

原

《word2vec Parameter Learning Explained》论文学习笔记

置顶

2018年04月26日 18:12:57

lanyu_01

阅读量 5578

CSDN

版权声明：本文为博主原创文章，未经博主允许不得转载。https://blog.csdn.net/lanyu_01/article/details/80097350

目录：

- 1 Continuous Bag-of-Word Model
- 1.1 One-word context
- Update equation for hidden→output weights
- Update equation for input→hidden weights
- 1.2 Multi-word context
- 2 Skip-Gram Model
- 3 Optimizing Computational Efficiency
- 3.1 Hierarchical Softmax
- 3.2 Negative Sampling

由于word2vec模型学习生成的词向量表示方法能够携带句子的语义信息（semantic meanings），因此非常适用于多种NLP任务。这篇论文详细地推导和解释了word2vec模型的参数更新公式，包括：**CBOW**（continuous bag-of-word）模型和**SG**（skip-gram）模型，以及两种术：**hierarchical softmax** 和 **negative sampling**。

1 Continuous Bag-of-Word Model

1.1 One-word context

我们从CBOW模型的最简单版本开始介绍——One-word context。即我们假定context（预测目标单词的上下文信息）只有一个单词，也就是说One-context 模型是在只要一个上下文单词（one context word）的情况下来预测一个目标单词（one target word）的。（注：对于初学神经网络的读者，建议后，在回到此处阅读下文）。

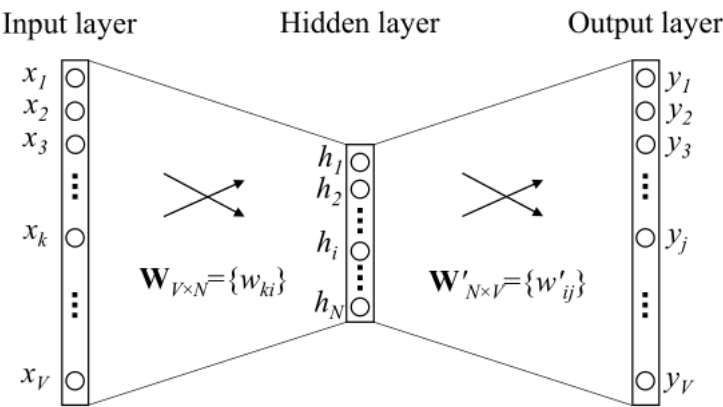


Figure 1: A simple CBOW model with only one word in the context

如图1描述的就是One-word context定义之下的神经网络模型。这里我们假设文本词汇量的大小为V,隐藏层的大小为N，相邻层的神经元是全连接的。个用**one-hot**方式编码的单词向量 $\mathbf{x} = (x_1, \dots, x_V)$ ，其中只有一个 x_i 为1，其余均为0。从输入层到隐藏层的权重值可以用一个 $V \times N$ 维的矩阵 \mathbf{W} 来表示，即

$$W = \begin{pmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1N} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2N} \\ \dots & \dots & \dots & \dots \\ \omega_{V1} & \omega_{V2} & \dots & \omega_{VN} \end{pmatrix}$$

其中 W 矩阵的每一行代表的是一个与输入层相关的单词的 N 维向量表示形式 \mathbf{v}_ω 。那么假设我们给定了一个输入单词 (a context), 其单词向量的第 k 个 $x_k = 1$, 其余均为0, 则有

$$\mathbf{h} = \mathbf{W}^T \mathbf{x} = \mathbf{W}_{(k, \bullet)}^T x_k = \mathbf{v}_{\omega_I}^T$$

从 (1) 式我们可以看出, h 向量完全是从 W 矩阵第 k 行复制过来的 (同 \mathbf{v}_{ω_I} 均为 N 维向量)。 \mathbf{v}_{ω_I} 即为输入单词 ω_I 的一种向量表示 (其实就是**输入向量**会提到)。

分析完输入层到隐藏层之后, 我们再看隐藏层到输出层, 同样连接权重用一个新的 $N \times V$ 矩阵 $\mathbf{W}' = \{\omega'_{ij}\}$ 来表示如下:

$$\mathbf{W}' = \begin{pmatrix} \omega'_{11} & \omega'_{12} & \dots & \omega'_{1V} \\ \omega'_{21} & \omega'_{22} & \dots & \omega'_{2V} \\ \dots & \dots & \dots & \dots \\ \omega'_{N1} & \omega'_{N2} & \dots & \omega'_{NV} \end{pmatrix}$$

通过这些权重, 我们可以为词表中的每一个单词都计算出一个得分 μ_j

$$\mu_j = \mathbf{v}'_{\omega_j}{}^T \mathbf{h}$$

其中, \mathbf{v}'_{ω_j} 即为矩阵 \mathbf{W}' 的第 j 列向量 (也是 N 维向量, 其实就是单词 w 的输出向量, 我们后面会提到)。

经过以上讨论之后, 我们可以使用一种对数-线性分类模型softmax函数来计算单词的后验分布 (是多项式分布)

$$p(\omega_j | \omega_I) = y_j = \frac{\exp(\mu_j)}{\sum_{j'=1}^V \exp(\mu_{j'})}$$

其中, y_j 表示输出层第 j 个神经元的输出值。将 (1) 式和 (2) 式代入 (3) 式我们可以得到:

$$p(\omega_j | \omega_I) = \frac{\exp(\mathbf{v}'_{\omega_j}{}^T \mathbf{v}_{\omega_I})}{\sum_{j'=1}^V \exp(\mathbf{v}'_{\omega_{j'}}{}^T \mathbf{v}_{\omega_I})}$$

注意: 正如前文所述, \mathbf{v}_ω 和 \mathbf{v}'_ω 是单词的两种向量表示形式。其中 \mathbf{v}_ω 实际上是权重矩阵 \mathbf{W} (input->hidden) 的某一行向量, \mathbf{v}'_ω 则是权重矩阵 \mathbf{W}' (hidden->output) 的某一列向量。我们将 \mathbf{v}_ω 和 \mathbf{v}'_ω 分别称为 “**输入向量** (input vector) ” 和 “**输出向量** (output vector) ” (二者均为 N 维向量)。

Update equation for hidden→output weights

接下来让我们推到权重矩阵的更新公式, 尽管在实际的计算过程中这样做是不切实际的 (我们在之后再谈)。

在我们推导hidden→output权重的更新公式的过程中, 需要用到神经网络的反向传播算法, 对这部分内容不熟悉的读者可以参考附录A的内容。

由以上描述可知, 该模型训练的目标就是求公式 (4) 的最大值。公式 (4) 代表的就是给定上下文信息 (这里为一个单词 ω_I) 以及其权重矩阵的情况实际输出单词 (即上下文信息的中心词 ω_O) 的条件概率。

$$\begin{aligned} \max p(\omega_O | \omega_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= \mu_{j^*} - \log \sum_{j'=1}^V \exp(\mu_{j'}) := -E \end{aligned}$$

其中, $E = -\log p(\omega_O | \omega_I)$ 为该模型的损失函数 (我们需要找出它的最小值), μ_{j^*} 的表示方式由公式 (2) 而来, j^* 则为实际输出单词的索引下标。我们注意到该损失函数特殊情形下的**交叉熵**计算。

现在我们开始推导从隐藏层到输出层的权重矩阵在模型训练过程中的参数更新公式。首先我们对损失函数 $E = -\log p(\omega_O | \omega_I)$ 求关于得分 μ_j 的偏导为:

$$\frac{\partial E}{\partial \mu_j} = y_j - t_j := e_j$$

其中, $t_j = 1 (j = j^*)$, 即当且仅当输出层的第 j 个神经元为真实的输出单词时 t_j 的取值为1。接下来我们根据链式法则求出损失函数 E 关于矩阵 W' 导数为:

$$\frac{\partial E}{\partial \omega'_{ij}} = \frac{\partial E}{\partial \mu_j} \cdot \frac{\partial \mu_j}{\partial \omega'_{ij}} = e_j \cdot h_i$$

因此，采用随机梯度下降算法（SGD），我们最终得到了隐藏层到输出层（hidden→output）权重的更新公式如下：

$$\omega'_{ij}^{(new)} = \omega'_{ij}^{(old)} - \eta \cdot e_j \cdot h_i$$

or

$$\mathbf{v}'_{\omega_j}{}^{(new)} = \mathbf{v}'_{\omega_j}{}^{(old)} - \eta \cdot e_j \cdot \mathbf{h} \text{ for } j = 1, 2, \dots, V.$$

其中， $\eta > 0$ 为参数更新的学习速率； $e_j = y_j - t_j$ ； h_i 为隐藏层的第i个神经单元； \mathbf{v}'_{ω_j} 为 ω_j 的输出向量。

由公式（11）我们可以看出：在更新权重参数的过程中，我们需要检查词汇表中的每一个单词，计算出它的输出概率 y_j ，并与期望输出 t_j （取值只能为0或1）进行比较。比较过程如下：

- 1) 如果 $y_j > t_j$ （“overestimating”），那么就从向量 \mathbf{v}'_{ω_j} 中减去隐藏向量 \mathbf{h} 的一部分（例如 \mathbf{v}_{ω_I} ），这样向量 \mathbf{v}'_{ω_j} 就会与向量 \mathbf{v}_{ω_I} 相差更远。
- 2) 如果 $y_j < t_j$ （“underestimating”，这种情况只有在 $t_j = 1$ 时，才会发生，此时 $\omega_j = \omega_O$ ），则将隐藏向量 \mathbf{h} 的一部分加入 \mathbf{v}'_{ω_O} ，使得 \mathbf{v}'_{ω_O} 与 \mathbf{h} 更接近。
- 3) 如果 y_j 与 t_j 非常接近，则此时 $e_j = y_j - t_j$ 由于（公式（8））非常接近于0，故更新参数基本上没什么变化。

这里需要再次提醒的是： \mathbf{v}_{ω} 和 \mathbf{v}'_{ω} 是单词 ω 的两种不同的向量表示形式。

Update equation for input→hidden weights

在介绍完hidden→output的权重矩阵更新公式之后，我们接着介绍input→hidden的权重矩阵 W 的更新过程。我们继续对损失函数 E 求关于隐藏层 h 的偏导数：

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^V \frac{\partial E}{\partial \mu_j} \cdot \frac{\partial \mu_j}{\partial h_i} = \sum_{j=1}^V e_j \cdot \omega'_{ij} := EH_i$$

其中 h_i 为隐藏层第i个神经单元的输出； μ_j 在公式（2）中已经定义，表示输出层第j个神经单元的输入； $e_j = y_j - t_j$ 为输出层第j个单词的预测误差。是一个N维向量，它的每一个元素代表的是词汇表中的每个单词的预测误差 e_j 与 ω'_{ij} 在j=1到V上的乘积之和。

接下来，我们需要求出损失函数 E 关于权重矩阵 W 的偏导数。首先，分解公式（1），我们知道隐藏层激活单元的输出 h_i 是输入层 \mathbf{x} 与权重的线性组合：

$$h_i = \sum_{k=1}^V x_k \cdot \omega_{ki}$$

因此对于权重矩阵 W 的每一个元素，我们求关于 E 的偏导数，得到：

$$\frac{\partial E}{\partial \omega_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial \omega_{ki}} = EH_i \cdot x_k$$

因此我们利用张量乘积的方式，便可得到：

$$\frac{\partial E}{\partial W} = \mathbf{x} \otimes EH = \mathbf{x}EH^T$$

我们再次得到了一个 $N \times V$ 的矩阵。由于 \mathbf{x} 向量只有一个非0元素，因此 $\frac{\partial E}{\partial W}$ 只有一行是N维非0向量 EH^T ，因此矩阵 W 的更新公式为：

$$\mathbf{v}_{\omega_I}{}^{(new)} = \mathbf{v}_{\omega_I}{}^{(old)} - \eta \cdot EH^T$$

其中 \mathbf{v}_{ω_I} 是矩阵 W 的其中一行，是唯一的上下文单词（context word）的“输入向量”，也是矩阵 W 唯一的导数非0的行向量。除了 \mathbf{v}_{ω_I} 以外，矩阵 W 在参数更新迭代过程中都会保持不变（因为其导数为0）。

与矩阵 W' 的更新过程相似，对于公式（16），我们分析如下：

- 1) 如果过高地估计了某个单词 ω_j 作为最终输出单词的概率（即： $y_j > t_j$ ），则上下文单词 ω_I （context word）的输入向量与单词 ω_j 的输出向量在更新过程中会越来越接近。
- 2) 如果相反，某个单词 ω_j 作为最终输出单词的概率被低估（即： $y_j < t_j$ ），则单词 ω_I 的输入向量与单词 ω_j 的输出向量在更新过程中会越来越接近。
- 3) 如果对于单词 ω_I 的概率预测是准确的，则对于单词的输入向量在更新过程中几乎保持不变。

因此，上下文单词 ω_I （context word）的输入向量的更新取决于词汇表中所有单词的预测误差。预测误差越大，则该单词对于上下文单词的输入向量影响越大。

在介绍完One-word context的CBOW模型之后，我们接着介绍multi-word context下的CBOW模型。

1.2 Multi-word context

根据字面意思我们就可以看出，基于multi-word context的CBOW模型就是利用多个上下文单词来推测中心单词target word的一种模型。其结构如图

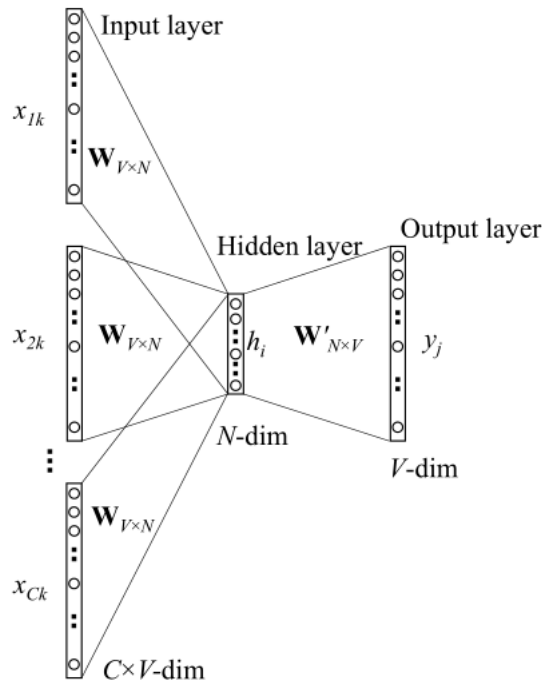


Figure 2: Continuous bag-of-words model https://blog.csdn.net/lanyu_01

其隐藏层的输出值的计算过程为：首先将输入的上下文单词（context words）的向量叠加起来并取其平均值，接着与input→hidden的权重矩阵相乘结果，公式如下：

$$\begin{aligned} \mathbf{h} &= \frac{1}{C} \mathbf{W}^T (\mathbf{x}_1 + \mathbf{x}_2 + \dots + \mathbf{x}_C) \\ &= \frac{1}{C} (\mathbf{v}_{\omega_1} + \mathbf{v}_{\omega_2} + \dots + \mathbf{v}_{\omega_C})^T \end{aligned}$$

其中 C 为上下文单词的个数， $\omega_1, \dots, \omega_C$ 为上下文单词， \mathbf{v}_{ω} 为单词 ω 的输入向量。损失函数为：

$$\begin{aligned} E &= -\log p(\omega_O | \omega_{I,1}, \dots, \omega_{I,C}) \\ &= -\mu_{j^*} + \log \sum_{j=1}^V \exp(\mu_{j'}) \\ &= -\mathbf{v}_{\omega_O}'^T \cdot \mathbf{h} + \log \sum_{j=1}^V \exp(\mathbf{v}_{\omega_j}'^T \cdot \mathbf{h}) \end{aligned}$$

同样，由hidden→output的权重更新公式与one-word-context模型下的一模一样，即类似于公式 (11)，我们直接写在下面：

$$\mathbf{v}_{\omega_j}'^{(new)} = \mathbf{v}_{\omega_j}'^{(old)} - \eta \cdot e_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, \dots, V$$

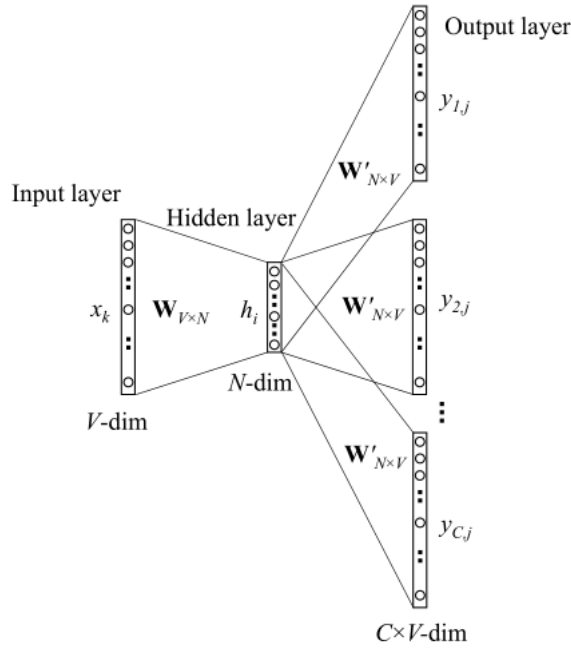
由input→hidden 的权重矩阵更新公式与公式 (16) 类似，只不过现在我们需要对每一个上下文单词 $\omega_{I,c}$ 都执行如下更新公式：

$$\mathbf{v}_{\omega_{I,c}}'^{(new)} = \mathbf{v}_{\omega_{I,c}}'^{(old)} - \frac{1}{C} \cdot \eta \cdot E \mathbf{H}^T \quad \text{for } c = 1, 2, \dots, C.$$

其中 $\mathbf{v}_{\omega_{I,c}}$ 为上下文context中第 c 个单词的输入向量； η 为正学习速率； $E \mathbf{H} = \frac{\partial E}{\partial \mathbf{h}_i}$ 由公式 (12) 给出。

2 Skip-Gram Model

与CBOW模型正好相反，Skip-Gram模型是根据中心单词（target word）来预测其上下文信息（context words）。如图3所示，为Skip-Gram模型图。

Figure 3: The skip-gram model. https://blog.csdn.net/lanyu_01

我们仍然使用 \mathbf{v}_{ω_I} 来表示输入层上唯一的那个单词的**输入向量**，因此，我们对于隐藏层的输出值 \mathbf{h} 的计算公式与第一节公式 (1) 相同，表示如下：

$$\mathbf{h} = \mathbf{W}_{(k,\bullet)}^T := \mathbf{v}_{\omega_I}$$

公式 (24) 显示： \mathbf{h} 向量其实就是 input \rightarrow hidden 权重矩阵 \mathbf{W} 的某一行结合输入单词 ω_I 的向量拷贝。在输出层，与 CBOW 模型的输出为单个多项式分布，SG 模型在输出层输出了 C 个多项式分布。每个输出都使用相同的 hidden \rightarrow output 矩阵计算：

$$p(\omega_{c,j} = \omega_{O,c} | \omega_I) = y_{c,j} = \frac{\exp(\mu_{c,j})}{\sum_{j'=1}^V \exp(\mu_{j'})}$$

其中， $\omega_{c,j}$ 表示输出层的第 c 个 panel 的第 j 个单词（何为 panel？就是输出层的表示每个上下文单词的神经元的组合，图中一种有 C 个 context words，所以总共有 C 个 panel）；表示的是输出上下文单词（output context words）的第 c 个单词； ω_I 是唯一的输入单词； $y_{c,j}$ 为输出层的第 c 个 panel 上的第 j 个神经元的概率输出值； $\mu_{c,j}$ 为输出层第 c 个 panel 的第 j 个神经元的输入值；**由于输出层的所有 panels 共享同一权重矩阵 \mathbf{W}'** ，因此：

$$\mu_{c,j} = \mu_j = \mathbf{v}'_{\omega_j} \cdot \mathbf{h}, \text{ for } c = 1, 2, \dots, C$$

其中， \mathbf{v}'_{ω_j} 为词汇表第 j 个单词 ω_j 的输出向量；同样，它也是取自于 hidden \rightarrow output 权重矩阵 \mathbf{W}' 的一列。

SG 模型参数更新公式的推导过程与 one-word-context 模型的推导过程大体上一样。这里我们将损失函数变为：

$$\begin{aligned} E &= -\log p(\omega_{O,1}, \omega_{O,2}, \dots, \omega_{O,C} | \omega_I) \\ &= -\log \prod_{c=1}^C \frac{\exp(\mu_{c,j_c^*})}{\sum_{j'=1}^V \exp(\mu_{j'})} \\ &= -\sum_{c=1}^C \mu_{j_c^*} + C \cdot \log \sum_{j'=1}^V \exp(\mu_{j'}) \end{aligned}$$

其中， j_c^* 为第 c 个输出层输出的上下文单词在词汇表中的真实索引。

在得到损失函数 E 之后，我们对输出层的每一个 panel 上的所有激活单元的输入值 $\mu_{c,j}$ 均求其关于 E 的偏导数，得：

$$\frac{\partial E}{\partial \mu_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j}$$

其中 $e_{c,j}$ 为输出层神经元的预测误差，与公式 (8) 类似。为了简化符号，我们定义一个 V 维的向量 $EI = \{EI_1, \dots, EI_V\}$ 作为所有上下文单词的预测误差 EI_j 用公式定义如下：

$$EI_j = \sum_{c=1}^C e_{c,j}$$

接下来，我们计算 hidden \rightarrow output 权重矩阵 \mathbf{W}' 关于 E 的偏导数为：

$$\frac{\partial E}{\partial \omega'_{ij}} = \sum_{c=1}^C \frac{\partial E}{\partial \mu_{c,j}} \cdot \frac{\partial \mu_{c,j}}{\partial \omega'_{ij}} = EI_j \cdot h_i$$

这样，我们就得到了hidden→output权重矩阵 \mathbf{W}' 的参数更新公式为：

$$\omega'_{ij}^{(new)} = \omega'_{ij}^{(old)} - \eta \cdot EI_j \cdot h_i$$

或者

$$\mathbf{v}'_{\omega_j}^{(new)} = \mathbf{v}'_{\omega_j}^{(old)} - \eta \cdot EI_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, \dots, V.$$

上述参数更新公式的直观概念理解与上文公式 (11) 无二，除了一点就是：输出层的预测误差的计算是基于多个上下文单词context words,而不是单个target word;需要注意的是对于每一个训练样本，我们都要利用该参数更新公式来更新hidden→output权重矩阵 \mathbf{W}' 的每个元素。

同样，对于input→hidden权重矩阵 \mathbf{W} 的参数更新公式的推导过程，除了考虑要将预测误差 e_j 替换为 EI_j 外，其他也与上文公式 (12) 到公式 (16) 我们直接给出更新公式：

$$\mathbf{v}_{\omega_l}^{(new)} = \mathbf{v}_{\omega_l}^{(old)} - \eta \cdot EH^T$$

其中， EH 是一个 N 维向量，组成该向量的每一个元素可以用如下公式表示：

$$EH_i = \sum_{j=1}^V EI_j \cdot \omega'_{ij}$$

公式 (36) 的直观理解与公式 (16) 类似，这里不作描述。

3 Optimizing Computational Efficiency

总结以上的模型介绍，我们发现所有模型的词汇表中的每个单词都存在两个向量表示形式：输入向量 \mathbf{v}_ω 与输出向量 \mathbf{v}'_ω 。对于输入向量的参数学习成本并不同于输出向量的学习成本代价是非常昂贵的。根据更新公式 (22) 和 (23)，我们可以发现，为了更新输出向量 \mathbf{v}'_ω ，对于每一个训练样例，我们必须选中所有的单词 ω_j ，计算出它们的输入值 μ_j 、概率预测值 y_j （或者SG模型中的 $y_{c,j}$ ），预测误差 e_j （或者SG模型的 EI_j ）。最终使用预测误差更新它们 \mathbf{v}'_j 。

显然，对于每一个训练样例都要对所有单词计算上述各值，其成本是昂贵的。特别是对于大型的词汇表，这种计算方式是不切实际的。因此为了解决这的方式是限制必须要更新的训练样例的输出向量的数目。一种有效的实现方式就是：hierarchical softmax（分层softmax），另一种实现通过采样的们在下章节来讨论。

这两种方法都是通过只优化输出向量更新的计算过程来实现的。在我们的公式推导过程中，我们关心的有三个值：（1） E ，新的目标函数；（2） $\frac{\partial E}{\partial \mathbf{v}'_\omega}$ 输出向量的更新公式；（3） $\frac{\partial E}{\partial \mathbf{h}}$ ，为了更新输入向量反向传播的预测误差的加权和。

3.1 Hierarchical Softmax

Hierarchical softmax 是一种有效的计算 softmax 的方式。该模型使用一棵二叉树来表示词汇表中的所有单词。所有的 V 个单词都在二叉树的叶节点点一共有 $V - 1$ 个。对于每个叶子节点，从根节点root到该叶子节点只有一条路径；这条路径用来评估用该叶子节点代表该叶子节点上单词的概率值。构如图4所示：

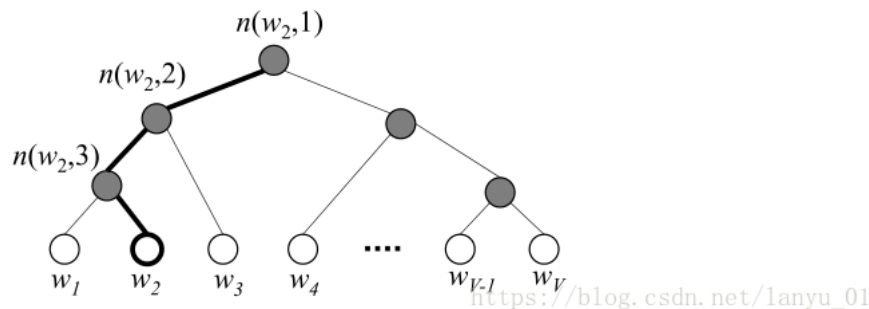


Figure 4: An example binary tree for the hierarchical softmax model.

其中白色的树节点代表的是词汇表中的单词，灰色节点为内部节点。图中高亮显示的是一条从根节点到 w_2 的路径。该条路径的长度为 $L(w_2) = 4$ 。 n_j 根节点到单词 ω 的路径上的第 j 个节点。

在hierarchical softmax模型中，所有的词汇单词没有输出向量表示形式。不同的是，二叉树的每一个内部节点都有一个输出向量 $\mathbf{v}'_{n(\omega,j)}$ 。因此一个单词的概率计算公式定义如下：

$$p(\omega = \omega_O) = \prod_{j=1}^{L(\omega)-1} \sigma \left(\left[[n(\omega, j+1) = ch(n(\omega, j))] \right] \cdot \mathbf{v}'_{n(\omega, j)}^T \mathbf{h} \right)$$

其中， $ch(n)$ 为节点 n 的左孩子节点； $\mathbf{v}'_{n(\omega, j)}$ 是内部节点 $n(\omega, j)$ 的向量表示（输出向量）； \mathbf{h} 是隐藏层的输出值（在SG模型中， $\mathbf{h} = \mathbf{v}_{\omega_I}$ ；而在CBOW模型 $\mathbf{h} = \frac{1}{C} \sum_{c=1}^C \mathbf{v}_{\omega_c}$ ）； $[[x]]$ 是一种特殊的函数定义如下：

$$[[x]] = \begin{cases} 1 & \text{if } x \text{ is true} \\ -1, & \text{otherwise} \end{cases}$$

接下来，我们通过一个直观地例子来理解公式（37）。如图4所示，假定我们需要计算单词 ω_2 作为输出单词的概率。我们将这个概率定义为从根节点 ω_1 到叶节点 ω_2 的概率。则在每一个内部节点（包括根节点），我们都需要确定其路径指向左孩子节点还是右孩子节点的概率。我们将经过内部节点的路径的概率定义为：

$$p(n, left) = \sigma(\mathbf{v}'_n{}^T \cdot \mathbf{h})$$

我们可以看出，公式（39）的值取决于内部节点的向量表示 \mathbf{v}'_n 和隐藏层的输出值 \mathbf{h} （ \mathbf{h} 的值取决于输入单词的向量表示）。显然，内部节点的路径指向左可以表示为：

$$p(n, right) = 1 - \sigma(\mathbf{v}'_n{}^T \cdot \mathbf{h}) = \sigma(-\mathbf{v}'_n{}^T \cdot \mathbf{h})$$

顺着图4中从根节点到单词 ω_2 节点的路径，我们可以计算出 ω_2 作为输出单词的概率为：

$$\begin{aligned} p(\omega_2 = \omega_O) &= p(n(\omega_2, 1), left) \cdot p(n(\omega_2, 2), left) \cdot p(n(\omega_2, 3), right) \\ &= \sigma(\mathbf{v}'_{n(\omega_2, 1)}{}^T \mathbf{h}) \cdot \sigma(\mathbf{v}'_{n(\omega_2, 2)}{}^T \mathbf{h}) \cdot \sigma(-\mathbf{v}'_{n(\omega_2, 3)}{}^T \mathbf{h}). \end{aligned}$$

不难证明

$$\sum_{i=1}^V p(\omega_i = \omega_O) = 1$$

现在我们开始推导内部节点的向量表示形式的参数更新公式。为了简化步骤，我们首先考虑单个上下文单词（one-word context）的模型。为了简化公式，我们定义子公式的简化符号如下：

$$[[\cdot]] := [[n(\omega, j+1) = ch(n(\omega, j))]]$$

$$\mathbf{v}'_j := \mathbf{v}'_{n_{\omega, j}}$$

则，给定一个训练样例，其误差函数我们可以定义如下：

$$E = -\log p(\omega = \omega_O | \omega_I) = -\sum_{j=1}^{L(\omega)-1} \log \sigma([[\cdot]] \mathbf{v}'_j{}^T \mathbf{h})$$

对于误差函数 E ，我们取其关于 $\mathbf{v}'_j \mathbf{h}$ 的偏导数，得：

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} &= \left(\sigma([[\cdot]] \mathbf{v}'_j{}^T \mathbf{h}) - 1 \right) [[\cdot]] \\ &= \begin{cases} \sigma(\mathbf{v}'_j{}^T \mathbf{h}) - 1, & [[\cdot]] = 1 \\ \sigma(\mathbf{v}'_j{}^T \mathbf{h}), & [[\cdot]] = -1 \end{cases} \\ &= \sigma(\mathbf{v}'_j{}^T \mathbf{h}) - t_j \end{aligned}$$

其中 $t_j = 1$ （如果 $[[\cdot]] = 1$ ）或者 $t_j = 0$ （如果 $[[\cdot]] = -1$ ）。

紧接着我们计算内部节点 $n(\omega, j)$ 的向量表示 \mathbf{v}'_j 关于函数 E 的偏导数，得：

$$\frac{\partial E}{\partial \mathbf{v}'_j} = \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{v}'_j} = \left(\sigma(\mathbf{v}'_j{}^T \mathbf{h}) - t_j \right) \cdot \mathbf{h}$$

因此，更新公式为：

$$\mathbf{v}'_j^{(new)} = \mathbf{v}'_j^{(old)} - \eta \left(\sigma(\mathbf{v}'_j{}^T \mathbf{h}) - t_j \right) \cdot \mathbf{h}, \text{ for } j = 1, 2, \dots, L(\omega) - 1$$

我们可以将 $\sigma(\mathbf{v}'_j{}^T \mathbf{h}) - t_j$ 理解为内部节点 $n(\omega, j)$ 的预测误差。每一个内部节点的“任务”就是预测其随机游走路径是指向左孩子节点还是指向右孩子； $t_j = 1$ 意味着节点 $n(\omega, j)$ 的路径指向左孩子节点； $t_j = 0$ 则表示指向右孩子节点。 $\sigma(\mathbf{v}'_j{}^T \mathbf{h})$ 是预测结果。对于一个训练实例，如果内部节点的预测真实值，则它的向量表示 \mathbf{v}'_j 的更新变化很小；否则 \mathbf{v}'_j 向量指向一个适当的方向是使该实例的预测误差逐渐减小。以上更新公式既能应用于CBOW模型SG模型。当在SG模型中使用该更新公式时，我们需要对 C 个output context words的每一个单词都重复此更新过程。

为了使用反向传播该预测误差来学习训练input→hidden的权重，我们对误差函数 E 求关于隐藏层输出值的偏导数，如下：

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{h}} &= \sum_{j=1}^{L(\omega)-1} \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{h}} \\ &= \sum_{j=1}^{L(\omega)-1} \left(\sigma(\mathbf{v}'_j \mathbf{h}) - t_j \right) \cdot \mathbf{v}'_j \\ &:= EH\end{aligned}$$

接下来我们根据公式 (23) 便可以获得CBOW模型输入向量的更新公式。对于SG模型，我们需要计算上下文信息中的每个单词的EH值,并将EH值的和 (35), 就能够得到输入向量的更新公式。

从以上更新公式我们可以看出：经过改进的模型Hierarchical softmax的每个训练样例的每个上下文单词的计算复杂度从 $O(V)$ 降为 $O(\log(V))$ 级别。参数几乎没有什么改变（内部节点对应V-1维向量，而原始模型的单词的输出向量维数为V）。

3.2 Negative Sampling

Negative Sampling模型的思想比hierarchical softmax模型更直接了当,即：在每次迭代的过程中，有大量的输出向量需要更新，为了解决这一困难，sampling提出了只更新其中一部分输出向量的解决方案。

显然，最终需要输出的上下文单词（正样本）在采样的过程中应该保留下来并更新，同时我们需要采集一些单词作为负样本（因此称为“negative sampling”）。在采样的过程中，我们可以任意选择一种概率分布。我们将这种概率分布称为“噪声分布”（the noise distribution），用 $P_n(\omega)$ 来表示。我们可以以某种较好的分布。

在 word2vec 中，我们无需使用一种能够产生良好定义的后验多项式分布的负采样形式，本文作者证明了使用下面简单的训练目标函数能够产生可靠的 word embeddings:

$$E = -\log \sigma(\mathbf{v}'_{\omega_O} \mathbf{h}) - \sum_{\omega_j \in W_{neg}} \log \sigma(-\mathbf{v}'_{\omega_j} \mathbf{h})$$

其中 ω_O 是输出单词（the positive sample）， \mathbf{v}'_{ω_O} 是输出向量； \mathbf{h} 是隐藏层的输出值：在CBOW模型中 $\mathbf{h} = \frac{1}{C} \sum_{c=1}^C \mathbf{v}_{\omega_c}$ ，在SG模型中 $\mathbf{h} = \mathbf{v}_{\omega_I}$ ； $W_{neg} = \{\omega_j | j = 1, \dots, K\}$ 是基于分布 $P_n(\omega)$ 采样的一系列单词。

为了获得negative sampling模型的词向量更新公式，我们首先计算E关于输出单元 ω_j 的输入 $\mathbf{v}'_{\omega_j} \mathbf{h}$ 的偏导数：

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{v}'_{\omega_j} \mathbf{h}} &= \begin{cases} \sigma(\mathbf{v}'_{\omega_j} \mathbf{h}) - 1, & \text{if } \omega_j = \omega_O \\ \sigma(\mathbf{v}'_{\omega_j} \mathbf{h}), & \text{if } \omega_j \in W_{neg} \end{cases} \\ &= \sigma(\mathbf{v}'_{\omega_j} \mathbf{h}) - t_j\end{aligned}$$

其中，当 ω_j 是一个正样本时， $t_j = 1$ ；否则 $t_j = 0$ 。接下来我们计算E关于单词 ω_j 的输出向量的偏导数：

$$\frac{\partial E}{\partial \mathbf{v}'_{\omega_j}} = \frac{\partial E}{\partial \mathbf{v}'_{\omega_j} \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_{\omega_j} \mathbf{h}}{\partial \mathbf{v}'_{\omega_j}} = \left(\sigma(\mathbf{v}'_{\omega_j} \mathbf{h}) - t_j \right) \mathbf{h}$$

因此输出向量的更新公式为：

$$\mathbf{v}'_{\omega_j}^{(new)} = \mathbf{v}'_{\omega_j}^{(old)} - \eta \left(\sigma(\mathbf{v}'_{\omega_j} \mathbf{h}) - t_j \right) \mathbf{h}$$

negative sampling的关键就是公式 (59) 的更新过程只应用于词汇表的子集 $\{\omega_j | \omega_j \in \{\omega_O\} \cup W_{neg}\}$ ，而非应用于整个词汇表。

以上更新公式 (59) 的直观理解与公式 (11) 类似。公式 (59) 对两种应用模型CBOW和SG都适用。对于SG模型，我们每次更新一个上下文单词。

接着利用反向传播机制，计算E关于隐藏层输出 \mathbf{h} 的偏导数：

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{h}} &= \sum_{\omega_j \in \{\omega_O\} \cup W_{neg}} \frac{\partial E}{\partial \mathbf{v}'_{\omega_j} \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_{\omega_j} \mathbf{h}}{\partial \mathbf{h}} \\ &= \sum_{\omega_j \in \{\omega_O\} \cup W_{neg}} \left(\sigma(\mathbf{v}'_{\omega_j} \mathbf{h}) - t_j \right) \mathbf{v}'_{\omega_j} := EH\end{aligned}$$

将EH代入公式 (23)，我们就可以得到CBOW模型关于输入向量的更新公式；对于SG模型，我们需要计算出每个上下文单词的EH值，将EH值的和代入，就能够得到其输入向量的更新公式。


PS:明人不说暗话，阁下留下一赞可好？



想对作者说点什么

 jiangnangogogo: 楼主你好, 公式8那里总是推导不出来。。请问可以指点一下嘛? 谢谢啦 (2周前 #6楼)

 洛虞: 楼主, 请问, 你有相关的实现代码吗? (2个月前 #5楼) [查看回复\(1\)](#)

 __Nerv: 赞赞赞 (7个月前 #4楼) [查看回复\(1\)](#)

[登录](#) [查看 11 条热评](#) 

word2vec 中的数学原理详解 (一) 目录和前言

阅读数 19万+

word2vec是Google于2013年开源推出的一个用于获取wordvector的工具包, 它简单、高效, 因此引... [博文](#) 来自: [peghoty](#)

word2vec 过程理解&词向量的获取

阅读数 6001

网上有很多这方面的资源, 详细各位都能够对于word2vec了解了大概, 这里只讲讲个人的理解, 目的... [博文](#) 来自: [BVL的博客](#)

Word2vec Parameter Learning Explained 论文学习笔记

阅读数 3929

原始论文: <http://www-personal.umich.edu/~ronxin/pdf/w2vexp.pdf>之前学习Word2vec时, 脱离... [博文](#) 来自: [li8630的专栏](#)

【深度学习&分布式】Parameter Server 详解

阅读数 3万+

MXNet是李沐和陈天奇等各路英雄豪杰打造的开源深度学习框架(最近不能更火了), 其中最吸引我的... [博文](#) 来自: [AutoVision \(b...](#)

negative sampling负采样和nce loss

阅读数 294

negativesampling负采样和nce loss一、Noisecontrastiveestimation (NCE) 语言模型中, 在最后一... [博文](#) 来自: [weixin_40901...](#)

word2vec Parameter Learning Explained学习笔记

阅读数 410

目录原因: 看了几篇提及CBOW (ContinuousBag-of-Word) 的综述, 都没直接看懂。综述中都指向... [博文](#) 来自: [boywaiter的专...](#)

word2vec Parameter Learning Explained

阅读数 356

由word2vec获得的词向量代表可以捕获语义信息1.ContinuousBag-of-Word模型现在只考虑仅一个... [博文](#) 来自: [qq_27009517...](#)

word2vec Parameter Learning Explained笔记

阅读数 14

目录1.CBOW模型2.SkipGram模型3.Hierarchicalsoftmax4.NegativeSampling前言, 下面公式多次用... [博文](#) 来自: [liunianhuakai...](#)

word2vec的学习资料

阅读数 66

一个全面的了解 <https://blog.csdn.net/itplus/article/details/37969519>和XinRong的论文: 『word... [博文](#) 来自: [Gin077的专栏](#)

论文《word2vec Parameter Learning Explained》笔记

阅读数 73

###XinRong的论文《word2vecParameterLearningExplained》CBOWSkipgramHsoftmaxnegativ... [博文](#) 来自: [Amy_mmm的博客](#)

node2vec论文阅读

阅读数 8208

本文要阅读的论文来自数据挖掘领域的顶级会议KDD-2016.本篇文章是一种Graphrepresentataion得... [博文](#) 来自: [myofficials的...](#)

word2vec 模型的详细数学推导和直观理解

阅读数 1505

word2vec作为一个优秀的用于产生词向量开源工具, 在自然语言处理和计算机视觉领域有着很多应用... [博文](#) 来自: [hsiffish的博客](#)

云摘录 | Word2Vec 作者Tomas Mikolov 的三篇代表作解析

阅读数 9527

本文来源于公众号paperweekly 谈到了word2vec作者的三篇论文: 1、EfficientEstimationofWordRe... [博文](#) 来自: [素质云笔记/Re...](#)

[强化学习]OpenAI官方发布: 强化学习中的关键论文

阅读数 728

【导读】OpenAI在教学资源合集SpinningUp中发布了强化学习中的关键论文, 列举了强化学习不同领... [博文](#) 来自: [sinat_3348796...](#)

Word2vec原理及其Python实现

阅读数 1067

目录一、为什么需要WordEmbedding二、Word2vec原理1、CBOW模型2、Skip-gram模型三、行业... [博文](#) 来自: [huacha_的博客](#)

秒懂词向量Word2vec的本质

阅读数 8111

[NLP]秒懂词向量Word2vec的本质穆文4个月前转自我的公众号:『数据挖掘机养成记』1.引子大家好我... [博文](#) 来自: [凌风探梅的专栏](#)

word2vec 数学原理

阅读数 150

word2vec是Google于2013年推出的一个用于获取词向量的开源工具包。我们在项目中多次使用到它... [博文](#) 来自: [ipaomi的博客](#)

word2vec

阅读数 117

在自然语言处理中常常使用预训练的word2vec,来自GoogleNews-vectors-negative300.bin, 下面函... [博文](#) 来自: [芦金宇的专栏](#)