前面一篇MIT的学习笔记介绍了统计语言模型，但传统的统计语言模型有一些缺点：

1. 由于维度灾难(特别是离散变量)，在高维下，数据的稀缺性，导致统计语言模型存在很多为零的条件概率，传统的统计语言模型也花费了很大的精力来处理零概率问题，比如现在有很多的平滑、插值、回退等方法用来解决该问题。
2. 语言模型的参数个数随阶数呈指数增长，所以一般情况统计语言模型使用的阶数不会很高，这样n-gram语言模型无法建模更远的关系。
3. n-gram无法建模出多个相似词的关系。比如在训练集中有这样的句子，The cat is walking in the bedroom，但用n-gram测试时，遇到 A dog was running in a room这个句子，并不会因为两个句子非常相似而让该句子的概率变高。

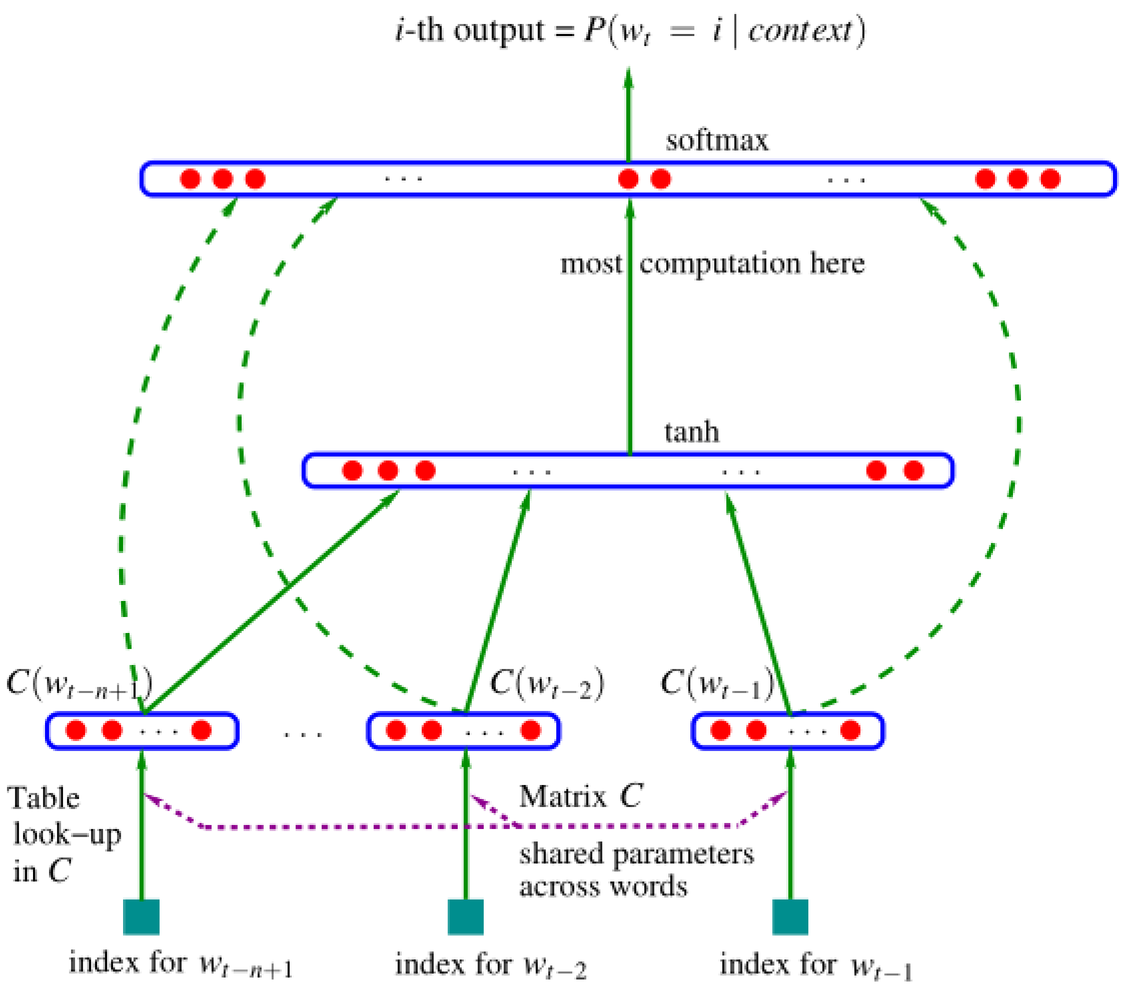
这篇文章使训练得到的模型比n-gram能够建模更远的关系，并且考虑到了词的相似性，一些相似词获得了自然的平滑。前者是因为神经网络的结构可以使得，后者是因为使用了词向量。

词向量

下面先介绍本文中的词向量(distributed representation for words)，本文中单词的特征向量是把单词映射为一个具有一定维度实数向量(比如50,100维，这里记为m)，每一个词都和一个特征向量相关联，词向量初始化可以为随机的数，文中介绍也可以使用一些先验知识来初始化词向量，随着训练的结束，词向量便获得了。词向量的引入把n-gram的离散空间转换为连续空间，并且两个相似的词之间它们的词向量也相似，所以当训练完毕时，一个句子和其所有相似的句子都获得了概率。而把词映射到词向量是作为整个网络的第一层的，这个在后面会看到。

神经模型

神经网络的模型如图：



先从整体来看一下模型，其中概率函数表示如下：

$$f(w_t,\cdots,w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$$

在这个模型中，它可以分解为两部分：

1. 一个 |V|×m映射矩阵C，每一行表示某个单词的特征向量，是m维，共|V|列，即|V|个单词都有对应的特征向量在C中
2. 将条件概率函数表示如下：

$$f(i,w_{t-1},\cdots,w_{t-n+1}) = g(i,C(w_{t-1}),\cdots,C(w_{t-n+1}))$$

即该函数是来估计 $\hat{P}(w_t = i \mid w_1^{t-1})$，其中i有|V|种取值。如果把该网络的参数记作ω，那么整个模型的参数为 θ = (C,ω)。我们要做的就是在训练集上使下面的似然函数最大化：

$$L = \frac{1}{T}\sum_t \log f(w_t,w_{t-1},\cdots,w_{t-n+1};\theta) + R(\theta)$$

下面看一下网络里面的每一层，输出层是用softmax做归一化，这个能够保证概率和为1，如下：

$$\hat{P}(w_t|w_{t-1},\cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

其中yi是未规范化的概率，yi即是输出层的输入，它的计算如下：

$$y = b + Wx + U\tanh(d + Hx)$$

其中Wx表示输入层与输出层有直接联系，如果不要这个链接，直接设置W为0即可，b是输出层的偏置向量，d是隐层的偏置向量，里面的x即是单词到特征向量的映射，计算如下：

$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}))$$

记录隐层的神经元个数为h，那么整个模型的参数可以细化为θ = (b, d, W, U, H, C)。下面计算一下整个模型的参数个数：

输出层偏置向量：|V|

隐层偏置向量：h

隐层到输出层之间的权重矩阵：|V|*h

输入层到输出层的权重矩阵：|V|*(n-1)*m

输入层到隐层的权重矩阵：h*(n-1)*m

特征矩阵C：|V|*m

所有的参数和计算如下：

$$\begin{aligned}
&|V| + h + |V|h + |V|\times(n-1)m + h\times(n-1)m + |V|\times m \\
&= |V| + h + |V|h + |V|mn - |V|m + hnm - hm + |V|m \\
&= |V| + |V|h + |V|mn + h + hnm - hm \\
&= |V|(1 + h + mn) + h(1 + m(n-1))
\end{aligned}$$

从这个计算式，我们看到网络的参数个数与V是呈线性相关的，与n也是呈线性相关的，而n-gram是指数级递增，这使得论文中的模型可以更有利于建立更远关系的语言模型。

模型训练使用的是梯度下降的方式来更新参数：

$$\theta \leftarrow \theta + \varepsilon \frac{\partial \log \hat{P}(w_t|w_{t-1},\cdots w_{t-n+1})}{\partial \theta}$$

该模型虽然是与V，n线性相关的，但是计算量比n-gram要大得多，由于在输出层的计算量最大，所以论文后面讨论了如何并行使得训练更快。

模型的算法

下面介绍的是模型的算法，因为在我实现的是非并行的算法，这里我略去了并行那块，并且省略了输入层到输出层的直接连接，直接简化了算法。

前向算法：

(a) 将单词映射为特征向量，该特征向量作为输入层的输出

$$x(k) \leftarrow C(w_{t-k}),$$
$$x = (x(1), x(2), \cdots, x(n-1))$$

(b)计算隐层的输入向量o，隐层的输出层向量a

$$o \leftarrow d + Hx$$
$$a \leftarrow \tanh(o)$$

(c)计算输出层输入向量y，输出层输出向量p

$s \leftarrow 0$

j在 $|V|$ 中循环：

1. $y_j \leftarrow b_j + a^T U_j$

2. $p_j \leftarrow e^{y_j}$

3. $s \leftarrow s + p_j$

j在 $|V|$ 中循环(归一化概率)：

$$p_j \leftarrow \frac{p_j}{s}$$

(d)计算L = logpwt，准备进行反向更新

反向更新：

(a)计算及更新输出层相关量

$$\frac{\partial L}{\partial a}=0,\frac{\partial L}{\partial x}=0$$

令j在V中循环：

$$\text{i. } \frac{\partial L}{\partial y_j} \leftarrow 1_{j==w_t} - p_j$$

$$\text{ii. } b_j \leftarrow b_j + \varepsilon \frac{\partial L}{\partial y_j}$$

$$\frac{\partial L}{\partial a} \leftarrow \frac{\partial L}{\partial a} + \frac{\partial L}{\partial y_j} U_j$$

$$U_j \leftarrow U_j + \varepsilon \frac{\partial L}{\partial y_j} a$$

(b)计算及更新隐层

For k=0 to h-1

$$\frac{\partial L}{\partial o_k} \leftarrow (1-a_k^2)\frac{\partial L}{\partial a_k}$$

$$\frac{\partial L}{\partial x} \leftarrow \frac{\partial L}{\partial x} + H' \frac{\partial L}{\partial o}$$

$$d \leftarrow d + \varepsilon \frac{\partial L}{\partial o}$$

$$H \leftarrow H + \varepsilon \frac{\partial L}{\partial o} x'$$

(c)计算及更新输入层

For k=1 to n-1

$$C(w_{t-k}) \leftarrow C(w_{t-k}) + \varepsilon \frac{\partial L}{\partial x(k)}$$

公式推导

这里将上面反向更新利用的梯度上升进行公式推导

用梯度上升对输出层偏置向量b_j进行更新：

$$b_j \leftarrow b_j + \varepsilon \frac{\partial L}{\partial b_j}$$

在前向算法中有：$y_j \leftarrow b_j + aU_j$

所以 $\frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial y_j} \cdot \frac{\partial y_j}{\partial b_j} = \frac{\partial L}{\partial y_j} \cdot 1 = \frac{\partial L}{\partial y_j}$

下面算 $\frac{\partial L}{\partial y_j}$：

$$L = \log_e p_{w_t} = \log_e \frac{e^{y_{j==w_t}}}{\sum_j e^{y_j}} = \log_e e^{y_{j==w_t}} - \log_e \sum_j e^{y_j} = y_{j==w_t} - \log_e \sum_j e^{y_j}$$

$$\frac{\partial L}{\partial y_j} = 1_{j==w_t} - \frac{1}{\sum_j e^{y_j}} \cdot e^{y_j} = \begin{cases} 1 - p_j & \text{当} j == w_t \\ -p_j & \text{否则} \end{cases}$$

所以得到如下调整公式：

1. $\frac{\partial L}{\partial y_j} = 1_{j==w_t} - p_j$

2. $b_j \leftarrow b_j + \varepsilon \frac{\partial L}{\partial y_j}$

下面对隐层到输出层的权值矩阵U进行更新：

$$U_j \leftarrow U_j + \varepsilon \frac{\partial L}{\partial U_j} \text{(梯度上升)}$$

在前向算法中：$y_j \leftarrow b_j + a^T U_j$

$$\frac{\partial L}{\partial U_j} = \frac{\partial L}{\partial y_j} \cdot \frac{\partial y_j}{\partial U_j} = \frac{\partial L}{\partial y_j} \cdot a^T$$

代入进去得到更新公式：

$$U_j \leftarrow U_j + \varepsilon \frac{\partial L}{\partial y_j} a$$

下面计算$\frac{\partial L}{\partial a}$,因为后面计算会使用到该梯度向量：

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial y_j} \cdot \frac{\partial y_j}{\partial a} = \frac{\partial L}{\partial y_j} \cdot U_j$$

注意这里对每一个不同的j, $\frac{\partial L}{\partial y_j} \cdot U_j$是不一样的,即$\frac{\partial L}{\partial a}$是不同的,论文中采取全部求和的形式求得$\frac{\partial L}{\partial a}$,即：

$$\frac{\partial L}{\partial a} \leftarrow \frac{\partial L}{\partial a} + \frac{\partial L}{\partial y_j} \cdot U_j$$

下面推导对隐层参数的更新，首先要计算$\frac{\partial L}{\partial o}$, 因为下面计算需要该梯度向量：

$\frac{\partial L}{\partial o_k} = \frac{\partial L}{\partial a_k} \cdot \frac{\partial a_k}{\partial o_k}$,所以下面需要计算$\frac{\partial a_k}{\partial o_k}$

由于在前向算法中$a_k = \tanh(o_k)$(tanh是双曲正切函数)

$$\frac{\partial a_k}{\partial o_k} = \frac{\partial}{\partial o_k} \cdot \frac{\sinh(o_k)}{\cosh(o_k)} = \frac{\partial}{\partial o_k} \cdot \frac{\sin h'(o_k) \cdot \cosh(o_k) - \cos h'(o_k) \cdot \sinh(o_k)}{\cosh^2(o_k)}$$

由于$\sinh x = \frac{e^x - e^{-x}}{2}$ $\cosh x = \frac{e^x + e^{-x}}{2}$,所以有

$\sin h' x = \cosh x$ $\cos h' x = \sinh x$,故

$$\frac{\partial a_k}{\partial o_k} = \frac{\cosh^2(o_k) - \sinh^2(o_k)}{\cosh^2(o_k)} = 1 - \frac{\sinh^2(o_k)}{\cosh^2(o_k)} = 1 - \tanh^2(o_k) = 1 - a_k^2$$

故 $\frac{\partial L}{\partial o_k} = (1 - a_k^2) \cdot \frac{\partial L}{\partial a_k}$

下面计算 $\frac{\partial L}{\partial x}$，因为后面需要用到该梯度向量：

由前向算法 $o \leftarrow d + Hx$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial x} = H^T \cdot \frac{\partial L}{\partial o}$$

同样的，如果考虑上输入到输出层的连接，需要对 $\frac{\partial L}{\partial x}$ 进行求和,即：

$$\frac{\partial L}{\partial x} \leftarrow \frac{\partial L}{\partial x} + H^T \cdot \frac{\partial L}{\partial o}$$

对于隐层的偏置向量 d 进行更新：

$$d_j \leftarrow d_j + \varepsilon \cdot \frac{\partial L}{\partial d_j} \text{(梯度上升)}$$

$$\frac{\partial L}{\partial d_j} = \frac{\partial L}{\partial o_j} \cdot \frac{\partial o_j}{\partial d_j} = \frac{\partial L}{\partial o_j} \cdot 1$$

所以 d_j 的更新公式为：

$$d_j \leftarrow d_j + \varepsilon \cdot \frac{\partial L}{\partial o_j}$$

对于输入层到隐层的矩阵参数 H 更新如下：

$$H \leftarrow H + \varepsilon \cdot \frac{\partial L}{\partial H} \text{(梯度上升)}$$

$$\frac{\partial L}{\partial H} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial H} = \frac{\partial L}{\partial o} \cdot x^T$$

所以 H 的更新公式为：

$$H \leftarrow H + \varepsilon \cdot \frac{\partial L}{\partial o} \cdot x^T$$

最后对更新词向量矩阵进行推导：

$$C(w_{t-k}) \leftarrow C(w_{t-k}) + \varepsilon \cdot \frac{\partial L}{\partial C(w_{t-k})} \text{(梯度上升)}$$

这里 $\frac{\partial L}{\partial C(w_{t-k})} = \frac{\partial L}{\partial x(k)}$，表示 $\frac{\partial L}{\partial x}$ 向量中第k块，每一块长度为m，即更新公式为：

$$C(w_{t-k}) \leftarrow C(w_{t-k}) + \varepsilon \cdot \frac{\partial L}{\partial x(k)}$$