

# Lecture 3

Thursday, January 26, 2017 1:07 AM



CS224d-Lecture3

## **CS224d Deep Learning for Natural Language Processing**

### **Lecture 3: More Word Vectors**

**Richard Socher**

## Refresher: The simple word2vec model

- Main cost function  $J$ :

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

- With probabilities defined as:  $\underline{p(o|c)} = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$
- We derived the gradient for the internal vectors  $v_c$

Lecture 1, Slide 2

Richard Socher

4/5/16

## Calculating all gradients!

- We went through gradients for each center vector  $v$  in a window
- We also need gradients for outside vectors  $u$
- Derive at home!
- Generally in each window we will compute updates for all parameters that are being used in that window.
- For example window size  $c = 1$ , sentence:  
"I like learning."
- First window computes gradients for:
  - internal vector  $v_{\text{like}}$  and external vectors  $u_I$  and  $u_{\text{learning}}$
- Next window in that sentence?

Lecture 1, Slide 3

Richard Socher

4/5/16

## Compute all vector gradients!

- We often define the set of ALL parameters in a model in terms of one long vector  $\theta$
- In our case with  $d$ -dimensional vectors and  $V$  many words:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

*Concatenate*

Lecture 1, Slide 4

Richard Socher

4/5/16

## Gradient Descent

- To minimize  $J(\theta)$  over the full batch (the entire training data) would require us to compute gradients for all windows
- Updates would be for each element of  $\mu$  :

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

- With step size  $\alpha$
- In matrix notation for all parameters:

$$\theta^{new} = \theta^{old} - \alpha \frac{\partial}{\partial \theta^{old}} J(\theta)$$

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

Lecture 1, Slide 5

Richard Socher

4/5/16

## Vanilla Gradient Descent Code

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

```
while True:
    theta_grad = evaluate_gradient(J, corpus, theta)
    theta = theta - alpha * theta_grad
```

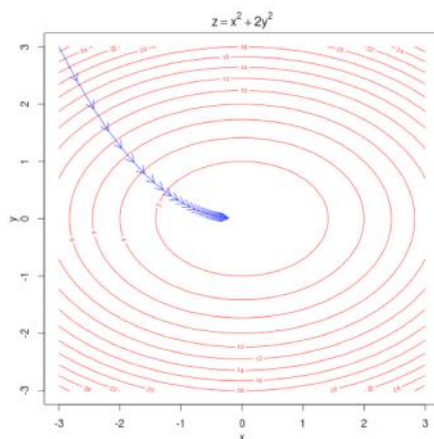
Lecture 1, Slide 6

Richard Socher

4/5/16

## Intuition

- For a simple convex function over two parameters.
- Contour lines show levels of objective function
- 



Lecture 1, Slide 7

Richard Socher

4/5/16

## Stochastic Gradient Descent

- But Corpus may have 40B tokens and windows
- You would wait a very long time before making a single update!
- Very bad idea for pretty much all neural nets!
- Instead: We will update parameters after each window  $t$   
→ Stochastic gradient descent (SGD)

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J_t(\theta)$$

*take a window from corpus.*

```
while True:
    window = sample_window(corpus)
    theta_grad = evaluate_gradient(J, window, theta)
    theta = theta - alpha * theta_grad
```

Lecture 1, Slide 8

Richard Socher

4/5/16

## Stochastic gradients with word vectors!

- But in each window, we only have at most  $2c - 1$  words, so  $\nabla_{\theta} J_t(\theta)$  is very sparse!

$$\nabla_{\theta} J_t(\theta) = \begin{bmatrix} 0 \\ \vdots \\ \nabla_{v_{like}} \\ \vdots \\ 0 \\ \nabla_{u_I} \\ \vdots \\ \nabla_{u_{learning}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2dV}$$

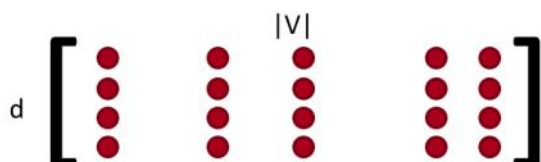
Lecture 1, Slide 9

Richard Socher

4/5/16

## Stochastic gradients with word vectors!

- We may as well only update the word vectors that actually appear!
- Solution: either keep around hash for word vectors or only update certain columns of full embedding matrix U and V



- Important if you have millions of word vectors and do distributed computing to not have to send gigantic updates around.

Lecture 1, Slide 10

Richard Socher

4/5/16

## Approximations: PSet 1

- The normalization factor is too computationally expensive

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$

- Hence, in PSet1 you will implement the skip-gram model
- Main idea: train binary logistic regressions for a true pair (center word and word in its context window) and a couple of random pairs (the center word with a random word)

Lecture 1, Slide 11

Richard Socher

4/5/16



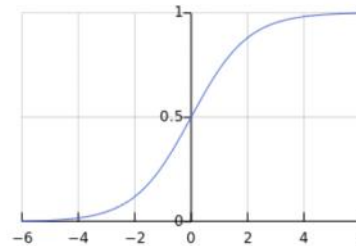
## PSet 1: The skip-gram model and negative sampling

- From paper: "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013)
- Overall objective function:  $J(\theta) = \frac{1}{T} \sum_{t=1}^T J_t(\theta)$

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^k \mathbb{E}_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

- Where k is the number of negative samples and we use,

- The sigmoid function!  $\sigma(x) = \frac{1}{1+e^{-x}}$  (we'll become good friends soon)
- So we maximize the probability of two words co-occurring in first log  
→



Lecture 1, Slide 12

Richard Socher

4/5/16

negative sampling  
true pair  
+ random  
pairs.

## PSet 1: The skip-gram model and negative sampling

- Slightly clearer notation:

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

- Max. probability that real outside word appears,  
minimize prob. that random words appear around center word
- $P(w) = U(w)^{3/4} / Z$ ,  
the unigram distribution  $U(w)$  raised to the 3/4th power  
(We provide this function in the starter code).
- The power makes less frequent words be sampled more often

so that  
more likely to sample  
less likely words

Lecture 1, Slide 13

Richard Socher

4/5/16

## PSet 1: The continuous bag of words model

CBOW.

- Main idea for continuous bag of words (CBOW): Predict center word from sum of surrounding word vectors instead of predicting surrounding single words from center word as in skip-gram model

- To make PSet slightly easier:

The implementation for the CBOW model is not required and for bonus points!

Lecture 1, Slide 14

Richard Socher

4/5/16

## Count based vs direct prediction

LSA, HAL (Lund & Burgess),  
COALS (Rohde et al),  
Hellinger-PCA (Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

• NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

15

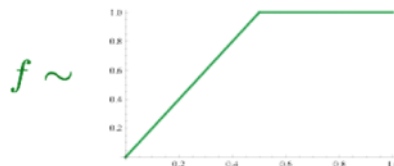
Richard Socher

4/5/16



## Combining the best of both worlds: GloVe

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$



- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

16

Richard Socher

4/5/16

## Glove results

Nearest words to  
frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus



litoria



leptodactylidae



rana



eleutherodactylus

17

Richard Socher

4/5/16

## What to do with the two sets of vectors?

- We end up with U and V from all the vectors u and v (in columns)  $= =$
- Both capture similar co-occurrence information. It turns out, the best solution is to simply sum them up:

$$X_{\text{final}} = U + V$$

- One of many hyperparameters explored in *GloVe: Global Vectors for Word Representation* (Pennington et al. (2014))

Lecture 1, Slide 18

Richard Socher

4/5/16

## How to evaluate word vectors?

Evaluation

- Related to general evaluation in NLP: Intrinsic vs extrinsic
- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing one subsystem with another improves accuracy → Winning!

e.g. analogy

→ evaluate in terms of performance.

Lecture 1, Slide 19

Richard Socher

4/5/16

## Intrinsic word vector evaluation

- Word Vector Analogies

a:b :: c:?

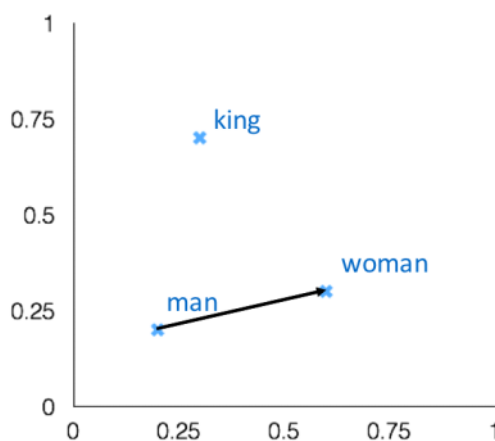
man:woman :: king:?

$$d = \arg \max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!

- Problem: What if the information is there but not linear?

*not all models do it well.*

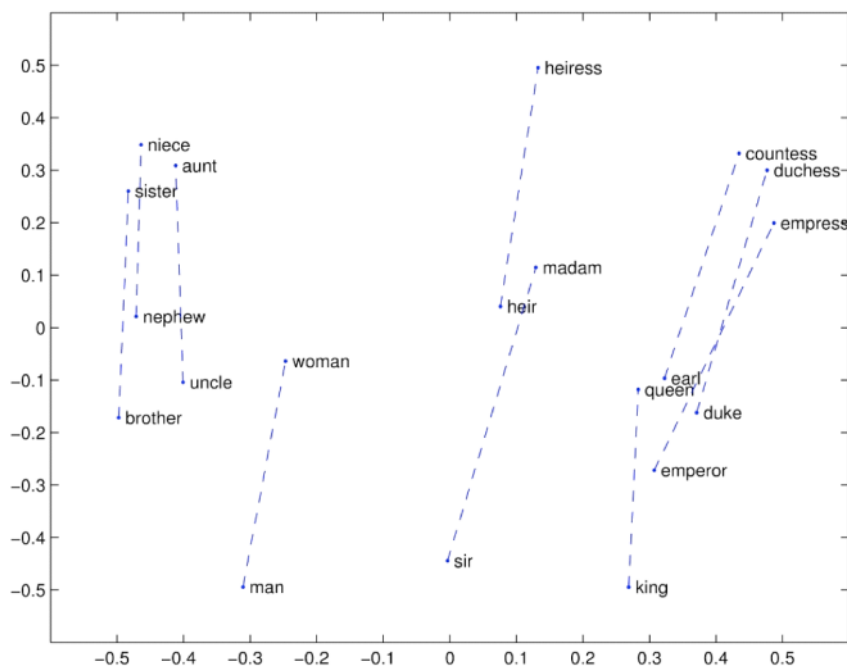


Lecture 1, Slide 20

Richard Socher

4/5/16

## Glove Visualizations

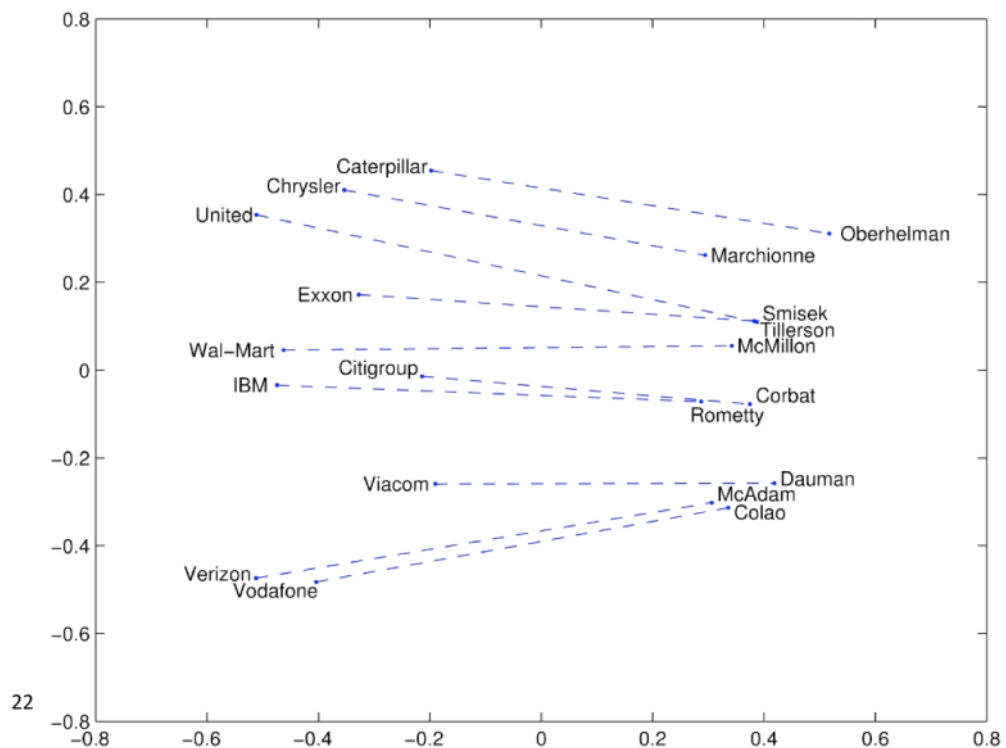


21

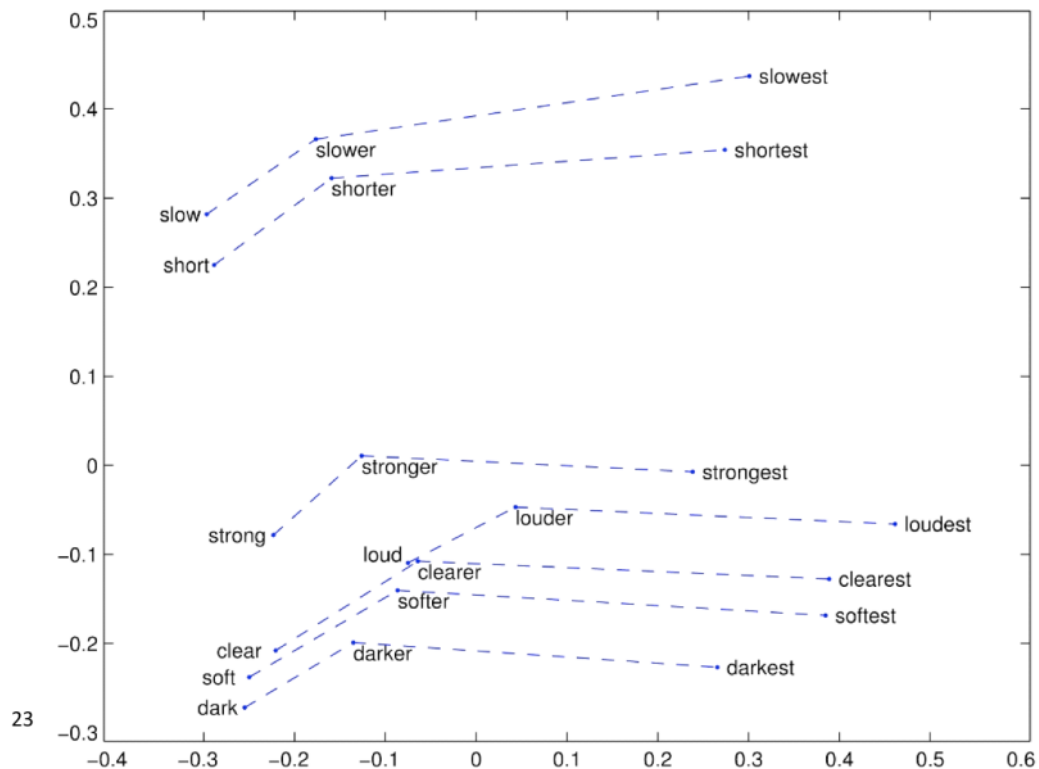
Richard Socher

4/5/16

## Glove Visualizations: Company - CEO



## Glove Visualizations: Superlatives



## Details of intrinsic word vector evaluation

- Word Vector Analogies: Syntactic and **Semantic** examples from <http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt>

: city-in-state

Chicago Illinois Houston Texas  
Chicago Illinois Philadelphia Pennsylvania  
Chicago Illinois Phoenix Arizona  
Chicago Illinois Dallas Texas  
Chicago Illinois Jacksonville Florida  
Chicago Illinois Indianapolis Indiana  
Chicago Illinois Austin Texas  
Chicago Illinois Detroit Michigan  
Chicago Illinois Memphis Tennessee  
Chicago Illinois Boston Massachusetts

problem: different cities  
may have same name

Lecture 1, Slide 24

Richard Socher

4/5/16

## Details of intrinsic word vector evaluation

- Word Vector Analogies: Syntactic and **Semantic** examples from

: capital-world

Abuja Nigeria Accra Ghana  
Abuja Nigeria Algiers Algeria  
Abuja Nigeria Amman Jordan  
Abuja Nigeria Ankara Turkey  
Abuja Nigeria Antananarivo Madagascar  
Abuja Nigeria Apia Samoa  
Abuja Nigeria Ashgabat Turkmenistan  
Abuja Nigeria Asmara Eritrea  
Abuja Nigeria Astana Kazakhstan

problem: can change

Lecture 1, Slide 25

Richard Socher

4/5/16

## Details of intrinsic word vector evaluation

- Word Vector Analogies: **Syntactic** and Semantic examples from

: gram4-superlative  
bad worst big biggest  
bad worst bright brightest  
bad worst cold coldest  
bad worst cool coolest  
bad worst dark darkest  
bad worst easy easiest  
bad worst fast fastest  
bad worst good best  
bad worst great greatest

Lecture 1, Slide 26

Richard Socher

4/5/16

## Details of intrinsic word vector evaluation

- Word Vector Analogies: **Syntactic** and Semantic examples from

: gram7-past-tense  
dancing danced decreasing decreased  
dancing danced describing described  
dancing danced enhancing enhanced  
dancing danced falling fell  
dancing danced feeding fed  
dancing danced flying flew  
dancing danced generating generated  
dancing danced going went  
dancing danced hiding hid  
dancing danced hitting hit

Lecture 1, Slide 27

Richard Socher

4/5/16



## Analogy evaluation and hyperparameters

- Very careful analysis: GloVe word vectors

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	<u>67.5</u>	<u>54.3</u>	<u>60.3</u>
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	<u>64.8</u>	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	<u>80.8</u>	61.5	<u>70.3</u>
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW <sup>†</sup>	300	6B	63.6	<u>67.4</u>	65.7
SG <sup>†</sup>	300	6B	73.0	66.0	69.1
GloVe	300	6B	<u>77.4</u>	67.0	<u>71.7</u>
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	<u>81.9</u>	<u>69.3</u>	<u>75.0</u>

Larger corpus -- usually better  
But higher dimension vectors not always  
lead to better performance

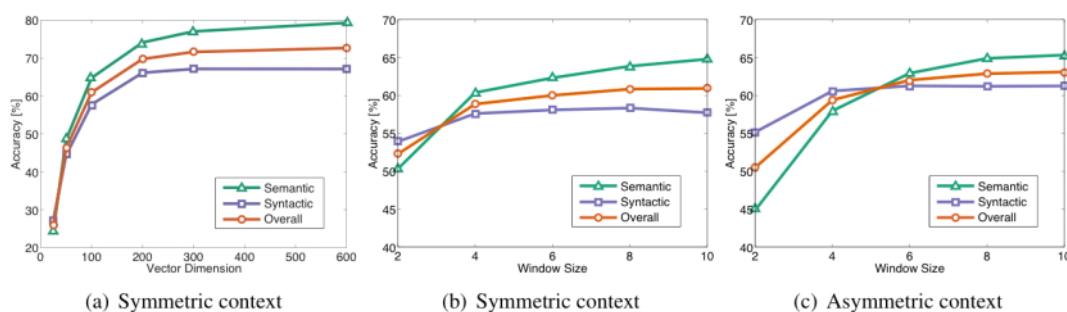
Lecture 1, Slide 28

Richard Socher

4/5/16

## Analogy evaluation and hyperparameters

- Asymmetric context (only words to the left) are not as good



- Best dimensions ~300, slight drop-off afterwards
- But this might be different for downstream tasks!

Eventually there'll be some dropoff when  
window size/ vector dimension increases

- Window size of 8 around each center word is good for GloVe vectors

Lecture 1, Slide 29

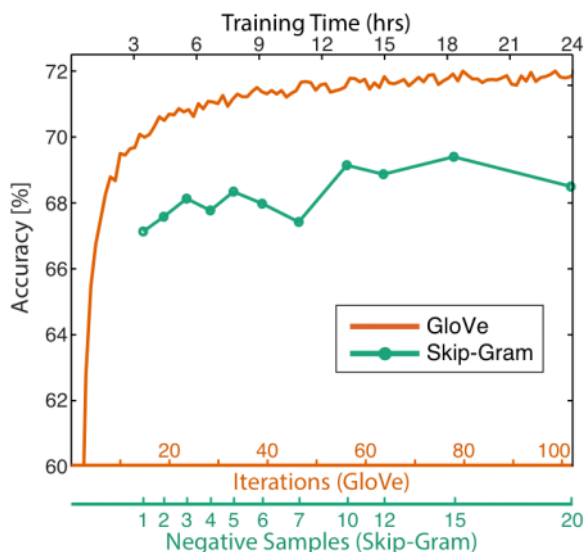
Richard Socher

4/5/16

## Analogy evaluation and hyperparameters

- More training time helps

Training longer helps!



Lecture 1, Slide 30

Richard Socher

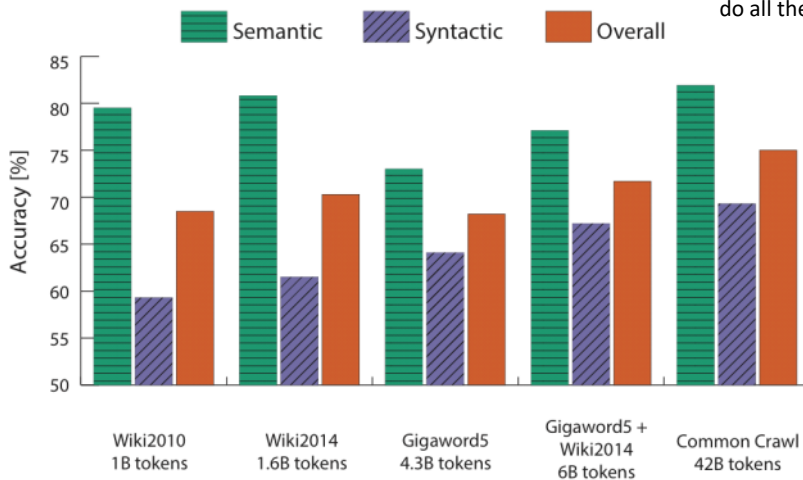
4/5/16

## Analogy evaluation and hyperparameters

- More data helps, Wikipedia is better than news text!

Wikipedia is better than news text!

Need data because asking the machine to do all these by itself



Lecture 1, Slide 31

Richard Socher

4/5/16

## Intrinsic word vector evaluation

- Word vector distances and their correlation with human judgments
- Example dataset: WordSim353  
<http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/>

Word 1   Word 2   Human (mean)

tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Lecture 1, Slide 32

Richard Socher

4/5/16

## Correlation evaluation

- Word vector distances and their correlation with human judgments

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW <sup>†</sup>	6B	57.2	65.6	68.2	57.0	32.5
SG <sup>†</sup>	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	<b>75.9</b>	<b>83.6</b>	<b>82.9</b>	<b>59.6</b>	<b>47.8</b>
CBOW <sup>*</sup>	100B	68.4	79.6	75.4	59.4	45.5

- Some ideas from Glove paper have been shown to improve skip-gram (SG) model also (e.g. sum both vectors)

Lecture 1, Slide 33

Richard Socher

4/5/16



## Extrinsic word vector evaluation

- Extrinsic evaluation of word vectors: All subsequent tasks in this class
- One example where good word vectors should help directly: named entity recognition: finding a person, organization or location

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	<b>88.7</b>	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	<b>93.2</b>	88.3	<b>82.9</b>	<b>82.2</b>

- Next: How to use word vectors in neural net models!

Lecture 1, Slide 36

Richard Socher

4/5/16

## Simple single word classification

- What is the major benefit of deep learned word vectors?
  - Ability to also classify words accurately
    - Countries cluster together → classifying location words should be possible with word vectors
  - Incorporate **any** information into them other tasks
    - Project sentiment into words to find most positive/negative words in corpus

Lecture 1, Slide 37

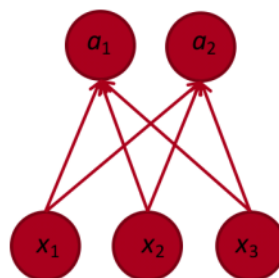
Richard Socher

4/5/16

## The softmax

Logistic regression = Softmax classification on word vector  $x$  to obtain probability for class  $y$ :

$$p(y|x) = \frac{\exp(W_y \cdot x)}{\sum_{c=1}^C \exp(W_c \cdot x)}$$



where:  $W \in \mathbb{R}^{C \times d}$

Generalizes >2 classes  
(for just binary sigmoid unit would suffice as in skip-gram)

## The softmax - details

- Terminology: Loss function = cost function = objective function
- Loss for softmax: Cross entropy
- To compute  $p(y|x)$ : first take the  $y$ 'th row of  $W$  and multiply that with row with  $x$ :

$$W_y \cdot x = \sum_{i=1}^d W_{yi} x_i = f_y$$

- Compute all  $f_c$  for  $c=1, \dots, C$
- Normalize to obtain probability with softmax function:

$$p(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^C \exp(f_c)}$$



## The softmax and cross-entropy error

- The loss wants to maximize the probability of the correct class  $y$
- Hence, we minimize the negative log probability of that class:

$$-\log p(y|x) = -\log \left( \frac{\exp(f_y)}{\sum_{c=1}^C \exp(f_c)} \right)$$

- As before: we sum up multiple cross entropy errors if we have multiple classifications in our total error function over the corpus (**more next lecture**)

Lecture 1, Slide 40

Richard Socher

4/5/16

## Background: The Cross entropy error

- Assuming a ground truth (or gold or target) probability distribution that is 1 at the right class and 0 everywhere else:  $p = [0, \dots, 0, 1, 0, \dots, 0]$  and our computed probability is  $q$ , then the cross entropy is:

$$H(p, q) = - \sum_{c=1}^C p(c) \log q(c)$$

- Because of one-hot  $p$ , the only term left is the negative probability of the true class
- Cross-entropy can be re-written in terms of the entropy and **Kullback-Leibler divergence** between the two distributions:

$$H(p, q) = H(p) + D_{KL}(p||q)$$

Lecture 1, Slide 41

Richard Socher

4/5/16

## The KL divergence

- Cross entropy:  $H(p, q) = H(p) + D_{KL}(p||q)$
- Because  $p$  is zero in our case (and even if it wasn't it would be fixed and have no contribution to gradient), to minimize this is equal to minimizing the KL divergence
- The KL divergence is **not a distance** but a non-symmetric measure of the difference between two probability distributions  $p$  and  $q$

$$D_{KL}(p||q) = \sum_{c=1}^C p(c) \log \frac{p(c)}{q(c)}$$

Lecture 1, Slide 42

Richard Socher

4/5/16

## PSet 1

- Derive the gradient of the cross entropy error with respect to the input word vector  $x$  and the matrix  $W$

Lecture 1, Slide 43

Richard Socher

4/5/16

## Simple single word classification

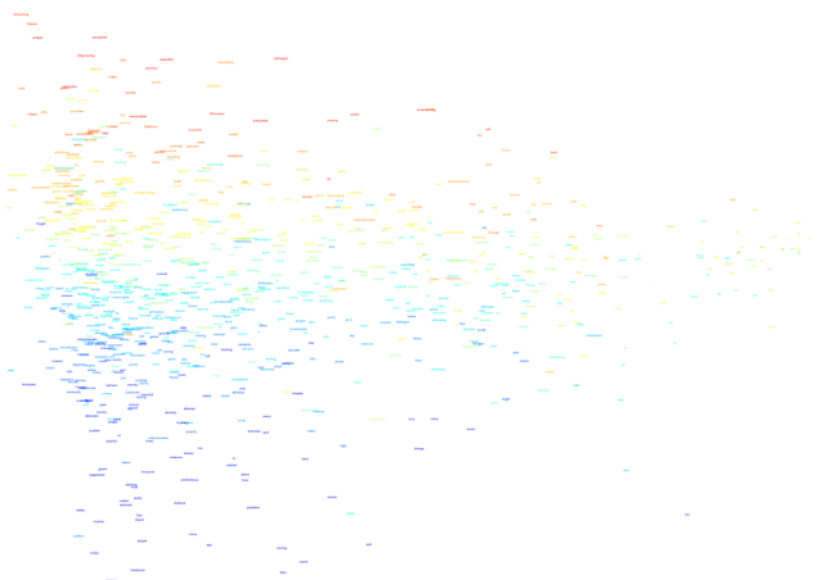
- Example: Sentiment
- Two options: train only *softmax* weights  $W$  and fix word vectors or also train word vectors
- Question: What are the advantages and disadvantages of training the word vectors?
- Pro: better fit on training data
- Con: Worse generalization because the words move in the vector space

Lecture 1, Slide 44

Richard Socher

4/5/16

## Visualization of sentiment trained word vectors



Lecture 1, Slide 45

Richard Socher

4/5/16

## Next level up: Window classification

- Single word classification has no context!
- Let's add context by taking in windows and classifying the center word of that window!
- Possible: Softmax and cross entropy error or **max-margin loss**
- Next class!

Lecture 1, Slide 46

Richard Socher

4/5/16

## References

47

Richard Socher

4/5/16