Knowledge Graph Construction with External Task

Dongwoo Kim

1 Knowledge Graph Construction with External Task

Many active learning algorithms for knowledge graph construction only focus on obtaining as many positive triples as possible. This is not a crazy idea, however, it does not agree with a practical usage of knowledge graphs. For example, a content provider may want to use the knowledge graph to recommend items that fit to user's preference. This could be achieved by considering the similarity between different items in the knowledge graph based on a history of a user. If the active knowledge graph construction does not care the performance of following tasks, it may end up to discover some facts that do not lead any improvement in recommendation. Therefore, 'good' active construction algorithms should reflect the following usages of the knowledge graph.

A recent study [Zhang et al., 2016] examines how a knowledge graph can be used for a movie recommendation. Their approach resembles a co-factorisation where two different loss functions for user-movie matrix and knowledge graph tensor are used to estimate the latent embeddings of movies and users. Let $u_i \in \mathbb{R}^p$ be a latent embedding of user i, $e_j \in \mathbb{R}^p$ be a latent embedding of entity j, and $R_k \in \mathbb{R}^{p \times p}$ be a latent embedding of relation k. $X \in \mathcal{X}^{U \times E}$ is a user-movie rating matrix, and $\mathcal{G} \in [0,1]^{E \times K \times E}$ is a tensor-represented knowledge graph. The goal of co-factorisation is to minimise the following loss

$$L_{X,\mathcal{G}}(U,E,R) = L_X(U,E) + \lambda L_{\mathcal{G}}(E,R) \tag{1}$$

$$= \sum_{ij \in X} \ell_X(X_{ij}, u_i^\top e_j) + \lambda \sum_{jj'k \in \mathcal{G}} \ell_G(\mathcal{G}_{jj'k}, e_j^\top R_k e_{j'}). \tag{2}$$

A squared error is typically used as a loss for the matrix factorisation. We may consider the recommendation as a binary classification problem. In this case, we can choose an appropriate loss ℓ_X to optimise. We often measure the performance of these problems using AUC to take into account an imbalanced distribution of labels. In this case, it would be preferable to directly optimise AUC on a training set, or in practice, optimise a surrogate loss. However, the current objective optimise both losses for recommendation and knowledge graph construction, which is not an optimal choice.

One way to back propagate the loss of the classification is to design nested objectives. One possible candidate is

$$L_{X,\mathcal{G}}(U) = L_X(U) + \Omega(U) \tag{3}$$

$$= \sum_{ij \in X} \ell(X_{ij}, u_i^\top f_j(\mathcal{G})) \tag{4}$$

where f_j extract p-dimensional feature of entity j from the entire knowledge graph $\mathcal{G}$. Two questions are naturally occurring from here: 1) How to extract feature of entity j from the entire knowledge graph? 2) How does the back-propagated loss help to construct a KG?

References

[Zhang et al., 2016] Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). Collaborative Knowledge Base Embedding for Recommender Systems. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD 2016)*, pages 353–362.