Lecture 7: Word Representations

Practical Natural Language Processing

Rasika Bhalerao
r.bhalerao@northeastern.edu

Reminders

- Project proposal due today
- Assignment 5 due Mar 2
- (Spring break is Mar 4 - 12)
- Midterm exam is Mar 14 - 16
- Assignment 6 due Mar 23

About the midterm exam

- We will go over the practice midterm exam solutions today
- The midterm exam will be available on Canvas at the beginning of March 14
- It will be due 11:59pm March 16
- There will be no lecture on March 16

What did we do last week?

- Coreference resolution (determining which terms co-refer to the same NP)
- Unsupervised learning
  - Clustering
    - Flat (iterative until you get one solution, like KMeans)
    - Hierarchical (gives a tree of nested clusters)
  - Topic modeling via LDA
    - Analyze “themes” in document collection
    - Understand which topics are present in each document
    - Use LDA vectors anywhere you need document vectors
  - Dimension reduction via PCA
    - Understand which dimensions are most important
    - Reduce data down to 2 most important dimensions for plotting

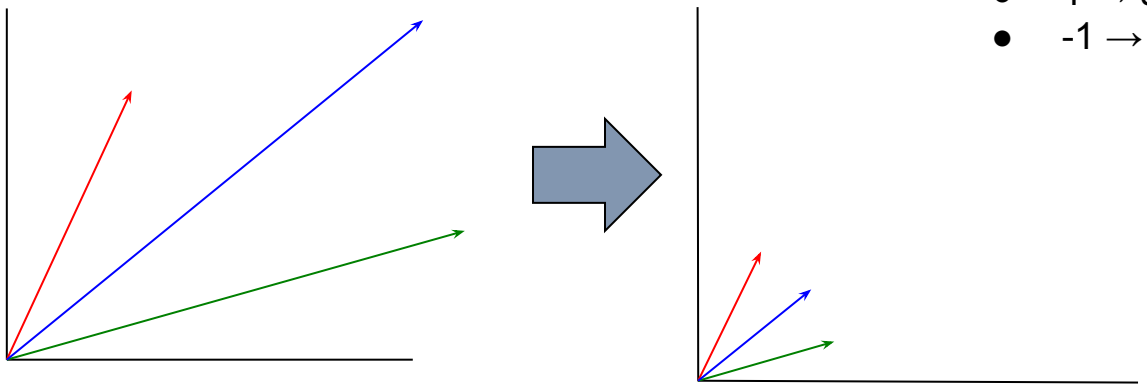
Remember: Cosine Similarity

- A common measurement in NLP is cosine similarity

$$sim(x,y) = \frac{x \bullet y}{|x||y|} = \frac{x}{|x|} \bullet \frac{y}{|y|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

- Measures the angle between two vectors

- 0 → orthogonal
- 1 → same direction
- -1 → opposite direction



Today’s agenda

- Word Representations
  - J&M chapter 6
- Intro to Neural Networks
  - Start of J&M chapter 7
- Discuss practice midterm exam

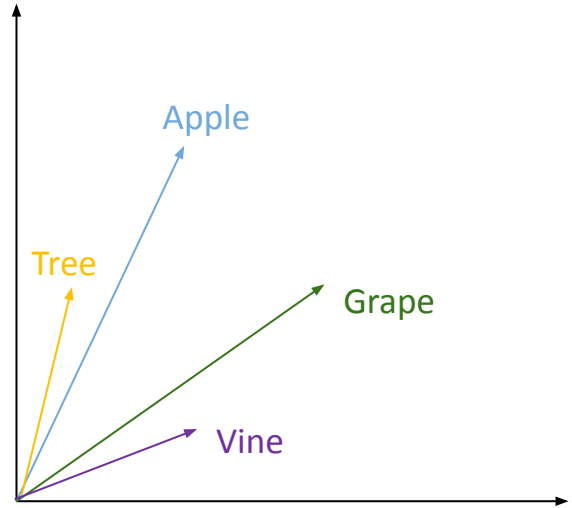
Word Representations

Jurafsky & Martin chapter 6

Word Vector Intuition

	Apple	Tree	Vine	Grape
Edible	0.95	0.07	0.06	0.94
Washingtonian	0.85	0.82	0.08	0.11
Californian	0.09	0.09	0.85	0.85
Natural	0.7	0.7	0.7	0.7

.
.
.



Apple – Tree + Vine = Grape

We will calculate these vectors in a self-supervised way.

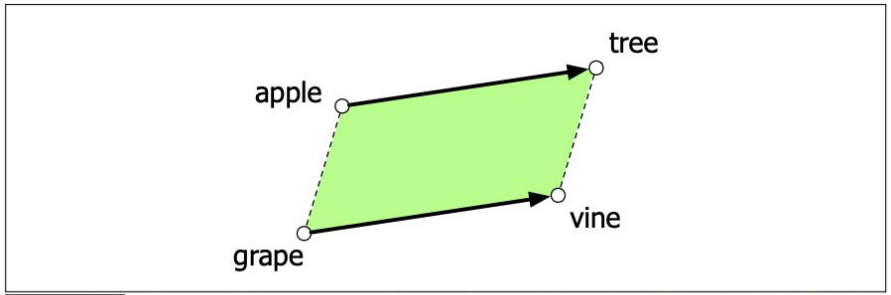


Figure 6.15 The parallelogram model for analogy problems (Rumelhart and Abrahamson, 1973): the location of $\overrightarrow{\text{vine}}$ can be found by subtracting $\overrightarrow{\text{apple}}$ from $\overrightarrow{\text{tree}}$ and adding $\overrightarrow{\text{grape}}$.

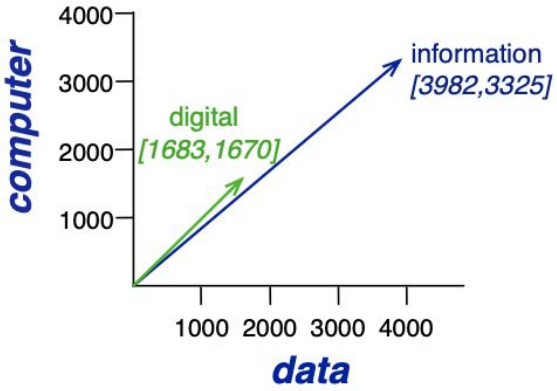
Word Vector Intuition

- Distributional Hypothesis: words with similar meanings appear in similar contexts in the text
- First attempt at word vectors: co-occurrence matrix (aka term-context matrix)

is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

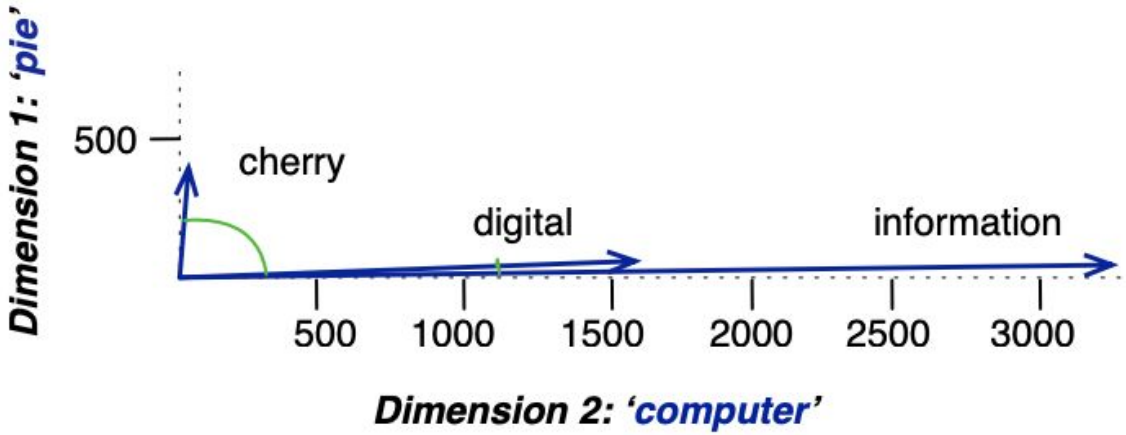
Figure 6.5 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.



Word Vector Evaluation

- SimLex-999 is one of several datasets of human judgements of word similarity (averaged over multiple annotators)
- Evaluation: correlation between cosine similarity of vectors and human similarity judgements

vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3



Drawbacks of that “first attempt”

- Huge sparse vectors
- Some words show up overall less often, but they should still matter more because they are more important
    - For example, “extraordinary” shows up less often than “the” but is probably more important in determining word meanings
    - Possible solution: take into account how often you would expect two words to co-occur given their proportion in the text (PMI)
- Better solution: Word2Vec (and GloVe and FastText)

Using context

Assumption: words that show up in similar contexts have similar meaning

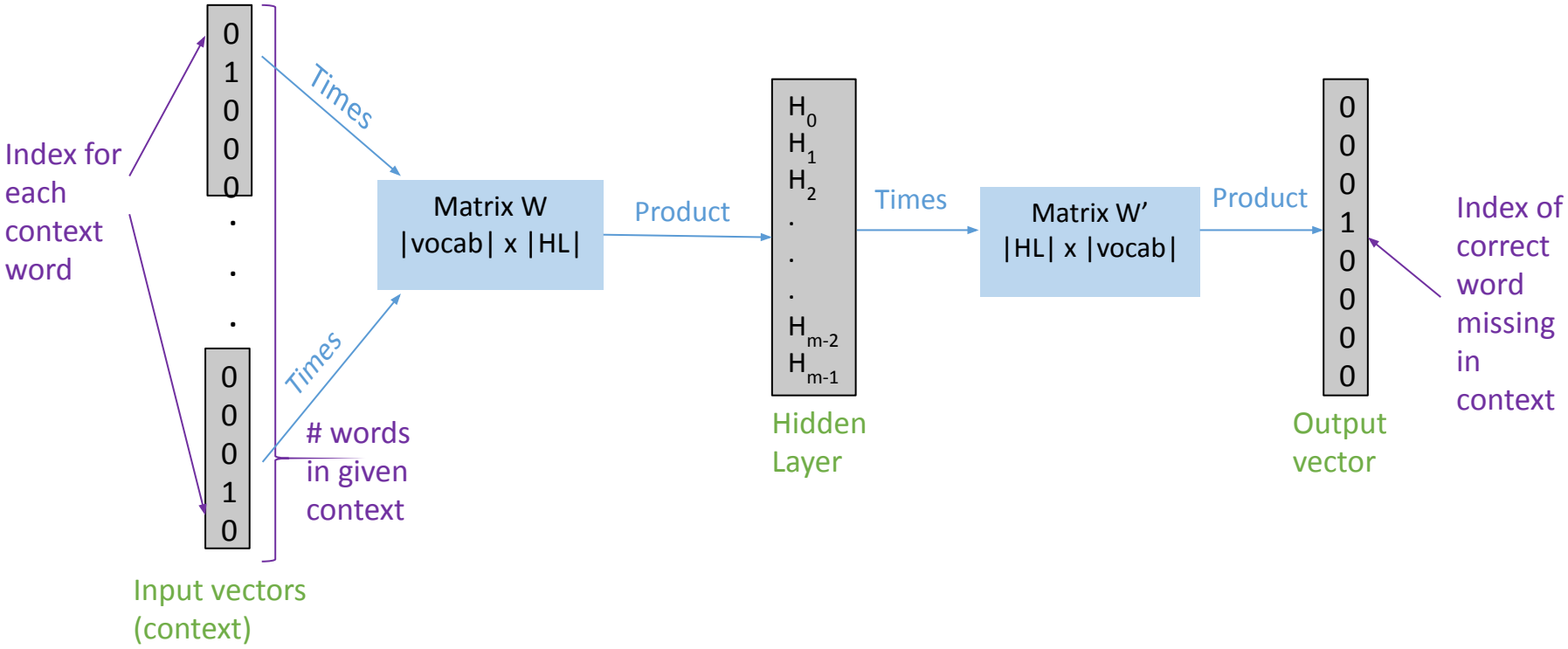
Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

Word2Vec: A Fake Task

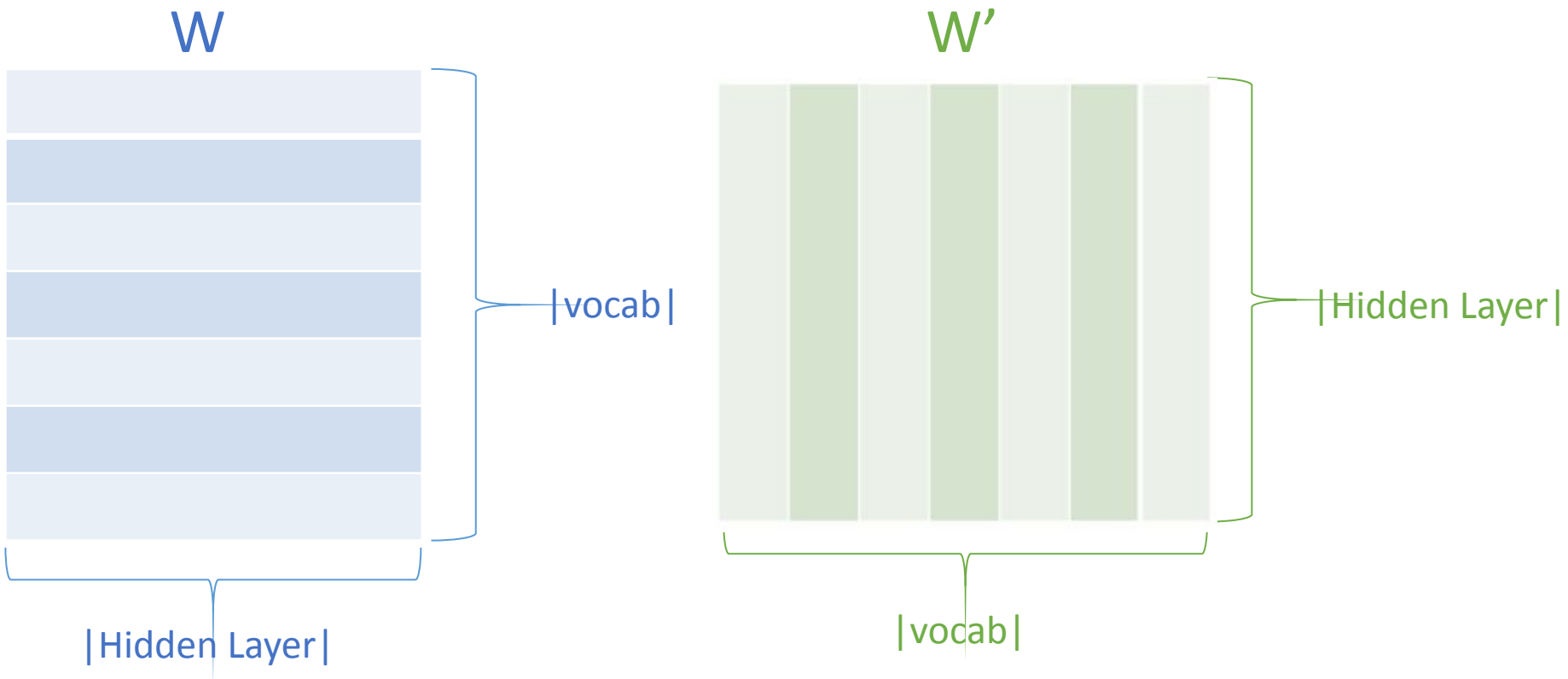
荃者所以在鱼，得鱼而忘荃 Nets are for fish;
Once you get the fish, you can forget the net.
言者所以在意，得意而忘言 Words are for meaning;
Once you get the meaning, you can forget the words
庄子(Zhuangzi), Chapter 26

- Make vectors where each word has an index
  - (like Bag of Words/Tfidf vectors)
  - Represent each word as a “one-hot encoded” vector (1 for that word, 0 everywhere else)
  - These vectors are the input and output for the fake task
- There are two options for the fake task:
  - Continuous Bag of Words (CBOW)
    - Input: context words that surround *v* in the text, knowing word *v* is missing
    - Output: word *v*
  - Skip-Gram
    - Input: word *v*
    - Output: words that surround *v* in the text

Training CBOW



W and W' Matrices



How to train the network

- Data
  - Positive examples: pairs of words that are in context of each other
  - Negative examples: pairs of words that are not in context of each other
    - Negative sampling: sample these, because if you use all such pairs, that’s way too many
- Training
  - Goal: Find values in matrices W and W’
  - How: stochastic gradient descent

Stochastic Gradient Descent

1. Make random guesses for all the values in W and W’
2. Repeat until loss stops decreasing:
    a. For each example in the training data:
        i. Forward pass:
            1. Do the multiplications in the network with the current values in W and W’, get an output vector VEC of length |V|
            2. Do **softmax** on VEC to get the actual output vector VEC
        ii. Backward pass:
            1. Calculate the **gradient of the loss function** at this point (derivative w.r.t. each element of W and W’)
            2. Update values in W and W’ by taking a “step” in the direction of the gradient
                a. The “stochastic” part means that sometimes we sometimes randomly take a step in the opposite direction to avoid local minima

Softmax

- Turns a vector of all sorts of numbers (positive, negative, extremely large) into a vector where all numbers are between 0 and 1, and the vector adds up to 1.
- Numbers that were bigger in the original vector are still bigger after softmax

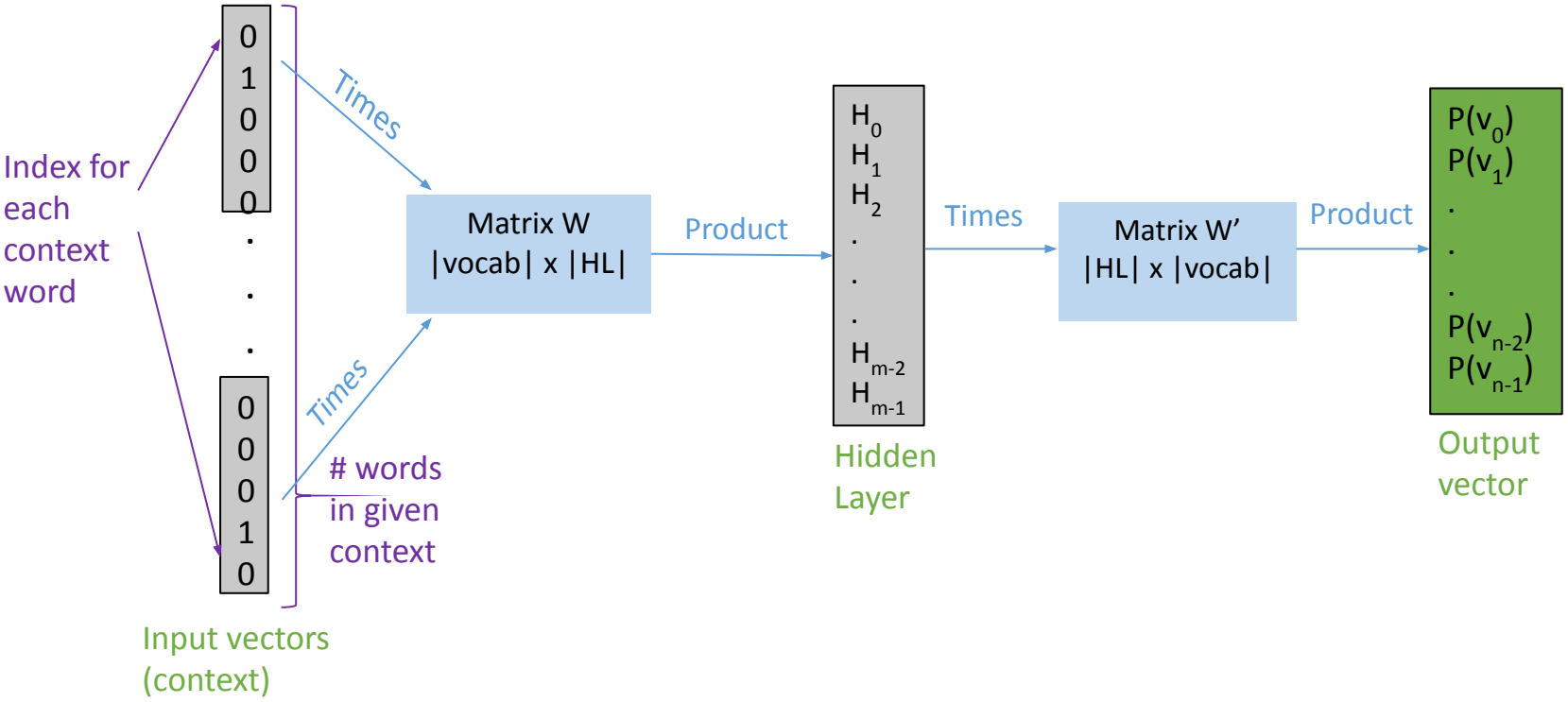
$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K$$

- This way, you can compare the one-hot vector of the expected output to this softmaxed output

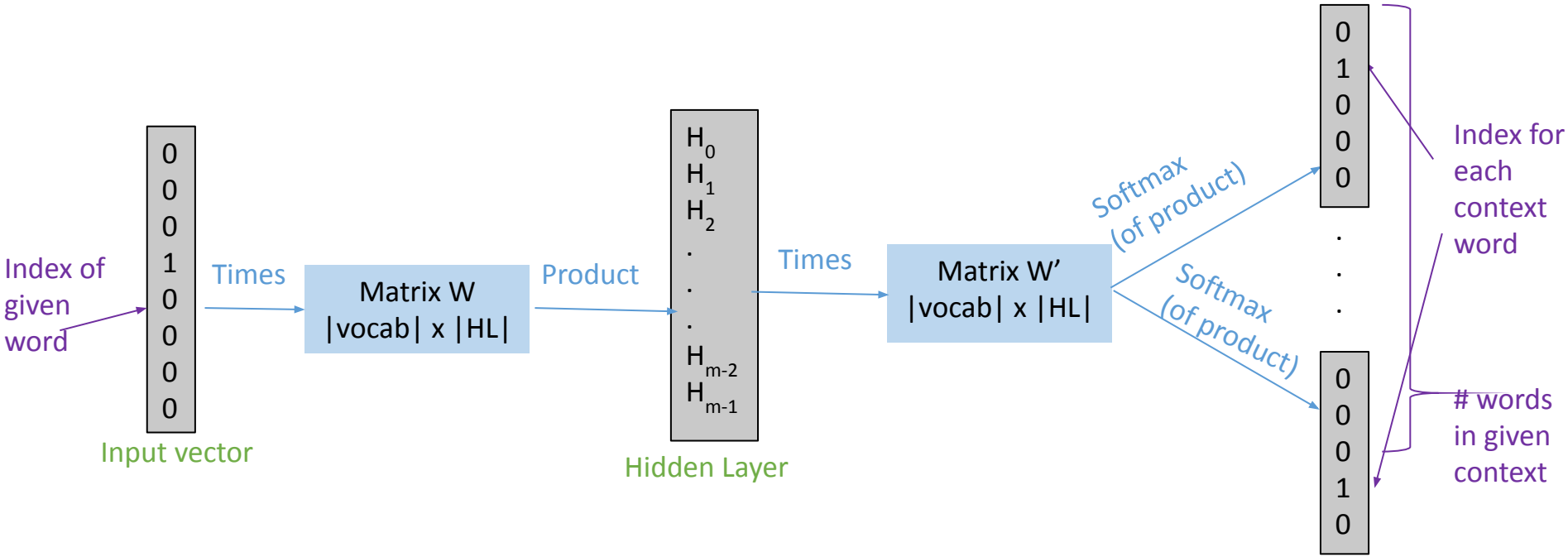
Loss function

- Loss = difference between the softmaxed output and the expected output (one-hot encoded target word)
- To minimize loss, we take its derivative and take a “step” in the direction of the slope
- We keep taking “steps” until we hit a local minimum (the loss stops changing)
- Taking a “step” = adjusting the weights in W and W’

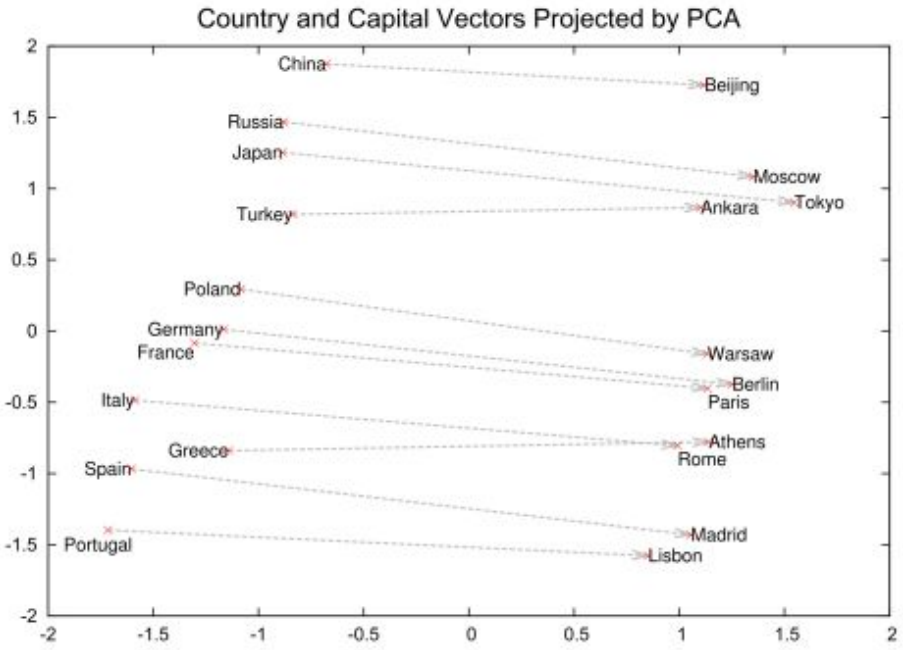
Revisit CBOW



Skip-Gram


Index of given word
0 0 0 1 0 0 0 0
Input vector
Times
Matrix W
|vocab| x |HL|
Product
H0
H1
H2
Hm-2
Hm-1
Hidden Layer
Times
Matrix W'
|vocab| x |HL|
Softmax (of product)
Softmax (of product)
0 1 0 0 0
0 0 0 1 0
Index for each context word
words in given context

Some results



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

These vectors from Distributed Representations of Words and Phrases and their Compositionality (Mikilov et al.) were trained on a day of Google News.

Other results from J&M

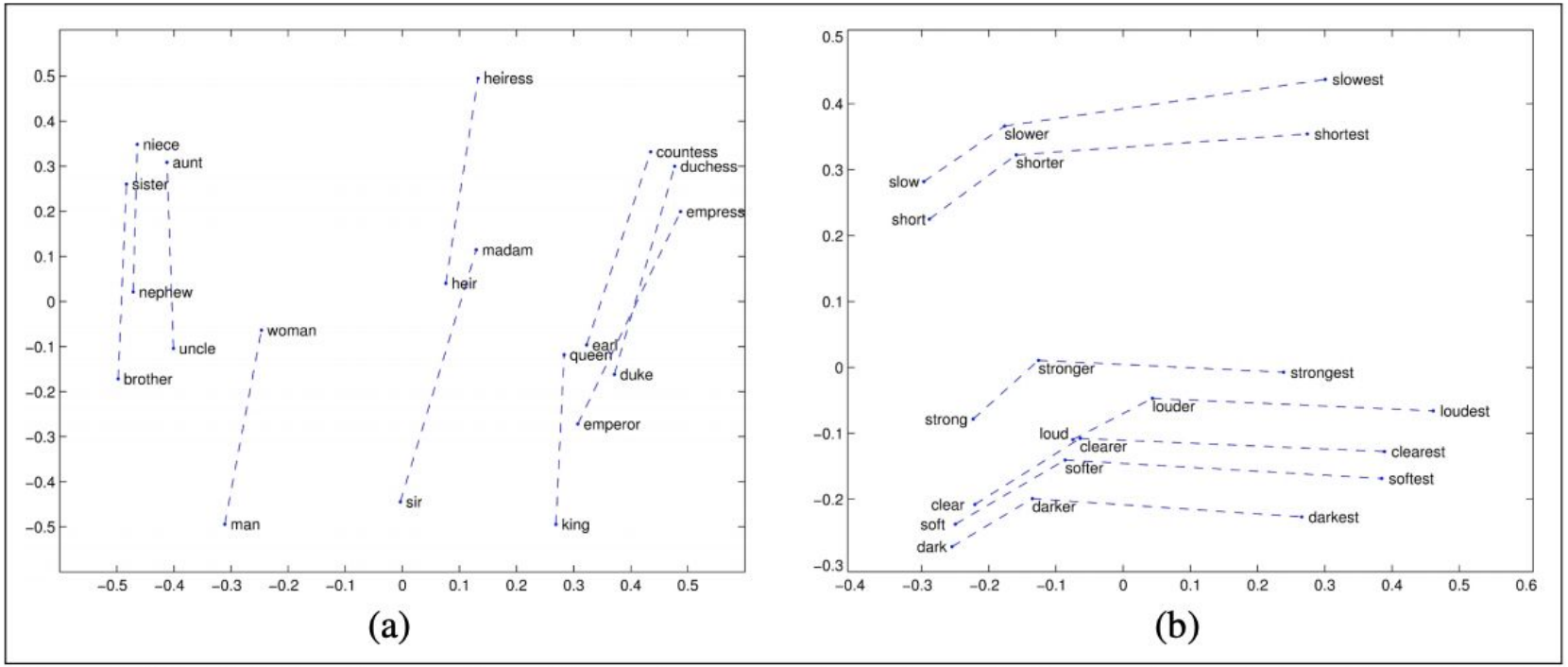
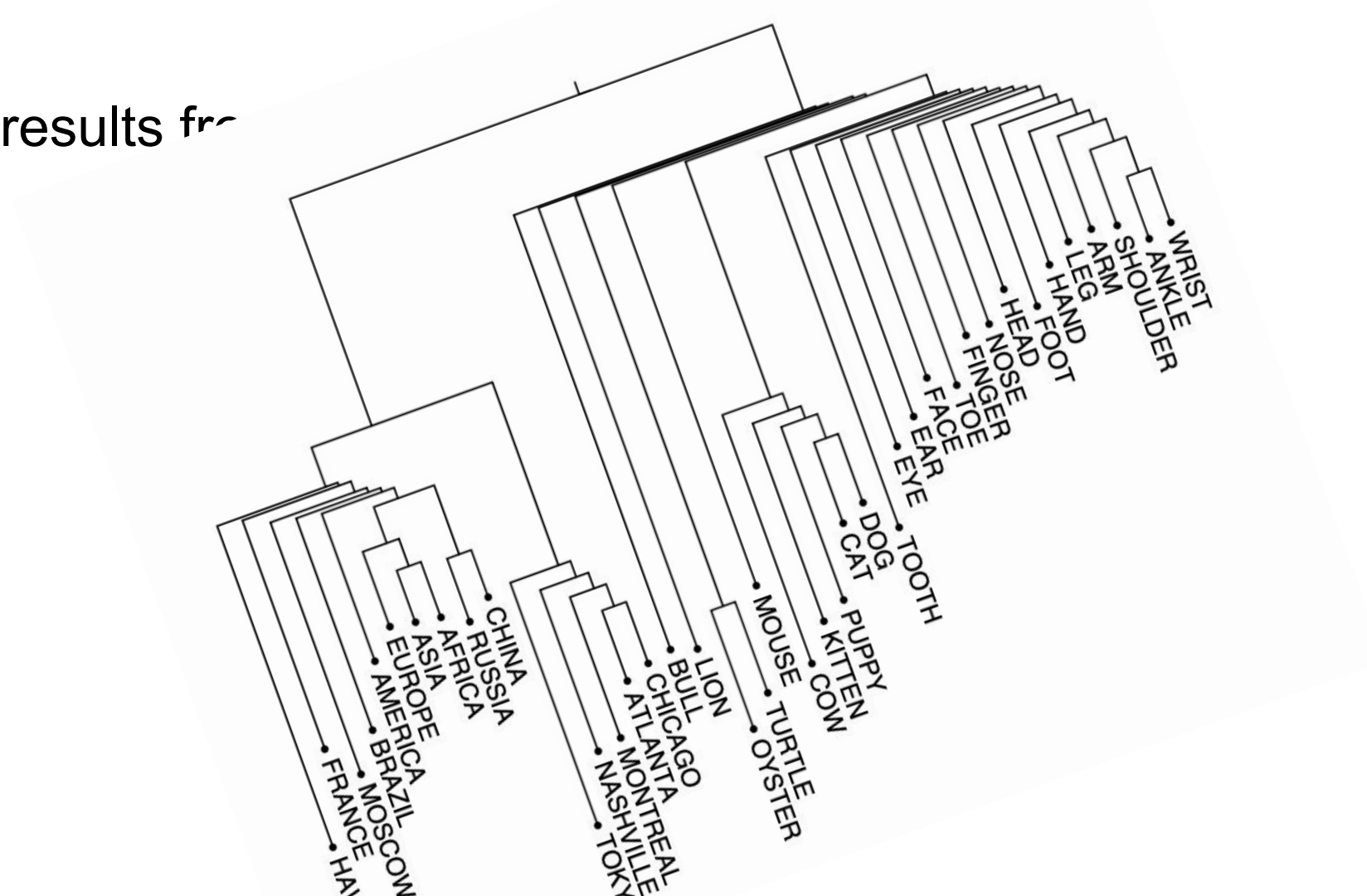


Figure 6.13 Relational properties of the vector space, shown by projecting vectors onto two dimensions. (a) 'king' - 'man' + 'woman' is close to 'queen' (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

Other results fr



GloVe and FastText

- So far, we covered Word2Vec with CBOW and Skip Gram
- GloVe
  - Like Word2Vec, but it is trained by doing matrix factorization on the co-occurrence matrix (so it reduces the dimension of the co-occurrence matrix)
- FastText
  - Made by Facebook AI
  - Like Word2Vec, but using character ngrams instead of words
    - Each word's vector is the average of the vectors of its character ngrams
  - This has the advantage that if there is a word that wasn't in your training set, you can still assign it a vector from its ngrams

Historical shift in word meanings

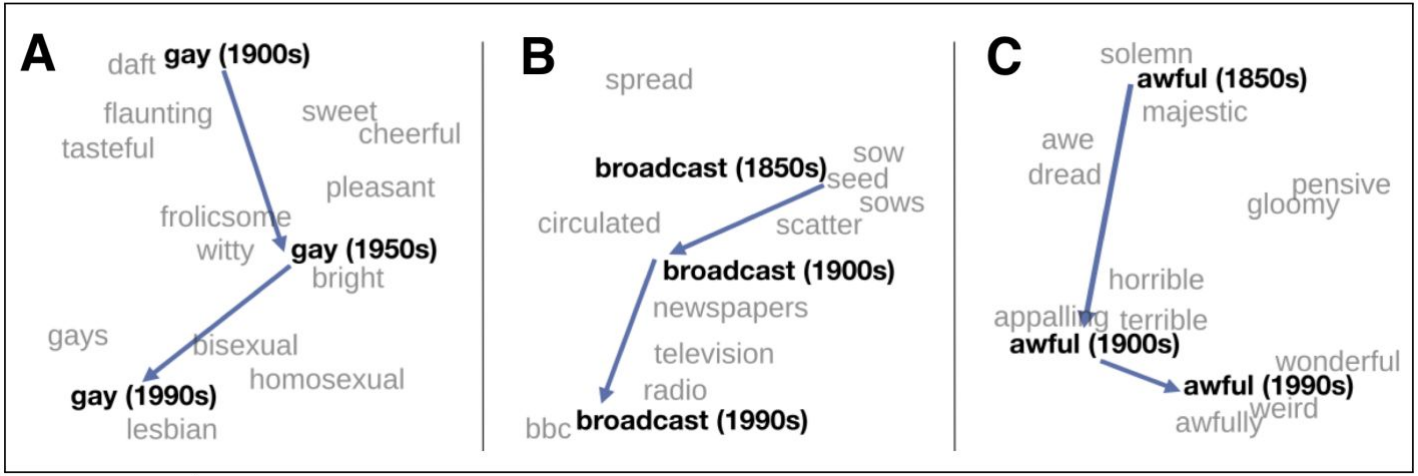


Figure 6.17 A t-SNE visualization of the semantic change of 3 words in English using word2vec vectors. The modern sense of each word, and the grey context words, are computed from the most recent (modern) time-point embedding space. Earlier points are computed from earlier historical embedding spaces. The visualizations show the changes in the word *gay* from meanings related to "cheerful" or "frolicsome" to referring to homosexuality, the development of the modern "transmission" sense of *broadcast* from its original sense of sowing seeds, and the pejoration of the word *awful* as it shifted from meaning "full of awe" to meaning "terrible or appalling" (Hamilton et al., 2016).

Bias

- Some datasets yield computer programmer:man::homemaker:woman (Bolukbasi 2016)
    - Allocation bias: a computer may downrank job applications based on gender
- Vectors for African-American names are closer to unpleasant words while vectors for European-American names are closer to pleasant words (Caliskan 2017)
    - Representational harm: a computer may represent some groups as worse (or ignore them)
- Bias amplification: sometimes the social bias encoded in word embeddings is more exaggerated than actual employment statistics (Garg 2018)

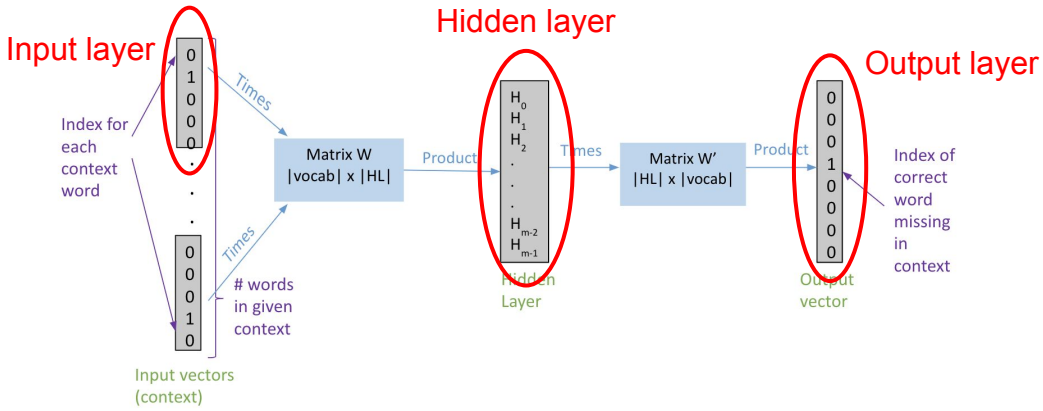
Neural Networks

Jurafsky & Martin chapter 7

Neural Networks

The network that we saw for Word2Vec is an example of a fully connected neural network with 1 hidden layer

- The elements of the vectors in each layer are the neurons
- “Fully connected” means that each neuron in each layer is used to calculate each neuron in the next layer
- It was also “feedforward” because the computation goes from each layer to the next (no looping back to previous layers)



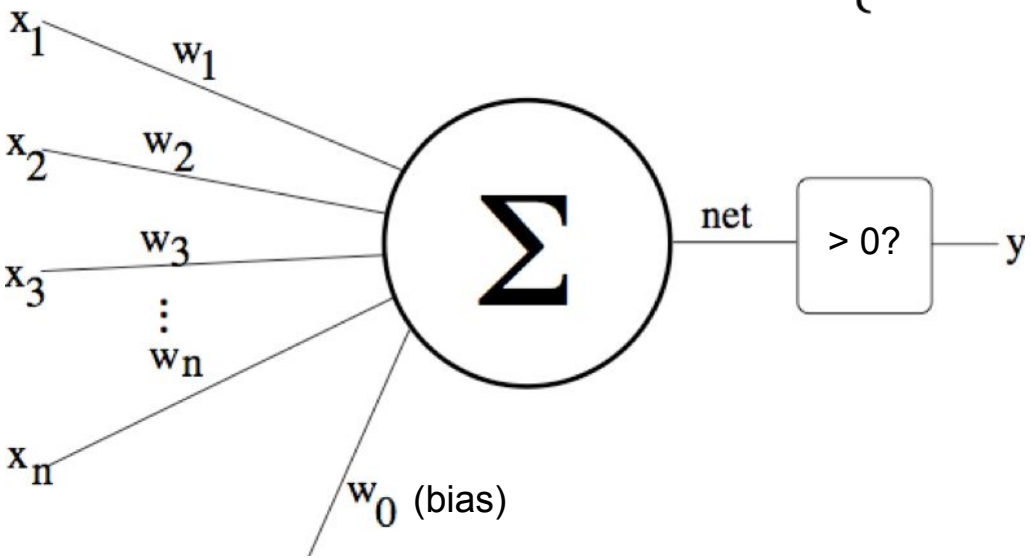
Neural Networks

- “Neural” because it is inspired by biological neurons
- Universal Approximation Theorem for Neural Networks: you can approximate any function using a neural network
- You don’t need fancy feature templates based on domain knowledge - you can just throw the words as features, and the network learns what is important
- Deep learning = neural networks with many layers

Building Block: Perceptron

- A perceptron is a linear classifier
- A weight for each feature, plus a bias

$$f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{else} \end{cases}$$



Neural unit: Perceptron + activation function

- Why? differentiable, has numbers between 0 and 1
- As before, x_1, x_2, … are input and y is output
- w_1, w_2, … and b are the weights we want to learn
- Sigmoid is a popular activation function

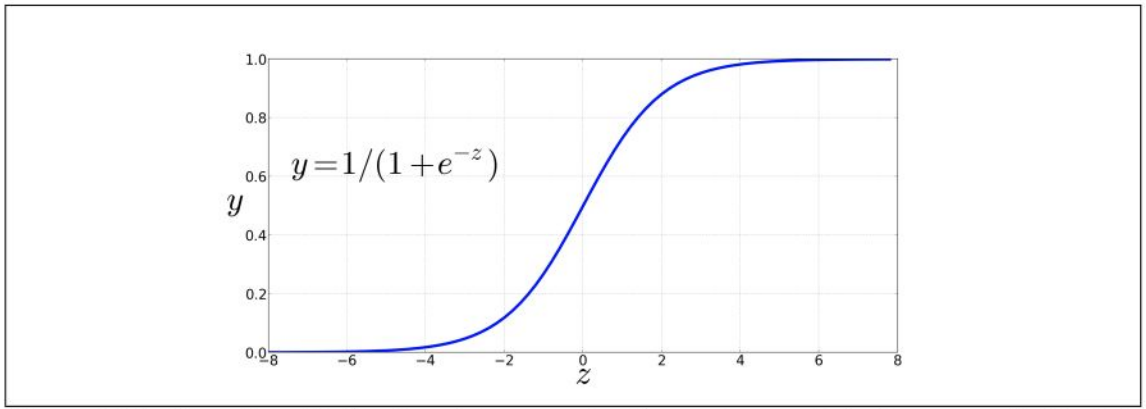
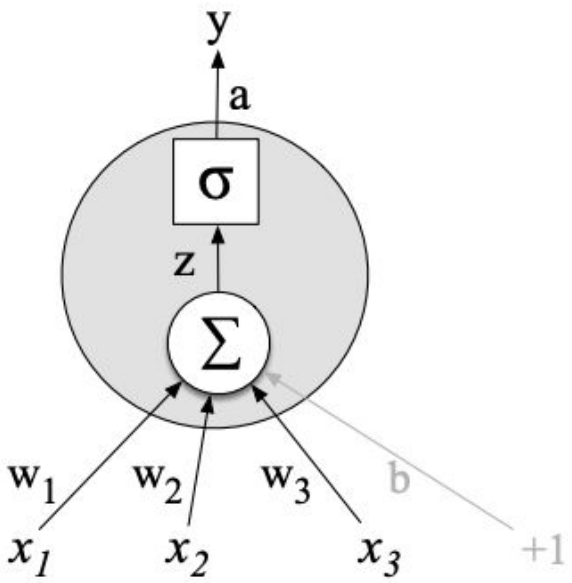
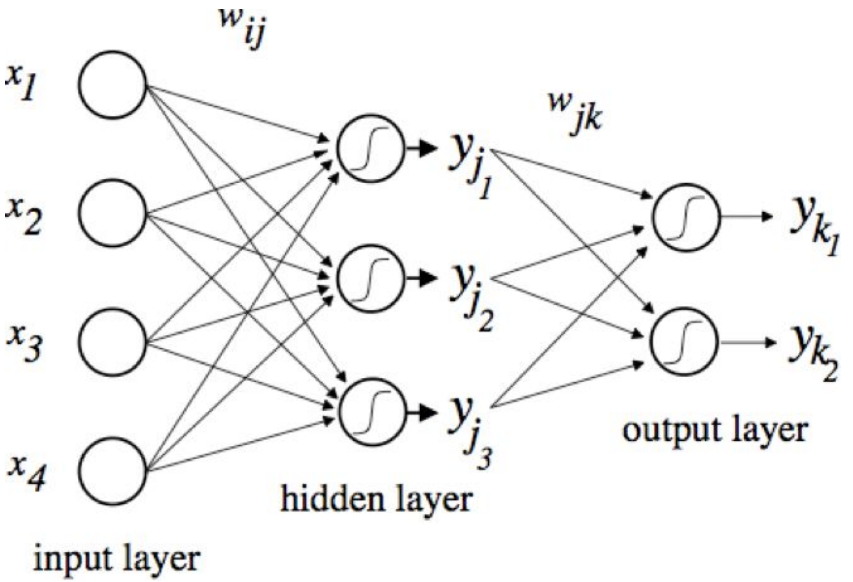


Figure 7.1 The sigmoid function takes a real value and maps it to the range $[0,1]$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

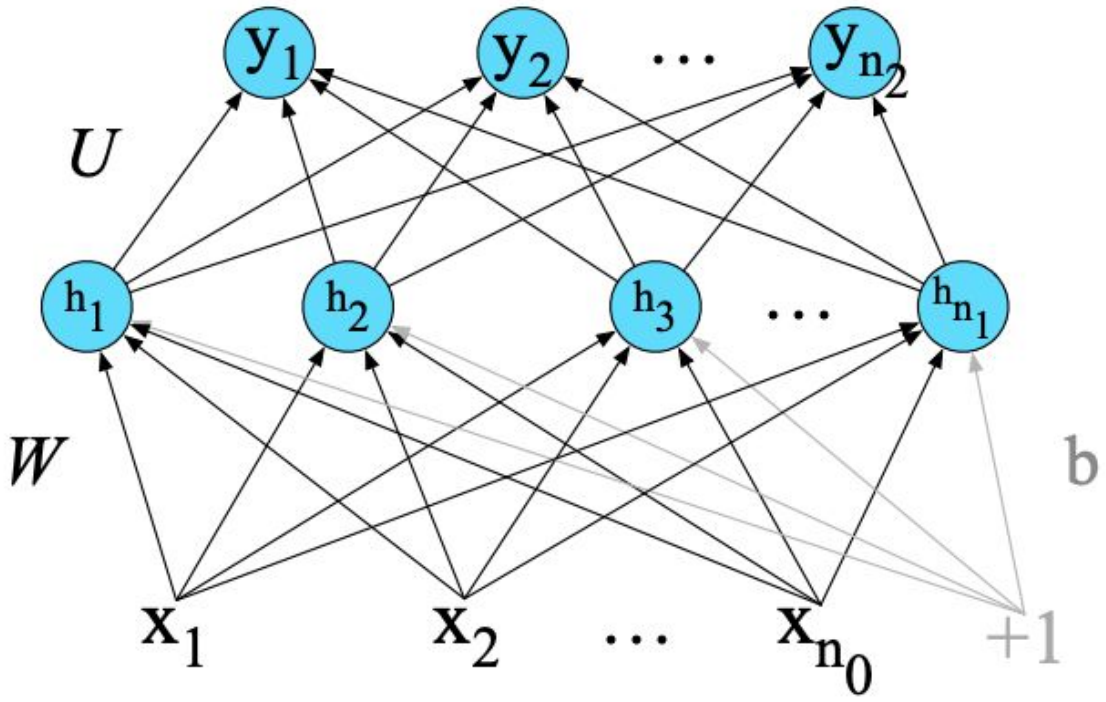


Multi-layer Perceptrons (MLP)

- Is actually multiple layers of neural units (including the activation function), not just perceptrons
- Word2Vec is a MLP with a linear activation function (y = x) for the hidden layer
- MLPs often have the softmax on the output layer to turn it into a probability distribution, like in Word2Vec
- Why? Because MLPs can model functions that are not linearly separable



Multi-layer Perceptrons (MLP)



$$h = \sigma(Wx + b)$$
$$z = Uh$$
$$y = \text{softmax}(z)$$

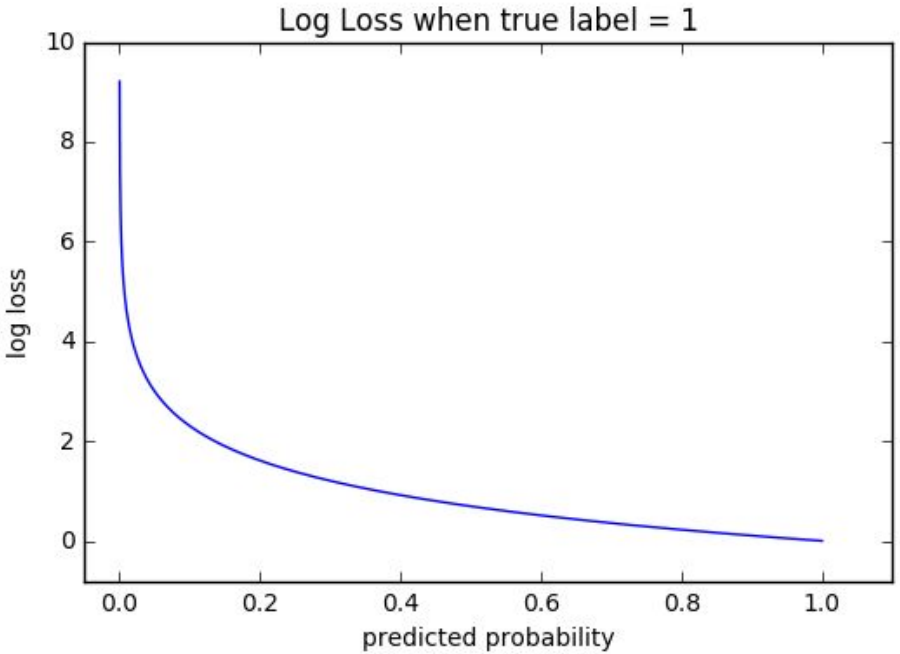
$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$
$$\hat{y} = a^{[2]}$$

U = W[2]
Each g is an activation function including g[2], which is softmax

Training a Neural Network

- Minimize a loss function
- Common loss function for classification: **cross-entropy loss** (aka log-loss or negative log-likelihood)
- y_i is the true labels and $\hat{y}_i$ is the output
- So, y_i is 0 for all “negative” labels and 1 for positive
- Ps. Cross-entropy between distributions p and q = entropy of p + KL-divergence from p of q

$$L_{CE}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$



https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Training a Neural Network: Computation Graph

- We use forward pass and backward pass as described for Word2Vec
- How do we update weights when the outputs depend on each other?
- Example computation graph:

$$d = 2 * b$$
$$e = a + d$$
$$L = c * e$$

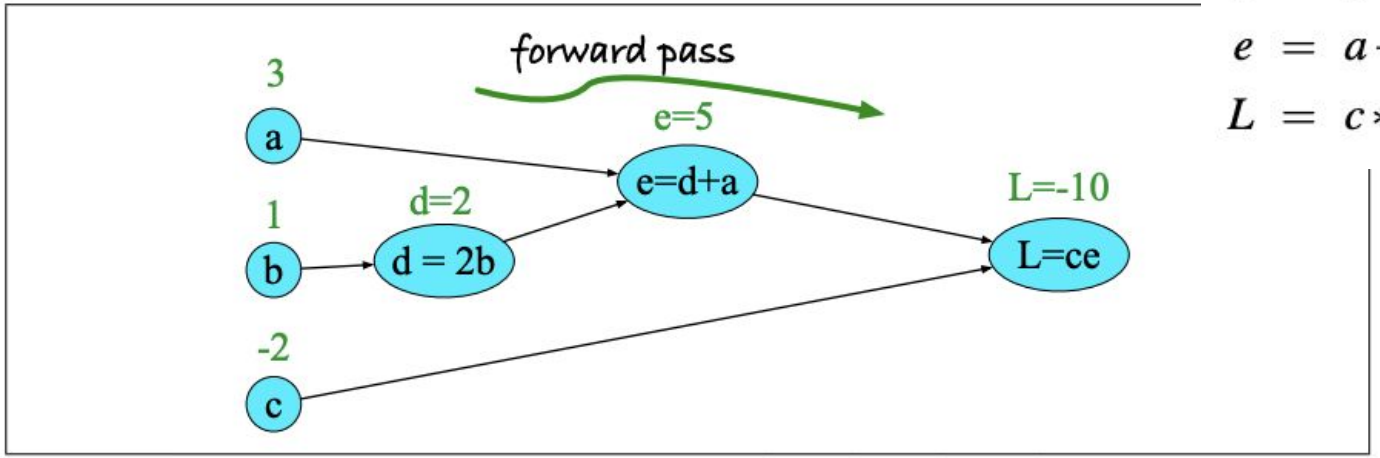


Figure 7.9 Computation graph for the function $L(a,b,c) = c(a+2b)$, with values for input nodes $a = 3$, $b = 1$, $c = -2$, showing the forward pass computation of L.

Computation Graph + Chain Rule

$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

$$L = ce : \quad \frac{\partial L}{\partial e} = c, \frac{\partial L}{\partial c} = e$$

$$e = a + d : \quad \frac{\partial e}{\partial a} = 1, \frac{\partial e}{\partial d} = 1$$

$$d = 2b : \quad \frac{\partial d}{\partial b} = 2$$

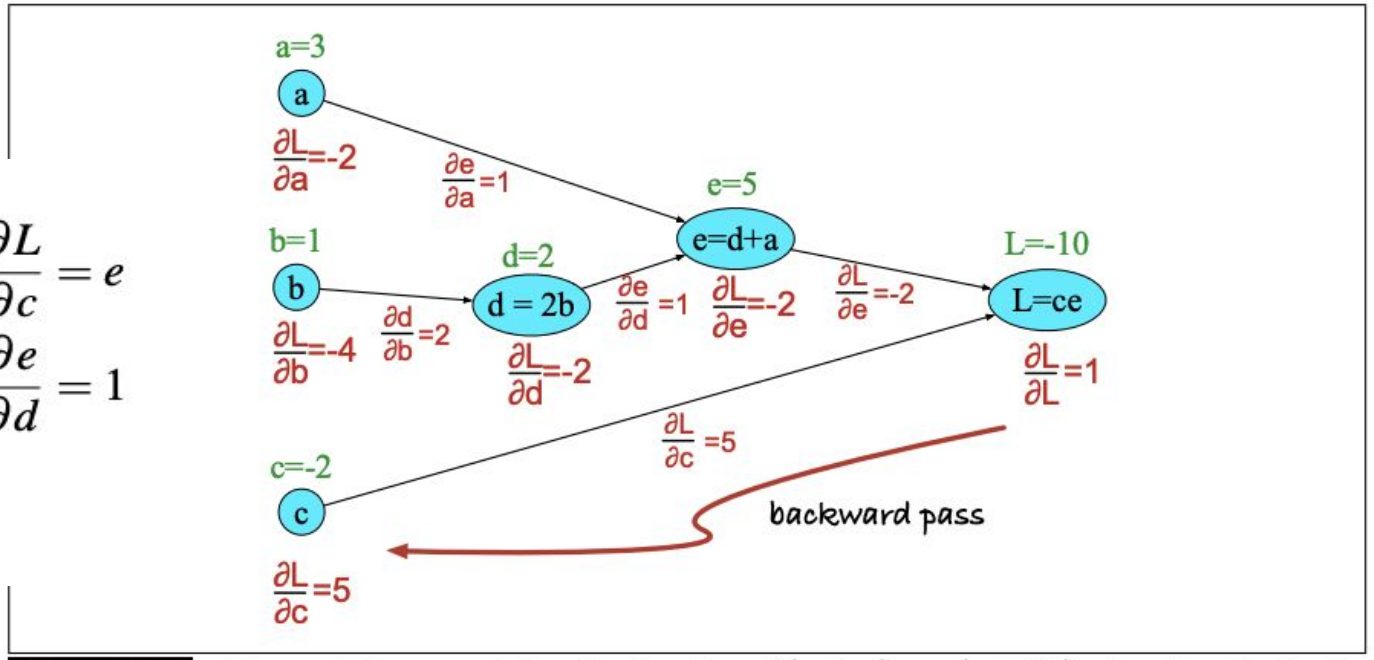


Figure 7.10 Computation graph for the function $L(a,b,c) = c(a+2b)$, showing the backward pass computation of $\frac{\partial L}{\partial a}$, $\frac{\partial L}{\partial b}$, and $\frac{\partial L}{\partial c}$.

Training a Neural Network

$$
\begin{aligned}
z^{[1]} &= W^{[1]}\mathbf{x} + b^{[1]} \\
a^{[1]} &= \text{ReLU}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]}) \\
\hat{y} &= a^{[2]}
\end{aligned}
$$

$$
\frac{d\,\text{ReLU}(z)}{dz} = \begin{cases} 0 & for \quad x < 0 \\ 1 & for \quad x \geq 0 \end{cases}
$$

$$
\frac{d\tanh(z)}{dz} = 1 - \tanh^2(z)
$$

$$
\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))
$$

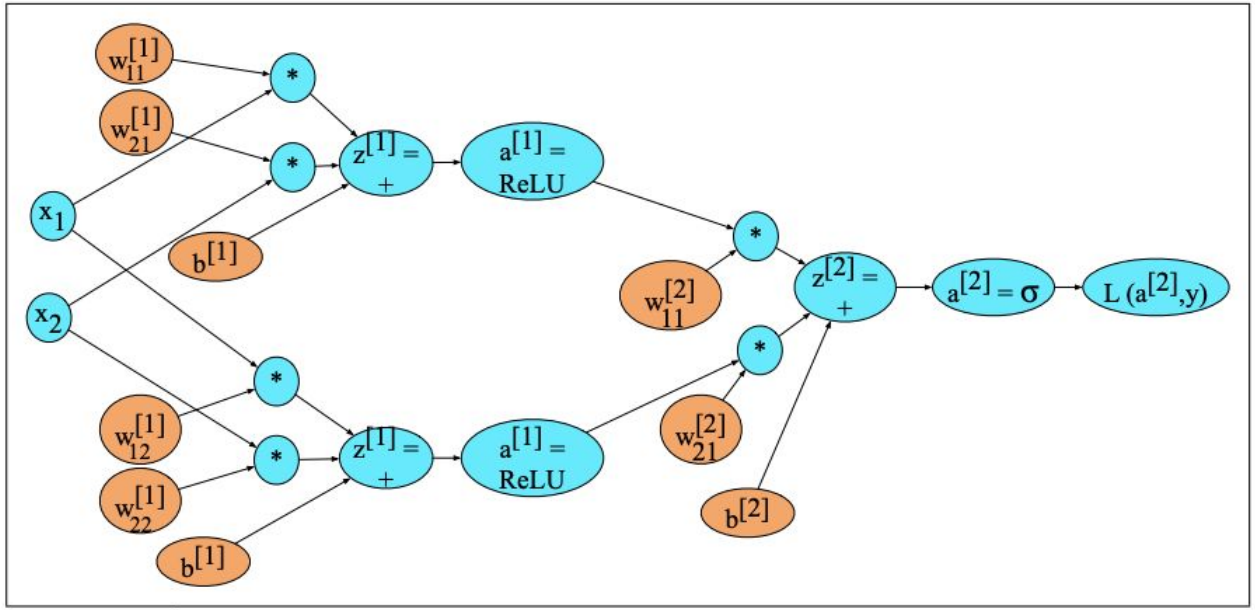
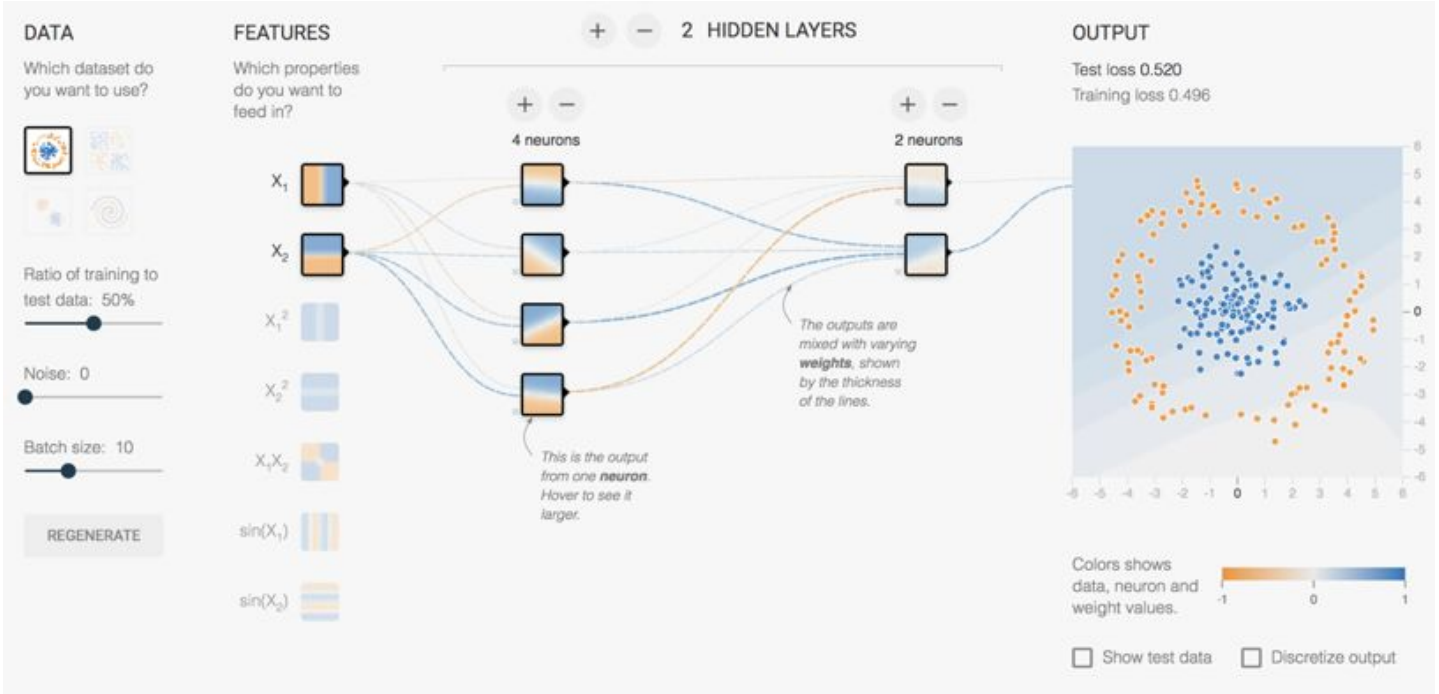


Figure 7.11 Sample computation graph for a simple 2-layer neural net (= 1 hidden layer) with two input dimensions and 2 hidden dimensions.

Elements of a Neural Network



http://playground.tensorflow.org/

Preview of next week (J&M ch. 7)

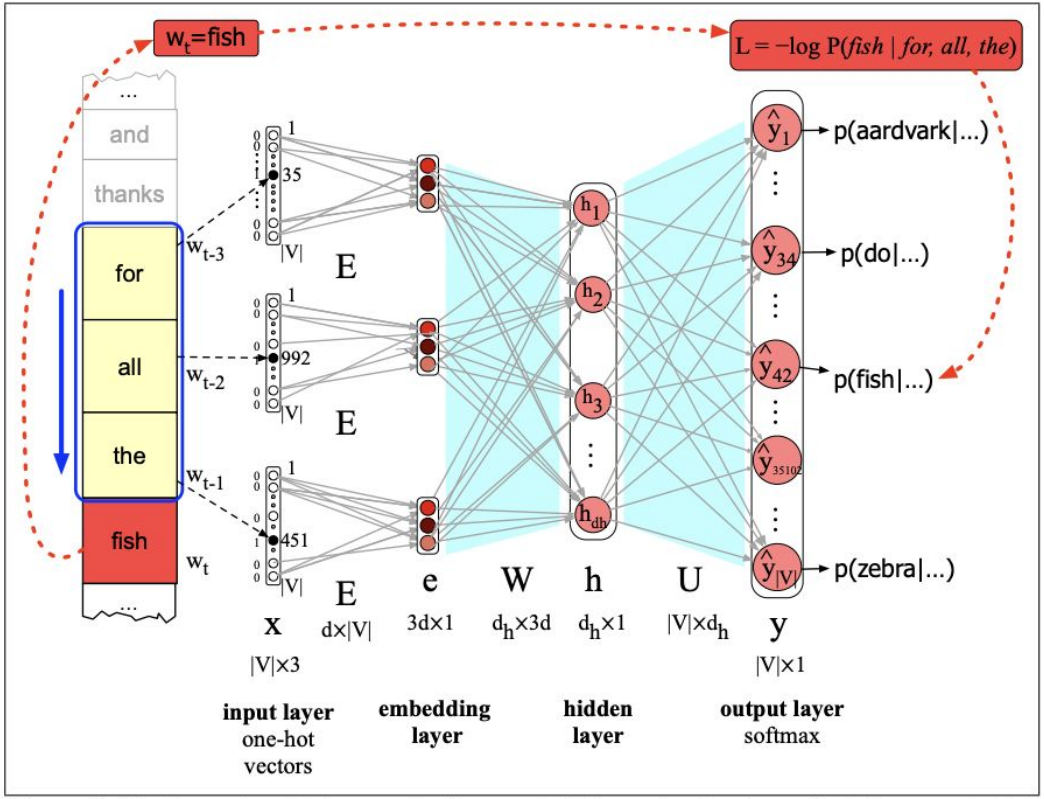


Figure 7.18 Learning all the way back to embeddings. Again, the embedding matrix **E** is shared among the 3 context words.

Word2Vec notebook

Discuss practice midterm exam