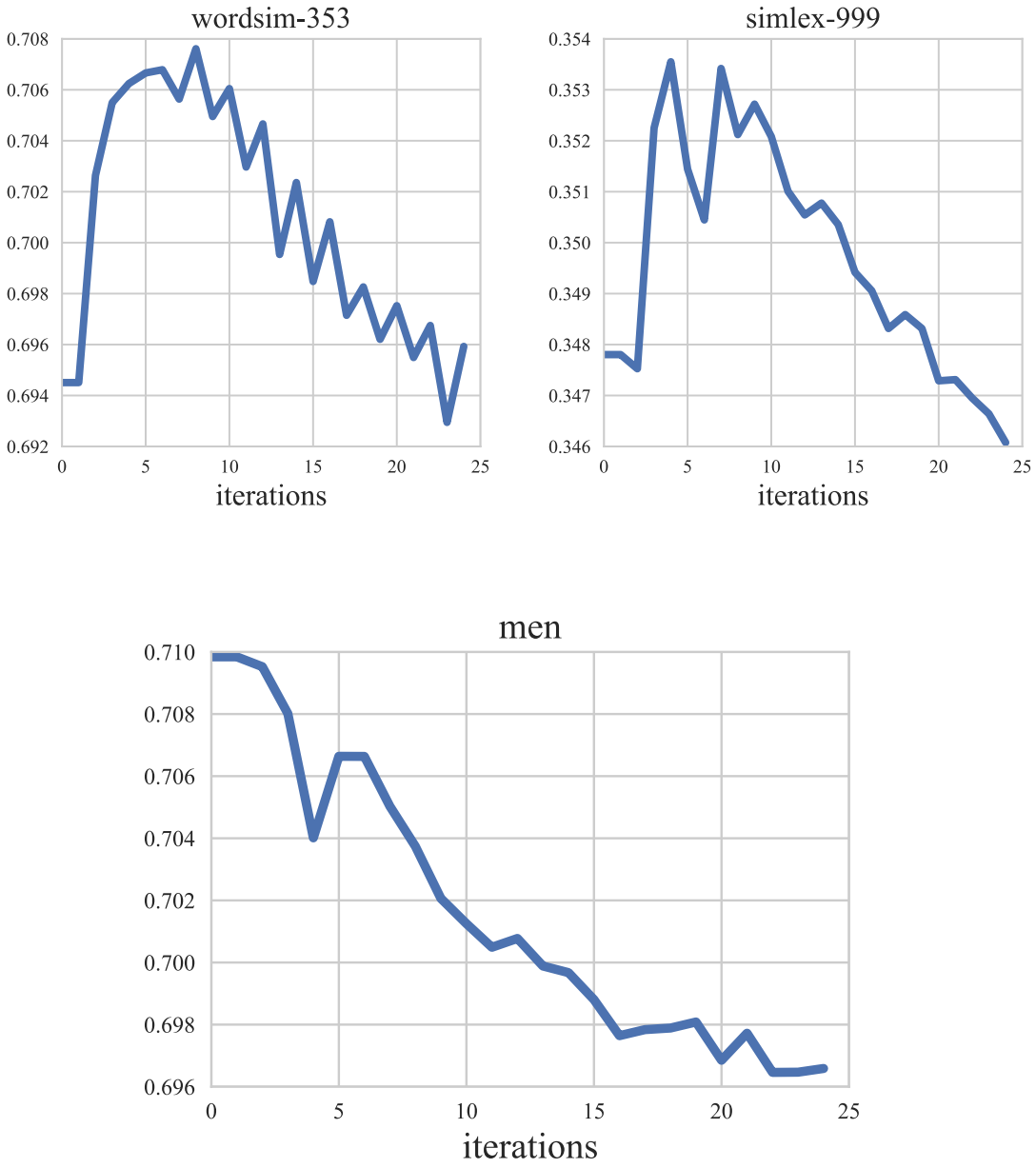

Figure 9: Illustration of why it is important to choose the optimal iteration and stop optimization procedure after it. The graphs show semantic similarity metric in dependence on the iteration of optimization procedure. The embeddings obtained by SVD-SPPMI method were used as initialization. Parameters: $d = 200$, $\lambda = 5 \cdot 10^{-5}$.

ing embeddings with a given low-rank matrix.

6 Cold User and Item Embeddings in Recommender Systems

6.1 Section Overview

As it was discussed in the Section 2.5.6, there is an approach to solve a cold start problem in collaborative filtering, called rating elicitation. Its main idea is to ask cold users to rate a seed set of representative set of items or conversely ask a representative set of users to rate cold items. These obtained ratings form embeddings of user and item that could be used for further data analysis.

From the overview in the background chapter, we know that one of the most popular and successful approaches [72] to perform rating elicitation is based on the maximal-volume concept [38]. Its general intuition is that the most representative seed set should consist of the most representative and diverse latent vectors, i.e., they should have the largest length yet be as orthogonal as possible to each other. Formally, the degree to which these two requirements are met is measured by the volume of the parallelepiped spanned by these latent vectors. In matrix terms, the algorithm, called Maxvol [37], searches very efficiently for a submatrix of a factor matrix with the locally maximal determinant. Unfortunately, the determinant is defined only for square matrices, which means that a given fixed size of a seed set requires the same rank of the matrix factorization that may be not optimal. For example, the search for a sufficiently large seed set requires a relatively high rank of factorization, and hence a higher rank implies a larger number of the model parameters and a higher risk of overfitting, which, in turn, decreases the quality of recommendations.

To overcome the intrinsic "squareness" of the ordinary Maxvol, which is entirely

based on the determinant, we introduce the notion of rectangular matrix volume, a generalization of the usual determinant. Searching a submatrix with high rectangular volume allows using ranks of the factorization that are lower than the size of a seed set. However, the problem of searching for the globally optimal rectangular submatrix is NP-hard in the general case. In this work, we propose a novel efficient algorithm, called Rectangular Maxvol, which generalizes the original Maxvol algorithm. It works in a greedy fashion and adds representative objects into a seed set one by one. This incremental update has a low computational complexity, which results in high algorithm efficiency. In this work, we provide a detailed complexity analysis of the algorithm and its competitors and present a theoretical analysis of its error bounds. Moreover, as demonstrated by the experiments, the rectangular volume notion leads to a noticeable quality improvement of recommendations on popular recommender datasets.

The major contribution of this chapter, which is described in Section 6.3, is a novel method to perform Step 1 from the Algorithm 2. It is based on PureSVD [25] Collaborative Filtering technique, that is described in Section 2.5. Before going into the details of the proposed approach, in Section 6.2, we discuss how to effectively perform Step 3 from Algorithm 2 using the similar factorization based approach.

6.2 Predicting Ratings with a Seed Set

Let us assume that some algorithm has selected a seed set with L_0 representative items with indices $\mathbf{k} \in \mathbb{N}^{L_0}$, and assume a cold user has been asked to rate only items $\mathbf{k}$, according to Steps 1–2 of the rating elicitation scheme described by Algorithm 2 from Section 2.5. In this section, we explain how to perform Step 3, i.e., how to predict ratings $\mathbf{z}$ for all items using only the ratings of the seed set.

As shown in [6], the most accurate way to do it is to find a coefficient matrix $\mathbf{C} \in \mathbb{R}^{L_0 \times m}$ that allows to linearly approximate each item rating via ratings $\mathbf{z}'$ of items from the seed set, which in fact is a pseudo-skeleton factorization, described in Section 2.1.5. Each column of $\mathbf{C}$ contains the coefficients of the representation of an item rating via the ratings of the items from the seed set. Shortly, this approximation can be written in the following way:

$$\mathbf{z} \leftarrow \mathbf{z}'\mathbf{C}. \tag{67}$$

We highlight two different approaches to compute matrix $\mathbf{C}$.

6.2.1 Computing coefficients via rating matrix

First approach is called Representative Based Matrix Factorization (RBMF) [72] (see Section 2.5 for details and notation). It aims to solve the following optimization task:

$$\|\mathbf{R} - \mathbf{R}[:, \mathbf{k}]\mathbf{C}\|_2 \to \min_{\mathbf{C}}. \tag{68}$$

Note that $\mathbf{z}'$ is not a part of $\mathbf{R}[:, \mathbf{k}]$, because there is still no information about a cold user ratings. This optimization task corresponds to the following approximation:

$$\mathbf{R} \approx \mathbf{R}[:, \mathbf{k}]\mathbf{C}. \tag{69}$$

The solution of (68) is:

$$\mathbf{C}_R = (\mathbf{R}[:, \mathbf{k}]^\top \mathbf{R}[:, \mathbf{k}])^{-1} \mathbf{R}[:, \mathbf{k}]^\top \mathbf{R}. \tag{70}$$

Since $L_0 \ll n$, the matrix $\mathbf{R}[:, \mathbf{k}]$ is often well-conditioned. Therefore, the regularization term used in [72] is unnecessary and does not give a quality gain.

6.2.2 Computing coefficients via a low-rank factor.

In this work, we also propose a more efficient second approach that considers the rank-d factorization, $d \leq L_0$. Let $\mathbf{Q}[\mathbf{k}, :] \in \mathbb{R}^{L_0 \times d}$ be the matrix formed by L_0 rows of $\mathbf{Q}$ that correspond to the items of the seed set. Let us try to linearly recover all item latent vectors via the latent vectors from the seed set:

$$\|\mathbf{Q} - \mathbf{C}^\top \mathbf{Q}[\mathbf{k}, :]\|_2 \to \min_{\mathbf{C}}. \tag{71}$$

It is a low-rank version of the problem given by (68) and, therefore, is computationally easier. Solution $\mathbf{C}$ of this optimization problem can be also used for recovering all ratings using (69).

Unlike (68), the optimization problem given by (71) has different exact solutions $\mathbf{C}$ in general case, because there are infinitely many ways to linearly represent an d-dimensional vector via more than d other vectors. Therefore, we should find a solution of the underdetermined system of linear equations:

$$\mathbf{Q}^\top = \mathbf{S}\mathbf{C}, \tag{72}$$

where we denote $\mathbf{S} = \mathbf{Q}[\mathbf{k}, :]^\top$. Since the seed set latent vectors surely contain some noise and coefficients in $\mathbf{C}$ show how all item latent vectors depend on the seed set latent vectors, it is natural to find "small" $\mathbf{C}$, because larger coefficients produce larger noise in predictions. We use the least-norm solution $\mathbf{C}$ in the research, which is addi-

tionally theoretically motivated in Section 6.5. The least-norm solution of (72) should be computed as follows:

$$\mathbf{C} = \mathbf{S}^{\dagger}\mathbf{Q}^{\top}, \tag{73}$$

where

$$\mathbf{S}^{\dagger} = \mathbf{S}^{\top}(\mathbf{S}\mathbf{S}^{\top})^{-1} \tag{74}$$

is the right pseudo-inverse of $\mathbf{S}$.

Such linear approach to rating recovering results in the following factorization model. Taking the latent vectors of the representative items $\mathbf{S}$ as a new basis of the decomposition given by Equation (44), we have

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^{\top} = \mathbf{P}\mathbf{S}\mathbf{C} = \left(\mathbf{P}\mathbf{Q}[\mathbf{k}, :]^{\top}\right)\mathbf{C} \approx \mathbf{R}[:, \mathbf{k}]\mathbf{C} = \mathbf{F}\mathbf{C}, \tag{75}$$

where

$$\mathbf{F} = \mathbf{R}[:, \mathbf{k}]. \tag{76}$$

In this way, we approximate an unknown rating r_{ui} by the corresponding entry of matrix $\mathbf{FC}$, where factor $\mathbf{F}$ consists of the known ratings for the seed set items. This scheme is illustrated in Figure 10.

6.3 Volume of Rectangular Matrices

The obvious disadvantage of the Maxvol-based approach to rating elicitation is the fixed size of the decomposition rank $d = L_0$, because the matrix determinant is defined only for square matrices. That makes it impossible to build a seed set with fixed size L_0 using an arbitrary rank of decomposition. This section introduces a generalization of

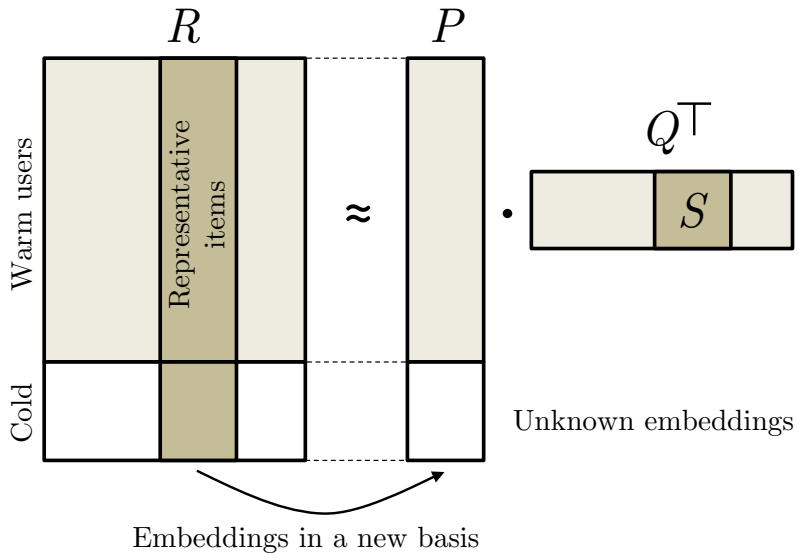

Figure 10: The rating elicitation procedure scheme in case of the cold user problem — the subset $\mathbf{S}$ of columns of $\mathbf{Q}^\top$ specifies the set of representative items, and their ratings are used as embeddings of cold users.

the maximal-volume concept to rectangular submatrices, which allows overcoming the intrinsic "squareness" of the ordinary maximal-volume concept, which is entirely based on the determinant of a square matrix. As we further demonstrate in Section 6.6 with experiments, using the rectangular Maxvol generalization with a decomposition of rank d smaller than the size L_0 of the seed set could result in better accuracy of recommendations for cold users.

Consider $\mathbf{S} \in \mathbb{R}^{d \times L_0}$, $d \leq L_0$. It is easy to see that the volume of a square matrix is equal to the product of its singular values. In the case of a rectangular matrix $\mathbf{S}$, its volume [97] can be defined in a similar way:

$$\text{Rectvol}(\mathbf{S}) := \prod_{s=1}^{L_0} \sigma_s = \sqrt{\det(\mathbf{S}\mathbf{S}^\top)}. \tag{77}$$

We call it *rectangular volume*. The simple intuition behind this definition is that it is the volume of the ellipsoid defined as the image of a unit sphere under the linear transformation defined by $\mathbf{S}$:

$$\text{Rectvol}(\mathbf{S}) = \text{Vol}\left\{\mathbf{v} \in \mathbb{R}^d : \exists \mathbf{c} \in \mathbb{R}_0^L, \|\mathbf{c}\|_2 \leq 1 \,|\, \mathbf{v} = \mathbf{S}\mathbf{c}\right\}. \tag{78}$$

This can be easily verified using the singular value decomposition of S and the unitary invariance of the vector norm. Moreover, in the case of a square matrix S, the rectangular volume is equal to the ordinary square volume:

$$\text{Rectvol}(\mathbf{S}) = \sqrt{\det(\mathbf{S}\mathbf{S}^\top)} = |\det(\mathbf{S})| = \text{Vol}(\mathbf{S}). \tag{79}$$

Note that, if $d > L_0$, then $\det \mathbf{S}\mathbf{S}^\top = 0$.

Overall, searching for a seed set transforms to the following optimization task that is a generalization of Problem (25):

$$\mathbf{k} \leftarrow \underset{\mathbf{k}}{\operatorname{argmax}} \ \text{Rectvol}(\mathbf{S}), \tag{80}$$

where $\mathbf{S} = \mathbf{Q}[\mathbf{k}, :]^\top$. It is important to note that this maximization problem does not depend on the basis of the latent vectors from $\mathbf{S}$. Indeed, let $\mathbf{X} \in \mathbb{R}^{d \times d}$ be an invertible matrix of some linear transformation. Then the optimization transforms to

$$\underset{\mathbf{k}}{\operatorname{argmax}} \left((\det \mathbf{X})^2 \cdot \text{Rectvol}(\mathbf{S}) \right) = \underset{\mathbf{k}}{\operatorname{argmax}} \ \text{Rectvol}(\mathbf{S}). \tag{81}$$

The simplest method to find a suboptimal solution is to use a greedy algorithm that iteratively adds rows of $\mathbf{Q}$ to the seed set. Unfortunately, the straightforward greedy optimization (trying to add each item to the current seed set and computing its rectangular volume) costs $O(mL_0^2d^2)$, that often is too expensive considering typical sizes of modern recommender datasets and a number of model hyperparameters. Therefore, we developed a fast algorithm with complexity $O(mL_0^2)$ that is described in the following section.

6.4 Algorithm

In this section, we introduce an algorithm for the selection of L_0 representative items using the notion of rectangular volume. At the first step, the algorithm computes the best rank-d approximation of the rating matrix $\mathbf{R}$, PureSVD (see Section 2.5.4 for details), and selects d representative items with the pivot indices from LU-decomposition of $\mathbf{Q}^\top$ or with the Maxvol algorithm. This seed set is further expanded by the Algorithm 4 in a

greedy fashion: by adding new representative items one by one maximizing rectangular volume of the seed set. Further, we show that new representative item should have the maximal norm of the coefficients that represent its latent vector by the latent vectors of the current seed set. The procedure of such norm maximization is faster than the straightforward approach. At the end of this section, we describe the algorithm for even faster rank-1 updating norms of coefficients.

6.4.1 Maximization of coefficients norm

Suppose, at some step, we have already selected $L < L_0$ representative items with the indices $\mathbf{k} \in \mathbb{N}^L$. Let $\mathbf{S} \in \mathbb{R}^{d \times L}$ be the corresponding submatrix of $\mathbf{Q}^\top \in \mathbb{R}^{d \times m}$. On the next step, the algorithm selects a column $\mathbf{q}_i \subset \mathbf{Q}^\top,\ i \notin \mathbf{k}$ and adds it to the seed set:

$$S \leftarrow [\mathbf{S}, \mathbf{q}_i]\,, \tag{82}$$

where $[\mathbf{A}, \mathbf{B}]$ is an operation of horizontal concatenation of two matrices $\mathbf{A}$ and $\mathbf{B}$. This column should maximize the following volume:

$$\mathbf{q}_i = \underset{i \notin \mathbf{k}}{\operatorname{argmax}} \operatorname{Rectvol}\left([\mathbf{S}, \mathbf{q}_i]\right). \tag{83}$$

Suppose $\mathbf{C} \in \mathbb{R}^{L \times m}$ is the current matrix of coefficients from Equation (72), and let $\mathbf{c}_i \in \mathbb{R}^L$ be an i-th column of matrix $\mathbf{C}$. Then the updated seed set from (83) can be written as following:

$$[\mathbf{S}, \mathbf{q}_i] = [\mathbf{S}, \mathbf{S}\mathbf{c}_i] = \mathbf{S} \cdot [\mathbf{I}_L, \mathbf{c}_i]. \tag{84}$$

Then the volume of the seed set can be written in the following way:

$$\begin{aligned}\text{Rectvol}\left([\mathbf{S}, \mathbf{q}_i]\right) &= \sqrt{\det\left([\mathbf{S}, \mathbf{q}_i] \cdot [\mathbf{S}, \mathbf{q}_i]^\top\right)} = \\ &= \sqrt{\det\left(\mathbf{S}\mathbf{S}^\top + \mathbf{S}\mathbf{c}_i\mathbf{c}_i^\top\mathbf{S}^\top\right)}.\end{aligned} \tag{85}$$

Taking into account the identity $\det(\mathbf{X} + \mathbf{AB}) = \det(\mathbf{X})\det(\mathbf{I} + \mathbf{B}\mathbf{X}^{-1}\mathbf{A})$, the volume (85) can be written as following:

$$\text{Rectvol}\left([\mathbf{S}, \mathbf{q}_i]\right) = \text{Rectvol}(\mathbf{S})\sqrt{1 + w_i}, \tag{86}$$

where $w_i = \|\mathbf{c}_i\|_2^2$. Thus, the maximization of rectangular volume is equivalent to the maximization of the l_2-norm of the coefficients vector $\mathbf{c}_i$, which we know only after recomputing (73). Total recomputing of coefficient matrix $\mathbf{C}$ on each iteration is faster than the straightforward approach described in Section 6.3 and costs $O(mL_0^2dm)$. However, in the next section, we describe even faster algorithm with an efficient recomputation of the coefficients.

6.4.2 Fast Computation of Coefficients

Since the matrix of coefficients $\mathbf{C}$ is the least-norm solution (44), after adding column $\mathbf{q}_i$ to the seed set, $\mathbf{C}$ should be computed using Equation (84):

$$\mathbf{C} \leftarrow [\mathbf{S}, \mathbf{q}_i]^\dagger\mathbf{Q}^\top = [\mathbf{I}_L, \mathbf{c}_i]^\dagger\mathbf{S}^\dagger\mathbf{Q}^\top = [\mathbf{I}_L, \mathbf{c}_i]^\dagger\mathbf{C}. \tag{87}$$

The pseudoinverse from (87) can be obtained in this way:

$$[\mathbf{I}_L, \mathbf{c}_i]^\dagger = [\mathbf{I}_L, \mathbf{c}_i]^\top \left([\mathbf{I}_L, \mathbf{c}_i] \cdot [\mathbf{I}_L, \mathbf{c}_i]^\top\right)^{-1} = \begin{bmatrix} \mathbf{I}_L \\ \mathbf{c}_i^\top \end{bmatrix} \left(\mathbf{I}_L + \mathbf{c}_i \mathbf{c}_i^\top\right)^{-1}, \tag{88}$$

where $\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$ is an operation of vectical concatenation of $\mathbf{A}$ and $\mathbf{B}$. The inversion in this formula can be computed by the Sherman-Morrison formula:

$$\left(\mathbf{I}_L + \mathbf{c}_i \mathbf{c}_i^\top\right)^{-1} = \mathbf{I}_L - \frac{\mathbf{c}_i \mathbf{c}_i^\top}{1 + \mathbf{c}_i^\top \mathbf{c}_i}.$$

Putting it into (87), we finally get the main update formula for $\mathbf{C}$:

$$\mathbf{C} \leftarrow \begin{bmatrix} \mathbf{I}_L - \frac{\mathbf{c}_i \mathbf{c}_i^\top}{1+\mathbf{c}_i^\top \mathbf{c}_i} \\ \\ \mathbf{c}_i^\top - \frac{\mathbf{c}_i^\top \mathbf{c}_i \mathbf{c}_i^\top}{1+\mathbf{c}_i^\top \mathbf{c}_i} \end{bmatrix} \cdot \mathbf{C} = \begin{bmatrix} \mathbf{C} - \frac{\mathbf{c}_i \mathbf{c}_i^\top \mathbf{C}}{1+\mathbf{c}_i^\top \mathbf{c}_i} \\ \\ \frac{\mathbf{c}_i^\top \mathbf{C}}{1+\mathbf{c}_i^\top \mathbf{c}_i} \end{bmatrix}. \tag{89}$$

Recall that we should efficiently recompute norms w_i of coefficients. Using Equation (89), we arrive at the following formula for the update of all norms w_j:

$$w_j \leftarrow w_j - \frac{(\mathbf{c}_i^\top \mathbf{c}_j)^2}{1 + \mathbf{c}_i^\top \mathbf{c}_i}. \tag{90}$$

It is natural to see that coefficients norms are decreasing, because adding each new latent vector to the seed set gives more flexibility of representing all latent vectors via representative ones.

Equations (89) and (90) allow to recompute $\mathbf{C}$ and $\mathbf{w}$ using the simple rank-1 update. Thus, the complexity of adding a new column into the seed set is low, which is shown

in Section 6.5.1. The pseudocode of the algorithm is provided in Algorithm 4.

Algorithm 4 Searching representative items using Rectangular Maxvol

Require: Rating matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, number of representative items L_0, rank of decomposition $d \leq L_0$

Ensure: Indices $\mathbf{k} \in \mathbb{N}^{L_0}$ of L_0 representative items

1: Compute rank-d PureSVD of the matrix $\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top$
2: Initialize a square seed set: $\mathbf{k} \leftarrow L_0$ pivot indices from LU-decomposition of $\mathbf{Q}^\top$
3: $\mathbf{S} \leftarrow \mathbf{Q}[\mathbf{k}, :]^\top$
4: $\mathbf{C} = \mathbf{S}^{-1}\mathbf{Q}^\top$
5: $\forall i: \; w_i \leftarrow \|\mathbf{c}_i\|_2^2$, where $\mathbf{c}_i$ is the i-th column of $\mathbf{C}$
6: **while** $\text{len}(\mathbf{k}) < L_0$ **do**
7: $\quad i \leftarrow \text{argmax}_{\text{i} \notin \mathbf{k}}(w_i)$
8: $\quad \mathbf{k} \leftarrow [\mathbf{k}, i]$
9: $\quad \mathbf{S} \leftarrow [\mathbf{S}, \mathbf{q}_i]$
10: $\quad \mathbf{C} \leftarrow \begin{bmatrix} \mathbf{C} - \frac{\mathbf{c}_i \mathbf{c}_i^\top \mathbf{C}}{1 + \mathbf{c}_i^\top \mathbf{c}_i} \\ \frac{\mathbf{c}_i^\top \mathbf{C}}{1 + \mathbf{c}_i^\top \mathbf{c}_i} \end{bmatrix}$
11: $\quad \forall j: \; w_j \leftarrow w_j - \frac{(\mathbf{c}_i^\top \mathbf{c}_j)^2}{1 + \mathbf{c}_i^\top \mathbf{c}_i}$
12: **end while**
13: **return** $\mathbf{k}$

The seed sets provided by the algorithm can be used for rating elicitation and further prediction of ratings for the rest of the items, as demonstrated in Section 6.2. Moreover, if the size of the seed set L_0 is not limited by a fixed budget, alternative stopping criteria is proposed in Section 6.5.

6.5 Algorithm Analysis

6.5.1 Complexity analysis

The proposed algorithm has two general steps: the initialization (Steps 1–5) and the iterative addition of columns or rows into the seed set (Steps 6–12). The initialization step corresponds to the LU-decomposition or Square Maxvol, which have $O(md^2)$ complex-

Algorithm	Complexity
Square Maxvol	$O(mL_0^2)$
Rectangular Maxvol	$O(mL_0^2)$
GreedyExtend	$O(m^2nL_0^2)$
Backward Greedy	$O(m^3d^2)$

Table 8: Complexity of the algorithms

ity. Addition of one element into the seed set (Steps 7–11) requires the recomputation of the coefficients $\mathbf{C}$ (Step 10) and lengths of coefficient vectors (Step 11). The recomputation (Step 10) requires a rank-1 update of the coefficients matrix $\mathbf{C} \in \mathbb{R}^{L \times m}$ and the multiplication $\mathbf{c}_i^\top \mathbf{c}$, where $\mathbf{c}_i \in \mathbb{R}^L$ is a column of $\mathbf{C}$. The complexity of each of the two operations is $O(Lm)$, so the total complexity of one iteration (Steps 7–11) is $O(Lm)$. Since this procedure is iterated over $L \in \{f, ..., L_0\}$, the complexity of the loop (Step 6) is equal to $O(m(L_0^2 - d^2))$. So, in total, the complexity of Algorithm 4 is $O(mL_0^2)$.

6.5.2 Comparing Complexity to Existing Approaches

Let us overview the computational complexity of the proposed Rectangular Maxvol and its competitors overviewed in Section 2.5. Some of these methods use low-rank factorizations of matrices, whose detailed complexity analysis is provided in [36]. However, as this is not a key point of the work, we neglect the computational cost of factorizations in the further analysis, because it is same for all rating elicitation algorithms and usually is previously computed for the warm Collaborative Filtering method. The summary of the complexity analysis is shown in Table 8. The detailed complexity analysis of Square Maxvol and Rectangular Maxvol is provided in Sections 2.1.5 and 6.5.1 respectively.

6.5.3 Analysis of Error

In this section, we theoretically analyse the estimation error of the method proposed in Section 6.4. According to Section 2.5.4 we have a low-rank approximation of the rating matrix $\mathbf{R} = \mathbf{P}\mathbf{Q}^\top + \mathcal{E}$, where $\mathcal{E} \in \mathbb{R}^{n \times m}$ is a random error matrix. On the other hand, we have RBMF approximation (69). Let us represent its error via $\mathcal{E}$.

First of all, we have

$$\mathbf{R}[:, \mathbf{k}] = \mathbf{P}\mathbf{Q}[\mathbf{k}, :]^\top + \mathcal{E}[:, \mathbf{k}] = \mathbf{P}\mathbf{S} + \mathcal{E}[:, \mathbf{k}]. \tag{91}$$

Since $\mathbf{C} = \mathbf{S}^\dagger \mathbf{Q}^\top$ (see Section 6.2 for details), the RBMF approximation of $\mathbf{R}$ can be written in the following form:

$$\mathbf{R}[:, \mathbf{k}]\mathbf{C} = \mathbf{P}\mathbf{S}\mathbf{S}^\dagger \mathbf{Q}^\top + \mathcal{E}[:, \mathbf{k}]\mathbf{C} = \mathbf{R} - \mathcal{E} + \mathcal{E}[:, \mathbf{k}]\mathbf{C}, \tag{92}$$

which means:

$$\mathbf{R} = \mathbf{R}[:, \mathbf{k}]\mathbf{C} + \mathcal{E} - \mathcal{E}[:, \mathbf{k}]\mathbf{C}. \tag{93}$$

The smaller in modulus the noise terms are, the better approximation of $\mathbf{R}$ we have. It means that we are interested in the small values of the matrix $\mathbf{C}$, such as the least-norm solution of (71). Further, we prove a theorem providing an approximated bound for the maximal length of $\mathbf{c}_i$.

6.5.4 Upper Bound of Coefficients Norm

Similarly to Square Maxvol algorithm, a rectangular submatrix is called *dominant*, if its rectangular volume does not increase by replacing one row with another one from the

source matrix.

Theorem 1. *Let* $\mathbf{Q} \in \mathbb{R}^{m \times d}$ *be a matrix of rank* d*. Assume* $\mathbf{k} \in \mathbb{N}^{L_0}$ *is a vector of seed set element indices that produces rank-*d *dominant submatrix of* $\mathbf{S} = \mathbf{Q}[\mathbf{k}, :]^\top$*, where* $\mathbf{S} \in \mathbb{R}^{d \times L_0}$ *and* $m \geq L_0 \geq d$*. Let* $\mathbf{C}$ *be a matrix of coefficients* $\mathbf{C} \in \mathbb{R}^{L_0 \times m}$*, such that* $\mathbf{Q}^\top = \mathbf{S}\mathbf{C}$*. Then* l_2*-norm of a column* $\mathbf{c}_i$ *of* $\mathbf{C}$ *for* i *not from the seed set is bounded as:*

$$\|\mathbf{c}_i\|_2 \leq \sqrt{\frac{d}{L_0 + 1 - d}}, \quad i \notin \mathbf{k}. \tag{94}$$

Proof. Since $\mathbf{S}$ is a dominant submatrix of the matrix $\mathbf{Q}$, it has the maximal rectangular volume among all possible submatrices of $[\mathbf{S}, \mathbf{q}_i]$ with the shape $d \times L_0$. Therefore, applying Lemma 1 (see Appendix) to the matrix $[\mathbf{S}, \mathbf{q}_i]$, we get

$$\det\left([\mathbf{S}, \mathbf{q}_i] \cdot [\mathbf{S}, \mathbf{q}_i]^\top\right) \leq \frac{L_0 + 1}{L_0 + 1 - d} \det(\mathbf{S}\mathbf{S}^\top). \tag{95}$$

Using Equation (86), we get:

$$\|\mathbf{c}_i\|_2^2 = \frac{\det\left([\mathbf{S}, \mathbf{q}_i] \cdot [\mathbf{S}, \mathbf{q}_i]^\top\right)}{\det(\mathbf{S}\mathbf{S}^\top)} - 1 \leq \frac{d}{L_0 + 1 - d}, \tag{96}$$

which finishes the proof. □

The similar theoretical result was obtained in [27]. However, the proof seems to be much easier and closely related to the notation used in the work and the proposed algorithm.

Theorem 1 demonstrates that, if we have an existing decomposition with the fixed rank d, it is enough to take $L_0 = 2d$ items to the seed set for getting all coefficients norm lesser than 1. This condition of representativeness has a very natural geometric

meaning: all item latent vectors are inside the ellipsoid spanned by the latent vectors from the seed set. The numerical experiments with randomly generated $d \times m$ matrices have shown, that Algorithm 4 requires only $L_0 \approx 1.2d$ rows to reach upper bound 2 for the length of each row of $\mathbf{C}$ and only $L_0 \approx 2d$ to reach the upper bound 1 for the length of each row of $\mathbf{C}$. So, although, the algorithm does not guarantee that the seed set submatrix is dominant, the experiment results are entirely consistent with the theory.

6.6 Experimental Setup

The proposed experiments compare two algorithms: Square Maxvol based (our primary baseline) and Rectangular Maxvol[5] based. Other competitors have either an infeasible computational complexity (see Section 6.5.2 for details) or have a lower quality than the baseline, as it is shown in [72]. Moreover, it is important to note that the experiments in [72] used smaller versions of the datasets. Therefore, the performance of Square Maxvol on the extended datasets is different from that reported in [72].

6.6.1 Datasets

We used two popular publicly available datasets in the experiments: the Movielens dataset[6] that contains 20,000,263 ratings and the Netflix dataset[7] that contains 100,480,507 ratings. The rating matrix R was formed in the same way as in [72].

[5]The source code is available here: https://bitbucket.org/muxas/rectmaxvol recommender
[6]http://grouplens.org/datasets/movielens/
[7]http://www.netflixprize.com/

6.6.2 Evaluation Protocol

Our evaluation pipeline for the comparison of the rating elicitation algorithms is similar to the one introduced in [72]. All the experiments are provided for both the user and the item cold start problems. However, without loss of generality, this section describes the evaluation protocol for the user cold start problem only. The item cold start problem can be evaluated in the same way after the transposition of the rating matrix.

We evaluate the algorithms for selecting representatives by the assessing the quality of the recommendations recovered after the acquisition of the actual ratings of the representatives, which can be done as shown in Section 6.2. Note that users may not have scores for the items from the seed set: if user u was asked to rate item i with an unknown rating, then, according to PureSVD model, r_{ui} is set to 0. We use the 5-fold cross validation with respect to the set of users in all experiments.

Pointwise quality measures are easy to be optimized directly, but they are not very suitable for recommendation quality evaluation, because the goal of a recommender system is not to predict particular rating values but to predict the most relevant recommendations that should be shown to the user. That is why we use ranking measures to evaluate all methods [44]. For evaluation, we divided all items for every user into relevant and irrelevant ones, as it was done in the baseline paper [72].

One of the most popular and interpretable ranking measures for the recommender systems evaluation are Precision@h and Recall@h [25] that measure the quality of top-h recommendations in terms of their relevance. Note that in the case of the item cold start problem, Precision@h and Recall@h are computed on the transposed rating matrix $\mathbf{R}$. It means that we sort items in the decreasing order of their relevance to a user and assess the top of this list with these measures. Moreover, following the methodology

from [72], we compare algorithms in terms of coverage and diversity.

6.7 Results of Experiments

As we mentioned in Section 6.2, there are two different ways to compute the coefficients for representing the hidden ratings via the ratings from a seed set. The first one is to calculate them via the low-rank factors, as shown in Equation (73). The second one is to compute them via the source rating matrix $\mathbf{R}$, as shown in Equation (70). The experiments show that the latter approach demonstrates the significantly better quality. Therefore, we use this method in all the experiments.

We processed experiments for the seed set sizes from 5 to 100 with a step of 5. These computations become possible for such dense grid of parameters, because of the high computational efficiency of the algorithm (see Section 6.5.2). The average computational time of Rectangular Maxvol on the datasets is 1.12 seconds (Intel Xeon CPU 2.00GHz, 26Gb RAM). The average computational time of Square Maxvol is almost the same which confirms the theoretical complexity analysis.

In the case of Rectangular Maxvol, for every size of the seed set, we used the rank that gives the best performance on a separate evaluation set of ratings. Figures 11 and 12 demonstrate the superiority of the approach over the ordinal Square Maxvol for all cold start problems types (user and item) and both datasets. Moreover, it can be seen from the magnitudes of the differences that Rectangular Maxvol gives much more stable results than the square one. The same conclusions can be made for any combination of Precision/Recall, h, and seed set sizes.

As mentioned above, Rectangular Maxvol used the optimal rank value in the experiments. Figure 13 demonstrates the averaged optimal rank over all experiments for all

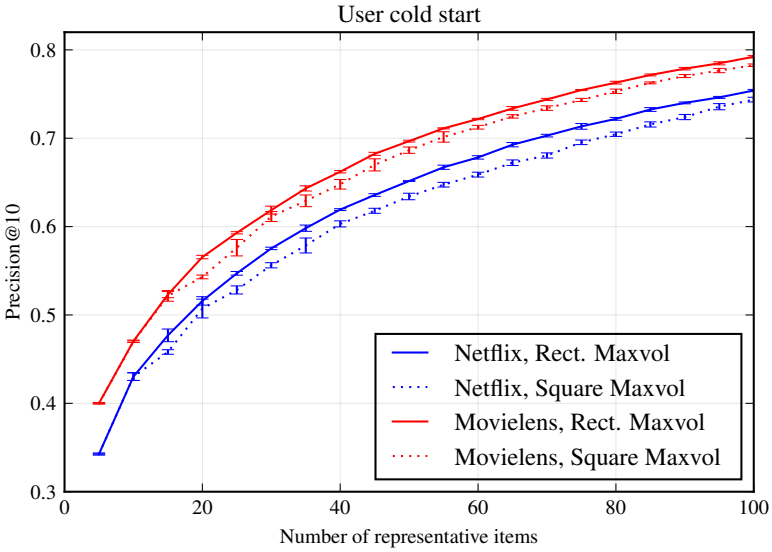

Figure 11: Precision@10 dependence on the size of the seed set. The comparison of Square Maxvol and Rectangular Maxvol. The errorbars indicate σ deviation. The Rectangular Maxvol approach outpreforms the baseline.

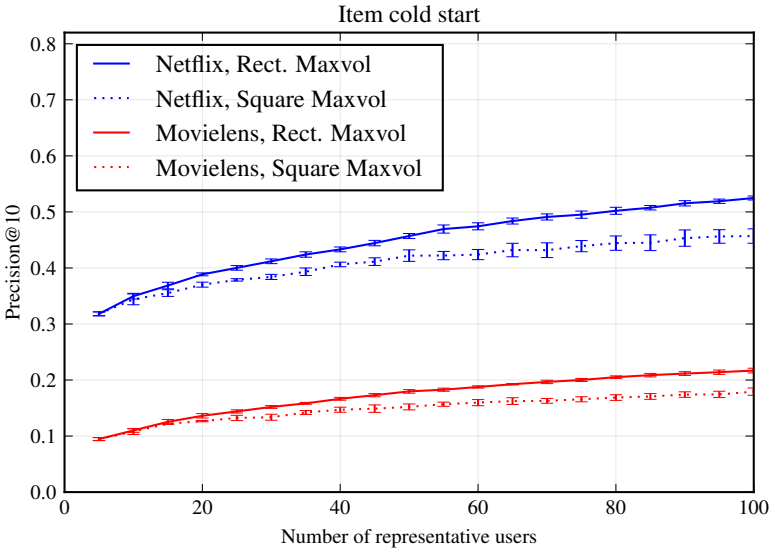

Figure 12: Precision@10 dependence on the size of the seed set. The comparison of Square Maxvol and Rectangular Maxvol. The errorbars indicate σ deviation. The Rectangular Maxvol approach outpreforms the baseline.

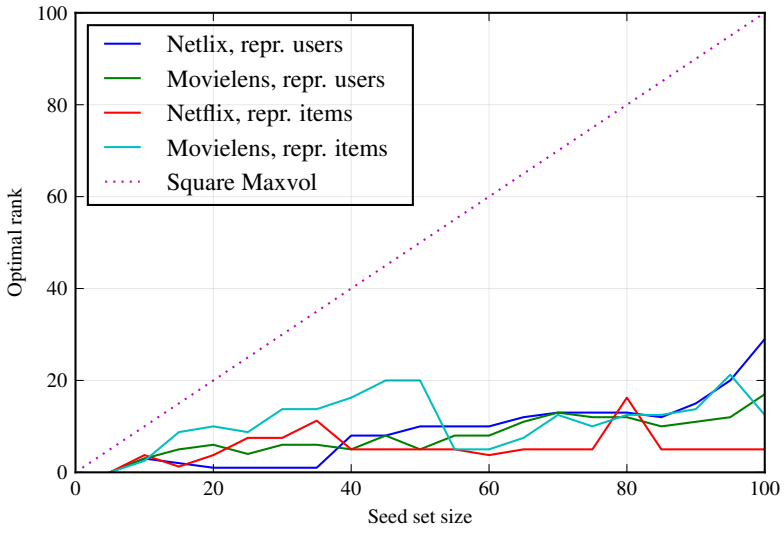

Figure 13: The optimal rank dependence on the size of the seed set — the optimal rank is much lower than the seed set size, which additionally proves that the Maxvol-based rating elicitation approach can be improved.

datasets and all cold start problem types. It is easy to see that, in each case, the required optimal rank is significantly smaller than the corresponding size of the seed set. This unequivocally confirms that the rectangular generalization of the square maximal-volume concept makes a great sense. Moreover, since Rectangular Maxvol requires a smaller rank of the rating matrix factorization, it is more computationally and memory efficient.

In Figure 14, we can see that the coverage and diversity measures of the representative Netflix items selected by Rectangular Maxvol are higher than the measures of Square Maxvol. The cases of representative users and Movielens dataset lead to the same results.

In the end, it is interesting to analyze the behavior of the automatic stopping criterion that adds objects into the seed set until all latent vectors are covered by the ellipsoid spanned by the latent vectors of the representatives. The experiments show that increas-

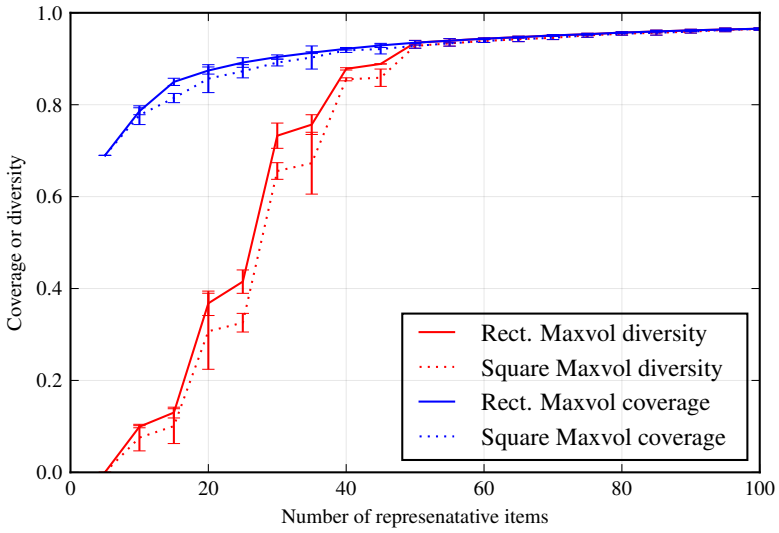

Figure 14: Coverage and diversity depending on the seed set size — the Rectangular Maxvol approach outpreforms the baseline.

ing of the rank results in a quality fall in the case of representative users and the ranks higher than 50, which means overfitting of PureSVD. In case of the representative items, the quality becomes almost constant starting from the same ranks.

6.8 Chapter Summary

In this part of the work, we introduced the efficient algorithm based on the novel definition of rectangular matrix volume which allows efficiently build pseudo-skeleton factorization and, as a result, constructing embeddings of cold users and items in recommender systems. In order to demonstrate the superiority of the proposed method, we provided the analytical and experimental comparison to the existing approaches.

7 Conclusion

Embedding methods play a crucial role in modern data science applications, from machine translation and speech recognition to recommender systems and search engines. The thesis overviews methods for training embeddings from data and explores the problem of training embeddings using low-rank approximations. There are the following main contributions of the work:

1. Chapter 3 introduces the simple low-rank factorization framework to train embeddings that generalizes existing approaches to train embeddings using matrix factorizations. Unfortunately, such techniques they have not been studied intensively, despite of the fact that they often show a performance that is not achievable the competitors. This framework allowed us to develop several new state-of-the-art embedding methods described in the following sections;

2. In Chapter 4, the thesis formulates the problem of unsupervised search for embeddings of categorical features' values and introduces the novel unsupervised method to train such embeddings;

3. In Chapter 5, we formulate the generalization of the Skip-Gram Negative Sampling word embeddings training procedure and, according to this formulation, introduce the new embeddings training method based on Riemannian optimization, which outperforms existing state-of-the-art approaches. The source code of the method is publicly available on the Web. Moreover, the introduced theoretical findings were recognized as useful and were used by other researchers [132].

4. In Chapter 6, we introduce the novel method to obtain embeddings of cold users and cold items in recommender systems based on the rectangular generalization

of the Maxvol algorithm. The developed method outperforms existing state-of-the-art approaches in terms of quality and computational complexity. The source code of the method is published on the Web and is used by other researchers in the community [118]. Furthermore, the method is successfully applied within an industrial recommender system at Yandex, which additionally proves the practical relevance of the approach.

As future directions of research, we wanted to highlight developing new methods in accordance with the low-rank framework proposed in the thesis. Currently, the community tends to use neural network based embedding methods, however, the low-rank based approaches often are much more suitable because of the developed theoretical and computational tools for low-rank matrix operations. We see a big room for new developments in word embeddings and recommender systems fields using advanced low-rank optimization techniques, such as Riemannian optimization, and in the active learning field using pseudo-skeleton based factorizations.

References

[1] P.-A. Absil and I. V. Oseledets. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 62(1):5–29, 2015.

[2] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27, 2009.

[3] M. Aharon, O. Anava, N. Avigdor-Elgrabli, D. Drachsler-Cohen, S. Golan, and O. Somekh. Excuseme: Asking users to help in item cold-start recommendations. In *Recsys'15*, pages 83–90, 2015.

[4] Q. Ai, Y. Zhang, K. Bi, X. Chen, and W. B. Croft. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 645–654. ACM, 2017.

[5] N. Almarwani and M. Diab. Arabic textual entailment with word embeddings. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 185–190, 2017.

[6] O. Anava, S. Golan, N. Golbandi, Z. Karnin, R. Lempel, O. Rokhlenko, and O. Somekh. Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design. In *WWW'15*, pages 45–54, 2015.

[7] M. Arioli and I. S. Duff. Preconditioning of linear least-squares problems by identifying basic variables. *Preprint RAL-P-2014-007*, 2014.

[8] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, December 2007.

[9] I. Barandiaran. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8), 1998.

[10] I. Barjasteh, R. Forsati, F. Masrour, A.-H. Esfahanian, and H. Radha. Cold-start item and user recommendation with decoupled completion and transduction. In *Recsys'15*, pages 91–98, 2015.

[11] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM'07*, pages 43–52, 2007.

[12] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[13] A. Bentbib and A. Kanber. Block power method for svd decomposition. *Analele Stiintifice Ale Unversitatii Ovidius Constanta-Seria Matematica*, 23(2):45–58, 2015.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[15] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[16] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[17] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[18] E. Bruni, N.-K. Tran, and M. Baroni. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(1-47), 2014.

[19] W. B. Cavnar, J. M. Trenkle, et al. N-gram-based text categorization. *Ann arbor mi*, 48113(2):161–175, 1994.

[20] Y. Chauvin and D. E. Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 2013.

[21] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[22] A. Civril and M. Magdon-Ismail. Finding maximum volume sub-matrices of a matrix. *RPI Comp Sci Dept TR*, pages 07–08, 2007.

[23] K. Clark and C. D. Manning. Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323*, 2016.

[24] P. Cremonesi, F. Garzottto, and R. Turrin. User effort vs. accuracy in rating-based elicitation. In *Recsys'12*, pages 27–34, 2012.

[25] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Recsys'10*, pages 39–46. ACM, 2010.

[26] A. Das, D. Ganguly, and U. Garain. Named entity recognition with word embeddings and wikipedia categories for a low-resource language. *ACM Transac-*

tions on Asian and Low-Resource Language Information Processing (TALLIP), 16(3):18, 2017.

[27] F. De Hoog and R. Mattheij. Subset selection for matrices. *Linear Algebra and its Applications*, 422(2):349–359, 2007.

[28] C. Févotte and J. Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

[29] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *WWW*, pages 406–414, 2001.

[30] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics New York, 2001.

[31] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[32] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[33] N. Golbandi, Y. Koren, and R. Lempel. On bootstrapping recommender systems. In *CIKM'10*, pages 1805–1808, 2010.

[34] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM'11*, pages 595–604, 2011.

[35] Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[36] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[37] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, and N. Zamarashkin. How to find a good submatrix. *Matrix methods: theory, algorithms and applications*, page 247, 2010.

[38] S. A. Goreinov and E. E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 280:47–52, 2001.

[39] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261(1):1–21, 1997.

[40] M. P. Graus and M. C. Willemsen. Improving the user experience during cold start through choice-based preference elicitation. In *Recsys'15*, pages 273–276, 2015.

[41] E. Grave, T. Mikolov, A. Joulin, and P. Bojanowski. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431, 2017.

[42] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, R. Baeza-Yates, A. Feng, E. Ordentlich, L. Yang, and G. Owens. Scalable semantic matching of queries to ads in sponsored search advertising. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 375–384. ACM, 2016.

[43] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1809–1818. ACM, 2015.

[44] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *The Journal of Machine Learning Research*, 10:2935–2962, 2009.

[45] M. A. Hardy. *Regression with dummy variables*. Number 93. Sage, 1993.

[46] B. Heinzerling and M. Strube. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. *arXiv preprint arXiv:1710.02187*, 2017.

[47] F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2016.

[48] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[49] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM'08*, pages 263–272, 2008.

[50] D. Jurafsky and H. James. Speech and language processing an introduction to natural language processing, computational linguistics, and speech. 2000.

[51] R. Karimi, A. Nanopoulos, and L. Schmidt-Thieme. Improved questionnaire trees for active learning in recommender systems. In *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM*, pages 34–44, 2014.

[52] R. Karimi, M. Wistuba, A. Nanopoulos, and L. Schmidt-Thieme. Factorized decision trees for active learning in recommender systems. In *ICTAI'13*, pages 404–411. IEEE, 2013.

[53] S. S. Keerthi, T. Schnabel, and R. Khanna. Towards a better understanding of predict and count models. *arXiv preprint arXiv:1511.02024*, 2015.

[54] D. Kiela and L. Bottou. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 36–45, 2014.

[55] D. Kluver and J. A. Konstan. Evaluating recommender behavior for new users. In *Recsys'14*, pages 121–128, 2014.

[56] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[57] D. Kolesnikov and I. Oseledets. Convergence analysis of projected fixed-point iteration on a low-rank matrix manifold. *Numerical Linear Algebra with Applications*, page e2140, 2016.

[58] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[59] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*, pages 426–434, 2008.

[60] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[61] S. Kottur, X. Wang, and V. R. Carvalho. Exploring personalized neural conversational models. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3728–3734. AAAI Press, 2017.

[62] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[64] S. Lai, K. Liu, S. He, and J. Zhao. How to generate a good word embedding? *arXiv preprint arXiv:1507.05523*, 2015.

[65] S. Lai, K. Liu, S. He, and J. Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.

[66] T. K. Landauer. *Latent semantic analysis*. Wiley Online Library, 2006.

[67] R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. *DAIMI Report Series*, 27(537), 1998.

[68] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[69] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference

resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, 2017.

[70] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[71] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *ACL*, 3:211–225, 2015.

[72] N. N. Liu, X. Meng, C. Liu, and Q. Yang. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *Recsys'11*, pages 37–44, 2011.

[73] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54(1):171–188, 2014.

[74] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[75] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[76] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[77] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre. Fixed-rank matrix factorizations and riemannian low-rank optimization. *Computational Statistics*, 29(3-4):591–621, 2014.

[78] A. Mukherjee, K. Chen, N. Wang, and J. Zhu. On the degrees of freedom of reduced-rank estimators in multivariate regression. *Biometrika*, 102(2):457–477, 2015.

[79] B. Murphy, P. Talukdar, and T. Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. *Proceedings of COLING 2012*, pages 1933–1950, 2012.

[80] M. Neishi, J. Sakuma, S. Tohda, S. Ishiwatari, N. Yoshinaga, and M. Toyoda. A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 99–109, 2017.

[81] D. W. Oard, J. Kim, et al. Implicit feedback for recommender systems. In *AAAI workshop on recommender systems*, pages 81–83, 1998.

[82] C. W. Omlin and C. L. Giles. Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants. *Neural Computation*, 8(4):675–696, 1996.

[83] I. Oseledets and E. Tyrtyshnikov. Tt-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[84] N. Othman, R. Faiz, and K. Smaili. A word embedding based method for ques-

tion retrieval in community question answering. In *ICNLSSP 2017-International Conference on Natural Language, Signal and Speech Processing*, 2017.

[85] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[86] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[87] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI'02*, pages 127–134, 2002.

[88] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100, 2008.

[89] S. Reddy, I. Labutov, and T. Joachims. Learning student and content embeddings for personalized lesson sequence recommendation. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pages 93–96. ACM, 2016.

[90] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[91] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[92] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820. ACM, 2010.

[93] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[95] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[96] X. Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[97] N. J. Rose. Linear algebra and its applications (gilbert strang). *SIAM Review*, 24(4):499–501, 1982.

[98] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[99] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR'02*, pages 253–260. ACM, 2002.

[100] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims. Evaluation methods for unsupervised word embeddings. In *EMNLP*, 2015.

[101] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[102] M. Seok, H.-J. Song, C.-Y. Park, J.-D. Kim, and Y.-s. Kim. Named entity recognition using word embedding as a feature. *International Journal of Software Engineering and Its Applications*, 10(2):93–104, 2016.

[103] R. Sharp, M. Surdeanu, P. Jansen, P. Clark, and M. Hammond. Creating causal embeddings for question answering with minimal supervision. *arXiv preprint arXiv:1609.08097*, 2016.

[104] Y. Shen, W. Rong, N. Jiang, B. Peng, J. Tang, and Z. Xiong. Word embedding based correlation model for question/answer matching. In *AAAI*, pages 3511–3517, 2017.

[105] S. K. Sienčnik. Adapting word2vec to named entity recognition. In *Proceedings of the 20th nordic conference of computational linguistics, nodalida 2015, may 11-13, 2015, vilnius, lithuania*, number 109, pages 239–243. Linköping University Electronic Press, 2015.

[106] I. Simova and H. Uszkoreit. Word embeddings as features for supervised coreference resolution. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 686–693, 2017.

[107] O.-M. Şulea. Recognizing textual entailment in twitter using word embeddings. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 31–35, 2017.

[108] M. Sun, F. Li, J. Lee, K. Zhou, G. Lebanon, and H. Zha. Learning multiple-question decision trees for cold-start recommendation. In *WSDM'13*, pages 445–454. ACM, 2013.

[109] A. Sysoev, I. Andrianov, and A. Khadzhiiskaia. Coreference resolution in russian: state-of-the-art approaches application and evolvement. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference Dialogue*, volume 1, pages 327–338, 2017.

[110] M. Tan, I. W. Tsang, L. Wang, B. Vandereycken, and S. J. Pan. Riemannian pursuit for big matrix recovery. In *ICML*, volume 32, pages 1539–1547, 2014.

[111] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[112] C. Udriste. *Convex functions and optimization methods on Riemannian manifolds*, volume 297. Springer Science & Business Media, 1994.

[113] B. Vandereycken. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[114] C. Vania and A. Lopez. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2016–2027, 2017.

[115] K. Vorontsov and A. Potapenko. Tutorial on probabilistic topic modeling: Additive regularization for stochastic matrix factorization. In *International Conference on Analysis of Images, Social Networks and Texts_x000D_*, pages 29–46. Springer, 2014.

[116] T. Vu, D. Q. Nguyen, M. Johnson, D. Song, and A. Willis. Search personalization

with embeddings. In *European Conference on Information Retrieval*, pages 598–604. Springer, 2017.

[117] B. H. Wang, H. T. Hui, and M. S. Leong. Global and fast receiver antenna selection for mimo systems. *Communications, IEEE Transactions on*, 58(9):2505–2510, 2010.

[118] S. Wang, Z. Chen, Q. Yan, K. Ji, L. Wang, B. Yang, and M. Conti. Deep and broad learning based detection of android malware via network traffic.

[119] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[120] B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.

[121] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*, 2016.

[122] K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *ICML'06*, pages 1081–1088. ACM, 2006.

[123] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 227–236. ACM, 2016.

[124] H. Zamani and W. B. Croft. Estimating embedding vectors for queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 123–132. ACM, 2016.

[125] X. Zhang, J. Cheng, and H. Lu. Less is more: Sparse representative based preference elicitation for cold start recommendation. In *IMCS'14*, page 117, 2014.

[126] X. Zhang, J. Cheng, S. Qiu, G. Zhu, and H. Lu. Dualds: A dual discriminative rating elicitation framework for cold start recommendation. *Knowledge-Based Systems*, 73:161–172, 2015.

[127] X. Zhang, J. Cheng, T. Yuan, B. Niu, and H. Lu. Semi-supervised discriminative preference elicitation for cold-start recommendation. In *CIKM'13*, pages 1813–1816. ACM, 2013.

[128] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[129] K. Zhao, L. Huang, and M. Ma. Textual entailment with structured attentions and composition. *arXiv preprint arXiv:1701.01126*, 2017.

[130] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *CIKM'13*, pages 1411–1420, 2013.

[131] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR'11*, pages 315–324, 2011.

[132] A. Zobnin. Rotations and interpretability of word embeddings: the case of the

russian language. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 116–128. Springer, 2017.

8 Appendix

Lemma 1. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{B} \in \mathbb{R}^{m\times n}, m > n$. *Let* $\mathbf{A}_{-i}$ *be* $n \times (m-1)$ *submatrix of* $\mathbf{A}$ *without* i*-th column and* $\mathbf{B}_{-i}$ *be* $(m-1) \times n$ *submatrix of* $\mathbf{B}$ *without* i*-th row. Then,*

$$\det(\mathbf{AB}) \leq \frac{m}{m-n} \max_i \left(\det(\mathbf{A}_{-i}\mathbf{B}_{-i})\right) \tag{97}$$

Proof. From the Cauchy-Binet formula we get

$$\det(\mathbf{AB}) = \sum_k \det \mathbf{A}[:, \mathbf{k}] \cdot \det \mathbf{B}[\mathbf{k}, :], \tag{98}$$

where $\mathbf{k} \in \mathbb{N}^n$ is a vector of n different indices. Since $\mathbf{A}_{-i}$ contains all columns of $\mathbf{A}$ except i-th column, then $\mathbf{A}[:, \mathbf{k}]$ is a submatrix of $\mathbf{A}_{-i}$ for any $i \notin \mathbf{k}$. Since $\mathbf{k}$ consists of n different numbers, we have $m - n$ different i, such that $\mathbf{A}[:, \mathbf{k}]$ is a submatrix of $\mathbf{A}_{-i}$. The same is true for the matrix $\mathbf{B}$. So get

$$\sum_{i=1}^{m} \det(\mathbf{A}_{-i}\mathbf{B}_{-i}) = (m-n) \det(\mathbf{AB}) \tag{99}$$

applying Cauchy-Binet formula to each summand. Therefore,

$$\det(\mathbf{AB}) = \frac{1}{m-n} \sum_{i=1}^{m} \det(\mathbf{A}_{-i}\mathbf{B}_{-i}), \tag{100}$$

which finishes the proof. □