CS 188: Artificial Intelligence
Fall 2009

Lecture 23: Perceptrons
11/17/2009

Announcements

- Project 4: Due Thursday!
- Final Contest: Qualifications are on!
  - P5 will be due late enough to give you plenty of contest time
Recap: General Naïve Bayes

- A general naïve Bayes model:
  - $Y$: label to be predicted
  - $F_1, \ldots, F_n$: features of each instance

\[
P(Y, F_1 \ldots F_n) = P(Y) \prod_i P(F_i | Y)
\]

Example Naïve Bayes Models

- Bag-of-words for text
  - One feature for every word position in the document
  - All features share the same conditional distributions
  - Maximum likelihood estimates: word frequencies, by label

- Pixels for images
  - One feature for every pixel, indicating whether it is on (black)
  - Each pixel has a different conditional distribution
  - Maximum likelihood estimates: how often a pixel is on, by label
Naïve Bayes Training

- Data: labeled instances, e.g. emails marked as spam/ham by a person
  - Divide into training, held-out, and test

- Features are known for every training, held-out and test instance

- Estimation: count feature values in the training set and normalize to get maximum likelihood estimates of probabilities

- Smoothing (aka regularization): adjust estimates to account for unseen data

Recap: Laplace Smoothing

- Laplace’s estimate (extended):
  - Pretend you saw every outcome \( k \) extra times
  
  \[
P_{\text{LAP}, k}(x) = \frac{c(x) + k}{c(\cdot) + k|X|}
\]
  
  - What’s Laplace with \( k = 0 \)?
  - \( k \) is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition:
  - Can be derived by dividing
  
  \[
P_{\text{LAP}, k}(x|y) = \frac{c(x, y) + k}{c(\cdot, y) + k|X|}
\]
Better: Linear Interpolation

- Linear interpolation for conditional likelihoods
  - Idea: the conditional probability of a feature $x$ given a label $y$ should be close to the marginal probability of $x$
  - Example: A rare word like “interpolation” should be similarly rare in both ham and spam (a priori)
  - Procedure: Collect relative frequency estimates of both conditional and marginal, then average

$$P_{ML}(x|y) = \frac{\text{count}(x,y)}{\text{count}(.|y)} \quad P_{ML}(x) = \frac{\text{count}(x)}{\text{count}(\cdot)}$$

$$P_{LIN}(x|y) = (1 - \alpha)P_{ML}(x|y) + (\alpha)P_{ML}(x)$$

- Effect: Features have odds ratios closer to 1

Real NB: Smoothing

- Odds ratios without smoothing:

|            | $P(W|$ham$)$ | $P(W|$spam$)$ |
|------------|--------------|---------------|
| south-west | inf          | inf           |
| nation     | inf          | inf           |
| morally    | inf          | inf           |
| nicely     | inf          | inf           |
| extent     | inf          | inf           |
| ...        | inf          | inf           |

|            | $P(W|$ham$)$ | $P(W|$spam$)$ |
|------------|--------------|---------------|
| screens    | inf          | inf           |
| minute     | inf          | inf           |
| guaranteed | inf          | inf           |
| $205.00$   | inf          | inf           |
| delivery   | inf          | inf           |
| ...        | inf          | inf           |
Real NB: Smoothing

- Odds ratios after smoothing:

\[
\frac{P(W|\text{ham})}{P(W|\text{spam})} \quad \frac{P(W|\text{spam})}{P(W|\text{ham})}
\]

| helvetica | 11.4 | verdana | 28.8 |
| seems     | 10.8 | Credit  | 28.4 |
| group     | 10.2 | ORDER   | 27.2 |
| ago       | 8.4  | <FONT> | 26.9 |
| areas     | 8.3  | money   | 26.5 |
| ...       |      | ...     |      |

Do these make more sense?

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: P(F_i|Y) and P(Y)
  - Hyperparameters, like the amount of smoothing to do: k, α

- Where to learn which unknowns
  - Learn parameters from training set
  - Can't tune hyperparameters on training data (why?)
  - For each possible value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data

\[
\alpha
\]

Proportion of $P_{ML}(x)$ in $P(x|y)$
Baselines

- First task when classifying: get a baseline
  - Baselines are very simple “straw man” procedures
  - Help determine how hard the task is
  - Help know what a “good” accuracy is

- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g. for spam filtering, might label everything as spam
  - Accuracy might be very high if the problem is skewed

- When conducting real research, we usually use previous work as a (strong) baseline

Confidences from a Classifier

- The confidence of a classifier:
  - Posterior of the most likely label
    \[
    \text{confidence}(x) = \max_y P(y|x)
    \]
  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee confidence is correct

- Calibration
  - Strong calibration: confidence predicts accuracy rate
  - Weak calibration: higher confidences mean higher accuracy
  - What’s the value of calibration?
Naïve Bayes Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Confidences are useful when the classifier is calibrated

Example Errors

Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just $99.99* - the regular list price is $499! The most common question we've received about this offer is - Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

. . . To receive your $30 Amazon.com promotional certificate, click through to http://www.amazon.com/apparel and see the prominent link for the $30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .
What to Do About Errors

- Problem: there’s still spam in your inbox

- Need more features – words aren’t enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?

- Naïve Bayes models can incorporate a variety of features, but tend to do best in homogeneous cases (e.g. all features are word occurrences)

Features

- A feature is a function that signals a property of the input
  - Naïve Bayes: features are random variables & each value has conditional probabilities given the label.
  - Most classifiers: features are real-valued functions
  - Common special cases:
    - Indicator features take values 0 and 1 (or -1 and 1)
    - Count features return non-negative integers

- Features are anything you can think of for which you can write code to evaluate on an input
  - Many are cheap, but some are expensive to compute
  - Can even be the output of another classifier or model
  - Domain knowledge goes here!
Feature Extractors

- Features: anything you can compute about the input
- A feature extractor maps inputs to feature vectors

Many classifiers take feature vectors as inputs
- Feature vectors usually very sparse, use sparse encodings (i.e. only represent non-zero keys)

Generative vs. Discriminative

- Generative classifiers:
  - E.g. naïve Bayes
  - A causal model with evidence variables
  - Query model for causes given evidence

- Discriminative classifiers:
  - No causal model, no Bayes rule, often no probabilities at all!
  - Try to predict the label Y directly from X
  - Robust, accurate with varied features
  - Loosely: mistake driven rather than model driven
### Some (Simplified) Biology

- Very loose inspiration: human neurons

![Neuron Diagram](image)

### Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**

\[
\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)
\]

- If the activation is:
  - Positive, output +1
  - Negative, output -1

![Linear Classifier Diagram](image)
Example: Spam

- Imagine 4 features (spam is “positive” class):
  - free (number of occurrences of “free”)
  - money (occurrences of “money”)
  - BIAS (intercept, always has value 1)

\[ w \cdot f(x) \]

\[
\begin{array}{c|c|c|c}
 x & f(x) & w & \sum_i w_i \cdot f_i(x) \\
 \hline
 \text{“free money”} & & & \\
 \text{BIAS : 1} & \text{free : 1} & \text{BIAS : -3} & (1)(-3) + \\
 \text{free : 4} & \text{money : 1} & \text{free : 4} & (1)(4) + \\
 \text{money : 2} & \text{...} & \text{money : 2} & (1)(2) + \\
 \text{...} & \text{...} & \text{...} & \text{...} \\
 \hline
\end{array}
\]

\[ = 3 \]

Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to \( Y=+1 \)
  - Other corresponds to \( Y=-1 \)

\[ w \cdot f(x) = 0 \]

\[ +1 = \text{SPAM} \]

\[ -1 = \text{HAM} \]
Binary Perceptron Update

- Start with zero weights
- For each training instance:
  - Classify with current weights
    \[
    y = \begin{cases} 
    +1 & \text{if } w \cdot f(x) \geq 0 \\
    -1 & \text{if } w \cdot f(x) < 0 
    \end{cases}
    \]
  - If correct (i.e., \(y = y^*\)), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if \(y^*\) is -1.

\[
w = w + y^* \cdot f
\]

Multiclass Decision Rule

- If we have more than two classes:
  - Have a weight vector for each class: \(w_y\)
  - Calculate an activation for each class

\[
\text{activation}_w(x, y) = w_y \cdot f(x)
\]

- Highest activation wins

\[
y = \arg \max_y (\text{activation}_w(x, y))
\]
**Example**

```

```

**w_{SPORTS}**

<table>
<thead>
<tr>
<th>BIAS</th>
<th>win</th>
<th>game</th>
<th>vote</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**w_{POLITICS}**

<table>
<thead>
<tr>
<th>BIAS</th>
<th>win</th>
<th>game</th>
<th>vote</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**w_{TECH}**

<table>
<thead>
<tr>
<th>BIAS</th>
<th>win</th>
<th>game</th>
<th>vote</th>
<th>the</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

---

**The Perceptron Update Rule**

- Start with zero weights
- Pick up training instances one by one
- Classify with current weights
  
  \[
  y = \text{arg max } y \ w_y \cdot f(x)
  = \text{arg max } y \ \sum_i w_{y,i} \cdot f_i(x)
  \]

- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer

\[
\begin{align*}
  w_y &= w_y - f(x) \\
  w_{y^*} &= w_{y^*} + f(x)
\end{align*}
\]
Example

“win the vote”
“win the election”
“win the game”

\[ w_{SPORTS} \quad w_{POLITICS} \quad w_{TECH} \]

BIAS : win : game : vote : the :
...}

Examples: Perceptron

- Separable Case
Examples: Perceptron

- Separable Case

Mistake-Driven Classification

- For Naïve Bayes:
  - Parameters from data statistics
  - Parameters: causal interpretation
  - Training: one pass through the data

- For the perceptron:
  - Parameters from reactions to mistakes
  - Parameters: discriminative interpretation
  - Training: go through the data until held-out accuracy maxes out
Properties of Perceptrons

- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the margin or degree of separability

\[
mistakes < \frac{k}{\delta^2}
\]

Examples: Perceptron

- Non-Separable Case
**Examples: Perceptron**

- **Non-Separable Case**

**Issues with Perceptrons**

- **Overtraining**: test / held-out accuracy usually rises, then falls
  - Overtraining isn’t quite as bad as overfitting, but is similar

- **Regularization**: if the data isn’t separable, weights might thrash around
  - Averaging weight vectors over time can help (averaged perceptron)

- **Mediocre generalization**: finds a “barely” separating solution