Deep Learning for NLP

Instructor: Wei Xu
Ohio State University
CSE 5525

Many slides from Greg Durrett
Outline

‣ Motivation for neural networks
‣ Feedforward neural networks
‣ Applying feedforward neural networks to NLP
‣ Convolutional neural networks
‣ Application examples
‣ Tools
Sentiment Analysis

the movie was very good 👍
### Sentiment Analysis with Linear

<table>
<thead>
<tr>
<th>Example</th>
<th>Label</th>
<th>Feature</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>the movie was very good</em></td>
<td>👍</td>
<td><img src="good" alt="good" /></td>
<td>Unigrams</td>
</tr>
<tr>
<td><em>the movie was very bad</em></td>
<td>👎</td>
<td><img src="bad" alt="bad" /></td>
<td>Unigrams</td>
</tr>
<tr>
<td><em>the movie was not bad</em></td>
<td>👍</td>
<td>![not bad](not bad)</td>
<td>Bigrams</td>
</tr>
<tr>
<td><em>the movie was not very good</em></td>
<td>👎</td>
<td>![not very good](not very good)</td>
<td>Trigrams</td>
</tr>
<tr>
<td><em>the movie was not really very enjoyable</em></td>
<td></td>
<td></td>
<td>4-grams!</td>
</tr>
</tbody>
</table>
Drawbacks

‣ More complex features capture interactions but scale badly (13M unigrams, 1.3B 4-grams in Google n-grams)

‣ Can we do better than seeing every n-gram once in the training data?

not very good not so great

‣ Instead of more complex linear functions, let’s use simpler nonlinear functions, namely neural networks

the movie was not really very enjoyable
Neural Networks: XOR

- Let’s see how we can use neural nets to learn a simple nonlinear function
- **Inputs** $x_1, x_2$
  (generally $x = (x_1, \ldots, x_m)$)
- **Output** $y$
  (generally $y = (y_1, \ldots, y_n)$)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y = x_1 \text{ XOR } x_2$</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>
Neural Networks: XOR

\[ y = a_1 x_1 + a_2 x_2 \]

\[ y = a_1 x_1 + a_2 x_2 + a_3 \tanh(x_1 + x_2) \]

“or”

(looks like action potential in neuron)
Neural Networks: XOR

\[ y = a_1 x_1 + a_2 x_2 \]

\[ y = a_1 x_1 + a_2 x_2 + a_3 \tanh(x_1 + x_2) \]

\[ y = -x_1 - x_2 + 2 \tanh(x_1 + x_2) \]

"or"
Neural Networks: XOR

\[ y = -2x_1 - x_2 + 2 \tanh(x_1 + x_2) \]
Neural Networks

(Linear model: $y = \mathbf{w} \cdot \mathbf{x} + b$

$y = g(\mathbf{w} \cdot \mathbf{x} + b)$

$y = g(\mathbf{Wx} + \mathbf{b})$

Nonlinear transformation  Warp space  Shift

Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
Neural Networks

Linear classifier

...possible because we transformed the space!

Neural network

Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
Deep Neural Networks

(this was our neural net from the XOR example)

\[ y_1 = g(w_1 \cdot x + b_1) \]
Deep Neural Networks

\[ y_1 = g(w_1 \cdot x + b_1) \]

\[ y = g(Wx + b) \]
Deep Neural Networks

\[ y = g(Wx + b) \]
\[ z = g(Vg(Wx + b) + c) \]

output of first layer

\[ z = g(Vy + c) \]
Neural Networks

Linear classifier

Neural network

…possible because we transformed the space!

Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology
Deep Neural Networks

Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
Deep Neural Networks

\[ y = g(Wx + b) \]
\[ z = g(Vg(Wx + b) + c) \]
output of first layer

With no nonlinearity:
\[ z = VWx + Vb + c \]
Equivalent to \[ z = Ux + d \]

Adopted from Chris Dyer
Deep Neural Networks

- Nodes in the hidden layer can learn interactions or conjunctions of features

\[ y = -2x_1 - x_2 + 2 \tanh(x_1 + x_2) \]
Learning Neural Networks

Change in output w.r.t. input

Change in output w.r.t. hidden

Change in hidden w.r.t. input

Change in output w.r.t. input

- Computing these looks like running this network in reverse (backpropagation)
Outline

- Motivation for neural networks
- Feedforward neural networks
  - Applying feedforward neural networks to NLP
- Convolutional neural networks
- Application examples
- Tools
Feedforward Bag-of-words

\[ x_2 \]
\[ \mathbb{I}[good] \]
\[ 1 \]
\[ -1 \]
\[ 0 \]
\[ 0 \]
\[ x_1 \]
\[ \mathbb{I}[not] \]

\[ y = g(Wx + b) \]

real-valued matrix, 
dims = vocabulary size (~10k) x 
hidden layer size (~100)

binary vector, 
length = vocabulary size
Drawbacks to FFBoW

- Lots of parameters to learn
- Doesn’t preserve ordering in the input
- *really not very good* and *really not very enjoyable* — we don’t know the relationship between *good* and *enjoyable*
- Doesn’t preserve ordering in the input
Word Embeddings

- word2vec: turn each word into a 100-dimensional vector
- Context-based embeddings: find a vector predictive of a word’s context
- Words in similar contexts will end up with similar vectors
Feedforward with word vectors

- Can capture word similarity

- Each $x$ now represents multiple bits of input
- Can capture word similarity

The movie was good.

$y = g(Wx + b)$

hidden layer size $\sim 100 \times$ (sentence length ($\sim 10$) $\times$ vector size ($\sim 100$))

binary vector, length = sentence length $\times$ vector size
Feedforward with word vectors

\[ y = g(Wx + b) \]

Need our model to be shift-invariant, like bag-of-words is
Comparing Architectures

‣ Instead of more complex linear functions, let’s use simpler nonlinear functions

‣ Feedforward bag-of-words: didn’t take advantage of word similarity, lots of parameters to learn

‣ Feedforward with word vectors: our parameters are attached to particular indices in a sentence

‣ Solution: convolutional neural nets
Outline

‣ Motivation for neural networks
‣ Feedforward neural networks
‣ Applying feedforward neural networks to NLP
‣ Convolutional neural networks
‣ Application examples
‣ Tools
Convolutional Networks

The movie was good.

<table>
<thead>
<tr>
<th>Word</th>
<th>Filter Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.03</td>
</tr>
<tr>
<td>movie</td>
<td>0.02</td>
</tr>
<tr>
<td>was</td>
<td>0.1</td>
</tr>
<tr>
<td>good</td>
<td>1.1</td>
</tr>
<tr>
<td>.</td>
<td>0.0</td>
</tr>
</tbody>
</table>

"good" filter output

max = 1.1
the movie was good.

max = 1.1
Convolutional Networks

- Input: $n$ vectors of length $m$ each
  - $k$ filters of length $m$ each
  - $k$ filter outputs of length 1 each
- Takes variable-length input and turns it into fixed-length output
- Filters are initialized randomly and then learned
Convolutional Networks

- Word vectors for similar words are similar, so convolutional filters will have similar outputs.
Convolutional Networks

Analogous to bigram features in bag-of-words models
Comparing Architectures

- Instead of more complex linear functions, let’s use simpler nonlinear functions

- Convolutional networks let us take advantage of word similarity

- Convolutional networks are translation-invariant like bag-of-words

- Convolutional networks can capture local interactions with filters of width $> 1$ (i.e. “not good” )
Outline

‣ Motivation for neural networks
‣ Feedforward neural networks
‣ Applying feedforward neural networks to NLP
‣ Convolutional neural networks
‣ Application examples
‣ Tools
Sentence Classification

The movie was not good. convolutional fully connected
Object Recognition

Convolutional layers

Fully connected layers

AlexNet (2012)
Neural networks are

- NNs are built from convolutional layers, fully connected layers, and some other types
- Can chain these together into various architectures
- Any neural network built this way can be learned from data!
the movie was not good.

Sentence Classification

convolutional fully connected

$\mathbf{x} \rightarrow \mathbf{W} \rightarrow \mathbf{y}$

prediction
Sentence Classification

- Outperforms highly-tuned bag-of-words model

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>SST-1</th>
<th>SST-2</th>
<th>Subj</th>
<th>TREC</th>
<th>CR</th>
<th>MPQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-multichannel</td>
<td>81.1</td>
<td>47.4</td>
<td>88.1</td>
<td>93.2</td>
<td>92.2</td>
<td>85.0</td>
<td>89.4</td>
</tr>
<tr>
<td>NBSVM (Wang and Manning, 2012)</td>
<td>79.4</td>
<td>–</td>
<td>–</td>
<td>93.2</td>
<td>–</td>
<td>81.8</td>
<td>86.3</td>
</tr>
</tbody>
</table>

Taken from Kim (2014)
Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified Armstrong from his seven consecutive Tour de France wins from 1999–2005.

Lance Edward Armstrong is an American former professional road cyclist.

Armstrong County is a county in Pennsylvania.

- Conventional: compare vectors from tf-idf features for overlap
- Convolutional networks can capture many of the same effects: distill notions of topic from n-grams

Francis-Landau, Durrett, and Klein (NAACL 2016)
Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified Lance Edward Armstrong from his seven consecutive Tour de France wins from 1999–2005.
Syntactic Parsing

He wrote a long report on Mars.

He wrote a long report on Mars.

My report—on Mars

report—on Mars

wrote—on Mars
He wrote a long report on Mars.

chart value = score(rule) + chart(left child) + chart(right child)
Syntactic Parsing

Features need to combine surface information and syntactic information, but looking at words directly ends up being very sparse.
Scoring parses with neural nets

$$\text{score}
\begin{pmatrix}
\text{NP} \\
\text{NP} \\
\text{PP}
\end{pmatrix}
= s^\top \cdot \text{vector representation of rule being applied}$$

Durrett and Klein (ACL 2015)
He wrote a long report on Mars

Parsing a sentence:
- Feedforward pass on nets
- Discrete feature computation
- Run CKY dynamic program

Durrett and Klein (ACL 2015)
Machine Translation

- Long short-term memory units
Long Short-Term Memory Networks

- Map sequence of inputs to sequence of outputs

Taken from [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
Google is moving towards this architecture, performance is constantly improving compared to phrase-based methods.
Neural Network

- Tensorflow: [https://www.tensorflow.org/](https://www.tensorflow.org/)
  - By Google, actively maintained, bindings for many languages

- Theano: [http://deeplearning.net/software/theano/](http://deeplearning.net/software/theano/)
  - University of Montreal, less and less maintained

- Torch: [http://torch.ch/](http://torch.ch/)
  - Facebook AI Research, Lua
Neural Network

```
import theano
import theano.tensor as T

# Define symbolic variables
x = T.matrix('x')
y = T.matrix('y')
z = T.matrix('z')

# Compute some other values symbolically
a = x + y
b = a * z
c = a + b

# Compile a function that computes c
f = theano.function(
    inputs=[x, y, z],
    outputs=c,
)

# Evaluate the compiled function
# on some real values
xx = np.random.randn(4, 5)
yy = np.random.randn(4, 5)
zz = np.random.randn(4, 5)
print f(xx, yy, zz)

# Repeat the same computation
# explicitly using numpy ops
aa = xx + yy
bb = aa * zz
cc = aa + bb
```

Compile a function that produces c from x, y, z (generates code)

http://tmmse.xyz/content/images/2016/02/theano-computation-graph.png
Word Vector Tools

- **Word2Vec**: [https://radimrehurek.com/gensim/models/word2vec.html](https://radimrehurek.com/gensim/models/word2vec.html)
  - [https://code.google.com/archive/p/word2vec/](https://code.google.com/archive/p/word2vec/)
  - Python code, actively maintained

  - Word vectors trained on very large corpora
Convolutional Networks

- CNNs for sentence class.: [https://github.com/yoonkim/CNN_sentence](https://github.com/yoonkim/CNN_sentence)
  - Based on tutorial from: [http://deeplearning.net/tutorial/lenet.html](http://deeplearning.net/tutorial/lenet.html)
  - Python code
  - Trains very quickly
Takeaways

- Neural networks have several advantages for NLP:
  - We can use *simpler nonlinear functions* instead of more complex linear functions
  - We can take advantage of word similarity
  - We can build models that are both position-dependent (feedforward neural networks) and position-independent (convolutional networks)
- NNs have natural applications to many problems
- While conventional linear models often still do well, neural nets are increasingly the state-of-the-art for many tasks