word2vec Parameter Learning Explained

Abstract

Mikolov等人的 word2vec 模型及其应用。近两年来引起了极大的关注。通过word2vec模型学习的单词的向量表示已经被证明具有语义特性，并且在各种 NLP 任务中发挥作用。随着越来越多的研究人员愿意尝试 word2vec 或类似的技术，我注意到目前还缺乏全面详细地解释 word embedding 模型的参数学习过程的材料，从而阻碍了非神经网络专家的研究人员理解这些模型的工作机制。

本问详细推导和解释了 word2vec 模型的参数更新方程，包括原始的 continuous-bag-of-word (CBOW) 和 skip-gram (SG)模型，以及包括 hierarchical softmax 和 negative sampling 在内的高级优化技术。此外，还提供了梯度方程的直观解释以及数学推导。

在附录中，提供了有关神经元网络和反向传播基础知识的综述。 我还创建了一个交互式演示wevi，以促进对模型的直观理解。1

1 Continuous Bag-of-Word Model

1.1 One-word context

我们从Mikolov等人介绍的 continuous bag-of-word model (CBOW) 的最简单版本开始。(2013A)。我们假设每个上下文只考虑一个单词，这意味着模型将在给定一个上下文单词的情况下预测一个目标单词，这类似于 bigram 语法模型。对于刚接触神经网络的读者，建议您先阅读附录A，快速复习一下重要的概念和术语，然后再继续学习。

图1 显示了简化上下文定义2下的网络模型。在我们的设置中，词汇表大小为 V ，隐藏层大小为 N 。相邻层上的单元完全连接在一起。输入是一个 one-hot 向量，这意味着对于给定的输入上下文词，V 单元 $\{x_1, \cdots, x_V\}$ 中只有一个将是 1，所有其他单元都是 0。

输入层和输出层之间的权重可以用 $V \times N$ 矩阵 $\mathbf{W}$ 来表示。$\mathbf{W}$ 的每行是输入层的关联词的 N 维向量表示 $\mathbf{v}_w$。形式化来讲，$\mathbf{W}$ 的第 i 行是 $\mathbf{v}_w^T$ 。给定一个上下文(一个单词)，假设 $x_k = 1$ 且 $x_{k'} = 0$, $k^{'} \neq k$, 我们有

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, \qquad (1)$$

这实质上是将 $\mathbf{W}$ 的第 k 行复制到 $\mathbf{h}$ 。$\mathbf{v}_{w_I}$ 是输入词 w_I 的向量表表示。这意味着隐藏层单元的链接(激活)函数是简单地线性关系(即，将其输入的加权和直接传递到下一层)。

从隐藏层到输出层，有一个不同的权重矩阵 $\mathbf{W}^{'} = w^{'}_{ij}$，它是一个 $N \times V$ 矩阵。使用这些权重，我们可以为词典的每个词计算分数 u_j，

$$u_j = {\mathbf{v}^{'}_{w_j}}^T\mathbf{h}, \qquad (2)$$

其中 $\mathbf{v}^{'}_{w_j}$ 是矩阵 $\mathbf{W}^{'}$ 的 第 j 列。然后，我们可以使用 log-linear classification model-softmax 来得到单词的后验分布，它是一个多项式分布。

$$p(w_j|w_I) = y_j = \frac{exp(u_j)}{\sum^V_{j^{'}=1} exp(u_{j^{'}})} \qquad (3)$$

其中，y_j 是输出层中单元 j 的输出。将(1) (2)代入 (3) ，有

$$p(w_j|w_I) = \frac{exp({\mathbf{v}^{'}_{w_j}}^T\mathbf{v}_{w_I})}{\sum^V_{j^{'}=1} exp({\mathbf{v}^{'}_{w_{j^{'}}}}^T\mathbf{v}_{w_I})} \qquad (4)$$

请注意，$\mathbf{v}_w$ 和 $\mathbf{v}^{'}_w$ 是词 w 的两个向量表示形式。$\mathbf{v}_w$ 来自 $\mathbf{W}$ 的行，$\mathbf{W}$ 是 input → hidden 的权重矩阵，$\mathbf{v}^{'}_w$ 来自 $\mathbf{W}^{'}$ 的列，$\mathbf{W}^{'}$ 是 hidden → output 的权重矩阵。在随后的分析中，我们将 $\mathbf{v}_w$ 称为**“输入向量”**，将 $\mathbf{v}^{'}_w$ 称为词 w 的**“输出向量”**。

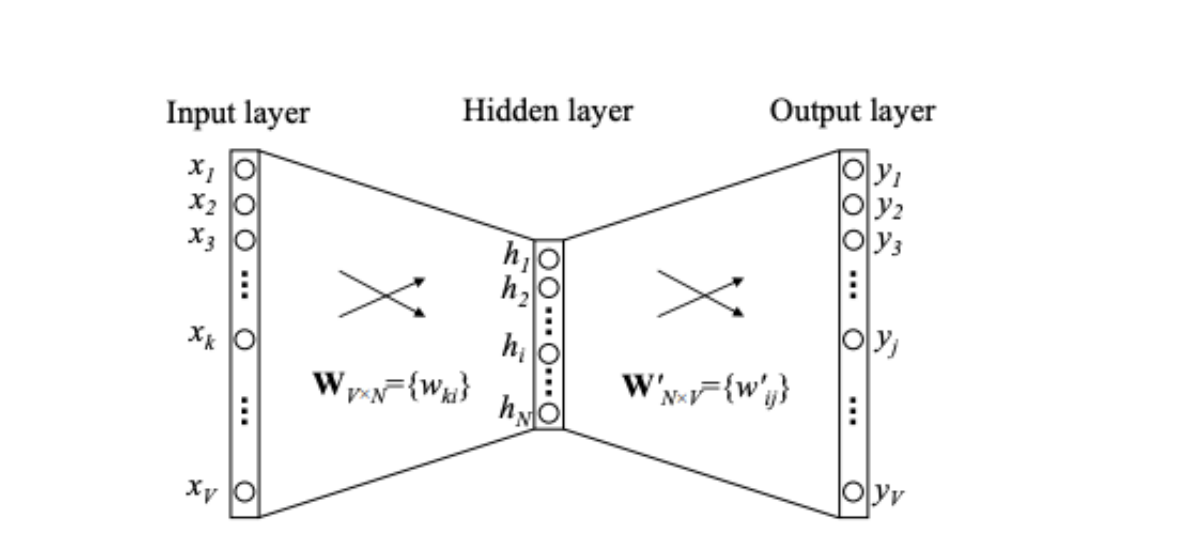

Figure 1: 上下文中只有一个词的简化 CBOW 模型

Update equation for hidden→output weights

现在让我们推导出该模型的权重更新方程。虽然真的去计算的话是不切实际的(下面解释)，但我们进行推导是为了深入了解这个没有应用任何优化技巧的原始模型。有关反向传播的基础知识的回顾，请参见附录A。

训练目标(对于一个训练样本)是最大化(4)，给定输入上下文单词 w_I 的权值，观察实际输出单词 w_O (其在输出层的索引为 j^*)的条件概率。

$$\begin{aligned} max\ p(w_O|w_I) &= max\ y_j^* \qquad (5) \\ &= max\ log\ y_{j^*} \qquad (6) \\ &= u_{j^*} - log \sum_{j'=1}^{V} exp(u_{j'}) := -E, \qquad (7) \end{aligned}$$

其中 $E = -log\ p(w_O|w_i)$ 是我们的损失函数(我们希望最小化 E)，j^* 是输出层中实际输出词的索引。**注意，该损失函数可以理解为两个概率分布之间的交叉熵测量的特例。**

现在让我们推导隐藏层和输出层之间权值的更新方程。求 E 对 j 个输出层输入 u_j 的导数，得到

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \qquad (8)$$

其中 $t_j = 1(j = j^*)$，即当第 j 个单位为实际输出词时，t_j 将仅为 1，否则 $t_j = 0$。请注意，该导数只是输出层的预测误差 e_j。(注：y_j 是 $log \sum_{j'=1}^{V} exp(u_{j'})$ 对 u_j 的导数 t_j 是 u_{j^*} 对 u_j 的导数)

接下来，我们对 w'_{ij} 求导，以获得 hidden → output 权重的梯度。

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \qquad (9)$$

因此，利用随机梯度下降，我们得到了 hidden →output 权重的权重更新方程：

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \eta \cdot e_j \cdot h_i. \qquad (10)$$

或者

$$\mathbf{v}'^{(new)}_{w_j} = \mathbf{v}'^{(old)}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad for\ j = 1, 2, \cdots, V. \qquad (11)$$

其中 $\eta > 0$ 是学习率，$e_j = y_j - t_j$，h_i 是隐藏层中的 i 个单位；$\mathbf{v}'_{w_j}$ 是 w_j 的输出层向量。

注意，这个更新公式意味着我们必须遍历词典中的每个单词，检查其输出概率 y_j，并将 y_j 与其预期输出 t_j (0 或 1)进行比较。

- 如果 $y_j > t_j$ (“overestimating”)，则从 $\mathbf{v}'_{w_j}$ 中减去一定比例的隐藏层向量 $\mathbf{h}$ (即 $\mathbf{v}_{w_I}$)，从而使 $\mathbf{v}'_{w_j}$ 远离 $\mathbf{v}_{w_I}$ ；
- 如果 $y_j < t_j$ (“underestimating”，只有当 $t_j = 1$，即 $w_j = w_O$ 时才是正确的)，我们将一定比例的 $\mathbf{h}$ 加到 $\mathbf{v}'_{w_O}$ 上，从而使 $\mathbf{v}'_{w_O}$ 接近 $\mathbf{v}_{w_I}$ 。
- 如果 y_j 与 t_j 非常接近，则根据更新公式，权重几乎不会发生变化。再次注意，$\mathbf{v}_w$ (输入向量)和 $\mathbf{v}'_w$ (输出向量)是单词 w 的两个不同的向量表示。

这里，当我说“更近”或“更远”时，我的意思是用内积而不是欧几里得作为距离的度量。

(注：此处的 $\mathbf{v}_{w_I}$ 与 $\mathbf{v}'_{w_O}$ 更加接近，两者的内积越大，否则越小，对于负样本同理)

Update equation for input→hidden weights

获得 $\mathbf{W}'$ 的更新方程后，我们现在开始转到 $\mathbf{W}$。我们对 E 求对隐藏层的输出的导数，得到

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathbf{EH}_i \qquad (12)$$

其中 h_i 是隐藏层的 i 个单元的输出；

u_j 在 (2) 中定义，是输出层中 j 个单元的输入；

$e_j = y_j - t_j$ 是输出层中 j 个词的预测误差。

$\mathbf{EH}$ 是一个 N 维向量，是词典中所有词的输出向量的预测误差加权之和。

接下来，我们应该求 E 对 $\mathbf{W}$ 的导数。首先，回想一下，隐藏层是输入层的线性计算。展开(1)中的向量，我们得

$$h_i = \sum_{k=1}^{V} x_k \cdot w_{ki} \qquad (13)$$

现在我们可以求 E 对 $\mathbf{W}$ 的每个元素的导数，得

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathbf{EH}_i \cdot x_k \qquad (14)$$

这相当于 $\mathbf{x}$ 和 EH 的张量积，即

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{x} \bigotimes \mathbf{EH} = \mathbf{x}\mathbf{E}\mathbf{H}^T \qquad (15)$$

由此我们得到一个 $V \times N$ 矩阵。由于 $\mathbf{x}$ 只有一个分量是非零的，所以只有 $\frac{\partial E}{\partial \mathbf{W}}$ 的一行非零，该行的值是 $\mathbf{EH}^T$ ，即 N - dim向量。我们得到了 $\mathbf{W}$ 的更新方程

$$\mathbf{v}_{w_I}^{(new)} = \mathbf{v}_{w_I}^{(old)} - \eta \mathbf{E}\mathbf{H}^T \qquad (16)$$

其中 $\mathbf{v}_{w_I}$ 是 $\mathbf{W}$ 的行，即唯一 context 词的“输入向量”，并且是导数不为零的 $\mathbf{W}$ 的唯一行。在此迭代之后， $\mathbf{W}$ 的所有其他行将保持不变，因为它们的导数为零。

直观地，由于向量 $\mathbf{EH}$ 是词典中所有词的输出向量经其预测误差 $e_j = y_j - t_j$ 加权后的总和，我们可以理解为 (16) 将词典中每个输出向量的一部分加到上下文单词的输入向量上。

- overestimating : 如果在输出层中，一个词 w_j 作为输出词的概率被 overestimating $(y_j > t_j)$，那么上下文单词 w_I 的输入向量将倾向于远离 w_j 的输出向量；

- underestimating : 如果 underestimating 了 w_j 作为输出词的概率 $(y_j < t_j)$，则输入向量 w_I 将趋向于接近 w_j 的输出向量;
- 如果对 w_j 的概率预测得比较准确，则对 w_I 的输入向量的移动影响不大。

w_I 的输入向量的移动由词典中所有向量的预测误差决定; 预测误差越大，词对上下文词输入向量的运动影响就越大。

由于在训练过程中，我们是通过迭代训练语料生成的 context-target 词对来更新模型参数，每次迭代更新对向量的影响也是累积的。我们可以想象成词 w 的输出向量被 w 的共现邻居的输入向量的来回往复的拖拽。

就好比有真实的弦在词 w 和其邻居词之间。同样的，输入向量也可以被想象成被很多输出向量拖拽。这种解释可以提醒我们想象成一个重力，或者其他力导向的布局。每个假想的弦的平衡长度与相关单词对之间同现的强度以及学习率有关。经过多次迭代，输入和输出向量的相对位置最终将稳定下来。

1.2 Multi-word context

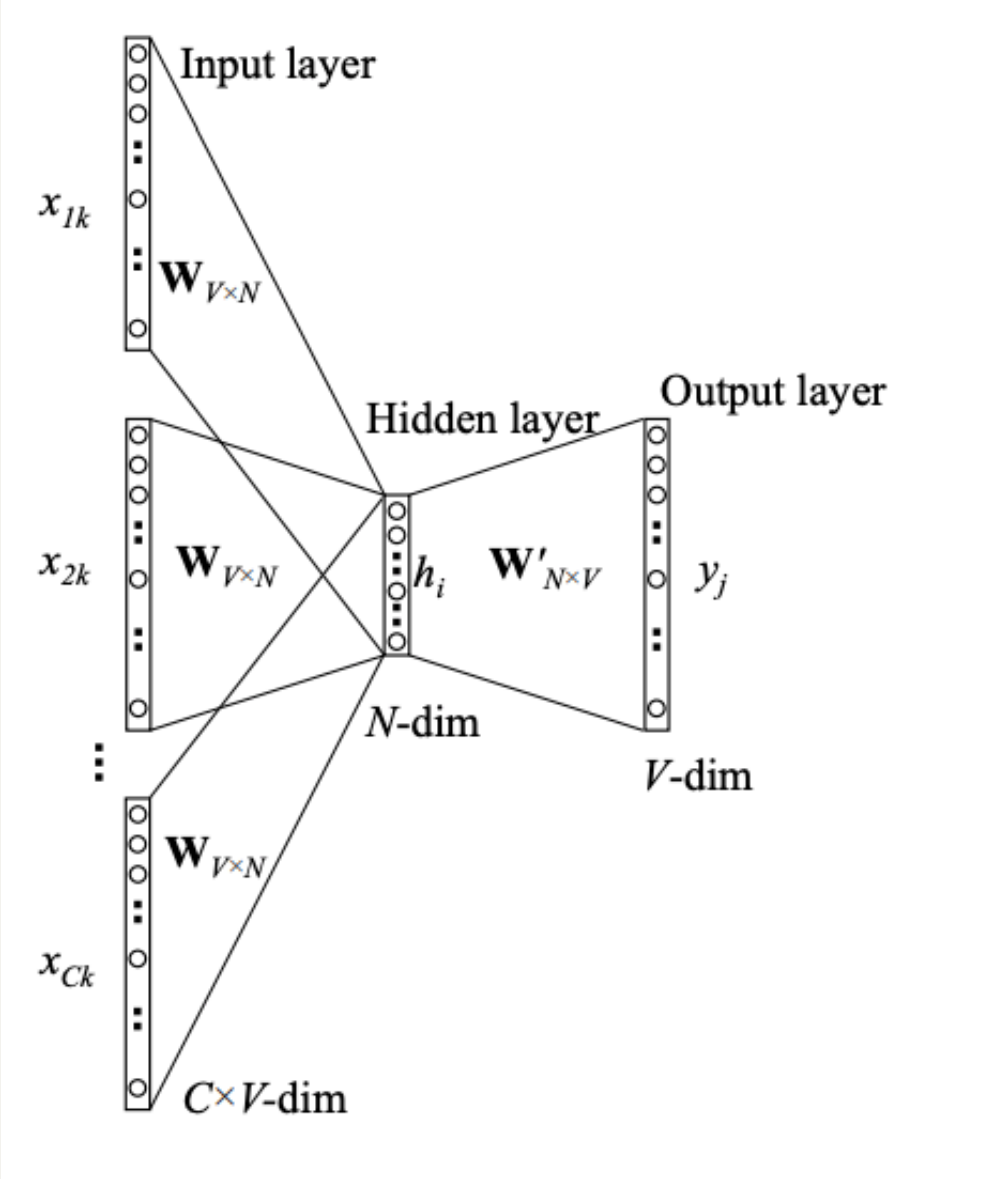

Figure 2: Continuous bag-of-word model

Figure 2 显示了具有 multi-word-context 设置的 CBOW 模型。在计算隐藏层输出时，CBOW 模型不是直接复制输入上下文词的输入向量，而是取输入上下文词向量的平均值，并用 input →hidden 的权值矩阵与平均向量的乘积作为输出。

$$\mathbf{h} = \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \qquad (17)$$

$$= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T \qquad (18)$$

其中 C 是上下文中的单词数，$w_1, \cdots, w_C$ 是上下文中的单词， $\mathbf{v}_w$ 是单词 w 的输入向量。

损失函数为

$$E = -log\, p(w_O|w_{I,1},\cdots,w_{I,C}) \qquad (19)$$

$$= -u_{j^*} \; + \; log \sum_{j'=1}^{V} exp(u_{j'}) \qquad (20)$$

$$= \text{-}\mathbf{v}'_{w_O}{}^T \cdot \mathbf{h} \; + \; log \sum_{j'=1}^{V} exp(\mathbf{v}'_{w_j}{}^T \cdot \mathbf{h}) \qquad (21)$$

它与 one-word-context 的目标函数(7)相同，只是 $\mathbf{h}$ 不同，$\mathbf{h}$ 如(18)中定义的，而不是(1)中定义的。

hidden → output 的权重的更新公式与 one-word-context 模型 (11) 相同。我们将其复制到此处：

$$\mathbf{v}'_{w_j}{}^{(new)} = \mathbf{v}'_{w_j}{}^{(old)} - \eta \cdot e_j \cdot \mathbf{h} \qquad for\; j = 1, 2, \cdots, V. \qquad (22)$$

请注意，我们需要将其应用于每个训练实例的 hidden→output 权重矩阵的每个元素。

input → hidden 权重的更新公式类似于(16)，不同之处在于现在我们需要对上下文中的每个单词 $w_{I,c}$ 应用以下公式

$$\mathbf{v}_{w_{I,c}}^{(new)} = \mathbf{v}_{w_{I,c}}^{(old)} - \frac{1}{C} \cdot \eta \cdot \mathbf{EH}^T \qquad for\; c = 1, 2, \cdots, C. \qquad (23)$$

其中 $\mathbf{v}_{w_{I,c}}$ 是输入上下文中 的第 c 个单词输入向量；

η 为正学习率; $\mathbf{EH} = \frac{\partial E}{\partial h_i}$ 由 (12) 给出。这个更新方程的直观理解与(16)相同。

2 Skip-Gram Model

Figure 3: The skip-gram model.

Mikolov等人提出了 skip-gram 模型。(2013a,b)。Figure 3 显示了 skip-gram模型。与 CBOW 模型相反。目标词(即中心词)现在位于输入层，上下文词位于输出层。

我们仍然使用 $\mathbf{v}_{w_I}$ 来表示输入层上唯一单词的输入向量，因此我们具有与 (1) 中相同的隐藏层输出 $\mathbf{h}$ 的定义，这意味着 $\mathbf{h}$ 只是复制(并转置)与输入单词 w_I 相关联的 input→hidden 权重矩阵 $\mathbf{W}$ 的行。我们将 $\mathbf{h}$ 的定义复制到下面：

$$\mathbf{h} = \mathbf{W}_{(k,\cdot)}^T := \mathbf{v}_{w_I}^T, \qquad (24)$$

在输出层，我们不是输出一个多项分布，而是输出 C 个多项分布。每个输出都使用相同的 hidden→output 矩阵来计算:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \qquad (25)$$

其中，$w_{c,j}$ 是相对于第 c 个输出层的第 j 个单词；$w_{O,c}$ 是相对于输出上下文词中的第 c 个单词；w_I 是唯一的输入词；$y_{c,j}$ 是第 c 个输出层上的第 j 个单元的输出；$u_{c,j}$ 是第 c 个 输出层的第 j 个单元的输入。

(注：原文用的是 panel , 对应的应该是图里的 C 个 panel, 我觉得这里意思应该是在正负样本的角度会有 C 组，即 context 的长度为 C , 其中 j 对应的是相对于正负样本集合中的第 j 个词，这里的正负样本集合指的是词典中的所有词)

由于所有输出层共享相同的权重矩阵，因此

$$u_{c,j} = u_j = \mathbf{v}_{w_j}^{'T} \cdot \mathbf{h}, \quad for\ c = 1, 2, \cdots, C \qquad (26)$$

其中 $\mathbf{v}_{w_j}^{'}$ 是词典中第 j 个词的输出向量 w_j ，并且 $\mathbf{v}_{w_j}^{'}$ 取自 hidden → output 权重矩阵 $\mathbf{W}^{'}$ 的列。

参数更新方程的推导与 one-word-context 模型没有太大不同。损失函数更改为

$$\begin{aligned} E &= -log\, p(w_{O,1}, w_{O,2}, \cdots, w_{O,C}|w_I) \qquad (27) \\ &= -log \prod_{c=1}^{C} \frac{exp(u_{c,j_c^*})}{\sum_{j'}^{V} exp(u_{j'})} \qquad (28) \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot log \sum_{j'=1}^{V} exp(u_{j'}) \qquad (29) \end{aligned}$$

其中 j_c^* 上下文中第 c 个词(注：正样本)在词典中的索引。

我们求 E 对 $u_{c,j}$ 的偏导(注：是$u_{c,j}$，不是 $u_{j_c^*}$),得到：

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j} \qquad (30)$$

跟公式(8)一致，这是一个单元上的预测误差。为方便表示，我们将 V 维向量 $\mathbf{EI} = \{\mathbf{EI}_1, \cdots, \mathbf{EI}_V\}$, 向量是预测了 C 个预测词的误差总和：

$$\mathbf{EI}_j = \sum_{c=1}^{C} e_{c,j} \qquad (31)$$

接下来，我们求 E 对 hidden → output 的权重矩阵 $\mathbf{W}^{'}$ 的偏导，可得：

$$\frac{\partial E}{\partial w_{ij}^{'}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w_{ij}^{'}} = \mathbf{EI}_j \cdot h_i \qquad (32)$$

因此，我们得到了 hidden → ouput 矩阵 $\mathbf{W}^{'}$ 的更新方程

$$w_{ij}^{'\ (new)} = w_{ij}^{'\ (old)} - \eta \cdot \mathbf{EI}_j \cdot h_i \qquad (33)$$

或者

$$\mathbf{v}_{w_j}^{'\ (new)} = \mathbf{v}_{w_j}^{'\ (old)} - \eta \cdot \mathbf{EI}_j \cdot \mathbf{h} \qquad for\ j = 1, 2, \cdots, V. \qquad (34)$$

对这个更新方程的直观理解与 (11) 相同，除了预测误差是 C 个输出层的总和。注意，我们需要对每个训练实例的 hidden → output 矩阵的每个元素应用这个更新方程。

除了预测误差 e_j 被 $\mathbf{EI}_j$ 替换之外，input → hidden 矩阵的更新方程的推导与 (12) 至 (16) 相同。我们直接给出更新公式：

$$\mathbf{v}_{w_I}^{(new)} = \mathbf{v}_{w_I}^{(old)} - \eta \cdot \mathbf{EH}^T \qquad (35)$$

其中 $\mathbf{EH}$ 是 N 维向量，其每个分量定义为

$$\mathbf{EH}_i = \sum_{j=1}^{V} \mathbf{EI}_j \cdot w_{ij}^{'} \qquad (36)$$

(35) 的直观理解与 (16) 相同。

3 Optimizing Computational Efficiency

到目前为止，我们讨论的模型(bigram、CBOW、skip-gram)都是它们的原始形式，没有应用任何优化技巧。

对于所有这些模型，词典中的每个单词都有两个向量表示：输入向量 $\mathbf{v}_w$ 和输出向量 $\mathbf{v}'_w$ 。学习输入向量的代价较小，但是学习输出向量是非常昂贵的。从更新公式(22)和(33)中我们可以发现，为了更新 $\mathbf{v}'_w$ ，对于每个训练实例，我们必须迭代词典中的每个单词 w_j，计算它们的输入 u_j 、预测概率 y_j (或对于 skip-gram 为 $y_{c,j}$)、它们的预测误差 e_j (或对于 skip-gram 为 $\mathbf{EI}_j$)，并最终使用它们的预测误差去更新输出向量 $\mathbf{v}'_j$

对每个训练实例的所有词进行这样的计算代价十分高昂，这使得扩展到大型词典或大型训练语料库是不切实际的。要解决这个问题，直觉上是限制每个训练实例的必须更新的输出向量的数量。实现这一目标的一种优雅方法是 hierarchical softmax; 另一种方法是通过抽样，这将在下一节中讨论。

这两种技巧都只优化了输出向量更新的计算。在推导过程中，我们关心三个值:

1. E ，新的目标函数;
2. $\frac{\partial E}{\partial \mathbf{v}'_w}$，新的输出向量更新方程;
3. $\frac{\partial E}{\partial \mathbf{h}}$，为更新输入向量而反向传播的预测误差的加权和。

3.1 Hierarchical Softmax

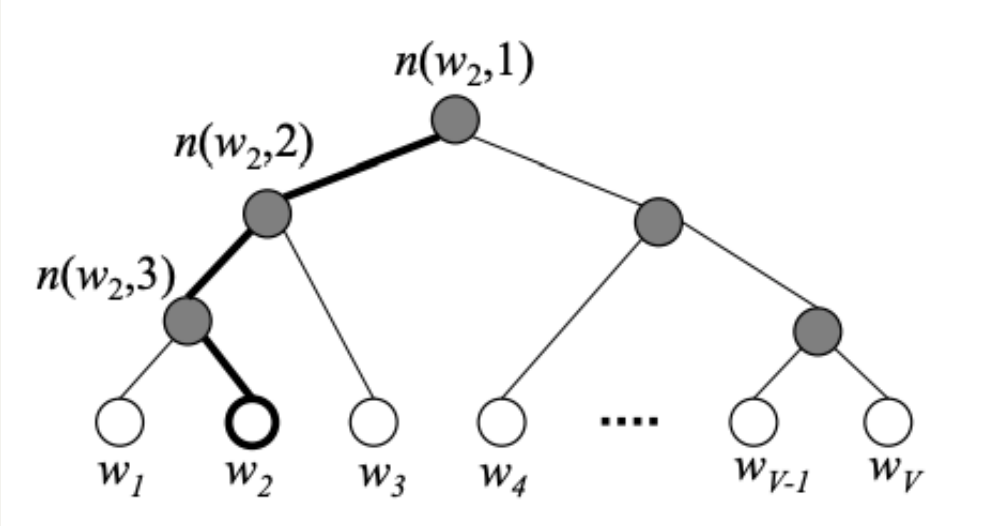

Figure 4: 一个基于二叉树的 hierarchical softmax 模型的示例. 白色的叶子节点是词典中的词, 深色的是非叶子结点. 从跟节点到 w_2 节点的路径被标出.在例子中, 路径的长度为 $L(w_2) = 4$. $n(w, j)$ 表示从根节点到词 w 上的第 j 个词.

Hierarchical softmax是一种有效的计算 softmax 的方法(Morin 和 Bengio, 2005; Mnih 和 Hinton, 2009)。该模型使用二叉树来表示词典中的所有单词。V 个单词必须存储于二叉树的叶子节点。可以被证明一共有 $V-1$ 个内部节点(非叶子节点)。对于每个叶子节点，有一条唯一的路径可以从根节点到达该叶子节点；该路径被用来计算该叶子节点代表的词的概率。参考 Figure 4。

Hierarchical softmax 模型没有词的输出向量，取而代之的是，$V-1$ 个内部节点 (非叶子节点) 都有一个输出向量 $\mathbf{v}'_{n(w,j)}$ 。一个词作为输出词的概率被定义为：

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma([[n(w, j+1) = ch(n(w, j))]] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}) \qquad (37)$$

- 其中 $ch(n)$ 是 n 的左子节点。
- $\mathbf{v}'_{n(w,j)}$ 是非叶子节点 $n(w,j)$ 的向量表示，即输出向量;
- $\mathbf{h}$ 是隐藏层的输出值 (在 skip-gram 模型中 $\mathbf{h} = \mathbf{v}_{w_I}$; 在 CBOW 中, $\mathbf{h} = \frac{1}{C}\sum_{c=1}^{C} \mathbf{v}_{w_c}$);

  $[[x]]$ 是一个特殊函数，定义如下（注：类似指示函数）：

$$[[x]] = \begin{cases} 1 & if\ x\ is\ true; \\ -1 & otherwise. \end{cases} \qquad (38)$$

让我们通过一个例子直观地理解这个方程。看图4，假设我们想计算 w_2 作为输出单词的概率。我们把这个概率定义为从根节点到叶子节点的随机游走的概率。在每个非叶子结点(包括根节点)，我们需要分配向左和向右的概率。(4虽然二叉树的内部节点不一定都有两个子节点，但二叉Huffman树的内部节点总是有两个子节点。虽然理论上可以使用许多不同类型的树来进行 hierarchical softmax，但word2vec使用二叉Huffman树来进行快速训练。)

我们定义在一个内部节点 n 处向左的概率为

$$p(n, left) = \sigma({\mathbf{v}'_n}^T \cdot \mathbf{h}) \qquad (39)$$

它由内部节点结点的向量表示和隐藏层的输出值(由输入层的向量表示决定)共同决定。显然，在非叶子结点 n 向右的概率是

$$p(n, right) = 1 - \sigma({\mathbf{v}_n^{'}}^T \cdot \mathbf{h}) = \sigma(-{\mathbf{v}_n^{'}}^T \cdot \mathbf{h}) \qquad (40)$$

根据 Figure 4 中从根节点到 w_2 节点的路径，我们可以计算出 w_2 作为输出词的概率

$$\begin{aligned} & p(w_2 = w_O) \\ = & p(n(w_2, 1), left) \cdot p(n(w_2, 2), left) \cdot p(n(w_2, 3), right) \qquad (41) \\ = & \sigma({\mathbf{v}_{n(w_2,1)}^{'}}^T \mathbf{h}) \cdot \sigma({\mathbf{v}_{n(w_2,2)}^{'}}^T \mathbf{h}) \cdot \sigma(-{\mathbf{v}_{n(w_2,3)}^{'}}^T \mathbf{h}) \qquad (42) \end{aligned}$$

这正是由 (37) 给出的。不难看出 (注：这里如何证明??)：

$$\sum_{i=1}^{V} p(w_i = w_O) = 1 \qquad (43)$$

这使得 hierarchical softmax 模型是一个的关于所有单词的良好定义的多项分布。

现在我们推导出内部节点的向量表示的参数更新方程。为了简单起见，我们先看看一个 one-word-context 模型。然后很容易再将更新公式扩展到 CBOW 和 skip-gram 模型。

为简化表示，我们定义了下列不会引起歧义缩写：

$$[[\cdot]] := [[n(w, j+1) = ch(n(w, j))]] \qquad (44)$$

$$\mathbf{v}_j^{'} := \mathbf{v}_{n_{w,j}}^{'} \qquad (45)$$

对于一个训练实例，误差函数定义为

$$E = -log\, p(w = w_O | w_I) = -\sum_{j=1}^{L(w)-1} log(\sigma([[\cdot]] {\mathbf{v}_j^{'}}^T \mathbf{h}) \qquad (46)$$

求 E 对 $\mathbf{v}_j^{'}\mathbf{h}$,的导数，得

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{v}_j^{'}\mathbf{h}} &= (\sigma([[\cdot]] {\mathbf{v}_j^{'}}^T \mathbf{h}) - 1)[[\cdot]] \qquad (47) \\ &= \begin{cases} \sigma({\mathbf{v}_j^{'}}^T \mathbf{h}) - 1 & ([[\cdot]] = 1) \\ \sigma({\mathbf{v}_j^{'}}^T \mathbf{h}) & ([[\cdot]] = -1) \end{cases} \qquad (48) \\ &= \sigma({\mathbf{v}_j^{'}}^T \mathbf{h}) - t_j \qquad (49) \end{aligned}$$

其中，如果$[[\cdot]]=1$，则$t_j=1$；否则，$t_j=0$。

接下来求 E 对内部节点 $n(w,j)$ 的偏导数，可得：

$$\frac{\partial E}{\partial \mathbf{v}_j^{'}}=\frac{\partial E}{\partial \mathbf{v}_j^{'}\mathbf{h}}\cdot\frac{\partial \mathbf{v}_j^{'}\mathbf{h}}{\partial \mathbf{v}_j^{'}}=(\sigma({\mathbf{v}_j^{'}}^T\mathbf{h})-t_j)\cdot\mathbf{h} \qquad (50)$$

最终的更新公式为：

$${\mathbf{v}_j^{'}}^{(new)}={\mathbf{v}_j^{'}}^{(old)}-\eta(\sigma({\mathbf{v}_j^{'}}^T\mathbf{h})-t_j)\cdot\mathbf{h} \qquad (50)$$

公式会从 $j=1,2,\cdots,L(w)-1$ 依次迭代. 我们可以将 $\sigma({\mathbf{v}_j^{'}}^T\mathbf{h})-t_j$ 理解为内部结点 $n(w,j)$ 的预测误差. 每个非叶子节点的“任务”是预测在随机游走中是向左还是向右。$t_j=1$ 表示向左; $t_j=0$ 表示向右。

$\sigma({\mathbf{v}_j^{'}}^T\mathbf{h})$ 表示预测结果. 对于训练实例来说, 如果节点的预测结果和真实路径非常相似, 那么 $\mathbf{v}_j^{'}$ 的向量表示只需要微小的改动; 否则 $\mathbf{v}_j^{'}$ 就会按适当的方向进行调整（要么靠近，要么远离 $\mathbf{h}$) 来减小预测误差。

这个更新公式既可以用于 CBOW 模型，也可以用于 skip-gram 模型。当用于 skip-gram 模型时，需要对输出的 C 个单词中的每个词重复这个更新过程。

为了反向传播误差去学习 input -> hidden 之间的权重矩阵，我们对 E 求隐藏层的输出的导数，得：

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{h}}&=\sum_{j=1}^{L(w)-1}\frac{\partial E}{\partial \mathbf{v}_j^{'}\mathbf{h}}\cdot\frac{\partial \mathbf{v}_j^{'}\mathbf{h}}{\partial \mathbf{h}} \qquad (52)\\&=\sum_{j=1}^{L(w)-1}(\sigma({\mathbf{v}_j^{'}}^T\mathbf{h})-t_j)\cdot\mathbf{v}_j^{'} \qquad (53)\\&:=\mathbf{EH} \qquad (54)\end{aligned}$$

可直接代入(23)得到 CBOW 输入向量的更新方程。对于 skip-gram 模型，我们需要为 skip-gram 上下文中的每个单词计算一个 **EH** 值，并将 **EH** 值的总和代入(35)，得到输入向量的更新方程。

从更新方程中可以看出，每个训练实例的每个上下文词的计算复杂度从 $O(V)$ 降低到 $O(log(V))$ ，在速度上有很大的提高。我们仍然有大致相同数量的参数(非叶子结点向量 $V-1$个，原始 V 个输出向量)。

3.2 Negative Sampling

Negative sampling 的理念比 Hierarchical softmax更简单：为了解决每次迭代都需要更新太多输出向量的问题，我们只更新其中的部分样本。

显然，输出词 (即 ground-truth，或正样本) 应该保存在我们的样本中并得到更新，我们需要将一些单词作为负样本 (即“负样本”) 进行采样。抽样过程需要一个概率分布，可以任意选择。我们称这个分布为噪声分布，表示为 $P_n(w)$。我们可以通过经验来确定一个好的分布 (如(Mikolov et al., 2013b)所述，word2vec使用 unigram 分布的 $\frac{3}{4}$ 次方以获得最佳质量的词向量)。

在 word2vec 中，作者认为用以下简化的训练目标取代用一个定义好的后验多项分布的负采样形式，也能够产生高质量的词嵌入 (Goldberg和Levy(2014) 对使用该目标函数的原因进行了理论分析。)：

$$E = -log\ \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{neg}} log\ \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h}) \qquad (55)$$

其中 w_O 是输出词(即正样本)， $\mathbf{v}'_{w_O}$ 是其输出向量； $\mathbf{h}$ 是隐藏层的输出值： CBOW： $\mathbf{h} = \frac{1}{C}\sum_{c=1}^{C} \mathbf{v}_{w_c}$, skip-gram： $\mathbf{h} = \mathbf{v}_{w_c}$。 $\mathcal{W}_{neg} = \{w_j | j = 1, \cdots, K\}$ 是基于 $P_n(w)$ 采样的单词集合，即负样本。

为了获取负采样词向量的更新方程，我们首先求 E 对 输出单元的输入 w_j的导数：

$$\begin{aligned} \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} &= \begin{cases} \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - 1 & if\ w_j = w_O \\ \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) & if\ w_j \in \mathcal{W}_{neg} \end{cases} \qquad (56) \\ &= \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j \qquad (57) \end{aligned}$$

其中 t_j 是单词 w_j 的“标签”。当 w_j 为正样本时，$t = 1$；否则，$t = 0$。

接下来，我们取 E 对词 w_j 的输出向量的导数，

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}} = \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} \cdot \frac{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}}{\partial \mathbf{v}'_{w_j}} = (\sigma({\mathbf{v}'_{w_j}}^T \mathbf{h} - t_j)) \cdot \mathbf{h} \qquad (58)$$

以下的输出向量更新公式：

$${\mathbf{v}'_{w_j}}^{(new)} = {\mathbf{v}'_{w_j}}^{(old)} - \eta(\sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j) \cdot \mathbf{h} \qquad (59)$$

其中只需要更新 $w_j \in \{w_O\} \cup \mathcal{W}_{neg}$，而不是词典中的每个词。这也解释了为什们我们可以在一次迭代中节省巨大的计算量。

直觉上对该更新公式的理解和公式(11)一致。该公式可以通用于 CBOW 模型和 skip-gram 模型。对于 skip-gram 模型，我们一次作用于一个上下文单词。

为了使误差反向传播到隐藏层来更新词的输入向量，我们求 E 对隐藏层输出 $\mathbf{h}$ 的偏导：

$$\frac{\partial E}{\partial \mathbf{h}} = \sum_{w_j \in \{w_O\} \bigcup \mathcal{W}_{neg}} \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} \cdot \frac{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}}{\partial \mathbf{h}} \qquad (60)$$

$$= \sum_{w_j \in \{w_O\} \bigcup \mathcal{W}_{neg}} (\sigma({v'_{w_j}}^T \mathbf{h}) - t_j) \mathbf{v}'_{w_j} := \mathbf{EH} \qquad (61)$$

把 $\mathbf{EH}$ 带入公式(23)可得 CBOW 模型的输入向量的更新公式。对于 skip-gram 模型，我们计算每个单词的 $\mathbf{EH}$ 值并加和再带入公式(35)就可得到输入向量的更新公式。

Acknowledgement

The author would like to thank Eytan Adar, Qiaozhu Mei, Jian Tang, Dragomir Radev, Daniel Pressel, Thomas Dean, Sudeep Gandhe, Peter Lau, Luheng He, Tomas Mikolov, Hao Jiang, and Oded Shmueli for discussions on the topic and/or improving the writing of the note.

References

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negativesampling word-embedding method. arXiv:1402.3722 [cs, stat]. arXiv: 1402.3722.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119.

Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, Advances in Neural Information Processing Systems 21, pages 1081–1088. Curran Associates, Inc.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In AISTATS, volume 5, pages 246–252. Citeseer.

Q&A

- 输入向量如何更新

  > *h_i 的更新是所有经过误差加权的相关输出向量的第 i 个维度的和。*

- Hierarchical softmax 的预测原理
- Negative sampling 从什么分布采样
-