Admin

Assignment 5b due Monday 3/27

Schedule for the rest of the semester mostly up to date

A Single Neuron/Perceptron

Activation functions

- **hard threshold**:
  \[ g(in) = \begin{cases} 
  1 & \text{if } in \geq T \\
  0 & \text{otherwise} 
  \end{cases} \]

- **sigmoid**:
  \[ g(x) = \frac{1}{1 + e^{-x}} \]

- **tanh x**
Many other activation functions

Rectified Linear Unit

Softmax (for probabilities)

Neural network

Neural network

Neural network

Neural network
Neural network inputs those answers become inputs for the next level finally get the answer after all levels compute.

T = ?
Output y
Input x
Input x
W_1 = ?
W_2 = ?

XOR

Input x_1
Input x_2

T = ?
Output y

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_1 XOR x_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Learning in multilayer networks

Challenge: for multilayer networks, we don’t know what the expected output/error is for the internal nodes!

how do we learn these weights?

expected output?

perceptron/ linear model
neural network
Backpropagation: intuition

Gradient descent method for learning weights by optimizing a loss function

1. calculate output of all nodes
2. calculate the weights for the output layer based on the error
3. "backpropagate" errors through hidden layers

We can calculate the actual error here

Key idea: propagate the error back to this layer

Trained a NN with 1B connections on 10M snapshots from youtube on 16,000 processors
Deep learning

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations.

Deep learning is part of a broader family of machine learning methods based on learning representations of data.

---

Deep learning

Key: learning better features that abstract from the “raw” data

- Using learned feature representations based on large amounts of data, generally unsupervised
- Using classifiers with multiple layers of learning

---

Deep learning

- Train multiple layers of features/abstractions from data.
- Try to discover representation that makes decisions easy.

Deep Learning: train layers of features so that classifier works well.

Slide adapted from: Adam Coates
Deep learning for neural networks

- Traditional NN models: 1-2 hidden layers
- Deep learning NN models: 3+ hidden layers

Challenges

What makes “deep learning” hard for NNs?

- Modified errors tend to get diluted as they get combined with many layers of weight corrections

Deep learning

Growing field

- Increase in data availability
- Increase in computational power
- Parallelizability of many of the algorithms

Involves more than just neural networks (though, they’re a very popular model)
**word2vec**

How many people have heard of it?

What is it?

---

**Word representations generalized**

Project words into a multi-dimensional “meaning” space

word $\rightarrow [x_1, x_2, \ldots, x_d]$

What was our projection for assignment 5?

---

**Word representations**

Project words into a multi-dimensional “meaning” space

word $\rightarrow [x_1, x_2, \ldots, x_d]$

red $\rightarrow [r_1, r_2, \ldots, r_d]$

crimson $\rightarrow [c_1, c_2, \ldots, c_d]$

yellow $\rightarrow [y_1, y_2, \ldots, y_d]$
Word representations

Project words into a multi-dimensional “meaning” space

\[ \text{word} \rightarrow [x_1, x_2, \ldots, x_d] \]

The idea of word representations is not new:
• Co-occurrence matrices
• Latent Semantic Analysis (LSA)

New idea: learn word representation using a task-driven approach

A prediction problem

I like to eat bananas with cream cheese

Given a context of words
Predict what words are likely to occur in that context

A prediction problem

Given text, can generate lots of examples:

I like to eat bananas with cream cheese

<table>
<thead>
<tr>
<th>input</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>___ like to eat</td>
<td>I</td>
</tr>
<tr>
<td>I ___ to eat bananas</td>
<td>like</td>
</tr>
<tr>
<td>I like ___ eat bananas</td>
<td>to</td>
</tr>
<tr>
<td>I like ___ bananas with</td>
<td></td>
</tr>
<tr>
<td>I like to ___ bananas with cream</td>
<td>eat</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A prediction problem

Use data like this to learn a distribution:

\[
p(\text{word} | \text{context})
\]

\[
p(w_i | w_{i-2} w_{i-1} w_{i+1} w_{i+2})
\]

words before   words after
Train a neural network on this problem

Encoding words

How can we input a “word” into a network?

“One-hot” encoding

For a vocabulary of $V$ words, have $V$ input nodes

All inputs are 0 except the for the one corresponding to the word

“One-hot” encoding

For a vocabulary of $V$ words, have $V$ input nodes

All inputs are 0 except the for the one corresponding to the word
“One-hot” encoding

For a vocabulary of \( V \) words, have \( V \) input nodes.

All inputs are 0 except the one corresponding to the word.

```
    0  1  ...  0
apple  a  banana  ...  zebra
    0  1  ...  0
```

\( N = 100 \) to \( 1000 \)

Another view

V input nodes \( \rightarrow \) V output nodes \( \rightarrow \) N hidden nodes

Training: backpropagation

```
I like to eat
I like to eat bananas
I like to eat bananas with
I like to eat bananas with cream
...```

---

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/
Training: backpropagation

Word representation

The weights for each word provide an N dimensional mapping of the word.

Words that predict similarly should have similar weights.

Why does this work?

Results

\[ \text{vector(word1)} - \text{vector(word2)} = \text{vector(word3)} - X \]

word1 is to word2 as word3 is to X

<table>
<thead>
<tr>
<th>Type of relationship</th>
<th>Word Pair 1</th>
<th>Word Pair 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common capital city</td>
<td>Athens</td>
<td>Oslo</td>
</tr>
<tr>
<td>All capital cities</td>
<td>Astana</td>
<td>Helsinki</td>
</tr>
<tr>
<td>Currency</td>
<td>Angola</td>
<td>Iran</td>
</tr>
<tr>
<td>City-in-state</td>
<td>Chicago</td>
<td>Stockton</td>
</tr>
<tr>
<td>Man-Woman</td>
<td>brother</td>
<td>granddaughter</td>
</tr>
</tbody>
</table>

Results

\[ \text{vector(word1)} - \text{vector(word2)} = \text{vector(word3)} - X \]

word1 is to word2 as word3 is to X

<table>
<thead>
<tr>
<th>Type of relationship</th>
<th>Word Pair 1</th>
<th>Word Pair 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjective to adverb</td>
<td>apparently</td>
<td>rapidly</td>
</tr>
<tr>
<td>Opposite</td>
<td>possibly</td>
<td>unethical</td>
</tr>
<tr>
<td>Comparative</td>
<td>great</td>
<td>tough</td>
</tr>
<tr>
<td>Superlative</td>
<td>easy</td>
<td>lucky</td>
</tr>
<tr>
<td>Present Participle</td>
<td>think</td>
<td>hardest</td>
</tr>
<tr>
<td>Nationality adjective</td>
<td>Switzerland</td>
<td>reading</td>
</tr>
<tr>
<td>Past tense</td>
<td>walking</td>
<td>Cambodian</td>
</tr>
<tr>
<td>Plural nouns</td>
<td>mouse</td>
<td>swimming</td>
</tr>
<tr>
<td>Plural verbs</td>
<td>work</td>
<td>swim</td>
</tr>
<tr>
<td></td>
<td>works</td>
<td>speaks</td>
</tr>
</tbody>
</table>
vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

<table>
<thead>
<tr>
<th>Newspapers</th>
<th>STHL Teams</th>
<th>NBA Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York Times</td>
<td>Boston Bruins</td>
<td>Detroit Pistons</td>
</tr>
<tr>
<td>San Jose Mercury News</td>
<td>Phoenix Coyotes</td>
<td>Golden State Warriors</td>
</tr>
<tr>
<td>Baltimore Sun</td>
<td>Montreal Canadiens</td>
<td>Toronto Raptors</td>
</tr>
<tr>
<td>Baltimore Sun</td>
<td>Nashville Predators</td>
<td>Memphis Grizzlies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Airlines</th>
<th>Airlines</th>
<th>Airlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austrian Airlines</td>
<td>Brussels Airlines</td>
<td>Aegean Airlines</td>
</tr>
<tr>
<td>Belgium</td>
<td>Spain</td>
<td>Spain</td>
</tr>
<tr>
<td></td>
<td>Greece</td>
<td>Greece</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company executives</th>
<th>Company executives</th>
<th>Company executives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steve Ballmer</td>
<td>Larry Page</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Samuel J. Palmisano</td>
<td>Werner Vogels</td>
<td>Google</td>
</tr>
</tbody>
</table>

Country and Capital Vectors Projected by PCA

Continuous Bag Of Words
Other models: skip-gram

word2vec

A model for learning word representations from large amounts of data

Has become a popular pre-processing step for learning a more robust feature representation

Models like word2vec have also been incorporated into other learning approaches (e.g. translation tasks)

word2vec resources

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

https://code.google.com/archive/p/word2vec/

https://deeplearning4j.org/word2vec


Playing with word2vec

http://vectors.nlpl.eu/explore/embeddings/en/
10 minutes

https://www.youtube.com/watch?v=zl99lZvW7rE

Quiz 2

Average: 22.8 (89%)
Median: 23 (90%)