Introduction

- Sometimes probabilistic information unavailable or mathematically intractable
- Many alternatives to Bayesian classification, but optimality guarantee may be compromised!
- Linear classifiers use a decision hyperplane to perform classification
- Simple and efficient to train and use
- Optimality requires linear separability of classes

Linear Discriminant Functions

- Let \( w = [w_1, \ldots, w_L]^T \) be a weight vector and \( w_0 \) (a.k.a. \( \theta \)) be a threshold
- Decision surface is a hyperplane:
  \[
  w^T \cdot x + w_0 = 0
  \]
- E.g. predict \( \omega_2 \) if \( \sum_{i=1}^L w_i x_i > w_0 \), otherwise predict \( \omega_1 \)
- Focus of this lecture: How to find \( w_i \)'s
  - Perceptron algorithm
  - Winnow
  - Least squares methods (if classes not linearly separable)

The Perceptron Algorithm

- Assume linear separability, i.e. \( \exists w^* \) s.t.
  \[
  w^T \cdot x > 0 \quad \forall x \in \omega_1 \\
  w^T \cdot x \leq 0 \quad \forall x \in \omega_2
  \]
  \( w_0^* \) is included in \( w^* \)
- So \( \exists \) deterministic function classifying vectors
  (contrary to Ch. 2 assumptions)

  \[
  y(t+1) = \begin{cases} 
  \omega_1 & \text{if } \sum w_j x_j > w_0 \\
  \omega_2 & \text{otherwise}
  \end{cases}
  \]

  May also use +1 and -1
- Given actual label \( y(t) \) for trial \( t \), update weights:
  \[
  w(t+1) = w(t) + \rho (y(t) - \hat{y}(t)) x(t)
  \]
  \( \rho > 0 \) is learning rate
  \( (y(t) - \hat{y}(t)) \) moves weights toward correct prediction for \( x \)
The Perceptron Algorithm

Example

\[ x_2 \]
\[ x_1 \]
\[ (\omega_1) \]
\[ (\omega_2) \]

our dec. line

our new dec. line

opt. dec. line

The Perceptron Algorithm

Intuition

• Compromise between **correctiveness** and **conservativeness**
  
  – Correctiveness: Tendency to improve on \( x(t) \) if prediction error made
  
  – Conservativeness: Tendency to keep \( w(t+1) \) close to \( w(t) \)

• Use **cost function** that measures both:

\[
U(w) = \|w(t+1) - w(t)\|_2^2 + \eta (y(t) - w(t+1) \cdot x(t))^2
\]

\[
= \sum_{i=1}^{\ell} (w_i(t+1) - w_i(t))^2 + \eta \left( y(t) - \sum_{i=1}^{\ell} w_i(t+1) x_i(t) \right)^2
\]

The Perceptron Algorithm

Intuition (cont’d)

• Take gradient w.r.t. \( w(t+1) \) and set to 0:

\[ 0 = 2 \left( w_i(t+1) - w_i(t) \right) - 2\eta \left( y(t) - \sum_{i=1}^{\ell} w_i(t+1) x_i(t) \right) x_i(t) \]

• Approximate with

\[ 0 \approx 2 \left( w_i(t+1) - w_i(t) \right) - 2\eta \left( y(t) - \sum_{i=1}^{\ell} w_i(t) x_i(t) \right) x_i(t), \]

which yields

\[ w_i(t+1) = w_i(t) + \eta \left( y(t) - \sum_{i=1}^{\ell} w_i(t) x_i(t) \right) x_i(t) \]

• Applying threshold to **summation** yields

\[ w_i(t+1) = w_i(t) + \eta \left( y(t) - \hat{y}(t) \right) x_i(t) \]

The Perceptron Algorithm

Miscellany

• If classes linearly separable, then by cycling through vectors, guaranteed to converge in finite number of steps

• For real-valued output, can replace threshold function on sum with

  – Identity function: \( f(x) = x \)

  – Sigmoid function: e.g. \( f(x) = \frac{1}{1+\exp(-ax)} \)

  – Hyperbolic tangent: e.g. \( f(x) = c \tanh(ax) \)
Winnow/Exponentiated Gradient

- Same as Perceptron, but update weights:
  \[ w_i(t+1) = w_i(t) \exp(-2\eta(y(t) - y(t))x_i(t)) \]

- If \( y(t), \hat{y}(t) \in \{0,1\} \forall t \), then set \( \eta = (\ln \alpha)/2 \) (\( \alpha > 1 \)) and get Winnow:
  \[
  w_i(t + 1) = \begin{cases} 
  w_i(t)/\alpha^x_i(t) & \text{if } \hat{y}(t) = 1, y(t) = 0 \\
  w_i(t)\alpha^{x_i(t)} & \text{if } \hat{y}(t) = 0, y(t) = 1 \\
  w_i(t) & \text{if } \hat{y}(t) = y(t)
  \end{cases}
  \]

Intuition

- Measure distance in cost function with unnormalized relative entropy:
  \[
  U(w) = \sum_{i=1}^{\ell} \left( w_i(t) - w_i(t+1) + w_i(t+1) \ln \frac{w_i(t+1)}{w_i(t)} \right)
  \]

- Take gradient w.r.t. \( w(t+1) \) and set to 0:
  \[
  0 = \ln \frac{w_i(t+1)}{w_i(t)} - 2\eta \left( y(t) - \sum_{i=1}^{\ell} w_i(t+1)x_i(t) \right)x_i(t)
  \]

Miscellany

- Winnow and EG update wts by multiplying by a pos const: impossible to change sign
  - Weight vectors restricted to one quadrant

- Solution: Maintain wt vectors \( w^+(t) \) and \( w^-(t) \)
  - Predict \( \hat{y}(t) = (w^+(t) - w^-(t)) \cdot x(t) \)
  - Update:
    \[
    r_i^+(t) = \exp(-2\eta(\hat{y}(t) - y(t))x_i(t) U) \\
    r_i^-(t) = 1/r_i^+(t)
    \]
    \[
    w_i^+(t + 1) = U \cdot \frac{w_i^+(t) r_i^+(t)}{\sum_{j=1}^{\ell} (w_i^+(t) r_j^+(t) + w_i^-(t) r_j^-(t))}
    \]

- Winnow and EG are multiplicative weight update schemes versus additive weight update schemes, e.g. Perceptron

- Winnow and EG work well when most attributes (features) are irrelevant, i.e. optimal weight vector \( w^* \) is sparse (many 0 entries)

- E.g. \( x_i \in \{0,1\} \), \( x \)'s are labelled by a monotone \( k \)-disjunction over \( \ell \) attributes, \( k \ll \ell \)
  - Remaining \( \ell - k \) are irrelevant
  - E.g. \( x_5 \lor x_9 \lor x_{12} \), \( \ell = 150, k = 3 \)

- For disjunctions, number of on-line prediction mistakes is \( O(k \log \ell) \) for Winnow and worst-case \( \Omega(k \ell) \) for Perceptron

- So in worst case, need exponentially fewer updates for training in Winnow than Perceptron

- Other bounds exist for real-valued inputs and outputs
Non-Linearly Separable Classes

- What if no hyperplane completely separates the classes?
- Add extra inputs that are nonlinear combinations of original inputs (Section 4.14)
  - E.g. attrs. $x_1$ and $x_2$, so try $x = [x_1, x_2, x_1 x_2, x_1^2, x_2^2, x_1 x_2^3, x_2 x_1, x_1 x_1 x_2]^T$
  - Perhaps classes linearly separable in new feature space
  - Useful, especially with Winnow/EG logarithmic bounds
  - Kernel functions/SVMs
- Pocket algorithm (p. 63) guarantees convergence to a best hyperplane
- Winnow’s & EG’s agnostic results
- Least squares methods (Sec. 3.4)
- Networks of classifiers (Ch. 4)

Winnow’s Agnostic Results

- Winnow’s total number of prediction mistakes loss (in on-line setting) provably not much worse than best linear classifier
  - Loss bound related to performance of best classifier and total distance under $|| \cdot ||_1$ that feature vectors must be moved to make best classifier perfect [Littlestone, COLT ’91]
- Similar bounds for EG [Kivinen & Warmuth]

Least Squares Methods

- Recall from Slide 7:
  \[ w_i(t+1) = w_i(t) + \eta \left( y(t) - \sum_{i=1}^{t} w_i(t)x_i(t) \right) x_i(t) \]
  \[ = w_i(t) + \eta \left( y(t) - w(t)^T \cdot x(t) \right) x_i(t) \]
- If we don’t threshold dot product during training and allow $\eta$ to vary each trial (i.e. substitute $\eta_t$), get\(^*\) Eq. 3.38, p. 69:
  \[ w(t+1) = w(t) + \eta_t x(t) \left( y(t) - w(t)^T \cdot x(t) \right) \]
- This is Least Mean Squares (LMS) Algorithm
- If e.g. $\eta_t = 1/t$, then
  \[ \lim_{t \to \infty} P \left( w(t) = w^* \right) = 1, \]
  where
  \[ w^* = \arg\min_{w \in \mathbb{R}^d} \left\{ E \left[ || y - w^T \cdot x ||^2 \right] \right\} \]
  is vector minimizing mean square error (MSE)
\(^*\)Note that here $w(t)$ is weight before trial $t$. In book it is weight after trial $t$. 

Multiclass learning

Kessler’s Construction

- For\(^*\) $x = [2, 2, 1]^T$ of class $\omega_1$, want
  \[ \sum_{i=1}^{\ell+1} w_{1_i} x_i > \sum_{i=1}^{\ell+1} w_{2_i} x_i \quad \text{AND} \quad \sum_{i=1}^{\ell+1} w_{1_i} x_i > \sum_{i=1}^{\ell+1} w_{3_i} x_i \]
\(^*\)The extra 1 is added so threshold can be placed in $w$. 

13

14
Multiclass learning
Kessler’s Construction (cont’d)

• So map $x$ to
  $x_1 = \begin{bmatrix} 2, 2, 1, -2, -2, -1, 0, 0, 0 \end{bmatrix}^T$
  $x_2 = \begin{bmatrix} 2, 2, 1, 0, 0, 0, -2, -2, -1 \end{bmatrix}^T$
  (all labels = +1) and let
  $w = [w_{11}, w_{12}, w_{10}, w_{21}, w_{22}, w_{20}, w_{31}, w_{32}, w_{30}]^T$

• Now if $w^T \cdot x_1 > 0$ and $w^T \cdot x_2 > 0$, then
  $\ell + 1 \sum_{i=1}^{\ell+1} w_{1i}x_i > \ell + 1 \sum_{i=1}^{\ell+1} w_{2i}x_i$ AND
  $\ell + 1 \sum_{i=1}^{\ell+1} w_{1i}x_i > \sum_{i=1}^{\ell+1} w_{3i}x_i$

• In general, map $(\ell + 1) \times 1$ feature vector $x$ to
  $x_1, \ldots, x_{M-1}$, each of size $(\ell + 1)M \times 1$

• $x \in \omega_i \Rightarrow x$ in $i$th block and $-x$ in $j$th block, 
  ($\text{rest are 0s}$). Repeat for all $j \neq i$

• Now train to find weights for new vector space 
  via perceptron, Winnow, etc.

Error-Correcting Output Codes (ECOC)

• Since Win. & Percep. learn binary functions, learn individual bits of binary encoding of classes

• E.g. $M = 4$, so use two linear classifiers:

<table>
<thead>
<tr>
<th>Class</th>
<th>Binary Encoding</th>
<th>Classifier 1</th>
<th>Classifier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_1$</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$\omega_2$</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\omega_3$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$\omega_4$</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

and train simultaneously

• Problem: Sensitive to individual classifier errors, so use a set of encodings per class to improve robustness

• Similar to principle of error-correcting output codes used in communication networks 
  [Dietterich & Bakiri, 1995]

• General-purpose, independent of learner

Topic summary due in 1 week!