Linear Discriminants

- \( x = (x_1, ..., x_d) \) = vector of attributes (features)
- \( w = (w_1, ..., w_d) \) = weight vector
- \( w \cdot x = \sum_{i}^{d} w_ix_i \)
  - \( = wx^t \)
  - (or \( w^tx \) if the vectors are column vectors, as in Duda, Hart & Stork)

**Linear discriminant** is a function of the form:

- \( g(x) = w \cdot x + w_0 = \sum_{i}^{d} w_ix_i + w_0 \)

- \( w \) is normal to the **hyperplane** \( H \) defined by:
  - \( H = \{x: g(x)=0\} \)
  - **Proof:**
    - \( x_1, x_2 \) in \( H \) \( \Rightarrow \) \( w \cdot (x_1-x_2) = g(x_1)-w_0-(g(x_2)-w_0) = 0 \)
**Hyperplanes**

Use $g$ as a classifier: $x$ is classified $+1$ if $g(x) > 0$ i.e. $\sum_{i}^{d} w_i x_i > -w_0$

- $-1$ if $g(x) < 0$ i.e. $\sum_{i}^{d} w_i x_i < -w_0$

Thus the classifier is $\text{Sign}(g(x))$

Extend from binary case (2 classes) to multiple classes later.

Distance of $H$ from origin

$= \frac{|w_0|}{\|w\|}$

$w_0$ is called "bias" (confusing!) or "threshold weight"
**Threshold is an Extra Weight**

- $w_0$ can be incorporated into $w$ by setting $x_0 = 1$
  - Then $g(x) = w \cdot x = \sum_{i} w_i x_i$

- Example:
  - $d=1$, "hyperplane" is just a point separating +ve from -ve points
  - Embed the points into $d=2$ space by making their first component 1
  - "hyperplane" passes through origin

\[ x_0 \quad x_1 \]

- Origin

- "hyperplane"

\[ 1 \quad x_0 \]

- Origin

- "hyperplane"
Perceptron

- Input is \( x = (1, x_1, ..., x_d) \)
- Weight vector = \((w_0, w_1, ..., w_d)\)
- Output is \( g(x) = w \cdot x = \sum_{i=0}^{d} w_i x_i \)
- Classify using \( \text{Sign}(g(x)) \)
**Linearly Separable**

- Given: Training Sample $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where:
  - each $x_i$ is a vector $x_i = (x_{i1}, \ldots, x_{id})$
  - each $y_i = \pm 1$
- $T$ is **linearly separable** if there is a hyperplane separating the $+ve$ points from the $-ve$ points i.e. there exists $w$ such that
  - $g(x_i) = w \cdot x_i > 0$ if and only if $y_i = +1$
  - i.e. $y_i(w \cdot x_i) > 0$ for all $i$
Finding a Weight Vector

- Hypothesis Space = all weight vectors (with $d+1$ coordinates)
- If $w$ separates +ve from -ve points, so does $aw$ for any $a > 0$.
- How do we find a "good" weight vector $w$?
- Could try "all" $w$ until find one
  - e.g. try all integer-valued $w$ in order of increasing length.
- Better idea
  - repeatedly change $w$ to correct the points it classifies incorrectly.

$g(x)$ must be decreased

$g(x)$ must be increased
Updating the Weights

- If a +ve point $x_r$ is incorrectly classified i.e. $w \cdot x_r < 0$, then:
  - INCREASE $w \cdot x_r$ by:
    - increasing $w_i$ if $x_{ri} > 0$
    - decreasing $w_i$ if $x_{ri} < 0$
- If a -ve point $x_r$ is incorrectly classified i.e. $w \cdot x_r > 0$, then:
  - DECREASE $w \cdot x_r$ by:
    - increasing $w_i$ if $x_{ri} < 0$
    - decreasing $w_i$ if $x_{ri} > 0$
- For both +ve and -ve points, do:
  - $w \leftarrow w + y_r x_r$ if $x_r$ is incorrectly classified
- NOTE: A point $x_r$ is classified correctly if and only if
  - $y_r(w \cdot x_r) > 0$
- Notational "trick" used by some texts:
  - multiply -ve points by -1
  - can express formulas more simply (without $y_i$)
**Perceptron Algorithm**

- Algorithm:
  - $w=0$
  - Repeat until all points $x_i$ are correctly classified
    - If $x_r$ is incorrectly classified, do $w \leftarrow w + y_r x_r$
  - Output $w$

1st component of $w$ increases
2nd component of $w$ decreases
new $w$ is not guaranteed to be closer to $w^*$ than $w$ was, but will be closer to some multiple of $w^*$
**Perceptron Convergence Proof**

- Proposition: If the training set is linearly separable, the perceptron algorithm converges to a solution vector in a finite number of steps.

**Proof**
- Let \(w^*\) be some solution vector i.e. \(y_i(w^* \cdot x_i) > 0\) for all \(i\) (Eqn 1)
  - \(w^*\) exists because the sample is linearly separable
  - \(\alpha w^*\) is a solution vector for any \(\alpha > 0\).

- The update is:
  - \(w(k+1) = w(k) + y_r x_r\) if \(x_r\) is misclassified

- We want to show that:
  - \(|w(k+1) - aw^*|^2 \leq |w(k) - aw^*|^2 - c\) for some constant \(c > 0\)

- We have:
  - \(|w(k+1) - aw^*|^2 = |w(k) - aw^* + y_r x_r|^2 = (w(k) - aw^* + y_r x_r) \cdot (w(k) - aw^* + y_r x_r)\)
    \[= |w(k) - aw^*|^2 + 2(w(k) - aw^*) \cdot y_r x_r + |y_r x_r|^2\]

- Since \(w(k) \cdot y_r x_r < 0\) because \(x_r\) was misclassified, we have:
  - \(|w(k+1) - aw^*|^2 \leq |w(k) - aw^*|^2 - 2\alpha y_r (w^* \cdot x_r) + |y_r x_r|^2\) (Eqn 2)
Convergence Proof (contd)

- Since $y_r(w^* \cdot x_r) > 0$ from (Eqn 1), our goal is:
  
  pick $\alpha$ so large that $-2\alpha y_r(w^* \cdot x_r) + |y_r x_r|^2 < -c$ for some constant $c>0$

- Let $\beta = \max_i |y_i x_i| = \max_i |x_i|$
  
  $\gamma = \min_i y_r(w^* \cdot x_r) > 0$ (Neither $\beta$ nor $\gamma$ depend on $k$!)

Then:

- $-2\alpha y_r(w^* \cdot x_r) + |y_r x_r|^2 < -2\alpha \gamma + \beta^2$

Pick $\alpha = \beta^2 / \gamma$

Then:

- $-2\alpha y_r(w^* \cdot x_r) + |y_r x_r|^2 < -2\beta^2 + \beta^2 = -\beta^2$
  
  and so, from (Eqn 2)

- $|w(k+1) - \alpha w^*|^2 \leq |w(k) - \alpha w^*|^2 - \beta^2$

Since squared distances are never negative, this decrease must eventually stop;
  
  i.e. the update rule "$w(k+1) = w(k) + y_r x_r$ if $x_r$ is misclassified" stops changing $w$ -
  
  at that point $w(k) = \alpha w^*$ for some $\alpha$ and so $w(k)$ separates the training points.

- Same proof (different notation!) as in Duda, Hart & Stork, pg 230-232
Perceptron Algorithm (Details)

- Implementing the algorithm in practice:
  - Need to cycle through examples multiple times
  - Update \( w \) after each cycle, not every example ("batch perceptron")
- **Learning Rate** \( \eta \)
  - \( w \leftarrow w + \eta y_r x_r \)
  - small \( \eta \) gives slow convergence
  - large \( \eta \) may cause overshoot
  - \( \eta \) can be updated each iteration, want \( \eta = \eta(k) \rightarrow 0 \) as iteration \( k \rightarrow \infty \)
    - \( \eta(k) = \eta(1) / k \)
    - Decrease \( \eta(k) \) if performance improves on \( k^{th} \) step
Non Linearly-Separable

- Perceptron Algorithm does not converge if training set is not linearly separable
  - Cannot learn X-OR or any non-linearly separable concept.
  - Pointed out by Minsky & Papert (1969) - set back research for many years
- Linearly Separable training sample ⇔ underlying concept is linearly separable
  - As d, the number of dimensions, increases, random training set is increasingly likely to be linearly separable
- In practice, try get low error if not lin sep.
- Heuristics:
  - Terminate when (length of) w stops fluctuating
  - Average recent w's
  - Choice of learning rate

X-OR

Non Linearly-Separable

+ -

X-OR

Points pointed out by Minsky & Papert (1969) set back research for many years.
Gradient Descent

- Suppose $J$ is some function of the weight $w$ which we want to minimize.
- Gradient Descent searches iteratively for this minimum by moving from the current choice of $w$ in the direction of $J$'s steepest descent:
  \[ w \leftarrow w - \eta \nabla J(w), \]
  where $\nabla J$ is the vector $(\partial J/\partial w_0, \partial J/\partial w_1, \ldots, \partial J/\partial w_d)$
  - Terminate when $|\eta \nabla J(w)|$ is sufficiently small
- Example: $J(w) = -\sum_M y_i(w \cdot x_i)$
  where the sum is ONLY over the set $M$ of $x_i$ misclassified by this hyperplane
  - $y_i(w \cdot x_i) < 0$ if $x_i$ is misclassified, so $J(w) \geq 0$, we would like to minimize $J$.
- Since $y_i(w \cdot x_i) = y_i(w_i x_{i1} + \ldots + w_d x_{id})$,
  $\partial J/\partial w_r = -\sum_M y_i x_{ir}$
  $\nabla J = -\sum_M y_i x_i$
  and gradient descent becomes:
  \[ w \leftarrow w + \eta \sum_M y_i x_i \] ("batch perceptron")
- Thus Perceptron Algorithm does gradient descent search in weight space.
Least-Mean-Squared

- $J(w) = \text{Squared Error}(w) = 0.5 \sum_i^n (y_i - (w \cdot x_i))^2$
- Since $y_i - (w \cdot x_i) = y_i - (w_1 x_{i1} + \ldots + w_d x_{id})$,
  \[ \frac{\partial J}{\partial w_r} = 0.5 \sum_i^n 2(y_i - (w \cdot x_i))(-x_{ir}) \]
  \[ \nabla J = -\sum_i^n (y_i - (w \cdot x_i))x_i \]
  \[ w \leftarrow w + \eta \sum_i^n (y_i - (w \cdot x_i))x_i \]
- For faster convergence, consider the samples one-by-one:
  \[ w \leftarrow w + \eta (y_i - (w \cdot x_i))x_i \]
  - the LMS (or Delta or Widrow-Hoff) learning rule.
  - same algorithm (different notation!) as Duda, Hart and Stork, pg 246.
  - basis of backpropagation algorithm for training neural networks.
- LMS rule converges asymptotically to the weight vector yielding minimum squared error whether or not the training sample is linearly separable.
- However, minimizing the error does NOT necessarily minimize the number of misclassified examples.
Multiple Classes

- Suppose there are \( n \) classes \( c_1, \ldots, c_n \)
  
  (1) 1 vs rest
  - Use 1 linear discriminant for each class \( c_i \), where points in \( c_i \) are +ve, all points not in \( c_i \) are -ve.
  - Need \( n \) linear discriminants
  - Assign ambiguous elements to nearest class

  (2) pairwise
  - Use 1 linear discriminant for each pair of classes
  - Need \( n(n-1)/2 \) linear discriminants
  - Assign points to class that gets most votes
  - Assign ambiguous elements to nearest class

  (3) linear machine
  - Use \( g_i(x) = w_i^T x + w_{i0} \) for \( i=1 \) to \( n \); Assign \( x \) to \( c_i \) if \( g_i(x) > g_j(x) \) for all \( j \neq i \)
  - Need \( n \) linear discriminants
  - No ambiguous elements
Multiple classes (1 vs rest)

Use $n$ linear discriminants for $n$ classes

Ambiguous region (?) - use distance to nearest class
Multiple class (pairwise)

Use \( \frac{n(n-1)}{2} \) linear discriminants for \( n \) classes.

Ambiguous region (?) - use distance to nearest class.
Linear Machine

- Define $n$ linear discriminants:
  - $g_i(x) = w_i^t x + w_{i0}$ \(i=1\) to \(n\)
  - Note typo in Duda Hart and Stork, pg 218! ($g_i(x) = w^t x + w_{i0}$)
- Assign $x$ to class with largest value:
  - $x$ belongs to $c_i$ if $g_i(x) > g_j(x)$ for all $j \neq i$
- Divides space into $n$ regions, where each $g_i$ is largest
  - Regions are convex and single connected
  - No ambiguous region
- The boundary between any 2 contiguous regions is a hyperplane:
  - $H_{ij} = \{x: g_i(x) = g_j(x)\} = \{x: (w_i-w_j)x^t + w_{i0}-w_{j0} = 0\}$
  - Thus differences between weight vectors are normal to the boundaries
  - May not have all $n(n-1)/2$ boundaries
- How does the definition of linearly separable generalize to multiple classes? (See Homework)
Multiple classes (Linear machine)

Use $n$ linear discriminants for $n$ classes
No ambiguous region