Announcements

- HW1 due today
- Most of last lecture was on the blackboard.
- Gather hand data set today.

Linear Discriminant Functions (Sections 5.1-5.2)

Linear discriminant functions and decisions surfaces

- Definition

  It is a function that is a linear combination of the components of x

  $g(x) = w^t x + w_0$ (1)

  where w is the weight vector and w_0 the bias

- A two-category classifier with a discriminant function of the form (1) uses the following rule:

  Decide ω_1 if $g(x) > 0$ and ω_2 if $g(x) < 0$

  $\Leftrightarrow$ Decide ω_1 if $w^t x > -w_0$ and ω_2 otherwise

  If $g(x) = 0 \Rightarrow x$ is assigned to either class

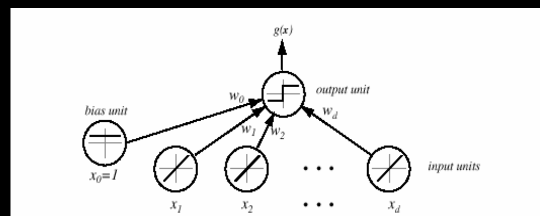


FIGURE 5.1. A simple linear classifier having d input units, each corresponding to the values of the components of an input vector. Each input feature value x_i is multiplied by its corresponding weight w_i; the effective input at the output unit is the sum all these products, $\sum w_i x_i$. We show in each unit its effective input-output function. Thus each of the d input units is linear, emitting exactly the value of its corresponding feature value. The single bias unit unit always emits the constant value 1.0. The single output unit emits a +1 if $\mathbf{w}^t\mathbf{x} + w_0 > 0$ or a −1 otherwise. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- The equation $g(x) = 0$ defines the decision surface that separates points assigned to the category ω_1 from points assigned to the category ω_2
- When $g(x)$ is linear, the decision surface is a hyperplane
- Algebraic measure of the distance from x to the hyperplane (interesting result!)

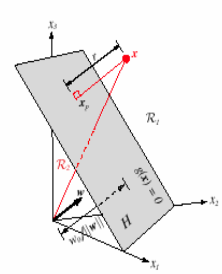


FIGURE 5.2. The linear decision boundary H, where $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x}+w_0 = 0$, separates the feature space into two half-spaces $\mathcal{R}_1$ (where $g(\mathbf{x}) > 0$) and $\mathcal{R}_2$ (where $g(\mathbf{x}) < 0$). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

$$x = x_p + \frac{r.w}{\|w\|} \quad (\text{since } w \text{ is colinear with } x - x_p \text{ and } \frac{w}{\|w\|} = 1)$$

$$\text{since } g(x) = 0 \text{ and } w^t.w = \|w\|^2$$

$$\text{therefore } r = \frac{g(x)}{\|w\|}$$

$$\text{in particular } d(0, H) = \frac{w_0}{\|w\|}$$

- In conclusion, a linear discriminant function divides the feature space by a hyperplane decision surface
- The orientation of the surface is determined by the normal vector w and the location of the surface is determined by the bias

- **The multi-category case**
  - We define c linear discriminant functions

$$g_i(x) = w_i^t x + w_{i0} \qquad i = 1,\ldots,c$$

  and assign x to ω_i if $g_i(x) > g_j(x)$ $\forall j \neq i$; in case of ties, the classification is undefined
  - In this case, the classifier is a "linear machine"
  - A linear machine divides the feature space into c decision regions, with $g_i(x)$ being the largest discriminant if x is in the region R_i
  - For a two contiguous regions $\mathcal{R}_i$ and $\mathcal{R}_j$; the boundary that separates them is a portion of hyperplane H_{ij} defined by:

    $g_i(x) = g_j(x)$

    $\Leftrightarrow (w_i - w_j)^t x + (w_{i0} - w_{j0}) = 0$
  - $w_i - w_j$ is normal to H_{ij} and

$$d(x, H_{ij}) = \frac{g_i - g_j}{\|w_i - w_j\|}$$

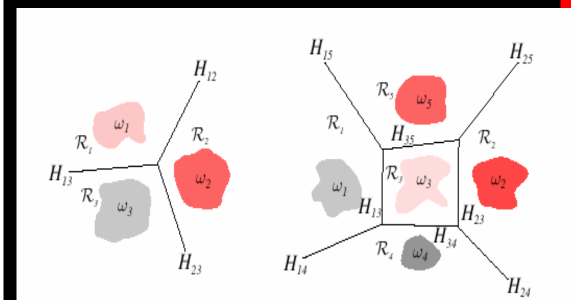
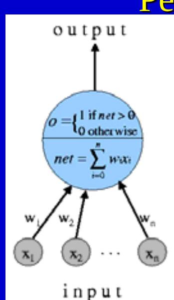


FIGURE 5.4. Decision boundaries produced by a linear machine for a three-class problem and a five-class problem. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- It is easy to show that the decision regions for a linear machine are convex, this restriction limits the flexibility and accuracy of the classifier

Perceptron



Linear, threshold units

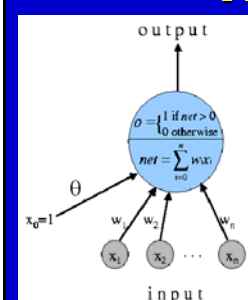
$$o(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } w_1x_1 + \cdots + w_nx_n > \theta \\ -1 & \text{otherwise.} \end{cases}$$

X_i : inputs

W_i : weights

θ : threshold

Perceptron



The threshold can be easily forced to 0 by introducing an additional weight input $W_0 = \theta$

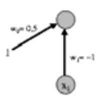
$$o(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + \cdots + w_nx_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

How powerful is a perceptron?

Threshold = 0

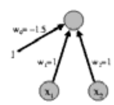
Inverter

input x1	output
0	1
1	0



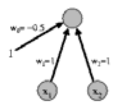
Boolean AND

input x1	input x2	ouput
0	0	0
0	1	0
1	0	0
1	1	1



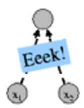
Boolean OR

input x1	input x2	ouput
0	0	0
0	1	1
1	0	1
1	1	1

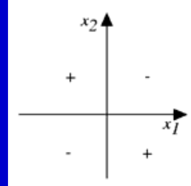


Boolean XOR

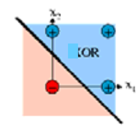
input x1	input x2	ouput
0	0	0
0	1	1
1	0	1
1	1	0



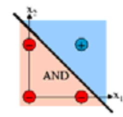
Concept Space & Linear Separability



Linear Separability



Linear Separability



Linear Separability

Training Perceptron

Perceptron Training Rule

new weight, old weight, increment

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta (t - o) x_i$$

increment, step size, target, perceptron output, input

Where:

- t is target value
- o is perceptron output
- η is small constant (e.g., .1) called *learning rate*

Converges, if…

- … training data linearly separable
- … step size η sufficiently small
- … no "hidden" units

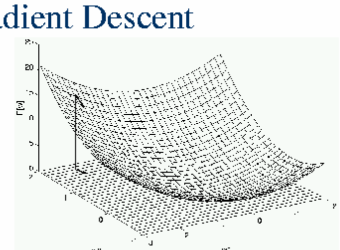
Gradient Descent

- Learn w_i's that minimize squared error

$$E[\vec{w}] = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

D = training data

Gradient Descent



Gradient: $\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]$

Training rule: $\Delta \vec{w} = -\eta \, \nabla E[\vec{w}]$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradient Descent

- To find the best direction in the feature space we compute the gradient of E with respect to each of the components of $\vec{w}$

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

- This vector specifies the direction the produces the steepest increase in E;
- We want to modify $\vec{w}$ in the direction of $-\nabla E(\vec{w})$

$$\vec{w} = \vec{w} + \Delta \vec{w}$$

- Where:

$$\Delta \vec{w} = -R \, \nabla E(\vec{w})$$

Batch Learning

- Initialize each w_i to small random value
- Repeat until termination:

$\Delta w_i = 0$

For each training example d do

$o_d \leftarrow \sigma(\Sigma_i\, w_i\, x_{i,d})$

$\Delta w_i \leftarrow \Delta w_i + \eta\, (t_d - o_d)\, o_d\, (1\text{-}o_d)\, x_{i,d}$

$w_i \leftarrow w_i + \Delta w_i$

Incremental (Online) Learning

- Initialize each w_i to small random value
- Repeat until termination:

For each training example d do

$\Delta w_i = 0$

$o_d \leftarrow \Sigma_i\, w_i\, x_{i,d}$

$\Delta w_i \leftarrow \Delta w_i + \eta\, (t_d - o_d)\, o_d\, (1\text{-}o_d)\, x_{i,d}$

$w_i \leftarrow w_i + \Delta w_i$

Summary: Single Layer Networks

- Variety of update rules
  - Multiplicative
  - Additive

$$\Delta \mathbf{w}_i = \mathbf{R}(\mathbf{t}_d - \mathbf{o}_d)\mathbf{x}_{id}$$

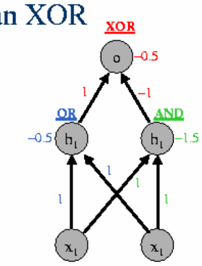
- Batch and incremental algorithms
- Various convergence and efficiency conditions
- There are other ways to learn linear functions
  - Linear Programming (general purpose)
  - Probabilistic Classifiers (some assumption)

- Although simple and restrictive -- linear predictors perform very well on many realistic problems
- However, the representational restriction is limiting in many applications

Increasing Expressiveness: Multi-Layer Neural Networks

Boolean XOR

input x1	input x2	ouput
0	0	0
0	1	1
1	0	1
1	1	0



2-layer Neural Net