CSE446: Perceptron
Winter 2015

Luke Zettlemoyer

Slides adapted from Dan Klein
Who needs probabilities?

• **Previously:** model data with distributions

• **Joint:** $P(X,Y)$
  – e.g. Naïve Bayes

• **Conditional:** $P(Y|X)$
  – e.g. Logistic Regression

• But wait, why probabilities?

• Lets try to be error-driven!

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>modelyear</th>
<th>make</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>4</td>
<td>97</td>
<td>75</td>
<td>2265</td>
<td>18.2</td>
<td>77</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>199</td>
<td>90</td>
<td>2648</td>
<td>15</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>121</td>
<td>110</td>
<td>2600</td>
<td>12.8</td>
<td>77</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>350</td>
<td>175</td>
<td>4100</td>
<td>13</td>
<td>73</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>198</td>
<td>95</td>
<td>3102</td>
<td>16.5</td>
<td>74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>108</td>
<td>94</td>
<td>2379</td>
<td>16.5</td>
<td>73</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>113</td>
<td>95</td>
<td>2228</td>
<td>14</td>
<td>71</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>302</td>
<td>139</td>
<td>3570</td>
<td>12.8</td>
<td>78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>120</td>
<td>79</td>
<td>2625</td>
<td>18.6</td>
<td>82</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>455</td>
<td>225</td>
<td>4425</td>
<td>10</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>107</td>
<td>86</td>
<td>2464</td>
<td>15.5</td>
<td>76</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>131</td>
<td>103</td>
<td>2830</td>
<td>15.9</td>
<td>78</td>
<td>europe</td>
</tr>
</tbody>
</table>
Generative vs. Discriminative

• Generative classifiers:
  – E.g. naïve Bayes
  – A joint probability model with evidence variables
  – Query model for causes given evidence

• Discriminative classifiers:
  – No generative 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
Linear Classifiers

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

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

• If the activation is:
  – Positive, output \textit{class 1}
  – Negative, output \textit{class 2}
Example: Spam

- Imagine 3 features (spam is “positive” class):
  - free (number of occurrences of “free”)
  - money (occurrences of “money”)
  - BIAS (intercept, always has value 1)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS</td>
<td>1</td>
</tr>
<tr>
<td>free</td>
<td>1</td>
</tr>
<tr>
<td>money</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<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>

\[ w \cdot x > 0 \rightarrow \text{SPAM!!!} \]
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 $y=-1$

\[ w \cdot x = 0 \]
Binary Perceptron Algorithm

• Start with zero weights: $w=0$
• For $t=1..T$ (T passes over data)
  – For $i=1..n$: (each training example)
    • Classify with current weights
      $$y = sign(w \cdot x^i)$$
      – $sign(x)$ is +1 if $x>0$, else -1
    • If correct (i.e., $y=y^i$), no change!
    • If wrong: update
      $$w = w + y^i x^i$$
Examples: Perceptron

• Separable Case

Examples: Perceptron

- Inseparable Case

• For $t=1..T$, $i=1..n$:
  
  - $y = \text{sign}(w \cdot x^i)$
  
  - if $y \neq y^i$
    
    $w = w + y^i x^i$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
<td>-3</td>
<td>1</td>
</tr>
</tbody>
</table>

Initial:

- $w = [0,0]$
  
  $t=1, i=1$

- $[0,0] \cdot [3,2] = 0$, $\text{sign}(0)=-1$

- $w = [0,0] + [3,2] = [3,2]$
  
  $t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$, $\text{sign}(-2)=-1$
  
  $t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$, $\text{sign}(-12)=-1$

- $w = [3,2] + [-2,-3] = [1,-1]$
  
  $t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$, $\text{sign}(1)=1$

  $t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$, $\text{sign}(-4)=-1$

  $t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$, $\text{sign}(1)=1$

Converged!!!

- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 + -x_2$

- So, at $y=0 \rightarrow x_2 = x_1$
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
    \[
    \text{activation}_w(x, y) = w_y \cdot x
    \]
  - Highest activation wins
    \[
    y^* = \arg \max_y (\text{activation}_w(x, y))
    \]

Example: \( y \) is \{1,2,3\}
- We are fitting three planes: \( w_1, w_2, w_3 \)
- Predict \( i \) when \( w_i \cdot x \) is highest
Example

“win the vote”

\[ \mathbf{w}_{SPORTS} \]

\[
\begin{array}{ll}
\text{BIAS} : & -2 \\
\text{win} : & 4 \\
\text{game} : & 4 \\
\text{vote} : & 0 \\
\text{the} : & 0 \\
\ldots & \\
\end{array}
\]

\[ x \cdot \mathbf{w}_{SPORTS} = 2 \]

\[ \mathbf{w}_{POLITICS} \]

\[
\begin{array}{ll}
\text{BIAS} : & 1 \\
\text{win} : & 2 \\
\text{game} : & 0 \\
\text{vote} : & 4 \\
\text{the} : & 0 \\
\ldots & \\
\end{array}
\]

\[ x \cdot \mathbf{w}_{POLITICS} = 7 \]

\[ \mathbf{w}_{TECH} \]

\[
\begin{array}{ll}
\text{BIAS} : & 2 \\
\text{win} : & 0 \\
\text{game} : & 2 \\
\text{vote} : & 0 \\
\text{the} : & 0 \\
\ldots & \\
\end{array}
\]

\[ x \cdot \mathbf{w}_{TECH} = 2 \]

POLITICS wins!!!
The Multi-class Perceptron Alg.

- Start with zero weights
- For \( t=1..T, \ i=1..n \) (\( T \) times over data)
  - Classify with current weights
    \[
    y = \arg \max_y w_y \cdot x^i
    \]
  - If correct (\( y=y_i \)), no change!
  - If wrong: subtract features \( x^i \) from weights for predicted class \( w_y \) and add them to weights for correct class \( w_{y_i} \)
    \[
    w_y = w_y - x^i \\
    w_{y_i} = w_{y_i} + x^i
    \]
Linearly Separable (binary case)

- The data is linearly separable with margin $\gamma$, if:
  \[ \exists w. \forall t. y^t (w \cdot x^t) \geq \gamma > 0 \]

- For $y^t=1$
  \[ w \cdot x^t \geq \gamma \]

- For $y^t=-1$
  \[ w \cdot x^t \leq -\gamma \]
Mistake Bound for Perceptron

• Assume data is separable with margin $\gamma$:

$$\exists w^* \text{ s.t. } \|w^*\|_2 = 1 \text{ and } \forall t. y^t(w^* \cdot x^t) \geq \gamma$$

• Also assume there is a number $R$ such that:

$$\forall t. \|x^t\|_2 \leq R$$

• Theorem: The number of mistakes (parameter updates) made by the perceptron is bounded:

$$\text{mistakes} \leq \frac{R^2}{\gamma^2}$$
Perceptron Convergence (by Induction)

- Let \( w^k \) be the weights after the k-th update (mistake), we will show that:
  \[
k^2 \gamma^2 \leq \|w^k\|_2^2 \leq kR^2
\]

- Therefore:
  \[
k \leq \frac{R^2}{\gamma^2}
\]

- Because \( R \) and \( \gamma \) are fixed constants that do not change as you learn, there are a finite number of updates!

- Proof does each bound separately (next two slides)
Lower bound

• Remember our margin assumption:
  \[ \exists w^* \text{ s.t. } \|w^*\|_2 = 1 \text{ and } \forall t. y^t (w^* \cdot x^t) \geq \gamma \]

• Now, by the definition of the perceptron update, for k-th mistake on t-th training example:
  \[
  w^{k+1} \cdot w^* = (w^k + y^t x^t) \cdot w^*
  = w^k \cdot w^* + y^t (w^* \cdot x^t)
  \geq w^k \cdot w^* + \gamma
  \]

• So, by induction with \( w^0 = 0 \), for all \( k \):
  \[
  k\gamma \leq w^k \cdot w^*
  \leq \|w^k\|_2
  \]
  \[
  k^2 \gamma^2 \leq \|w^k\|_2
  \]

Because:
  \[
  w^k \cdot w^* \leq \|w^k\|_2 \times \|w^*\|_2
  \]
  and \( \|w^*\|_2 = 1 \)

Perceptron update:
  \[
  w = w + y^t x^t
  \]
Upper Bound

- By the definition of the Perceptron update, for \( k \)-th mistake on \( t \)-th training example:

\[
\| w^{k+1} \|_2^2 = \| w^k + y^t x^t \|_2^2 \\
= \| w^k \|_2^2 + (y^t)^2 \| x^t \|_2^2 + 2 y^t x^t \cdot w^k \\
\leq \| w^k \|_2^2 + R^2 \\
\leq 0 \text{ because Perceptron made error (} y^t \text{ has different sign than } x^t \cdot w^t \text{)}
\]

- So, by induction with \( w_0=0 \) have, for all \( k \):

\[
\| w_k \|_2^2 \leq kR^2
\]

Perceptron update:

\[
w = w + y^t x^t
\]

Data Assumption:

\[
\forall t. \| x^t \|_2 \leq R
\]
Perceptron Convergence (by Induction)

• Let $w^k$ be the weights after the k-th update (mistake), we will show that:

$$k^2 \gamma^2 \leq \|w^k\|_2^2 \leq kR^2$$

• Therefore:

$$k \leq \frac{R^2}{\gamma^2}$$

• Because $R$ and $\gamma$ are fixed constants that do not change as you learn, there are a finite number of updates!

• If there is a linear separator, Perceptron will find it!!!

• Proof does each bound separately (last two slides)
From Logistic Regression to the Perceptron: 2 easy steps!

• Logistic Regression: (in vector notation): $y$ is $\{0, 1\}$

$$w = w + \eta \sum_{j} [y^j - P(y^j | x^j, w)] x^j$$

• Perceptron: when $y$ is $\{0, 1\}$:

$$w = w + [y^j - sign^0 (w \cdot x^j)] x^j$$

  • $sign^0(x) = +1$ if $x > 0$ and 0 otherwise

Differences?

• Drop the $\Sigma_j$ over training examples: online vs. batch learning

• Drop the dist’n: probabilistic vs. error driven learning
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

\[ \text{mistakes} \leq \frac{R^2}{\gamma^2} \]
Problems with the Perceptron

- Noise: if the data isn’t separable, weights might 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
Linear Separators

- Which of these linear separators is optimal?
Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- Support vector machines (SVMs) find the separator with max margin

\[ \min_w \frac{1}{2} \|w\|^2 \]

\[ \forall i, y \quad w_y^* \cdot x^i \geq w_y \cdot x^i + 1 \]
Three Views of Classification (more to come later in course!)

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

- Logistic Regression:
  - Parameters from gradient ascent
  - Parameters: linear, probabilistic model, and discriminative
  - Training: gradient ascent (usually batch), regularize to stop overfitting

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