Problems with count vectorizers:

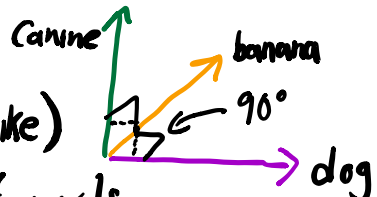
① each word is equidistant in its vector representation:

intuitively, we know dog should be more similar to canine than banana.

dog: [0 0 1]
banana: [1 0 0]
canine: [0 1 0]

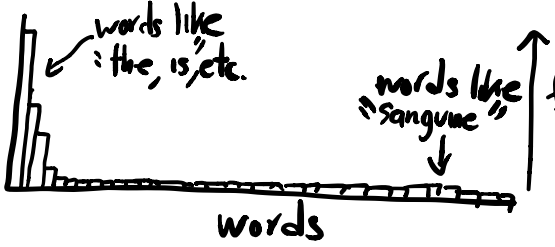
all dot products are the same (0).

The angles made by a pair of vectors are all orthogonal:



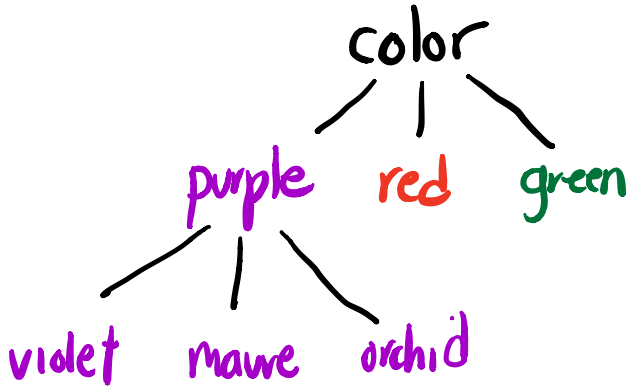
② counts skew towards high frequency stopwords (the, is, on, I, like)

Remember the long tail distribution of words we saw in Week 1 (Zipf's Law):



frequency (count of times a word appears in sentence)

Problems with rule-based domain trees:



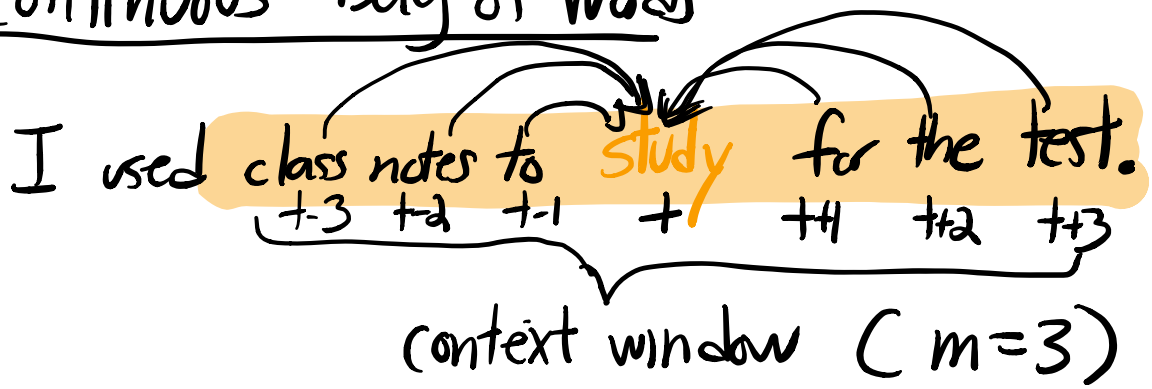
(ie. hyponyms and hypernyms)

- extremely labor intensive.
- slow to adapt to new words
- difficult to maintain
- "subjective"; two SMEs (subject matter experts) can have two totally valid trees.

So... what do we do?

We make a new assumption: the distributional hypothesis: "You shall know a word by the company it keeps" - JR Firth (British linguist).

Continuous Bag of Words



$P(x_t = study | x_{t-3} = class) x$

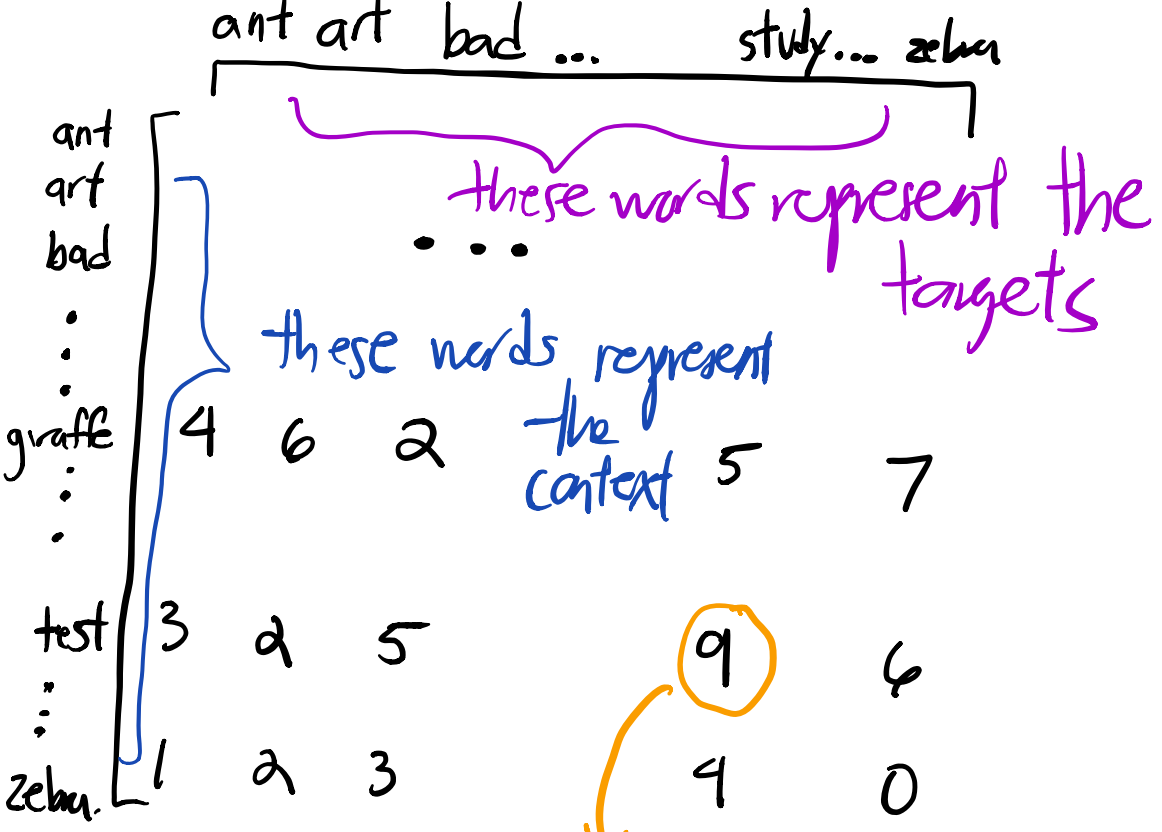
$P(x_t = study | x_{t-2} = notes) x$

...

$P(x_t = study | x_{t+3} = test)$

↖ probability that study is the target word given that the context word is test.

How would we find this probability?



$$\frac{\text{count(test, study)}}{\text{count(test)}} = \frac{9}{3+2+5+9+6} = \frac{9}{25} = .36$$

However, in practice, we typically do not use CBOW, but rather skipgram, which is the opposite of CBOW:

	Inputs	Target
CBOW	context	target word
skipgram	target word	context

$$P(X_{t-3}=\text{class} \mid X_t=\text{study})$$

$$P(X_{t-1}=\text{to} \mid X_t=\text{study}) \times \cdots \times P(X_{t+3}=\text{test} \mid X_t=\text{study}) \Rightarrow \prod_{-M \le m \le M,\ m \neq 0} P(X_{t+m} \mid X_t)$$

"for each context word in the window"

$M = 3$ → context window size is 3.

We do this for each target word:

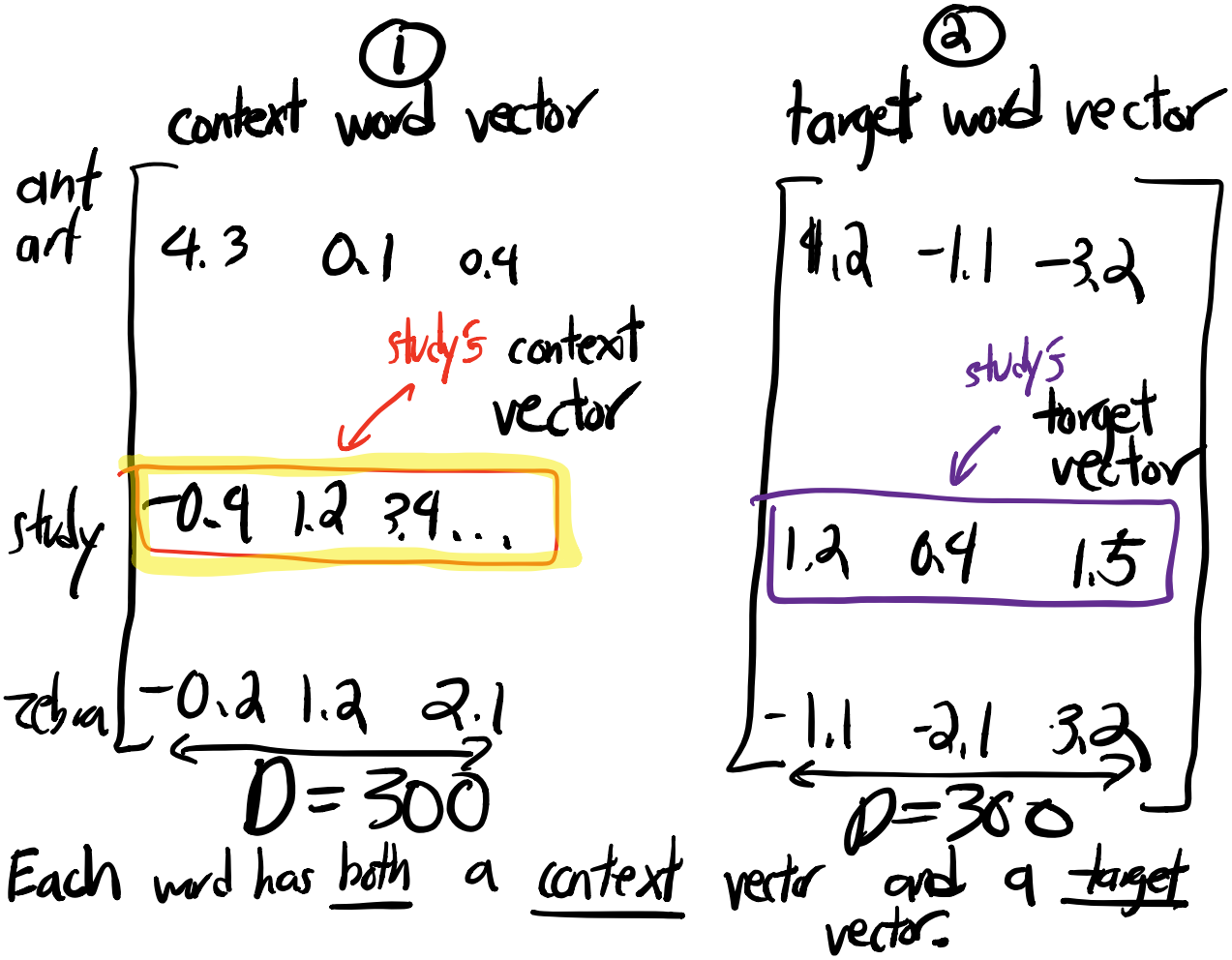
$$\prod_{t=1}^{\text{for each target word}} \prod_{-M \le m \le M,\ m \neq 0}^{\text{for each context word}} P(X_{t+m} \mid X_t, \theta)$$

- X_{t+m}: context word
- X_t: target word
- θ: weights (our parameters)

This becomes our objective function that we optimize (the higher this value the more "correct" our model is):

$$J(\theta) = \prod_{t=1}^{T} \prod_{\substack{-M \le m \le M \\ m \neq 0}} P(x_{t+m} | x_t, \theta)$$

So what is θ? It is 2 matrices:



Each word has <u>both</u> a <u>context</u> vector and a <u>target</u> vector.

Therefore, in skip gram, for each target word, we typically have 2x window size context words to predict on:

this is fed into the model { (study, to), (study, like), (study, midterm) } ← these are our Y variables

Our objective function:

① maximize probability:

$$J(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p(w_{t+j} | w_t, \theta)$$

② convert to negative log-likelihood:

$$\text{NLL } J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w_{t+j} | w_t, \theta)$$

$$\text{NLL } J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j} \mid w_t, \theta)$$

we converted $\prod \Rightarrow \sum$ so we end up with this log inside.

$p(w_{t+j} \mid w_t, \theta)$: probability of finding the word w_{t+j} in the same context window as the word w_t.

take the average over all T words in corpus

for each word in the corpus

for each context word of the word w_t

We attempt to minimize this NLL $J(\theta)$ our objective function.

How do we find $p(w_{t+j} \mid w_t, \theta)$?

w_t = study (our target word)

w_{t+j} = like (our context word)

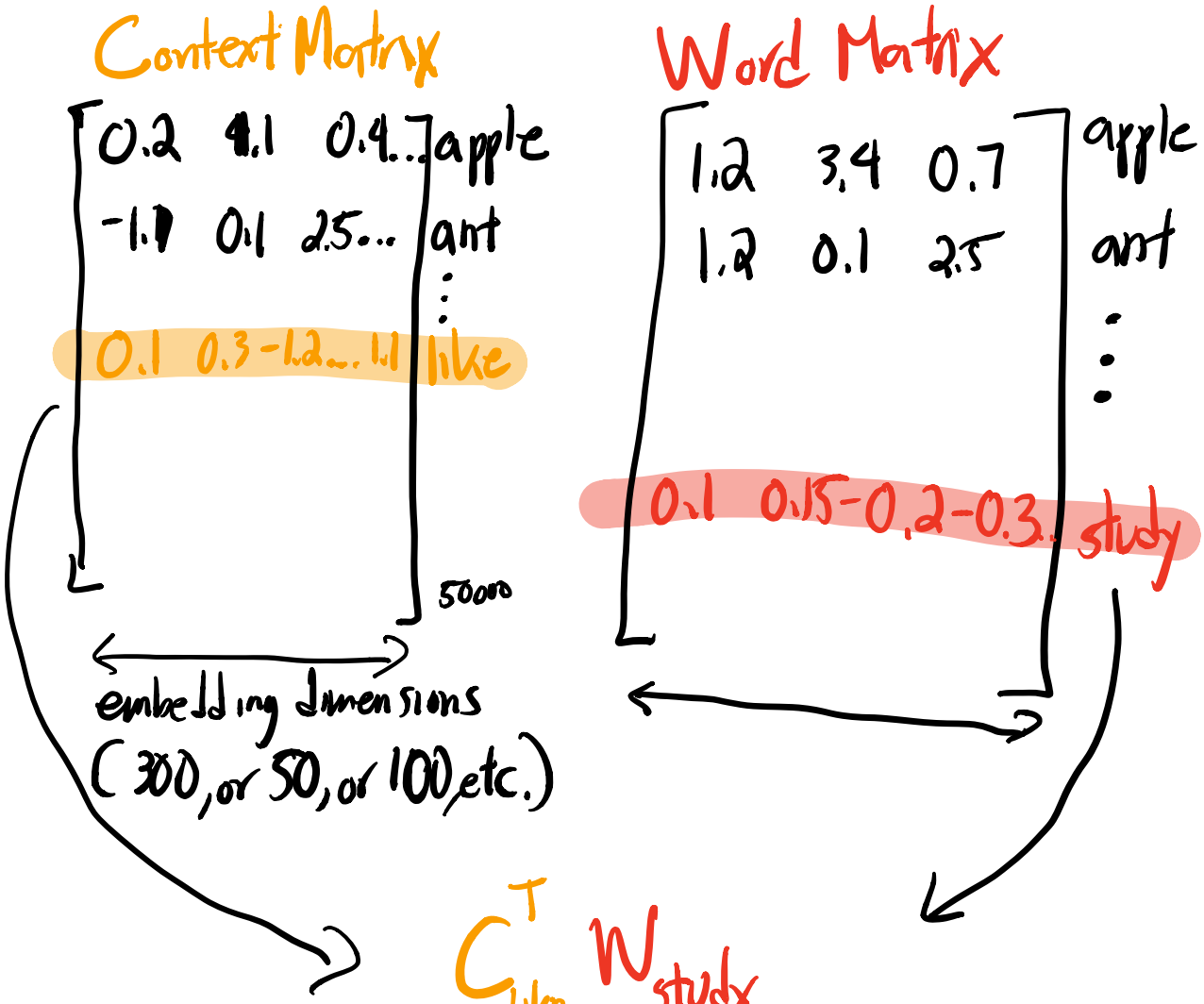
$\theta = \{W, C\}$

$W_{study} = [-0.1, 1.2, 2.7 \ldots 1.1]$ (our word vector for "study")

$C_{like} = [1.1, -0.2, 0.3, \ldots 0.4]$ (our context vector for "like")

our word matrix

our context matrix



$C_{like}^{T} W_{study}$

dot product, measures "agreement" between like and study. If dot product is high, then that context word is frequently co-occuring in the same context window as the target word. We can

use the dot product as a proxy for measuring the probability of finding the word like in the same context window as the word study.

However, there's two problems:

① dot products can be negative

② dot products can be ≥ 1

must be between 0 and 1 and must sum to 1.

So we use the Softmax Function:

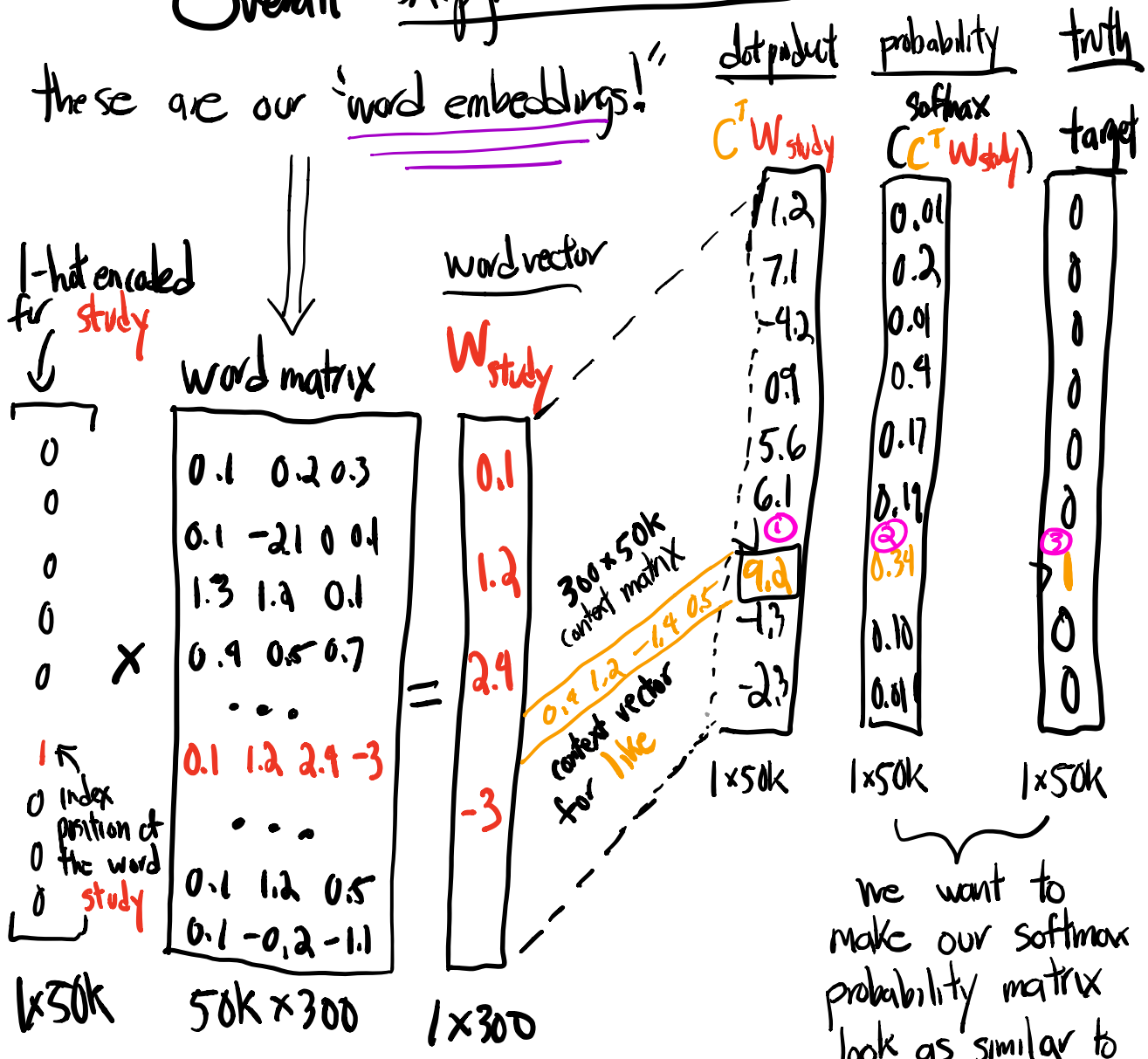
$$p(\text{like} \mid \text{study}) = \frac{\exp(C_{\text{like}}^T W_{\text{study}})}{\sum_{i=1} \exp(C_i^T W_{\text{study}})}$$

our original dot product

level of agreement between a random word i and study

we do this V times (50,000 times, for each word in vocab)

Overall skipgram word2vec architecture



① This is the dot product of the word vector for study and context vector for like.

② This is the predicted probability of seeing the word like in the context window of study

③ This is our target that we are optimizing for

You see in our first pass we had a prediction for $P(\text{like} \mid \text{study}) = 0.34$. The truth was 1. How do we update our parameters? Using derivatives! (more specifically, gradient descent)

$$NLL\ J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w_{t+j} \mid w_t, \theta)$$

$$\frac{\partial}{\partial w} \log p(w_{t+j} \mid w_t, \theta)$$

$$= \frac{\partial}{\partial w} \log \frac{\exp(C_{\sim t}^T w_t)}{\sum_i \exp(C_i^T w_t)}$$

$\sim t$ = a context word of the word w_t

Important identity:

$$\log\left(\frac{A}{B}\right) = \log(A) - \log(B)$$

Therefore,

① $\frac{\partial}{\partial w} \log \frac{\exp(C_{at}^T W_t)}{\sum_i (C_i^T W_t)} =$

② $\frac{\partial}{\partial w} \log \exp(C_{at}^T W_t) - \log \sum_i \exp(C_i^T W_t)$

$\log \exp(C_{at}^T W_t)$: these cancel each other out!

$\Downarrow$

$\frac{\partial}{\partial w} C_{at}^T W_t$

$\Downarrow$ (using chain rule of derivatives and skipping several steps)

③ $= C_{at} - \frac{\exp(C_i^T W_t)}{\sum_j \exp(C_j^T W_t)} C_i$

④ $= C_{at} - \sum_i p(W_{at} | W_t) C_i$

Important: in the lecture, I wrote W_t - this was NOT correct

C_{at}: the context target

$\sum_i p(W_{at} | W_t) C_i$: weighted average of all the context word probabilities

Since we are finding the derivative, and the min of this function comes when $\frac{\partial J(\theta)}{\partial \theta} = 0$, we are essentially looking to update W so that the difference between $C_{\sim t}$ and $\sum p(W_{\sim i} | W_i) \cdot C_i$ are equal. This is essentially saying we want to find the best configuration for the word embeddings that best predicts the target word W_t's context word $W_{\sim t}$