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!**
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

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

- 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)

  \[ w \cdot f(x) = \sum_{i} w_i \cdot f_i(x) \]

<table>
<thead>
<tr>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS : -3</td>
</tr>
<tr>
<td>free : 4</td>
</tr>
<tr>
<td>money : 2</td>
</tr>
<tr>
<td>\ldots</td>
</tr>
</tbody>
</table>

\[ w \cdot f(x) > 0 \rightarrow \text{SPAM!!!} \]
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 $Y=-1$

\[ w \]

<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>

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

• Start with zero weights
• For each training instance \((x,y^*)\):
  – 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: update

\[
w = w + y^* f(x)
\]
Examples: Perceptron

• Separable Case

Examples: Perceptron

- Inseparable Case

From Logistic Regression to the Perceptron: 2 easy steps!

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

\[ w = w + \eta \sum_{j} [y_j^* - p(y_j^* | x_j, w)] f(x_j) \]

• Perceptron: y is \{0,1\}, y(x;w) is prediction given w

\[ w = w + [y^* - y(x;w)] f(x) \]

Differences?

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

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

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

“win the vote”

<table>
<thead>
<tr>
<th>(w_{SPORTS})</th>
<th>(w_{POLITICS})</th>
<th>(w_{TECH})</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS : -2</td>
<td>BIAS : 1</td>
<td>BIAS : 2</td>
</tr>
<tr>
<td>win : 4</td>
<td>win : 2</td>
<td>win : 0</td>
</tr>
<tr>
<td>game : 4</td>
<td>game : 0</td>
<td>game : 2</td>
</tr>
<tr>
<td>vote : 0</td>
<td>vote : 4</td>
<td>vote : 0</td>
</tr>
<tr>
<td>the : 0</td>
<td>the : 0</td>
<td>the : 0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The Multi-class Perceptron Alg.

• Start with zero weights
• Iterate training examples
  – 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
    \[ w_y = w_y - f(x) \]
    \[ w_{y^*} = w_{y^*} + f(x) \]
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

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

\[
\min_w \frac{1}{2} \|w\|^2 \\
\forall i, y \; w_y^* \cdot f(x_i) \geq w_y \cdot f(x_i) + 1
\]
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: one pass through the data per gradient step, use validation to stop

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