Spring 2017

CIT651 – Introduction to Machine Learning and Statistical Data Analysis

Linear Classifiers (1)

Seif Eldawlatly

Supervised Learning

- Definition

  The task of inferring a function from labeled data

- Typically involves two phases
  - Training phase: Infer the function from provided input vectors and their corresponding labels
  - Test phase: Use the inferred function to predict the label of a new input vector (different from input vectors used during training)

- Formally

  Given a training dataset of N observations $\{x_n\}$, where $n = 1, 2, \ldots, N$ together with the corresponding target values $\{t_n\}$, the goal is to predict the value of t for a new value of x

Linear Classification

- Classification

  Take an input $\mathbf{x}$ and assign it to one of K discrete classes

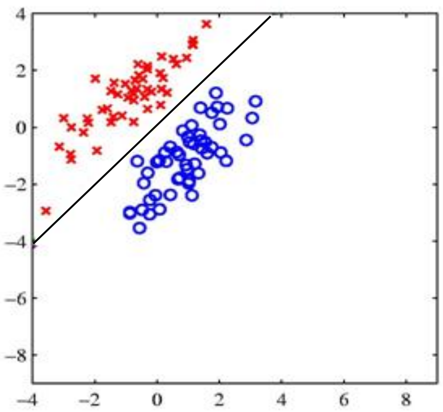
- Decision Boundary

  A boundary (could be linear or non-linear) between two decision regions

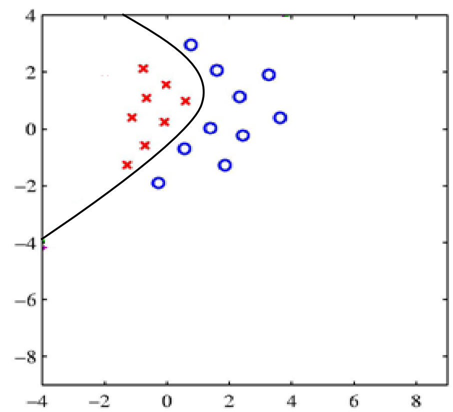
- Decision Regions:

  - Red or Blue, 1 or -1, Friend or Enemy

Linearly Separable



Non-linearly Separable



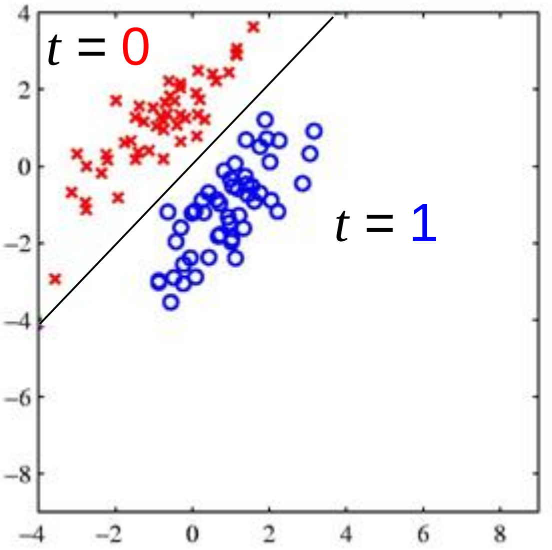
Linear Classification

- Classification Problem

  Goal: Determine the target value (label) for a data point

  Input vector: $\mathbf{x}$

  Target variable: t



- Two Classes (K = 2)

$$t \in \{0,1\} \begin{cases} t = 0 \rightarrow \text{Class } C_1 \\ t = 1 \rightarrow \text{Class } C_2 \end{cases}$$

- K Classes (K > 2)

$$\mathbf{t}_i = [t_1, t_2, \ldots, t_K], \text{where } t_n = 1 \text{ for } \mathbf{x}_i \in C_n \text{ and } t_m = 0, m \neq n$$

Example: $\mathbf{x}_i \in C_3$, K = 5 → $\mathbf{t}$ = [0, 0, 1, 0, 0]

Linear Classifiers

- We will discuss 3 major types of linear classifiers:
  - Discriminant Functions
  - Probabilistic Generative Models
  - Probabilistic Discriminative Models

Discriminant Functions

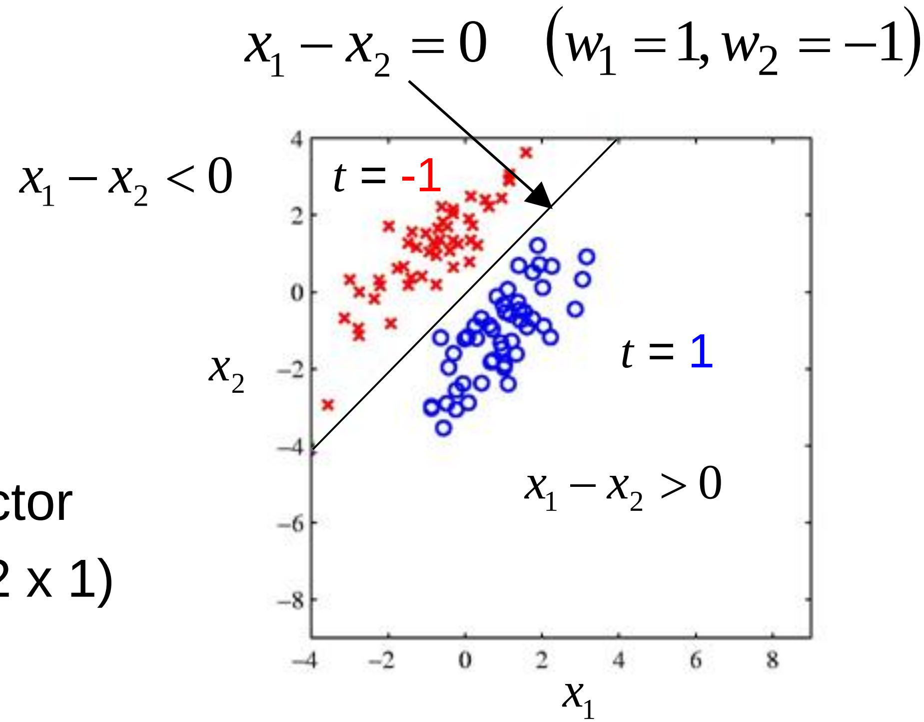
- For the case of two classes

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

$\mathbf{x}$: Input vector

$\mathbf{w}$: Weight vector

$w_\mathbf{0}$: bias



- For this example, $\mathbf{w}$ is a (2 x 1) vector and each input vector $\mathbf{x}$ is also a (2 x 1) vector

$$y(\mathbf{x}) = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_0$$

If the total number of input vectors is 100, then the input dataset consists of $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{100}$, where $\mathbf{x}_i = [x_{i1}, x_{i2}]$, $i = 1{:}100$

- Decision Surface is a hyperplane

Discriminant Functions

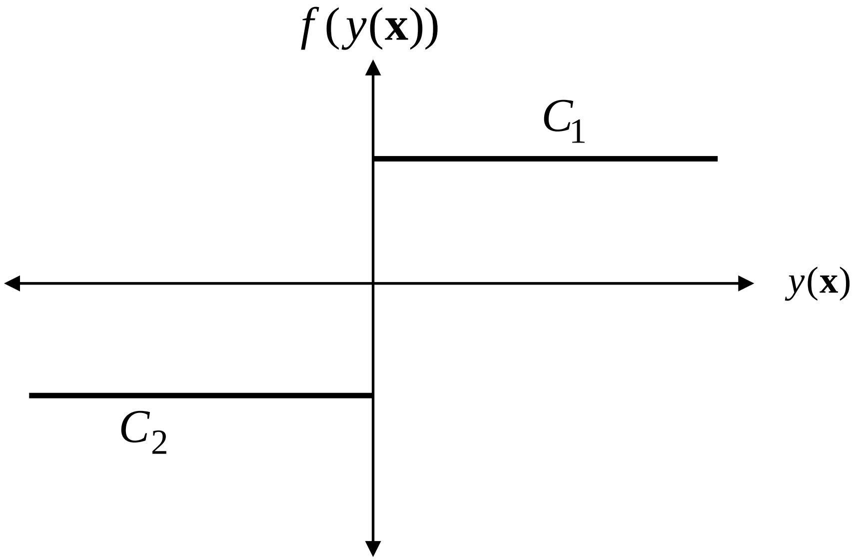
- In this type of methods, the decision boundary is linear but the classification decision is always non-linear

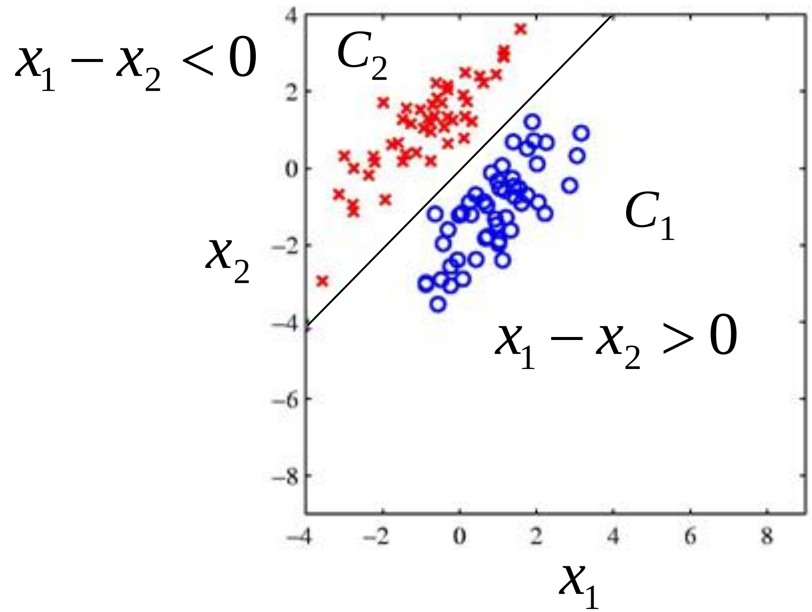
$$\text{if } y(\mathbf{x}) > 0, C = C_1$$
$$\text{if } y(\mathbf{x}) < 0, C = C_2$$

Non-linear Function

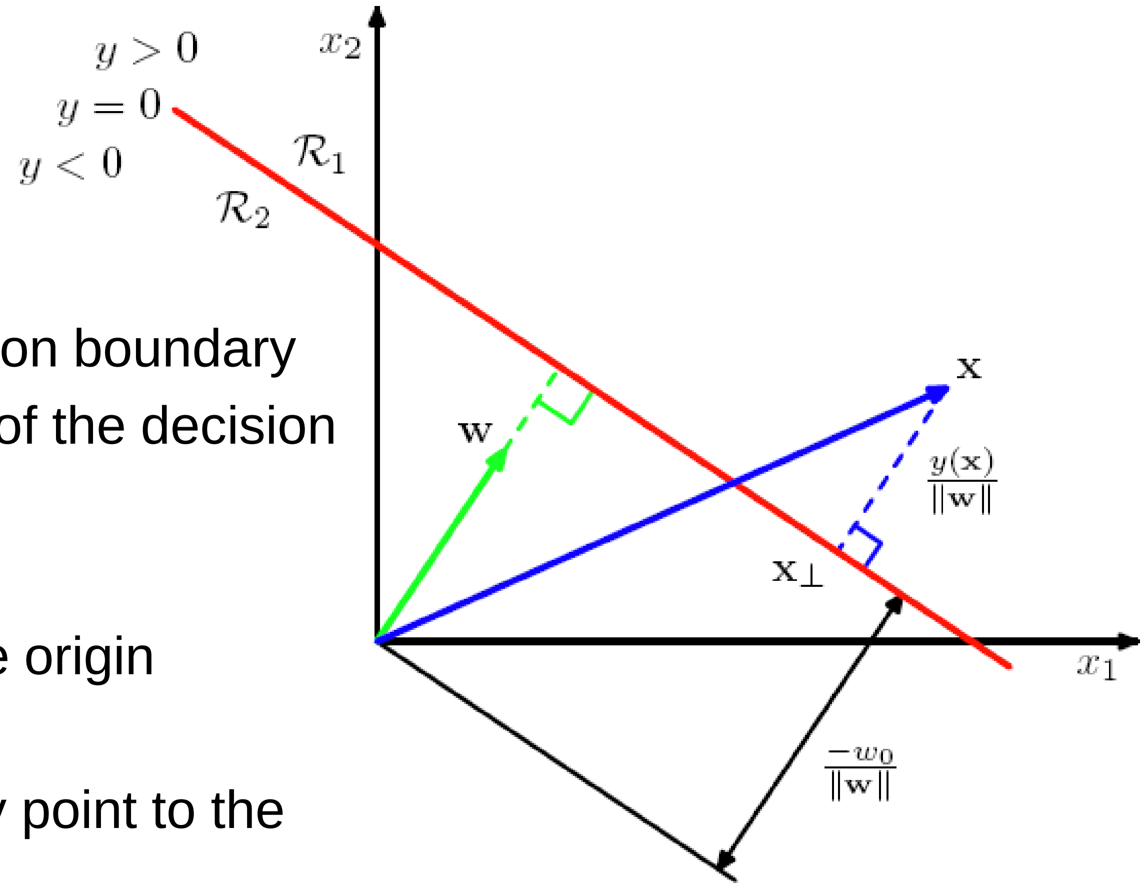
$$Decision = f(y(\mathbf{x}))$$

(Generalized Linear Model)





Discriminant Function Properties



$\mathbf{w}$ is orthogonal to the decision boundary so it defines the orientation of the decision boundary

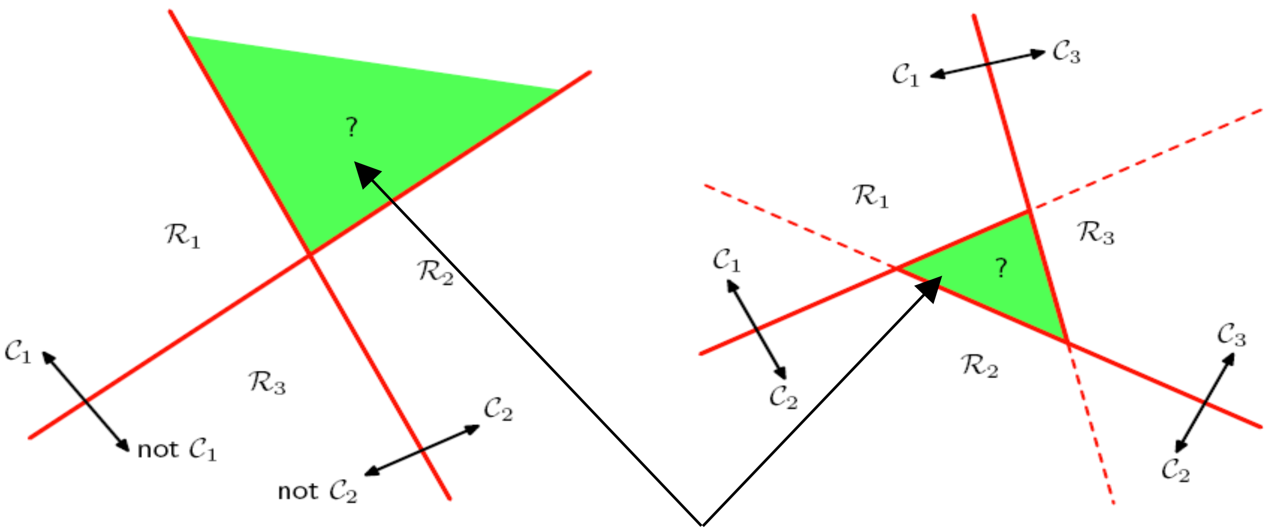
$\frac{-w_0}{\|\mathbf{w}\|}$ is the distance from the origin

$\frac{y(\mathbf{x})}{\|\mathbf{w}\|}$ is the distance from any point to the decision boundary

K-class Discriminant Function

- To classify to multiple classes, a number of ways could be used

One-versus-the-rest One-versus-one



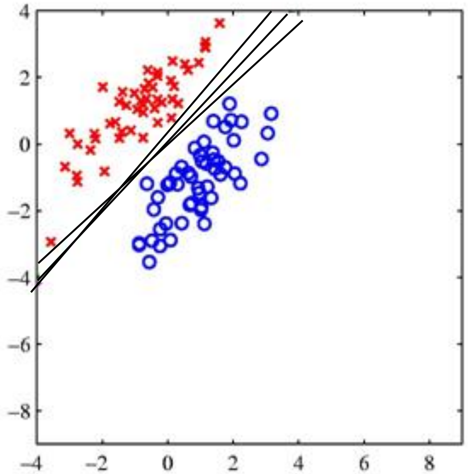
Ambiguous

- Solution: Use K linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, k \in \{1,2,\ldots,K\}$$

$$\text{if } \; y_k(\mathbf{x}) > y_j(\mathbf{x}) \; \text{for all } \; j \neq k, \; \text{then } \; C = C_k$$

Learning Classifier Parameters



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

How to find $\mathbf{w}$ and w_0 ?

Least Squares

Fisher's Linear Discriminant

Perceptron

Least Squares for Classification

- A Simple Solution

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

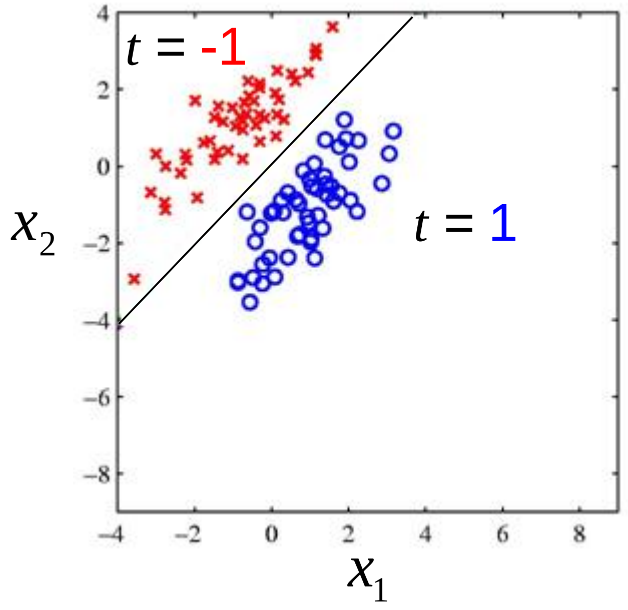
Goal: Find $\mathbf{w}$ and w_0 such that $y(\mathbf{x}) = t$

Let $\tilde{\mathbf{w}} = [\mathbf{w}; w_0]$ $\quad \tilde{\mathbf{x}} = [\mathbf{x}; 1]$

We define an error function as

$$E_D(\tilde{\mathbf{w}}) = \frac{1}{2} \sum_{i=1}^{n} \left(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i - t_i \right)^2$$

where n is the total number of input vectors



- Least squares classifier tries to minimize the difference between the actual ($y(\mathbf{x})$) and desired (t) target values for all input vectors

Least Squares for Classification

- Let

$$\tilde{\mathbf{w}} = [\mathbf{w}; w_0] \qquad \tilde{\mathbf{x}} = [\mathbf{x}; 1]$$

$$E_D(\tilde{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{n}\left(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i - t_i\right)^2$$

To find $\tilde{\mathbf{w}}$ such that error E_D is minimum, take derivative with respect to $\tilde{\mathbf{w}}$ and equate with zero. This results in

$$\tilde{\mathbf{w}} = \left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T \mathbf{t} \qquad \rightarrow \qquad y(\tilde{\mathbf{x}}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

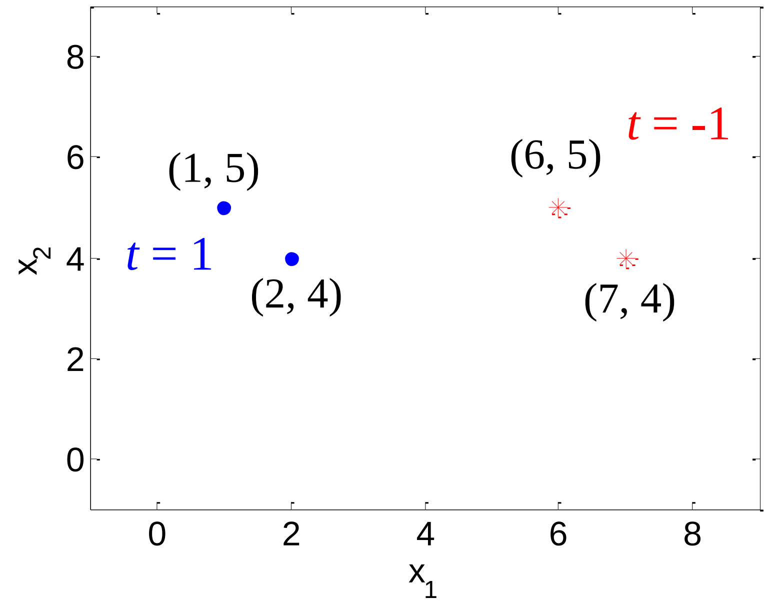
where for 2-dimensional input vectors

$$\tilde{\mathbf{w}} = \begin{bmatrix} w_1 \\ w_2 \\ w_0 \end{bmatrix} \qquad \tilde{\mathbf{X}} = \begin{bmatrix} x_{11} & x_{12} & 1 \\ x_{21} & x_{22} & 1 \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & 1 \end{bmatrix} \qquad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix}$$

Weights Vector   Input Data Matrix   Targets Vector

A Simple Example

- Consider the data given by



- The least squares solution is

$$\tilde{\mathbf{w}} = \left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T \mathbf{t}$$

$$\tilde{\mathbf{X}} = \begin{bmatrix} 1 & 5 & 1 \\ 2 & 4 & 1 \\ 6 & 5 & 1 \\ 7 & 4 & 1 \end{bmatrix}$$

$$\tilde{\mathbf{X}}^T \tilde{\mathbf{X}} = \begin{bmatrix} 1 & 2 & 6 & 7 \\ 5 & 4 & 5 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 5 & 1 \\ 2 & 4 & 1 \\ 6 & 5 & 1 \\ 7 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 90 & 71 & 16 \\ 71 & 82 & 18 \\ 16 & 18 & 4 \end{bmatrix}$$

$$\left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} = \begin{bmatrix} 0.04 & 0.04 & -0.34 \\ 0.04 & 1.04 & -4.84 \\ -0.34 & -4.84 & 23.39 \end{bmatrix}$$

$$\left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T = \begin{bmatrix} -0.1 & -0.1 & 0.1 & 0.1 \\ 0.4 & -0.6 & 0.6 & -0.4 \\ -1.15 & 3.35 & -2.85 & 1.65 \end{bmatrix}$$

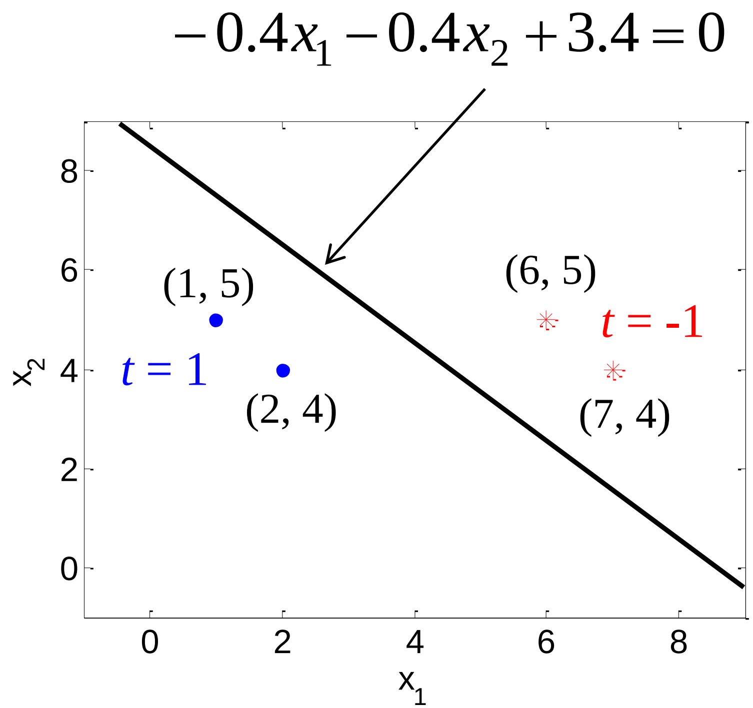
A Simple Example

$$\mathbf{t} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$\left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T \mathbf{t} = \begin{bmatrix} -0.1 & -0.1 & 0.1 & 0.1 \\ 0.4 & -0.6 & 0.6 & -0.4 \\ -1.15 & 3.35 & -2.85 & 1.65 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

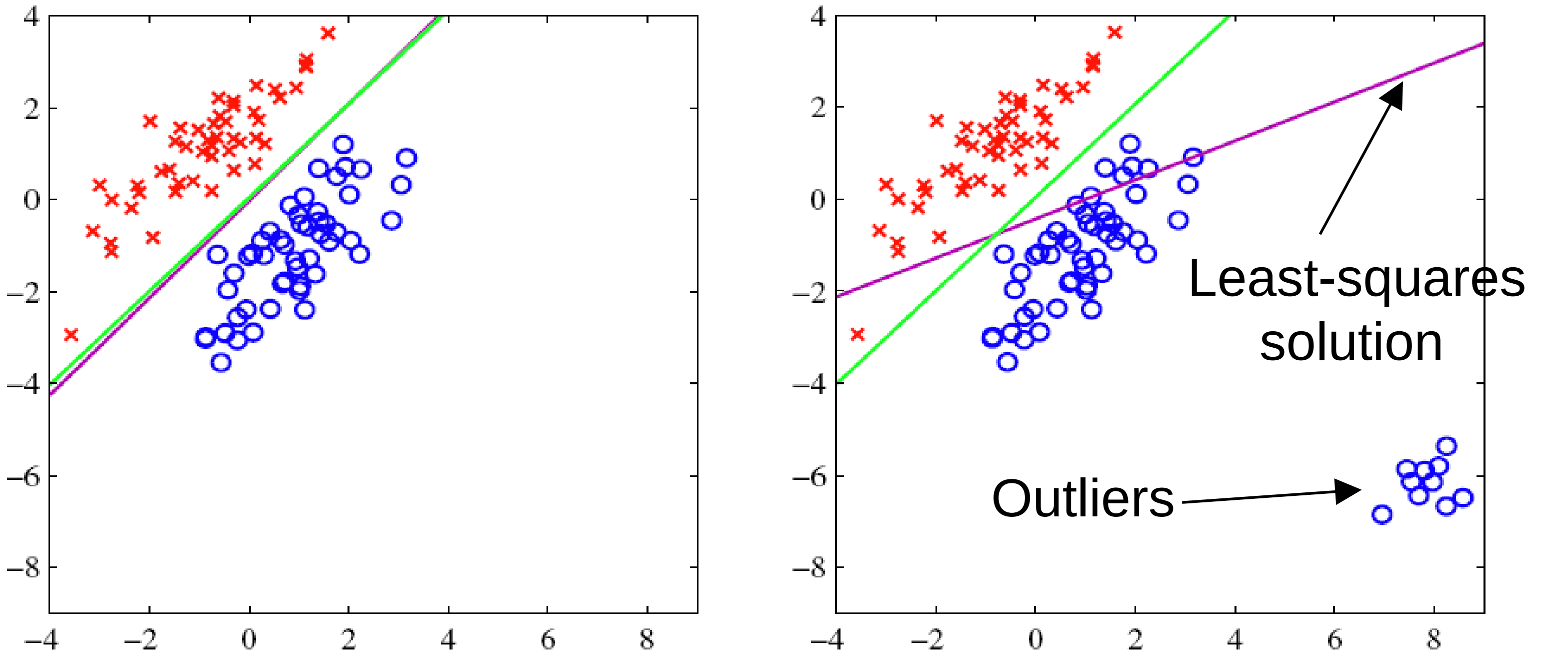
$$= \begin{bmatrix} -0.4 \\ -0.4 \\ 3.4 \end{bmatrix} = \tilde{\mathbf{w}}$$

$$y(\tilde{\mathbf{x}}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} = -0.4x_1 - 0.4x_2 + 3.4$$



Least Squares for Classification

- Problems: Not robust to outliers



- Reason: The error function penalizes points that are too correct

$$E_D(\tilde{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{n}\left(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i - t_i\right)^2$$

Learning Classifier Parameters

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

How to find $\mathbf{w}$ and w_0 ?

Least Squares

Fisher's Linear Discriminant

Perceptron

Fisher's Linear Discriminant

- Discriminant function performs dimensionality reduction

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

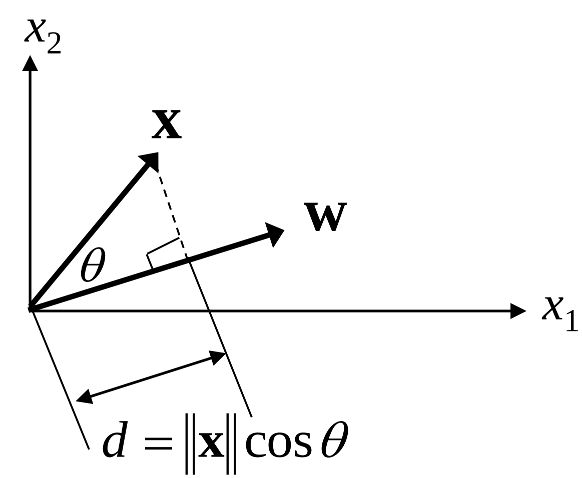
If $\mathbf{x}$ is n x 1, $\mathbf{w}$ must be n x 1 and so $y(\mathbf{x})$ is 1 x 1. Therefore, discriminant function reduces the dimensionality of the input data from n-dimensions to 1 dimension.

- Dimensionality reduction is achieved through the dot product of $\mathbf{w}$ and $\mathbf{x}$

$$\mathbf{w}^T\mathbf{x} = \mathbf{w} \cdot \mathbf{x}$$

- The dot product of $\mathbf{w}$ and $\mathbf{x}$ is equivalent to projecting $\mathbf{x}$ on $\mathbf{w}$

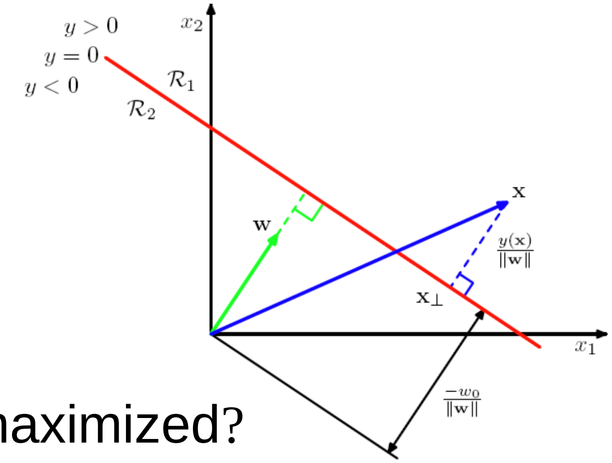
$$\mathbf{w}^T\mathbf{x} = \mathbf{w} \cdot \mathbf{x} = \|\mathbf{w}\|.\|\mathbf{x}\| \cos\theta$$



Fisher's Linear Discriminant

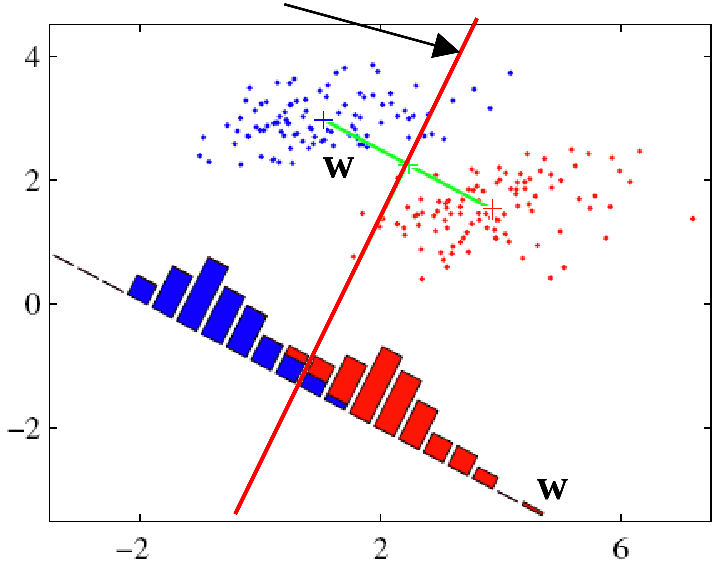
- Projected data might be less separable compared to original data

- Recall that the weights vector $\mathbf{w}$ is perpendicular to the decision boundary

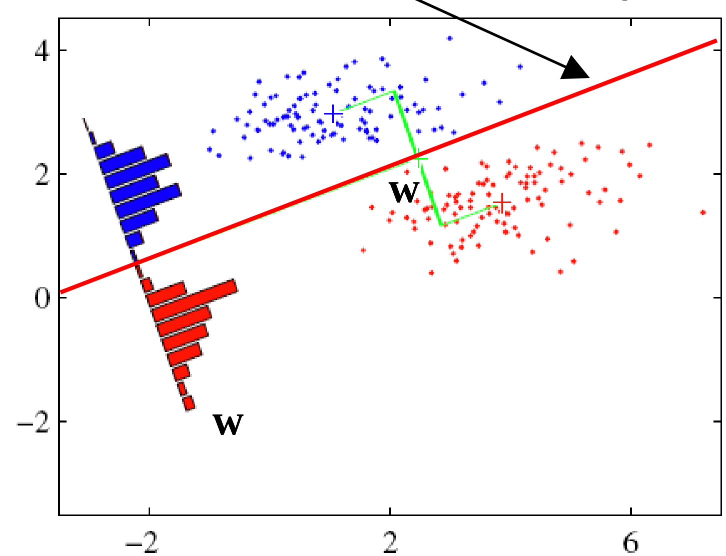


- How to choose $\mathbf{w}$ and w_0 so that separation is maximized?

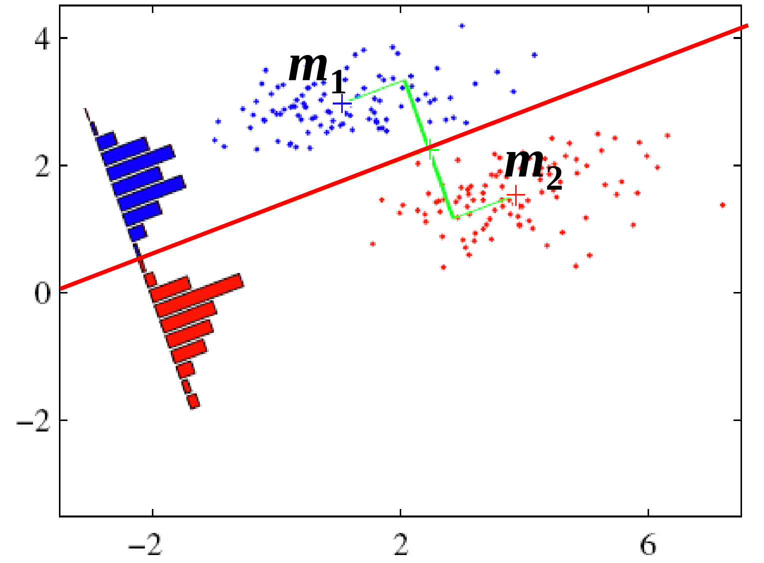
Not good decision boundary



Good decision boundary



Fisher's Linear Discriminant



- Class Means

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

$$m_k = \mathbf{w}^{\mathrm{T}} \mathbf{m}_k$$

- Class Variance

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

- Goal:

  Maximize after-projection separation while minimizing the within-class variance

- Simplest measure of separation is the separation between the means

- Within-class variance can be approximated as the summation of the variances of both classes

Fisher's Linear Discriminant

- Fisher's criterion:
  Maximize separation while minimizing the within-class variance

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \longrightarrow J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}}$$

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}$$

$$\mathbf{S}_{\mathrm{W}} = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}$$

- Solution: Take the derivative of $J(\mathbf{w})$ with respect to $\mathbf{w}$ and equate with 0

$$\frac{d}{d\mathbf{w}} J(\mathbf{w}) = \frac{\left(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}\right)\left(\frac{d}{d\mathbf{w}} \mathbf{w}^T S_{\mathrm{B}} \mathbf{w}\right) - \left(\mathbf{w}^T S_{\mathrm{B}} \mathbf{w}\right)\left(\frac{d}{d\mathbf{w}} \mathbf{w}^T S_{\mathrm{W}} \mathbf{w}\right)}{\left(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}\right)^2}$$

$$= \frac{\left(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}\right)\left(2 S_{\mathrm{B}} \mathbf{w}\right) - \left(\mathbf{w}^T S_{\mathrm{B}} \mathbf{w}\right)\left(2 S_{\mathrm{W}} \mathbf{w}\right)}{\left(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}\right)^2} = 0$$

$$\therefore \left(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}\right)\left(S_{\mathrm{B}} \mathbf{w}\right) = \left(\mathbf{w}^T S_{\mathrm{B}} \mathbf{w}\right)\left(S_{\mathrm{W}} \mathbf{w}\right)$$

Fisher's Linear Discriminant

- Divide both sides by $\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}$

$$\therefore S_{\mathrm{B}} \mathbf{w} = \frac{\mathbf{w}^T S_{\mathrm{B}} \mathbf{w}}{\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}} S_{\mathrm{W}} \mathbf{w}$$

- Since $S_{\mathrm{B}} \mathbf{w}$ is always in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$

$$S_{\mathrm{B}} \mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1) \underbrace{(\mathbf{m}_2 - \mathbf{m}_1)^T}_{(1 \times 2)} \underbrace{\mathbf{w}}_{(2 \times 1)} = (\mathbf{m}_2 - \mathbf{m}_1) c$$

$$\therefore (\mathbf{m}_2 - \mathbf{m}_1) c = \frac{\mathbf{w}^T S_{\mathrm{B}} \mathbf{w}}{\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}} S_{\mathrm{W}} \mathbf{w}$$

$$S_{\mathrm{W}} \mathbf{w} = \frac{\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}}{\mathbf{w}^T S_{\mathrm{B}} \mathbf{w}} c (\mathbf{m}_2 - \mathbf{m}_1)$$

$$S_{\mathrm{W}} \mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$$

$$\therefore \mathbf{w} \propto S_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1) \longrightarrow \textit{Fisher's Solution}$$

Learning Classifier Parameters

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

How to find $\mathbf{w}$ and w_0?

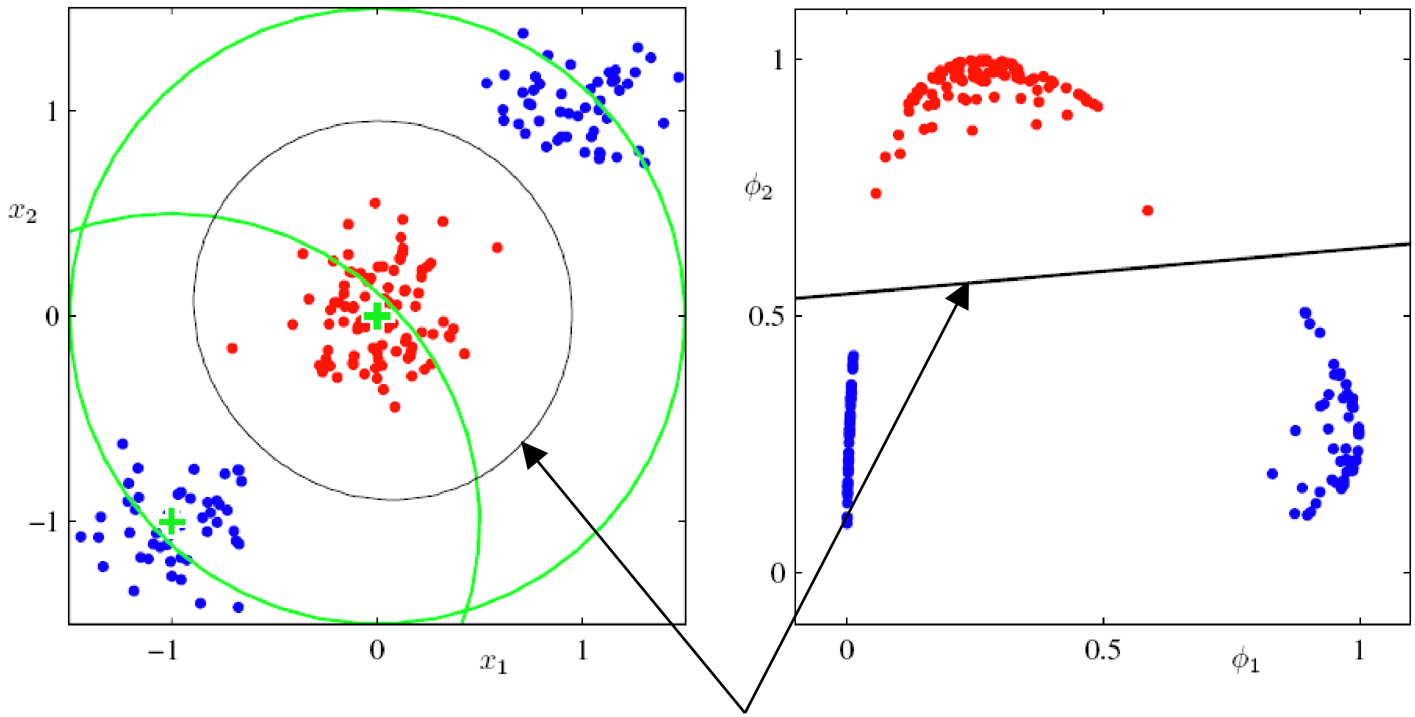
Least Squares

Fisher's Linear Discriminant

Perceptron

Perceptron

- First, let's deal with a nonlinear transformation of the data $\phi(\mathbf{x})$ (basis function)



Decision Boundary

Perceptron

- Define

$$y(\mathbf{x}) = f\left(\mathbf{w}^T\phi(\mathbf{x})\right) = \begin{cases} +1, & \mathbf{w}^T\phi(\mathbf{x}) \geq 0 \\ -1, & \mathbf{w}^T\phi(\mathbf{x}) < 0 \end{cases} \quad \begin{matrix} \rightarrow \text{Class } C_1 \\ \rightarrow \text{Class } C_2 \end{matrix}$$

$\phi(\mathbf{x})$: Feature vector (with a bias component $\phi_0(\mathbf{x}) = 1$)

$f(.)$: Activation function = $t \in \{-1, +1\}$

- Goal: Find $\mathbf{w}$ such that $\mathbf{w}^T\phi(\mathbf{x}_n) \geq 0$ if $\mathbf{x}_n \in C_1$ and $\mathbf{w}^T\phi(\mathbf{x}_n) < 0$ if $\mathbf{x}_n \in C_2$

Or $\mathbf{w}^T\phi(\mathbf{x}_n)t_n \geq 0$

Perceptron

- Perceptron Criterion
  - For correctly classified patterns, error = 0
  - For misclassified patterns, minimize the quantity $-\mathbf{w}^T \phi(\mathbf{x}_n) t_n$

Or minimize $$E_P(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$ M: Misclassified patterns

If t_n = 1 and $\mathbf{w}^T \phi(\mathbf{x}_n)$ < 0, then $\mathbf{w}^T \phi(\mathbf{x}_n) t_n$ < 0
If t_n = -1 and $\mathbf{w}^T \phi(\mathbf{x}_n)$ > 0, then $\mathbf{w}^T \phi(\mathbf{x}_n) t_n$ < 0

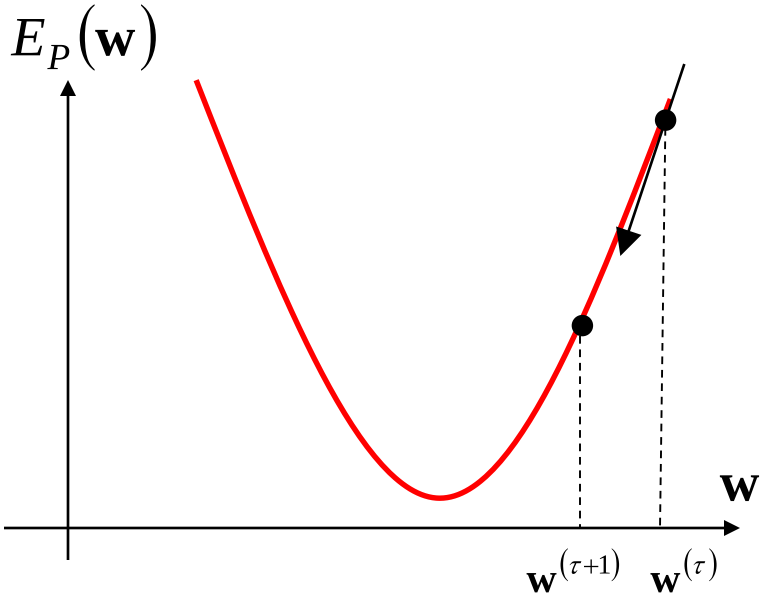
$$\therefore E_P(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$ is always positive

Perceptron

- Using gradient descent we try to iteratively minimize

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

- Consider a 1-dimension $\mathbf{w}$:

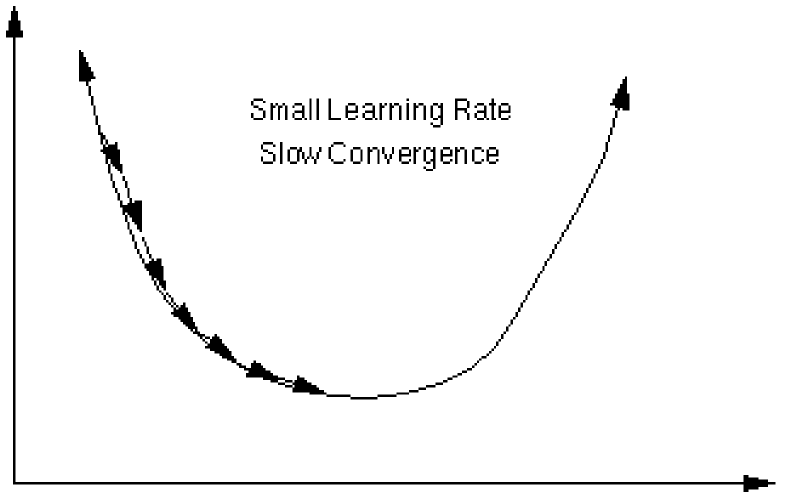


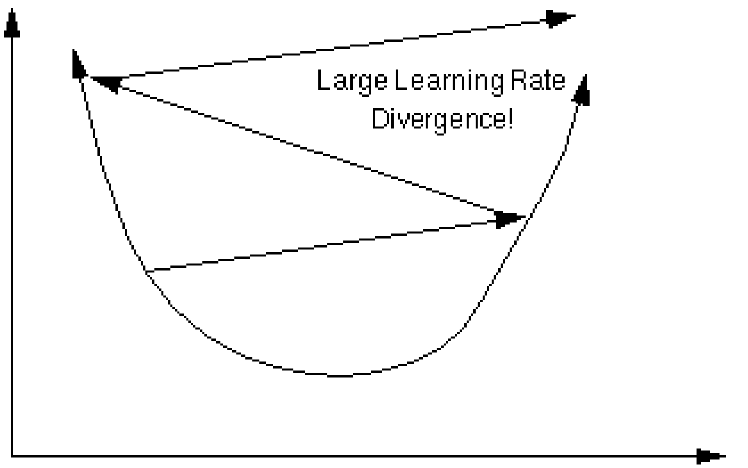
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \frac{\partial E_P}{\partial \mathbf{w}^{(\tau)}} = \mathbf{w}^{(\tau)} + \eta \phi(\mathbf{x}_n) t_n$$

where η is the learning rate parameter

Perceptron

- Choice of η



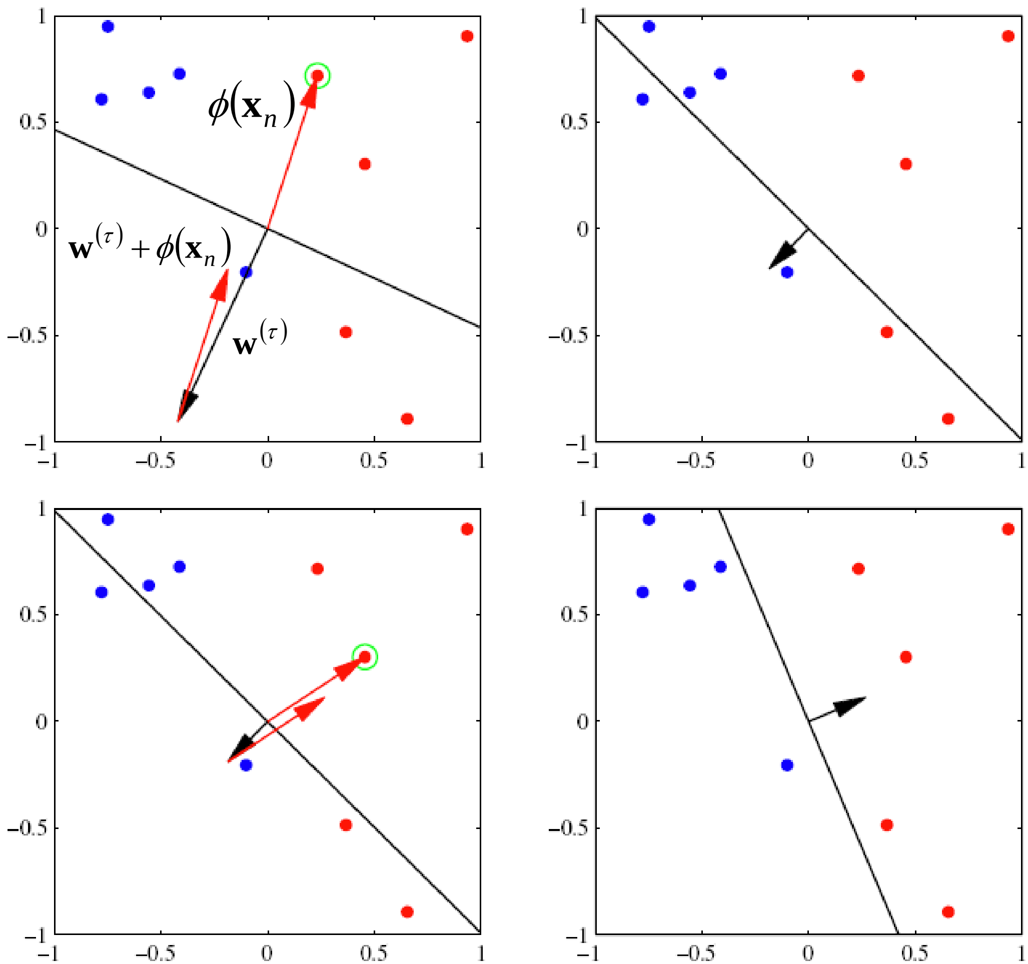


Perceptron

- Example

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta\phi(\mathbf{x}_n)t_n$$

Assume $\eta = 1$ and t_n for red class = +1



Perceptron

- Perceptron algorithm always converges

$$\because \mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi(\mathbf{x}_n)t_n \quad \text{for} \quad \eta = 1$$

Multiply both sides by $-\phi(\mathbf{x}_n)t_n$

$$-\mathbf{w}^{(\tau+1)T}\phi(\mathbf{x}_n)t_n = -\mathbf{w}^{(\tau)T}\phi(\mathbf{x}_n)t_n - (\phi(\mathbf{x}_n)t_n)^T\phi(\mathbf{x}_n)t_n$$

$$\because -\mathbf{w}^{(\tau)T}\phi(\mathbf{x}_n)t_n > 0 \text{ and } (\phi(\mathbf{x}_n)t_n)^T\phi(\mathbf{x}_n)t_n > 0$$

True for any miss-classified point

True since it's equivalent to squaring

$$\therefore -\mathbf{w}^{(\tau+1)T}\phi(\mathbf{x}_n)t_n < -\mathbf{w}^{(\tau)T}\phi(\mathbf{x}_n)t_n$$

Error at iteration τ+1

Error at iteration τ

Since the error is always decreasing, then the algorithm is converging