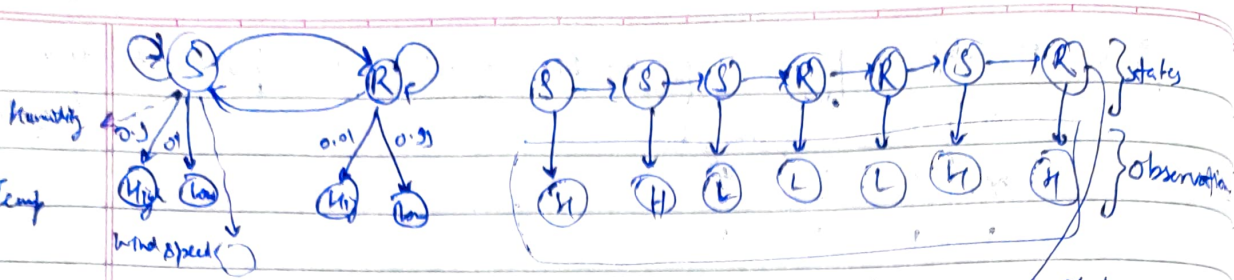
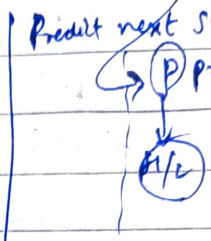
Hidden Markov Model



- Transition probability table
- Prior values
- Observation table (on sunny day $p(High) = 0.9$, $p(Low) = 0.1$; on rainy day $P(H) = 0.01$, $P(L) = 0.99$)

(states)	H (obse)	L
S	0.9	0.1
R	0.01	0.99

Predict next state



$P(S|SSSRRSRHHLLHH) = ?$

$P(R|SSSRRSRHHLLLHH) = ?$

Lec 15

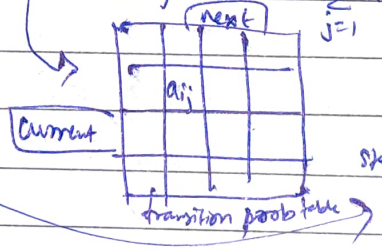
An HMM is specified by the following components

(Set of N states) $Q = q_1 q_2 \cdots q_N$ $\quad A = a_{11} a_{12} \cdots a_{nn}$

(Sequence of T observations) $O = O_1 O_2 \cdots O_T$

transition prob matrix A each a_{ij} represents of moving from state i to state j such that $\sum_{j=1}^{N} a_{ij} = 1 \ \forall i$

Sequence of obs likelihoods $B = b_i(O_t)$ an observation O_t being generated from state i



q_0, q_F (initial start) (final state / end)

priors (initial probability distribution over states) $\pi = \pi_2 \ \pi_1, \pi_2, \cdots \pi_N$

A set $QA \subseteq Q$ of legal accepting states $QA = \{q_a, q_y \cdots\}$

markov assumption :- $P(q_i | q_1 q_2 \cdots q_{i-1}) = P(q_i | q_{i-1})$

output Assumption :- $P(O_i | q_1 \cdots q_i O_1 \cdots O_i O_T) = P(O_i / q_i)$

Problem 1 (likelihood): Given an HMM $\lambda = (A, B)$ and an observation seq O, determine the likelihood of $P(O|A)$

Problem 2 (decoding): given an observation sequence O and HMM $\lambda = (A, B)$ discover the best state sequence Q.

Problem 3 (learning): Given an observation sequence O and the set of states Q, learn the HMM parameter A and B

Problem 1 (likelihood) Given :- HMM = $\lambda = (A, B)$
Observation sequence O

Find likelihood $P(O|\lambda)$

Transition Observation

[A] [B]

no of ice cream eaten by Jason = O (observation sequence)

3, 1, 3

$P(3, 1, 3|\lambda) = ?$

Problem 2 (Decoding) Given :- HMM = $\lambda(A, B)$

Observation seq O,

discover best hidden state seq Q

O → 3, 1, 3

best P(States | 3, 1, 3)

$2^n = 2^3 = 8$ patterns

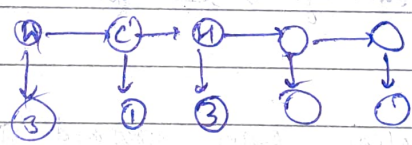
- P(hot hot hot | 3, 1, 3)
- P(hot hot cold | 3, 1, 3)
- ...
- P(cold cold cold | 3, 1, 3)

Problem 3 (learning of HMM) given Observation seq O and set of states.

learn HMM parameters A and B

S → hot cold hot ---- cold, hot cold hot hot

O → 3 1 3 ---- 1, 6, 2, 8, 9

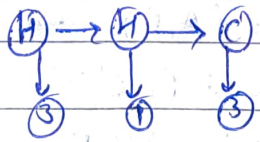


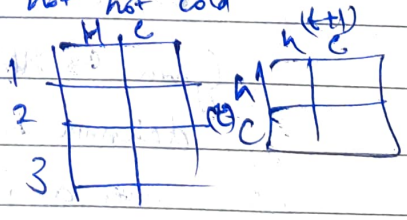
[A?] [B?]

problem 1 given $\lambda = (A, B)$, O = 3, 1, 3

$P(O|A) = ?$ ~~let states~~ Q

Assumption: Suppose corresponding seq of state is hot hot cold





$P(3, 1, 3 | HHC) = P(3|H) \cdot P(1|H) \cdot P(3|C)$

Since we don't know the hidden state seq, we need to compute the probability of 3,1,3 by summing over all possible state sequences weighted by their probability

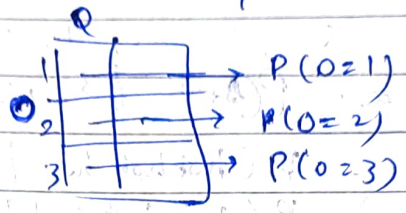
$$P(O,Q) = P(O|Q) \cdots P(Q) = \prod_{i=1}^{N} P(O_i|q_i) \cdot \prod_{j=1}^{N} P(q_i|q_{i-1})$$

$P(3,1,3 \mid \text{hot hot cold}) = P(H|\text{start}) \cdot P(H|H) \cdot P(C|H)$

$P(3|H) \cdot P(1|H) \cdot P(3|C)$

$$P(O) = \sum_{Q} P(O,Q)$$

$$P(O) = \sum_{Q} P(O|Q) \cdot P(Q)$$



$P(3,1,3) = P(3,1,3 \mid \text{hot hot hot}) \cdot P(\text{hot hot hot}) +$

$P(3,1,3 \mid x\,x\,x) \cdot P(x\,x\,x) +$

$P(3,1,3 \mid y\,y\,y) \cdot P(y\,y\,y) +$ ... (8 sequences)

This is not practically feasible (computationally)
direct method

SOP is dynamic programming based forward algorithm.

Lec 16

Likelihood computation using forward algorithm computing recursively the probabilities of observations, one-by-one it proceeds to the states and calculates the obs probabilities with the help of prev state observation probabilities

Given an HMM $\gamma = (A, B)$ and observation seq O determine the likelihood $P(O|\gamma)$

Eg: given an icecream eating HMM, what is probability of 3,1,3 (observation seq.) icecreams are sold. (day 1 = 3)

The problem with hidden states is that we don't know the underlying seq of states.

Given $Q = q_0 \cdot q_1 \cdots q_T$ and $O = o_1, o_2 \cdots o_T$

P(the likelihood of observations given states

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

eg $P(3\,1\,3 \mid \text{hot hot cold}) = P(3|\text{hot}) \cdot P(1|\text{hot}) \cdot P(3|\text{cold})$

$$P(O|Q) = P(O|Q) \cdot P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} (q_i|q_{i-1})$$

eg $P(3\,1\,3, \text{hot hot cold}) = P(\text{hot}|\text{start}) \cdot P(\text{hot}|\text{hot}) \cdot P(\text{cold}|\text{hot})$
$\cdot P(3|\text{hot}) \cdot P(1|\text{hot}) \cdot P(3|\text{cold})$

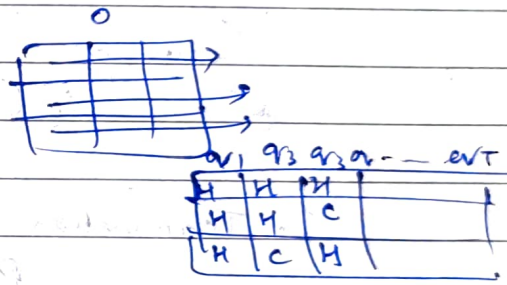
we know that $P(A) = \sum_{B} P(A, B)$

$$P(O) = \sum_{Q} P(O, Q)$$

$$= \sum P(O|Q) \cdot P(Q)$$

$$= \cancel{P(O_1|q_1) \cdot P(q_1) \cdot}$$

$$P(o_1 o_2 \cdots o_T | q_1 q_2 \cdots q_T) \cdot P(q_1, q_2 \cdots q_T)$$

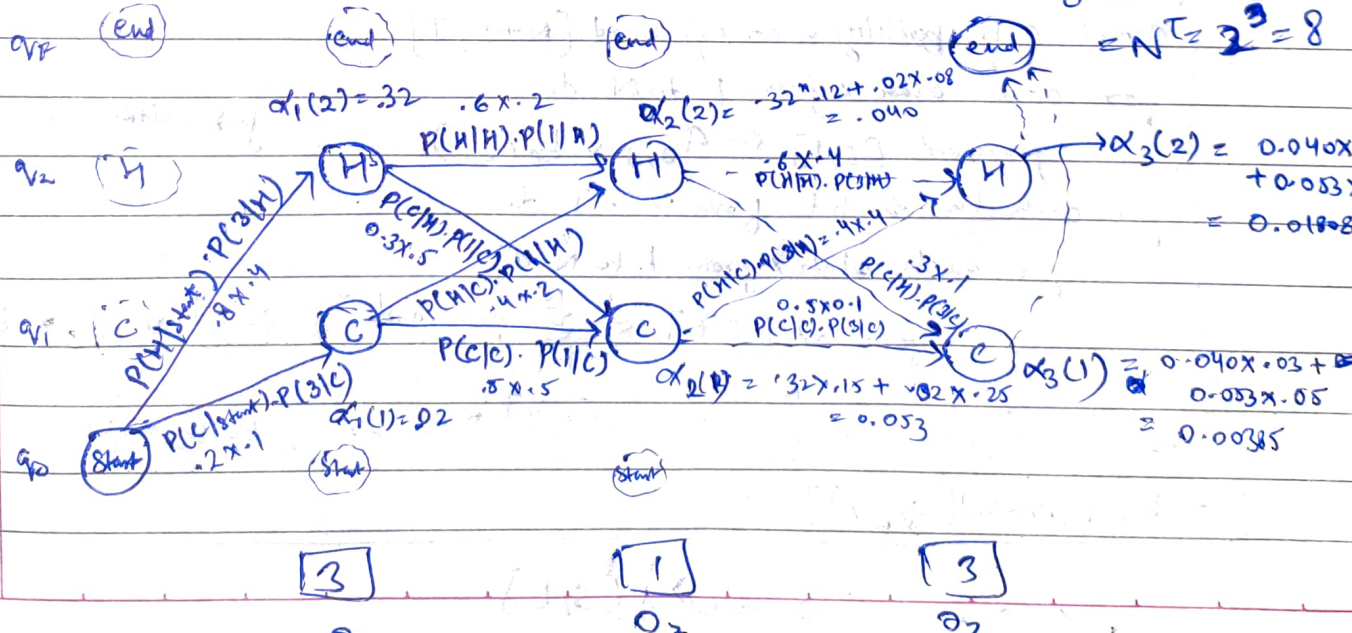


$P(3,1,3) = P(3|3, \text{hot hot hot}) + P(3|3, \text{cold, cold, cold})$

8 combination

$= N^T = 2^3 = 8$

Lecture - 17



Date
DELTA Pg No.

$\alpha_t(j)$ = represents the probability of being in state j after seeing t observations given λ (probability table for markov model)

$$\alpha_t(j) = P(o_1 o_2 \cdots o_t, q_t = j \mid \lambda)$$

↳ the t^{th} state in the seq of states is state j

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \left[a_{ij} \, b_j(o_t) \right]$$

a_{ij}: transition prob; $b_j(o_t)$: observation probability

1. Initialisation

$$\alpha_1(j) = a_{0j} \, b_j(o_1) \qquad 1 \le j \le N$$

2. Recursion

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \, a_{ij} \, b_j(o_t) \qquad 1 \le j \le N, \quad 1 \le t \le T$$

3. Termination

$$P(O \mid \lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) \, a_{iF}$$

function Forward (observations of len T, state-graph of len N) returns forward prob

Create a probability matrix forward [N+2, T]

for each state s from 1 to N do

    forward [s, 1] ← $a_{0,s} * b_s(o_1)$] → initialise

for each time step t from 2 to T do] recursion

    for each state s from 1 to N do

        forward [s, t] ← $\sum_{i=1}^{N}$ forward [s', t-1] $* a_{s',s} * b_s(o_t)$

forward [q_F, T] ← $\sum_{i=1}^{N}$ forward (s, T) $* a_{s,q_F}$] termination

return forward [q_F, T]

$\alpha_t(s)$ ⟶ forward [s, t]

Lecture 18 - Vector Semantics

We define a word as a vector

Called an "embedding" because it's embedded into a space.

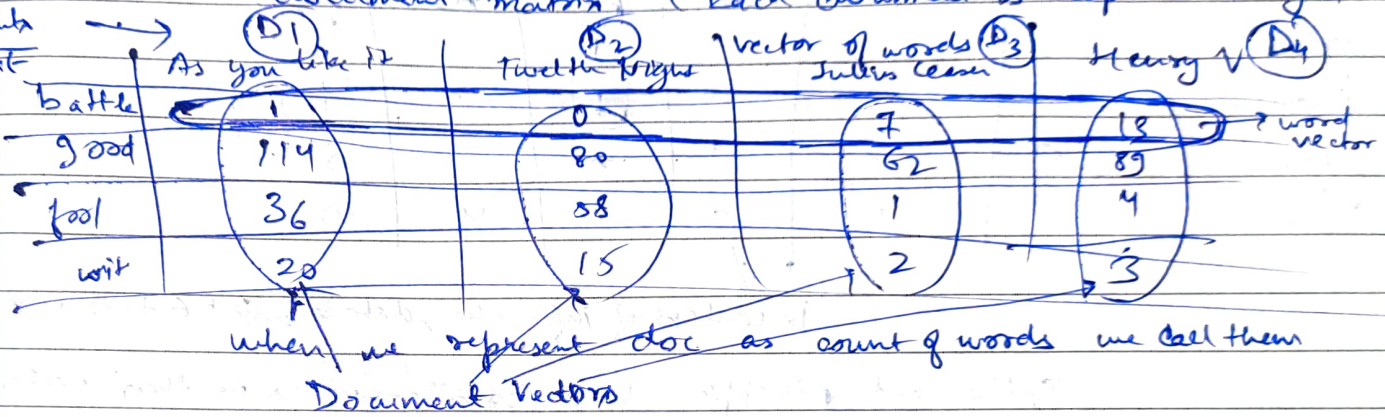
The standard way to represent meaning in NLP

Fine-grained model of meaning for similarity.

- NLP tasks like sentiment analysis
  - With words, requires same word to be in training and test.
  - With embedding : ok if similar words occurred.

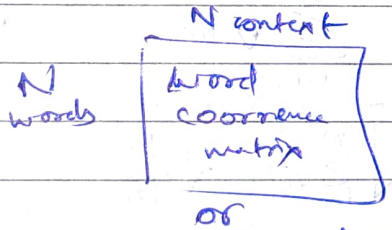
~~Words~~

Term - document matrix (Each document is represented by a vector of words)



battle = 1, 0, 7, 13 word vector

All the other words present in D_1 are known as context of ~~battle~~ battle

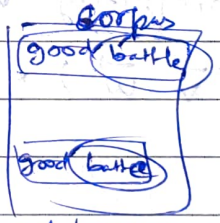


or

word word matrix

or

word context matrix

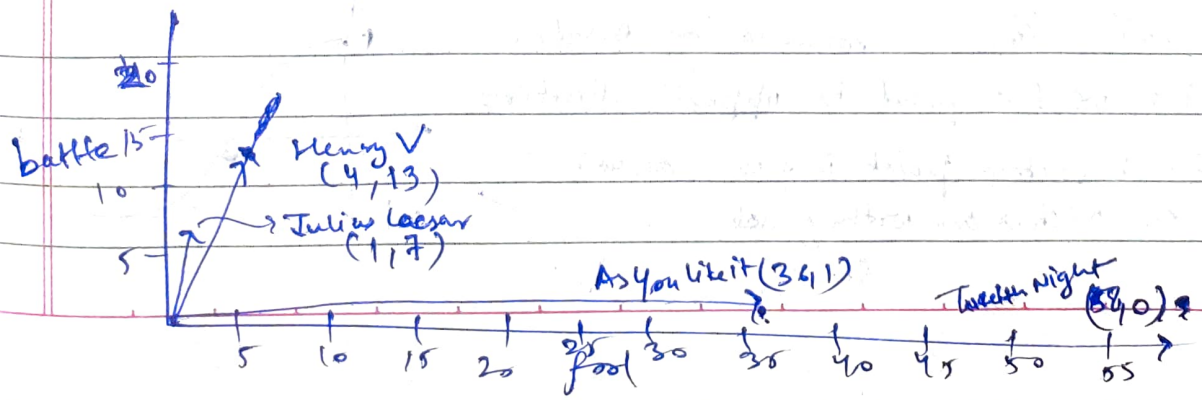


good is the context

battle is word

battle is ~~good~~ in context with good 2 times

rows are known as vectors or word matrix



Date
DELTA Pg No.

Word - word matrix (or term - context matrix)

Two words are similar in meaning if their context vectors are similar.

context of apricot: Sugar, a sliced lemon, a tablespoonful of apricot jam, a pinch ea

ayment. Cautiously she sampled her first pineapple and another fru

well suited to programming on the digital computer. In finding the c

for the purpose of gathering data and information necessary for th.

	aardvark	computer	data	pinch	result	sugar	...... more words
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

in context of information, data is present 6 times

Apricot and pineapple have similar values in their vectors then we say they words are similar

⇒ apricot is similar to pineapple

⇒ Cosine for computing word similarity

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

v_i = count for word v in context i

w_i = count for word w in context i

$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta$

$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$

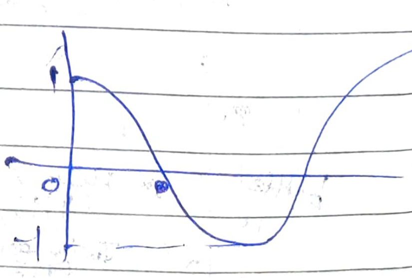
$\cos(\vec{v}, \vec{w})$ is the cosine similarity of $\vec{v}$ and $\vec{w}$

⇒ Cosine as a similarity metric

-1 : vectors point in opposite directions

+1 : vectors point in same direction

0 : vectors are orthogonal



	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

$$\text{cosine}(\text{apricot}, \text{information}) = \frac{1+0+0}{\sqrt{1+0+0}\cdot\sqrt{1+36+1}} = \frac{1}{1\cdot\sqrt{38}} = 0.16$$

$$\text{cosine}(\text{digital}, \text{information}) = \frac{0+6+2}{\sqrt{0+1+4}\cdot\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\cdot\sqrt{5}} = 0.58$$

$$\text{cosine}(\text{apricot}, \text{digital}) = \frac{0+0+0}{\sqrt{1+0+0}\cdot\sqrt{0+1+4}} = 0$$

raw freq is bad representation

tf-idf : value for the word t in document d : $w_{t,d} = tf_{t,d} \times idf_t$

term freq – inverse document freq

words like "the" or "good" have very low idf

(pointwise mutual information) $PMI(w_1, w_2) = \log \frac{p(w_1, w_2)}{p(w_1)\cdot p(w_2)} = \frac{\text{probability of occurring together}}{\text{probability of occurring independently}}$

See if words like "good" appear more often with "great" than we would expect by chance

lecture 13

PMI bw a target word 'w' and context word 'c' may be defined as

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)\cdot P(c)}$$

Co-occurring things less often leads to -ve PMI values, However, we are concerned with co-occurring thing very often.

⇒ Use Positive PMI, $PPMI(w, c) = \max\left(\log_2 \frac{P(w, c)}{P(w)\cdot P(c)}, 0\right)$

Given a co-occurrence matrix with W rows of words and C colums of contexts and f_{ij} is the no of times words w_i occurs in context c_j. Then PPMI matrix :–

$$PPMI_{ij} = \max\left(\log_2 \frac{p_{ij}}{p_{i*}\, p_{*j}}, 0\right)$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$P(w = \text{information}, c = \text{data}) = \frac{3982}{11716} = 0.3399$$

$$P(w = \text{information}) = \frac{7703}{11716} = 0.6575$$

$$P(c = \text{data}) = \frac{5673}{11716} = 0.4842$$

$$\text{ppmi}(\text{information}, \text{data}) = \max\left(\log_2 \frac{0.3399}{0.6575 \times 0.4842}, 0\right) = \max(0.0944, 0) = 0.0944$$

Lecture 20

- Vectors based on term-freq (counts of words in a corpus wrt to each document)
- Vectors based on Positive PMI

Characteristics :- - Sparse vectors (large no. of zero entries)
- Large size vectors (length of vector is equal to vocabulary size |V|)
- However, we can reduce the size using dimensionality reduction techniques such as PCA, SVD etc.

Word2Vec – uses a technique called skip gram
– negative sampling

Aim :- Instead of counting the values or calculating some values, train a classifier

Task :- Prediction – Is a target word 't' is likely to show up near the context 'c'?

However, we skip the prediction and the weights of the learned classifier will be used as embeddings.

For a classifier, we require a supervised or labelled training data.

The running text from the corpus can be considered as supervised training data.

running text from corpus.

Skip grams follows the steps as

1) Treat the target word t and its neighbouring context c as +ve sample (↑ ± 2 window size)
2) Randomly use the other words in the vocab to obtain the negative sample
3) Train the classifier such as logistic regression
4) Use the regression weights as embeddings.

The classifier.

Train a classifier such that the tuple (t, c) generate the probabilities as $P(+|t,c)$

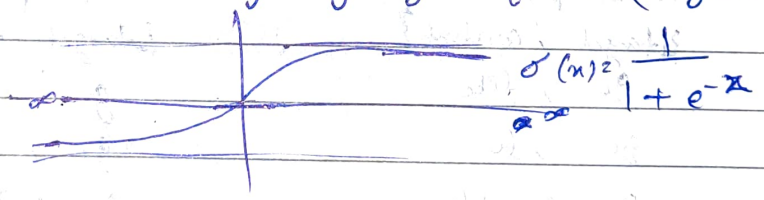
$$P(-|t,c) = 1 - P(+|t,c)$$

a target word is likely to appear near the context, if its embedding is similar to the context embedding

$$\text{Similarity}(t,c) \propto t \cdot c \text{ (dot product)}$$ // for $-\infty$ to ∞

is used in place of

We bound this similarity using logistic function (sigmoid function)



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$P(+|t,c) = \frac{1}{1+e^{-t \cdot c}}, \quad P(-|t,c) = 1 - P(+|t,c)$$

$$= 1 - \frac{1}{1+e^{-t \cdot c}}$$

$$= \frac{e^{-t \cdot c}}{1+e^{-t \cdot c}}$$

positive examples +		negative examples −	
t	c	t	c
apricot	tablespoon	apricot	aardvark
apricot	of	apricot	my
apricot	jam	apricot	where
apricot	a	apricot	coaxial
		apricot	seven
		apricot	forever
		apricot	dear
		apricot	if

Date
DELTA Pg No.

$$P(+ \mid t, c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1 + e^{-t \cdot c_i}}$$

$$\log P(+ \mid t, c_{1:k}) = \log \sum_{i=1}^{k} \log \frac{1}{1 + e^{-t \cdot c_i}}$$

Since all the context words of target word are independent of each other we can multiply the probabilities together

$$L(\theta) = \sum_{(t,c) \in +} \log P(+ \mid t, c) + \sum_{(t,c) \in -} \log P(- \mid t, c)$$

(log function)

Maximize — minimize

Optimizer like gradient descent may be used to optimize the weights or specifically, the randomly generated embeddings

Lecture 21

I would like to go someplace nearby for lunch

the sample {target, context}
go, like
go, to
go, someplace
go, nearby

the sample {target, context}
go, I
go, lunch
go, twelve
go, therefore

$Sim(t,c) = t \cdot c =$