

# Linear Models for Classification

## Discriminant Functions

Sargur N. Srihari

University at Buffalo, State University of New York  
USA

# Topics

- Linear Discriminant Functions
  - Definition (2-class), Geometry
  - Generalization to  $K > 2$  classes
  - Distributed representation
- Methods to learn parameters
  1. Least Squares Classification
  2. Fisher's Linear Discriminant
  3. The Perceptron Algorithm

# Discriminant Functions

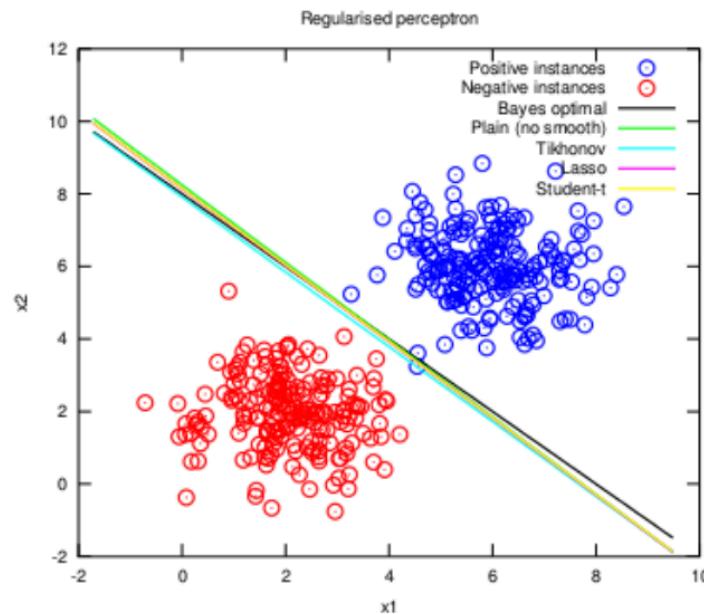
- A discriminant function assigns input vector  $\mathbf{x} = [x_1, \dots, x_D]^T$  to one of  $K$  classes denoted by  $C_k$
- We restrict attention to linear discriminants
  - i.e., Decision surfaces are hyperplanes
  - Each surface represented by equation

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

$$\mathbf{w} = [w_1, \dots, w_D]^T$$

- First consider  $K = 2$ , and then extend to  $K > 2$

# Linear discriminant examples

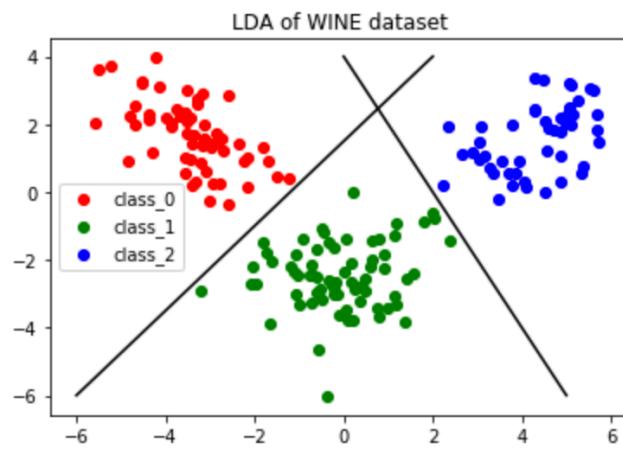


## Regularization methods

$$\text{Tikhonov: } \lambda \sum_{\forall i} w_i^2$$

$$\text{Lasso: } \lambda \sum_{\forall i} |w_i|$$

$$\text{Student-}t: \lambda \sum_{\forall i} \log(1+w_i^2)$$



# What does value of $y(\mathbf{x})$ tell you?

- 2-class case:

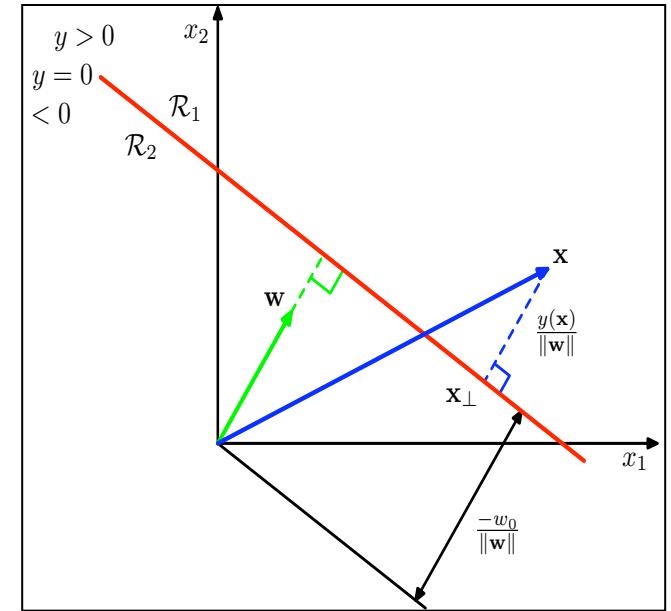
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

A linear function of input vector

$\mathbf{w}$  is weight vector and  $w_0$  is bias

Negative of bias sometimes called  
*threshold*

- Three cases:  $y > 0$ ,  $y = 0$ ,  $y < 0$ 
  - Assign  $x$  to  $C_1$  if  $y(\mathbf{x}) \geq 0$  else  $C_2$
- Defines decision boundary  $y(\mathbf{x}) = 0$ 
  - It corresponds to a  $(D-1)$ -dimensional hyperplane in a  $D$ -dimensional input space



# Distance of Origin to Surface is $w_0$

Let  $x_A$  and  $x_B$  be points on surface  $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = 0$

- Because  $y(x_A) = y(x_B) = 0$ , we have  $\mathbf{w}^T(x_A - x_B) = 0$ ,
- Thus  $\mathbf{w}$  is orthogonal to every vector lying on decision surface
- So  $\mathbf{w}$  determines orientation of the decision surface

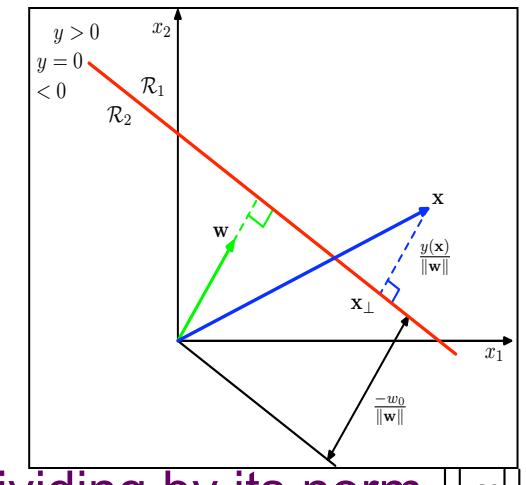
– If  $\mathbf{x}$  is a point on surface then  $y(\mathbf{x}) = 0$  or  $\mathbf{w}^T \mathbf{x} = -w_0$

- Normalized distance from origin to surface:

$$\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

where  $\|\mathbf{w}\|$  is the norm defined as

$$\|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w} = w_1^2 + \dots + w_{M-1}^2$$



- Where elements of  $\mathbf{w}$  are normalized by dividing by its norm  $\|\mathbf{w}\|$ 
  - » By definition of a normalized vector which has length 1
- $w_0$  sets distance of origin to surface

# Distance of $x$ to surface is $y(x)/\|w\|$

Let  $x$  be an arbitrary point

- We can show that  $y(x)$  gives signed measure of perpendicular distance  $r$  from  $x$  to surface as follows:
  - If  $x_p$  is orthogonal projection of  $x$  to surface then

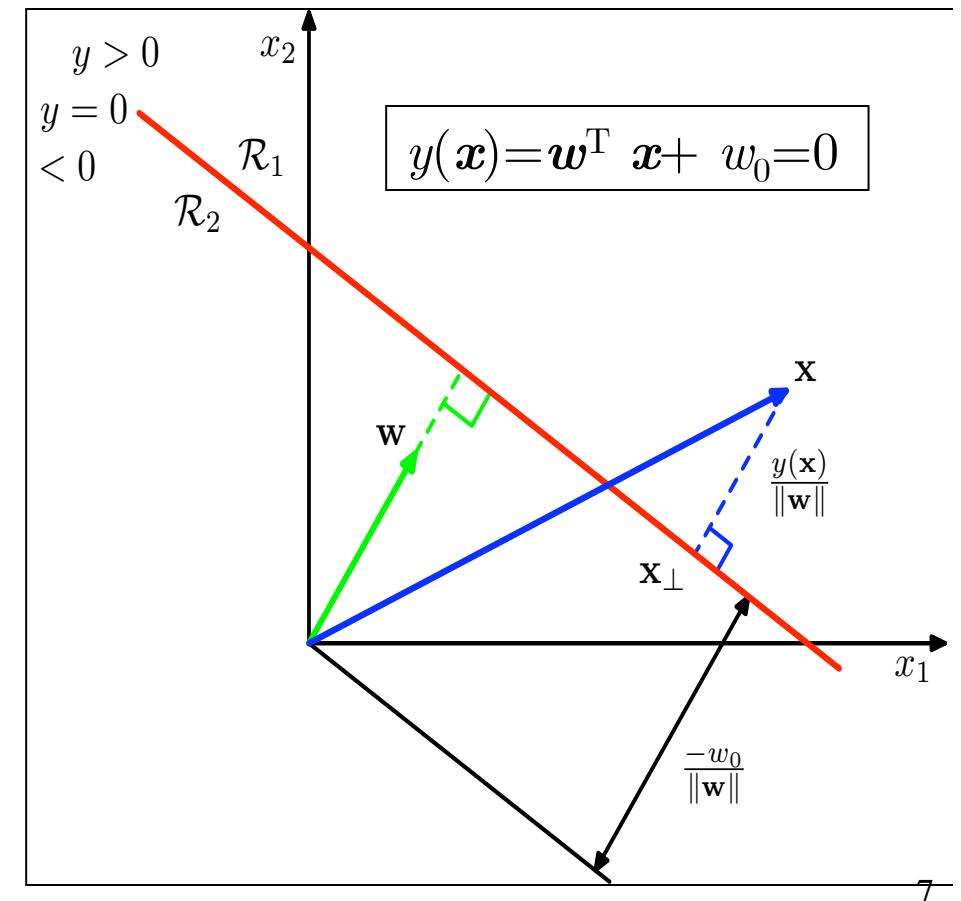
$$x = x_p + r \frac{w}{\|w\|} \text{ by vector addition}$$

Second term is a vector normal to surface.

This vector is parallel to  $w$   
which is normalized by length  $\|w\|$ .  
Since a normalized vector has length 1  
we need to scale by  $r$ .

From which we can get

$$r = \frac{y(x)}{\|w\|}$$



# Augmented vector

- We have  $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$   
where  $\mathbf{x} = [x_1, \dots, x_D]^T$   $\mathbf{w} = [w_1, \dots, w_D]^T$
- With dummy input  $x_0 = 1$ 
  - We have  $\tilde{\mathbf{w}} = (w_0, \mathbf{w})$   $\tilde{\mathbf{x}} = [1, \mathbf{x}]$  and  $y(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$
  - It passes through origin in *augmented*  $D+1$  dimensional space
    - Since the bias term is zero in this space
  - Example:
    - Data point  $\mathbf{x} = [0.2, 0.3, 0.4, 0.5]$  is augmented to  $[1.0, 0.2, 0.3, 0.4, 0.5]$
- We can discard bias term with augmentation

# Extension to Multiple Classes

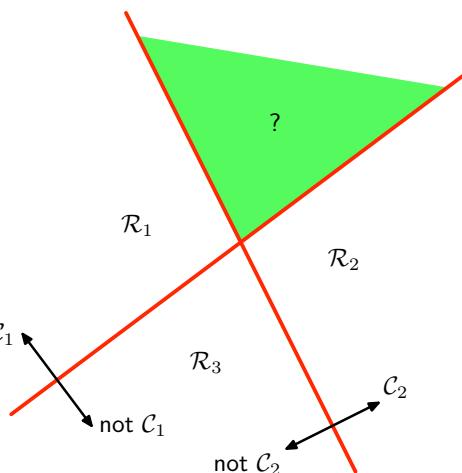
- Two approaches:
  - Using several two-class classifiers
    - But leads to serious difficulties
  - Use  $K$  linear functions

# Multiple Classes with 2-class classifiers

- By using several 2-class classifiers

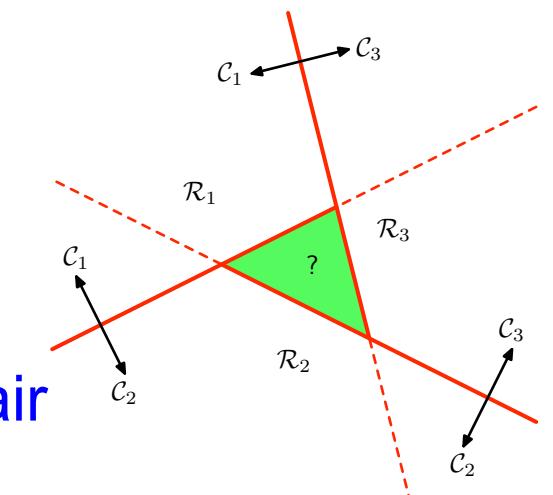
## One-versus-the-rest

Build a  $K$  class discriminant  
Use  $K - 1$  classifiers,  
each solve a two-class problem



## One-versus-one

Alternative is  $K(K - 1)/2$  binary discriminant functions, one for every pair



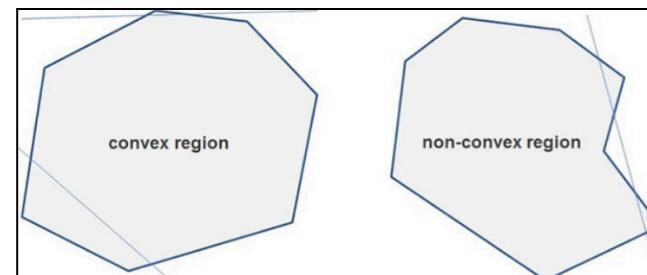
Both result in ambiguous regions of input space

# Multiple Classes with $K$ discriminants

- Consider a single  $K$  class discriminant of the form

$$y_k(\mathbf{x}) = \mathbf{w}^T_k \mathbf{x} + w_{k0}$$

- Assign a point  $\mathbf{x}$  to class  $C_k$  if  $y_k(\mathbf{x}) > y_j(\mathbf{x})$  for all  $j \neq k$
- Decision boundary between  $C_k$  and  $C_j$  is given by  $y_k(\mathbf{x}) = y_j(\mathbf{x})$ 
  - This corresponds to  $D - 1$  dimensional hyperplane defined by
  - $(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$
  - Same form as the decision boundary for 2-class case  $\mathbf{w}^T \mathbf{x} + w_0 = 0$
- Decision regions of such discriminants are always singly connected and convex
  - Proof follows



# Convexity of Decision Regions (Proof)

Consider two points  $x_A$  and  $x_B$  both in decision region  $\mathcal{R}_k$

Any point  $\hat{x}$  on line connecting  $x_A$  and  $x_B$  can be expressed as

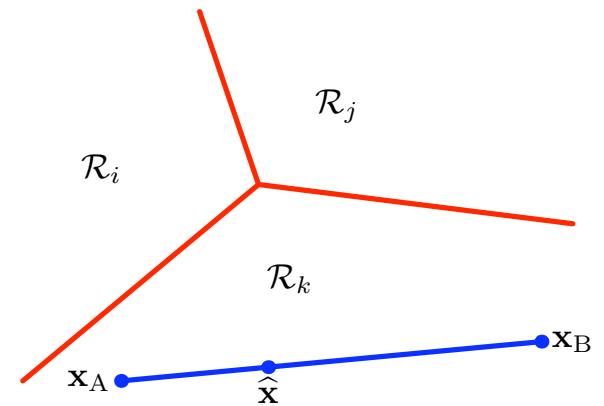
$$\hat{x} = \lambda x_A + (1 - \lambda)x_B \quad \text{where } 0 \leq \lambda \leq 1.$$

From linearity of discriminant functions

$$y_k(x) = \mathbf{w}^T_k \mathbf{x} + w_{k0}$$

Combining the two, we have

$$\hat{x} = \lambda x_A + (1 - \lambda)x_B$$



Because  $x_A$  and  $x_B$  lie inside  $R_k$  it follows that

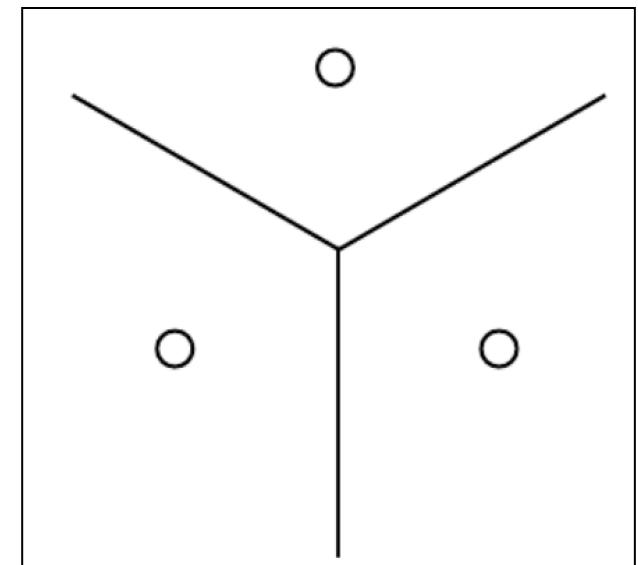
$$y_k(x_A) > y_j(x_A) \text{ and } y_k(x_B) > y_j(x_B) \text{ for all } j \neq k.$$

Hence  $\hat{x}$  also lies inside  $R_k$

Thus  $R_k$  is singly-connected and convex  
(single straight line connects any two points in region)

# No. of examples and no. of regions

- $K$  discriminants need  $O(K)$  examples:
- Nearest-neighbor : each training sample (circle) defines at most one region
  - $y$  value associated with each example defines the output for all points within region

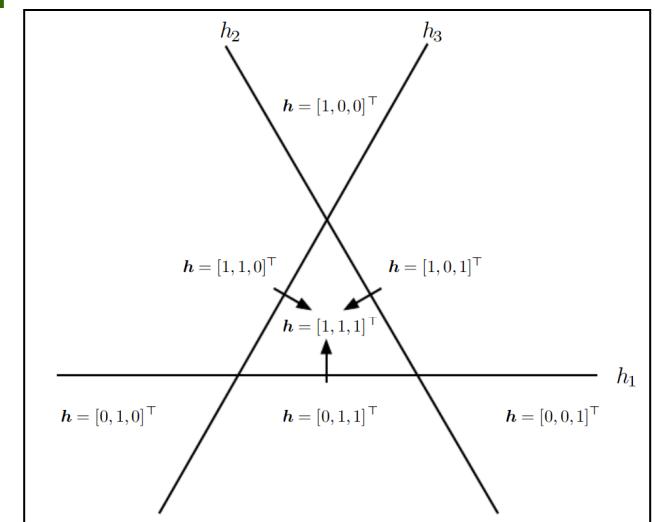


# More regions than examples

- Suppose we need more regions than examples
- Two questions of interest
  1. Is it possible represent a complicated function efficiently?
  2. Is it possible for the estimated function to generalize well for new inputs?
- Answer to both is yes
  - $O(2^K)$  regions can be defined with  $O(K)$  examples
    - By introducing dependencies between regions through assumptions on data generating distribution

# Core idea of deep learning

- Assume data was generated by composition of factors, at multiple levels in a hierarchy
  - Many other similarly generic assumptions
- These mild assumptions allow exponential gain in no of samples and no of regions
  - An example of a distributed representation is a vector of  $n$  binary features
    - It can take  $2^n$  configurations
      - Whereas in a symbolic representation, each input is associated with a single symbol (or category)
      - Here  $h_1$ ,  $h_2$  and  $h_3$  are three binary features



# Learning the Parameters of Linear Discriminant Functions

- Three Methods
  - Least Squares
  - Fisher's Linear Discriminant
  - Perceptrons
- Each is simple but several disadvantages

# Least Squares for Classification

- Analogous to regression: simple closed-form solution exists for parameters
- Each  $C_k, k = 1,..K$  is described by its own linear model

$$y_k(\mathbf{x}) = \mathbf{w}^T_k \mathbf{x} + w_{k0}$$

Note:  $\mathbf{x}$  and  $\mathbf{w}$  have  $D$  dimensions each

- Create augmented vector
  - replace  $\mathbf{x}$  by  $(1, \mathbf{x}^T)$  and  $\mathbf{w}_k$  by  $(w_{k0}, \mathbf{w}_k^T)$
- Grouping  $y_k$ s into a  $K \times 1$  vector:  $\mathbf{y}(\mathbf{x}) = \mathbf{W}^T \mathbf{x}$ 

$\mathbf{W}^T$  is the *parameter matrix* whose  $k^{\text{th}}$  column is a  $D+1$  dimensional vector  $\mathbf{w}_k$  (including bias). It is  $K \times (D+1)$ .
- New input vector  $\mathbf{x}$  is assigned to class for which output  $y_k = \mathbf{w}^T_k \mathbf{x}$  is largest
- Determine  $\mathbf{W}$  by minimizing squared error

Now  $\mathbf{x}$  and  $\mathbf{w}$  are  $(D+1) \times 1$

# Parameters using Least Squares

- Training data  $\{x_n, t_n\}, n = 1, \dots, N$   
 $t_n$  is a column vector of  $K$  dimensions using 1-of- $K$  form

- Define matrices

$T \equiv n^{\text{th}}$  row is the vector  $t_n^T$  This is  $N \times K$

$X \equiv n^{\text{th}}$  row of which is  $x_n^T$

This is the  $N \times (D+1)$  design matrix

- Sum of squares error function

$$E_D(W) = \frac{1}{2} \text{Tr} \{(XW - T)^T (XW - T)\}$$

Notes:

$(XW - T)$  is error vector, whose square is a diagonal matrix

Trace is the sum of diagonal elements

# Minimizing Sum of Squares

- Sum of squares error function

$$E_D(\mathbf{W}) = \frac{1}{2} \operatorname{Tr} \{ (\mathbf{X}\mathbf{W} - \mathbf{T})^T (\mathbf{X}\mathbf{W} - \mathbf{T}) \}$$

- Set derivative w.r.t.  $\mathbf{W}$  to zero, gives solution

$$\mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{T} = \mathbf{X}^\dagger \mathbf{T}$$

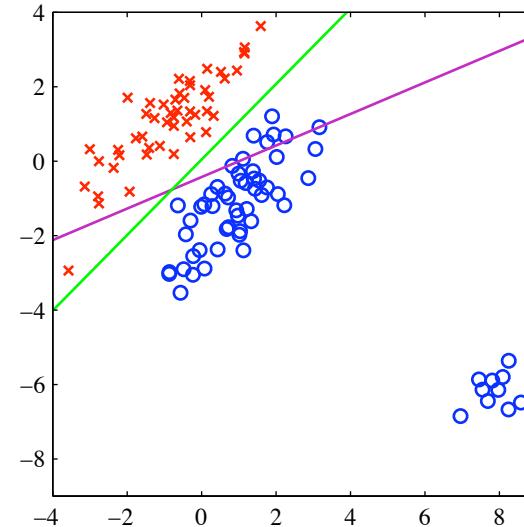
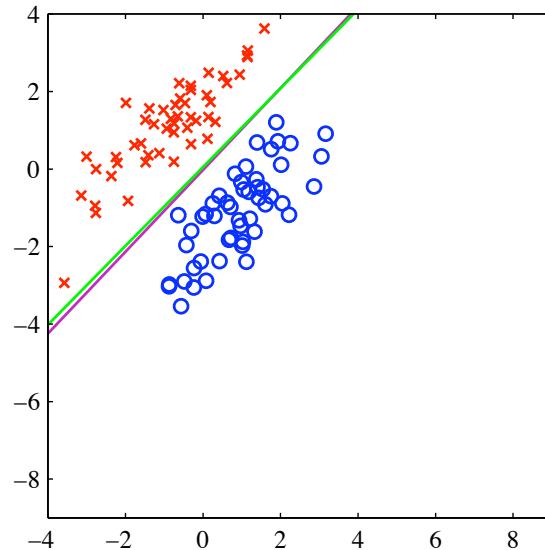
where  $\mathbf{X}^\dagger$  is pseudo-inverse of matrix  $\mathbf{X}$

- Discriminant function, after rearranging, is

$$y(\mathbf{x}) = \mathbf{W}^T \mathbf{x} = \mathbf{T}^T (\mathbf{X}^\dagger)^T \mathbf{x}$$

- An exact closed form solution for  $\mathbf{W}$  using which we can classify  $\mathbf{x}$  to class  $k$  for which  $y_k$  is maximum but has severe limitations

# Least Squares is Sensitive to Outliers



Magenta: Least Squares

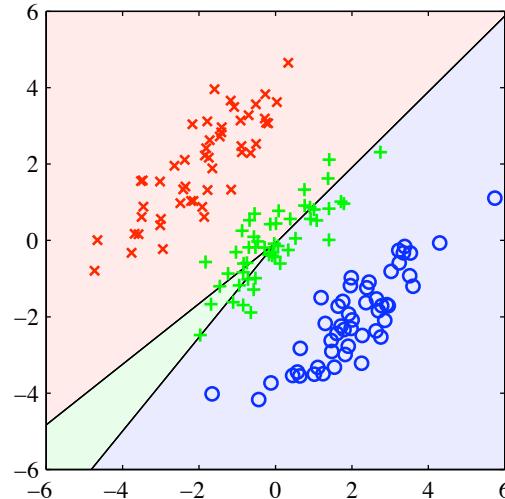
Green: Logistic Regression (more robust)

Sum of squared errors penalizes predictions that are “too correct”  
Or long way from decision boundary

SVMs have an alternate error function (hinge function)  
that does not have this limitation

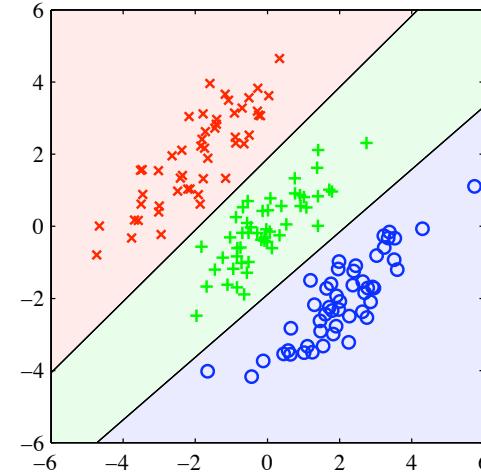
# Disadvantages of Least Squares

Least Squares



Three classes  
2-D space

Logistic Regression

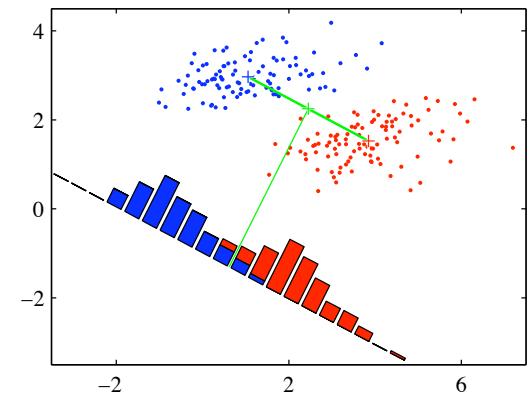


Region assigned to green class is too small, mostly misclassified  
Yet linear decision boundaries of logistic regression can give perfect results

- Lack robustness to outliers
- Certain datasets unsuitable for least squares classification
- Decision boundary corresponds to ML solution under Gaussian conditional distribution
- But binary target values have a distribution far from Gaussian

## 4. Fisher Linear Discriminant

- View classification in terms of dimensionality reduction
  - Project  $D$ -dimensional input vector  $x$  into one dimension using  $y = \mathbf{w}^T \mathbf{x}$
  - Place threshold on  $y$  to classify  $y \geq -w_0$  as  $C_1$  and otherwise  $C_2$   
we get standard linear classifier
- Classes well-separated in  $D$ -space may strongly overlap in 1-dimension
  - Adjust component of the weight vector  $w$
  - Select projection to maximize class-separation



# Fisher: Maximizing Mean Separation

- Two class problem:
  - $N_1$  points of class  $C_1$  and  $N_2$  points of class  $C_2$

- Mean Vectors
 
$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathbf{x}_n$$

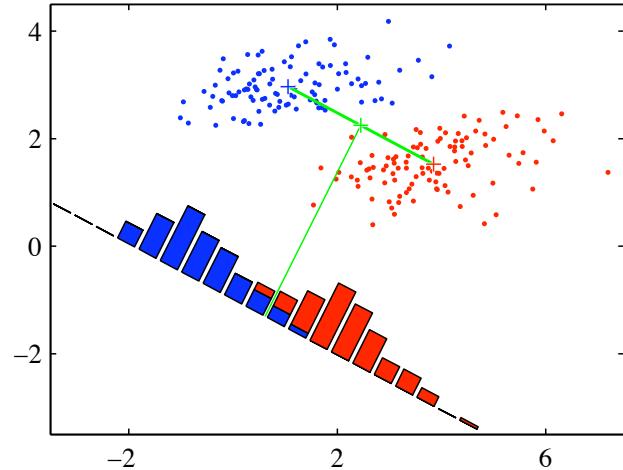
$$\mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathbf{x}_n$$

- Choose  $\mathbf{w}$  to best separate class means
- Maximize  $m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$ ,

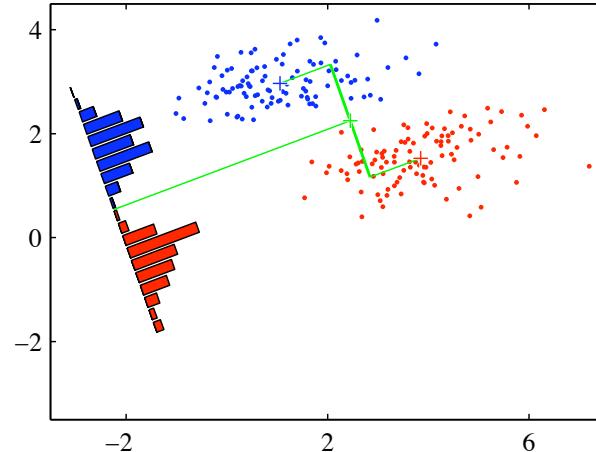
where  $m_k = \mathbf{w}^T \mathbf{m}_k$  is the mean of projected data of class  $C_k$

- Can be made arbitrarily large by increasing  $\mathbf{w}$ 
  - Introduce Lagrange multiplier to enforce ( $\mathbf{w}$  to have unit length)  $\sum_i w_i^2 = 1$
  - There is still a problem with this approach, and Fisher proposed a solution

# Fisher: Minimizing Variance



Means are well-separated  
but classes overlap



Projection based on Fisher  
showing greatly improved  
class separation

- Maximizing mean separation is insufficient for classes with non-diagonal covariance
- Fisher formulation
  1. Maximize function to separate projected class means
  2. Also give small variance within each class, thereby minimizing the class overlap

# Fisher Criterion and its Optimization

- In 1-dimensional space, within class variance is

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2, \text{ where } y_n = \mathbf{w}^T \mathbf{x}_n$$

– Total within-class variance is given by  $s_1^2 + s_2^2$

- Fisher criterion =  $J(\mathbf{w}) = (m_2 - m_1)^2 / s_1^2 + s_2^2$

Rewriting  $J(\mathbf{w}) = \mathbf{w}^T S_B \mathbf{w} / \mathbf{w}^T S_W \mathbf{w}$  where  $S_B$  is the between class covariance  $S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$  and  $S_W$  is the within-class covariance matrix

$$S_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

- Differentiating wrt  $\mathbf{w}$ ,  $J(\mathbf{w})$  is maximized when  $(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}$

Dropping scalar factors (in parentheses) & noting  $S_B$  is in same direction as  $(\mathbf{m}_2 - \mathbf{m}_1)$  & multiplying by  $S_W^{-1}$

$$\boxed{\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$$

# Relation to Least Squares

- Least Squares: goal of making Model predictions as close as possible to target values
- Fisher: require maximum class separation
- For two-class problem Fisher is special case of least squares
  - Proof starts with sum-of-square errors and shows that weight vector found coincides with Fisher criterion

# Fisher's Discriminant for Multiple Classes

- Can be generalized for multiple classes
- Derivation is fairly involved [Fukunaga 1990]

# The Perceptron Algorithm

- Another example of a linear discriminant model
  - Others: discriminant functions, Fisher linear discrim
- Occupies an important place in the history of machine learning
- It corresponds to a two-class model
  - In which the input vector  $x$  is first transformed to give feature vector  $\phi(x)$
  - Then used to construct a generalized linear model

$$y(x) = f(\mathbf{w}^T \phi(x))$$

# Perceptron Model

- Two-class model
  - Input vector  $x$  transformed by a fixed nonlinear transformation to give feature vector  $\phi(x)$ 
$$y(x) = f(\mathbf{w}^T \phi(x))$$
where non-linear activation  $f(\bullet)$  is a step function

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

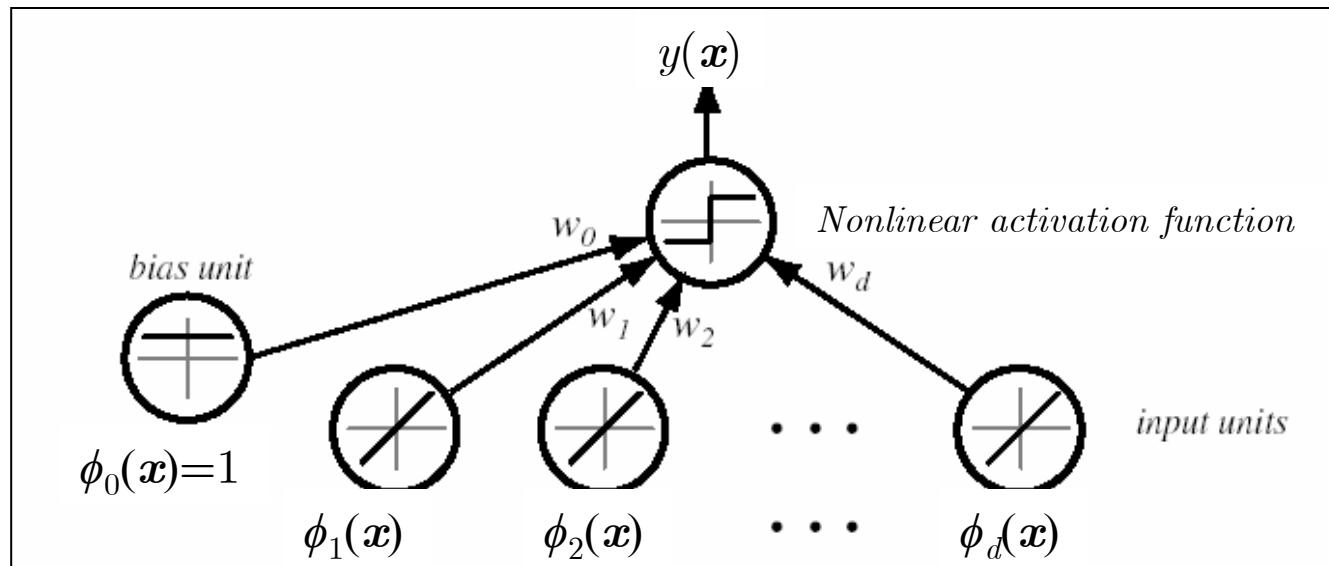
The vector  $\phi(x)$  includes a bias component  $\phi_0(x)=1$

# Perceptron functional diagram

- $y(x) = f(\mathbf{w}^T \phi(x)) \quad \phi(x) = [\phi_0(x), \dots, \phi_d(x)], \mathbf{w} = [w_0, \dots, w_d]$

Non-linear activation function definition:

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$



# Perceptron Target Coding

- Use a target coding scheme
  - In earlier discussion we focused on  $t \in \{0,1\}$ 
    - Which is appropriate for probabilistic models
  - For perceptrons it is more convenient to use target values
    - $t = +1$ , for class  $C_1$  and  $t = -1$  for  $C_2$  matching the activation function

# Determining parameters $w$

- Perceptron criterion is motivated by error function minimization
- A natural choice is total number of misclassifications,  $E(w)$ 
  - This error function is a piecewise constant function of  $w$  with discontinuities (unlike regression)
    - A change in  $w$  causes the decision boundary to move across one of the data points
  - Methods based on changing  $w$  using gradient of error function cannot be applied
    - Gradient is zero almost everywhere
  - Alternative error criterion is the *perceptron criterion*

# Perceptron Criterion $E_P(\mathbf{w})$

- Seek  $\mathbf{w}$  such that  $x_n \in C_1$  will have  $\mathbf{w}^T \phi(x_n) \geq 0$   
whereas patterns  $x_n \in C_2$  will have  $\mathbf{w}^T \phi(x_n) < 0$
- Using  $t \in \{+1, -1\}$ , it follows that all patterns  
need to satisfy  $\mathbf{w}^T \phi(x_n) t_n > 0$ 
  - Perceptron criterion associates zero error with any  
input correctly classified
  - For each misclassified sample, it tries to minimize

$-\mathbf{w}^T \phi(x_n) t_n$  or

$$E_P(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T \phi_n t_n$$

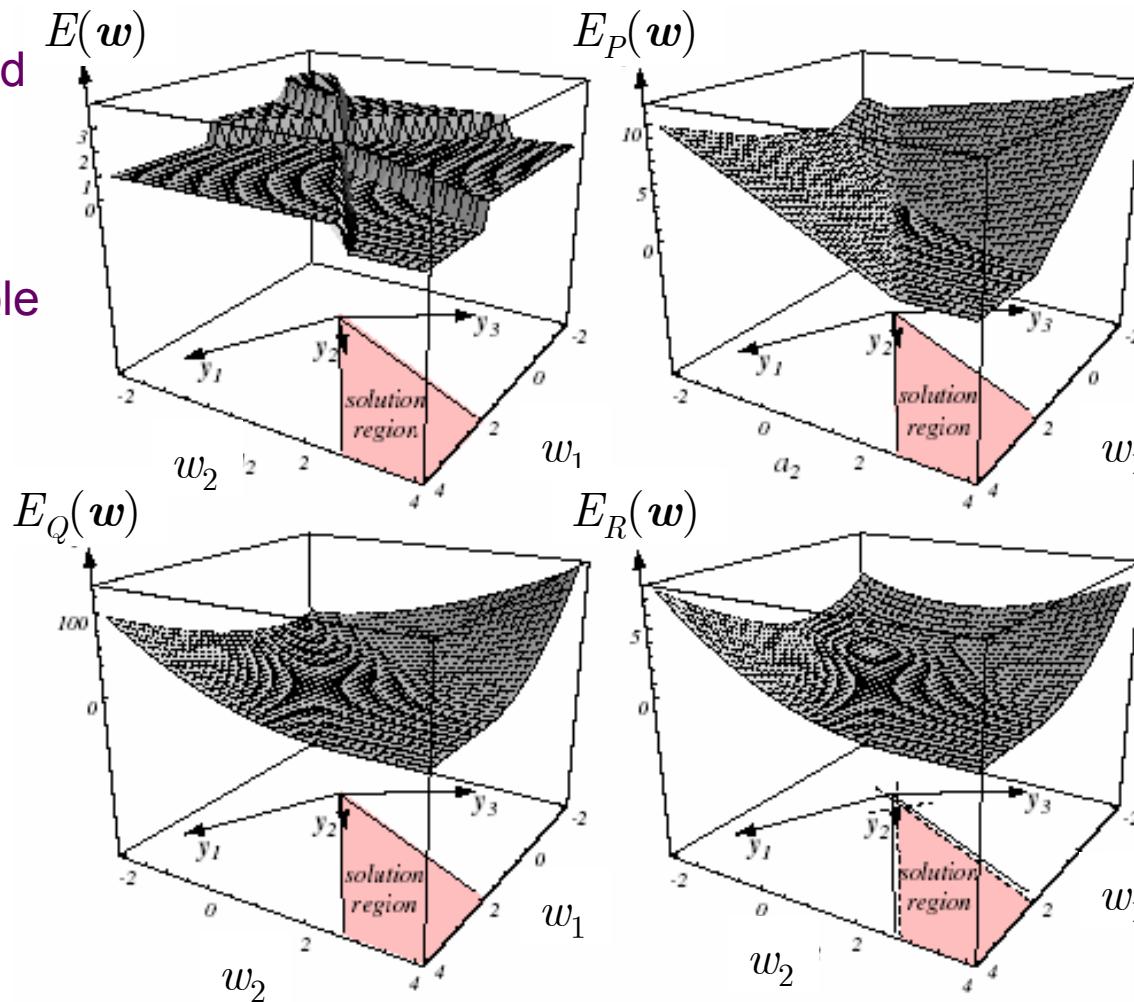
$M$  denotes set of all misclassified patterns and  $\phi_n = \phi(x_n)$

# Property of Perceptron Criterion

- Contribution of a misclassified sample
  - $w^T \phi(x_n) t_n$  is a linear function of  $w$  in regions of  $w$  space where it is misclassified and zero in regions where it is correctly classified
- The total error function is therefore piecewise linear

# Comparison of Four Criterion functions

No of misclassified samples:  
*Piecewise constant, unacceptable*



Perceptron criterion:  
*Piecewise linear, acceptable for gradient descent*

Squared Error with margin

$$E_R(\mathbf{w}) = \frac{1}{s} \sum_{n \in M} \frac{(\mathbf{w}^T \phi_n - b)^2}{\|\phi_n\|}$$

Squared error:  
 Useful when patterns  
 are not linearly separable

$$E_Q(\mathbf{w}) = \sum_{n \in M} (\mathbf{w}^T \phi_n)^2$$

# Perceptron Algorithm

- Error function

$$E_P(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T \phi_n t_n$$

- Apply Stochastic Gradient Descent to  $E_P$
- Change in weight is given by

$$\mathbf{w}^{\tau+1} = \mathbf{w}^\tau - \eta \nabla E_P(\mathbf{w}^\tau) = \mathbf{w}^\tau + \eta \phi_n t_n$$

$\eta$  is learning rate,  $\tau$  indexes the steps

- The algorithm

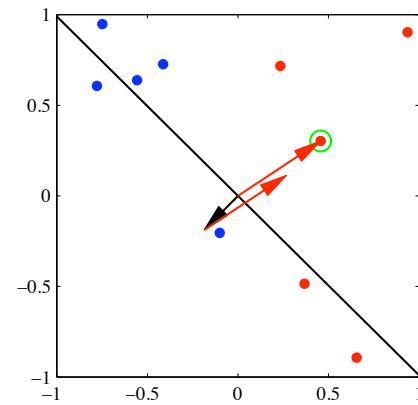
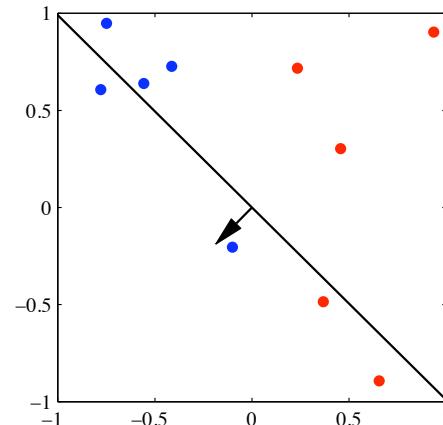
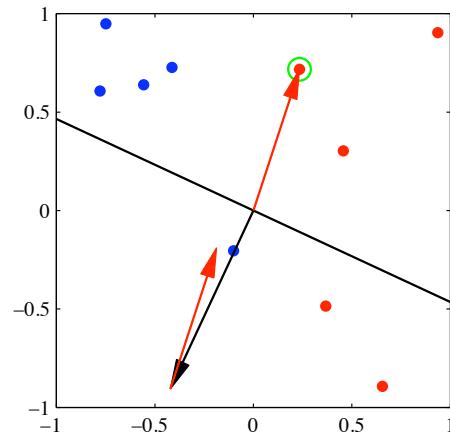
Cycle through the training patterns in turn

If incorrectly classified for class  $C_1$  add to weight vector

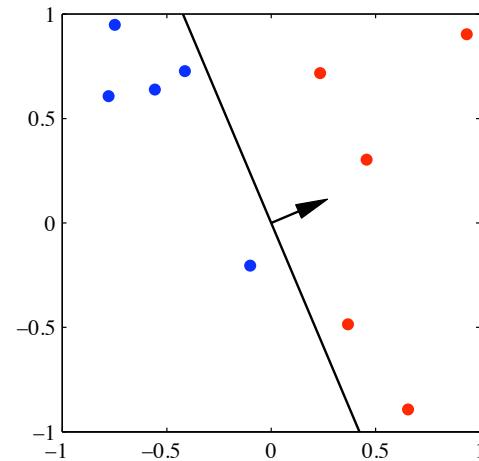
If incorrectly classified for class  $C_2$  subtract from weight vector

# Convergence in feature space $\phi(x)$

2-d Feature space ( $\phi_1, \phi_2$ ) and Two-classes

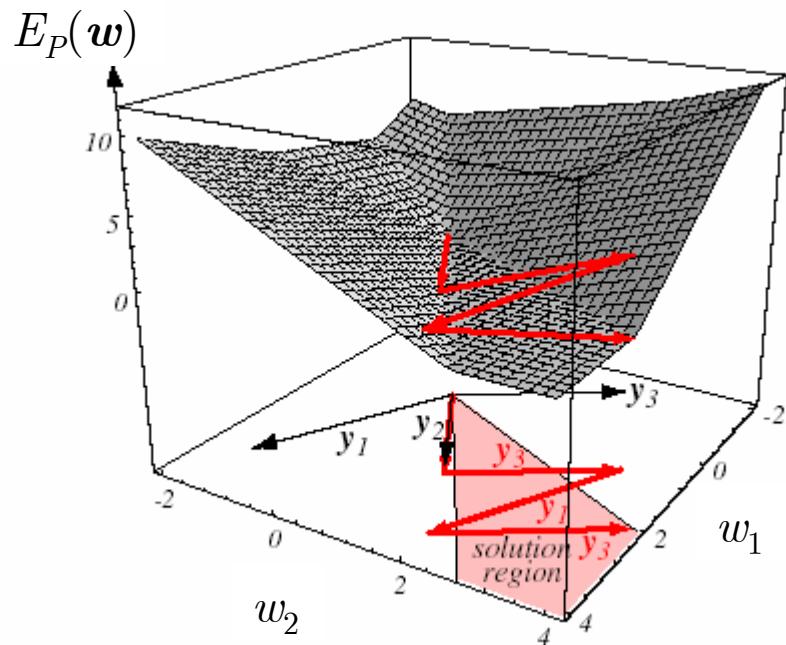


Black arrow: initial parameter vector  $w$   
 Black line: decision boundary  
 Arrow points to red class  
 Circled point in green is misclassified  
 Which is added to current  $w$  to give  
 new decision boundary shown in next plot



Data points  
Correctly classified

# Perceptron Solution in weight space $w$



Criterion function plotted as a function of weights  $w_1$  and  $w_2$

Three samples.

Weight vector starts at origin..  
Sequentially add to it  
“normalized” misclassified  
samples themselves

Sequence is  $y_2, y_3, y_1, y_3$

Second update by  $y_3$  takes solution farther from solution region than first update by  $y_3$

# Perceptron Convergence

Samples  $y_1, y_2, y_3$  considered cyclically  
 Misclassified samples are marked

$$\downarrow \quad \downarrow \quad \dots$$

$$y_1, \ y_2, \ y_3, \ y_1, \ y_2, \ y_3, \ y_1, \ y_2, \ \dots$$

$$\begin{aligned} \mathbf{a}(1) & \quad \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k & \quad k \geq 1 \end{aligned}$$

- Fixed increment single sample correction

```

1 begin initialize a,  $k \leftarrow 0$ 
2           do  $k \leftarrow (k + 1) \bmod n$ 
3           if  $\mathbf{y}^k$  is misclassified by a then a  $\leftarrow \mathbf{a} + \mathbf{y}^k$ 
4           until all patterns properly classified
5   return a
6 end

```

# Perceptron Convergence Theorem

## Theorem (Perceptron Convergence)

If training samples are linearly separable, then the sequence of weight vectors given by *Fixed Increment Error Correction Algorithm* will terminate at a solution vector

## Proof for Single-Sample Correction

Let  $\hat{\mathbf{a}}$  be any solution vector, so that  $\hat{\mathbf{a}}^t \mathbf{y}_i$  is strictly positive for all  $i$ , and let  $\alpha$  be a positive scale factor. From Eq. 20 we have

$$\mathbf{a}(k + 1) - \alpha \hat{\mathbf{a}} = (\mathbf{a}(k) - \alpha \hat{\mathbf{a}}) + \mathbf{y}^k$$

and hence

$$\|\mathbf{a}(k + 1) - \alpha \hat{\mathbf{a}}\|^2 = \|\mathbf{a}(k) - \alpha \hat{\mathbf{a}}\|^2 + 2(\mathbf{a}(k) - \alpha \hat{\mathbf{a}})^t \mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

# Proof for Single-Sample Correction

Because  $\mathbf{y}^k$  was misclassified,  $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$  and thus

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\hat{\mathbf{a}}^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

Because  $\hat{\mathbf{a}}^t\mathbf{y}^k$  is strictly positive, the second term will dominate the third if  $\alpha$  is sufficiently large. In particular, if we let  $\beta$  be the maximum length of a pattern vector,

$$\beta^2 = \max_i \|\mathbf{y}_i\|^2, \quad (21)$$

and let  $\gamma$  be the smallest inner product of the solution vector with any pattern vector, that is,

$$\gamma = \min_i [\hat{\mathbf{a}}^t \mathbf{y}_i] > 0, \quad (22)$$

# Proof for Single-Sample Correction

then we have the inequality

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\gamma + \beta^2.$$

If we choose

$$\alpha = \frac{\beta^2}{\gamma}, \quad (23)$$

we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - \beta^2.$$

Thus, the squared distance from  $\mathbf{a}(k)$  to  $\alpha\hat{\mathbf{a}}$  is reduced by at least  $\beta^2$  at each correction, and after  $k$  corrections we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2 - k\beta^2. \quad (24)$$

Because this squared distance cannot become negative, it follows that the sequence of corrections must terminate after no more than  $k_0$  corrections, where

$$k_0 = \frac{\|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2}{\beta^2}. \quad (25)$$

# Bound on no.of corrections

$$k_0 = \frac{\alpha^2 \|\hat{\mathbf{a}}\|^2}{\beta^2} = \frac{\beta^2 \|\hat{\mathbf{a}}\|^2}{\gamma^2} = \frac{\max_i \|\mathbf{y}_i\|^2 \|\hat{\mathbf{a}}\|^2}{\min_i [\mathbf{y}_i^t \hat{\mathbf{a}}]^2}.$$

# Some Direct Generalizations

Correction whenever  $\mathbf{a}^t(k) \mathbf{y}^k$  fails to exceed a margin

$$\begin{aligned}\mathbf{a}(1) & \text{ arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k & k \geq 1,\end{aligned}$$

---

## ■ Algorithm 5. (Variable-Increment Perceptron with Margin)

```
1 begin initialize a, threshold  $\theta$ , margin  $b$ ,  $\eta(\cdot)$ ,  $k \leftarrow 0$ 
2      do  $k \leftarrow (k + 1) \bmod n$ 
3          if  $\mathbf{a}^t \mathbf{y}^k \leq b$  then a  $\leftarrow \mathbf{a} + \eta(k)\mathbf{y}^k$ 
4      until  $\mathbf{a}^t \mathbf{y}^k > b$  for all  $k$ 
5      return a
6 end
```

---

It can be shown that if the samples are linearly separable and if

$$\eta(k) \geq 0, \quad (28)$$

$$\lim_{m \rightarrow \infty} \sum_{k=1}^m \eta(k) = \infty \quad (29)$$

and

$$\lim_{m \rightarrow \infty} \frac{\sum_{k=1}^m \eta^2(k)}{\left(\sum_{k=1}^m \eta(k)\right)^2} = 0, \quad (30)$$

then  $\mathbf{a}(k)$  converges to a solution vector  $\mathbf{a}$  satisfying  $\mathbf{a}^t \mathbf{y}_i > b$  for all  $i$  (Problem 19).

# Some Direct Generalizations: original gradient descent algorithm for $E_P$

$$\begin{aligned}\mathbf{a}(1) & \quad \text{arbitrary} \\ \mathbf{a}(k+1) &= \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y},\end{aligned}$$

---

## ■ Algorithm 6. (Batch Variable Increment Perceptron)

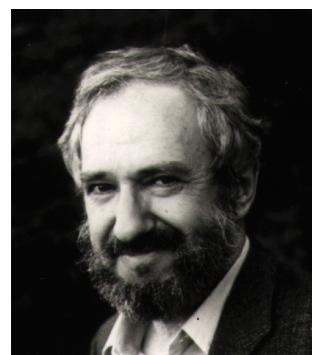
```
1 begin initialize a,  $\eta(\cdot)$ ,  $k \leftarrow 0$ 
2      do  $k \leftarrow (k + 1) \bmod n$ 
3           $\mathcal{Y}_k = \{\}$ 
4           $j = 0$ 
5          do  $j \leftarrow j + 1$ 
6              . . .
7              if  $\mathbf{y}_j$  is misclassified then Append  $\mathbf{y}_j$  to  $\mathcal{Y}_k$ 
8              until  $j = n$ 
9               $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y}$ 
10             until  $\mathcal{Y}_k = \{\}$ 
11 return a
12 end
```

---

# History of Perceptrons

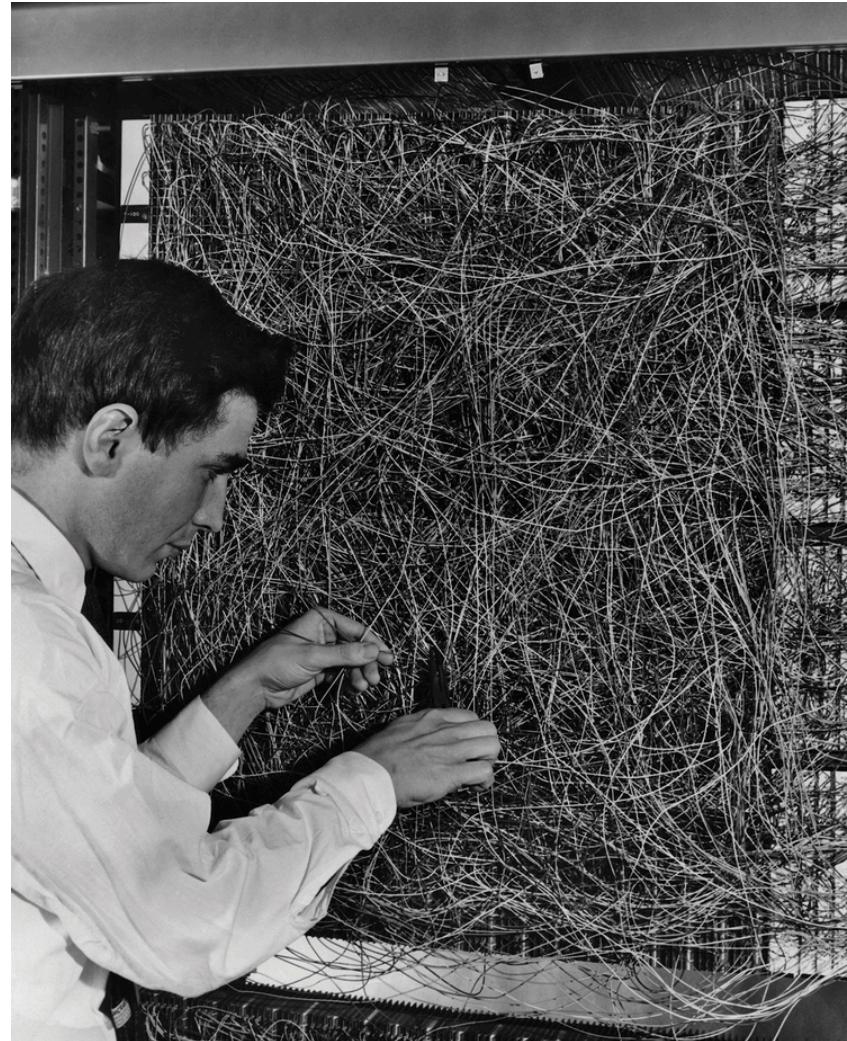
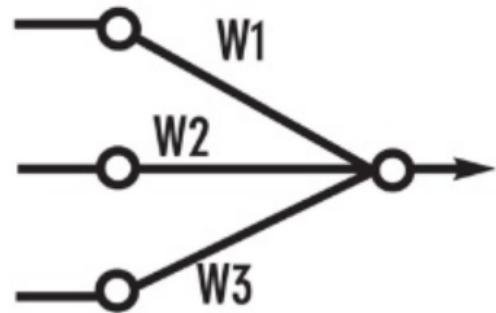
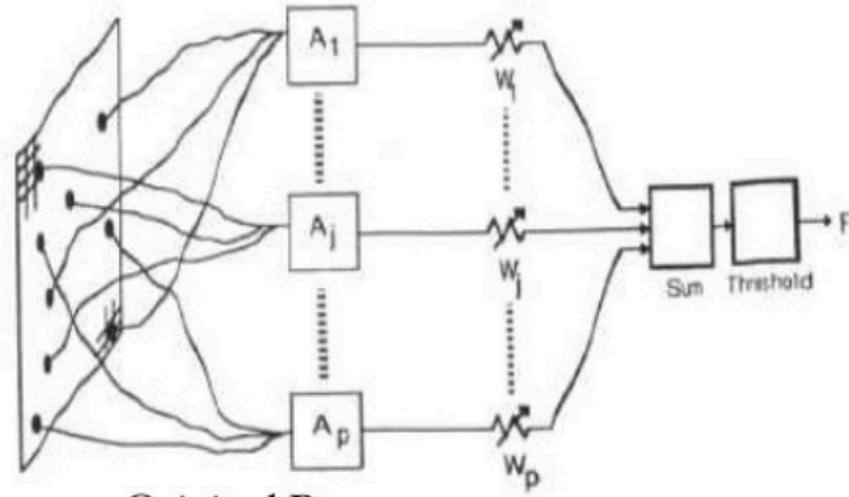


Perceptron  
Invented at Calspan Buffalo, NY  
Rosenblatt, Frank,  
The Perceptron--a perceiving and  
recognizing automaton.  
Report 85-460-1, 1957  
Cornell Aeronautical Laboratory



Minsky and Papert  
dedicated book to him  
Minsky M. L. and Papert S. A. 1969  
*Perceptrons*, MIT Press

# Perceptron



# Perceptron Hardware (Analog)



Learning to discriminate  
shapes of characters

20x20 cell

Image of character

Patch-board  
to allow  
different  
configurations of  
input features  $\phi$

Racks of  
Adaptive Weights  
Implemented  
as potentiometers

Known as Mark 1 Perceptron. It is now in the Smithsonian.

# Disadvantages of Perceptrons

- Does not converge if classes not linearly separable
- Does not provide probabilistic output
- Not readily generalized to  $K > 2$  classes

# Summary

- Linear Discrimin. Funcs have simple geometry
- Extensible to multiple classes
- Parameters can be learnt using
  - Least squares
    - not robust to outliers, model close to target values
  - Fisher' s linear discriminant
    - Two class is special case of least squares
    - Not easily generalized to more classes
  - Perceptrons
    - Does not converge if classes not linearly separable
    - Does not provide probabilistic output
    - Not readily generalized to  $K > 2$  classes