CS 343: Artificial Intelligence

Perceptrons

Prof. Scott Niekum — The University of Texas at Austin

[These slides based on those of Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]
Error-Driven Classification
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

- Problem: there’s still spam in your inbox

- Need more features – words aren’t enough!
  - Have you emailed the sender before?
  - Have 1M 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 when homogeneous (e.g. all features are word occurrences) and/or roughly independent
Linear Classifiers
Hello,
Do you want free printr cartriges? Why pay more when you can get them ABSOLUTELY FREE! Just
Some (Simplified) Biology

- Very loose inspiration: human neurons
Linear Classifiers

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

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
Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples

\[
\begin{align*}
\begin{array}{l}
\text{# free} : 4 \\
\text{YOUR_NAME} : -1 \\
\text{MISSPELLED} : 1 \\
\text{FROM_FRIEND} : -3 \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{l}
\text{# free} : 2 \\
\text{YOUR_NAME} : 0 \\
\text{MISSPELLED} : 2 \\
\text{FROM_FRIEND} : 0 \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{l}
\text{# free} : 0 \\
\text{YOUR_NAME} : 1 \\
\text{MISSPELLED} : 1 \\
\text{FROM_FRIEND} : 1 \\
\end{array}
\end{align*}
\]

Dot product $w \cdot f$ positive means the positive class
Decision Rules
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 $Y=-1$

$$w$$

<table>
<thead>
<tr>
<th>BIAS</th>
<th>-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td>4</td>
</tr>
<tr>
<td>money</td>
<td>2</td>
</tr>
</tbody>
</table>

...
Weight Updates
Learning: Binary Perceptron

- **Start with weights** = 0
- **For each training instance:**
  - Classify with current weights

- If correct (i.e., $y=y^*$), no change!

- If wrong: adjust the weight vector
Learning: Binary Perceptron

- Start with weights = 0
- 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.
    \[ w = w + y^* \cdot f \]
Examples: Perceptron

- Separable Case
Multiclass Decision Rule

- If we have multiple classes:
  - A weight vector for each class:
    \[ w_y \]
  - Score (activation) of a class \( y \):
    \[ w_y \cdot f(x) \]
  - Prediction highest score wins
    \[ y = \arg \max_y w_y \cdot f(x) \]

*Binary = multiclass where the negative class has weight zero*
Learning: Multiclass Perceptron

- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights
  \[ y = \text{arg max}_y w_y \cdot f(x) \]
- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer
  \[ w_y = w_y - f(x) \]
  \[ w_{y^*} = w_{y^*} + f(x) \]
Properties of Perceptrons

- **Separability**: true if 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:
  \[ \text{mistakes} < \frac{k}{\delta^2} \]
Examples: Perceptron

- Non-Separable Case
Improving the Perceptron
Problems with the Perceptron

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

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

- **Overtraining:** test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting
Fixing the Perceptron

- Idea: adjust the weight update to mitigate these effects
- MIRA*: choose an update size that fixes the current mistake...
- ... but, minimizes the change to $w$

$$\min_w \frac{1}{2} \sum_y ||w_y - w'_y||^2$$

$$w_{y^*} \cdot f(x) \geq w_y \cdot f(x) + 1$$

- The +1 helps to generalize

* Margin Infused Relaxed Algorithm
Minimum Correcting Update

\[
\min_w \frac{1}{2} \sum_y \|w_y - w'_y\|^2
\]

\[
w_y^* \cdot f \geq w_y \cdot f + 1
\]

\[
\min_\tau \|\tau f\|^2
\]

\[
w_y^* \cdot f \geq w_y \cdot f + 1
\]

\[
(w'_y + \tau f) \cdot f = (w'_y - \tau f) \cdot f + 1
\]

\[
\tau = \frac{(w'_y - w'_y^*) \cdot f + 1}{2f \cdot f}
\]

\[
w_y = w'_y - \tau f(x)
\]

\[
w_y^* = w'_y^* + \tau f(x)
\]

\[
\text{min not } \tau=0, \text{ or would not have made an error, so min will be where equality holds}
\]
In practice, it’s also bad to make updates that are too large

- Example may be labeled incorrectly
- You may not have enough features
- Solution: cap the maximum possible value of $\tau$ with some constant $C$

$$
\tau^* = \min \left( \frac{(w_y' - w_{y*}') \cdot f + 1}{2f \cdot f}, C \right)
$$

- Corresponds to an optimization that assumes non-separable data
- Usually converges faster than perceptron
- Usually better, especially on noisy data
Which of these linear separators is optimal?
Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- Only support vectors matter; other training examples are ignorable
- Support vector machines (SVMs) find the separator with max margin
- Basically, SVMs are MIRA where you optimize over all examples at once

**MIRA**

\[
\min_w \frac{1}{2} \|w - w'\|^2 \\
\forall i, \ y \ w_y^* \cdot f(x_i) \geq w_y \cdot f(x_i) + 1
\]

**SVM**

\[
\min_w \frac{1}{2} \|w\|^2 \\
\forall i, \ y \ w_y^* \cdot f(x_i) \geq w_y \cdot f(x_i) + 1
\]
Classification: Comparison

- **Naïve Bayes**
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)

- **Perceptrons / MIRA:**
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate
Apprenticeship
Pacman Apprenticeship!

- Examples are states $s$
- Candidates are pairs $(s,a)$
- “Correct” actions: those taken by expert
- Features defined over $(s,a)$ pairs: $f(s,a)$
- Score of a q-state $(s,a)$ given by:

$$w \cdot f(s,a)$$

- How is this VERY different from reinforcement learning?
Video of Pacman Apprentice
Extension: Web Search

- Information retrieval:
  - Given information needs, produce information
  - Includes, e.g. web search, question answering, and classic IR

- Web search: not exactly classification, but rather ranking

\[x = \text{“Apple Computers”}\]
Feature-Based Ranking

\( x = "\text{Apple Computer}" \)

\[
f(x, \text{Apple Computer}) = [0.3 \ 5 \ 0 \ 0 \ \ldots] \]

\[
f(x, \text{Apple Inc.}) = [0.8 \ 4 \ 2 \ 1 \ \ldots] \]
Perceptron for Ranking

- Inputs $x$
- Candidates $y$
- Many feature vectors: $f(x, y)$
- One weight vector: $w$

Prediction:

$$ y = \arg \max_y w \cdot f(x, y) $$

Update (if wrong):

$$ w = w + f(x, y^*) - f(x, y) $$