Self-Supervised Learning Word Representations

CSCI 601 471/671
NLP: Self-Supervised Models

https://self-supervised.cs.jhu.edu/sp2023/

[Slide credit: Chris Manning, John Canny, Reda Bouadjenek, and many others]

HW Poll

- How was it?

- **A note about solutions:**
  - We will not release solutions.
  - But we will be as clear as we can in our grading.
  - If there are any lingering questions about homework solutions, come discuss during office hours.

Recap

- Inherently, there is no meaning to symbols:

"apple" ≈ 01100001011100000111000001101100011001 01

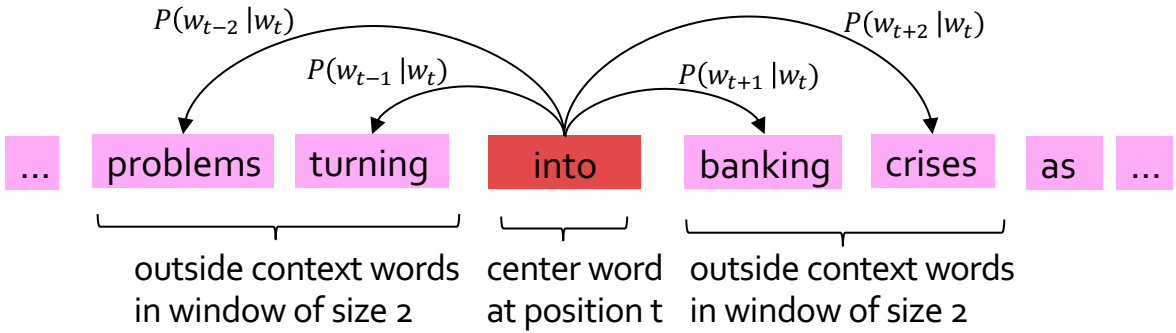
- Lots of literature on mapping **symbols** to their **"meaning"** (e.g., formal semantics)
- **Focus of this class:** distributional semantics —learning ***some*** meaning by regurgitating language use.
  - **Simplest form:** learning word representations via Word2Vec

Recap: Word2Vec Probabilities

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)}$$

If o and c tend to co-occur, they should have similar representation.

$$P(w_{t-2}|w_t) = p(\text{problems}|\text{into}) = \frac{\exp\left(u_{\text{problems}}^T v_{\text{into}}\right)}{\sum_{x \in V} \exp(u_x^T v_{\text{into}})}$$



Recap: Word2Vec Objective Function

- Minimize the objective function (log-likelihood) via Gradient Descent.

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P(w_{t+j}\,|\,w_t;\theta)$$

text length

window size

- Derivatives of "center" vectors: $\frac{\partial}{\partial v_c}J(\theta) = ?$
- Derivatives of "outside" vectors: $\frac{\partial}{\partial u_c}J(\theta) = ?$

We will derive the gradients with respect to the "center" vectors. Similar calculations for "outside" vectors (homework!!)

Chapter: Self-Supervised Meaning of "Words": Continued

1. ~~Human language and word meaning~~
2. ~~Word2vec overview~~
3. ~~Word2vec objective function and~~ gradients
4. Inspecting the resulting word vectors
5. Extrinsic evaluation of word vectors

Computing the Gradients for Word2Vec

- Minimize the objective function (log-likelihood):

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P(w_{t+j}\,|w_t;\theta)$$

text length

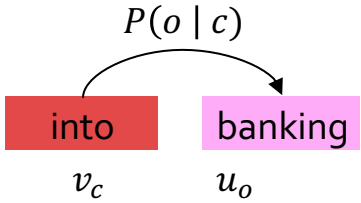
window size

$$\frac{\partial}{\partial v_c}J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \frac{\partial}{\partial v_c}\log P(w_{t+j}\,|\,w_t;\theta)$$

(distributive property)

?

Computing the Gradients for Word2Vec (2)



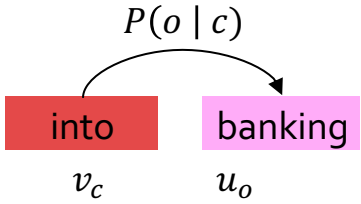
$$\frac{\partial}{\partial v_c} \log P(c \mid o; \theta) = \frac{\partial}{\partial v_c} \left[\log \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)} \right]$$

$$= \frac{\partial}{\partial v_c} \left[\log \exp(u_o^T v_c) - \log \sum_{x \in V} \exp(u_x^T v_c) \right]$$

$$= \frac{\partial}{\partial v_c} [\log \exp(u_o^T v_c)] - \frac{\partial}{\partial v_c} \left[\log \sum_{x \in V} \exp(u_x^T v_c) \right]$$ (distributive property)

$$\frac{\partial}{\partial v_c} [\log \exp(u_o^T v_c)] = \frac{\partial}{\partial v_c} [u_o^T v_c] = u_o$$

Computing the Gradients for Word2Vec (3)



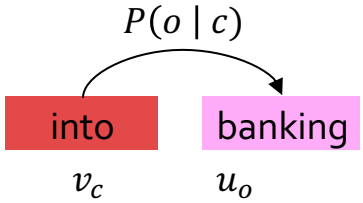
$$\frac{\partial}{\partial v_c} \log P(c \mid o; \theta) = \frac{\partial}{\partial v_c} \left[\log \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)} \right]$$

$$= \frac{\partial}{\partial v_c} \left[\log \exp(u_o^T v_c) - \log \sum_{x \in V} \exp(u_x^T v_c) \right]$$

$$= \frac{\partial}{\partial v_c} [\log \exp(u_o^T v_c)] - \underbrace{\frac{\partial}{\partial v_c} \left[\log \sum_{x \in V} \exp(u_x^T v_c) \right]}_{?}$$

(distributive property)

Computing the Gradients for Word2Vec (4)



$$\frac{\partial}{\partial v_c}\left[\log \sum_{x \in V} \exp(u_x^T v_c)\right] = \frac{1}{\sum_{x \in V} \exp(u_x^T v_c)} \times \frac{\partial}{\partial v_c} \sum_{x \in V} \exp(u_x^T v_c)$$

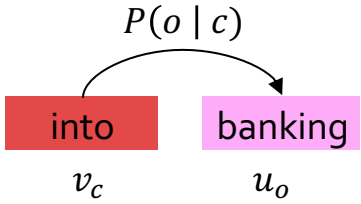
$$\frac{\partial}{\partial x} \log(f(x)) = \frac{1}{f(x)} \frac{\partial f}{\partial x}$$

$$= \frac{1}{\sum_{x \in V} \exp(u_x^T v_c)} \times \sum_{x \in V} u_x \exp(u_x^T v_c)$$

$$\frac{\partial}{\partial x} \exp(f(x)) = \frac{\partial f}{\partial x} \times \exp(f(x))$$

$$= \sum_{x \in V} u_x \times P(x \mid c; \theta)$$

Computing the Gradients for Word2Vec (5)



- Putting things together:

$$\frac{\partial}{\partial v_c} \log P(c \mid o) = u_o - \sum_{x \in V} u_x \times P(x \mid c)$$

the current representation of o

The expected (weighted avg) representation of the words that tend to co-occur with c.

Intuition: This gradient incentivizes representation of o to be more similar to the avg representation of words co-occurring with c.

Pop Quiz

- What is the computational complexity of the followings?
  (assume that $P(o|c)$ is pre-computed we just need to look it up)

$$\frac{\partial}{\partial v_c}\log P(c\,|o) = u_o \;-\; \sum_{x\in V} u_x \times P(x\,|c)$$

(A) O(d)
(B) O(|V|)
(C) O(|V|+d)
(D) O(|V|.d)

$$\frac{\partial}{\partial v_c}J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}}\frac{\partial}{\partial v_c}\log P\left(w_{t+j}\,|w_t;\theta\right)$$

(A) O(T)
(B) O(T.m)
(C) O(T.m.|V|)
(D) O(T.m.|V|.d)

Stochastic Gradient Descent

- Challenge: $J(\theta)$ is a function of all windows in the corpus (potentially billions!)
  - So computing $\nabla_\theta J(\theta)$ is very expensive to compute
  - You would wait a very long time before making a single update!
- Very bad idea!

$$\frac{\partial}{\partial v_c} J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \frac{\partial}{\partial v_c} \log P(c \mid o)$$

- Solution: Stochastic gradient descent (SGD)
  - Repeatedly **sample windows and instances**, and update after each one

- This resolves the complexity of $\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}}$ but what about $\frac{\partial}{\partial v_c} \log P(c \mid o)$?

Skip-gram with Negative Sampling

- Let's see where the complexity is:

The expensive computation: O(|V|.d)

$$\log P(o|c) = \log \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)} = \log \exp(u_o^T v_c) - \log \sum_{x \in V} \exp(u_x^T v_c)$$

Skip-gram with Negative Sampling

- Let's see where the complexity is:

The expensive computation: O(|V|.d)

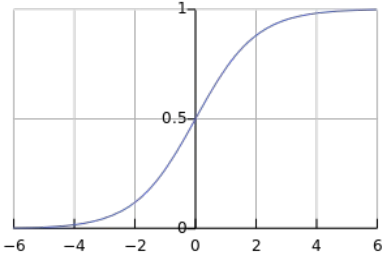
$$\log P(o|c) = \log \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)} = \log \exp(u_o^T v_c) - \log \sum_{x \in V} \exp(u_x^T v_c)$$

- **Idea:** rather than enumerating over all vocabulary, let's sample!

$$J_{NS}(\theta) = -\log \sigma(u_o^T v_c) - \sum_{k \in \{K\ samples\}} \log \sigma(-u_x^T v_c)$$

  - Maximize the prob that outside word co-occurs w/ the center
  - Minimize the prob of noise/random words far from the center (negatives)

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



Skip-gram with Negative Sampling (2)

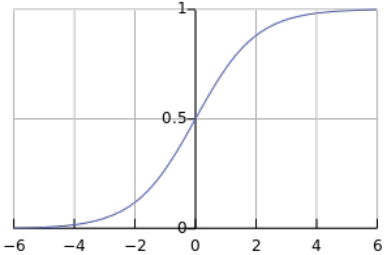
- Have to be careful with sampling negative examples
  - Challenge: uniform sampling will sample a lot of stop-words that are very popular.
- Mikolov et al. proposed to sample: $p(w_i) = f(w_i)^{3/4} \Big/ \sum_j f(w_j)^{3/4}$
  - Assigns more prob to less frequent words. No theory backing, but works!

- **Idea:** rather than enumerating over all vocabulary, let's sample!

$$J_{NS}(\theta) = -\log \sigma(u_o^T v_c) - \sum_{k \in \{K\ samples\}} \log \sigma(-u_x^T v_c)$$

  - Maximize the prob that outside word co-occurs w/ the center
  - Minimize the prob of noise/random words far from the center (negatives)

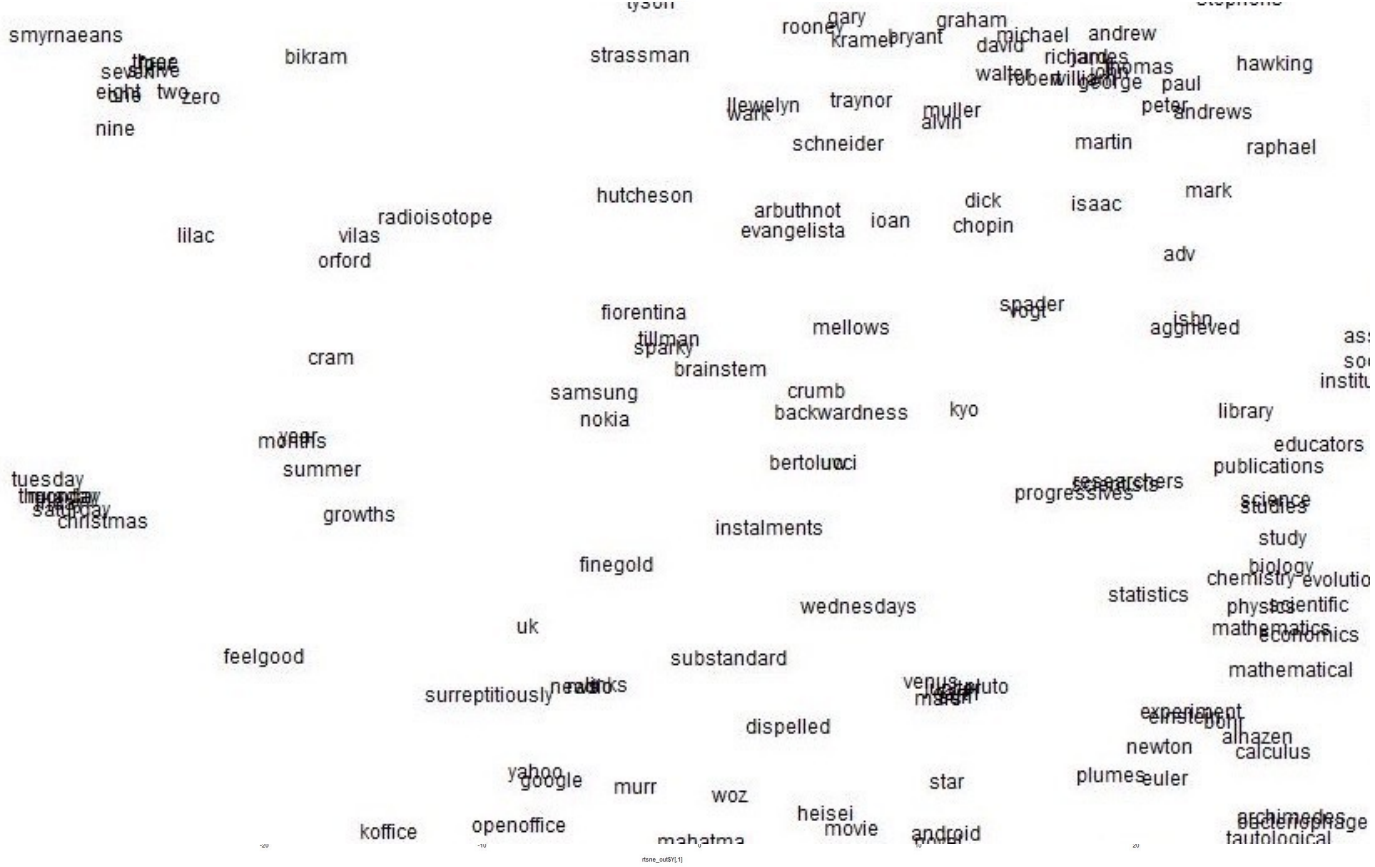
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



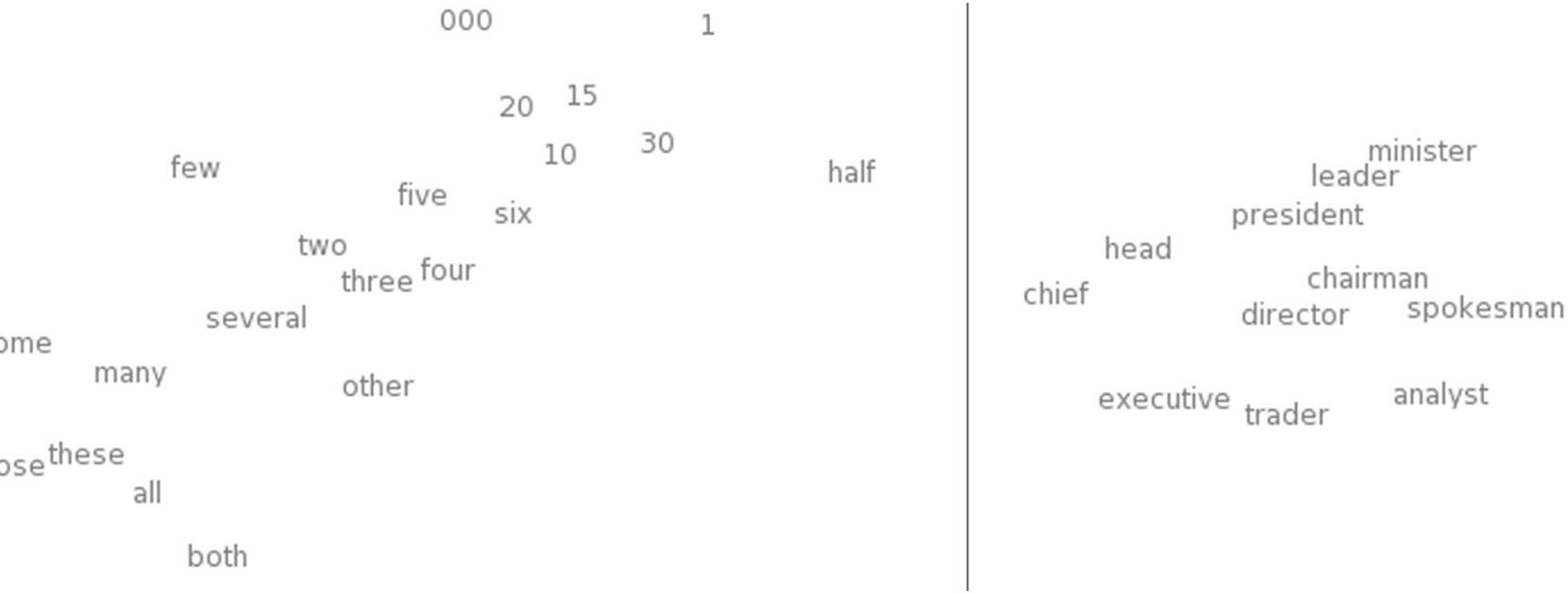
Chapter: Self-Supervised Meaning of "Words": Continued

1. ~~Human language and word meaning~~
2. ~~Word2vec overview~~
3. ~~Word2vec objective function and gradients~~
4. Inspecting the resulting word vectors
5. Extrinsic evaluation of word vectors

Word2Vec maximizes objective function by putting similar words nearby in space



Word2Vec maximizes objective function by putting similar words nearby in space



Left: Number Region

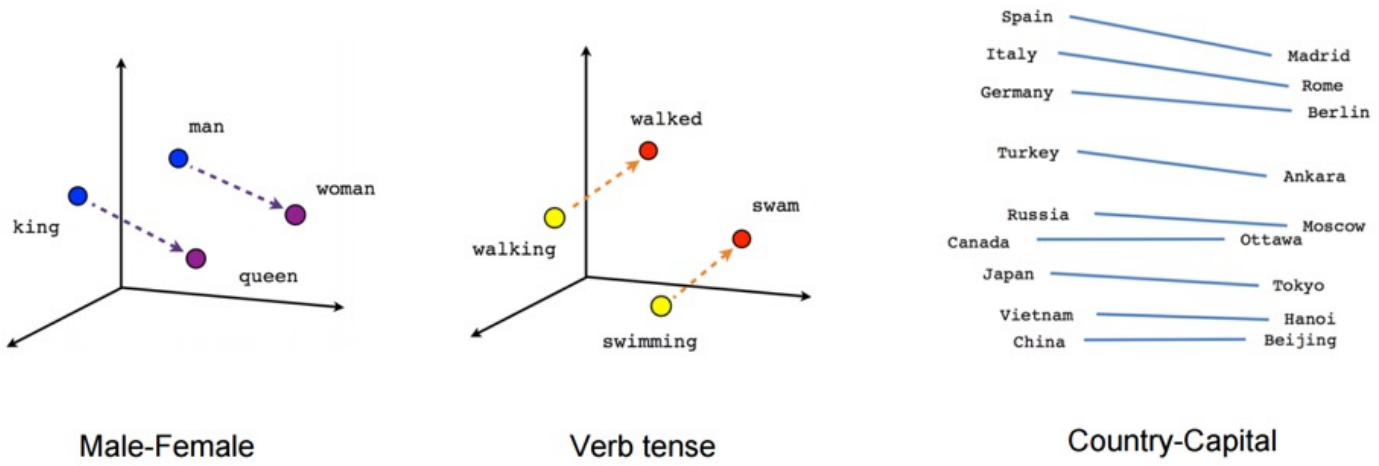
Right: Jobs Region

Also try: https://projector.tensorflow.org/

["Deep Learning, NLP, and Representations" by Chris Olah]

vec(“woman”)–vec(“man”) ≃ vec(“aunt”)–vec(“uncle”)

vec(“woman”)–vec(“man”) ≃ vec(“queen”)–vec(“king”)



Male-Female Verb tense Country-Capital

This can be interpreted as “France is to Paris as Italy is to Rome”.

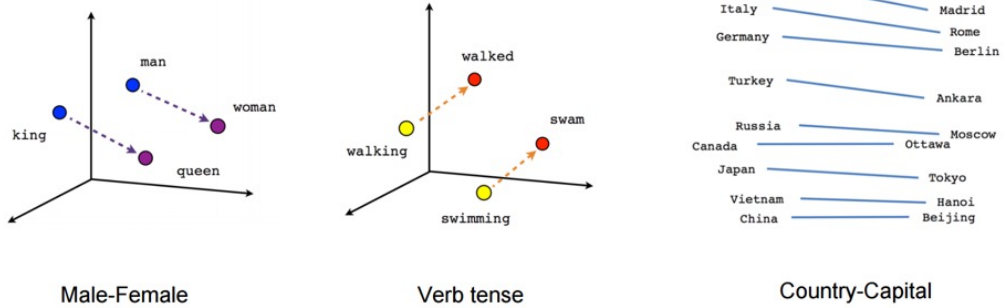
[Linguistic Regularities in Continuous Space Word Representations. Mikolov et al. 2013]

Evaluating Word Vectors: Word Analogies

- Evaluate word vectors by how well their cosine distance after addition captures intuitive analogical relations.

a:b :: c:?

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



Male-Female | Verb tense | Country-Capital

This can be interpreted as "France is to Paris as Italy is to Rome".

[Linguistic Regularities in Continuous Space Word Representations. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[Efficient Estimation of Word Representations in Vector Space. Mikolov et al. 2013]

Mismatch Between Cosine and Dot Product

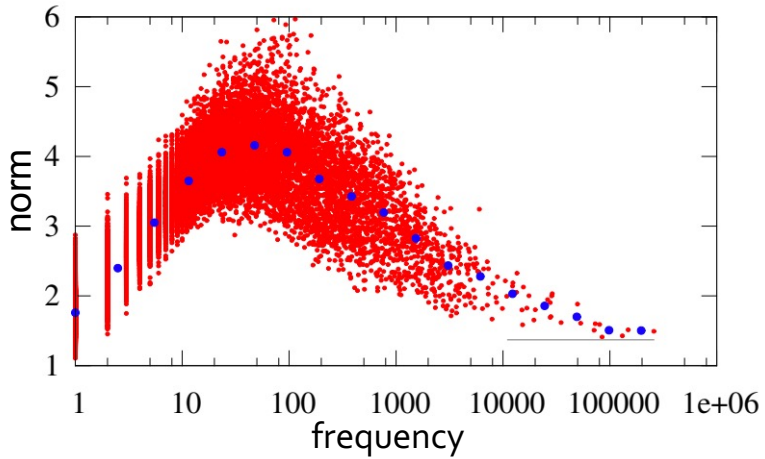
- **Observation:** there a mismatch between Word2Vec objective and cosine distance!

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{x \in V} \exp(u_x^T v_c)}$$

$$\text{distance}(x, y) = \cos(v_x, v_y) = \frac{v_x^T v_y}{\|v_y\| \|v_x\|}$$

1. Why use cosine distance instead of dot product?
   - Term frequencies affect the embedding norms.
   - Without normalization, frequent terms would seem more similar.

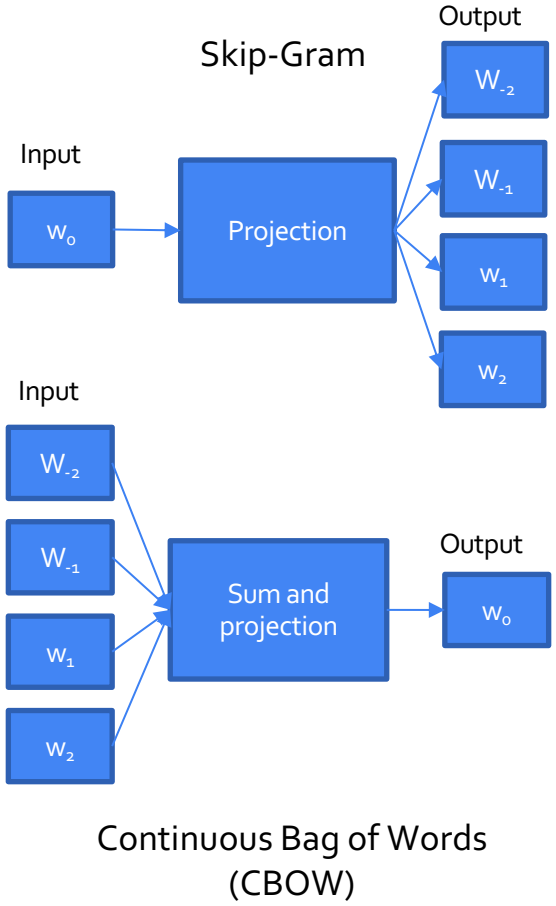
2. Why not change W2V objective to not use cos?
   - ¯_(ツ)_/¯
   - It's possible that the resulting vectors would conflate semantic similarity and frequency.



[Measuring Word Significance using Distributed Representations of Words]

Word2Vec: Variants

- What we just saw is the **Skip-Gram** model.
  - Predict context ("outside") words (position independent) given center
- There is also a **CBOW** (continuous-bag-of-words) variant.
  - Predict center word from (bag of) context words
- Additional efficiency:
  - The current gradient computation w/ softmax function is expensive.
  - Alternative: Negative Sampling



Continuous Bag of Words (CBOW)

Word Embeddings: Big Picture

- There are a variety of approaches:

- **Gradient-based algorithms:**
  - Skip-gram/CBOW (Mikolov et al.), GloVe (Pennington et al.)
  - NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Minh & Hinton)

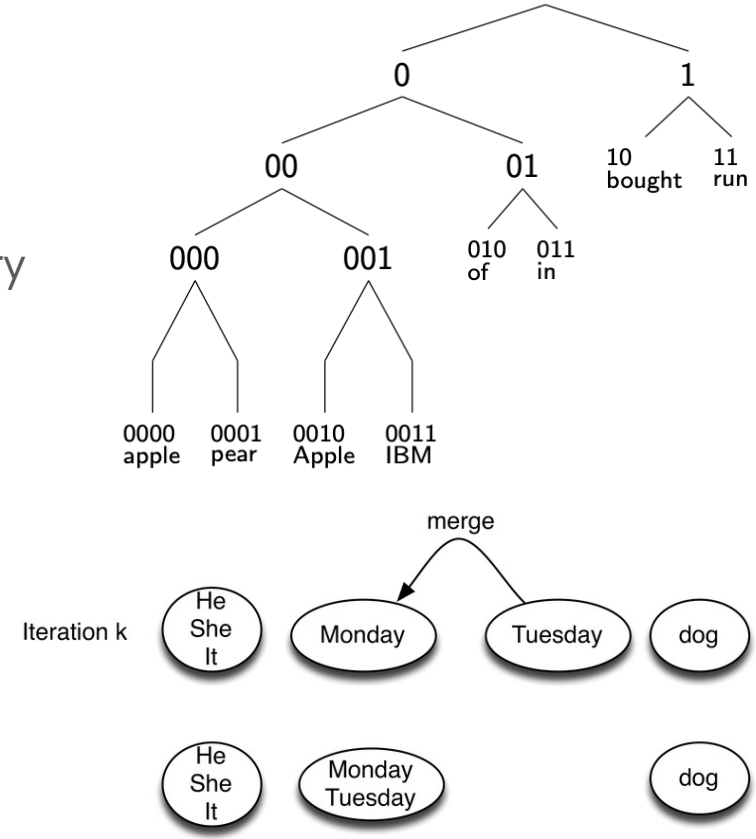
Scales with corpus size
Better representations Captures complex patterns

- **Count-based:**
  - LSA, HAL, COALS, Hellinger-PCA (Lund & Burges; Rohde et al; Lebret & Collobert; Deerwester et al)
  - Brown clustering (Brown et al.)

Fast-ish training
Sub-par representations

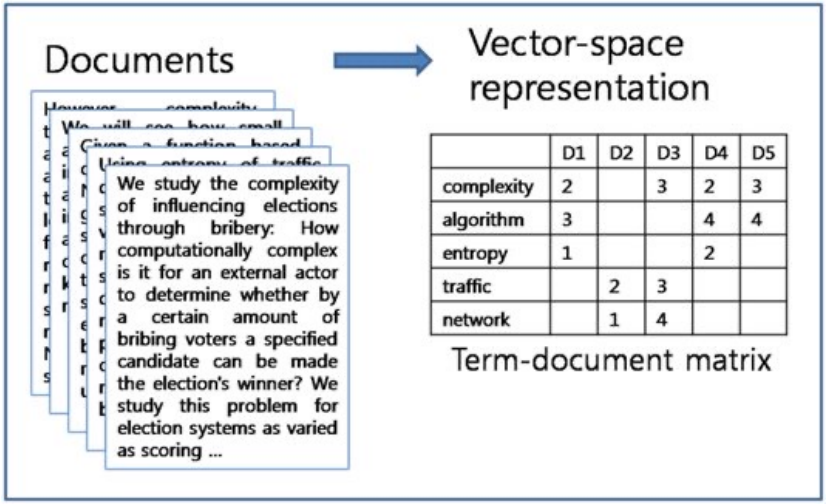
Brown Clusters

- Creates a binary tree for all words in a dictionary
- Algorithm sketch:
  1. Initialize with isolated nodes (words)
  2. Iteratively merge subtrees, so as to maximize some probabilistic notion of co-occurrence.
  3. Continue until everything connected
- Resulting word representation are sequence of 0's and 1's connecting a word to the tree root
- Note:
  - Result is hard cluster
  - Runs in O(|V|^3)



[Class-Based n-gram Models of Natural Language, Brown et al. 1992]

Latent Semantic Analysis



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

- Alg sketch:
  1. Create **word-document co-occurrence matrix**: each value is the number of appearances of that term in that doc.
  2. Use SVD (or a similar matrix decomposition alg) to create low-dimensional representation for words

$$A_{mxn} \approx U_{mxr} \quad \Sigma_{rxr} \quad V^T_{rxn}$$

Chapter: Self-Supervised Meaning of "Words": Continued

1. ~~Human language and word meaning~~
2. ~~Word2vec overview~~
3. ~~Word2vec objective function and gradients~~
4. Inspecting the resulting word vectors
5. Extrinsic evaluation of word vectors

Extrinsic Evaluation of Word Vectors: Classification Recap

- Supervised learning: we have a training dataset consisting of samples

$$D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$$

- $\mathbf{x}_i$ are inputs, e.g., words, sentences, documents, etc.
- y_i are labels (one of C classes) we try to predict, for example:
  - Sentiment labels (+/–), named entity types, buy/sell decisions, word senses, etc.

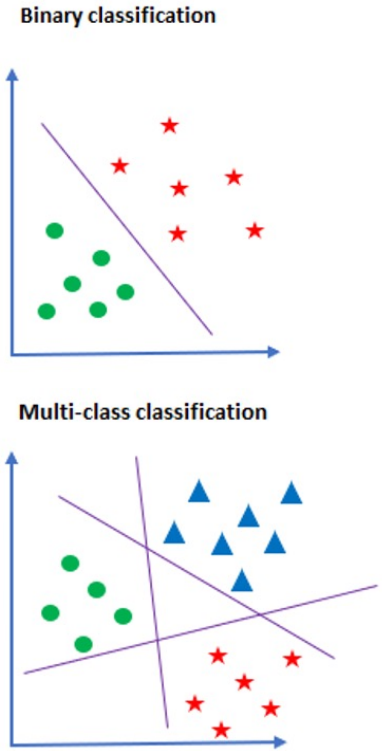
Classification Intuition

- Training data: $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$

- Linear separators:
  - Visualization of input vectors in 2D space
  - Linear decision boundary (hyperplane) for two classes:

$$f(\mathbf{x}) = \boldsymbol{w}.\mathbf{x}$$

- **Model:** multiclass classification, each $y_i \in \{0, \dots, C\}$
  - Softmax classifier
  - Acts like a combination of multiple linear classifiers



Classification with Softmax Classifier (Logistic Regression)

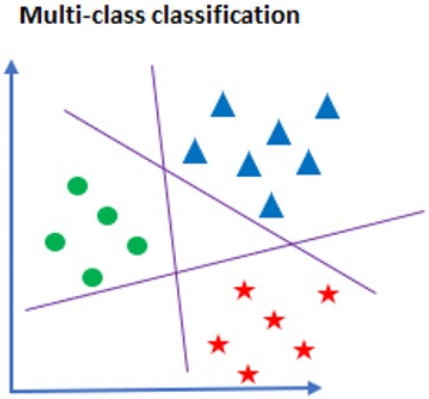
1. Compute the score assigned to each class:
   $f_c(\mathbf{x}) = \boldsymbol{w}_c.\mathbf{x}\ \ \forall c \in C$ and parameters: $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots., \boldsymbol{w}_C]^\top \in \mathbb{R}^{C \times d}$

2. Pass them through the Softmax to get probabilities:

$$\sigma(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \boldsymbol{z} = [z_1, \ldots., z_C]$$

Putting it together:

$$p(x|\boldsymbol{W}) = \frac{\exp(\boldsymbol{w_c}.\mathbf{x})}{\sum_{\boldsymbol{c'}} \exp(\boldsymbol{w_{c'}}.\mathbf{x})}, \qquad \boldsymbol{W} = [\boldsymbol{w}_1, \ldots., \boldsymbol{w}_C]^\top$$



Note: during inference we can just pick the class with the highest score $\hat{y} = \ \mathrm{argmax}_{c \in C} \boldsymbol{w}_c.\mathbf{x}$

Classification with Softmax Classifier (Logistic Regression)

- **Training:** for a collection of training example $(\mathbf{x}_i, y_i)$ optimize the parameters $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C]$ to
  - maximize the probability of the correct class y_i,
  - minimize the negative log probability of that class y_i: $-\log \frac{\exp(\boldsymbol{w}_{y_i}.\mathbf{x}_i)}{\sum_c \exp(\boldsymbol{w}_c.\mathbf{x}_i)}$

- Objective for full dataset $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$: $J(\boldsymbol{W}) = \frac{1}{N}\sum_i -\log \frac{\exp(\boldsymbol{w}_{y_i}.\mathbf{x}_i)}{\sum_c \exp(\boldsymbol{w}_c.\mathbf{x}_i)}$

- Minimization via gradient descent:
  - Programming assignment for HW2!
  - Word embeddings for sentiment classification

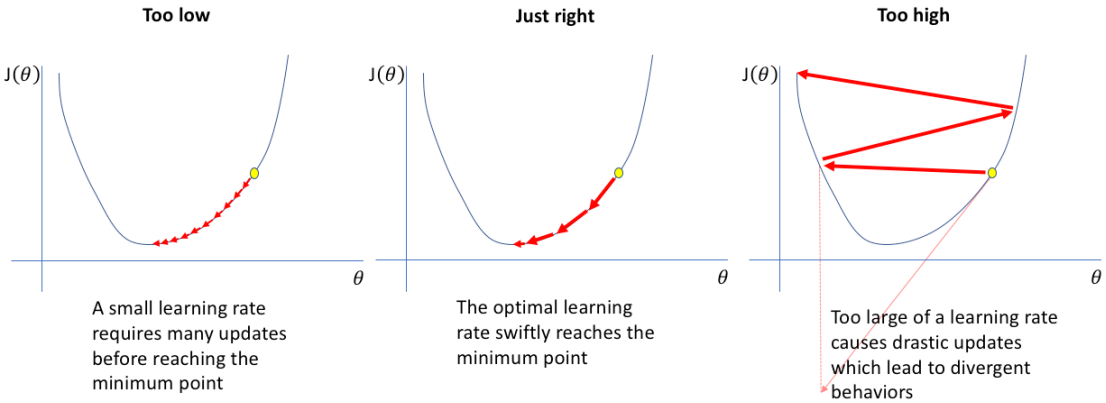
$$\nabla_{\boldsymbol{W}} J(\boldsymbol{W}) = \begin{bmatrix} \nabla_{\boldsymbol{w}_1} \\ \vdots \\ \nabla_{\boldsymbol{w}_C} \end{bmatrix} \in \mathbb{R}^{C \times d}$$

Aside: Setting the Step Size in Gradient Descent

- What is a good value for step size α?

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

  - If α = too small, it may be too slow
  - If α = too large, it may oscillate



- It may take trial-and-errors to find the sweet spot.
- Another trick is to define a "schedule" for gradually reducing the learning rate starting from a large number.
  - More on this in the homework! ☺

[figure from: https://www.jeremyjordan.me/nn-learning-rate/]

Limitations of Word Embeddings: Lack of Compositionality

- Lexical Semantics: focuses on the meaning of individual words.
- Compositional Semantics: meaning depends on the words, and on how they are combined.

Limitations of Word Embeddings: Lack of Compositionality

- Lexical Semantics: focuses on the meaning of individual words.
- Compositional Semantics: meaning depends on the words, and on how they are combined.
  - From HW:

Now, consider sentiment analysis on a phrase in which the predicted sentiments are

$$f(s;\theta) = \theta \cdot \text{repr}(s),$$

for some choice of parameters θ. Prove that in such a model, the following inequality cannot hold for any choice of θ:

$$f(\texttt{good};\theta) > f(\texttt{not good};\theta)$$
$$f(\texttt{bad};\theta) < f(\texttt{not bad};\theta)$$

Limitations of Word Embeddings: No Word Senses

- Most words have lots of meanings!
  - Especially common words
  - Or those that existed for a long time
- Example: "bear"
  - A mammal (🐻)
  - Something difficult (*the oven is a bear to clean*)
  - To accept (*couldn't bear the pain*)
  - To have features (*bears a likeness of her mom*)
  - To contain (*old-bearing shale*)
  - To hold in mind or emotions (*bear malice*)
  - …
- Do word vectors capture word senses?

Today's Recap

- W2V training and broader family of word embeddings
- Building classifiers with word embeddings
- Next time: moving beyond words by modeling sequences.