Introductions

• Professor: Doug Downey
• (part-time) Teaching Assistant: Nishant Subramani
• Course web site:
  – www.cs.northwestern.edu/~ddowney/courses/395_Winter2017
  – (linked off prof. home page)
Logistics

Grading

Participation (50%)
- Reading papers – two-paragraph summaries (10%)
- Attending and participating in discussions (10%)
  - Self-reported
- Presenting a paper (30%)
  - Teams of k

Self-directed Project (50%)
- Proposal (5%)
- Status update (5%)
- Final presentation (20%) and report (20%)
Outline for Today

- Statistical Language Models (SLMs): What are they?
- Brief Neural Net (NN) primer
- NN SLMs
- Class structure
  - Class participation
  - Projects
Modeling Sequences of Words

- Statistical language models assign **probabilities** to **sequences of words**

\[ P(“\text{the dog barked”}) = 4.203 \times 10^{-9} \]

- **Applications**
  - Speech Recognition
  - Machine Translation
  - Spelling Correction
  - Information Extraction
How to measure LM performance?

- Test model $M$ on held-out text $\bar{w}$ of length $N$

$$\text{Perplexity}(M, \bar{w}) = 2^{-\log_2 P_M(\bar{w})}$$
Models trained on 38 million words from the Wall Street Journal (WSJ) using a 19,979 word vocabulary. Evaluate on a disjoint set of 1.5 million WSJ words.

Perplexity: Trigram < Bigram < Unigram (in this case)
Which N-gram is the best in general?

<table>
<thead>
<tr>
<th>Perplexity</th>
<th>Unigram</th>
<th>Bigram</th>
<th>Trigram</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>962</td>
<td>170</td>
<td>109</td>
</tr>
</tbody>
</table>

**Unigram**

Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives

**Bigram**

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

**Trigram**

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions
Outline for Today

- Statistical Language Models (SLMs): What are they?
- Brief Neural Net (NN) primer
- NN SLMs
- Class structure
  - Class participation
  - Projects
Perceptrons

Problem def:

Let \( f \) be a target function from \( X = \langle x_1, x_2, \ldots \rangle \) where \( x_i \in \{0, 1\} \) to \( y \in \{0, 1\} \)

Given training data \( \{(X_1, y_1), (X_2, y_2)\ldots\} \)

Learn \( h(X) \), an approximation of \( f(X) \)
The sigmoid (logistic) unit

- Has differentiable function
  - Allows gradient descent
- Can be used to learn non-linear functions

\[ \sigma = \frac{1}{1 + e^{-\sum_{i=0}^{n} w_i x_i}} \]
Logistic function

Inputs

- Age 34
- Gender 1
- Stage 4

Coefficients

- 0.5
- 0.4
- 0.8

Output

“Probability of being Alive” 0.6

Prediction

\[
\sigma = \frac{1}{1 + e^{-\sum_{i=0}^{n} w_i x_i}}
\]
Neural Network Model

**Inputs**
- Age: 34
- Gender: 2
- Stage: 4

**Weights**
- From Age to Hidden Layer: 0.6
- From Gender to Hidden Layer: 0.1
- From Stage to Hidden Layer: 0.3

**Hidden Layer**
- Hidden Layer 1: 0.7
- Hidden Layer 2: 0.2

**Output**
- 0.6

"Probability of being Alive"

**Independent variables**
- Age
- Gender
- Stage

**Dependent variable**
- Prediction
Getting an answer from a NN

Inputs

- Age: 34
- Gender: 2
- Stage: 4

Weights

- Hidden Layer: 0.6, 0.1, 0.7
- Output: 0.5, 0.8

Output

- "Probability of being Alive": 0.6

Independent variables

Dependent variable

Prediction
Getting an answer from a NN

Inputs

Age
34

Gender
2

Stage
4

Weights

0.2

0.3

0.2

Output

“Probability of being Alive”

0.6

Independent variables

Weights

Hidden Layer

Dependent variable

Prediction
Getting an answer from a NN

Independent variables

Weights

Hidden Layer

Weights

Dependent variable

Prediction

Inputs

Age

Gender

Stage

Output

“Probability of being Alive”

0.6

Prediction

Weighted sum

Age: 34

Gender: 1

Stage: 4

Output: 0.6

Weights:

- Age: 0.6
- Gender: 0.1
- Stage: 0.3

Hidden Layer:

- Summation of weighted inputs

Weights:

- Hidden Layer: 0.8
- Output: 0.5
Gradient Descent

Gradient:

\[ \nabla E[\vec{w}] \equiv \begin{bmatrix} \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n} \end{bmatrix} \]

Training rule:

\[ \Delta \vec{w} = -\eta \nabla E[\vec{w}] \]

\[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \]
Minimizing the Error

Error surface

initial error

negative derivative

final error

local minimum

$w_{\text{initial}}$ $w_{\text{trained}}$

positive change
Backpropagation

- Sigmoid is easy to differentiate

\[
\frac{\partial \sigma(y)}{\partial y} = \sigma(y) \cdot (1 - \sigma(y))
\]

- For gradient descent on multiple layers, a little dynamic programming can help:
  - Compute errors at each output node
  - Use these to compute errors at each hidden node
  - Use these to compute weight gradient
The Backpropagation Algorithm

For each input training example, \( \langle \vec{x}, \vec{t} \rangle \)

1. Input instance \( \vec{x} \) to the network and compute the output \( o_u \) for every unit \( u \) in the network

2. For each output unit \( k \), calculate its error term \( \delta_k \)
   \[
   \delta_k \leftarrow o_k (1 - o_k)(t_k - o_k)
   \]

3. For each hidden unit \( h \), calculate its error term \( \delta_h \)
   \[
   \delta_h \leftarrow o_h (1 - o_h) \sum_{k \in \text{outputs}} w_{hk} \delta_k
   \]

4. Update each network weight \( w_{ji} \)
   \[
   w_{ji} \leftarrow w_{ji} + \eta \delta_k x_{ji}
   \]
Learning Weights

**Inputs**

- **Age**: 34
- **Gender**: 1
- **Stage**: 4

**Weights**

- Age to hidden layer: 0.6
- Gender to hidden layer: 0.1
- Stage to hidden layer: 0.2

**Hidden Layer**

- Weight sum: 0.5
- Weight sum: 0.8

**Output**

- "Probability of being Alive": 0.6

**Dependent variable**

- Prediction
The fine print

- Don’t implement back-propagation
  - Use a package
  - Second-order or variable step-size optimization techniques exist

- Feature normalization
  - Typical to normalize inputs to lie in $[0, 1]$  
    - (and outputs must be normalized)

- Problems with NN training:
  - Slow training times (though, getting better)
  - Local minima
Minimizing the Error

Error surface

initial error

negative derivative

final error

local minimum

\( w_{\text{initial}} \) \( w_{\text{trained}} \)

positive change
Expressive Power of NNs

- Universal Function Approximator:
  - Given enough hidden units, can approximate any continuous function $f$

- Need 2+ hidden units to learn XOR

- Why not use millions of hidden units?
  - Efficiency (training is slow)
  - Overfitting
Overfitting

Real Distribution

Overfitted Model
Combating Overfitting in Neural Nets

- Many techniques

- In language modeling, “early stopping” is the go-to technique
  - Use “a lot” of hidden units
  - Just don’t over-train
Early Stopping

Overfitted model

\[ a = \text{validation set} \]

\[ b = \text{training set} \]

Stopping criterion

\[ \min (\Delta \text{error}) \]

\[ \text{error}_a \]

\[ \text{error}_b \]

\[ \text{Epochs} \]
Learning Rate?

- A “knob” you twist empirically
  - Important

- One popular option: look for validation set acc to decrease/stabilize, then halve learning rate
Neural Networks Today

- Democratizing software packages exist
  - Tensorflow
    - Keras
  - Theano, Caffe, Torch...

- Use them!
Outline for Today

- Statistical Language Models (SLMs): What are they?
- Brief Neural Net (NN) primer
- **NN SLMs**
- Class structure
  - Class participation
  - Projects
Concatenation

$x(t) = w(t) + s(t-1)$

1 of $|V|$ encoding

Weights

$y_k(t) = g \left( \sum_j s_j(t) v_{kj} \right)$

CONTEXT (t)

$w(t) = f \left( \sum_i x_i(t) u_{ji} \right)$

CONCATENATION

CONTEXT (t-1)

$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$

$f(z) = \frac{1}{1 + e^{-z}}$
Word Embeddings

- The neural networks represent each word internally as a vector of weights
  - These have a surprising ability to capture semantics

- The *distributional hypothesis*:
  - A word is known by the company it keeps
  - Words with similar meanings tend to appear in similar contexts [Harris, 1954]

- Words with similar vectors tend to have similar meanings
RNNs

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
LSTMs
Intuition behind LSTMs

- Harness long-range dependencies
- “She’s _________ so I hope that she will _________ .”
  - brilliant
  - sick
  - join us
  - not breathe on me
LSTM example

Noraset et al., AAAI 2017
Incredibly good language modeling results

- Good Turing Smoothing: \( \sim 160 \) perplexity (lower is better)
- Kneser-Ney smoothing: \( \sim 140 \) perplexity
- Today’s best LSTMs: \( \sim 70 \) perplexity [Google, ca 2015]

Perplexity numbers on Penn TreeBank data set (approx.)
Outline for Today

- Statistical Language Models (SLMs): What are they?
- Brief Neural Net (NN) primer
- NN SLMs
- Class structure
  - Class participation
  - Projects
Class Participation

- Summarizing papers (10% of grade, in Canvas)
- Class participation (10% of grade, in Canvas)
  - After each class, report what you said. Number of comments, content of each.
- Presenting papers (30% of grade)
  - You will lead discussions of papers in teams of ~2
  - Crucial to solicit participation
    - Think about ways to structure discussion so that people can get involved, even if class is large
Projects

- Generate text, e.g.:
  http://karpathy.github.io/2015/05/21/rnn-effectiveness/
  https://arxiv.org/abs/1612.00394
- what other tasks could you do this with?
- Characteristics to look for: utility, tractability, degree to which you can exploit problem structure
- Classify text, e.g. sentiment classification, relation extraction
- Other stuff (you have ideas, I can also give more suggestions)