Naïve Bayes, Maxent and Neural Models

CMSC 473/673
UMBC

Some slides adapted from 3SLP
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
  Terminology: bag-of-words
  “Naïve” assumption
  Training & performance
  NB as a language

Maximum Entropy classifiers
  Defining the model
  Defining the objective
  Learning: Optimizing the objective
  Math: gradient derivation

Neural (language) models
Probabilistic Classification

Directly model the posterior

\[ p(X \mid Y) = h(X; Y) \]

Discriminatively trained classifier

Model the posterior with Bayes rule

\[ p(X \mid Y) \propto p(Y \mid X) \ast p(X) \]

Generatively trained classifier

Posterior Classification/Decoding
*maximum a posteriori*

Noisy Channel Model Decoding
Posterior Decoding: Probabilistic Text Classification

Assigning subject categories, topics, or genres
Spam detection
Authorship identification

Age/gender identification
Language Identification
Sentiment analysis
...

\[ p(X \mid Y) = \frac{p(Y \mid X) \ast p(X)}{p(Y)} \]

class-based likelihood (language model)
observation likelihood (averaged over all classes)
prior probability of class
observed data
Noisy Channel Model

what I want to tell you "sports"

what you actually see "The Os lost again..."

hypothesized intent "sad stories" "sports"

reweight according to what's likely "sports"
Noisy Channel

Machine translation
Speech-to-text
Spelling correction
Text normalization

Part-of-speech tagging
Morphological analysis
Image captioning
...

$$p(X | Y) = \frac{p(Y | X) * p(X)}{p(Y)}$$

possible (clean) output
observed (noisy) text

translation/decode model
(clean) language model
observation (noisy) likelihood
Use Logarithms

\[ p(X | Y) = \frac{p(Y | X) \times p(X)}{p(Y)} \]

\[ \arg\max_X \log p(Y | X) + \log p(X) \]
Accuracy, Precision, and Recall

**Accuracy**: % of items correct
\[
\frac{TP + TN}{TP + FP + FN + TN}
\]

**Precision**: % of selected items that are correct
\[
\frac{TP}{TP + FP}
\]

**Recall**: % of correct items that are selected
\[
\frac{TP}{TP + FN}
\]

<table>
<thead>
<tr>
<th>Actually Correct</th>
<th>Actually Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selected/Guessed</strong></td>
<td>True Positive (TP)</td>
</tr>
<tr>
<td><strong>Not select/not guessed</strong></td>
<td>False Negative (FN)</td>
</tr>
</tbody>
</table>
A combined measure: $F$

Weighted (harmonic) average of Precision & Recall

\[
F = \frac{(1 + \beta^2) \cdot P \cdot R}{(\beta^2 \cdot P) + R}
\]

Balanced F1 measure: $\beta=1$

\[
F_1 = \frac{2 \cdot P \cdot R}{P + R}
\]
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
  - Terminology: bag-of-words
  - “Naïve” assumption
  - Training & performance
  - NB as a language

Maximum Entropy classifiers
  - Defining the model
  - Defining the objective
  - Learning: Optimizing the objective
  - Math: gradient derivation

Neural (language) models
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!
Bag of Words Representation

\[ \gamma(\text{seen}) = 2 \]

\[ \gamma(\text{sweet}) = 1 \]

\[ \gamma(\text{whimsical}) = 1 \]

\[ \gamma(\text{recommend}) = 1 \]

\[ \gamma(\text{happy}) = 1 \]

...
Naïve Bayes Classifier

\[ \arg\max_x p(Y \mid X) \cdot p(X) \]

Start with Bayes Rule

Q: Are we doing discriminative training or generative training?
Naïve Bayes Classifier

$$\arg\max_x p(Y \mid X) \ast p(X)$$

*Start with Bayes Rule*

**Q:** Are we doing discriminative training or generative training?

**A:** generative training
Naïve Bayes Classifier

$$\arg\max_X \prod_i p(Y_i|X) \times p(X)$$

Adopt naïve bag of words representation $Y_i$
Naïve Bayes Classifier

\[
\text{argmax}_X \prod_i p(Y_i | X) \times p(X)
\]

Adopt naïve bag of words representation \( Y_i \).

Assume position doesn’t matter.
Naïve Bayes Classifier

\[ \arg\max_X \prod_i p(Y_i | X) \times p(X) \]

Adopt naïve bag of words representation \( Y_i \)

Assume position doesn’t matter

Assume the feature probabilities are independent given the class \( X \)
Multinomial Naïve Bayes: Learning

From training corpus, extract *Vocabulary*
Multinomial Naïve Bayes: Learning

From training corpus, extract Vocabulary

Calculate $P(c_j)$ terms
For each $c_j$ in $C$ do

$$docs_j = \text{all docs with class } = c_j$$

$$p(c_j) = \frac{|docs_j|}{\# docs}$$
With enough data, the classifier may not matter

Brill and Banko (2001)
Multinomial Naïve Bayes: Learning

From training corpus, extract \textit{Vocabulary}

Calculate $P(c_j)$ terms
For each $c_j$ in $C$ do

$docs_j = $ all docs with class $= c_j$

$$p(c_j) = \frac{|docs_j|}{\# docs}$$

Calculate $P(w_k | c_j)$ terms

$Text_j = $ single doc containing all $docs_j$

For each word $w_k$ in \textit{Vocabulary}

$n_k = $ # of occurrences of $w_k$ in $Text_j$

$$p(w_k | c_j) = \text{class (unigram) LM}$$
Naïve Bayes and Language Modeling

Naïve Bayes classifiers can use any sort of feature

But if, as in the previous slides
- We use only word features
- we use all of the words in the text (not a subset)

Then
- Naïve Bayes has an important similarity to language modeling
Naïve Bayes as a Language Model

<table>
<thead>
<tr>
<th>Positive Model</th>
<th>Negative Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1  I</td>
<td>0.2  I</td>
</tr>
<tr>
<td>0.1  love</td>
<td>0.001  love</td>
</tr>
<tr>
<td>0.01 this</td>
<td>0.01 this</td>
</tr>
<tr>
<td>0.05 fun</td>
<td>0.005 fun</td>
</tr>
<tr>
<td>0.1 film</td>
<td>0.1 film</td>
</tr>
</tbody>
</table>
Naïve Bayes as a Language Model

Which class assigns the higher probability to \( s \)?

<table>
<thead>
<tr>
<th>Positive Model</th>
<th>Negative Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 I</td>
<td>0.2 I</td>
</tr>
<tr>
<td>0.1 love</td>
<td>0.001 love</td>
</tr>
<tr>
<td>0.01 this</td>
<td>0.01 this</td>
</tr>
<tr>
<td>0.05 fun</td>
<td>0.005 fun</td>
</tr>
<tr>
<td>0.1 film</td>
<td>0.1 film</td>
</tr>
</tbody>
</table>

I love this fun film
Naïve Bayes as a Language Model

Which class assigns the higher probability to \( s \)?

<table>
<thead>
<tr>
<th>Positive Model</th>
<th>Negative Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>I</td>
</tr>
<tr>
<td>0.1</td>
<td>love</td>
</tr>
<tr>
<td>0.01</td>
<td>this</td>
</tr>
<tr>
<td>0.05</td>
<td>fun</td>
</tr>
<tr>
<td>0.1</td>
<td>film</td>
</tr>
<tr>
<td>0.2</td>
<td>I</td>
</tr>
<tr>
<td>0.001</td>
<td>love</td>
</tr>
<tr>
<td>0.01</td>
<td>this</td>
</tr>
<tr>
<td>0.005</td>
<td>fun</td>
</tr>
<tr>
<td>0.1</td>
<td>film</td>
</tr>
</tbody>
</table>
Naïve Bayes as a Language Model

Which class assigns the higher probability to $s$?

<table>
<thead>
<tr>
<th>Positive Model</th>
<th>Negative Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 l</td>
<td>0.2 l</td>
</tr>
<tr>
<td>0.1 love</td>
<td>0.001 love</td>
</tr>
<tr>
<td>0.01 this</td>
<td>0.01 this</td>
</tr>
<tr>
<td>0.05 fun</td>
<td>0.005 fun</td>
</tr>
<tr>
<td>0.1 film</td>
<td>0.1 film</td>
</tr>
</tbody>
</table>

$$5 \times 10^{-7} \approx P(s|\text{pos}) > P(s|\text{neg}) \approx 1 \times 10^{-9}$$
Summary: Naïve Bayes is Not So Naïve

Very Fast, low storage requirements

Robust to Irrelevant Features

Very good in domains with many equally important features

Optimal if the independence assumptions hold

Dependable baseline for text classification (but often not the best)
But: Naïve Bayes Isn’t Without Issue

Model the posterior in one go?

Are the features really uncorrelated?

Are plain counts always appropriate?

Are there “better” ways of handling missing/noisy data? (automated, more principled)
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
  Terminology: bag-of-words
  “Naïve” assumption
  Training & performance
  NB as a language

Maximum Entropy classifiers
  Defining the model
  Defining the objective
  Learning: Optimizing the objective
  Math: gradient derivation

Neural (language) models
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 :)
Maxent Models for Classification: Discriminatively or Generatively Trained

- Directly model the posterior
  \[ p(X \mid Y) = h(X; Y) \]
  Discriminatively trained classifier

- Model the posterior with Bayes rule
  \[ p(X \mid Y) \propto p(Y \mid X) \ast p(X) \]
  Generatively trained classifier
Maximum Entropy (Log-linear) Models

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

discriminatively trained: classify in one go
Maximum Entropy (Log-linear) Models

\[ p(x \mid y) \propto p(y \mid x)p(x) \]

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

generatively trained:
learn to model language
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junín department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junín department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junín department, central Peruvian mountain region.

We need to score the different combinations.
Score and Combine Our Possibilities

\[ \text{score}_1 (\text{fatally shot, ATTACK}) \]
\[ \text{score}_2 (\text{seriously wounded, ATTACK}) \]
\[ \text{score}_3 (\text{Shining Path, ATTACK}) \]

\ldots

\[ \text{score}_k (\text{department, ATTACK}) \]

\ldots

are all of these uncorrelated?

\[ \text{COMBINE} \]

posterior probability of ATTACK
Score and Combine Our Possibilities

\[ \text{score}_1(\text{fatally shot, ATTACK}) \]
\[ \text{score}_2(\text{seriously wounded, ATTACK}) \]
\[ \text{score}_3(\text{Shining Path, ATTACK}) \]

\[ \text{COMBINE} \]

\[ \text{posterior probability of ATTACK} \]

\[ \ldots \]

**Q:** What are the score and combine functions for Naïve Bayes?
Scoring Our Possibilities

\[
\text{score}(\text{ATTAck}) = \text{score}_1(\text{fatally shot, ATTAck}) + \text{score}_2(\text{seriously wounded, ATTAck}) + \text{score}_3(\text{Shining Path, ATTAck}) + \ldots
\]

Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
What function... operates on any real number?

is never less than 0?
What function... operates on any real number?
is never less than 0?

\[ f(x) = \exp(x) \]
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.

Learn the scores (but we’ll declare what combinations should be looked at)
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
Maxent Modeling: Feature Functions

\[ p(\text{ATTACK}) \propto \exp\left( \begin{array}{c}
\text{weight}_1 \times \text{occurs}_1(\text{fatally shot, ATTACK}) \\
\text{weight}_2 \times \text{occurs}_2(\text{seriously wounded, ATTACK}) \\
\text{weight}_3 \times \text{occurs}_3(\text{Shining Path, ATTACK}) \\
\ldots
\end{array} \right) \]

Feature functions help extract useful features (characteristics) of the data.

Generally templated

Often binary-valued (0 or 1), but can be real-valued

\[ \text{occurs}_{\text{target,type}}(\text{fatally shot, ATTACK}) = \begin{cases} 
1, & \text{target} == \text{fatally shot and type == ATTACK} \\
0, & \text{otherwise}
\end{cases} \]

binary

Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.
More on Feature Functions

Feature functions help extract useful features (characteristics) of the data

Generally \textit{templated}

Often binary-valued (0 or 1), but can be real-valued

\begin{align*}
\text{occurs}_{\text{target, type}}(\text{fatally shot, ATTACK}) &= \begin{cases} 1, & \text{target == fatally shot and type == ATTACK} \\ 0, & \text{otherwise} \end{cases} \\
\text{binary}
\end{align*}

\begin{align*}
\text{occurs}_{\text{target, type}}(\text{fatally shot, ATTACK}) &= \log p(\text{fatally shot} | \text{ATTACK}) \\
&\quad + \log p(\text{type} | \text{ATTACK}) \\
&\quad + \log p(\text{ATTACK} | \text{type}) \\
\text{Templated real-valued}
\end{align*}

\begin{align*}
\text{occurs}(\text{fatally shot, ATTACK}) &= \log p(\text{fatally shot} | \text{ATTACK}) \\
\text{Non-templated real-valued}
\end{align*}

???

\begin{align*}
\text{occurs}(\text{fatally shot, ATTACK}) &= \log p(\text{fatally shot} | \text{ATTACK}) \\
\text{Non-templated count-valued}
\end{align*}
More on Feature Functions

Feature functions help extract useful features (characteristics) of the data

Generally *templated*

Often binary-valued (0 or 1), but can be real-valued

\[
\text{occurs}_{\text{target}, \text{type}}(\text{fatally shot, ATTACK}) =
\begin{cases} 
1, & \text{target} == \text{fatally shot} \text{ and } \text{type} == \text{ATTACK} \\
0, & \text{otherwise}
\end{cases}
\]

*binary*

\[
\text{occurs}_{\text{target}, \text{type}}(\text{fatally shot, ATTACK}) = \log p(\text{fatally shot} | \text{ATTACK})
+ \log p(\text{type} | \text{ATTACK})
+ \log p(\text{ATTACK} | \text{type})
\]

*Templated real-valued*

\[
\text{occurs}(\text{fatally shot, ATTACK}) = \log p(\text{fatally shot} | \text{ATTACK})
\]

*Non-templated real-valued*

\[
\text{occurs}(\text{fatally shot, ATTACK}) = \text{count}(\text{fatally shot} | \text{ATTACK})
\]

*Non-templated count-valued*
Three people have been fatally shot, and five people, including a mayor, were seriously wounded as a result of a Shining Path attack today against a community in Junin department, central Peruvian mountain region.

Q: How do we define Z?
Normalization for Classification

\[ Z = \sum \exp \left( \begin{array}{l}
\text{weight}_1 \cdot \text{occurs}_1(\text{fatally shot, ATTACK}) \\
\text{weight}_2 \cdot \text{occurs}_2(\text{seriously wounded, ATTACK}) \\
\text{weight}_3 \cdot \text{occurs}_3(\text{Shining Path, ATTACK}) \\
\vdots
\end{array} \right) \]

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

classify doc y with label x in one go
Normalization for Language Model

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

general class-based (X) language model of doc y
Normalization for Language Model

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

Can be significantly harder in the general case
Normalization for Language Model

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

*general class-based (X) language model of doc y*

Can be significantly harder in the general case

Simplifying assumption: maxent n-grams!
Understanding Conditioning

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

Is this a good language model?
Understanding Conditioning

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

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)
https://www.csee.umbc.edu/courses/undergraduate/473/f18/loglin-tutorial/

https://goo.gl/BQCdH9

Lesson 11
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
  Terminology: bag-of-words
  “Naïve” assumption
  Training & performance
  NB as a language

Maximum Entropy classifiers
  Defining the model
  Defining the objective
  Learning: Optimizing the objective
  Math: gradient derivation

Neural (language) models
\[ p_{\theta}(x \mid y) \]

probabilistic model

\[ F(\theta; \{x, y\}) \]

goal (given observations)
Objective = Full Likelihood?

\[ \prod_i p_{\theta}(x_i | y_i) \propto \prod_i \exp(\theta^T f(x_i, y_i)) \]

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

$$\log \prod_{i} p_{\theta}(x_{i} | y_{i}) = \sum_{i} \log p_{\theta}(x_{i} | y_{i})$$

Wide range of (negative) numbers

Sums are more stable

Products $\Rightarrow$ 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

Inverse of \( \exp \):
\[
\log(\exp(x)) = x
\]

\[
\log \prod_i p_{\theta}(x_i | y_i) = \sum_i \log p_{\theta}(x_i | y_i)
\]

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

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

Wide range of (negative) numbers

Sums are more stable

\[
\log \prod_i p_{\theta}(x_i | y_i) = \sum_i \log p_{\theta}(x_i | y_i)
\]

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

\[
= F(\theta)
\]
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
- Terminology: bag-of-words
- “Naïve” assumption
- Training & performance
- NB as a language

Maximum Entropy classifiers
- Defining the model
- Defining the objective
- Learning: Optimizing the objective
- Math: gradient derivation

Neural (language) models
How will we optimize $F(\theta)$?

Calculus
$F(\theta)$
$F(\theta)$

$\theta^*$
$F'(\theta)$

*derivative of $F$ wrt $\theta*$
Example

\[ F(x) = -(x-2)^2 \]

Differentiate

\[ F'(x) = -2x + 4 \]

Solve \( F'(x) = 0 \)

\[ x = 2 \]
Common Derivative Rules

\[
\frac{d \exp x}{dx} = \exp x
\]

\[
\frac{df(x)g(x)}{dx} = \frac{df(x)}{dx}g(x) + \frac{dg(x)}{dx}f(x)
\]

\[
\frac{d \log x}{dx} = \frac{1}{x}
\]

\[
\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}
\]
What if you can’t find the roots?
Follow the derivative

$F'(\theta)$

*derivative of $F$ wrt $\theta*$
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 $y_t = F(\theta_t)$
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 $g_t = F'(\theta_t)$
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 \( \theta_{t+1} = \theta_t + \rho_t \cdot g_t \)
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 $\theta_{t+1} = \theta_t + \rho_t \cdot g_t$
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 $\theta_{t+1} = \theta_t + \rho_t \cdot g_t$
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 $t += 1$
Gradient = Multi-variable derivative

\[ \nabla_{\theta} F(\theta) = \left( \frac{\partial F}{\partial \theta_1}, \frac{\partial F}{\partial \theta_2}, \ldots, \frac{\partial F}{\partial \theta_K} \right) \]

K-dimensional input

K-dimensional output
Gradient Ascent
Gradient Ascent
Gradient Ascent
Gradient Ascent
Gradient Ascent
Gradient Ascent
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
  Terminology: bag-of-words
  “Naïve” assumption
  Training & performance
  NB as a language

Maximum Entropy classifiers
  Defining the model
  Defining the objective
  Learning: Optimizing the objective
  Math: gradient derivation

Neural (language) models
Expectations

number of pieces of candy

1/6 * 1 +
1/6 * 2 +
1/6 * 3 +
1/6 * 4 +
1/6 * 5 +
1/6 * 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

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

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

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

number of pieces of candy

\[
\begin{align*}
\frac{1}{2} \cdot 1 + \\
\frac{1}{10} \cdot 2 + \\
\frac{1}{10} \cdot 3 + \\
\frac{1}{10} \cdot 4 + \\
\frac{1}{10} \cdot 5 + \\
\frac{1}{10} \cdot 6
\end{align*}
\]

= 2.5
Log-Likelihood

\[
\log \prod_i p_\theta(x_i | y_i) = \sum_i \log p_\theta(x_i | y_i)
\]

Wide range of (negative) numbers

Sums are more stable

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

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

\[
= F(\theta)
\]
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 $f_k$
Lesson 6
Log-Likelihood Gradient Derivation

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

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

\[ \nabla_\theta \sum_i f(x_i, y_i) - \]

\[ Z(y_i) = \sum_{x'} \exp(\theta \cdot f(x', y_i)) \]
Log-Likelihood Gradient Derivation

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

\[
= \nabla_{\theta} \sum_{i} f(x_i, y_i) - \sum_{i} \sum_{x'} \frac{\exp(\theta^T f(x', y_i))}{Z(y_i)} f(x', y_i)
\]

*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(x' | y_i)*

*vector of functions*
Log-Likelihood Gradient Derivation

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

\[ = \nabla_\theta \sum_i f(x_i, y_i) - \sum_i \sum_{x'} \frac{\exp(\theta^T f(x', y_i))}{Z(y_i)} f(x', y_i) \]

Do we want these to fully match?

What does it mean if they do?

What if we have missing values in our data?
Gradient Optimization

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 $\theta_{t+1} = \theta_t + \rho_t \cdot g_t$
5. Set $t += 1$

$$\partial F / \partial \theta_k = \sum_i f_k(x_i, y_i) - \sum_i \sum_{y'} f_k(x_i, y') p(y'|x_i)$$

$$\sum_i \theta^T f(x_i, y_i) - \log Z(y_i)$$
\[ \nabla_\theta F(\theta) = \sum_i f(x_i, y_i) - \sum_i \mathbb{E}_{y' \sim p_\theta(\cdot|x_i)}[f(x_i, y')] \]

Do we want these to 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

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

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

Extremal values are 0 probabilities

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

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

Log-linear models

Extremal values are large \( \theta \) values

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

\[ F(\theta) = \sum_i \log P_\theta (y_i | x_i) - R(\theta) \]

regularization
(Squared) L2 Regularization

\[ R(\theta) = \| \theta \|_2^2 = \sum_\theta \theta_k^2 \]
Outline

Recap: classification (MAP vs. noisy channel) & evaluation

Naïve Bayes (NB) classification
  Terminology: bag-of-words
  “Naïve” assumption
  Training & performance
  NB as a language

Maximum Entropy classifiers
  Defining the model
  Defining the objective
  Learning: Optimizing the objective
  Math: gradient derivation

Neural (language) models
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...

\[ \text{predict the next word} \]
N-gram Language Models

Given some context...

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

Compute beliefs about what is likely...

Predict the next word
N-gram Language Models

given some context...

compute beliefs about what is likely...

predict the next word

\[ p(w_i \mid w_{i-3}, w_{i-2}, w_{i-1}) \propto \text{count}(w_{i-3}, w_{i-2}, w_{i-1}, w_i) \]
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...

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

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

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

predict the next word...
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...

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

predict the next word
Neural Language Models

given some context...

create/use “distributed representations”...

combine these representations...

compute beliefs about what is likely...

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

predict the next word
Neural Language Models

given some context...

create/use “distributed representations”...

combine these representations...

compute beliefs about what is likely...

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

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)