# 1 Extra Credit Challenge I: Backward pass in V in Skip-Gram

The objective now is to get the gradient so we can update our word vectors to increase the probability.

$$\frac{\partial J}{\partial \mathbf{v_c}} = -\mathbf{u_o} + \frac{\partial}{\partial \mathbf{v_c}} \log\left(\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)\right) \tag{1}$$

We leave the left term for the end, and keep progressing with the other part.
Applying the chain rule on $f(x) = log$ and $g(x) = \sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{v_c}} \log\left(\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)\right) &= \frac{1}{\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)} \frac{\partial}{\partial \mathbf{v_c}} \sum_k \exp\left(\mathbf{u_k}^\intercal \mathbf{v_c}\right) \\ &= \frac{1}{\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)} \sum_k \frac{\partial}{\partial \mathbf{v_c}} \exp\left(\mathbf{u_k}^\intercal \mathbf{v_c}\right)\end{aligned}$$

Again, applying the chain rule on $f(x) = exp$ and $g(x) = \mathbf{u_k}^\intercal \mathbf{v_c}$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{v_c}} \log\left(\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)\right) &= \frac{1}{\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)} \sum_k \exp\left(\mathbf{u_k}^\intercal \mathbf{v_c}\right)\mathbf{u_k} \\ &= \sum_k \frac{\exp\left(\mathbf{u_k}^\intercal \mathbf{v_c}\right)}{\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)}\mathbf{u_k}\end{aligned}$$

Note the fraction was a constant. When including it inside the summation, we end up having a probability similar to what we have started with:

$$\frac{\partial}{\partial \mathbf{v_c}} \log\left(\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)\right) = \sum_k P(K|C)\mathbf{u_k}$$

Putting it all together:

$$\frac{\partial J}{\partial \mathbf{v_c}} = -\mathbf{u_o} + \sum_k P(K|C)\mathbf{u_k} = -\mathbf{u_o} + \sum_k \hat{\mathbf{y}} \cdot \mathbf{u_k} \tag{2}$$

**Matrix Notation:**

$\mathbf{u_o}$ can be written as a matrix multiplication $Uy$, since $y$ will only be 1 for the real output vector of $U$, $\mathbf{u_o}$

$\sum_k \hat{\mathbf{y}} \cdot \mathbf{u_k}$ is a linear transformation of vectors $\mathbf{u_k}$ in $U$ scaled by $\hat{\mathbf{y}}$. This can be written as $U\hat{\mathbf{y}}$.

Rewritten both of them, we show that (3) and (4) from the statement of the question were equivalent:

$$\frac{\partial J}{\partial \mathbf{v_c}} = -\mathbf{u_o} + \sum_k \hat{\mathbf{y}} \cdot u_k = U(\hat{\mathbf{y}} - \mathbf{y}) \tag{3}$$

**Alternative 1: Back-propagate through all layers**
The derivative of the error directly w.r.t ($\frac{\partial J}{\partial s_j}$) the scores is fairly simply, resulting on $\hat{y} - y$. This is derived in Appendix B.2, Eq 24.
Having obtained the equation for $U$ first (although it is the second question on this exercise), we can back-propagate now the error into the hidden layer **h**:

$$\frac{\partial J}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial J}{\partial s_j} \frac{\partial s_j}{\partial h_i} = \sum_{j=1}^{V} (\hat{y_j} - y_j) \cdot u_{ij} = \sum_{j=1}^{V} e_j \cdot u_{ij} := EH_i$$

$EH$ is an $N$-dim vector representing the sum of the output vectors of all the words in the vocabulary, weighted by their prediction error. Now that we have the error w.r.t the hidden layer, we can move onto $V$:

$$\frac{\partial J}{\partial v_{ci}} = \frac{\partial J}{\partial h_i} \frac{\partial h_i}{\partial v_{ci}} = EH_i \cdot x_c$$

This is equivalent to a tensor product:

$$\frac{\partial J}{\partial V} = x \otimes EH_i = xEH_i^T$$

Resulting on a $VxN$ matrix, same as $V$, fulfilling the *shape convention.*
**Only one component of x is non-zero, therefore only one column of $\frac{\partial J}{\partial V}$ will be zero.**

**Alternative 2: Using the computation graph and matrix notation**
Applying the chain rule entirely from the loss to the matrix $V$ we are interested in:

$$\frac{\partial J}{\partial V} = \frac{\partial J}{\partial s} \frac{\partial s}{\partial h} \frac{\partial h}{\partial V} = U(\hat{y} - y)$$

We can compute the local gradients during the forward pass (derived at Appendix A.3) such that:

$$\frac{\partial J}{\partial s} = \hat{y} - y; \frac{\partial s}{\partial h} = U; \frac{\partial s}{\partial V} = 1$$

**Interpretation**

$$\frac{\partial J}{\partial \mathbf{v_c}} = -\mathbf{u_o} + \sum_k \hat{y} \cdot \mathbf{u_k}$$

This has an interesting interpretation:
The gradient w.r.t. the input word vector, is the difference between the real output vector of the word and what our model thinks the context of what the center words should look like.
That "thinking" is an expectation, as a **weighted average of each word in the vocabulary, weighted by their probability of occurring in the context of that word**.

$$\frac{\partial J}{\partial h_i} = EH_i$$

and,

$$\mathbf{v_c}^{new} = \mathbf{v_c}^{old} - \eta EH_i$$

$EH$ is the sum of output vectors for all words in the vocabulary weighted by their prediction error $e_j = \hat{y_j} - y_j$.
We can understand above equation as adding a portion of every output vector to the input vector.

If $\hat{y_j} > y_j$ (*overestimating*):

The input vector $\mathbf{v_c}$ will tend to move farther from the output vector $u_j$.

If $\hat{y_j} < y_j$ (*underestimating*) which is only true for the real target $y_j = 1$:

The input vector $\mathbf{v_c}$ will tend to move closer to the output vector $u_j$.

The movement of the input vector $\mathbf{v_c}$ if determined by the prediction error of all vectors in the vocabulary. The larger the error, the more significant effects a word will exert on the movement of the input vector of the context word.

## 2 Extra Credit Challenge II: Backward pass in U Skip-Gram

Chain rule to compute the derivative of the cost w.r.t. each weight $u_{ij}$.
We know also that the shape of the gradients of $U$ has to be the same shape as $U$.

$$\frac{\partial J}{\partial u_{ij}} = \frac{\partial J}{\partial s_j}\frac{\partial s_j}{\partial u_{ij}} = (\hat{y}_j - y_j)h_i = (\hat{y}_j - y_j)h_i = e_j \cdot h_i \tag{4}$$

As can be seen in Fig 3 on Appendix A.3 for a specific output an entire row from $U$ is contributing. Generalizing to it:

$$\frac{\partial J}{\partial u_j} = e_j \cdot \mathbf{h} \tag{5}$$

Because the error $e_j$ depends on the real value, or ground truth $y$ (which is 0 for all entries expect 1 for entry $j = o$, and $\mathbf{h} = \mathbf{v_c}$we can decompose 5 into two depending on the value of $j$:

$$\frac{\partial J}{\partial u_w} = \begin{cases} (\hat{y_w} - 1)\mathbf{v_c}, & \text{if } w = o \\ \hat{y_w}\mathbf{v_c}, & \text{otherwise} \end{cases} \tag{6}$$

**Interpretation**
Note that the update equation tell us that:

$$u_{ij}^{new} = u_{ij}^{old} - \eta \cdot e_j \cdot h_i \tag{7}$$

$$\text{Or equivalently: } \mathbf{u_j}^{new} = \mathbf{u_j}^{old} - \eta \cdot e_j \cdot \mathbf{h}$$

If $\hat{y_j} > y_j$ (*overestimating*):

> Then we sustract a proportion of the hidden vector (or the center word embedding $\mathbf{v_c}$) from the output word embedding $u_j$, making $u_j$ farther away from $\mathbf{v_c}$.

If $\hat{y_j} < y_j$ (*underestimating*) which is only true for the real target $y_j = 1$:

> Then we sustract a proportion of the center word embedding $\mathbf{v_c}$ from the output word embedding $u_o$, making the two embeddings for the same word closer to each other.

# A Setup, Notation and Forward Pass

## A.1 Setup

We have one-hot encoded pairs of words: (input,output), (center,outside).
The goal is to create a model which is able to assign maximum probability for the outside words of a center word, given their co-occurrence alongside the text (or corpus).
The matrix $W$ projects words from the Vocabulary space into a smaller space of a chosen dimensionality $N$. That is the our embedding space.
The matrix $W'$ projects back into the Vocabulary space. These are the **scores** after a forward pass.
These scores are the input of a softmax layer, which turns them into the **predicted probabilities** $p$ or $\hat{y}$.

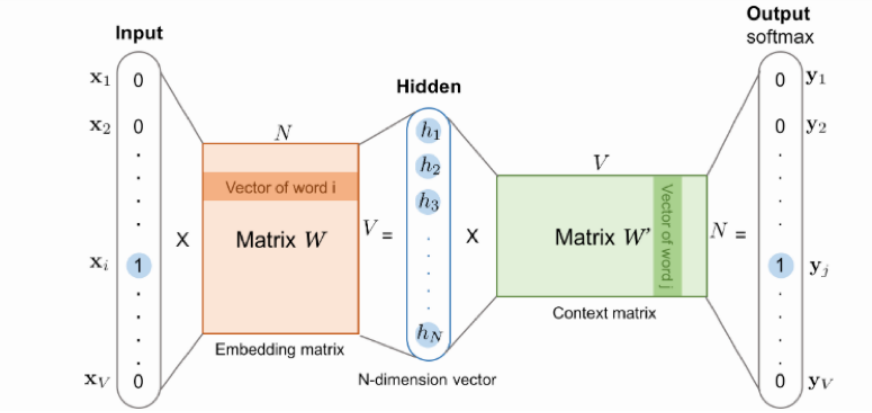


Figure 1: Skip-gram
Liliang Weng's Blog

## A.2 Notation

There are many different ways to name the variables of this setup.
We will first move into the exercise notation the convention in Fig. 1, which matches with this incredibly nice work on the math details behind skip-gram:

| Exercise | Figure 1 | Description |
|---|---|---|
| $x$ | – | One-hot encoded center (input) word |
| $V$ | $W^\top$ | Each column in V represents the input word embedding |
| $U$ | $W'$ | Each column in U represents the output word embedding |
| $\mathbf{v_c}$ | $W_{w_I}$ | Embedding for center (input) word with index $C$ |
| $\mathbf{u_o}$ | $W'_{w_O}$ | Embedding for context (output) word with index $O$ |
| $\mathbf{s_o}$ | – | Score for word with index $O$. $\mathbf{s_o} = \mathbf{u_o}^\intercal \mathbf{v_c}$ |
| $\hat{\mathbf{y}}_\mathbf{o} = \mathbf{p_o}$ | – | Probability for word with index $O$. $\hat{\mathbf{y}} = softmax(\mathbf{s})$ |

## A.3 Forward Pass

Our model predicts the probability of the output word $O$ given a center word $C$. $P(O|C)$.
For each word in the text as a center word, and for each word lying on a window around this center word, we want our model to give a high probability.
That probability is the result of the forward pass of our neural setup for the skip gram.
Let's make explicit the forward pass by taking a look at the computing graph of our setup:

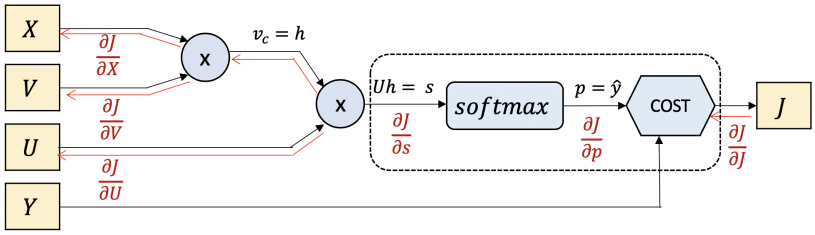


Figure 2: Skip-gram Computation Graph

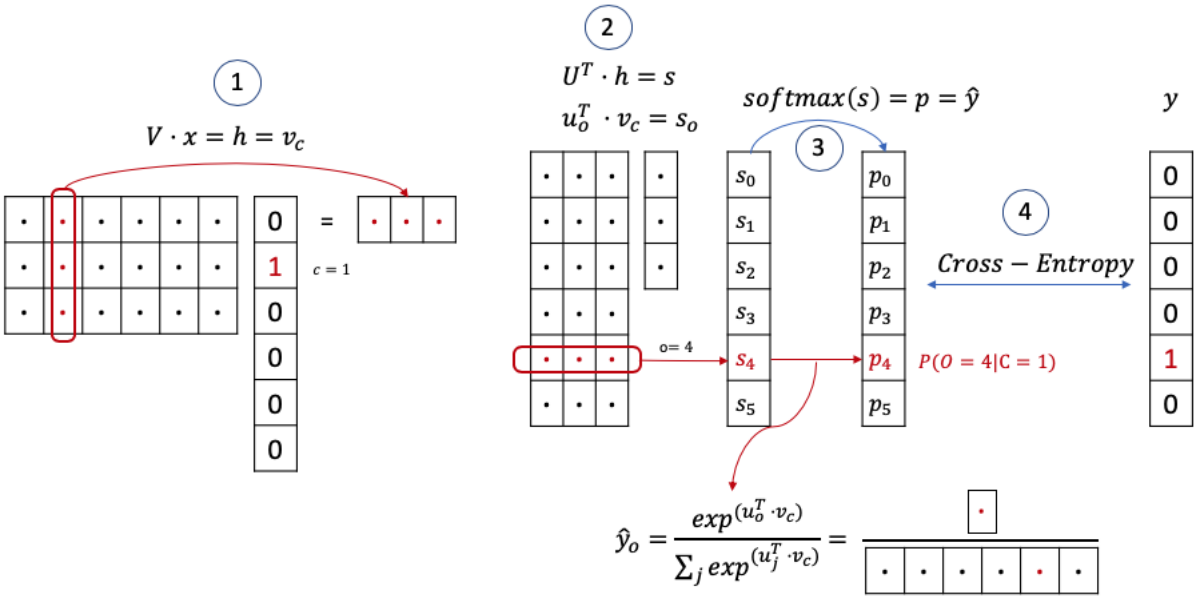


Figure 3: Skip-gram forward pass scheme for 1 prediction

$$\mathbf{h} = V\mathbf{x} := \mathbf{v_c} \tag{F1}$$

Or equivalently: $h_i = \sum_{j=1}^{V} x_i \cdot V_{ij}$

$$\mathbf{s} = U^T \mathbf{v_c} \tag{F2}$$

Or equivalently: $s_i = u_o^\intercal \mathbf{v_c} =$

$$\hat{\mathbf{y}} = softmax(\mathbf{s}) = \frac{\exp(\hat{\mathbf{s_i}})}{\sum_w \exp(\hat{\mathbf{s_w}})} = \frac{\exp(\mathbf{u_o}^\intercal \mathbf{v_c})}{\sum_w \exp(\mathbf{u_w}^\intercal \mathbf{v_c})} \tag{F3}$$

Therefore, a forward pass in our network setup outputs the probability of context (output) word to be in the context of a center (input) word:

$$\hat{\mathbf{y}} = P(O|C) = \frac{\exp\left(\mathbf{u_o}^\intercal \mathbf{v_c}\right)}{\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)}$$

In order to maximize that probability, we phrase it in terms of minimizing a cost function.
That cost function is the negative log loss of the prediction, which is equivalent to the cross-entropy loss between the real distribution and the prediction, since the real distribution is a one-hot vector, yielding to the negative log likelihood for the prediction at the index of the real word.

## A.4 Cost Function

$$\begin{aligned} J(\theta) = J(u_c, o, U) &= -\log \hat{y} = -\log\left(\frac{\exp\left(\mathbf{u_o}^\intercal \mathbf{v_c}\right)}{\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)}\right) \\ &= -\log\left(\exp\left(\mathbf{u_o}^\intercal \mathbf{v_c}\right)\right) + \log\left(\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)\right) \\ &= -\mathbf{u_o}^\intercal \mathbf{v_c} + \log\left(\sum_w \exp\left(\mathbf{u_w}^\intercal \mathbf{v_c}\right)\right) \end{aligned} \tag{F4}$$

# B Softmax & Cross-Entropy Backward Pass

In this section we derive mathematically the forward and backward passes of Softmax and Cross-Entropy:

## B.1 FORWARD PASS

**[1] Softmax** ⇒ Transforms scores into probabilities

$$p\left(y_i = c|X\right) = p_i = \frac{\exp\left(\overrightarrow{s_i}\right)}{\sum_{j=1}^{K} \exp\left(\overrightarrow{s_j}\right)} \tag{8}$$

**[2] Maximum Likelihood** ⇒ How close are these probabilities to the ground truth?

We are interested in the derivative of the loss with respect to the parameters.
This derivative is the one we will use to update the parameters to perform better next time.

We are modeling a categorical distribution and we know that the maximum likelihood solution for a it is equivalent to minimize the negative log-likelihood of our data w.r.t. the parameters, which is equivalent to minimize the cross entropy between the true distribution y and the distribution generated by our model .

Let's derive it. The likelihood function for a categorical distribution is:

$$P\left(Y|X,\theta\right) \overset{iid}{=} \prod_{n=1}^{N} p\left(y_n|X_n,\theta\right) = \prod_{n=1}^{N}\prod_{k=1}^{K} p_{nk}^{y_{nk}} \tag{9}$$

Where $N$ is the number of examples in the data set, and K the number of possible classes.
Taking the negative logarithm:

$$-\log p\left(y_i|X,\theta\right) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk}\log p_{nk} \tag{10}$$

Where $y_{nk}$ is a one-hot encoded vector, where every entry is zero except the $k$ -th entry, representing the $k$ class. That correspond to the cross-entropy.

## B.2 BACKWARD PASS

We are looking for $\frac{\partial L}{\partial s}$ to back-propagate the error. We can apply the chain rule to decompose:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial s}\frac{\partial s}{\partial \theta} = \frac{\partial L}{\partial s}\frac{\partial s}{\partial \theta} \tag{11}$$

where the first term is the derivative of the cross-entropy loss and the second term is the derivative of the softmax function. This is the reason why normally the derivative of the loss with respect to the scores is directly taken involving both softmax and cross-entropy together. This simplifies greatly the calculations

**[1] DERIVATIVE OF SOFTMAX**

$$\frac{\partial p_i}{\partial s_j} = \frac{\partial\left(\frac{\exp(s_i)}{\sum_{j=1}^{K}\exp(s_j)}\right)}{\partial s_j} \tag{12}$$

To differentiate the numerator, we can use:

$$f(x) = \frac{g(x)}{h(x)} \Rightarrow f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{h^2(x)} \tag{13}$$

Therefore, we have different solutions if $i = j$ or not, that we need to derive independently:

- Solution for $i = j$

$$\frac{\partial p_i}{\partial s_j} = \frac{\exp(s_i)\sum_{j=1}^{K}\exp(s_j) - \exp(s_j)\exp(s_i)}{(\sum_{j=1}^{K}\exp(s_j))^2} = \frac{\exp(s_i)(\sum_{j=1}^{K}\exp(s_j) - \exp(s_j))}{\sum_{j=1}^{K}\exp(s_j)\sum_{j=1}^{K}\exp(s_j)} = -p_i(1 - p_j) \tag{14}$$

- Solution for $i \neq j$

$$\frac{\partial p_i}{\partial s_j} = \frac{0 \cdot \sum_{j=1}^{K}\exp(s_j) - \exp(s_j) \cdot \exp(s_i)}{(\sum_{j=1}^{K}\exp(s_j))^2} = \frac{\exp(s_j)\exp(s_i)}{\sum_{j=1}^{K}\exp(s_j)\sum_{j=1}^{K}\exp(s_j)} = -p_j(p_i) \tag{15}$$

where in both cases we have made use of Equation 12.
If we combine both solutions making use of Kronecker Delta:

$$\frac{\partial p_i}{\partial s_j} = p_i \cdot (\delta_{ij} - p_j) \tag{16}$$

**[2] DERIVATIVE OF CROSS ENTROPY**

We have derived already the first component of the chain rule, the local gradient. We need know to compute the upcoming gradient to back-propagate it into the network so we can update their weights accordingly to their contribution to the error.
Starting with the NLL cost function for a single prediction, Equation 10:

$$J(\theta|y_i, X) = -\sum_{j=1}^{K} y_j \log p_j \tag{17}$$

$$\frac{\partial J}{\partial s_i} = -\sum_{j=1}^{K}\frac{\partial(y_j \log p_j)}{\partial s_i} = -\sum_{j=1}^{K} y_j \frac{\partial(\log p_j)}{\partial s_i} = -\sum_{j=1}^{K} y_j \frac{1}{p_i}\frac{\partial(p_j)}{\partial s_i} \tag{18}$$

Substituting 16:

$$\frac{\partial J}{\partial s_i} = y_j \frac{1}{p_i} \cdot (p_i \cdot (\delta_{ij} - p_j)) = y_j(\delta_{ij} - p_j) \tag{19}$$

If we decompose the sum now on terms where $i = j$ and $i \neq j$:

$$\frac{\partial J}{\partial s_i} = -y_j(1 - p_i) - \sum_{j \neq i}^{K} y_i(0 - p_i) = \tag{20}$$

$$-y_j(1 - p_i) + \sum_{j \neq i}^{K} y_j p_i = \tag{21}$$

$$y_i p_i - y_i + \sum_{j \neq i}^{K} y_j p_i = \tag{22}$$

$$p_i(y_i + \sum_{j \neq i}^{K} y_j) - y_i \tag{23}$$

Note how the above purple selection is 1 as it is the sum of all possible values of a one-hot encoded vector.

$$\frac{\partial J}{\partial s_i} = p_i - y_i = \hat{y}_i - y_i \tag{24}$$