Logistic Regression

Instructor: Wei Xu

Some slides adapted from Dan Jurafsky and Brendan O’Connor
Naïve Bayes Recap

• Bag of words (order independent)

• Features are assumed independent given class

\[ P(x_1, \ldots, x_n|c) = P(x_1|c) \ldots P(x_n|c) \]

Q: Is this really true?
The problem with assuming conditional independence

- Correlated features -> double counting evidence
  - Parameters are estimated independently

- This can hurt classifier accuracy and calibration
Logistic Regression

• Doesn’t assume features are independent

• Correlated features don’t “double count”
What are “Features”?

• A feature function, $f$
  – Input: Document, $D$ (a string)
  – Output: Feature Vector, $X$
What are “Features”?

\[
f(d) = \begin{pmatrix}
\text{count(“boring”)}
\\text{count(“not boring”)}
\text{length of document}
\text{author of document}
\vdots
\end{pmatrix}
\]

Doesn’t have to be just “bag of words”
Feature Templates

• Typically “feature templates” are used to generate many features at once

• For each word:
  – \${w}_count
  – \${w}_lowercase
  – \${w}_with_NOT_before_count
Logistic Regression: Example

• Compute Features:

\[ f(d_i) = x_i = \begin{pmatrix} \text{count(“nigerian”)} \\ \text{count(“prince”)} \\ \text{count(“nigerian prince”)} \end{pmatrix} \]

• Assume we are given some weights:

\[ w = \begin{pmatrix} -1.0 \\ -1.0 \\ 4.0 \end{pmatrix} \]
Logistic Regression: Example

• Compute Features
• We are given some weights
• Compute the dot product:

\[ z = \sum_{i=0}^{\left|X\right|} w_i x_i \]
Logistic Regression: Example

• Compute the dot product:

\[ z = \sum_{i=0}^{\mid X \mid} w_i x_i \]

• Compute the logistic function:

\[ P(\text{spam} \mid x) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}} \]
The Logistic function

\[ P(\text{spam}|x) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}} \]
Logistic Regression

• (Log) Linear Model - similar to Naïve Bayes
The Dot Product

\[ z = \sum_{i=0}^{\mid X \mid} w_i x_i \]

• Intuition: weighted sum of features
• All Linear models have this form
A log-linear model is a mathematical model has the form a function whose logarithm is a linear combination of the parameters.

\[
\exp\left( \sum_{i=0}^{|X|} w_i x_i \right)
\]
NAIVE BAYES

IS ALSO A LOG-LINEAR MODEL?
Naïve Bayes as a log-linear model

• Q: what are the features?

• Q: what are the weights?
Naïve Bayes as a Log-Linear Model

\[ P(\text{spam}|D) \propto P(\text{spam}) \prod_{w \in D} P(w|\text{spam}) \]

\[ P(\text{spam}|D) \propto P(\text{spam}) \prod_{w \in \text{Vocab}} P(w|\text{spam})^{x_i} \]

\[ \log P(\text{spam}|D) \propto \log P(\text{spam}) + \sum_{w \in \text{Vocab}} x_i \cdot \log P(w|\text{spam}) \]
Naïve Bayes as a Log-Linear Model

\[
\log P(\text{spam}|D) \propto \log P(\text{spam}) + \sum_{w \in \text{Vocab}} x_i \cdot \log P(w|\text{spam})
\]

In both naïve Bayes and logistic regression we compute the dot product!
NB vs. LR

• Both compute the dot product

• NB: sum of log probabilities

• LR: logistic function
NB vs. LR: Parameter Learning

• Naïve Bayes:
  – Learn conditional probabilities independently by counting

• Logistic Regression:
  – Learn weights jointly
LR: Learning Weights

• Given: a set of feature vectors and labels

• Goal: learn the weights
LR: Learning Weights

<table>
<thead>
<tr>
<th>Document</th>
<th>Feature</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{11}$ $x_{12}$ $x_{13}$ ... $x_{1n}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{21}$ $x_{22}$ $x_{23}$ ... $x_{2n}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{d1}$ $x_{d2}$ $x_{d3}$ ... $x_{dn}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_n$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Q: what parameters should we choose?

- What is the right value for the weights?

- Maximum Likelihood Principle:
  - Pick the parameters that maximize the probability of the $y$ labels in the training data given the observations $x$. 
Maximum Likelihood Estimation

\[ w_{\text{MLE}} = \arg \max_w \log P(y_1, \ldots, y_d | x_1, \ldots, x_d; w) \]

\[ = \arg \max_w \sum_i \log P(y_i | x_i; w) \]

\[ = \arg \max_w \sum_i \log \left\{ \begin{array}{ll}
p_i, & \text{if } y_i = 1 \\
1 - p_i, & \text{if } y_i = 0 \end{array} \right\} \]

\[ = \arg \max_w \sum_i \log p_i^{(y_i=1)} (1 - p_i)^{(y_i=0)} \]
Maximum Likelihood Estimation

\[ \arg\max_w \sum_i \log p_i \mathbb{1}(y_i=1)(1 - p_i) \mathbb{1}(y_i=0) \]

\[ = \arg\max_w \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i) \]

- Unfortunately there is no closed form solution
  - (like there was with naïve Bayes)
Closed Form Solution

- A Closed Form Solution is a simple solution that works instantly without any loops, functions etc.
- E.g. the sum of integer from 1 to n

Iterative Algorithm:

```python
s = 0
for i in 1 to n
    s = s + i
end for
print s
```

Closed Form Solution:

```
s = n(n + 1 )/2
```
Maximum Likelihood Estimation

• Solution:
  – Iteratively climb the log-likelihood surface through the derivatives for each weight

• Luckily, the derivatives turn out to be nice
Gradient Ascent
Gradient Ascent

Loop While not converged:

For all features $j$, compute and add derivatives:

$$w_{j}^{\text{new}} = w_{j}^{\text{old}} + \eta \frac{\partial}{\partial w_{j}} \mathcal{L}(w)$$

$\mathcal{L}(w)$: Training set log-likelihood

$$\left( \frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \ldots, \frac{\partial \mathcal{L}}{\partial w_n} \right)$$: Gradient vector
Gradient ascent
Gradient ascent