NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Applied Mathematics and Informatics"

UDC 004.852

Research Project Report on the Topic:

Riemannian Optimization for Link Prediction Using Tensor Decompositions

(interim, the first stage)

Submitted by the Student:

group #БПМИ212, 3rd year of study   Sudakov Ilya Alexandrovich

Approved by the Project Supervisor:

Rakhuba Maxim Vladimirovich
Research Fellow
Faculty of Computer Science, HSE University

Moscow 2024

Contents

Abstract

It's well known, that a set of tensors of fixed rank is a riemanian manifold. This information can be applied for construction of effective methods using low-rank tensor approximation. It this project we are advised to modify methods of riemanian optimization on fixed tensor rank manifolds using l1-regularixation. As an applicaton, we have to consider recommendation systems and knowledge graph completion.

Аннотация

Известно, что множество тензоров фиксированного ранга является римановым многообразием. Эту информацию можно использовать для построения эффективных методов поиска малоранговых тензорных приближений. В этом проекте предлагается модифицировать методы римановой оптимзиации на многообразии таккеровских тензоров для использования l1-регуляризации. В качестве приложений необходимо будет рассмотреть рекомендательные системы, а также заполнение баз знаний.

Keywords

Riemannian manifold, Tucker decomposition, knowledge graph, optimization.

1 Introduction

There are a lot of information, that can be represented as subject, object and relation between them. Databases, containing that called Knowledge Graph. Obviously, it's too costly to store the whole data, so there is a need in algorithms, that are able to restore and even predict relations, using just part of existing data. Newest, state of the art models use deep non-linear networks, but they are difficult to interpret. So there are also interest in using well researched tensor factorization models.

1.1 Knowledge graph

Let's consider knowledge graph, consisting of subjects, objects and relations between them. We will denote $\mathcal{E}$ and $\mathcal{R}$ as set of all entities and relations respectively presenting in a graph, and $(e_s, r, e_o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ we will call triple. Also denote as $\mathcal{D} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ the set of all triples corresponding knowledge graph, note that $\mathcal{D}$ fully describes it. $\mathcal{D}$ can be presented as a 3-dimensional binary tensor $T \in \mathbb{R}^{n_r \times n_e \times n_e}$: $T_{e_s,r,e_o} = I[(e_s, r, e_o) \in \mathcal{D}]$ [11].

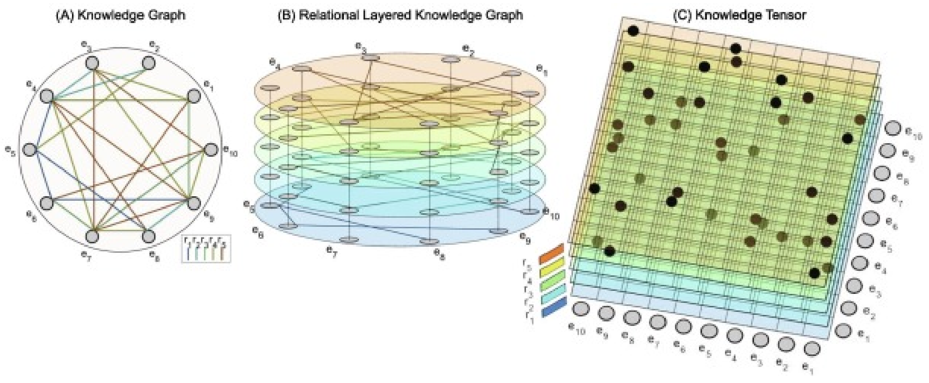

Figure 1.1: Example of Knowledge Graph and it's Tensor presentation.

As our task is to restore missed triples that not presenting in given data, we are given just part of knowledges, we denote it as $\mathcal{D}_{given}$, while set of missed triples as $\mathcal{D}_{missed}$.

1.2 Tucker decomposition

Let denote tensor presentation of the knowledge graph as T, so it contains $n_e n_r^2$ numbers, when number of entities is too big, it's impossible to do computations with the whole tensor, and to store it such way due to memory limitations. To cope with these issues, usually used different tensor factorization, in our article we use Tucker decomposition, because it was quite simple to implement certain operations that used while riemanian optimization.

Tucker decomposition [3] defined as:

$$T = W \times_1 E_s \times_2 R \times_3 E_o \tag{1}$$

where $\times_i$ indicating the tensor product along the i-th mode,
$W \in \mathbb{R}^{d_e \times d_r \times d_e}, E_s \in \mathbb{R}^{n_e \times d_e}, R \in \mathbb{R}^{n_r \times d_r}, E_o \in \mathbb{R}^{n_e \times d_e}$

It's multilinear rank defined as $mlrank(A) = (rank(A_{(1)}), rank(A_{(2)}), rank(A_{(3)}))$ where $A_{(k)}$ is matrix scan by k-th mode.

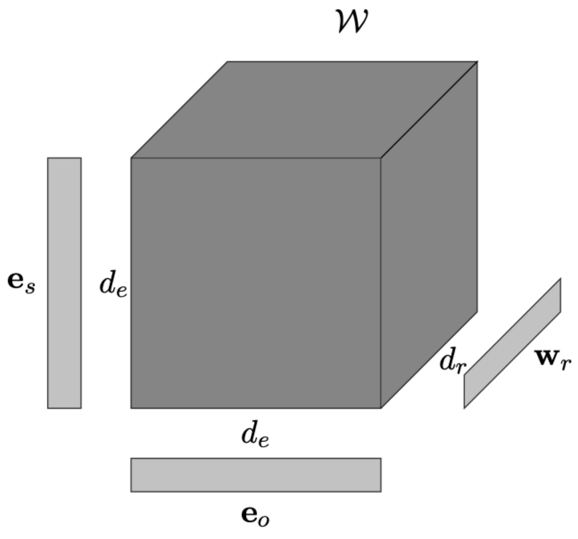

Figure 1.2: Tucker decomposition

We suppose, that T contains logits for each posible triple, that we convert to it's probability. Formally, to get prediction for triple (e_s, r, e_o), we will compute the probability of it taking T_{e_s,r,e_o}, and apply sigmoid function. To get T_{e_s,r,e_o}, we can compute it using properties of Tucker decomposition.

$$a_{i,j,k} = \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} g_{\alpha\beta\gamma} u_{i\alpha} v_{j\beta} w_{k\gamma} \tag{2}$$

it's computational difficulty: $O(r_1, r_2, r_3)$

To calculate loss, we will group data by pair of subject and relation, predict which objects has this relation with this subject. Knowing this vector of predicted probabilities and ground true, we calculate loss as negative log-likelihood of them.

Our metrics will be MRR, Hits@1, Hits@3, Hits@10. Hits@k equals number of test cases, where probability in place of true tripe has rank less then k. MMR is the mean value of inversed positions, where true triple located.

1.3 Riemanian manifold

Let's define manifold of tensors of fixed multilinear rank:

$$\mathcal{M}_k := \left\{ X \in \mathbb{R}^{n_e \times n_r \times n_r} : \text{mlrank}(X) = (d_e, d_r, d_r) \right\} \tag{3}$$

It is well known that $\mathcal{M}_k$ is a smooth manifold. When the objective function f is also smooth, problem is a smooth optimization problem, which can be solved by methods from Riemannian optimization.

1.4 Riemanian optimization

Let's consider generalization of standard unconstrained optimization, where the search space is $\mathbb{R}^n$, to optimization of a smooth objective function on a Riemannian manifold, named Riemanian optimization [10].
As we defined loss function, we can now define our optimization task:

$$\begin{array}{ll} \underset{X}{\text{minimize}} & f(X) := \mathcal{L}(\mathrm{P}_\Omega(X), \mathrm{P}_\Omega(A)) \\ \text{subject to} & X \in \mathcal{M}_k := \{X \in \mathbb{R}^{n_e \times n_r \times n_r} : \text{mlrank}(X) = (d_e, d_r, d_r)\} \end{array}$$

Now, to implement optimizatoin algorithm, we need to use some important concepts:

Riemanian gradient $f\left(X_i\right)$ is a tangent vector ξ_i corresponding to direction of fastest increase of f, but restricted to $T_{X_i}M_k$ tangent space of X_i

If we have our tangent vector, we can not just add it to X_i, as $\xi_i + X_i$ don't have to $\in M_k$. So we must be able to map $X \in T_{X_i}M_k$ to $X \in M_k$

If we will use gradient descent with momentum, accumulated gradient doesn't belong to current tangent space, so we have to move it to current point and restrict to current tangent space. It is called vector transport.

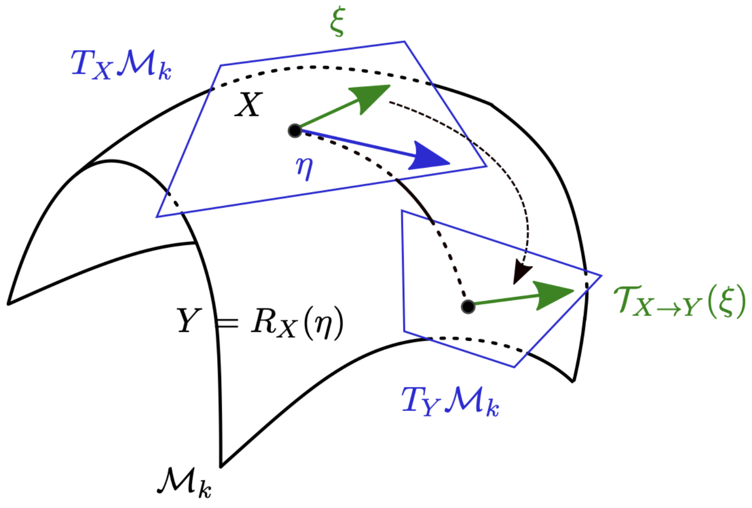

Figure 1.3: Vector transport

2 Literature review

Several linear models for link prediction have previously been proposed. Such as RESCAL [7], DistMult [12], ComplEx [9], but they have some drawbacks. Some of them are pron to overfitting, can't model assymetric relations. Also proposed non-linear models as SimplE [6], ConvE [5] and HypER [1], but they are too complicated, to interpret. Another linear model named TuckER model [2], coped with drawbacks of other linear models, it's based on Tucker decomposition of knowledge tensor. We will modify TuckER trying to beat it's result on some popular datasets.

3 Algorithm

Algorithm 1 Riemanian optimization

```
Require: initial iterate $T_0 \in \mathcal{M}_k$, tangent vector $\mu_0 = 0$
  for $i = 1, 2, ...$ do do
      Compute the gradient:
      $\xi_i \leftarrow \text{grad } f(T_i)$
      Compute a conjugate direction:
      $\mu_i \leftarrow -\xi_i + \beta_i \mathcal{T}_{T_{i-1}}(\mu_{i-1})$
      Update $T_i$:
      $T_i \leftarrow R_{T_i}(\mu_i)$
  end for
```

To program it, will use python package [8] tucker-riemopt, where implemented:

- tucker representation of tensor

- finding best approximation of tensor on manifold
- factorized representation of tangent vector
- computing of gradient scalar f at point X
- projection of X from one tangent space to another
- addition of tangent vectors
- computing norm of tangent vectors

The code for training model with such algorithm provided in github

4 Results

There are some options for model, we can use simmetric tucker decomposition, where $E_s = E_o$, as implemented in TuckER. Also there can be used different methods of gradient descent: GD, SGD and Adam with different hyperparameters and shedulers, that may improve achieved results.

For today, due to long time of training, I managed to train model on dataset 'FB15k-237' in the simplest way, using GD, constant learning rate, 300 epochs to get:
Hits @10: 0.452
Hits @3: 0.319
Hits @1: 0.214
Mean reciprocal rank: 0.293 These results a quite similar to our baseline.

5 Future work

In future research, we will experiment with different methods of optimization and train in on other datasets. Model TuckER uses regularization methods such as BatchNorm and Dropout, so we will try to implement some regularization [4], L1 as an example.

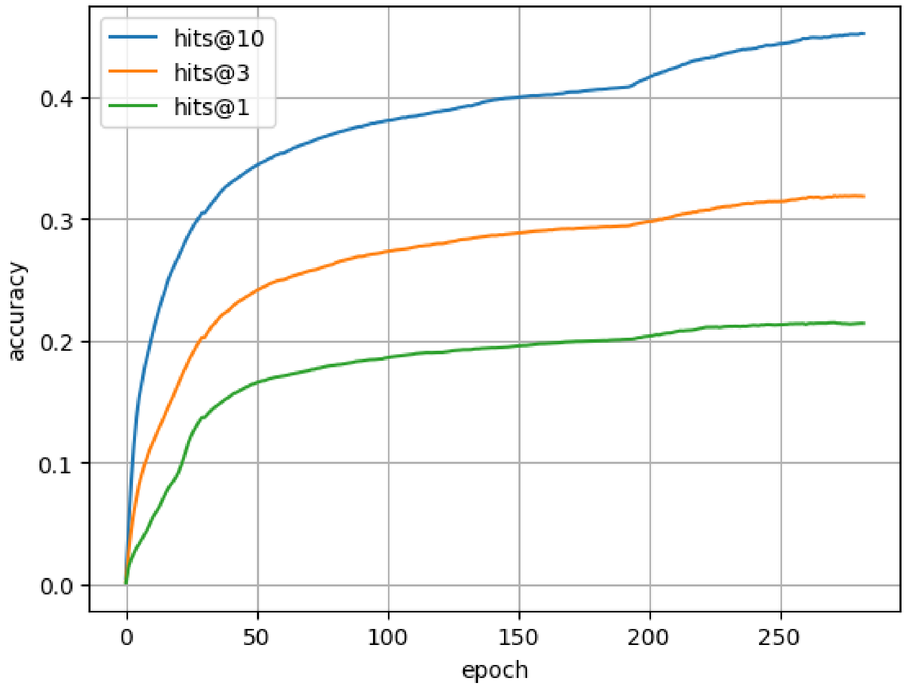

Figure 4.1: Training graph

References

[1] Ivana Balažević, Carl Allen, and Timothy M Hospedales. "Hypernetwork knowledge graph embeddings". In: *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*. Springer. 2019, pp. 553–565.

[2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. "Tucker: Tensor factorization for knowledge graph completion". In: *arXiv preprint arXiv:1901.09590* (2019).

[3] Vineet Bhatt, Sunil Kumar, and Seema Saini. "Tucker decomposition and applications". In: *Materials Today: Proceedings* 46 (2021), pp. 10787–10792.

[4] Léopold Cambier and P-A Absil. "Robust low-rank matrix completion by Riemannian optimization". In: *SIAM Journal on Scientific Computing* 38.5 (2016), S440–S460.

[5] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. "Convolutional 2d knowledge graph embeddings". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[6] Seyed Mehran Kazemi and David Poole. "Simple embedding for link prediction in knowledge graphs". In: *Advances in neural information processing systems* 31 (2018).

[7] Denis Krompaß, Maximilian Nickel, Xueyan Jiang, and Volker Tresp. "Non-negative tensor factorization with rescal". In: *Tensor Methods for Machine Learning, ECML workshop*. 2013, pp. 1–10.

[8] Ivan Peshekhonov. *tucker$_r$iemopt*. https://johanddc.github.io/tucker_riemopt/api.html. [Online; accessed 13-February-2024]. 2023.

[9] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. "Complex embeddings for simple link prediction". In: *International conference on machine learning*. PMLR. 2016, pp. 2071–2080.

[10] Bart Vandereycken. "Low-rank matrix completion by Riemannian optimization". In: *SIAM Journal on Optimization* 23.2 (2013), pp. 1214–1236.

[11] Jianing Xi, Zhaoji Miao, Longzhong Liu, Xuebing Yang, Wensheng Zhang, Qinghua Huang, and Xuelong Li. "Knowledge tensor embedding framework with association enhancement for breast ultrasound diagnosis of limited labeled samples". In: *Neurocomputing* 468 (2022), pp. 60–70.

[12] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding entities and relations for learning and inference in knowledge bases". In: *arXiv preprint arXiv:1412.6575* (2014).