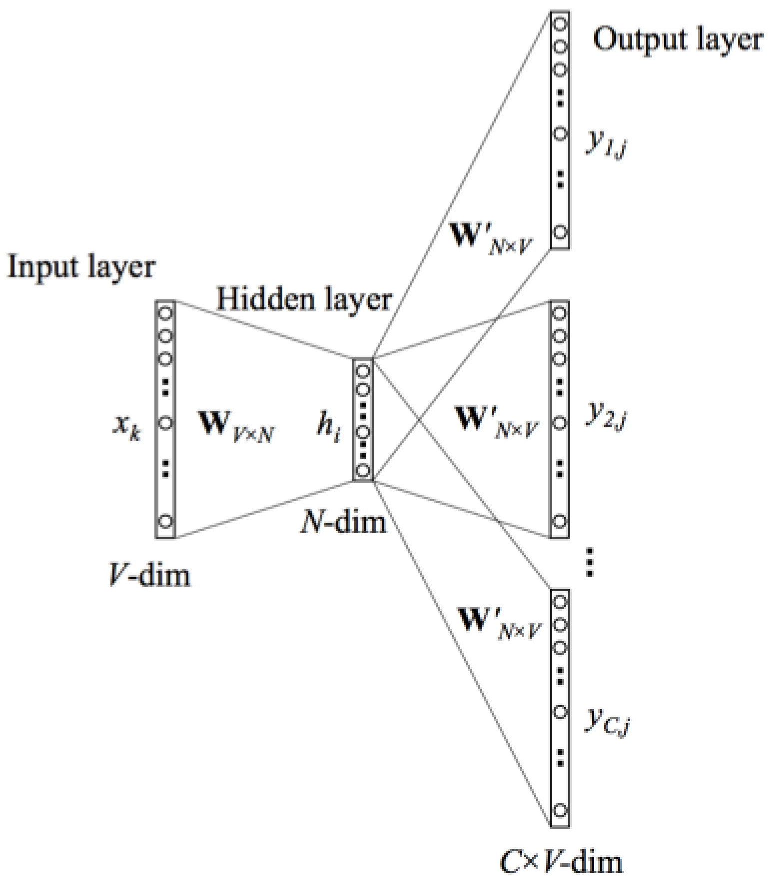
Learning Notes for Word2Vec

Skip-Gram Model



Predict context C given an input word

Training objective is to learn word vector representations that are good at predicting nearby words in the associated contexts

Input to hidden

$$\mathbf{h} = \mathbf{W}_{(k,.)} := v_{\mathrm{wI}}$$

Input each word x and retrieve their embedding $e^{[x]}$

Hidden to Output

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

$$u_{\mathrm{c,j}} = u_{\mathrm{c}} = \mathbf{v}'_{w_j}{}^{T} \cdot \mathbf{h}, \text{ for } c = 1, 2, \ldots, C$$

$y_{c,j}$ is the y_hat or y_pred, which is represented as a vector of normalized probabilities generated by softmax. The sum of these probabilities is equal to 1

v'_{wj} is the weight for output layer

$w_{c,j} = w_{o,c}$ implies that $w_{o,c}$ is the label (actual output), so the conditional probability actually tries to maximum the probability for each label output given the input w_I

Hidden to Output

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

(1) Provided by paper "How exactly does word2vec work"

$$\mathcal{L}^{(i)} = -\sum_{k=0}^{n_y - 1} Yoh_k^{(i)} * log(a_k^{(i)})$$

(2) Provided by Andrew Ng

Equation (1) and (2) actually talk about the same thing.

In Equation (2), Y_{ohk} represents the true label (one-hot encoding), and $ak^{(i)}$ is the softmax probability. And the purpose is to maximize the probability or minimize the loss for the labeled word correspondingly

Please see an example in next page

Hidden to Output

- Assume Y_{ohk} is one hot encoding [0 0 1 0 0]
- Assume $ak^{(i)}$ is probability [0.2 0.2 0.3 0.2 0.1]
- np.dot(Y_{ohk} , $ak^{(i)}$), non-labelled probability will be zero out and only keep the one that labelled as 1.
- So the purpose is actually try to maximize 0.3 for the third element in one hot encoding
- This is equivalent to explicitly mention $w_{o,c}$ in the conditional probability equation in (1)

Hidden to Output

- Let's go back to that paper, loss function is

$$
\begin{aligned}
E &= -\log p(w_{\mathrm{O},1}, w_{\mathrm{O},2}, \ldots, w_{0,\mathrm{C}} | w_{\mathrm{I}}) \\
&= -\log \prod_{c=1}^{C} \frac{exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} exp(u_{j'})} \\
&= -\sum_{c=1}^{C} u_{c,j_c^*} + C \cdot \log \sum_{j'=1}^{V} exp(u_{j'})
\end{aligned}
$$

Back propagation

- For output layer

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

$y_{c,j}$ is the y_hat or y_pred (form of probability)
$t_{c,j}$ is the true label (form of one-hot encoding)

Back propagation

- Update weight for Hidden->Output layer

$$\mathrm{EI}_j = \sum_{c=1}^{C} e_{c,j}$$

EI is the sum of the prediction errors over all context words

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{cj}} \cdot \frac{\partial u_{cj}}{\partial w'_{ij}} = \mathrm{EI}_j \cdot h_i$$

$$w'_{ij}{}^{(\text{new})} = w'_{ij}{}^{(\text{old})} - \eta \cdot \mathrm{EI}_j \cdot h_i$$

Back propagation

- Update weight for Input->Hidden layer

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i}$$
$$= \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) \cdot w'_{ij}$$

This step is actually the same as the one in last slide, computing the error derivative w.r.t the hidden layer

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}}$$
$$= \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) \cdot w'_{ij} \cdot x_k$$

Now we are able to compute the derivative w.r.t input layer weight matrix W

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \cdot \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) \cdot w'_{ij} \cdot x_j$$

Reference

- http://mccormickml.com/assets/word2vec/Alex_Minnaar_Word2Vec_Tutorial_Part_I_The_Skip-Gram_Model.pdf
- http://www.1-4-5.net/~dmm/ml/how_does_word2vec_work.pdf