Maxent Models (III), & Neural Language Models

CMSC 473/673
UMBC
September 25th, 2017

Some slides adapted from 3SLP
Recap from last time...
Maximum Entropy Models

*a more general language model*

\[
p(y \mid x) \propto \exp(\theta \cdot f(x, y))
\]

\[
\text{argmax}_X \ p(Y \mid X) \ast p(X)
\]

\[
p(w_1, \ldots, w_S) = \prod_{i=1}^{S} p(w_i \mid w_{i-N+1}, \ldots, w_{i-1})
\]

*classify in one go*

\[
p(x \mid y) \propto \exp(\theta \cdot f(x, y))
\]

\[
\text{argmax}_X p(X \mid Y)
\]
Maximum Entropy Models

\[ p_\theta (y \mid x) \propto \exp (\theta \cdot f(x, y)) \]

Feature Weights
- Natural parameters
- Distribution Parameters

Feature function(s)
- Sufficient statistics
- “Strength” function(s)
What if you can’t find the roots?
Follow the derivative

Set $t = 0$
Pick a starting value $\theta_t$
Until converged:
1. Get value $y_t = F(\theta_t)$
2. Get derivative $g_t = F'(\theta_t)$
3. Get scaling factor $\rho_t$
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set $t += 1$
What if you can’t find the roots?
Follow the derivative

Set $t = 0$
Pick a starting value $\theta_t$

Until converged:
1. Get value $y_t = F(\theta_t)$
2. Get derivative $g_t = F'(\theta_t)$
3. Get scaling factor $\rho_t$
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set $t += 1$
Connections to Other Techniques

Log-Linear Models

(Multinomial) logistic regression
Softmax regression
Maximum Entropy models (MaxEnt)
Generalized Linear Models
Discriminative Naïve Bayes
Very shallow (sigmoidal) neural nets

as statistical regression
based in information theory
a form of viewed as
to be cool today :)

today :)
Objective = Full Likelihood?

\[
\prod_i P_{\theta} (y_i \mid x_i) \propto \prod_i \exp (\theta \cdot f (x_i, y_i))
\]
Objective = Full Likelihood?

\[ \prod_i P_{\theta} (y_i | x_i) \propto \prod_i \exp (\theta \cdot f (x_i, y_i)) \]

These values can have very small magnitude \(\Rightarrow\) underflow

Differentiating this product could be a pain
Logarithms

$(0, 1] \rightarrow (-\infty, 0]$ 

Products $\rightarrow$ Sums

$\log(ab) = \log(a) + \log(b)$

$\log(a/b) = \log(a) - \log(b)$

Inverse of exp

$\log(\exp(x)) = x$
Log-Likelihood

Wide range of (negative) numbers

Sums are more stable

\[
\log \prod_i P_{\theta} (y_i \mid x_i) = \sum_i \log P_{\theta} (y_i \mid x_i)
\]

Products ➔ Sums

\[
\log(ab) = \log(a) + \log(b)
\]

\[
\log(a/b) = \log(a) - \log(b)
\]
Log-Likelihood

Wide range of (negative) numbers

Sums are more stable

\[
\log \prod_i P_\theta (y_i \mid x_i) = \sum_i \log P_\theta (y_i \mid x_i)
\]

Inverse of \( \exp \)
\log(\exp(x)) = x

\[ p(y \mid x) \propto \exp(\theta \cdot f(x, y)) \]

\[ = \sum_i \theta \cdot f(x_i, y_i) - \log Z(x_i) \]
Log-Likelihood

Wide range of (negative) numbers

Sums are more stable

\[
\log \prod_i P_{\theta} (y_i \mid x_i) = \sum_i \log P_{\theta} (y_i \mid x_i)
\]

Inverse of \( \exp \)

\( \log(\exp(x)) = x \)

\( p(y \mid x) \propto \exp(\theta \cdot f(x, y)) \)

\[
= \sum_i \theta \cdot f(x_i, y_i) - \log Z(x_i)
\]

Differentiating this becomes nicer (even though \( Z \) depends on \( \theta \))
Log-Likelihood

Wide range of (negative) numbers

Sums are more stable

\[
\log \prod_i P_{\theta}(y_i \mid x_i) = \sum_i \log P_{\theta}(y_i \mid x_i) \\
= \sum_i \theta \cdot f(x_i, y_i) - \log Z(x_i) \\
= F(\theta)
\]

Differentiating this becomes nicer (even though Z depends on \(\theta\))
Expectations

number of pieces of candy

\[
\frac{1}{6} \times 1 + \frac{1}{6} \times 2 + \frac{1}{6} \times 3 + \frac{1}{6} \times 4 + \frac{1}{6} \times 5 + \frac{1}{6} \times 6 = 3.5
\]
Expectations

number of pieces of candy

\[
\frac{1}{2} \times 1 + \frac{1}{10} \times 2 + \frac{1}{10} \times 3 + \frac{1}{10} \times 4 + \frac{1}{10} \times 5 + \frac{1}{10} \times 6 = 2.5
\]
Expectations

number of pieces of candy

\[ \frac{1}{2} \times 1 + \frac{1}{10} \times 2 + \frac{1}{10} \times 3 + \frac{1}{10} \times 4 + \frac{1}{10} \times 5 + \frac{1}{10} \times 6 = 2.5 \]

\[ \mathbb{E}[X] = \sum_{x} x \ p(x) \]
Expectations

The expected number of pieces of candy is given by:

\[ \mathbb{E}[X] = \sum_x x \cdot p(x) = 1/2 \cdot 1 + 1/10 \cdot 2 + 1/10 \cdot 3 + 1/10 \cdot 4 + 1/10 \cdot 5 + 1/10 \cdot 6 = 2.5 \]
Log-Likelihood Gradient

Each component $k$ is the difference between:
Log-Likelihood Gradient

Each component $k$ is the difference between:

the total value of feature $f_k$ in the training data

$$\sum_i f_k(x_i, y_i)$$
Log-Likelihood Gradient

Each component $k$ is the difference between:

the total value of feature $f_k$ in the training data

and

the total value the current model $p_\theta$ *thinks* it computes for feature $f_k$

\[
\sum_i f_k(x_i, y_i)
\]

\[
\sum_i \mathbb{E}_{p_\theta}[f(x_i, y')] 
\]
Log-Likelihood Gradient

Each component $k$ is the difference between:

the total value of feature $f_k$ in the training data

and

the total value the current model $p_\theta$ thinks it computes for feature $f_k$

\[
\sum_i f_k(x_i, y_i) \quad \text{and} \quad \sum_i \mathbb{E}_{p_\theta}[f(x_i, y')] \]

"moment matching"
https://www.csee.umbc.edu/courses/undergraduate/473/f17/loglin-tutorial/

https://goo.gl/B23Rxo

Lesson 6
Log-Likelihood Gradient Derivation

$$\nabla_{\theta} F(\theta) = \nabla_{\theta} \sum_{i} [\theta \cdot f(x_i, y_i) - \log Z(x_i)]$$
Log-Likelihood Gradient Derivation

\[ \nabla_{\theta} F(\theta) = \nabla_{\theta} \sum_{i} [\theta \cdot f(x_i, y_i) - \log Z(x_i)] = \sum_{i} f(x_i, y_i) - \text{depends on } \theta \]
Log-Likelihood Gradient Derivation

\[ \nabla_{\theta} F(\theta) = \nabla_{\theta} \sum_{i} [\theta \cdot f(x_i, y_i) - \log Z(x_i)] \]

\[ = \sum_{i} f(x_i, y_i) - \text{depends on } \theta \]

\[ Z(x_i) = \sum_{y'} \exp(\theta \cdot f(x_i, y')) \]
$$\nabla_\theta F(\theta) = \nabla_\theta \sum_i \left[ \theta \cdot f(x_i, y_i) - \log Z(x_i) \right]$$

$$= \sum_i f(x_i, y_i) - \text{depends on } \theta$$

$$Z(x_i) = \sum_{y'} \exp(\theta \cdot f(x_i, y'))$$

$$\frac{d \log g(x)}{dx} = \frac{1}{g(x)} \frac{d g(x)}{dx}$$
Log-Likelihood Gradient Derivation

$$\nabla_\theta F(\theta) = \nabla_\theta \sum_i [\theta \cdot f(x_i, y_i) - \log Z(x_i)]$$

$$= \sum_i f(x_i, y_i) - \sum_i \sum_{y'} \frac{\exp(\theta \cdot f(x_i, y'))}{Z(x_i)} f(x_i, y)$$
Log-Likelihood Gradient Derivation

\[
\nabla_\theta F(\theta) = \nabla_\theta \sum_i \left[ \theta \cdot f(x_i, y_i) - \log Z(x_i) \right]
\]

\[
= \sum_i f(x_i, y_i) - \sum_i \sum_{y'} \frac{\exp(\theta \cdot f(x_i, y'))}{Z(x_i)} f(x_i, y)
\]

*use the (calculus) chain rule*

\[
\frac{\partial}{\partial \theta} \log g(h(\theta)) = \left( \frac{\partial g}{\partial h(\theta)} \right) \left( \frac{\partial h}{\partial \theta} \right)
\]
Log-Likelihood Gradient Derivation

\[ \nabla_{\theta} F(\theta) = \nabla_{\theta} \sum_{i} \left[ \theta \cdot f(x_i, y_i) - \log Z(x_i) \right] \]

\[ = \sum_{i} f(x_i, y_i) - \sum_{i} \sum_{y'} \frac{\exp(\theta \cdot f(x_i, y'))}{Z(x_i)} f(x_i, y') \]

*use the (calculus) chain rule*

\[ \frac{\partial}{\partial \theta} \log g(h(\theta)) = \left( \frac{\partial g}{\partial h(\theta)} \right) \left( \frac{\partial h}{\partial \theta} \right) \]

*scalar \( p(y' | x_i) \)  
*vector of functions*
Log-Likelihood Gradient Derivation

$$\nabla_\theta F(\theta) = \nabla_\theta \sum_i \left[ \theta \cdot f(x_i, y_i) - \log Z(x_i) \right]$$

$$= \sum_i f(x_i, y_i) - \sum_i \sum_{y'} \frac{\exp(\theta \cdot f(x_i, y'))}{Z(x_i)} f(x_i, y)$$

$$= \sum_i f(x_i, y_i) - \sum_i \mathbb{E}_{y' \sim p_\theta(\cdot | x_i)} \left[ f(x_i, y) \right]$$
Log-Likelihood Derivative Derivation

$$\nabla_\theta F(\theta) = \nabla_\theta \sum_i \left[ \theta \cdot f(x_i, y_i) - \log Z(x_i) \right]$$

$$= \sum_i f(x_i, y_i) - \sum_i \sum_{y'} \frac{\exp(\theta \cdot f(x_i, y'))}{Z(x_i)} f(x_i, y)$$

$$= \sum_i f(x_i, y_i) - \sum_i \mathbb{E}_{y' \sim p_\theta(\cdot | x_i)}[f(x_i, y)]$$

$$\frac{\partial F}{\partial \theta_k} = \sum_i f_k(x_i, y_i) - \sum_i \sum_{y'} f_k(x_i, y') p(y' | x_i)$$
\[ \nabla_\theta F(\theta) = \sum_i f(x_i, y_i) - \sum_i \mathbb{E}_{y' \sim p_\theta(\cdot | x_i)}[f(x_i, y')] \]

Do we want these to \textit{fully} match?

What does it mean if they do?

What if we have missing values in our data?
Preventing Extreme Values

Naïve Bayes

Extreme values are 0 probabilities

\[ P(y \mid x) = \frac{\text{count}(x, y)}{\text{count}(x)} \]

\[ P(y \mid x) = \frac{\text{count}(x, y) + \alpha}{\text{count}(x) + L\alpha} \]
Preventing Extreme Values

Naïve Bayes

Extreme values are 0 probabilities

\[
P(y \mid x) = \frac{\text{count}(x, y)}{\text{count}(x)}
\]

\[
P(y \mid x) = \frac{\text{count}(x, y) + \alpha}{\text{count}(x) + L\alpha}
\]

Log-linear models

Extreme values are large \( \theta \) values

\[
F(\theta) = \sum_i \log P_\theta(y_i \mid x_i)
\]

\[
F(\theta) = \sum_i \log P_\theta(y_i \mid x_i) - R(\theta)
\]
Preventing Extreme Values

Naïve Bayes

Extreme values are 0 probabilities

\[ P(y \mid x) = \frac{\text{count}(x, y)}{\text{count}(x)} \]

\[ P(y \mid x) = \frac{\text{count}(x, y) + \alpha}{\text{count}(x) + L\alpha} \]

Log-linear models

Extreme values are large \( \theta \) values

\[ F(\theta) = \sum_i \log P_{\theta}(y_i \mid x_i) \]

\[ F(\theta) = \sum_i \log P_{\theta}(y_i \mid x_i) - R(\theta) \]
(Squared) L2 Regularization

\[ R(\theta) = \| \theta \|_2^2 = \sum_k \theta_k^2 \]
(More on) Connections to Other Machine Learning Techniques
Classification: Discriminative Naïve Bayes

Naïve Bayes

Label/class

Observed features
Classification:
Discriminative Naïve Bayes

Naïve Bayes

Label/class

Observed features

Maxent/
Logistic Regression
Multinomial Logistic Regression

$$\text{score}(x, y_i) = \theta \cdot f(x, y_i)$$
Multinomial Logistic Regression

\[
\text{score}(x, y_i) = \theta \cdot f(x, y_i)
\]

\[
\text{score}(x, y_i) = \sum_k \theta_k f_k(x, y_i)
\]
Multinomial Logistic Regression

$$\log \frac{p(y_i)}{p(y_K)} = \theta \cdot f(x, y_i)$$
Understanding Conditioning

\[ p(y \mid x) \propto \text{count}(x) \]

Is this a good language model?
Understanding Conditioning

\[ p(y \mid x) \propto \exp(\theta \cdot f(x)) \]

Is this a good language model?
Understanding Conditioning

\[ p(y \mid x) \propto \exp(\theta \cdot f(x)) \]

*Is this a good language model? (no)*
Understanding Conditioning

\[ p(x \mid y) \propto \exp(\theta \cdot f(y)) \]

Is this a good posterior classifier? (no)
Lesson 11
Connections to Other Techniques

Log-Linear Models

(Multinomial) logistic regression
Softmax regression
Maximum Entropy models (MaxEnt)
Generalized Linear Models
Discriminative Naïve Bayes

as statistical regression
based in information theory
a form of
viewed as
to be cool today :)

Very shallow (sigmoidal) neural nets
Revisiting the SNAP Function

\[ p(y \mid x) \propto \exp(\theta \cdot f(x, y)) \]

softmax
Revisiting the SNAP Function

\[ p(y \mid x) \propto \exp(\theta \cdot f(x, y)) \]

softmax

\[ \text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \]
N-gram Language Models

given some context...

$w_{i-3}$  $w_{i-2}$  $w_{i-1}$

predict the next word

$w_i$
N-gram Language Models

given some context...

compute beliefs about what is likely...

predict the next word

\[ p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{count}(w_{i-3}, w_{i-2}, w_{i-1}, w_i) \]
N-gram Language Models

given some context...

compute beliefs about what is likely...

predict the next word

\[ p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{count}(w_{i-3}, w_{i-2}, w_{i-1}, w_i) \]
Maxent Language Models

given some context...

compute beliefs about what is likely...

\[ p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta \cdot f(w_{i-3}, w_{i-2}, w_{i-1}, w_i)) \]

predict the next word
Neural Language Models

given some context...

compute beliefs about what is likely...

predict the next word

\[ p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta \cdot f(w_{i-3}, w_{i-2}, w_{i-1}, w_i)) \]

can we *learn* the feature function(s)?
Neural Language Models

given some context...

compute beliefs about what is likely...

\[
p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))
\]

can we learn the feature function(s) for just the context?

can we learn word-specific weights (by type)?
Neural Language Models

given some context...

create/use “distributed representations”...

compute beliefs about what is likely...

\[ p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1})) \]

predict the next word
Neural Language Models

given some context...

create/use "distributed representations"...

combine these representations...

compute beliefs about what is likely...

\[
p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta_w \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))
\]

predict the next word
Neural Language Models

given some context...

create/use “distributed representations”...

combine these representations...

compute beliefs about what is likely...

predict the next word

\[ p(w_i \mid w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1})) \]
Neural Language Models

given some context...

create/use “distributed representations”...

combine these representations...

compute beliefs about what is likely...

\[
p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))
\]

predict the next word

## Baselines

<table>
<thead>
<tr>
<th>LM Name</th>
<th>N-gram</th>
<th>Params.</th>
<th>Test Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation</td>
<td>3</td>
<td>---</td>
<td>336</td>
</tr>
<tr>
<td>Kneser-Ney backoff</td>
<td>3</td>
<td>---</td>
<td>323</td>
</tr>
<tr>
<td>Kneser-Ney backoff</td>
<td>5</td>
<td>---</td>
<td>321</td>
</tr>
<tr>
<td>Class-based backoff</td>
<td>3</td>
<td>500 classes</td>
<td>312</td>
</tr>
<tr>
<td>Class-based backoff</td>
<td>5</td>
<td>500 classes</td>
<td>312</td>
</tr>
</tbody>
</table>

### Baselines

<table>
<thead>
<tr>
<th>LM Name</th>
<th>N-gram</th>
<th>Params.</th>
<th>Test Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation</td>
<td>3</td>
<td>---</td>
<td>336</td>
</tr>
<tr>
<td>Kneser-Ney backoff</td>
<td>3</td>
<td>---</td>
<td>323</td>
</tr>
<tr>
<td>Kneser-Ney backoff</td>
<td>5</td>
<td>---</td>
<td>321</td>
</tr>
<tr>
<td>Class-based backoff</td>
<td>3</td>
<td>500 classes</td>
<td>312</td>
</tr>
<tr>
<td>Class-based backoff</td>
<td>5</td>
<td>500 classes</td>
<td>312</td>
</tr>
</tbody>
</table>

### NPLM

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>60</td>
<td>50</td>
<td>No</td>
<td>268</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>50</td>
<td>Yes</td>
<td>257</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>100</td>
<td>No</td>
<td>276</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>100</td>
<td>Yes</td>
<td>252</td>
</tr>
</tbody>
</table>

### Baselines

<table>
<thead>
<tr>
<th>LM Name</th>
<th>N-gram</th>
<th>Params.</th>
<th>Test Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation</td>
<td>3</td>
<td>---</td>
<td>336</td>
</tr>
<tr>
<td>Kneser-Ney backoff</td>
<td>3</td>
<td>---</td>
<td>323</td>
</tr>
<tr>
<td>Kneser-Ney backoff</td>
<td>5</td>
<td>---</td>
<td>321</td>
</tr>
<tr>
<td>Class-based backoff</td>
<td>3</td>
<td>500 classes</td>
<td>312</td>
</tr>
<tr>
<td>Class-based backoff</td>
<td>5</td>
<td>500 classes</td>
<td>312</td>
</tr>
</tbody>
</table>

### NPLM

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>60</td>
<td>50</td>
<td>No</td>
<td>268</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>50</td>
<td>Yes</td>
<td>257</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>100</td>
<td>No</td>
<td>276</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>100</td>
<td>Yes</td>
<td>252</td>
</tr>
</tbody>
</table>

“we were not able to see signs of over-fitting (on the validation set), possibly because we ran only 5 epochs (over 3 weeks using 40 CPUs)” (Sect. 4.2)