

Algorithm for the skipgram2vec model



let $U \in \mathbb{R}^{K \times N}$ be our center word embeddings corresponding to x_v

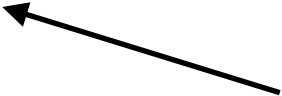
$$\forall (x_w, x_c) \in D_{\text{odo}}:$$

$$U_w \leftarrow U_w - \eta \nabla U_w NL$$

set $u_w = U_w$ via the non-zero indices of our one-hot center word x_w

$$\text{crossentropyloss: } L(U, V | \mathbf{x}_v, \mathbf{x}_c) \equiv -\mathbf{x}_c \cdot \log p(\mathbf{x}_c | \mathbf{x}_v; \mathbf{U}, \mathbf{V})$$

gradient descent:



$O(NK)$

let $V \in \mathbb{R}^{K \times N}$ be our centered word embeddings corresponding to \mathbf{x}_c

compute inner product between u_v and all context vectors $V: u_v \cdot V \in \mathbb{R}^N$

compute probability over all context words given center word : $\frac{\exp(\mathbf{u}_w \cdot \mathbf{V})}{\sum_{j=1}^N \exp(\mathbf{u}_w \cdot \mathbf{V})_j}$

gradients: $\nabla_{U_w} NLL = \mathbf{V} \cdot \left(P_{\mathbf{x}_c | \mathbf{x}_w} - \mathbf{x}_c \right)^T \in \mathbb{R}^K$

$$\nabla_v NLL = \mathbf{u}_v \cdot \left(P_{\mathbf{x}_c | \mathbf{x}_v} - \mathbf{x}_c \right) \in \mathbb{R}^{K \times N}$$

V ← V — n V V I N L L

encodes

distributional

hypothesis



only w^{th} row of U gets updated

entire Vgets updated

Algorithm for the skip gram word2vec model

let $\mathbf{U} \in \mathbb{R}^{K \times N}$ be our center word embeddings corresponding to \mathbf{x}_w

let $\mathbf{V} \in \mathbb{R}^{K \times N}$ be our center word embeddings corresponding to \mathbf{x}_c

$\forall (\mathbf{x}_w, \mathbf{x}_c) \in D$ do:

set $\mathbf{u}_w = \mathbf{U}_w$ via the non-zero indice of our one-hot center word \mathbf{x}_w

compute inner product between \mathbf{u}_w and all context vectors $\mathbf{V} : \mathbf{u}_w \cdot \mathbf{V} \in \mathbb{R}^N$

compute probability over all context words given center word :
$$\frac{\exp(\mathbf{u}_w \cdot \mathbf{V})}{\sum_{j=1}^N \exp(\mathbf{u}_w \cdot \mathbf{V})_j}$$

encodes
distributional
hypothesis

cross entropy loss: $\mathbf{L}(U, V | \mathbf{x}_w, \mathbf{x}_c) = -\mathbf{x}_c \cdot \log p(\mathbf{x}_c | \mathbf{x}_w ; \mathbf{U}, \mathbf{V})$

$O(NK)$

gradients: $\nabla_{U_w} NLL = \mathbf{V} \cdot (P_{\mathbf{x}_c | \mathbf{x}_w} - \mathbf{x}_c)^T \in \mathbb{R}^K$

$\nabla_V NLL = \mathbf{u}_w \cdot (P_{\mathbf{x}_c | \mathbf{x}_w} - \mathbf{x}_c) \in \mathbb{R}^{K \times N}$

gradient descent: $\mathbf{U}_w \leftarrow \mathbf{U}_w - \eta \nabla_{U_w} NLL$ only w^{th} row of \mathbf{U} gets updated

$\mathbf{V} \leftarrow \mathbf{V} - \eta \nabla_V NLL$ entire \mathbf{V} gets updated

For large N , we need to avoid the partition function

- The approach on the previous slide is the preferred method when N is a manageable size, say $N < 10^5$.
- When $N > 10^6$, the partition function (denominator of the softmax function) becomes prohibitively expensive to compute due the $O(NK)$ scaling.
- There are several approaches to get around having to compute the (full) partition function:
 - Hierarchical softmax
 - Importance sampling (IS)
 - Adaptive IS
 - Target sampling
 - Noise contrastive estimation (NCE)
 - Negative sampling



$\sim 10 - 100x$ speedup