

Problems with count vectorizers:

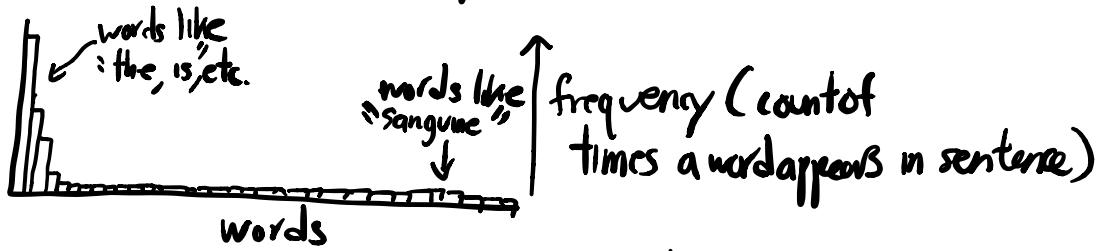
① each word is equidistant in its vector representation:

intuitively, we → dog: [0 0 1]
 know dog should
 be more similar
 to canine than banana:
 banana: [1 0 0]
 canine: [0 1 0]
 banana.

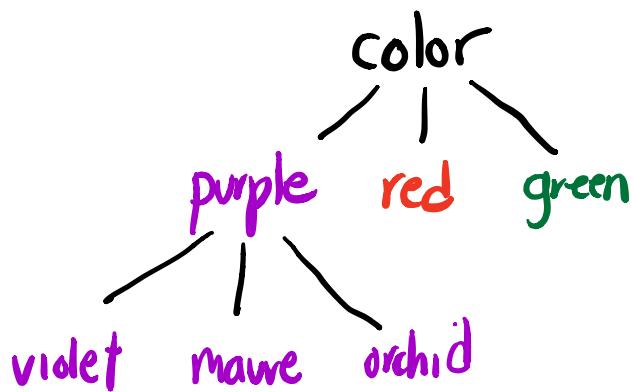
all dot products are
 the same (0).
 The angles made by a pair
 of vectors are all orthogonal:



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):



Problems with rule-based domain trees:



(i.e. 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

I used class notes to study for the test.

(+3 +2 +1) (+) (+1 +2 +3)

context window ($m=3$)

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

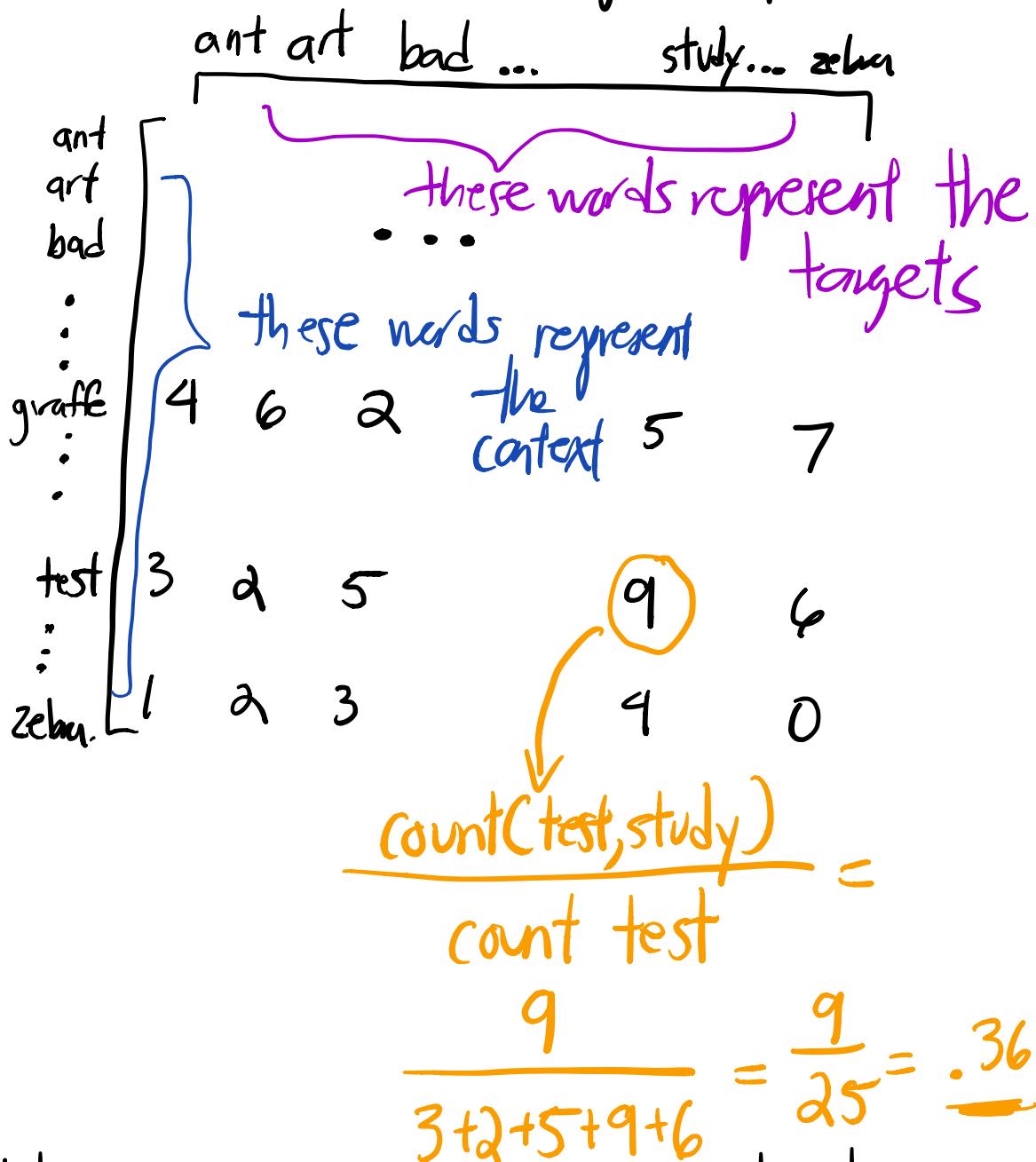
$$P(x_t = \text{study} | x_{t-2} = \text{notes})$$

...

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

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

How would we find this probability?



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

	<u>Inputs</u>	<u>Target</u>
CBOW	context words	target word
Skipgram	target word	context words
$P(X_{t+3} = \text{class} X_t = \text{study})$: for each context word in the window"

$P(X_{t-1} = t_6 | \dots | X_t = \text{study}) \times P(X_{t+3} = \text{test} | X_t = \text{study}) \Rightarrow \prod_{\substack{-M \leq m \leq M \\ m \neq 0}} P(X_{t+m} | X_t)$

$M = 3$

context window size is 3.

We do this for each target word : for each target word

$$\prod_{t=1}^T \prod_{\substack{-M \leq m \leq M \\ m \neq 0}} P(X_{t+m} | X_t, \theta)$$

↓ ↑ ↑
 context word target word Word vectors
 (our parameters,
 will learn
 more)

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

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

So what exactly is θ ? It's actually two matrices:

	context word vector	target word vector
ant	4.3 0.1 0.4	4.2 -1.1 -3.2
art	-0.9 1.2 3.4...	1.2 0.4 1.5
study	-0.2 1.2 2.1	-1.1 -2.1 3.2
zebra	D = 300	D = 300

Each word has both a context vector and a target vector.

Therefore, in skipgram, 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) ← these are our
variables
} (study, matter)

Our objective function:

① maximize probability:

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

② convert to negative log-likelihood:

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

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

we converted $\prod \rightarrow \sum$ so we end up with this log inside.
 $p(w_{t+j} | w_t, \theta)$
 probability of finding the word w_{t+j} in the same context window as the word w_t .
 for each context word of the word w_t
 for each word in the corpus
 +1 -m+j-m
 take the average over all T words in corpus

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

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

$w_t = \text{study}$ (our target word)

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

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

our word matrix

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

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

Context Matrix

0.2	0.1	0.4	..	apple
-1.1	0.1	2.5	..	ant
..
0.1	0.3	-1.2	..	like

Word Matrix

1.2	3.4	0.7	apple
1.2	0.1	2.5	ant
..

0.1 0.15 -0.2 -0.3 study

embedding dimensions
(300, or 50, or 100, etc.)

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

dot product, measures "agreement" between like and study. If dot product is high, then that context word is frequently co-occurring 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,

two issues:

- ① dot products can be negative
- ② dot products can be ≥ 1
probabilities must be between 0 and 1
and must sum to 1.

So we use the Softmax Function: our original

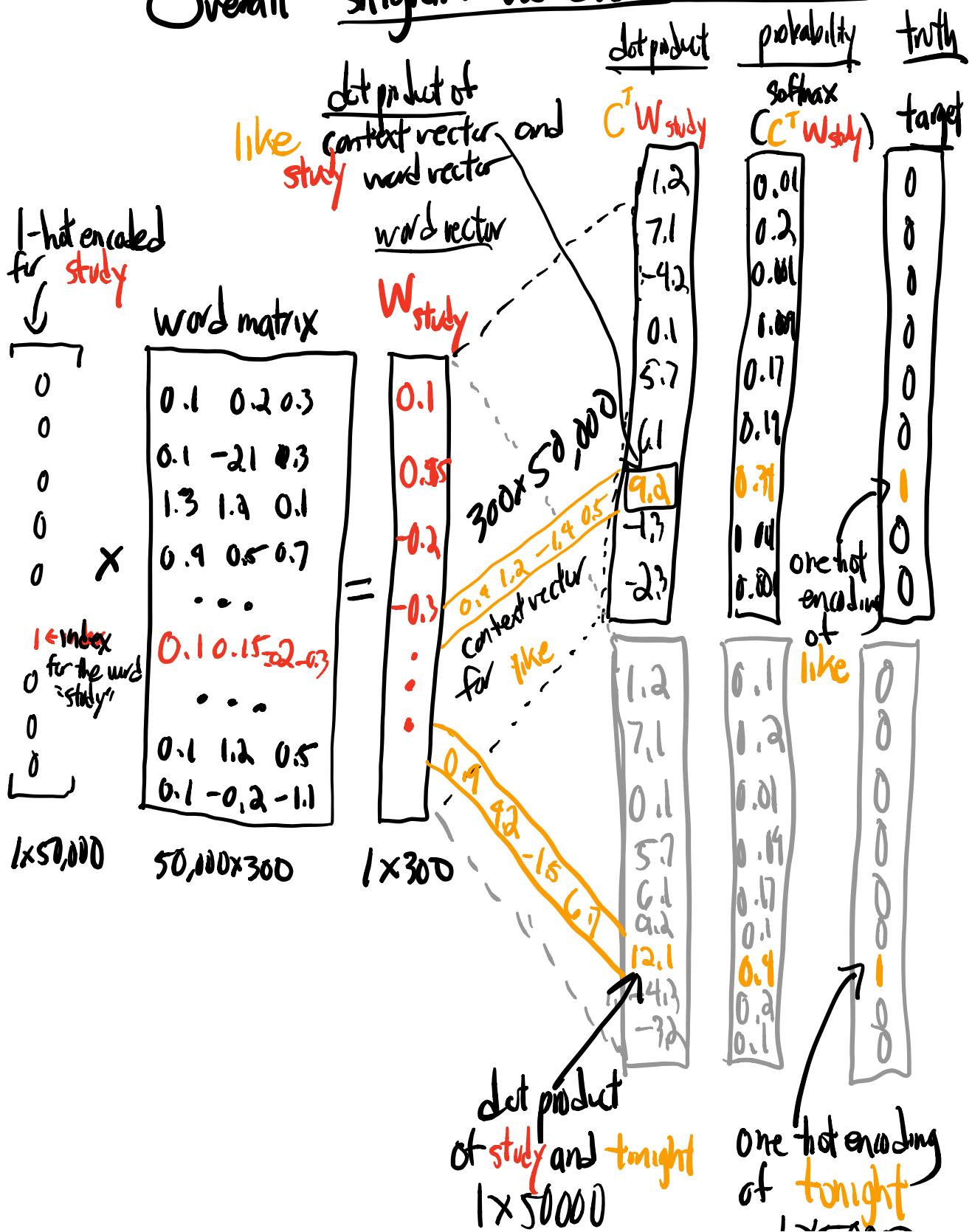
$$p(\text{like} | \text{study}) = \frac{\exp(C_{\text{like}}^T W_{\text{study}})}{\sum_{i=1}^V \exp(C_i^T W_{\text{study}})}$$

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

level of agreement
between a random word
 i and **study**

dot product

Overall skigram works architecture
highest probability



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

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

$$\begin{aligned} & \frac{\partial}{\partial \theta} \log p(w_{t+j} | w_t, \theta) \\ &= \frac{\partial}{\partial \theta} \log \frac{\exp(C_{\tilde{t}}^T w_t)}{\sum_i \exp(C_i^T w_t)} \end{aligned}$$

\tilde{t} = a context word of the word w_t

Important identity:

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

Therefore,

$$\textcircled{1} \frac{\partial}{\partial W} \log \frac{\exp(C_{nt}^T W_t)}{\sum_i (\exp(C_i^T W_t))} =$$

$$\textcircled{2} \frac{\partial}{\partial W} \log \underbrace{\exp(C_{nt}^T W_t)}_{\substack{\text{these cancel} \\ \text{each other} \\ \text{out!}}} - \log \sum_i \exp(C_i^T W_t)$$

↓

$$\frac{\partial}{\partial W} C_{nt}^T W_t$$

(using chain rule of
derivatives and skipping
several steps)

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

③

$$= C_{nt}$$

④

$$= C_{nt}$$

↑
the context
target

$$\begin{aligned} & - \frac{\exp(C_i^T W_t)}{\sum_j \exp(C_j^T W_t)} C_i \\ & - \sum_i p(W_{nt} | W_t) C_i \end{aligned}$$

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_{nt} and $\sum p(W_i | W_t) \cdot C_i$ 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_{nt} .