CS 188: Artificial Intelligence
Fall 2006

Lecture 24: Perceptrons
11/28/2006

Dan Klein – UC Berkeley

Announcements

- Midterms, 2.1, 1.3 regrades all graded, will be in glookup soon

- Our record of your late days is now in glookup
  - Make sure it’s right for you and your partner
  - If you accidentally submitted an assignment weeks later, you’ll have a very wrong record – let us know

- Projects
  - 4.1 up now
  - Due 12/06

- Contest details on web
Example: Spam Filtering

- Model: \( P(C, W_1 \ldots W_n) = P(C) \prod_i P(W_i|C) \)
- Parameters:

| Parameter | \( P(C) \) | \( P(W|\text{spam}) \) | \( P(W|\text{ham}) \) |
|-----------|-----------|----------------|----------------|
| ham       | 0.66      | the : 0.016   | the : 0.021    |
| spam      | 0.33      | to : 0.015    | to : 0.013     |
|           |           | and : 0.012   | and : 0.011    |
|           |           | ...           | ...            |
|           |           | free : 0.001  | free : 0.005   |
|           |           | click : 0.001 | click : 0.004  |
|           |           | ...           | ...            |
|           |           | morally : 0.001 | screens : 0.000 |
|           |           | nicely : 0.001 | minute : 0.000 |
|           |           | ...           | ...            |

Example: OCR
Generative vs. Discriminative

- **Generative classifiers:**
  - E.g. naïve Bayes
  - We build a causal model of the variables
  - We then query that model for causes, given evidence

- **Discriminative classifiers:**
  - E.g. perceptron (next)
  - No causal model, no Bayes rule, often no probabilities
  - Try to predict output directly
  - Loosely: mistake driven rather than model driven

Errors, and What to Do

- **Examples of 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?

- 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 which signals a property of the input

- Examples:
  - ALL_CAPS: value is 1 iff email in all caps
  - HAS_URL: value is 1 iff email has a URL
  - NUM_URLS: number of URLs in email
  - VERY_LONG: 1 iff email is longer than 1K
  - SUSPICIOUS_SENDER: 1 iff reply-to domain doesn’t match originating server

- Features are anything you can think of code to evaluate on an input
  - Some cheap, some very very expensive to calculate
  - Can even be the output of another classifier
  - Domain knowledge goes here!

- In naïve Bayes, how did we encode features?
Feature Extractors

- A feature extractor maps inputs to feature vectors

  Dear Sir.
  First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidencial and top secret.

<table>
<thead>
<tr>
<th>Term</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>dear</td>
<td>1</td>
</tr>
<tr>
<td>sir</td>
<td>1</td>
</tr>
<tr>
<td>this</td>
<td>2</td>
</tr>
<tr>
<td>wish</td>
<td>0</td>
</tr>
<tr>
<td>Misspelled</td>
<td>2</td>
</tr>
<tr>
<td>Nameless</td>
<td>1</td>
</tr>
<tr>
<td>All Caps</td>
<td>0</td>
</tr>
<tr>
<td>Num URLs</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Some (Vague) Biology

- Very loose inspiration: human neurons
The Binary Perceptron

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

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

- If the activation is:
  - Positive, output 1
  - Negative, output 0

Example: Spam

- Imagine 4 features:
  - Free (number of occurrences of “free”)
  - Money (occurrences of “money”)
  - BIAS (always has value 1)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>f(x)</td>
<td>w</td>
<td>(\sum_i w_i \cdot f_i(x))</td>
</tr>
<tr>
<td>free</td>
<td>1</td>
<td>1</td>
<td>(1)(4) +</td>
</tr>
<tr>
<td>money</td>
<td>1</td>
<td>2</td>
<td>(1)(2) +</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
<td>0</td>
<td>(0)(0) +</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>BIAS</td>
<td>1</td>
<td>-3</td>
<td>(1)(-3) +</td>
</tr>
<tr>
<td>free</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>money</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“free money”
**Binary Decision Rule**

- In the space of feature vectors
  - Any weight vector is a hyperplane
  - One side will be class 1
  - Other will be class 0

\[ f \cdot w = 0 \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS</td>
<td>-3</td>
</tr>
<tr>
<td>free</td>
<td>4</td>
</tr>
<tr>
<td>money</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**The Multiclass Perceptron**

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

\[
\text{activation}_w(x, c) = \sum_i w_{c,i} \cdot f_i(x)
\]

- Highest activation wins

\[
c = \arg\max_c (\text{activation}_w(x, c))
\]
Example

“win the vote”

<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>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \begin{align*}
  w_{\text{SPORTS}} & = \begin{bmatrix}
    \text{BIAS} : -2 \\
    \text{win} : 4 \\
    \text{game} : 4 \\
    \text{vote} : 0 \\
    \text{the} : 0 \\
  \end{bmatrix} \\
  w_{\text{POLITICS}} & = \begin{bmatrix}
    \text{BIAS} : 1 \\
    \text{win} : 2 \\
    \text{game} : 0 \\
    \text{vote} : 4 \\
    \text{the} : 0 \\
  \end{bmatrix} \\
  w_{\text{TECH}} & = \begin{bmatrix}
    \text{BIAS} : 2 \\
    \text{win} : 0 \\
    \text{game} : 2 \\
    \text{vote} : 0 \\
    \text{the} : 0 \\
  \end{bmatrix}
\end{align*} \]

The Perceptron Update Rule

- Start with zero weights
- Pick up training instances one by one
- Try to classify
  \[ c = \arg \max_c \ w_c \cdot f(x) \]
  \[ = \arg \max_c \ \sum_i w_{c,i} \cdot f_i(x) \]
- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer
  \[ w_c = w_c - f(x) \]
  \[ w_{c*} = w_{c*} + f(x) \]
Example

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

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

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

Mistake-Driven Classification

- In naïve Bayes, parameters:
  - From data statistics
  - Have a causal interpretation
  - One pass through the data

- For the perceptron parameters:
  - From reactions to mistakes
  - Have a discriminative interpretation
  - 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
  \[ \text{mistakes} < \frac{1}{\delta^2} \]

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
**Summary**

- **Naïve Bayes**
  - Build classifiers using model of training data
  - Smoothing estimates is important in real systems
  - Classifier confidences are useful, when you can get them

- **Perceptrons:**
  - Make less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data

**Similarity Functions**

- Similarity functions are very important in machine learning

- **Topic for next class: kernels**
  - Similarity functions with special properties
  - The basis for a lot of advance machine learning (e.g. SVMs)
Case-Based Reasoning

- Similarity for classification
  - Case-based reasoning
  - Predict an instance’s label using similar instances

- Nearest-neighbor classification
  - 1-NN: copy the label of the most similar data point
  - K-NN: let the k nearest neighbors vote (have to devise a weighting scheme)
  - Key issue: how to define similarity
  - Trade-off:
    - Small k gives relevant neighbors
    - Large k gives smoother functions
    - Sound familiar?

- [DEMO]
  http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html

Parametric / Non-parametric

- Parametric models:
  - Fixed set of parameters
  - More data means better settings

- Non-parametric models:
  - Complexity of the classifier increases with data
  - Better in the limit, often worse in the non-limit

- (K)NN is non-parametric
Collaborative Filtering

- Ever wonder how online merchants decide what products to recommend to you?
- Simplest idea: recommend the most popular items to everyone
  - Not entirely crazy! (Why)
  - Can do better if you know something about the customer (e.g. what they've bought)
- Better idea: recommend items that similar customers bought
  - A popular technique: collaborative filtering
  - Define a similarity function over customers (how?)
  - Look at purchases made by people with high similarity
  - Trade-off: relevance of comparison set vs confidence in predictions
  - How can this go wrong?

Nearest-Neighbor Classification

- Nearest neighbor for digits:
  - Take new image
  - Compare to all training images
  - Assign based on closest example

- Encoding: image is vector of intensities:
  \[ 1 = (0.0 \ 0.0 \ 0.3 \ 0.8 \ 0.7 \ 0.1 \ldots 0.0) \]

- What's the similarity function?
  - Dot product of two images vectors?
  \[ \text{sim}(x, y) = x \cdot y = \sum_i x_i y_i \]
  - Usually normalize vectors so \( ||x|| = 1 \)
  - min = 0 (when?), max = 1 (when?)
Basic Similarity

- Many similarities based on feature dot products:

\[ \text{sim}(x, y) = f(x) \cdot f(y) = \sum_i f_i(x) f_i(y) \]

- If features are just the pixels:

\[ \text{sim}(x, y) = x \cdot y = \sum_i x_i y_i \]

- Note: not all similarities are of this form

Invariant Metrics

- Better distances use knowledge about vision
- Invariant metrics:
  - Similarities are invariant under certain transformations
    - Rotation, scaling, translation, stroke-thickness…
  - E.g:
    - \( 16 \times 16 = 256 \) pixels; a point in 256-dim space
    - Small similarity in \( \mathbb{R}^{256} \) (why?)
  - How to incorporate invariance into similarities?

This and next few slides adapted from Xiao Hu, UIUC
Rotation Invariant Metrics

- Each example is now a curve in $\mathbb{R}^{256}$
- Rotation invariant similarity:
  
  \[ s' = \max \, s(\, r(\, ), \, r(\, )\, ) \]

- E.g. highest similarity between images’ rotation lines

Tangent Families

- Problems with $s'$:
  - Hard to compute
  - Allows large transformations ($6 \rightarrow 9$)

- Tangent distance:
  - 1st order approximation at original points.
    - Easy to compute
    - Models small rotations
Template Deformation

- **Deformable templates:**
  - An “ideal” version of each category
  - Best-fit to image using min variance
  - Cost for high distortion of template
  - Cost for image points being far from distorted template
- **Used in many commercial digit recognizers**

Examples from [Hastie 94]