(Multinomial) Logistic Regression + Feature Engineering
Reminders

• Homework 3: KNN, Perceptron, Lin.Reg.
  – Out: Wed, Feb 7
  – Due: Wed, Feb 14 at 11:59pm

• Homework 4: Logistic Regression
  – Out: Wed, Feb 14
  – Due: Fri, Feb 23 at 11:59pm
MULTINOMIAL LOGISTIC REGRESSION
**Multinomial Logistic Regression**

**Chalkboard**

- Background: Multinomial distribution
- Definition: Multi-class classification
- Geometric intuitions
- Multinomial logistic regression model
- Generative story
- Reduction to binary logistic regression
- Partial derivatives and gradients
- Applying Gradient Descent and SGD
- Implementation w/ sparse features
Debug that Program!

**In-Class Exercise: Think-Pair-Share**
Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

**Buggy Program:**

```python
while not converged:
    for i in shuffle([1,...,N]):
        for k in [1,...,K]:
            theta[k] = theta[k] - lambda * grad(x[i], y[i], theta, k)
```

**Assume:** `grad(x[i], y[i], theta, k)` returns the gradient of the negative log-likelihood of the training example `(x[i],y[i])` with respect to vector `theta[k]`. `lambda` is the learning rate. `N = #` of examples. `K = #` of output classes. `M = #` of features. `theta` is a `K` by `M` matrix.
Debug that Program!

**In-Class Exercise: Think-Pair-Share**

Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

**Buggy Program:**

```python
while not converged:
    for i in shuffle([1,...,N]):
        for k in [1,...,K]:
            for m in [1,..., M]:
                theta[k,m] = theta[k,m] + lambda * grad(x[i], y[i], theta, k,m)
```

**Assume:** `grad(x[i], y[i], theta, k, m)` returns the partial derivative of the negative log-likelihood of the training example `(x[i],y[i])` with respect to `theta[k,m]`. `lambda` is the learning rate. `N = # of examples. K = # of output classes. M = # of features. theta is a K by M matrix.`
FEATURE ENGINEERING
p(y|x) ∝ \exp(\Theta_y \cdot f(\cdot))

Handcrafted Features
Where do features come from?

- **Hand-crafted features**
  - Sun et al., 2011
  - Zhou et al., 2005

- **Feature Engineering**
  - First word before M1
  - Second word before M1
  - Bag-of-words in M1
  - Head word of M1
  - Other word in between
  - First word after M2
  - Second word after M2
  - Bag-of-words in M2
  - Head word of M2
  - Bigrams in between
  - Words on dependency path
  - Country name list
  - Personal relative triggers
  - Personal title list
  - WordNet Tags
  - Heads of chunks in between
  - Path of phrase labels
  - Combination of entity types

- **Feature Learning**
Where do features come from?

**Feature Engineering**

- **hand-crafted features**
  - Sun et al., 2011
  - Zhou et al., 2005

**Feature Learning**

- **word embeddings**
  - Mikolov et al., 2013

**Diagram**

- **Look-up table**
  - input (context words)
  - embedding
  - missing word

- **Classifier**
  - unsupervised learning

- **CBOW model in Mikolov et al. (2013)**
  - cat: 0.11 .23 ... -0.45
  - dog: 0.13 .26 ... -0.52

- **Similar words, similar embeddings**
  - dog: cat:
Where do features come from?

- **Feature Engineering**
  - Hand-crafted features:
    - Sun et al., 2011
    - Zhou et al., 2005

- **Feature Learning**
  - Word embeddings:
    - Mikolov et al., 2013
  - String embeddings:
    - Socher, 2011
    - Collobert & Weston, 2008

**Convolutional Neural Networks** *(Collobert and Weston 2008)*

**Recursive Auto Encoder** *(Socher 2011)*
Where do features come from?

- **Feature Engineering**
  - Hand-crafted features
  - Sun et al., 2011
  - Zhou et al., 2005

- **Feature Learning**
  - Word embeddings
    - Mikolov et al., 2013
  - String embeddings
    - Socher, 2011
    - Collobert & Weston, 2008
  - Tree embeddings
    - Socher et al., 2013
    - Hermann & Blunsom, 2013

Example sentence: The [movie] showed [wars]
Where do features come from?

- Hand-crafted features
  - Sun et al., 2011
  - Zhou et al., 2005

- Word embedding features
  - Turian et al., 2010
  - Koo et al., 2008
  - Mikolov et al., 2013

- Tree embeddings
  - Socher et al., 2013
  - Hermann & Blunsom, 2013

- String embeddings
  - Socher, 2011
  - Collobert & Weston, 2008

Refine embedding features with semantic/syntactic information.
Where do features come from?

1. **Hand-crafted features**
   - Sun et al., 2011
   - Zhou et al., 2005

2. **Word embeddings**
   - Mikolov et al., 2013

3. **Tree embeddings**
   - Socher et al., 2013
   - Hermann & Blunsom, 2013

4. **String embeddings**
   - Socher, 2011
   - Collobert & Weston, 2008

5. **Best of both worlds?**
   - Turian et al., 2010
   - Koo et al., 2008
   - Hermann et al., 2014
Feature Engineering for NLP

Suppose you build a logistic regression model to predict a part-of-speech (POS) tag for each word in a sentence.

What features should you use?

The movie I watched depicted hope
Feature Engineering for NLP

Per-word Features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>x(1)</th>
<th>x(2)</th>
<th>x(3)</th>
<th>x(4)</th>
<th>x(5)</th>
<th>x(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-capital(w_i)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>endswith(w_i,&quot;e&quot;)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>endswith(w_i,&quot;d&quot;)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>endswith(w_i,&quot;ed&quot;)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w_i == &quot;aardvark&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w_i == &quot;hope&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Feature Engineering for NLP

Context Features:

<table>
<thead>
<tr>
<th>...</th>
<th>x^{(1)}</th>
<th>x^{(2)}</th>
<th>x^{(3)}</th>
<th>x^{(4)}</th>
<th>x^{(5)}</th>
<th>x^{(6)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_i == “watched”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w_{i+1} == “watched”</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w_{i-1} == “watched”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w_{i+2} == “watched”</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w_{i-2} == “watched”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Feature Engineering for NLP

Context Features:

<table>
<thead>
<tr>
<th></th>
<th>x(1)</th>
<th>x(2)</th>
<th>x(3)</th>
<th>x(4)</th>
<th>x(5)</th>
<th>x(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wi == “I”</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wi+1 == “I”</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wi-1 == “I”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wi+2 == “I”</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wi-2 == “I”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The movie I watched depicted hope
The movie I watched depicted hope.

Feature Engineering for NLP

Table 3. Tagging accuracies with different feature templates and other changes on the WSJ 19-21 development set.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3gramMemm</td>
<td>See text</td>
<td>248,798</td>
<td>52.07%</td>
<td>96.92%</td>
<td>88.99%</td>
</tr>
<tr>
<td>NAACL 2003</td>
<td>See text and [1]</td>
<td>460,552</td>
<td>55.31%</td>
<td>97.15%</td>
<td>88.61%</td>
</tr>
<tr>
<td>Replication</td>
<td>See text and [1]</td>
<td>460,551</td>
<td>55.62%</td>
<td>97.18%</td>
<td>88.92%</td>
</tr>
<tr>
<td>Replication’</td>
<td>+rareFeatureThresh = 5</td>
<td>482,364</td>
<td>55.67%</td>
<td>97.19%</td>
<td>88.96%</td>
</tr>
<tr>
<td>5w</td>
<td>+⟨t₀, w₋₂⟩, ⟨t₀, w₂⟩</td>
<td>730,178</td>
<td>56.23%</td>
<td>97.20%</td>
<td>89.03%</td>
</tr>
<tr>
<td>5wShapes</td>
<td>+⟨t₀, s₋₁⟩, ⟨t₀, s₀⟩, ⟨t₀, s₊₁⟩</td>
<td>731,661</td>
<td>56.52%</td>
<td>97.25%</td>
<td>89.81%</td>
</tr>
<tr>
<td>5wShapesDS</td>
<td>+ distributional similarity</td>
<td>737,955</td>
<td>56.79%</td>
<td>97.28%</td>
<td>90.46%</td>
</tr>
</tbody>
</table>
Feature Engineering for NLP

Suppose you want to predict whether the word is the root (i.e. predicate) of the sentence.

What features should you use?

\[ \text{The [movie]}_{M_1} \] \text{ I watched depicted [hope]}_{M_2} \]
Feature Engineering for NLP

Per-word Features:

<table>
<thead>
<tr>
<th></th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_4 )</th>
<th>( f_5 )</th>
<th>( f_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>on-path(( w_i ))</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>is-between(( w_i ))</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>head-of-M1(( w_i ))</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>head-of-M2(( w_i ))</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>before-M1(( w_i ))</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>before-M2(( w_i ))</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Example sentence:

The \([movie]_{M_1}\) I watched depicted \([hope]_{M_2}\)
Feature Engineering for NLP

Per-word Features:

on-path(w_i)

is-between(w_i)

head-of-M1(w_i)

head-of-M2(w_i)

before-M1(w_i)

before-M2(w_i)

...
Feature Engineering for NLP

Per-word Features: (with conjunction)

\[
\begin{align*}
on-path(w_i) & \land w_i = "depicted" \\
is-between(w_i) & \land w_i = "depicted" \\
head-of-M1(w_i) & \land w_i = "depicted" \\
head-of-M2(w_i) & \land w_i = "depicted" \\
before-M1(w_i) & \land w_i = "depicted" \\
before-M2(w_i) & \land w_i = "depicted"
\end{align*}
\]

\[f_5 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \ldots \end{bmatrix}\]

The \text{[movie]}_{M_1} I \text{ watched } \text{depicted } \text{[hope]}_{M_2}
Feature Engineering for CV

Edge detection (Canny)

Corner Detection (Harris)

Figures from http://opencv.org
Feature Engineering for CV

Scale Invariant Feature Transform (SIFT)

Figure from Lowe (1999) and Lowe (2004)