（四）基于 Hierarchical Softmax 的模型

原创 皮果提 2014-07-19 22:53:39 148873 版权

分类专栏： 语言模型 文章标签： word2vec CBOW Skip-gram Hierarchical Softmax Negative Sampling

word2vec 是 Google 于 2013 年开源推出的一个用于获取 word vector 的工具包，它简单、高效，因此引起了很多人的关注。由于 word2vec 的作者 Tomas Mikolov 在两篇相关的论文 [3,4] 中并没有谈及太多算法细节，因而在一定程度上增加了这个工具包的神秘感。一些按捺不住的人于是选择了通过解剖源代码的方式来一窥究竟，出于好奇，我也成为了他们中的一员。读完代码后，觉得收获颇多，整理成文，给有需要的朋友参考。

相关链接

(Continuous Bag-of-Words Model) 和 Skip-gram 模型 (Continuous Skip-gram Model). 关于这两个模型, 作者 Tomas Mikolov 在文 [5] 给出了如图 8 和图 9 所示的示意图.

由图可见, 两个模型都包含三层: **输入层、投影层和输出层**. 前者是在已知当前词 w_t 的上下文 $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ 的前提下预测当前词 w_t (见图 8); 而后者恰恰相反, 是在已知当前词 w_t 的前提下, 预测其上下文 $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ (见图 9).

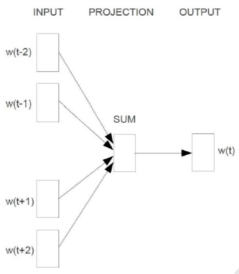

图 8 CBOW 模型

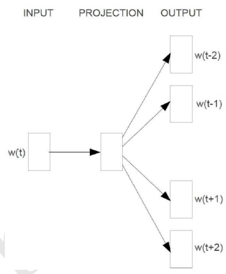

图 9 Skip-gram 模型

对于 CBOW 和 Skip-gram 两个模型, word2vec 给出了两套框架, 它们分别基于 Hierarchical Softmax 和 Negative Sampling 来进行设计. 本节介绍基于 Hierarchical Softmax 的 CBOW 和 Skip-gram 模型.

在 §3.2 中, 我们提到, 基于神经网络的语言模型的目标函数通常取为如下**对数似然函数**

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log p(w|Context(w)), \tag{4.1}$$

其中的关键是条件概率函数 $p(w|Context(w))$ 的构造, 文 [2] 中的模型就给出了这个函数的一种构造方法 (见 (3.6) 式).

对于 word2vec 中基于 Hierarchical Softmax 的 CBOW 模型, 优化的目标函数也形如 (4.1); 而对于基于 Hierarchical Softmax 的 Skip-gram 模型, 优化的目标函数则形如

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log p(Context(w)|w), \tag{4.2}$$

因此, 讨论过程中我们应将重点放在 $p(w|Context(w))$ 或 $p(Context(w)|w)$ 的构造上, 意识到这一点很重要, 因为它可以让我们目标明确、心无旁骛, 不致于陷入到一些繁琐的细节当中去. 接下来将从数学的角度对这两个模型进行详细介绍.

§4.1.1 网络结构

图 10 给出了 CBOW 模型的网络结构, 它包括三层: 输入层、投影层和输出层. 下面以样本 $(Context(w), w)$ 为例 (这里假设 $Context(w)$ 由 w 前后各 c 个词构成), 对这三个层做简要说明.

1. **输入层**: 包含 $Context(w)$ 中 $2c$ 个词的词向量 $\mathbf{v}(Context(w)_1), \mathbf{v}(Context(w)_2), \cdots, \mathbf{v}(Context(w)_{2c}) \in \mathbb{R}^m$. 这里, m 的含义同上表示词向量的长度.
2. **投影层**: 将输入层的 $2c$ 个向量做求和累加, 即 $\mathbf{x}_w = \sum_{i=1}^{2c} \mathbf{v}(Context(w)_i) \in \mathbb{R}^m$.

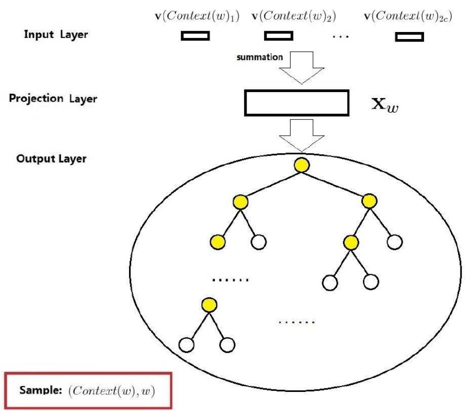

图 10 CBOW 模型的网络结构示意图

3. **输出层**: 输出层对应一棵二叉树, 它是以语料中出现过的词当叶子结点, 以各词在语料中出现的次数当权值构造出来的 Huffman 树. 在这棵 Huffman 树中, 叶子结点共 N $(= |\mathcal{D}|)$ 个, 分别对应词典 $\mathcal{D}$ 中的词, 非叶子结点 $N-1$ 个 (图中标成黄色的那些结点).

对比 §3.3 中神经概率语言模型的网络图 (见图 4) 和 CBOW 模型的结构图 (见图 10), 易知它们主要有以下三处不同:

1. (从输入层到投影层的操作) 前者是通过拼接, 后者通过**累加求和**.
2. (隐藏层) 前者有隐藏层, 后者**无隐藏层**.
3. (输出层) 前者是线性结构, 后者是**树形结构**.

在 §3.3 介绍的神经概率语言模型中, 我们指出, 模型的大部分计算集中在隐藏层和输出层之间的矩阵向量运算, 以及输出层上的 softmax 归一化运算. 而从上面的对比中可见, CBOW 模型对这些计算复杂度高的地方有针对性地进行了改变, 首先, 去掉了隐藏层, 其次, 输出层改用了 Huffman 树, 从而为利用 Hierarchical softmax 技术奠定了基础.

§4.1.2 梯度计算

Hierarchical Softmax 是 wor2vec 中用于提高性能的一项关键技术. 为描述方便起见, 在具体介绍这个技术之前, 先引入若干相关记号. 考虑 Huffman 树中的某个叶子结点, 假设它对应词典 $\mathcal{D}$ 中的词 w, 记

1. p^w: 从根结点出发到达 w 对应叶子结点的路径.
2. l^w: 路径 p^w 中包含结点的个数.
3. $p_1^w, p_2^w, \cdots, p_{l^w}^w$: 路径 p^w 中的 l^w 个结点, 其中 p_1^w 表示根结点, $p_{l^w}^w$ 表示词 w 对应的结点.
4. $d_2^w, d_3^w, \cdots, d_{l^w}^w \in \{0, 1\}$: 词 w 的 Huffman 编码, 它由 $l^w - 1$ 位编码构成, d_j^w 表示路径 p^w 中第 j 个结点对应的编码 (根结点不对应编码).
5. $\theta_1^w, \theta_2^w, \cdots, \theta_{l^w-1}^w \in \mathbb{R}^m$: 路径 p^w 中**非叶子结点**对应的向量, θ_j^w 表示路径 p^w 中第 j 个非叶子结点对应的向量.

> **注 4.1** *按理说, 我们要求的是词典 $\mathcal{D}$ 中每个词 (即 Huffman 树中所有叶子节点) 的向量, 为什么这里还要为 Huffman 树中每一个非叶子结点也定义一个同长的向量呢? 事实上, 它们只是算法中的辅助向量, 具体用途在下文中将会为大家解释清楚.*

好了, 引入了这么一大堆抽象的记号, 接下来, 我们还是通过一个简单的例子把它们落到实处吧, 看图 11, 仍以预备知识中例 2.1 为例, 考虑词 $w =$ "足球" 的情形.

图 11 中由 4 条红色边串起来的 5 个节点就构成路径 p^w, 其长度 $l^w = 5$. $p_1^w, p_2^w, p_3^w, p_4^w, p_5^w$ 为路径 p^w 上的 5 个结点, 其中 p_1^w 对应根结点. $d_2^w, d_3^w, d_4^w, d_5^w$ 分别为 1, 0, 0, 1, 即 "足球" 的 Huffman 编码为 1001. 此外, $\theta_1^w, \theta_2^w, \theta_3^w, \theta_4^w$ 分别表示路径 p^w 上 4 个非叶子结点对应的向量.

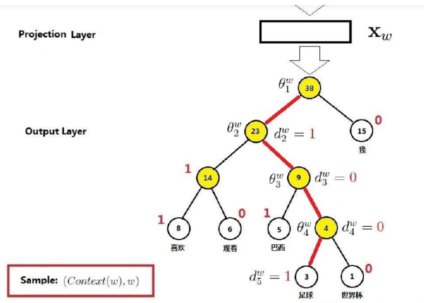

图 11 $w =$ “足球” 时的相关记号示意图

那么, 在如图 10 所示的网络结构下, 如何定义条件概率函数 $p(w|Contex(w))$ 呢? 更具体地说, 就是如何利用向量 $\mathbf{x}_w \in \mathbb{R}^m$ 以及 Huffman 树来定义函数 $p(w|Contex(w))$ 呢?

以图 11 中词 $w =$ “足球” 为例. 从根结点出发到达 “足球” 这个叶子节点, 中间共经历了 4 次分支 (每条红色的边对应一次分支), 而每一次分支都可视为进行了一次**二分类**.

既然是从二分类的角度来考虑问题, 那么对于每一个非叶子结点, 就需要为其左右孩子结点指定一个类别, 即哪个是正类 (标签为 1), 哪个是负类 (标签为 0). 碰巧, 除根结点以外, 树中每个结点都对应了一个取值为 0 或 1 的 Huffman 编码. 因此, 一种最自然的做法就是将 Huffman 编码为 1 的结点定义为正类, 编码为 0 的结点定义为负类. 当然, 这只是个约定而已, 你也可以将编码为 1 的结点定义为负类, 而将编码为 0 的结点定义为正类. 事实上, word2vec 选用的就是后者, 为方便读者对照着文档看源码, 下文中统一采用后者, 即约定

$$Label(p_i^w) = 1 - d_i^w, i = 2, 3, \cdots, l^w.$$

简言之就是, **将一个结点进行分类时, 分到左边就是负类, 分到右边就是正类.**

根据预备知识 §2.2 中介绍的逻辑回归, 易知, 一个结点被分为正类的概率是

$$\sigma(\mathbf{x}_w^\top \theta) = \frac{1}{1 + e^{-\mathbf{x}_w^\top \theta}},$$

被分为负类的概率当然就等于

$$1 - \sigma(\mathbf{x}_w^\top \theta),$$

注意, 上式中有个叫 θ 的向量, 它是待定参数, 显然, 在这里非叶子结点对应的那些向量 θ_i^w 就可以扮演参数 θ 的角色 (这也是为什么将它们取名为 θ_i^w 的原因).

对于从根结点出发到达 “足球” 这个叶子节点所经历的 4 次二分类, 将每次分类结果的概率写出来就是

1. 第 1 次: $p(d_2^w|\mathbf{x}_w, \theta_1^w) = 1 - \sigma(\mathbf{x}_w^\top \theta_1^w)$;
2. 第 2 次: $p(d_3^w|\mathbf{x}_w, \theta_2^w) = \sigma(\mathbf{x}_w^\top \theta_2^w)$;
3. 第 3 次: $p(d_4^w|\mathbf{x}_w, \theta_3^w) = \sigma(\mathbf{x}_w^\top \theta_3^w))$;
4. 第 4 次: $p(d_5^w|\mathbf{x}_w, \theta_4^w) = 1 - \sigma(\mathbf{x}_w^\top \theta_4^w)$,

但是, 我们要求的是 $p(\text{足球}|Contex(\text{足球}))$, 它跟这 4 个概率值有什么关系呢? 关系就是

$$p(\text{足球}|Contex(\text{足球})) = \prod_{j=2}^{5} p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w).$$

至此, 通过 $w =$ “足球” 的小例子, **Hierarchical Softmax** 的基本思想其实就已经介绍完了. 小结一下: **对于词典 $\mathcal{D}$ 中的任意词 w, Huffman 树中必存在一条从根结点到词 w 对应结点的路径 p^w (且这条路径是唯一的). 路径 p^w 上存在 $l^w - 1$ 个分支, 将每个分支看做一次二分类, 每一次分类就产生一个概率, 将这些概率乘起来, 就是所需的 $p(w|Context(w))$.**

条件概率 $p(w|Context(w))$ 的一般公式可写为

$$p(w|Context(w)) = \prod_{j=2}^{l^w} p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w), \tag{4.3}$$

其中

$$p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta_{j-1}^w), & d_j^w = 0; \\ 1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w), & d_j^w = 1, \end{cases}$$

或者写成整体表达式

$$p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w) = [\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{d_j^w}.$$

注 4.2 *在 §3.3 中, 最后得到的条件概率为*

$$p(w|Context(w)) = \frac{e^{y_{w,i_w}}}{\sum_{i=1}^{N} e^{y_{w,i}}}.$$

具体见 (3.6) 式, 由于这里有个归一化操作, 因此显然成立

$$\sum_{w \in \mathcal{D}} p(w|Context(w)) = 1.$$

然而, 对于由 (4.3) 定义的概率, 是否也能满足上式呢? 这个问题留给读者思考.

$$= \sum_{w\in\mathcal{C}}\sum_{j=2}^{l^w}\left\{(1-d_j^w)\cdot\log[\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]+d_j^w\cdot\log[1-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]\right\}, \tag{4.4}$$

为下面梯度推导方便起见，将上式中双重求和符号下花括号里的内容简记为 $\mathcal{L}(w,j)$，即

$$\mathcal{L}(w,j) = (1-d_j^w)\cdot\log[\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]+d_j^w\cdot\log[1-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]. \tag{4.5}$$

至此，已经推导出对数似然函数 (4.4)，这就是 CBOW 模型的目标函数，接下来讨论它的优化，即如何将这个函数最大化．word2vec 里面采用的是**随机梯度上升法***．而梯度类算法的关键是给出相应的梯度计算公式，因此接下来重点讨论梯度的计算．

随机梯度上升法的做法是：每取一个样本 $(Context(w), w)$，就对目标函数中的所有（相关）参数做一次刷新．观察目标函数 $\mathcal{L}$ 易知，该函数中的参数包括向量 $\mathbf{x}_w, \theta_{j-1}^w$，$w\in\mathcal{C}$，$j=2,\cdots,l^w$．为此，先给出函数 $\mathcal{L}(w,j)$ 关于这些向量的梯度．

首先考虑 $\mathcal{L}(w,j)$ 关于 θ_{j-1}^w 的梯度计算．

$$\begin{aligned}
\frac{\partial\mathcal{L}(w,j)}{\partial\theta_{j-1}^w} &= \frac{\partial}{\partial\theta_{j-1}^w}\left\{(1-d_j^w)\cdot\log[\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]+d_j^w\cdot\log[1-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]\right\}\\
&= (1-d_j^w)[1-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)]\mathbf{x}_w - d_j^w\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)\mathbf{x}_w \quad \text{（利用 (2.1) 式）}\\
&= \left\{(1-d_j^w)[1-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)] - d_j^w\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)\right\}\mathbf{x}_w \quad \text{（合并）}\\
&= \left[1-d_j^w-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)\right]\mathbf{x}_w.
\end{aligned}$$

于是，θ_{j-1}^w 的更新公式可写为

$$\theta_{j-1}^w := \theta_{j-1}^w + \eta\left[1-d_j^w-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)\right]\mathbf{x}_w,$$

其中 η 表示**学习率**，下同．

接下来考虑 $\mathcal{L}(w,j)$ 关于 $\mathbf{x}_w$ 的梯度．观察 (4.5) 可发现，$\mathcal{L}(w,j)$ 中关于变量 $\mathbf{x}_w$ 和 θ_{j-1}^w 是**对称**的（即两者可交换位置），因此，相应的梯度 $\frac{\partial\mathcal{L}(w,j)}{\partial\mathbf{x}_w}$ 也只需在 $\frac{\partial\mathcal{L}(w,j)}{\partial\theta_{j-1}^w}$ 的基础上对这两个向量交换位置就可以了，即

$$\frac{\partial\mathcal{L}(w,j)}{\partial\mathbf{x}_w} = \left[1-d_j^w-\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)\right]\theta_{j-1}^w.$$

到这里，细心的读者可能已经看出问题来了：我们的最终目的是要求词典 $\mathcal{D}$ 中每个词的词向量，而这里的 $\mathbf{x}_w$ 表示的是 $Context(w)$ 中各词词向量的累加．那么，如何利用 $\frac{\partial\mathcal{L}(w,j)}{\partial\mathbf{x}_w}$ 来对 $\mathbf{v}(\widetilde{w}), \widetilde{w}\in Context(w)$ 进行更新呢？word2vec 中的做法很简单，直接取

$$\mathbf{v}(\widetilde{w}) := \mathbf{v}(\widetilde{w}) + \eta\sum_{j=2}^{l^w}\frac{\partial\mathcal{L}(w,j)}{\partial\mathbf{x}_w}, \quad \widetilde{w}\in Context(w),$$

*求最小值用（随机）梯度下降法，求最大值用（随机）梯度上升法，这种基于梯度的方法通常统称为（随机）梯度下降法．这里强调“上升”是想提醒大家这是在求最大值．

即把 $\sum_{j=2}^{l^w}\frac{\partial\mathcal{L}(w,j)}{\partial\mathbf{x}_w}$ 贡献到 $Context(w)$ 中每一个词的词向量上．这个应该很好理解，既然 $\mathbf{x}_w$ 本身就是 $Context(w)$ 中各词词向量的累加，求完梯度后当然也应该将其贡献到每个分量上去．

注 4.3 *当然，读者这里需要考虑的是：采用*__平均__*贡献会不会更合理？即使用公式*

$$\mathbf{v}(\widetilde{w}) := \mathbf{v}(\widetilde{w}) + \frac{\eta}{|Context(w)|}\sum_{j=2}^{l^w}\frac{\partial\mathcal{L}(w,j)}{\partial\mathbf{x}_w}, \quad \widetilde{w}\in Context(w),$$

其中 $|Context(w)|$ 表示 $Context(w)$ 中词的个数．

下面以样本 $(Context(w), w)$ 为例，给出 CBOW 模型中采用随机梯度上升法更新各参数的伪代码．

1. $\mathbf{e} = \mathbf{0}$.
2. $\mathbf{x}_w = \sum_{u\in Context(w)}\mathbf{v}(u)$.
3. FOR $j = 2 : l^w$ DO
   {
   - 3.1 $q = \sigma(\mathbf{x}_w^\top\theta_{j-1}^w)$
   - 3.2 $g = \eta(1-d_j^w-q)$
   - 3.3 $\mathbf{e} := \mathbf{e} + g\theta_{j-1}^w$
   - 3.4 $\theta_{j-1}^w := \theta_{j-1}^w + g\mathbf{x}_w$

   }
4. FOR $u\in Context(w)$ DO
   {
   - $\mathbf{v}(u) := \mathbf{v}(u) + \mathbf{e}$

   }

注意，步 3.3 和步 3.4 不能交换次序，即 θ_{j-1}^w 应等贡献到 $\mathbf{e}$ 后再做更新．

注 4.4 *结合上面的伪代码，简单给出其与 word2vec 源码中的对应关系如下：syn0 对应 $\mathbf{v}(\cdot)$，syn1 对应 θ_{j-1}^w，neu1 对应 $\mathbf{x}_w$，neu1e 对应 $\mathbf{e}$．*

§4.2.1 **网络结构**

图 12 给出了 Skip-gram 模型的网络结构, 同 CBOW 模型的网络结构一样, 它也包括三层: 输入层、投影层和输出层. 下面以样本 $(w, Context(w))$ 为例, 对这三个层做简要说明.

1. **输入层**: 只含当前样本的中心词 w 的词向量 $\mathbf{v}(w) \in \mathbb{R}^m$.

2. **投影层**: 这是个恒等投影, 把 $\mathbf{v}(w)$ 投影到 $\mathbf{v}(w)$. 因此, **这个投影层其实是多余的**, 这里之所以保留投影层主要是方便和 CBOW 模型的网络结构做对比.

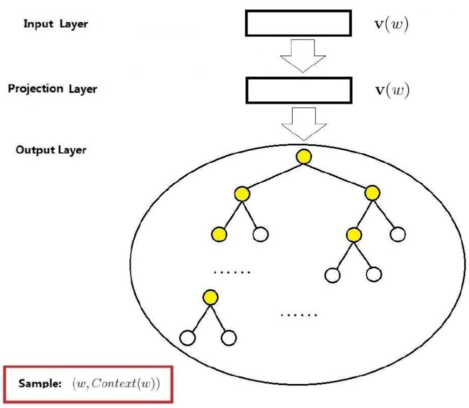

图 12 Skip-gram 模型的网络结构示意图

3. **输出层**: 和 CBOW 模型一样, 输出层也是一棵 Huffman 树.

§4.2.2 **梯度计算**

对于 Skip-gram 模型, 已知的是当前词 w, 需要对其上下文 $Context(w)$ 中的词进行预测, 因此目标函数应该形如 (4.2), 且关键是条件概率函数 $p(Context(w)|w)$ 的构造, Skip-gram 模型中将其定义为

$$p(Context(w)|w) = \prod_{u \in Context(w)} p(u|w),$$

上式中的 $p(u|w)$ 可按照上小节介绍的 Hierarchical Softmax 思想, 类似于 (4.3) 地写为

$$p(u|w) = \prod_{j=2}^{l^u} p(d_j^u|\mathbf{v}(w), \theta_{j-1}^u),$$

其中

$$p(d_j^u|\mathbf{v}(w), \theta_{j-1}^u) = [\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{d_j^u}. \tag{4.6}$$

将 (4.6) 依次代回, 可得对数似然函数 (4.2) 的具体表达式

$$\begin{aligned}\mathcal{L} &= \sum_{w \in \mathcal{C}} \log \prod_{u \in Context(w)} \prod_{j=2}^{l^u} \left\{ [\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{d_j^u} \right\} \\ &= \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \sum_{j=2}^{l^u} \left\{ (1 - d_j^u) \cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] \right\}.\end{aligned} \tag{4.7}$$

同样, 为下面梯度推导方便起见, 将三重求和符号下花括号里的内容简记为 $\mathcal{L}(w, u, j)$, 即

$$\mathcal{L}(w, u, j) = (1 - d_j^u) \cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)].$$

至此, 已经推导出对数似然函数的表达式 (4.7), 这就是 Skip-gram 模型的目标函数. 接下来同样利用**随机梯度上升法**对其进行优化, 关键是要给出两类梯度.

首先考虑 $\mathcal{L}(w, u, j)$ 关于 θ_{j-1}^u 的梯度计算 (与 CBOW 模型对应部分的推导完全类似).

$$\begin{aligned}\frac{\partial \mathcal{L}(w, u, j)}{\partial \theta_{j-1}^u} &= \frac{\partial}{\partial \theta_{j-1}^u} \left\{ (1 - d_j^u) \cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] \right\} \\ &= (1 - d_j^u)[1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]\mathbf{v}(w) - d_j^u \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)\mathbf{v}(w) \quad \text{(利用 (2.1) 式)} \\ &= \left\{ (1 - d_j^u)[1 - \sigma(\mathbf{v}(w)^\top \mathbf{v}_{j-1}^u)] - d_j^u \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u) \right\} \mathbf{v}(w) \quad \text{(合并)} \\ &= \left[1 - d_j^u - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u) \right] \mathbf{v}(w).\end{aligned}$$

于是, θ_{j-1}^u 的更新公式可写为

$$\theta_{j-1}^u := \theta_{j-1}^u + \eta \left[1 - d_j^u - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u) \right] \mathbf{v}(w).$$

接下来考虑 $\mathcal{L}(w, u, j)$ 关于 $\mathbf{v}(w)$ 的梯度. 同样利用 $\mathcal{L}(w, u, j)$ 中 $\mathbf{v}(w)$ 和 θ_{j-1}^w 的**对称性**, 有

$$\frac{\partial \mathcal{L}(w, u, j)}{\partial \mathbf{v}(w)} = \left[1 - d_j^u - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u) \right] \theta_{j-1}^u.$$

$u \in Context(w) \quad j=2 \qquad \partial \mathbf{v}(w)$

下面以样本 $(w, Context(w))$ 为例, 给出 Skip-gram 模型中采用随机梯度上升法更新各参数的伪代码.

```
e = 0
FOR u ∈ Context(w)  DO
{
    FOR j = 2 : l^u  DO
    {
        1.  q = σ(v(w)^⊤ θ^u_{j-1})
        2.  g = η(1 - d^u_j - q)
        3.  e := e + g θ^u_{j-1}
        4.  θ^u_{j-1} := θ^u_{j-1} + g v(w)
    }
}
v(w) := v(w) + e
```

但是, word2vec 源码中, 并不是等 $Context(w)$ 中的所有词都处理完后才刷新 $\mathbf{v}(w)$, 而是, 每处理完 $Context(w)$ 中的一个词 u, 就及时刷新一次 $\mathbf{v}(w)$, 具体为

```
FOR u ∈ Context(w)  DO
{
    e = 0
    FOR j = 2 : l^u  DO
    {
        1.  q = σ(v(w)^⊤ θ^u_{j-1})
        2.  g = η(1 - d^u_j - q)
        3.  e := e + g θ^u_{j-1}
        4.  θ^u_{j-1} := θ^u_{j-1} + g v(w)
    }
    v(w) := v(w) + e
}
```

同样, 需要注意的是, 循环体内的步 3 和步 4 不能交换次序, 即 θ^u_{j-1} 要等贡献到 e 后才更新.

注 4.5 *结合上面的伪代码, 简单给出其与 word2vec 源码中的对应关系如下: syn0 对应 $\mathbf{v}(\cdot)$, syn1 对应 θ^u_{j-1}, neu1e 对应 e.*

参考文献

[1] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. **Learning representations by backpropagating errors.** Nature, 323(6088):533-536, 1986.

[2] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. **A neural probabilistic language model.** Journal of Machine Learning Research (JMLR), 3:1137-1155, 2003.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. **Efficient Estimation of Word Representations in Vector Space.** arXiv:1301.3781, 2013.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. **Distributed Representations of Words and Phrases and their Compositionality.** arXiv:1310.4546, 2013.

[5] Tomas Mikolov, Quoc V. Le, Ilya Sutskever. **Exploiting Similarities among Languages for Machine Translation.** arXiv:1309.4168v1, 2013.

[6] Quoc V. Le, Tomas Mikolov. **Distributed Representations of Sentences and Documents.** arXiv:1405.4053, 2014.

[7] Xiaoqing Zheng, Hanyang Chen, Tianyu Xu. **Deep Learning for Chinese Word Segmentation and POS tagging.** Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 647-657.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu and Pavel Kuksa. **Natural Language Processing (Almost) from Scratch.** Journal of Machine Learning Research (JMLR), 12:2493-2537, 2011.

[9] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. The Journal ofMachine Learning Research, 13:307-361, 2012.

[10] 百度百科中的 "哈夫曼树" 词条.

[11] 吴军. **《数学之美》**. 人民邮电出版社, 2012.

[12] http://ml.nec-labs.com/senna/

[13] http://www.loooooker.com/archives/5621

[14] licstar. **Deep Learning in NLP (一) 词向量和语言模型**. http://licstar.net/archives/328

[15] **深度学习 word2vec 笔记之基础篇**. http://blog.csdn.net/mytestmy/article/details/26961315

[16] **深度学习 word2vec 笔记之算法篇**. http://blog.csdn.net/mytestmy/article/details/26969149

[17] 邓澍军, 陆光明, 夏龙. **Deep Learning 实战之 word2vec**, 2014.

[18] 杨超. **Word2Vec 的一些理解**. http://www.zhihu.com/question/21661274/answer/19331979

[19] **基于权值的微博用户采样算法研究**. http://blog.csdn.net/itplus/article/details/9079297

[20] **利用 word2vec 训练的字向量进行中文分词**. http://blog.csdn.net/itplus/article/details/17122431

[21] Yoav Goldberg, Omer Levy. word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. arXiv: 1402.3722v1, 2014. (http://arxiv.org/pdf/1402.3722v1.pdf)