Translating Embeddings for Modeling Multi-relational Data

金哲(11521043)

May 29, 2016

1 引言

多关系数据适用于有向图，实体用点表示，边表示关系，格式为($head$,$label$,$tail$)，或缩写为(h,l,t)。每个三元组表示在实体$head$ 和实体$tail$中有一个名为$label$的关系。例如社交网络分析，实体就是成员，边就是朋友关系或者社会关系的连接。再例如推荐系统中，用户是实体和产品，关系就是买、评价、查看和搜索。例如Freebase，Google Knowledge或者GeneOntology等知识库中，每个实体代表着一个抽象的概念或实际存在的一个具体的实体，关系则是两者之间事实的预测。本文的工作着眼于从知识库中（Wordnet和Freebase）对多关系数据进行建模，目标是在不需求额外知识的前提下，通过自动添加新的事实，来高效的完成建模工作。

对多关系数据进行建模：总的来说，建模过程可以归结为在实体之间抽取本地的或全局的关连模式，预测则是通过这些模式推广到该实体与其他实体的关系预测上。一个简单关系的局部性概念也许是完全结构化的，例如社交媒体中的“我的朋友的朋友是我的朋友”。但这也会随着实体的改变而改变，例如那些喜欢星球大战4的也会喜欢星球大战5，但他们也许喜欢也许不喜欢泰坦尼克。与单关系数据相比，关系数据的难度在于本地性概念也许会同时设计多个不同种类的关系和数据，因此对多关系数据建模需要更加泛化的方法来挑选合适的模式，从而能够同时处理异质化的关系。

基于单关系数据实体间的连接模式之间用户/项聚类和矩阵因式分解技术的成功，目前大多数多关系数据都被设计为一种学习隐藏属性的框架，即通过对成分的潜在表达进行学习和操作。将这些方法拓展到多关系领域，例如简单随机块模型SBM的无参数贝叶斯拓展，基于张量分解的模型等。这些方法中，绝大多数都通过贝叶斯聚类框架或者在低维度空间学习实体嵌入的基于能量的框架，来提高模型的表达性和普适性。这些模型的表达性越高，模型的复杂度也会越高，计算工作也会大幅增加。除此之外，这些模型很有可能因为模型建立难度过大导致难以选取合适的正则项导致过拟合，也有可能因为许多涉及局部最小值的非凸优化问题导致欠拟合。事实上，一个简单的模型在数个有着大量不同关系的多关系数据上与大多数复杂模型表现相当。这说明及时在复杂和异质的多关系领域中依然能找到平衡准确性和规模性的合适的模型假设。

本文提出了一种学习实体低维嵌入的基于能量的模型**TransE**。在**TransE**中，所有关系将用其在嵌入空间中的翻译来表示。如果(h,l,t) 成立，那么实体t应该与实体h加上一个基于关系l 得到的向量之和相接近。本文的主要思路正是基于层次关系在大多数知识库中非常常见，翻译也是表示这些层次关系的一种非常自然的转化方式。

2 方法概述

对于一个给定的三元组(h,l,t)的训练集S，其中两个实体$h,t \in E$（E是实体集），关系$l \in L$（L 是关系集），我们的模型将学习出实体与关系的嵌入向量。嵌入向量的取值在$\mathbb{R}^k$ 中（k是模型的超参数）。模型的基本思路是根据l标记的边来计算，例如，我们希望当(h,l,t)成立时$\mathbf{h}+\mathbf{l} \approx \mathbf{t}$（$t$应该是一个极接近$\mathbf{h}+\mathbf{l}$ 的取值），否则我们希望$\mathbf{h}+\mathbf{l}$远离$\mathbf{t}$。根据基于能量的框架，三元组的能量应该在某些消除相似性的函数d下等于$d(\mathbf{h}+\mathbf{l},\mathbf{t})$，分别采用$L_1$和$L_2$-norm。

为了学习嵌入向量，对训练集采用最小化margin-based ranking criterion的方法：

$$\mathcal{L} = \sum_{(h,l,t)\in S} \sum_{(h',l',t')\in S'_{(h,l,t)}} [\gamma + d(\mathbf{h}+\mathbf{l},\mathbf{t}) - d(\mathbf{h}'+\mathbf{l},\mathbf{t}')]_+ \tag{2.1}$$

其中$[x]_+$表示x中正的部分，$\gamma > 0$是一个边际超参数

$$S'_{h,l,t} = \{(h',l,t)|h' \in E\} \cup \{(h,l,t')|t' \in E\} \tag{2.2}$$

是一个错误三元组集，通过等式(2.2)获得，由训练集中的三元组随机替换掉head或tail中的一个而获得。损失函数(2.1)的目标是在取训练三元组的时候值要比在取错误三元组时有较小的

值。对于一个给定的实体，无论该实体是三元组的head或tail，其嵌入向量的值都是相同的。

可以用随机梯度下降对算法进行优化，对于**h**,**l**和**t**的取值，只有实体的嵌入向量的L_2-norm为1的额外限制（对**l**则没有标准化或其他限制）。这条限制十分重要，因为它通过提高实体嵌入向量的正交性，加速收敛过程。

Algorithm 1 Trans E学习过程

输入： 训练集$S = \{(h, l, t)\}$，实体集E，关系集L，边际超参数γ，嵌入向量维度k

1. **初始化** $\mathbf{l} \leftarrow$ 对每个$l \in L$，使其在$(-\frac{k}{\sqrt{6}}, \frac{k}{\sqrt{6}})$中均匀分布。
2. $\mathbf{l} \leftarrow$ 对每个$l \in L$，有$\mathbf{l}/\|\mathbf{l}\|$
3. $\mathbf{e} \leftarrow$ 对每个$e \in E$，使其在$(-\frac{k}{\sqrt{6}}, \frac{k}{\sqrt{6}})$中均匀分布。
4. **循环**
5. $\mathbf{e} \leftarrow$ 对每个$e \in E$，有$\mathbf{e}/\|\mathbf{e}\|$
6. $S_{batch} \leftarrow \text{sample}(S, b)$
7. $T_{batch} \leftarrow \emptyset$
8. **for** $(h, l, t) \in S_{batch}$ **do**
9. $(h', l, t') \leftarrow \text{sample}(S'_{h,l,t})$
10. $T_{batch} \leftarrow T_{batch} \cup \{((h, l, t), (h', l, t'))\}$
11. $(endfor)$
12. 更新嵌入向量：$\sum_{((h,l,t),(h',l,t')) \in T_{batch}} \nabla[\gamma + d(\mathbf{h} + \mathbf{l}, \mathbf{t}) - d(\mathbf{h}' + \mathbf{l}, \mathbf{t}')]_+$
13. **结束循环**

具体的算法如上所示。实体和关系的所有嵌入向量都通过一个随机过程进行初始化。在算法的每一次循环中，实体的向量都将先标准化。然后从训练集中采样数个三元组，对每一个三元组，将一个实体替换后当做训练集的反例。参数将通过一个固定学习速率的梯度下降来不断调整和更新。

3 实验结果

在实验中，尝试对文中的学习过程进行实现。选择的数据集为论文中提到的WN18数据集。实现过程如下：

首先，读取实体数据集和关系数据集，储存到map结构中。读取关系三元组，分别将（实体，关系，实体）存入各自的vector中。

其次，进行算法中的初始化部分。对每个实体向量和关系向量进行初始化：

```
for (int i=0; i<relation_num; i++){
    for (int ii=0; ii<n; ii++)
        relation_vec[i][ii] = randn(0,1.0/n,-6/sqrt(n),6/sqrt(n));
}
for (int i=0; i<entity_num; i++){
    for (int ii=0; ii<n; ii++)
        entity_vec[i][ii] = randn(0,1.0/n,-6/sqrt(n),6/sqrt(n));
    norm(entity_vec[i]);
}
```

最后，开始训练过程，设算法中每次取的batch值为100，即batches=100。

```
for (int it=0; it<itn; it++){
    for (int batch = 0; batch<batches; batch++){
```

```
        //将当前关系和实体向量赋值给tmp参数
        relation_tmp = relation_vec;
        entity_tmp = entity_vec;
        //batchsize = fb_h.size()/nbatches，为batch的数量
        for (int k=0; k<batchsize; k++){
            int i=rand_max(fb_h.size());
            //一半的三元组，对实体1进行取反例，并训练
            if (rand()%2 == 1){
                //随机从实体1的集合中选择一个不能组成三元组的实体
                while (exists[make_pair(fb_h[i],fb_r[i])].count(j)>0)
                    j=rand_max(entity_num);
                trainkb(fb_h[i],fb_l[i],fb_r[i],fb_h[i],j,fb_r[i]);
            }
            //另一半的三元组，则对实体2进行取反例，并训练
            else{
                while (exists[make_pair(j,fb_r[i])].count(fb_l[i])>0)
                    j=rand_max(entity_num);
                trainkb(fb_h[i],fb_l[i],fb_r[i],j,fb_l[i],fb_r[i]);
            }
            //根据算法要求，每次都对新的向量进行标准化
            norm(relation_tmp[fb_r[i]]);
            norm(entity_tmp[fb_h[i]]);
            norm(entity_tmp[fb_l[i]]);
            norm(entity_tmp[j]);
        }
        relation_vec = relation_tmp;
        entity_vec = entity_tmp;
    }
}
//将训练结果分别写入relation2vec和entity2vec两个文件
...
```

其中trainkb函数如下，采用了梯度下降的算法，梯度下降选择的速率为0.001。对于输入的两个三元组（一个正例三元组，一个负例三元组）。对每个三元组，用实体2的向量减去实体1的向量与关系向量之和，对向量的每一维取绝对值相加，即最基础的$d(\mathbf{h}+\mathbf{l},\mathbf{t})$实现方法，即calsum函数。

```
void trainkb(int e1_a,int e2_a,int rel_a,int e1_b,int e2_b,int rel_b){
    double sum1 = calcsum(e1_a,e2_a,rel_a);
    double sum2 = calcsum(e1_b,e2_b,rel_b);
    if (sum1>sum2){
        res += sum1-sum2;
        gradient( e1_a, e2_a, rel_a, e1_b, e2_b, rel_b);
    }
}
```

最终生成了entity2vec和relation2vec两个训练后的实体向量文件和关系向量文件。

4 小结与讨论

通过对本篇文章提出的算法TransE的学习和实现，加深了自己对关系数据集、三元组以及梯度下降等基础知识的理解。TransE的本质即把实体和关系都当做一个高维向量，通过一个距离函数来规定一个三元组的合理性，并通过在源数据集中随机选取实体创造反例，进而得到了损失函数，并通过梯度下降的方法进行训练，由于损失函数的约束条件较少，因此算法的实现相对来书较为简单。实际上，在TransE 算法提出之后，有很多学者也对TransE 进行了研究并改进或移植，提出了TransM[1]、TransH[2]等算法，还有待我进一步学习和研究。

参考文献

[1] Fan M, Zhou Q, Chang E, et al. Transition-based knowledge graph embedding with relational mapping properties[C]//Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation. 2014

[2] Wang Z, Zhang J, Feng J, et al. Knowledge Graph Embedding by Translating on Hyperplanes[C]//AAAI. 2014: 1112-1119.