CS 343: Artificial Intelligence

Perceptrons

Profs. Peter Stone and Yuke Zhu — 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.]
Announcements

- Project 5: Ghostbusters
  - Due 4/21, 11:59 pm
  - In-class demo
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
Feature Vectors

Hello,
Do you want free printr cartriges? Why pay more when you can get them ABSOLUTELY FREE! Just

# free : 2
YOUR_NAME : 0
MISSPELLED : 2
FROM_FRIEND : 0
...

SPAM or
+

PIXEL-7,12 : 1
PIXEL-7,13 : 0
...
NUM_LOOPS : 1
...

“2”
Some (Simplified) Biology

- Very loose inspiration: human neurons
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
Weights

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

\[
\begin{align*}
\text{# free} & : 4 \\
\text{YOUR NAME} & : -1 \\
\text{MISSPELLED} & : 1 \\
\text{FROM_FRIEND} & : -3 \\
\end{align*}
\]

\[
\mathbf{w} \cdot f(x_1)
\]

\[
\begin{align*}
\text{# free} & : 2 \\
\text{YOUR NAME} & : 2 \\
\text{MISSPELLED} & : 1 \\
\text{FROM_FRIEND} & : 2 \\
\end{align*}
\]

\[
\mathbf{w} \cdot f(x_2)
\]

Dot product $\mathbf{w} \cdot f$ positive means the positive class
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>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>...</td>
<td></td>
</tr>
</tbody>
</table>

$f \cdot w = 0$

+1 = SPAM

-1 = HAM
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., \(y=y^\star\)), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if \(y^\star\) is -1.

\[
w = w + y^\star \cdot f
\]
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.

\[
\text{mistakes} < \frac{k}{\delta^2}
\]
In-class Exercises

Perceptrons Exercise*
*adapted from UCB Su19 final

For each of the datasets represented by the graphs below, please select the feature maps for which the perceptron algorithm can perfectly classify the data.

Each data point is in the form \((x_1, x_2)\), and has some label \(Y\), which is either a 1 (dot) or \(-1\) (cross).

(i) [5 pts]

\[
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix}
\]

- [ ] \[
\begin{bmatrix}
  x_1 & x_2 & 1
\end{bmatrix}
\]
- [ ] \[
\begin{bmatrix}
  x_1 & x_2 & x_2^2
\end{bmatrix}
\]
- [ ] \[
\begin{bmatrix}
  x_1 & x_2 & |x_1| 
\end{bmatrix}
\]
- [ ] \[
\begin{bmatrix}
  x_1 & x_2 & Y
\end{bmatrix}
\]
- [ ] \[
\begin{bmatrix}
  x_1 & x_2 
\end{bmatrix}
\]
Improving the Perceptron
Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake
Non-Separable Case: Probabilistic Decision
How to get probabilistic decisions?

- **Perceptron scoring:**  
  \[ z = w \cdot f(x) \]

- If \( z = w \cdot f(x) \) very positive \( \rightarrow \) want probability going to 1

- If \( z = w \cdot f(x) \) very negative \( \rightarrow \) want probability going to 0

- **Sigmoid function**

\[
\phi(z) = \frac{1}{1 + e^{-z}}
\]
A 1D Example

\[ P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}} \]

- definitely blue
- not sure
- definitely red

Probability increases exponentially as we move away from boundary.
The Soft Max

\[ P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}} \]

Looks like \( \max_y w_y \cdot x \)
Best \( w \)?

- Maximum likelihood estimation:

\[
\begin{align*}
\max_w \quad & ll(w) = \max_w \sum_i \log P(y^{(i)}|x^{(i)}; w) \\
\text{with:} & \\
\quad & P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}} \\
\quad & P(y^{(i)} = -1|x^{(i)}; w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}
\end{align*}
\]

= Logistic Regression
Separable Case: Deterministic Decision – Many Options
Separable Case: Probabilistic Decision – Clear Preference
Multiclass Logistic Regression

- Recall Perceptron:
  - A weight vector for each class: $w_y$
  - Score (activation) of a class $y$: $w_y \cdot f(x)$
  - Prediction highest score wins $y = \arg \max_y w_y \cdot f(x)$

- How to make the scores into probabilities?

$$z_1, z_2, z_3 \rightarrow \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}, \quad \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}, \quad \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

original activations softmax activations
Best w?

- Maximum likelihood estimation:

\[
\max_w \ ll(w) = \max_w \sum_i \log P(y^{(i)}|x^{(i)}; w)
\]

with:

\[
P(y^{(i)}|x^{(i)}; w) = \frac{e^{w_y(i) \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}
\]

= Multi-Class Logistic Regression
Next Lecture

- Optimization

  - i.e., how do we solve:

\[
\max_w ll(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)
\]
Classification: Comparison

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

- Perceptrons
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate