Neural Network
Language Modeling

Yogarshi Vyas
02/24/16
LING773
Topics for Today

• Language Modeling recap
• Basics of Neural Networks
• Feed forward Neural Network Language Model
• Recurrent Neural Network Language Model
• Word Embeddings
• Word2Vec – Context Bag-of-Words and Skip-gram
Language Modeling Recap

• Assign a probability to a sentence
  – $P(\text{I went to the bank}) \gg P(\text{I goes to the bank})$

• Can be calculated using Chain Rule

• $P(\text{I went to the bank}) = P(\text{bank} | \text{I went to the}) *$
  $P(\text{the} | \text{I went to}) *$
  $P(\text{to} | \text{I went}) *$
  $P(\text{went} | \text{I}) *$
  $P(I)$
LM – Estimating Probabilities

• Count and divide

  – \( P(\text{bank} \mid \text{I went to the}) \)
    \[ = \frac{\text{Count (I went to the bank)}}{\text{Count (I went to the)}} \]

• But these estimates will be poor!
  – So we make Markov assumptions

• \( P(\text{bank} \mid \text{I went to the}) \approx P(\text{bank} \mid \text{to the}) \)
  or

• \( P(\text{bank} \mid \text{I went to the}) \approx P(\text{bank} \mid \text{to}) \)
Language Modeling Recap

• Assign a probability to a sentence
  – \( P(\text{I went to the bank}) \gg P(\text{Eye went to the bank}) \)

• Can be calculated using Chain Rule

\[
P(\text{I went to the bank}) = P(\text{bank} \mid \text{I went to the}) \times P(\text{the} \mid \text{I went to}) \times P(\text{to} \mid \text{I went}) \times P(\text{went} \mid \text{I}) \times P(\text{I})
\]
Language Modeling Recap

• Assign a probability to a sentence
  – P (I went to the bank) >> P (Eye vent to the bank)

• Can be calculated using Chain Rule

• \[ P( \text{I went to the bank} ) = P(\text{bank} | \text{I went to the}) \times P(\text{the} | \text{I went to}) \times P(\text{to} | \text{I went}) \times P(\text{went} | \text{I}) \times P(\text{I}) \]
Smoothing

• Zeros are bad!

• Give artificial counts to unseen words, by taking away counts from seen words
  – Laplace
  – Good Turing
  – Kneser Ney
Issues with n-gram LMs

• State of the art applications of LMs (say in MT), use 4/5-grams.
  – Parameters grow exponentially with n.

• Similarity between words is not taken into account.
  – P(“The cat is walking in the bedroom”) should give some indication about P(“A dog was running in a room”)
Neural Net Recap

- Linear Classifier/Perceptron
- Sigmoid/Softmax
- Multilayer Perceptron / Feed forward NN
- Recurrent NN
Linear classifiers

\[ f(x) = \begin{cases} 
1 & \text{if } w \cdot x + b > 0 \\
0 & \text{otherwise}
\end{cases} \]
Linear Classifiers

\[ f(x) = \begin{cases} 
1 & \text{if } w \cdot x + b > 0 \\ 
0 & \text{otherwise} 
\end{cases} \]
Non linearity – Heavyside Step Function

\[ f(x) = \begin{cases} 
0 & \text{if } x < 0 \\
1 & \text{if } x > 0 
\end{cases} \]

Output is either 0 or 1
Non linearity – Logistic Function

\[ f(x) = \frac{1}{1 + e^{-x}} \]

Output is a real value between 0 and 1
Softmax

- Multivariate version of Logistic Function
- Takes an arbitrary vector and squashes all values between 0 and 1 such that they sum to 1
- From a K-dimensional vector, you can get a K-class classifier
A neuron
Other non-linearities

- Tanh
- ReLU (Rectified Linear Unit)
Stacking up neurons…

x1

x2

x3
Stacking up neurons...
Stacking up neurons..

- $x_1$
- $x_2$
- $x_3$
Feed-forward Neural Network!

x1

x2

x3
Feed-forward Neural Network!
Feed-forward Neural Network!
Feed-forward Neural Network!

x1

x2

x3

Ouptut Layer
Training a Neural Net

• What are the parameters of a neural net?
  – Architecture (Number of layers, size, etc.)
  – Activation function
  – Input features
  – Weight

• Train weights using Backpropagation
  – Gradient descent + Chain Rule
  – Loss functions – Cross entropy, Log likelihood
Adding layers – making it deep
Matrix multiplication view

- Input
  3d vector $x$

- Hidden layer
  $y = f_i(W_1 x)$
  $W_1$ is 3x4

- Output layer
  $z = W_2 y$
  $W_2$ is 4x3
Recurrent NN
A simple NN bigram LM

Input layer - 1-of-V Vector
Hidden layer - Real valued vector in $\mathbb{R}^m$, $m << V$
Output layer - Probability distribution over $V$ words
A hint of Word Embeddings..

• Matrix between input and hidden layer
  – Vxm matrix
  – Vector representations for each word
Feedforward NN Language Model

- Approach summary –
  - Represent each word in the vocabulary as a feature vector (a real-valued vector in $\mathbb{R}^m$)
  - Express the joint probability function of a sequence of words in terms of these vectors
  - Learn probability function and vectors simultaneously

$i$-th output = $P(w_i = i \mid \text{context})$

Table look-up in $C$

Matrix $C$

shared parameters across words
Recurrent NN Language Model

• Recurrent NN
• Bengio et al still use fixed length context. By plugging in a recurrent NN, you can theoretically have infinite context!

Extend RNNLM

• Add more features

• What are some features you can add?
Word Embeddings

- Learn real valued vector space representations of words from large corpora

- Word vectors capture many linguistic properties
  - Syntactic - gender, tense, plurality
  - Semantic - “capital city of”

- We can do nearest neighbor search around result of vector operation “King - man + woman” and obtain “Queen”

*(Linguistic regularities in continuous space word representations (Mikolov et al, 2013))*
<table>
<thead>
<tr>
<th>Expression</th>
<th>Nearest token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris - France + Italy</td>
<td>Rome</td>
</tr>
<tr>
<td>bigger - big + cold</td>
<td>colder</td>
</tr>
<tr>
<td>sushi - Japan + Germany</td>
<td>bratwurst</td>
</tr>
<tr>
<td>Cu - copper + gold</td>
<td>Au</td>
</tr>
<tr>
<td>Windows - Microsoft + Google</td>
<td>Android</td>
</tr>
<tr>
<td>Montreal Canadiens - Montreal + Toronto</td>
<td>Toronto Maple Leafs</td>
</tr>
</tbody>
</table>
CBoW

- Language modelling – use previous $n$ words to predict the next word
- CBoW – Use previous and next words to predict current word
- Same as Feedforward NN, but no hidden layer, thus less expensive to compute

SkipGram

- CBoW – Use context to predict the word
- SkipGram – Use word to predict the context
Tricks of the Trade

• Hierarchical softmax
  – The output softmax layer is huge (size of vocabulary)
  – Do it hierarchically, by predicting only word classes, then predicting subclasses, then subsubclasses... And so on until you get to leaves.

• Negative Sampling
  – Pick random word context pairs as negative examples
Whistles and Bells - Compositionality

• How do you get representations for phrases/sentences?
  – Average!

• Skip-thought
  – Basically skip-gram over sentences

Whistles and Bells – Multilingual Representations

• Independently learn vectors for English and French, learn a mapping (usually linear) between the two spaces

• Train an objective jointly over two languages

• In both cases, parallel data is used as crosslingual signal
Shortcomings

• Very little to no theoretical grounding
  – People have offered explanations for SGNS that draw parallels to LSA (1)

• Polysemy
  – One vector for the word ‘bank’
    • Does it capture properties of the financial institution? Or the river bank? Or both? How do you disentangle? (2)

• Interpretability
  – 100 dimensional vector representations for each word – but what does each dimension capture?
  – By adding sparsity and non-negativity, you can get interpretability + some cognitive plausibility (3)

(1) - Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." NIPS 2014.
(2) - Huang, Eric H., et al. "Improving word representations via global context and multiple word prototypes." ACL 2012.