1 Hierarchical Softmax

- date: 2025-05-31 17:58:46
- tags: NLP

1.1 Intro

In traditional word embedding models like CBOW, we input a set of ont-hot vectors, which are the contexts of a target word. And introduce a weight matrix $W_{V\times N}$ to transform the vector from a very large dim V to a smaller dim N. Then transform them back with another matrix to dim V and employ Softmax to find the right target.

However, it seems not efficient enought when the word amount scales up to very large. Because the time complexity is $O(V)$.

In word2vec, a Huffman Tree was introduce to encode the words efficiently and then here comes the hierarchical softmax. The time complexity can be reduced to $O(log(V))$, which is very useful when the data is very large scale.

1.2 How it works?

The words are encoded into a Huffman Tree according to their frequency. Considering a case, the word with a higher frequency can be encoded faster than that with a lower frequency. Then the overall weighted complexity can be optimized. Now we use CBOW model to compute a word's embedding with its contextual words.

Definitions:

- w: the target word;
- p^w: the path to the target word with the give Huffman Tree;
- l^w: the length of the above path;
- p_i^w, where $1 <= i <= l^w$: each leaf in path p^w;
- d_i^w, where $2 <= i <= l^w$: $d_{l^w}^w \in \{0, 1\}$ represents the encoding of the word w. The others represents the encoding of the leaves in the path.

Target:

To learn the representation of each word. And optimize the loss:

$$\mathcal{L} = \sum_{w\in\mathcal{C}} log\ p(w|\text{context}(w)) \tag{1}$$

To compute this: ($l^w - 2$ times binary classification)

$$p(w|\text{context}(w)) = \prod_{j=2}^{l^w} p(d_j^w|\mathbf{X}_w, \theta_{j-1}^w) \tag{2}$$

We have:

$$\begin{aligned} p(w|\text{context}(w)) &= \begin{cases} \sigma(\mathbf{X}_w^T\theta_{j-1}^w) & , d_j^w = 0 \\ 1 - \sigma(\mathbf{X}_w^T\theta_{j-1}^w) & , d_j^w = 1 \end{cases} \\ &= \sigma(\mathbf{X}_w^T\theta_{j-1}^w)^{1-d_j^w}[1 - \sigma(\mathbf{X}_w^T\theta_{j-1}^w)]^{d_j^w} \end{aligned} \tag{3}$$

And the target loss can be formulated as:

$$\begin{aligned}
\mathcal{L} &= \sum_{w \in \mathcal{C}} log\, p(w|\text{context}(w)) \\
&= \sum_{w \in \mathcal{C}} log \prod_{j=2}^{l^w} \left\{ \sigma(\mathbf{X}_w^T \theta_{j-1}^w)^{1-d_j^w} [1 - \sigma(\mathbf{X}_w^T \theta_{j-1}^w)]^{d_j^w} \right\} \\
&= \sum_{w \in \mathcal{C}} \sum_{j=2}^{l^w} \left\{ (1 - d_j^w) log[\sigma(\mathbf{X}_w^T \theta_{j-1}^w)] + d_j^w log[1 - \sigma(\mathbf{X}_w^T \theta_{j-1}^w)] \right\}
\end{aligned} \tag{4}$$

Approximately, optimizing the loss for each item:

$$\mathcal{L}(w, j) = (1 - d_j^w) log[\sigma(\mathbf{X}_w^T \theta_{j-1}^w)] + d_j^w log[1 - \sigma(\mathbf{X}_w^T \theta_{j-1}^w)] \tag{5}$$

To optimize that, we find the partial derivative for $\mathbf{X}_w$ and θ_{j-1}^w, respectively.

$$\frac{\partial \mathcal{L}(w, j)}{\partial \theta_{j-1}^w} = [1 - \sigma(\mathbf{X}_w^T \theta_{j-1}^w) - d_j^w] \mathbf{X}_w^T \tag{6}$$

$$\frac{\partial \mathcal{L}(w, j)}{\partial \mathbf{X}_w} = [1 - \sigma(\mathbf{X}_w^T \theta_{j-1}^w) - d_j^w] \theta_{j-1}^w \tag{7}$$

Then, to update the representation of each context word, we add the gradient to each of them like:

$$\begin{gathered}
v(\tilde{w}) \leftarrow v(\tilde{w}) + \eta \sum_{j=2}^{l^w} \frac{\partial \mathcal{L}(w, j)}{\partial \mathbf{X}_w} \\
\text{where } \tilde{w} \in \text{context}(w).
\end{gathered} \tag{8}$$

By this way, we can learn the word embeddings by transforming the Softmax into a process of multi-binary classification problem with a computational overhead $O(log(V))$.