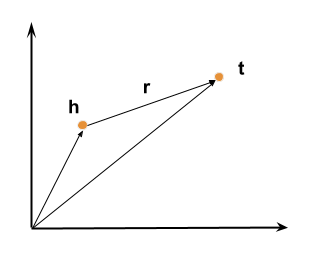
# TransE模型

最近想follow一些关于Knowledge Graph Embedding（知识图谱嵌入，可以参考这篇综述[[[1]](#endnote-1)]）的工作，而TransE模型[[[2]](#endnote-2)]应该是这个工作的代表之一。

## 1.简介

TransE模型的目标是学习出实体(entity)和关系(relation)的低维向量表示。对于一个三元组，其中和是实体，称为头实体，是尾实体，是它们的关系（也就是属性），TransE希望它们的嵌入表示（粗体表示向量）有如下关系：，也就是说要和尽可能接近，反之如果这三者不构成三元组，则要尽可能远离。用图直观表示如下：



图表1 TransE的直观表示

为此，TransE定义了三元组的“能量”，这里的“能量”（论文中说的the energy of a triplet）可以看作是损失函数，是一个距离度量方法(dissimilarity measure)，在这里就是向量之间的距离，即：

(公式1)

公式1中的指的是或距离，或者叫或范数。那么，只要最小化公式1这个损失函数就行，但为了增强区分度，TransE构造了一些反例三元组，希望反例的距离要尽可能的大，这样最终的优化目标（损失函数）就是：

(公式2)

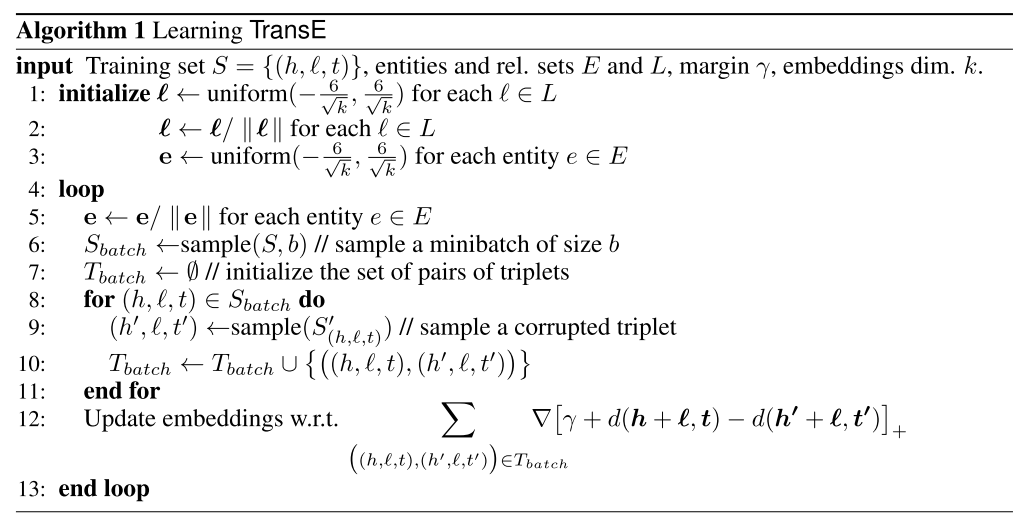
公式2中的是构造的反例三元组，构造方法是将正例三元组的头实体或尾实体替换成一个随机的实体，且二者不能同时被替换：

(公式 3)

为了防止过拟合，需要加入正则项，然后用随机梯度下降算法(stochastic gradient descent)来最小化损失函数。

## 2.代码实现

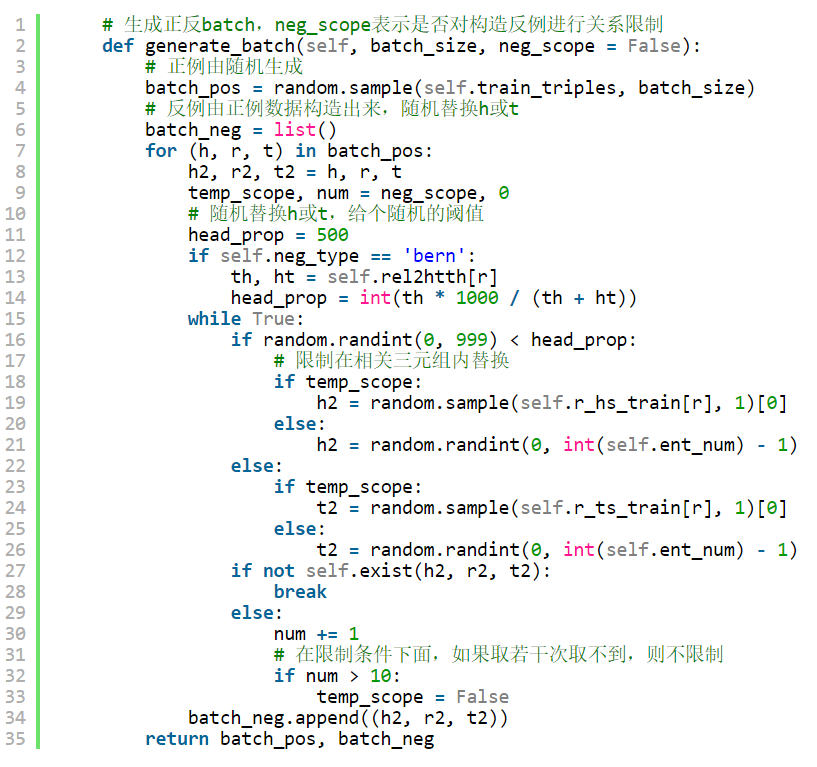
论文中对TransE算法的描述如图表2所示，官方的代码和数据在这里[[3]](#footnote-1)可以参考，除此之外，github上面也有相关实现[[4]](#footnote-2)可以参考。下面主要介绍一下TransE的TensorFlow[[5]](#footnote-3)实现，主要参考了2016年7月2-3号在北京语言大学参加的CIPS ATT1[[6]](#footnote-4)中讲解的代码，具体参见我的github[[7]](#footnote-5)。



图表2 TransE伪代码

### 2.1 反例三元组的生成

反例三元组的生成比较简单，参考下面的代码：



先随机决定替换头实体还是尾实体；然后对于具体的替换，有两种方案，一种是完全随机，任意一个实体都可以（上图代码21、26行）；另一种是有限制的替换，替换成和当前relation相关的随机实体（上图代码19、24行）。对于新生成的三元组要检查是否在正例三元组中（上图代码27行），如果不在，则反例三元组就产生成功了。

### 2.2 模型定义

模型定义的代码如下：



代码5-10行是定义变量。TensorFlow中的placeholder用于声明一个变量占位符，第一个参数是数值的类型，第二个参数shape是维度，None表示可以是任意维度。在TransE模型中，每一次迭代都需要6个输入，即正例三元组的头、尾实体和关系，以及反例三元组的头、尾实体和关系。

代码12-22行是初始化embedding。tf.Variable用于声明一个变量，tf.truncated\_normal产生正太分布的数据。在TransE模型中，每个实体都关系都会有对应的向量表示，所以embedding的维度是实体数或关系数向量维数。

代码24-29行的tf.nn.embedding\_lookup是建立实体或关系与向量表示的对应关系，相当于建立了实体或关系与其对应向量的索引，这样就可以根据实体或关系拿到对应的向量。

代码31-40行是计算损失函数。tf.reduce\_sum对张量降维，根据参数给的维数对张量元素进行求和，举个例子：

# 'x' is [[1, 1, 1]

# [1, 1, 1]]

tf.reduce\_sum(x) ==> 6

tf.reduce\_sum(x, 0) ==> [2, 2, 2]

tf.reduce\_sum(x, 1) ==> [3, 3]

tf.reduce\_sum(x, 1, keep\_dims=True) ==> [[3], [3]]

tf.reduce\_sum(x, [0, 1]) ==> 6

代码31行是计算正例三元组的“能量”，参考公式1，使用距离，即元素绝对值之和。举个例子，代码31行，tf.abs(phs + prs - pts)是计算头实体向量关系向量尾实体的结果，并将元素取绝对值，这里的phs等不是单一的向量，而是张量，张量的某一维是向量，可以将其看作是二维矩阵。tf.reduce\_sum的参数给1，也就是对每个向量求和，将元素相加。33行是计算损失函数，最大间隔的思想，正例三元组和反例三元组的“能量”之差要尽可能接近给定的间隔。tf.nn.relu是一个激活函数。

代码33-39行是正则化项，使用L2正则，即所有元素的平方之和。代码40行是最终的求解目标，由基础的损失函数加上正则项构成。

代码40行是求解，使用自适应的梯度下降方法进行求解，最小化目标函数。

总的来说，在熟悉了TensorFlow的基础上，使用TensorFlow对TransE进行实现还是比较简单的，简单的介绍就到这里。

## 3.References

1. [] 刘知远等. "知识表示学习研究进展." 计算机研究与发展 53.2(2016):247-261. [↑](#endnote-ref-1)
2. [] Bordes, Antoine, et al. "Translating Embeddings for Modeling Multi-relational Data." Neural Information Processing Systems (2013). [↑](#endnote-ref-2)
3. <https://everest.hds.utc.fr/doku.php?id=en:transe> [↑](#footnote-ref-1)
4. <https://github.com/thunlp/KG2E> [↑](#footnote-ref-2)
5. <https://www.tensorflow.org/> [↑](#footnote-ref-3)
6. <http://www.cipsc.org.cn/att1/> [↑](#footnote-ref-4)
7. <https://github.com/sunzequn/Knowledge_Graph_Embedding/tree/master/TransE> [↑](#footnote-ref-5)