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
Discriminative vs. generative

- Generative model
  
  (The artist)

- Discriminative model
  
  (The lousy painter)

- Classification function

\[ \text{label} = F_{\text{Zebra}}(\text{Data}) \]
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 **class 1**
  - Negative, output **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>( \mathbf{x} )</th>
<th>( \mathbf{w} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS : 1</td>
<td>BIAS : -3</td>
</tr>
<tr>
<td>free : 1</td>
<td>free : 4</td>
</tr>
<tr>
<td>money : 1</td>
<td>money : 2</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

\[ \mathbf{w} \cdot \mathbf{x} > 0 \Rightarrow \text{SPAM!!!} \]
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
      \[ y = \text{sign}(w \cdot x^i) \]
      – \( \text{sign}(x) \) 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”

\[ x \]

| BIAS   | 1 |
| win    | 1 |
| game   | 0 |
| vote   | 1 |
| the    | 1 |
| ...    |   |

\[ x \cdot w_{SPORTS} = 2 \]

\[ x \cdot w_{POLITICS} = 7 \]

\[ x \cdot w_{TECH} = 2 \]

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:
  \[ k \gamma \leq w^k \cdot w^* \]
  \[ \leq \|w^k\|_2 \]
  \[ k^2 \gamma^2 \leq \|w^k\|_2^2 \]

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

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

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

\[
\|w^{k+1}\|_2^2 = \|w^k + y^t x^t\|_2^2 \\
= \|w^k\|_2^2 + (y^t)^2 \|x^t\|_2^2 + 2y^t x^t \cdot w^k \\
\leq \|w^k\|_2^2 + R^2
\]

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

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

• If there is a linear separator, Perceptron will find it!!!
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 $\{0,1\}$:
  \[
  w = w + [y^j - \text{sign}^0 (w \cdot x^j)] x^j
  \]
  • $\text{sign}^0(x) = +1$ 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

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

\[ \forall i, y \quad w_y \cdot x^i \geq y \cdot x^i + 1 \]
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