Part-of-Speech Tagging

+

Neural Networks 3: Word Embeddings

CS 287

Review: Neural Networks

One-layer multi-layer perceptron architecture,

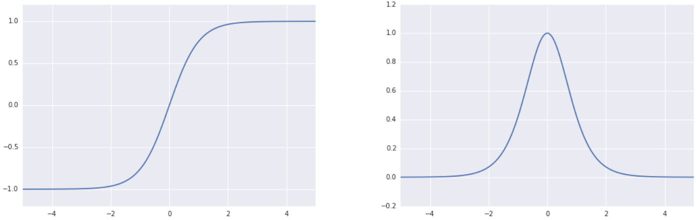
$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2$$

- $\mathbf{x}\mathbf{W} + \mathbf{b}$; *perceptron*
- $\mathbf{x}$ is the dense representation in $\mathbb{R}^{1 \times d_{\mathrm{in}}}$
- $\mathbf{W}^1 \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{hid}}}$, $\mathbf{b}^1 \in \mathbb{R}^{1 \times d_{\mathrm{hid}}}$; first affine transformation
- $\mathbf{W}^2 \in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{out}}}$, $\mathbf{b}^2 \in \mathbb{R}^{1 \times d_{\mathrm{out}}}$; second affine transformation
- $g : \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}}$ is an *activation non-linearity* (often pointwise)
- $g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$ is the *hidden layer*

Review: Non-Linearities Tanh

Hyperbolic Tangeant:

$$\tanh(t) = \frac{\exp(t) - \exp(-t)}{\exp(t) + \exp(-t)}$$



- Intuition: Similar to sigmoid, but range between 0 and -1.

Review: Backpropagation

$$
\begin{tikzcd}
f_i(\dots f_1(\mathbf{x}^0)) \arrow[dr] & & f_{i+1}(f_i(\dots f_1(\mathbf{x}^0))) \\
& \boxed{f_{i+1}(*;\boldsymbol{\theta}_{i+1})} \arrow[ur] \arrow[dl] & \\
\dfrac{\partial L}{\partial f_i(\dots f_1(\mathbf{x}^0))} & \dfrac{\partial L}{\partial \boldsymbol{\theta}_{i+1}} & \dfrac{\partial L}{\partial f_{i+1}(\dots f_1(\mathbf{x}^0))} \arrow[ul]
\end{tikzcd}
$$

Quiz

One common class of operations in neural network models is known as *pooling*. Informally a pooling layer consists of aggregation unit, typically unparameterized, that reduces the input to a smaller size.

Consider three pooling functions of the form $f : \mathbb{R}^n \mapsto \mathbb{R}$,

1. $f(\mathbf{x}) = \max_i x_i$
2. $f(\mathbf{x}) = \min_i x_i$
3. $f(\mathbf{x}) = \sum_i x_i / n$

What action do each of these functions have? What are their gradients? How would you implement backpropagation for these units?

Quiz

- **Max pooling**: $f(\mathbf{x}) = \max_i x_i$
  - Keeps only the most activated input
  - Fprop is simple; however must store $\arg\max$ ("switch")
  - Bprop gradient is zero except for switch, which gets gradoutput
- **Min pooling**: $f(\mathbf{x}) = \min_i x_i$
  - Keeps only the least activated input
  - Fprop is simple; however must store $\arg\min$ ("switch")
  - Bprop gradient is zero except for switch, which gets gradoutput
- **Avg pooling**: $f(\mathbf{x}) = \sum_i x_i / n$
  - Keeps the average activation input
  - Fprop is simply mean.
  - Gradoutput is averaged and passed to all inputs.

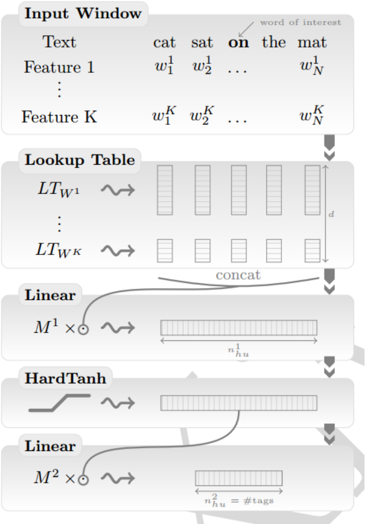
Quiz

- **Max pooling**: $f(\mathbf{x}) = \max_i x_i$
  - Keeps only the most activated input
  - Fprop is simple; however must store $\arg\max$ ("switch")
  - Bprop gradient is zero except for switch, which gets gradoutput
- **Min pooling**: $f(\mathbf{x}) = \min_i x_i$
  - Keeps only the least activated input
  - Fprop is simple; however must store $\arg\min$ ("switch")
  - Bprop gradient is zero except for switch, which gets gradoutput
- **Avg pooling**: $f(\mathbf{x}) = \sum_i x_i / n$
  - Keeps the average activation input
  - Fprop is simply mean.
  - Gradoutput is averaged and passed to all inputs.

Quiz

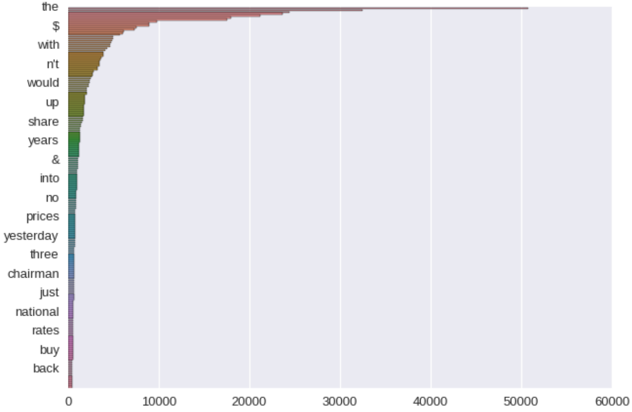
- **Max pooling**: $f(\mathbf{x}) = \max_i x_i$
  - Keeps only the most activated input
  - Fprop is simple; however must store $\arg\max$ ("switch")
  - Bprop gradient is zero except for switch, which gets gradoutput
- **Min pooling**: $f(\mathbf{x}) = \min_i x_i$
  - Keeps only the least activated input
  - Fprop is simple; however must store $\arg\min$ ("switch")
  - Bprop gradient is zero except for switch, which gets gradoutput
- **Avg pooling**: $f(\mathbf{x}) = \sum_i x_i / n$
  - Keeps the average activation input
  - Fprop is simply mean.
  - Gradoutput is averaged and passed to all inputs.

Contents



1. Use dense representations instead of sparse
2. Use windowed area instead of sequence models
3. Use neural networks to model windowed interactions

What about rare words?



Word Embeddings

Embedding layer,

$$\mathbf{x}^0 \mathbf{W}^0$$

- $\mathbf{x}^0 \in \mathbb{R}^{1 \times d_0}$ one-hot word.
- $\mathbf{W}^0 \in \mathbb{R}^{d_0 \times d_{\text{in}}}$, $d_0 = |\mathcal{V}|$

Notes:

- $d_0 >> d_{\text{in}}$, e.g. $d_0 = 10000$, $d_{\text{in}} = 50$

Pretraining Representations

- We would strong shared representations of words
- However, PTB only 1M labeled words, relatively small
- Collobert et al. (2008, 2011) use semi-supervised method.
- (Close connection to Bengio et al (2003), next topic)

Semi-Supervised Training

Idea: Train representations separately on more data

1. Pretrain word embeddings $\mathbf{W}^0$ first.
2. Substitute them in as first NN layer
3. Fine-tune embeddings for final task
   - Modify the first layer based on supervised gradients
   - Optional, some work skips this step

Large Corpora

To learn rare word embeddings, need many more tokens,

- C&W
  - English Wikipedia (631 million words tokens)
  - Reuters Corpus (221 million word tokens)
  - Total vocabulary size: 130,000 word types
- word2vec
  - Google News (6 billion word tokens)
  - Total vocabulary size: $\approx$ 1M word types

But this data has no labels...

Contents

C&W Embeddings

- Assumption: Text in Wikipedia is *coherent* (in some sense).
- Most randomly corrupted text is *incoherent*.
- Embeddings should distinguish coherence.
- Common idea in unsupervised learning (distributional hypothesis).

C&W Setup

Let $\mathcal{V}$ be the vocabulary of English and let s score any window of size $d_{\mathrm{win}} = 5$, if we see the phrase

[the dog walks to the]

It should score higher by s than

[the dog house to the]

[the dog cats to the]

[the dog skips to the]

...

C&W Setup

Can estimate score s as a windowed neural network.

$$s(w_1, \ldots, w_{d_{\mathrm{win}}}) = \mathrm{hardtanh}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}$$

with

$$\mathbf{x} = [v(w_1)\ v(w_2)\ \ldots\ v(w_{d_{\mathrm{win}}})]$$

- $d_{\mathrm{in}} = d_{\mathrm{win}} \times 50$, $d_{\mathrm{hid}} = 100$, $d_{\mathrm{win}} = 11$, $d_{\mathrm{out}} = 1$!

Example: Function s

$$\mathbf{x} = [v(w_3)\ v(w_4)\ v(w_5)\ v(w_6)\ v(w_7)]$$

		$\mathbf{x}$		
$d_{\mathrm{in}}/d_{\mathrm{win}}$	$d_{\mathrm{in}}/d_{\mathrm{win}}$	$d_{\mathrm{in}}/d_{\mathrm{win}}$	$d_{\mathrm{in}}/d_{\mathrm{win}}$	$d_{\mathrm{in}}/d_{\mathrm{win}}$

Training?

- Different setup than previous experiments.
- No direct supervision $\mathbf{y}$
- Train to rank good examples better.

Ranking Loss

Given only example $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ and for each example have set $\mathcal{D}(\mathbf{x})$ of alternatives.

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_i \sum_{\mathbf{x}' \in \mathcal{D}(\mathbf{x})} L_{ranking}(s(\mathbf{x}_i; \boldsymbol{\theta}), s(\mathbf{x}'; \boldsymbol{\theta}))$$

$$L_{ranking}(y, \hat{y}) = \max\{0, 1 - (y - \hat{y})\}$$

Example: C&W ranking

$$\mathbf{x} = [\text{the dog walks to the}]$$

$$\mathcal{D}(\mathbf{x}) = \{\ [\text{the dog } \textcolor{red}{\text{skips}} \text{ to the}], [\text{the dog } \textcolor{red}{\text{in}} \text{ to the}], \dots \}$$

- (Torch `nn.RankingCriterion`)
- Note: slightly different setup.

C&W Embeddings in Practice

- Vocabulary size $|\mathcal{D}(\mathbf{x})| > 100,000$
- Training time for 4 weeks
- (Collobert is main an author of Torch)

Sampling (Sketch of WSABIE (Weston, 2011))

Observation: in many contexts

$$L_{ranking}(y, \hat{y}) = \max\{0, 1 - (y - \hat{y})\} = 0$$

Particularly true later in training.

For difficult contexts, may be easy to find

$$L_{ranking}(y, \hat{y}) = \max\{0, 1 - (y - \hat{y})\} \neq 0$$

We can therefore sample from $\mathcal{D}(\mathbf{x})$ to find an update.

C&W Results

Approach	**POS** (PWA)	**CHUNK** (F1)	**NER** (F1)	**SRL** (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

1. Use dense representations instead of sparse
2. Use windowed area instead of sequence models
3. Use neural networks to model windowed interactions
4. Use semi-supervised learning to pretrain representations.

Contents

word2vec

- Contributions:
  - Scale embedding process to massive sizes
  - Experiments with several architectures
  - Empirical evaluations of embeddings
  - Influential release of software/data.
- Differences with C&W
  - Instead of MLP uses (bi)linear model (linear in paper)
  - Instead of ranking model, directly predict word (cross-entropy)
  - Various other extensions.
- Two different models
  1. Continuous Bag-of-Words (CBOW)
  2. Continuous Skip-gram

word2vec

- Contributions:
  - Scale embedding process to massive sizes
  - Experiments with several architectures
  - Empirical evaluations of embeddings
  - Influential release of software/data.
- Differences with C&W
  - Instead of MLP uses (bi)linear model (linear in paper)
  - Instead of ranking model, directly predict word (cross-entropy)
  - Various other extensions.
- Two different models
  1. Continuous Bag-of-Words (CBOW)
  2. Continuous Skip-gram

word2vec (Bilinear Model)

Back to pure bilinear model, but with much bigger output space

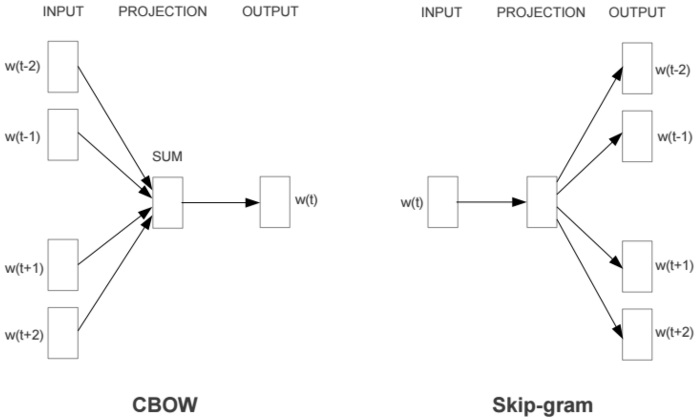
$$\hat{\mathbf{y}} = \text{softmax}\left(\left(\frac{\sum_i \mathbf{x}_i^0 \mathbf{W}^0}{d_{\mathrm{win}} - 1}\right)\mathbf{W}^1\right)$$

- $\mathbf{x}_i^0 \in \mathbb{R}^{1 \times d_0}$ input words one-hot vectors .
- $\mathbf{W}^0 \in \mathbb{R}^{d_0 \times d_{\mathrm{in}}}$; $d_0 = |\mathcal{V}|$, word embeddings
- $\mathbf{W}^1 \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}$; $d_{out} = |\mathcal{V}|$ output embeddings

Notes:

- Bilinear parameter interaction.
- $d_0 >> d_{\mathrm{in}}$, e.g. $50 \leq d_{\mathrm{in}} \leq 1000$, $10000 \leq |\mathcal{V}| \leq 1M$ or more

word2vec (Mikolov, 2013)



Continuous Bag-of-Words (CBOW)

$$\hat{\mathbf{y}} = \mathrm{softmax}\left(\left(\frac{\sum_i \mathbf{x}_i^0 \mathbf{W}^0}{d_{\mathrm{win}} - 1}\right)\mathbf{W}^1\right)$$

- Attempt to predict the middle word

[the dog walks to the]

Example: CBOW

$$\mathbf{x} = \frac{v(w_3) + v(w_4) + v(w_6) + v(w_7)}{d_{\mathrm{win}} - 1}$$

$$\mathbf{y} = \delta(w_5)$$

$\mathbf{W}^1$ is no longer partitioned by row (order is lost)

Continous Skip-gram

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{x}^0 \mathbf{W}^0)\mathbf{W}^1)$$

- Also a bilinear model
- Attempt to predict each context-word from middle

[the ___ dog ___ ___]

Example: Skip-gram

$$\mathbf{x} = v(w_5)$$

$$\mathbf{y} = \delta(w_3)$$

Done for each word in window.

Additional aspects

- The window d_{win} is sampled for each SGD step
- SGD is done less for frequent words.
- We have slightly simplified the training objective.

Softmax Issues

Use a softmax to force a distribution,

$$\text{softmax}(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\sum_{c \in \mathcal{C}} \exp(z_c)}$$

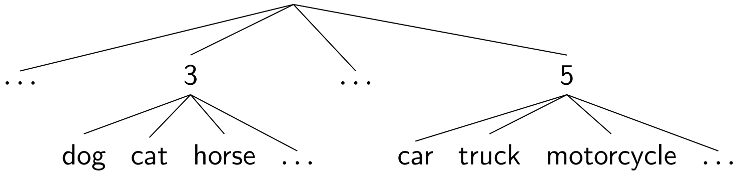
$$\log \text{softmax}(\mathbf{z}) = \mathbf{z} - \log \sum_{c \in \mathcal{C}} \exp(z_c)$$

- **Issue:** class $\mathcal{C}$ is huge.
- For C&W, 100,000, for word2vec 1,000,000 types
- Note largest dataset is 6 billion words

Two-Layer Softmax

First, clustering words into hard classes (for instance Brown clusters)

Groups words into classes based on word-context.



Two-Layer Softmax

Assume that we first generate a class C and then a word,

$$p(Y|X) \approx P(Y|C, X; \theta)P(C|X; \theta)$$

Estimate distributions with a shared embedding layer,

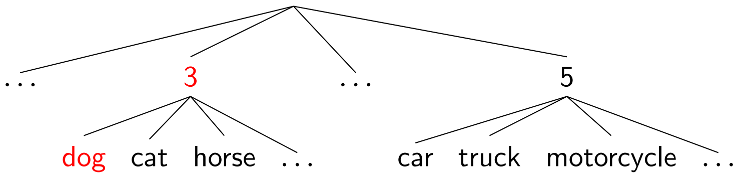
$P(C|X; \theta)$

$$\hat{\mathbf{y}}_1 = \mathrm{softmax}((\mathbf{x}^0\mathbf{W}^0)\mathbf{W}^1 + \mathbf{b})$$

$P(Y|C = class, X; \theta)$

$$\hat{\mathbf{y}}_2 = \mathrm{softmax}((\mathbf{x}^0\mathbf{W}^0)\mathbf{W}^{class} + \mathbf{b}))$$

Softmax as Tree

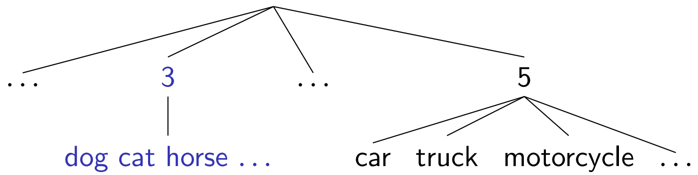


$$\hat{\mathbf{y}}^{(1)} = \text{softmax}((\mathbf{x}^0\mathbf{W}^0)\mathbf{W}^1 + \mathbf{b})$$

$$\hat{\mathbf{y}}^{(2)} = \text{softmax}((\mathbf{x}^0\mathbf{W}^0)\mathbf{W}^{class} + \mathbf{b}))$$

$$\begin{aligned} L_{2SM}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}) &= -\log p(\mathbf{y}|\mathbf{x}, class(\mathbf{y})) - \log p(class(\mathbf{y})|\mathbf{x}) \\ &= -\log \hat{y}^{(1)}_{c^1} - \log \hat{y}^{(2)}_{c^2} \end{aligned}$$

Speed



- Computing loss only requires walking path.
- Two-layer a balanced tree.
- Computing loss requires $O(\sqrt{|\mathcal{V}|})$
- (Note: computing full distribution requires $O(|\mathcal{V}|)$)

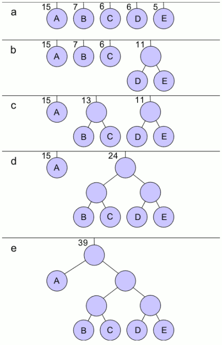
Hierarchical Softmax(HSM)

- Build multiple layer tree

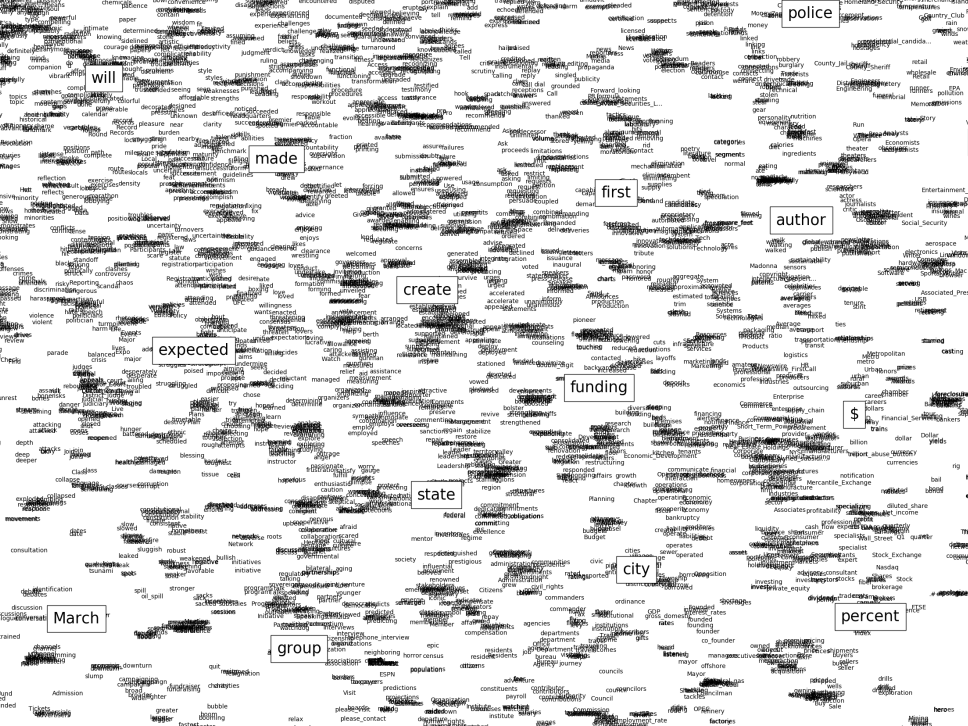
$$L_{HSM}(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(C)}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(C)}) = -\sum_i \log \hat{y}^{(i)}_{c^i}$$

- Balanced tree only requires $O(\log_2 |\mathcal{V}|)$
- Experiments on website (Mnih and Hinton, 2008)

HSM with Huffman Encoding



- Requires $O(\log_2 perp(unigram))$
- Reduces time to only 1 day for 1.6 million tokens


police
will
made
first
author
create
expected
funding
$
state
city
March
percent
group

Contents

How good are embeddings?

- Qualitative Analysis/Visualization
- Analogy task
-
- Extrinsic Metrics

Metrics

Dot-product

$$\mathbf{x}_{cat}\mathbf{x}_{dog}^\top$$

Cosine Similarity

$$\frac{\mathbf{x}_{cat}\mathbf{x}_{dog}^\top}{\left|\left|\mathbf{x}_{cat}\right|\right| \ \left|\left|\mathbf{x}_{dog}\right|\right|}$$

k-nearest neighbors (cosine sim)

dog	cat	0.921800527377
	dogs	0.851315870426
	horse	0.790758298322
	puppy	0.775492121034
	pet	0.772470734611
	rabbit	0.772081457265
	pig	0.749006160038
	snake	0.73991884888

- Intuition: trained to match words that act the same.

Empirical Measures: Analogy task

Analogy questions:

`A:B::C:__`

- 5 types of semantic questions, 9 types of syntactic

Embedding Tasks

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

Analogy Prediction

$$\texttt{A:B::C:\_\_}$$

$$\mathbf{x}' = \mathbf{x}_B - \mathbf{x}_A + \mathbf{x}_C$$

Project to the closest word,

$$\arg\max_{D \in \mathcal{V}} \frac{\mathbf{x}_D \mathbf{x}'^{\top}}{||\mathbf{x}_D|| ||\mathbf{x}'||}$$

- Code example

Extrinsic Tasks

- Text classification
- Part-of-speech tagging
- Many, many others over last couple years

Conclusion

- Word Embeddings
- Scaling issues and tricks
- Next Class: Language Modeling