Lecture 3: Word Embeddings, FFNN

Cornell Bowers C·IS
Computer Science

Tanya Goyal

CS 4740 (and crosslists): Introduction to Natural Language Processing

Reminders

- HW1 released, due on 21 February, 11.59 p.m.
- Declare your partner (if working in a group) by 15 February, 11.59 p.m.

Today

- Recap: Logistic Regression
- Word Vectors or Word Embeddings
  - Similarity?
  - TF-IDF
  - Word2Vec
- Feed Forward Neural Networks

Recap: Binary Logistic Regression

- **Training Data**
  - input text $\mathbf{x}$
  - output label $\mathbf{y} \in \{0,1\}$

Feature Engineering

$f_0 = 1 \longrightarrow w_0$

$f_1 =$ #words $\longrightarrow w_1$

$f_2 =$ #"great" $\longrightarrow w_2$

$f_3 =$ # positive words $\longrightarrow w_3$

$f_4 =$ # negative words $\longrightarrow w_4$

$$P(\mathbf{y} = 1 \mid \mathbf{x}) = \frac{e^{\sum_i w_i f_i}}{1 + e^{\sum_i w_i f_i}}$$

$$P(\mathbf{y} = 0 \mid \mathbf{x}) = \frac{1}{1 + e^{\sum_i w_i f_i}}$$

Goal: Learn Weights $\mathbf{w} = [w_0, w_1 \ldots w_K]$

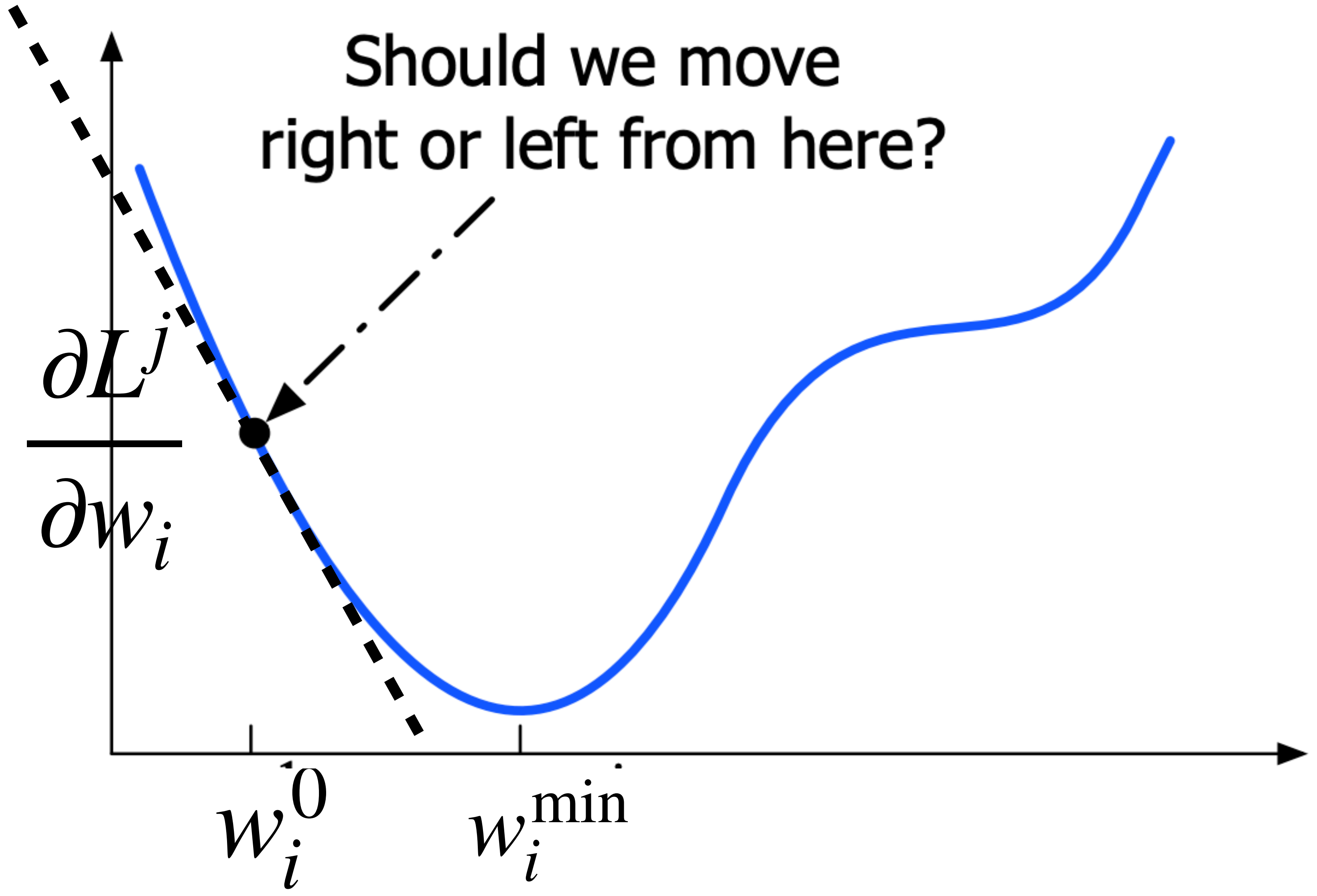
Recap: Binary Logistic Regression

j = Index of datapoint.
i = Index of feature.
t = Training time step.

Learning Weights $\mathbf{w} = [w_0, w_1 \dots w_K]$

Minimize negative log likelihood using **stochastic gradient descent.**

$$w^{\mathrm{MLE}} = \arg\min_{w} \sum_{j=1}^{N} \underbrace{\boxed{-\log P(y^j \mid x^j; w)}}_{L^j(y^j, x^j; w)}$$



$$w_i^{t+1} = w_i^t - \alpha \frac{\partial L(y^j, x^j, w^t)}{\partial w_i}$$

Recap: Binary Logistic Regression

$j =$ Index of datapoint.
$i =$ Index of feature.
$t =$ Training time step.

$t = 1$

Replace with w^1

- Initialize $w^{t=0}$
- $$\frac{\partial L^j}{\partial w_i} = \frac{\partial}{\partial w_i} - \log P(y = y^j \mid x^j; w^0)$$

$$= f_i^j \left[\underbrace{\sigma\left(\sum_i w_i f_i^j \right)}_{\text{Predicted } P(y^j = 1 \mid x^j)} - \underbrace{y^j}_{\text{True } y^j} \right]$$

- Update $w_i^{t+1} = w_i^t - \alpha \, . \, \dfrac{\partial L^j(y^j, x^j; w^0)}{\partial w_i}$

Multi-class Logistic Regression

- What if we have more than 2 classes?
- We need $P(y = y_i | x)$ for $i \in Y = \{1...L\}$
- In Binary Logistic Regression:

$$P(y = 1 | x) = \sigma(z)$$

$$P(y = 1 | x) = 1 - P(y = 0 | x)$$

- Multinomial Logistic Regression:

$$P(y = y_i) = \frac{e^{z_i}}{\sum_{j=1}^{L} e^{z_i}}, \qquad z_i = \mathbf{w}_i \,.\, \mathbf{x} + b$$

Loss?

HW1 - submission/other questions

HW1 release (+ important instructions) #17

T

Tanya Goyal STAFF
4 days ago in HWK1 – Announcements by staff

UNPIN STAR WATCHING 150 VIEWS

4

HW1 is released in this overleaf project. The homework has two sections:

- **Section A**: This is a written section and tests your conceptual knowledge of n-gram language models, text classification and word embeddings.
- **Section B:** This section outlines a programming assignment to implement a binary logistic regression classifier for the text entailment task.

Please keep in mind that this assignment has two deadlines:

- Your partner declaration is due **February 16, 11.59 p.m.** We will create a dummy assignment `hw1-partnerdeclaration` on gradescope for this purpose. Note that submission to this is ungraded but **mandatory!!**
- Your final submission, both for the written and the coding components are due **February 21, 11.59 p.m.** We will create two different assignments `hw1-written` and `hw1-programming` for you to submit these.

Please refer to the instructions at the start of the overleaf, as well as the submission instructions at the bottom of the page.

HW1-programming walkthrough

- Task: Binary classification for entailment.
- Input: [premise] [hypothesis]
- Output: Entailment / Contradiction

P: "Children are smiling and waving at camera"

H1: "The kids are frowning"

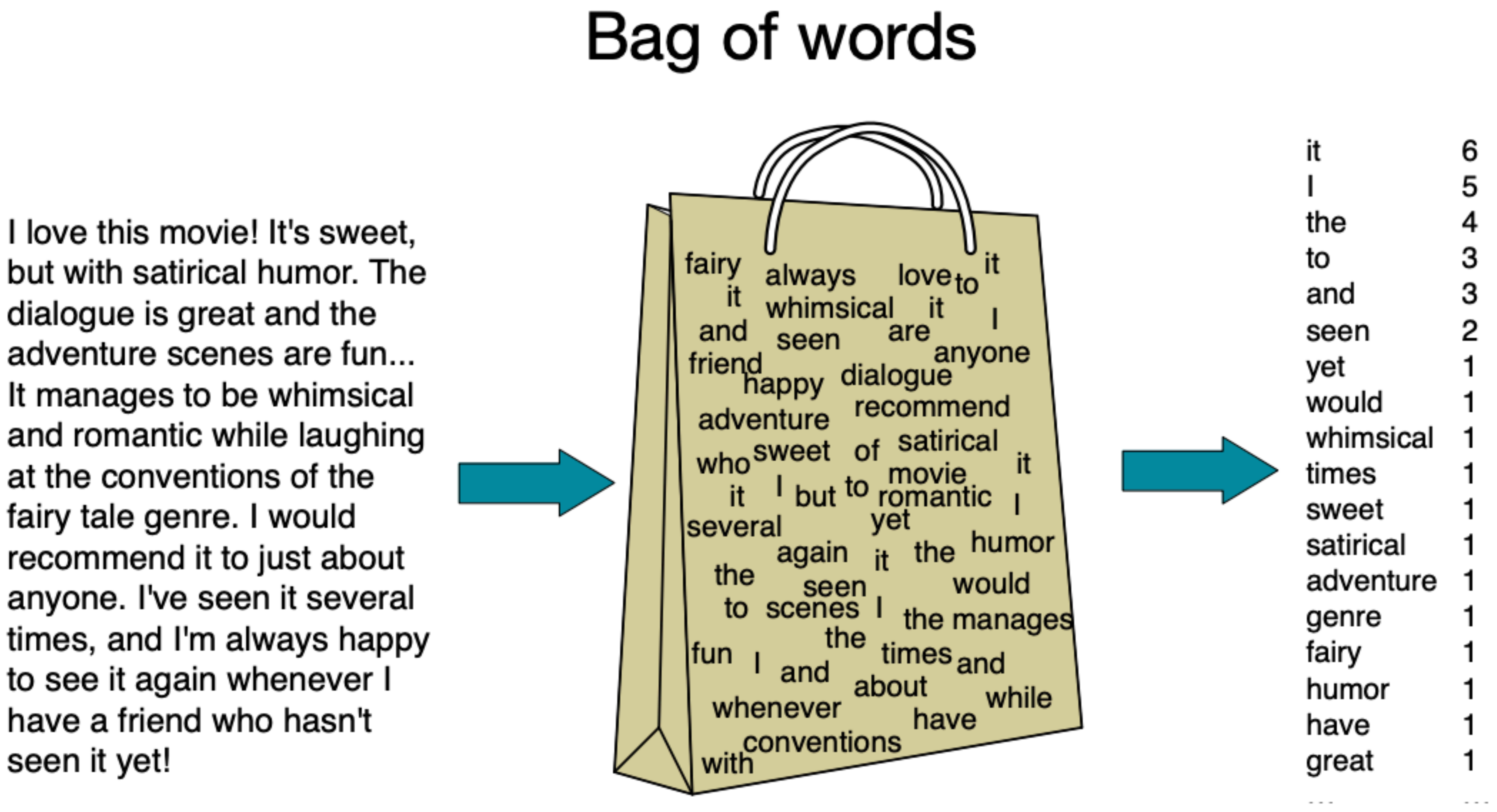
H2: "There are children present"

(P, H1) → Contradiction

(P, H2) → Entailment

HW1-programming walkthrough

- Part 2: Simple n-gram feature engineering



HW1-programming walkthrough

- Part 2: Simple n-gram feature engineering.
- Part 3: Training a Logistic Regression Model
- Part 4: Your own feature engineering!!

- **[IMPORTANT]** Do not import libraries like “scikit-learn”, “matplotlib”, etc. These packages are not downloaded on the autograder, will error out.

Today

- Recap: Logistic Regression
- Word Vectors or Word Embeddings
  - Similarity?
  - TF-IDF
  - Word2Vec
- Feed Forward Neural Networks

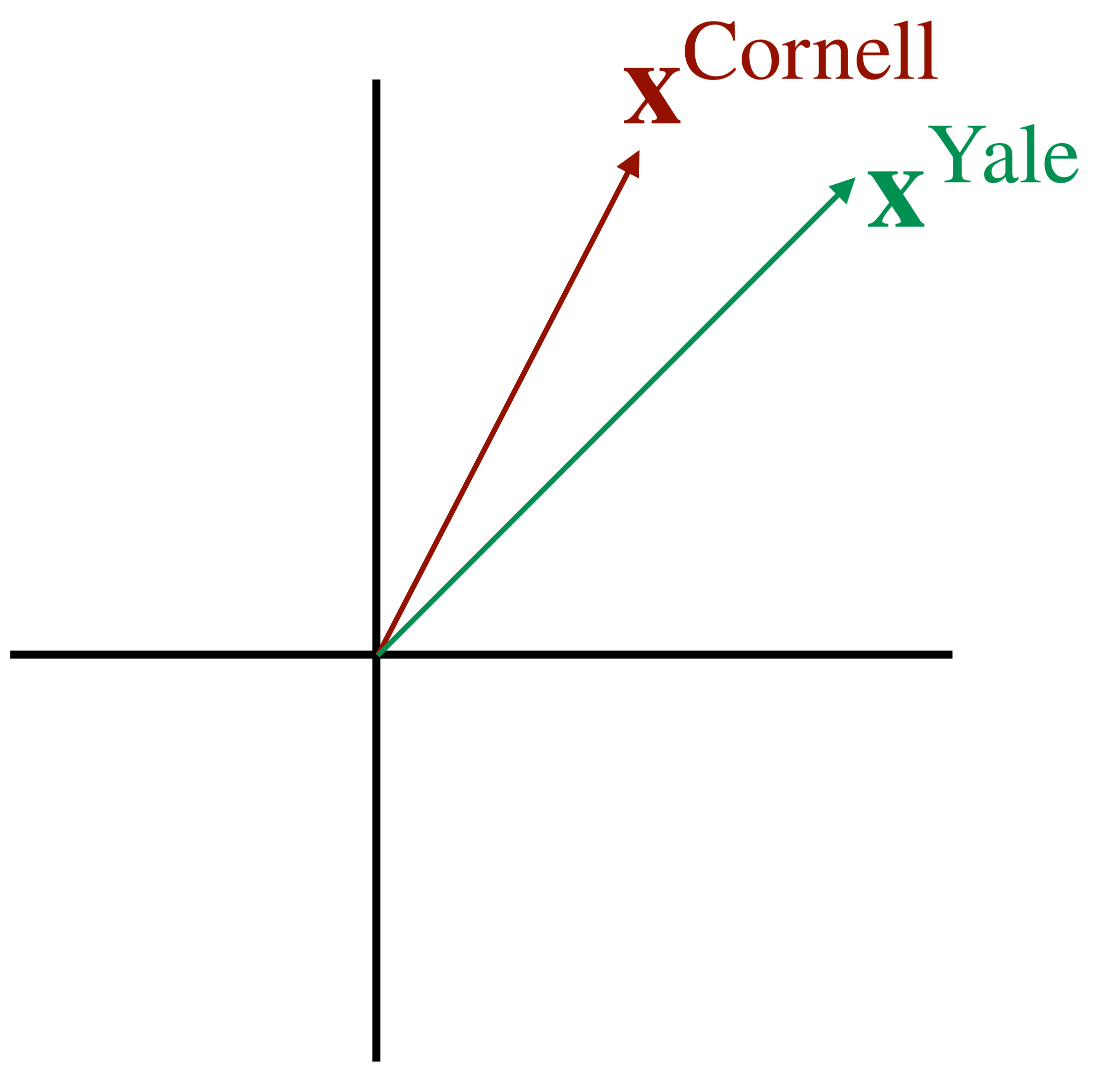
Word Vectors

- In NLP, we represent word types with **vectors**.

$$\mathbf{x}^{\text{Cornell}} = [x_1, x_2, x_3 \cdots, x_d]$$

d-dimension vector, d is fixed.

- Why vectors?



Computing similarity between two words (or sentences, or documents) is very useful in NLP!

Word Vectors: Naive Option

- Represent words as **one-hot vectors**

$$x^{\text{cat}} = [1\ 0\ 0\ 0..],\ \ x^{\text{dog}} = [0\ 1\ 0\ 0...], \ldots.$$

Lookup table:

Index	0	1	2	3	4
Word	cat	dog	the	Language	…

- Issue?

What word relations should similarity capture?

Similarity

- Less strict definition than synonyms.
- Share *some* element of meaning.

car / bicycle

But, car is *more* similar to truck

cow / tiger

But, cow is *more* similar to chicken

Distributional Hypothesis

"You shall know a word by the company it keeps!

-Firth (1957)

N words around the target work, N can be decided.

- Words that occur in the same **contexts** tend to have similar meaning.
  - E.g. car/bicycle

A bottle of ***Tesgüino*** *is on the table.*
Everybody likes ***tesgüino****.*
Tesgüino *makes you drunk.*
We make ***tesgüino*** *out of corn.*

What could **tesgüino** mean?

- *[ ] makes you drunk.*
- After *bottle of*
- Other words seen in this context? Alcohol, wine, whiskey, etc.

Distributional Hypothesis

"You shall know a word by the company it keeps!

-Firth (1957)

N words around the target work, N can be decided.

- Wo ... g.
  - E ... ?

A bott
Everyb
***Tesgüino** makes you drunk.*
*We make **tesgüino** out of corn.*

Use information about shared context to decide dimensions of word vector?

- After *bottle of*
- Other words seen in this context? Alcohol, wine, whiskey, etc.

Distributional Hypothesis

- Two words are similar if they occur in similar **contexts**. Represent **context** as a vector?

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

Word-word co-occurrence matrix

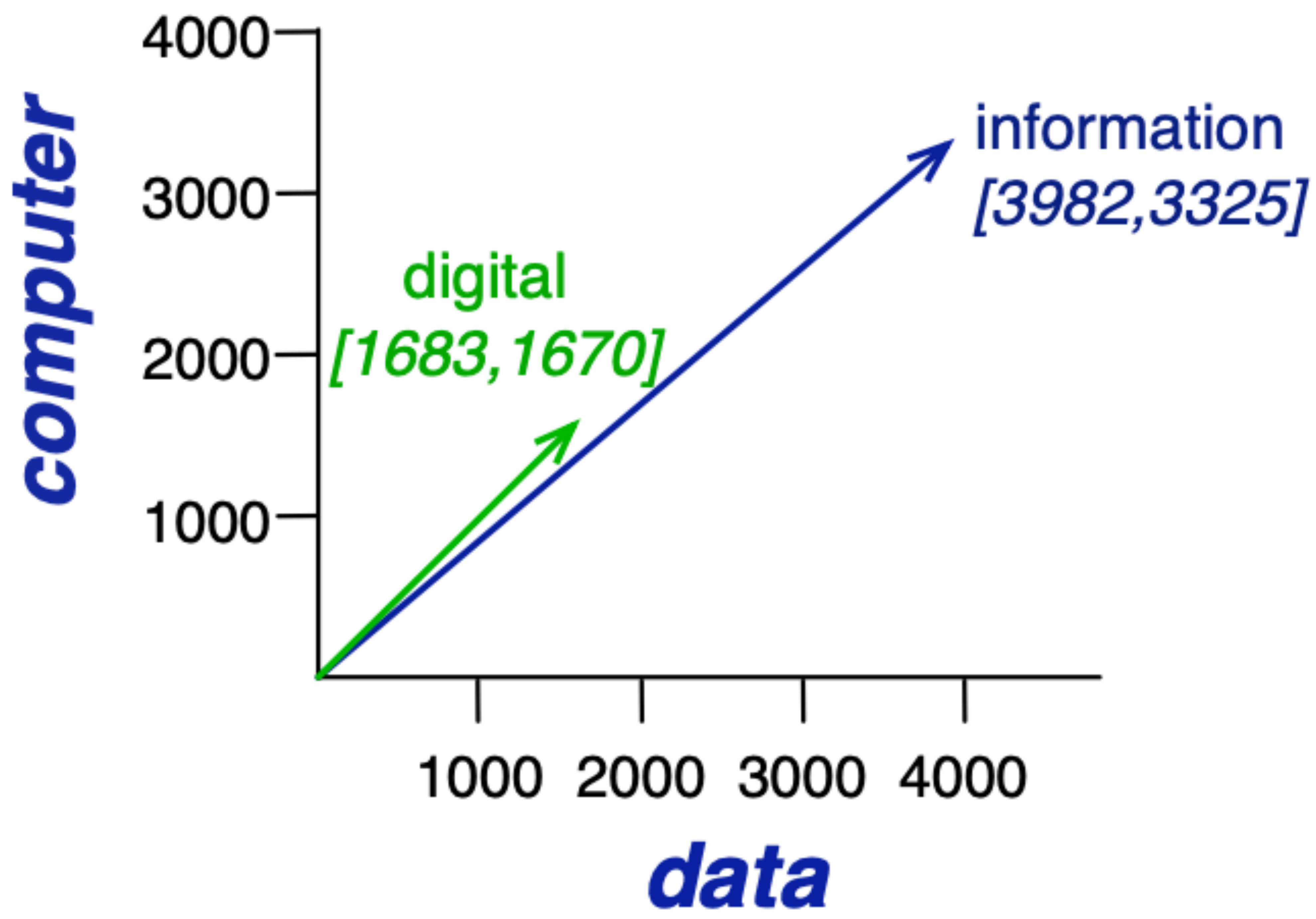
- Two words are similar if they occur in similar **contexts**. Represent **context** as a vector?

*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	...
stawberry	0	...	0	0	1	60	29	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Word-word co-occurrence matrix

	aardvark	...	computer	data	result	pie	sugar	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...



Properties of these vectors?

- Size = |vocabulary| , say 10K -50K
- Sparse

Cosine Similarity Metric

- Cosine similarity of vectors $\vec{w}$ and $\vec{v}$.

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

v_i is the count of word v in context of word i
w_i is the count of word v in context of word i

- Cosine similarity is 1/-1 when $\vec{w}$ and $\vec{v}$ point in the same/opposite direction.
- Cosine similarity is 0 when $\vec{w}$ and $\vec{v}$ are orthogonal.

Issues with raw frequency counts

	aardvark	...	computer	data	result	pie	sugar	...	**a**
cherry	0	...	2	8	9	442	25	...	**7543**
stawberry	0	...	0	0	1	60	29	...	**9121**
digital	0	...	1670	1683	85	5	4	...	**6923**
information	0	...	3325	3982	378	5	13	...	**8345**

- Overly frequent words like "a", "the", "it", etc. are not informative, they co-occur frequently with most words.
- They dominate cosine similarity computation.

tf-idf

- **tf: term frequency**

$$tf_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

count(t, d) = # occurrences of word t in doc d (context in our case).

- **idf: inverse document frequency**

$$\text{idf}_t = \log\left(\frac{N}{\text{df}_t}\right)$$

df_t = # documents containing word t.
N = # documents

What words will have low idf?

- **tf-idf**

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

Value of a word t in document d

dense word vectors

Dense word vectors

- What is the dimension of tf-idf vectors?
- dense word vectors: represent words as an **embedding** in the vector space.
  - Typically lower dimension than tf-idf (e.g. deepseek r1's embedding size is 7168)
  - Not sparse.
  - Dimensions do not have intuitive meanings (e.g. "denote co-occurrence with word j" as in sparse vectors.)
- How do we learn vector embeddings?
  - Multiple approaches: Skip-grams, CBOW.

Intuition: Skip-gram Model

- Word2Vec: Popular embedding methods from 2013.
- Very fast to train.
- Idea:
  - *Instead of:* counting how often a word w appears near ``cherry''.
  - Train a binary classifier on a **prediction** task:

Is word w *likely* to occur near word ``cherry''?

$P(+ \mid w, c) \leftarrow$ c is a context word of w

$P(- \mid w, c) = 1 - P(- \mid w, c) \leftarrow$ c is not a context word of w

Q: Why do we care about this task?

Intuition: Skip-gram Model

$P(+ \mid w, c) \leftarrow$ c is a context word of w

$P(- \mid w, c) = 1 - P(+ \mid w, c) \leftarrow$ c is not a context word of w

- From distribution hypothesis, we want:

$$P(+ \mid w, c) \approx \mathbf{c} \,.\, \mathbf{w}$$

Word vector for word w

Context vector for word c

This is not a probability.

$$P(+ \mid w, c) = \sigma(\mathbf{c} \,.\, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \,.\, \mathbf{w})}$$

Possible strategy

- Let's represent words as vectors of some length.
- Let's initialize those vectors w/ say 300 dimensions.
  - Total dimension of embeddings |V| * 300
- Get some training data:
  - (w, c) pairs of words that co-occur (+)
  - (w, n) pairs of words that do not co-occur (-):
- .Use a learning algorithm to adjust these word vectors such that
  - **Maximize** the similarity of (w, c) pairs with label +
  - **Minimize** the similarity of (w, n) pairs with label -

Skip-gram with negative sampling (SGNS)

- **Training Data?**
  - This is freely available! Use any text as supervision data.

<apricot jam>, +

<apricot a>, +

<apricot tablespoon>, +

<apricot of>, +

... lemon, a **tablespoon of apricot jam a** pinch ...

c1 c2 w c3 c4

Assume context words are in +/- 2 word window

<apricot aardvark>, -

<apricot digital>, -

- Negative data?
  - Randomly sample words other words from the vocab.

- No need for hand labeled supervision data.
- Similar idea as language modeling!

Possible strategy

- Let's represent words as vectors of some length.
- Let's initialize those vectors w/ say 300 dimensions.
    - Total dimension of embeddings |V| * 300
- Get some training data:
    - (w, c) pairs of words that co-occur (+)
    - (w, c) pairs of words that do not co-occur (-):
- .Use a learning algorithm to adjust these word vectors such that
    - **Maximize** the similarity of (w, c) pairs with label +
    - **Minimize** the similarity of (w, c) pairs with label -

Skip-gram with negative sampling (SGNS)

$$P(+|w,c) = \frac{\exp(\mathbf{c}\,.\,\mathbf{w})}{1+\exp(\mathbf{c}\,.\,\mathbf{w})}$$

$$P(-|w,c) = \frac{1}{1+\exp(\mathbf{c}\,.\,\mathbf{w})}$$

- Classification model. What is our objective? Maximize log likelihood of the data.

$$\sum_{(w,c)\in+} \log P(+|w,c) + \sum_{(w,c)\in-} \log P(-|w,c)$$

Q: Why are the features and what are the weights here?

- Focusing on one target word

$$L(\theta) = \log P(+|w,c_{\text{pos}}) + \log P(-|w,c_{\text{neg}})$$

negative word

Skip-gram with negative sampling (SGNS)

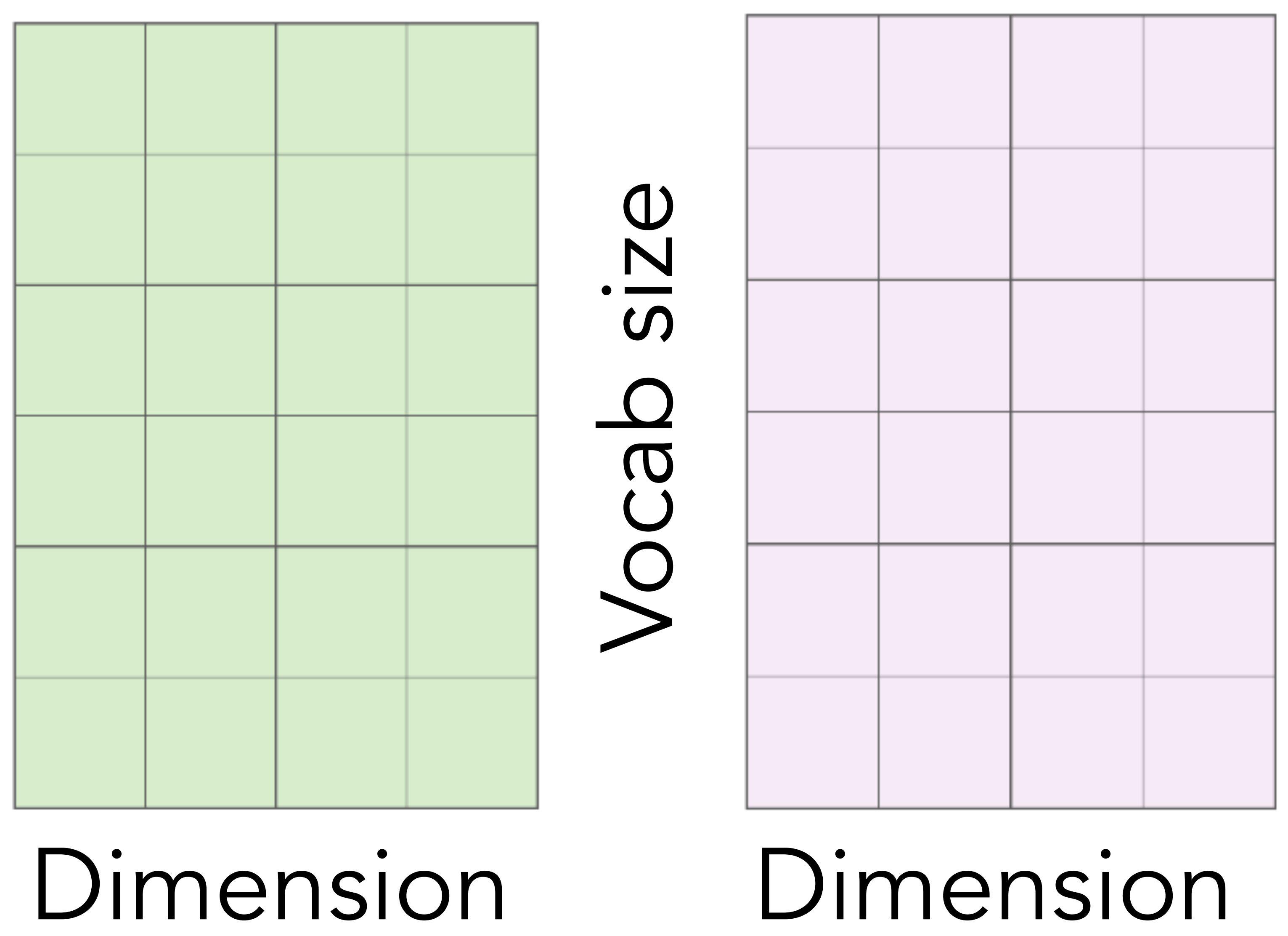
- Focusing on one target word (log likelihood)

$$P(+\mid w, c) = \frac{\exp(\mathbf{c}\,.\,\mathbf{w})}{1+\exp(\mathbf{c}\,.\,\mathbf{w})}$$

$$P(-\mid w, c) = \frac{1}{1+\exp(\mathbf{c}\,.\,\mathbf{w})}$$

$$L(\theta) = \log P(+\mid w, c_{\text{pos}}) + \log P(-\mid w, c_{\text{neg}})$$

$$= \log \frac{\exp(\mathbf{c}_{pos}\,.\,\mathbf{w})}{1+\exp(\mathbf{c}_{pos}\,.\,\mathbf{w})} + \log \frac{1}{1+\exp(\mathbf{c}_{neg}\,.\,\mathbf{w})}$$



Putting it all together: Skip-gram Also

- Initialize C^o, W^0
- For each training sample:

$$\frac{\partial L}{\partial \mathbf{c}_{\text{pos}}} = [\sigma(\mathbf{c}_{\text{pos}} . \mathbf{w}) - 1]\ \mathbf{w}$$

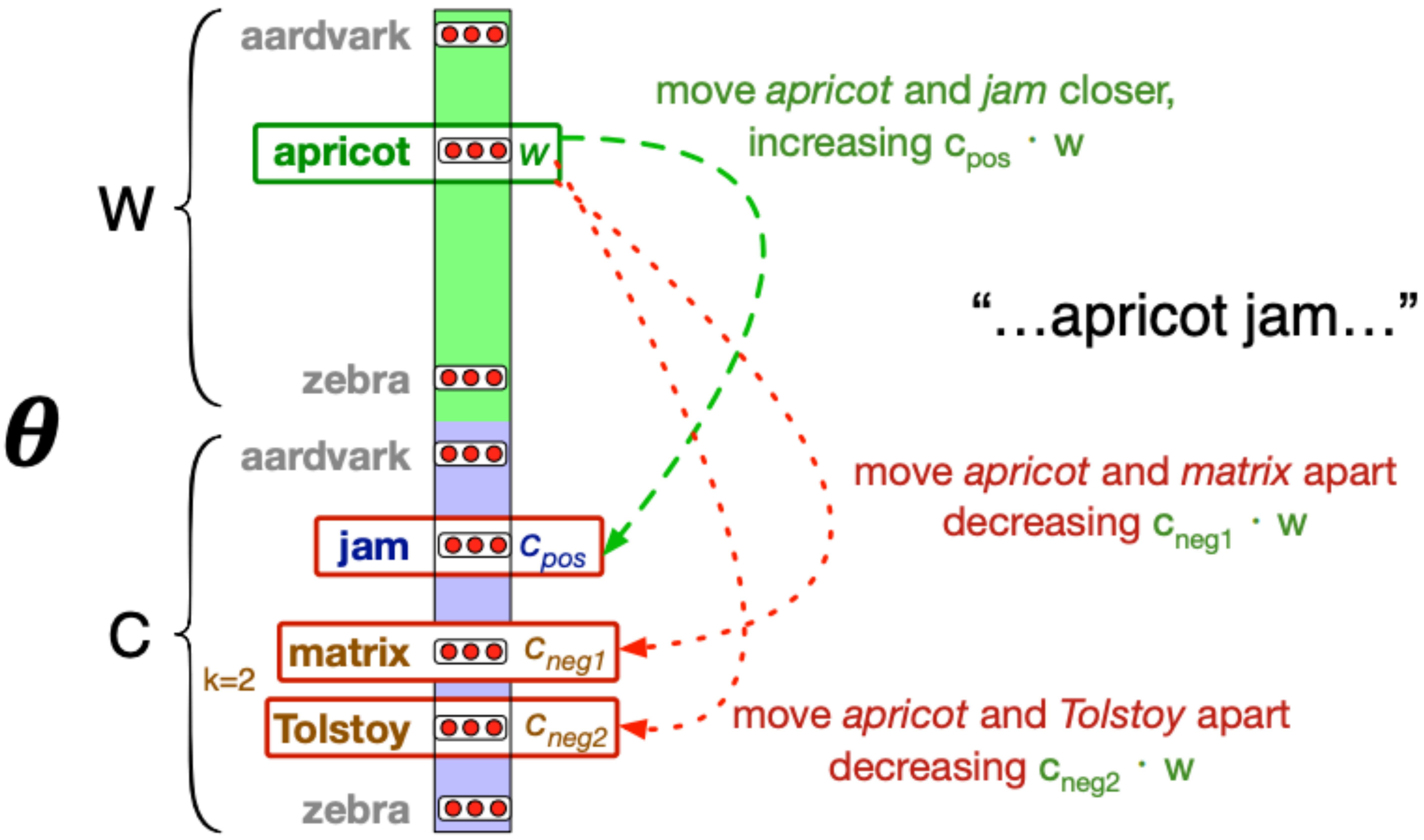
$$\frac{\partial L}{\partial \mathbf{c}_{\text{neg}}} = [\sigma(\mathbf{c}_{\text{neg}} . \mathbf{w})]\ \mathbf{w}$$

$$\frac{\partial L}{\partial \mathbf{w}} = [\sigma(\mathbf{c}_{\text{pos}} . \mathbf{w}) - 1]\ \mathbf{c}_{\text{pos}} + [\sigma(\mathbf{c}_{\text{neg}} . \mathbf{w})]\ \mathbf{c}_{\text{neg}}$$

Self-study: derive this!

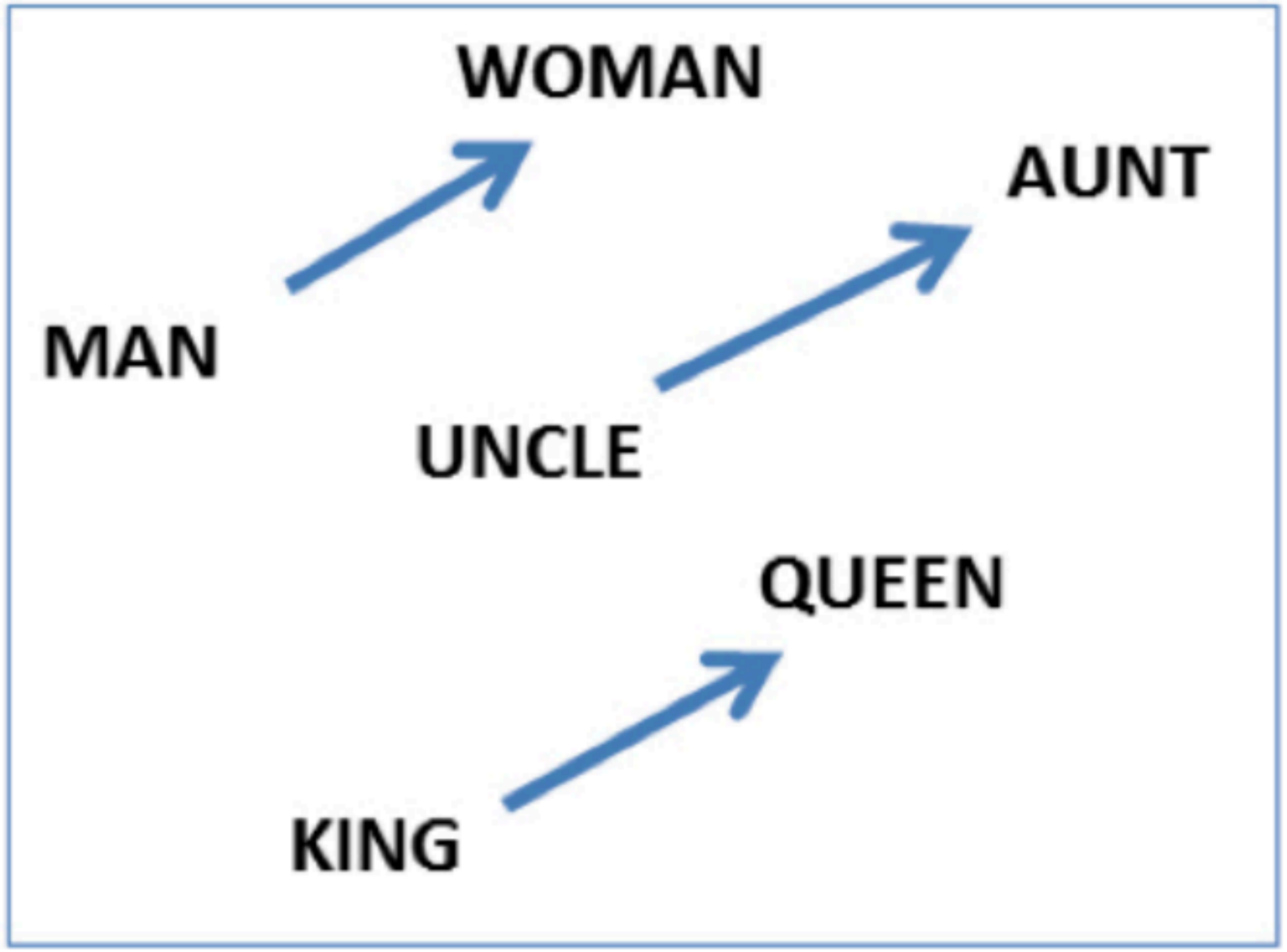
- Gradient update!

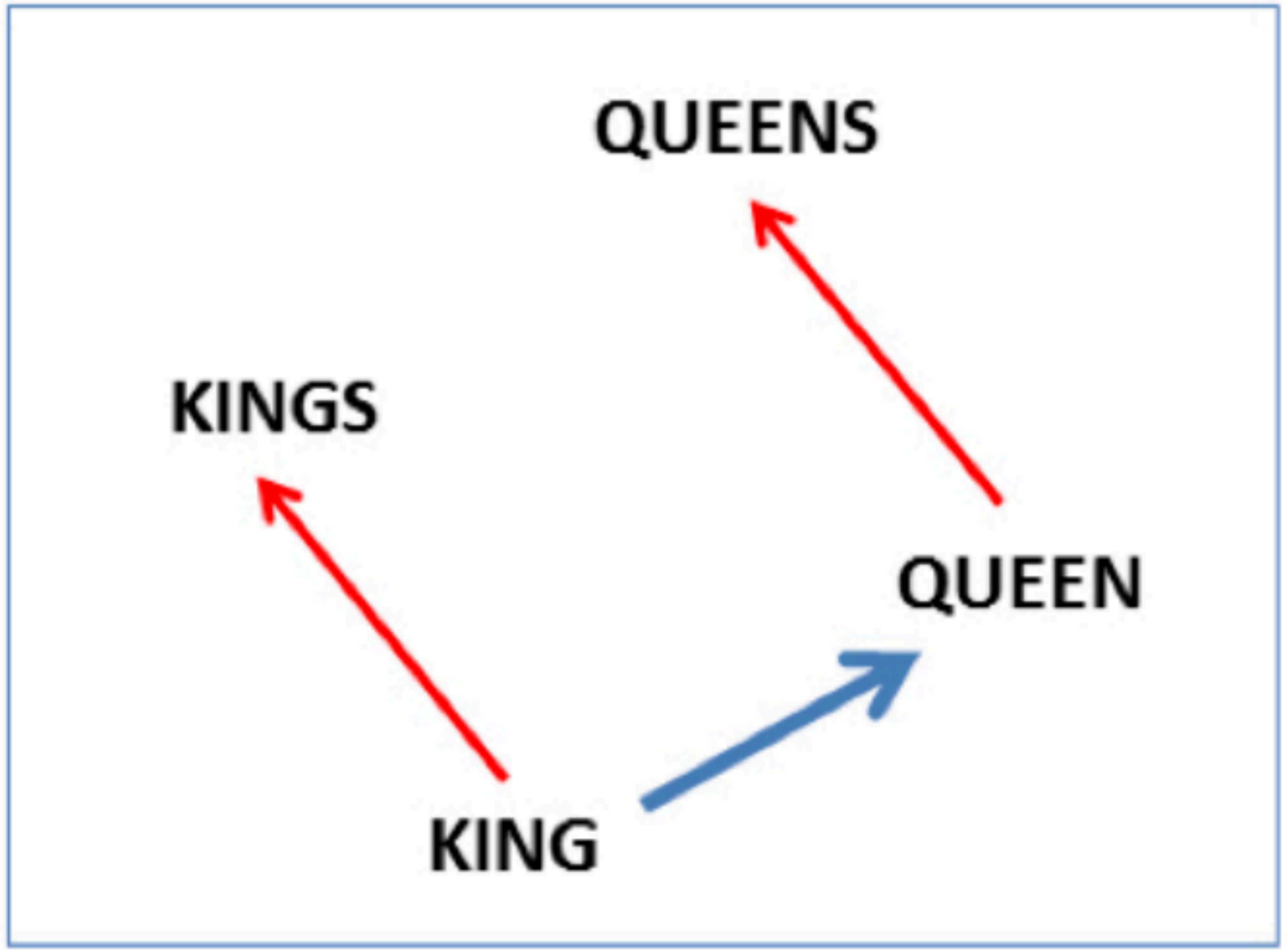
Putting it all together: Skip-gram Also



- To represent word w, we can
  - Concatenate $\mathbf{c}^i$ and $\mathbf{w}^i$
  - Keep $\mathbf{w}^i$

Word2Vec: Embeddings capture analogies

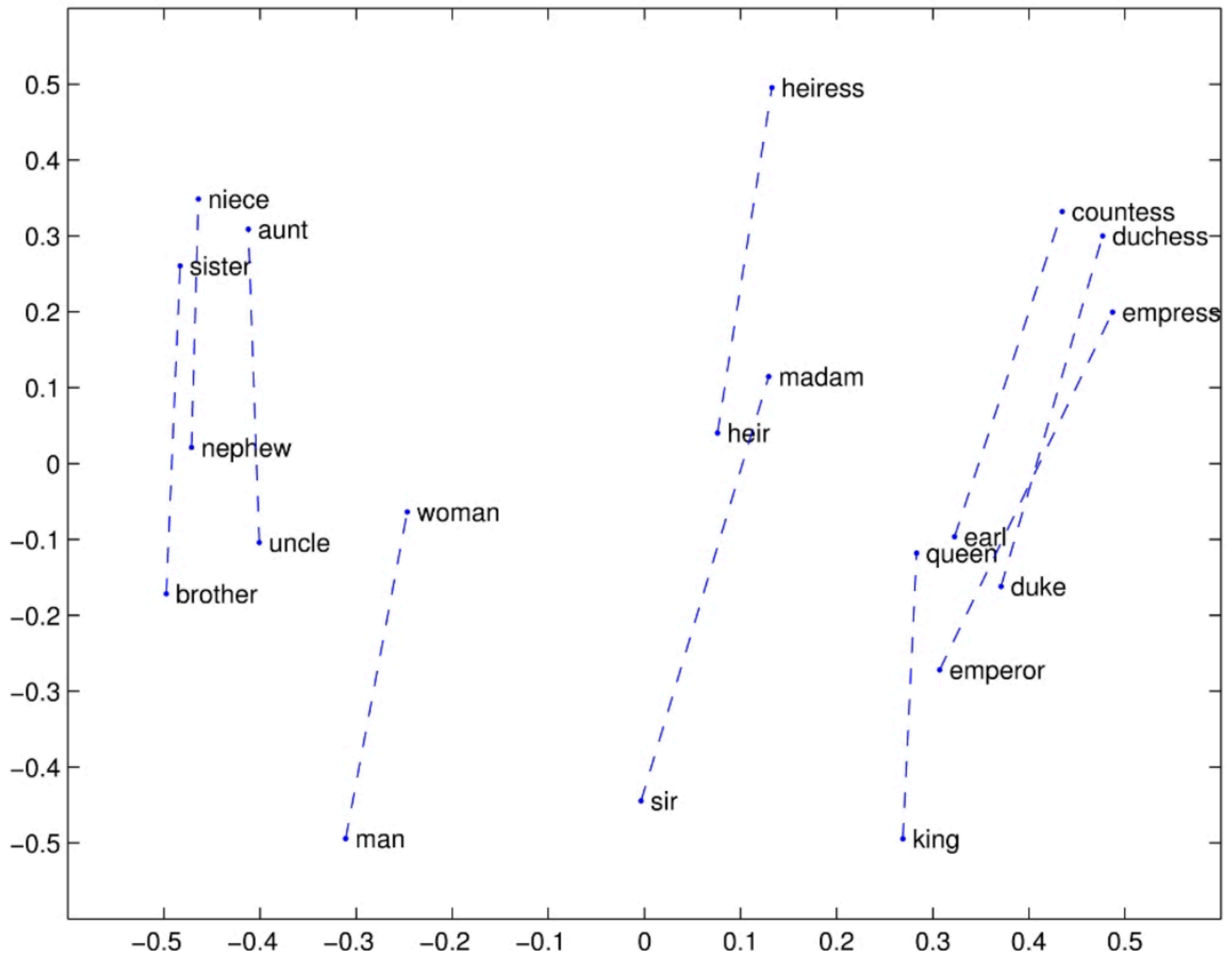




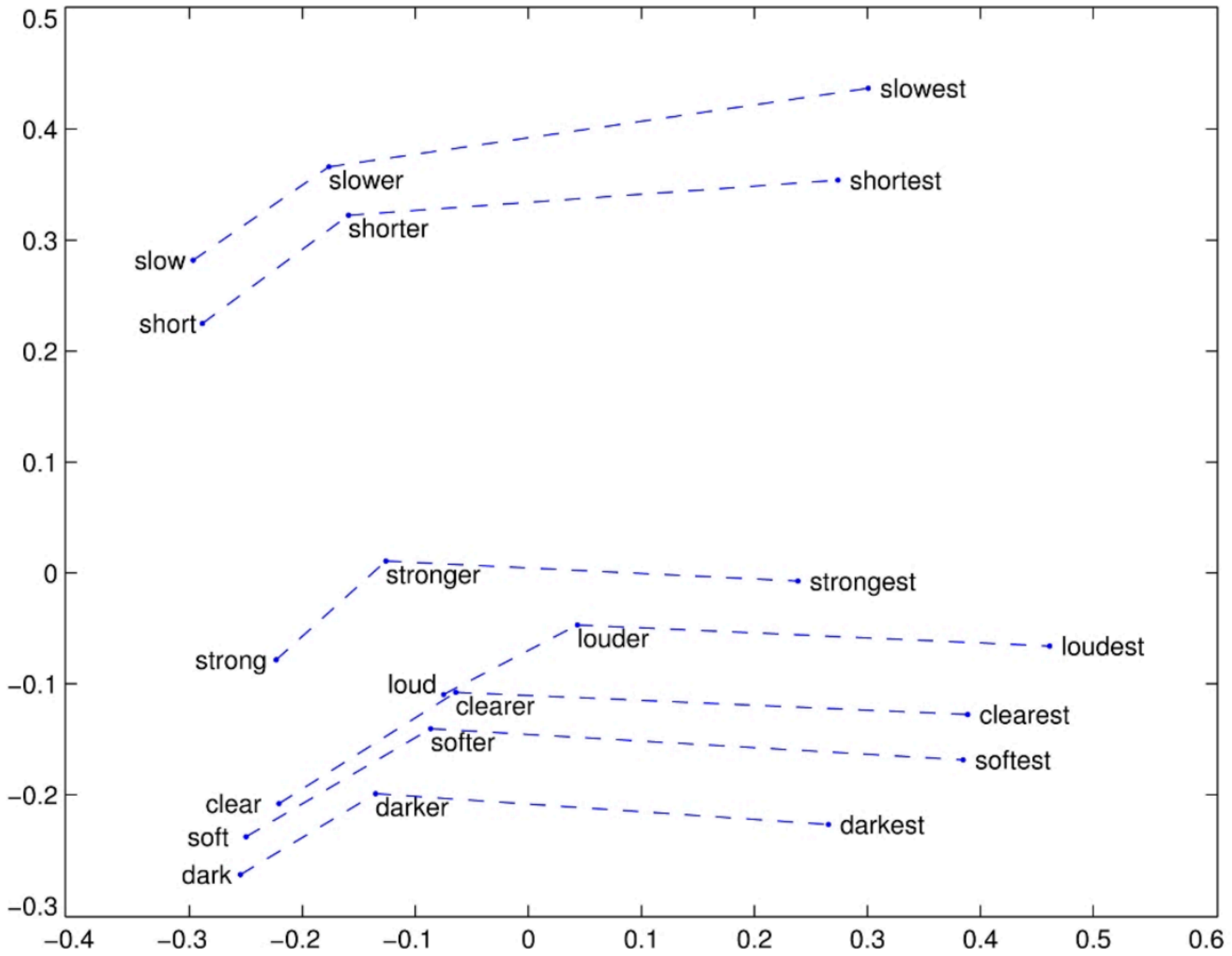
vector(king) - vector(man) + vector(woman) ≈ vector(queen)

vector(Paris) - vector(France) + vector(Italy) ≈ vector(Rome)

Word2Vec: Embeddings capture relations



Word2Vec: Embeddings capture relations



Word2Vec: Embeddings capture biases!

vector(doctor) - vector(father) + vector(mother) ≈ vector(nurse)

vector(man) - vector(computer programmer) + vector(woman) ≈ vector(homemaker)

Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

https://arxiv.org/pdf/1607.06520.pdf

Word2Vec: Embeddings capture biases!

vector(doctor) - vector(father) + vector(mother) ≈ vector(nurse)

vector(man) - vector(computer programmer) + vector(woman) ≈ vector(homemaker)

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

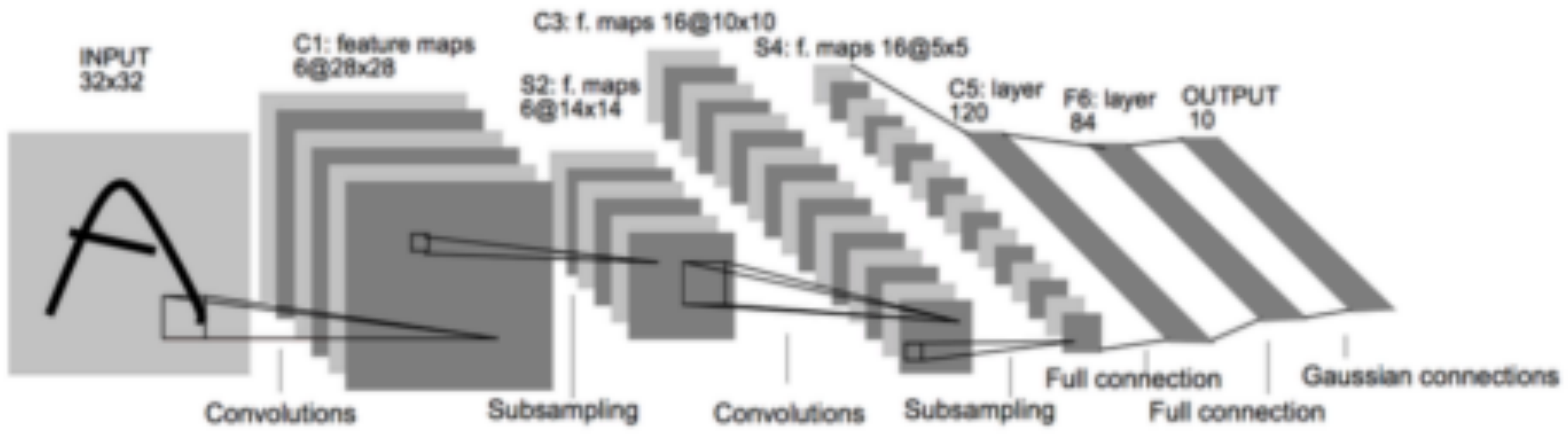
https://arxiv.org/pdf/1607.06520.pdf

Today

- Recap: Logistic Regression
- Word Vectors or Word Embeddings
  - Similarity?
  - TF-IDF
  - Word2Vec
- Feed Forward Neural Networks

NN "dark ages"

- Neural Network algorithms date from the 80s.
- ConvNets: Yann LeCun applied them to MNIST data in 1998.



- LSTMs (Long Short Term Memory) Networks: Hochreiter and Schmidhuber 1997

Slide Credit: Greg Durrett

2008-2013: A glimmer of light

- Collobert and Weston 2011: "**NLP (almost) from scratch**
  - Feedforward NNs can replace "feature engineering"
- AlexNet in 2012 for Image Classification

- **2014 onwards, things start working!**
- Kim (2014): ConvNets for NLP sentiment classification
- Sutskever et al. (2014): seq-to-seq models for machine translations.
- 2015: all tasks attacked by neural-first approach
- 2018-2019: NLP entered the pre-training paradigm (ELMo, GPT, BERT)
- 2020+: the emergence of large language models (GPT-3, ChatGPT, etc)

Slide Credit: Greg Durrett

Slide Acknowledgements

- Earlier versions of this course offerings including materials from Claire Cardie, Marten van Schijndel, Lillian Lee.
- NLP course by Mohit Iyyer.