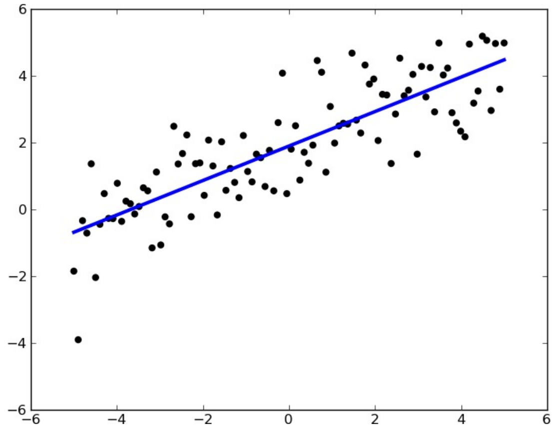
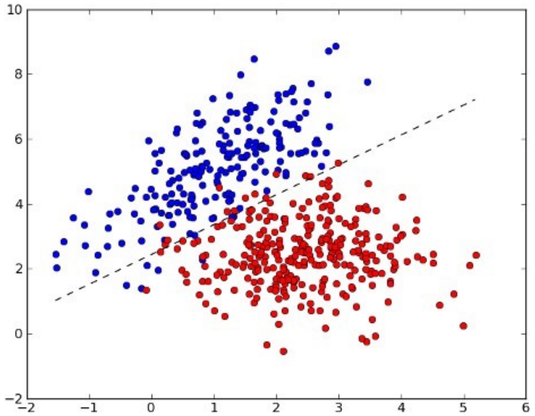
Regression vs. Classification



Regression



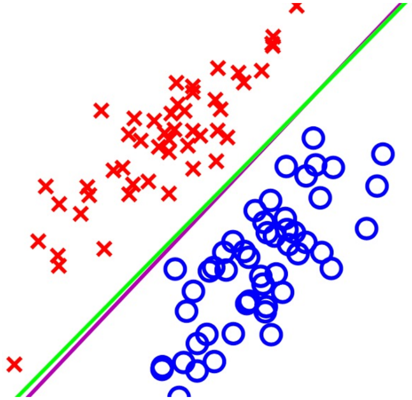
Classification

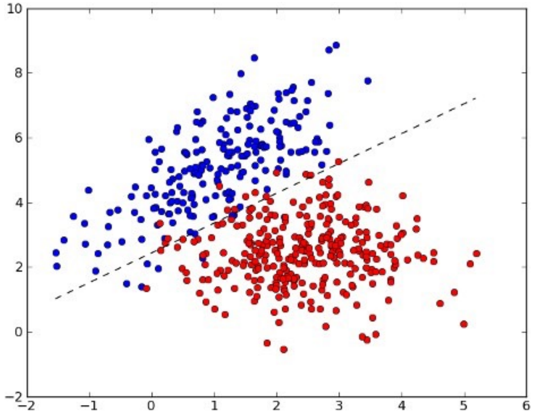
Classification Terminology

- **Goal**: Given data points $\{x\} \in R^D$, assign each data to one class Ck (k= 1, . . . , K)

- **Decision boundaries**: Input space is divided into regions, whose boundaries are called decision boundaries and each region corresponds to a class of data

- **Linearly separable**: Datapoints whose classes can be separated by linear decision boundaries
  - mean that decision boundaries are linear functions of the input x
  - hence are defined by (D – 1)-dimensional hyperplanes within the D-dimensional input space





Classification: Three Different Methods

- **Discriminant models**
  - Given training data, assign each data x to one class C_k via a discriminant function
  - *Do not consider distribution* of the training data
- **Probabilistic discriminant models**
  - Given training data, model the posterior class distribution $p(C_k|x)$
  - Use the distribution $p(C_k|x)$ to perform classification for testing data
- **Probabilistic generative models**
  - Given training data, model the joint (data, class) distribution $p(x,C_k)$
  - Find class-conditional distribution $p(x|C_k)$ and class prior distribution $p(C_k)$
  - Then use Bayes rule to compute $p(C_k|x) \sim p(x|C_k)\, p(C_k)$

Discriminant Models

Binary Classification

- The simplest representation of a linear discriminant function

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

w is called a *weight parameter vector*, and w_0 is a *bias*

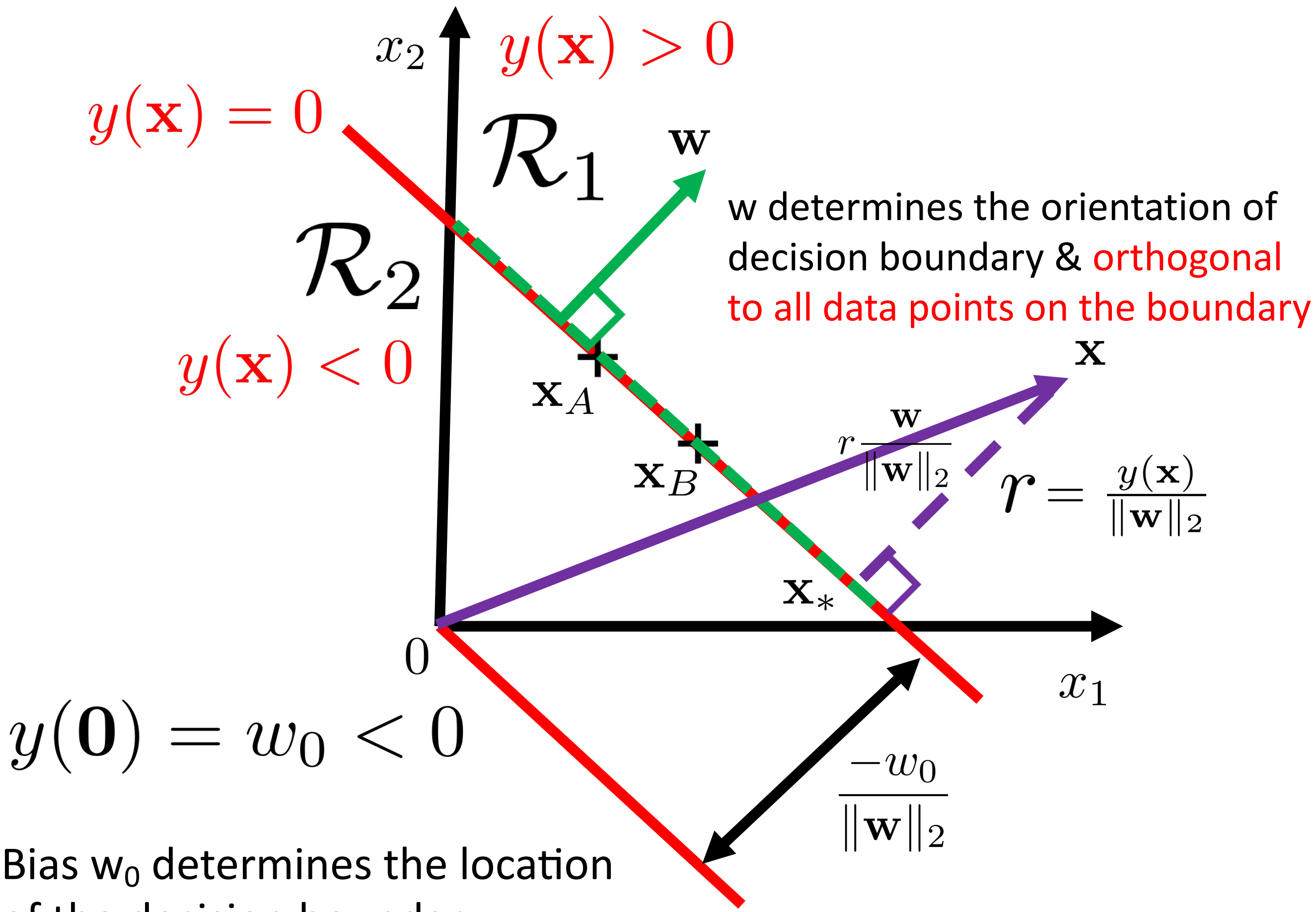
- An input data x is classified to
  - Class C_1 if $y(x) > 0$
  - Class C_2 if $y(x) < 0$

- The decision boundary is defined by

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$$

Geometry of Linear Discriminant Function

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



Bias w_0 determines the location of the decision boundary

$$y(\mathbf{x_A}) = \mathbf{w}^T\mathbf{x}_A + w_0 = 0$$

$$y(\mathbf{x_B}) = \mathbf{w}^T\mathbf{x}_B + w_0 = 0$$

$$\Longrightarrow \quad \mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$$

$$\mathbf{x} = \mathbf{x}_* + r\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$$

$$\mathbf{w}^T\mathbf{x} + w_0 = \mathbf{w}^T\mathbf{x}_* + w_0 + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|_2}$$

$$= 0 + r\|\mathbf{w}\|_2$$

$$\Longrightarrow \quad r = \frac{\mathbf{w}^T\mathbf{x} + x_0}{\|\mathbf{w}\|_2}$$

$$= \frac{y(\mathbf{x})}{\|\mathbf{w}\|_2}$$

Multi-Class Classification

- Build a K-class discriminant function by combining a number of two-class discriminant functions

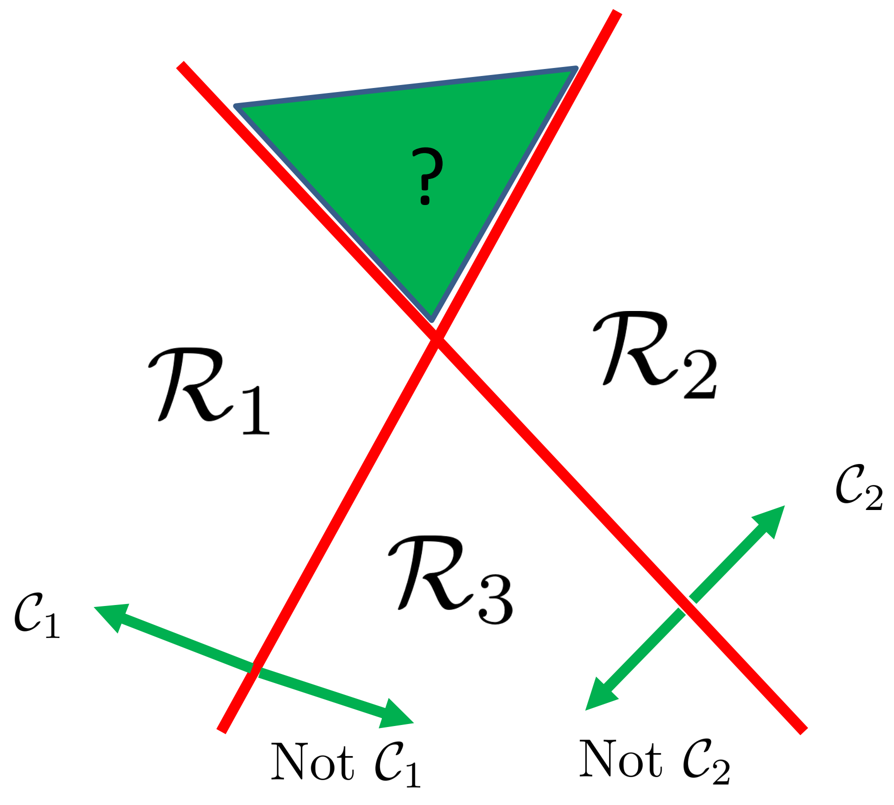
    - One-versus-the-rest classifier
        - Introduce $K-1$ binary discriminant functions, each of which solves a two-class problem of separating points in a particular class C_k from points not in that class

    - *One-versus-one* classifier
        - Introduce $K(K-1)/2$ binary discriminant functions, each one for every possible pair of classes
        - Data are classified according to a majority vote amongst the discriminant functions

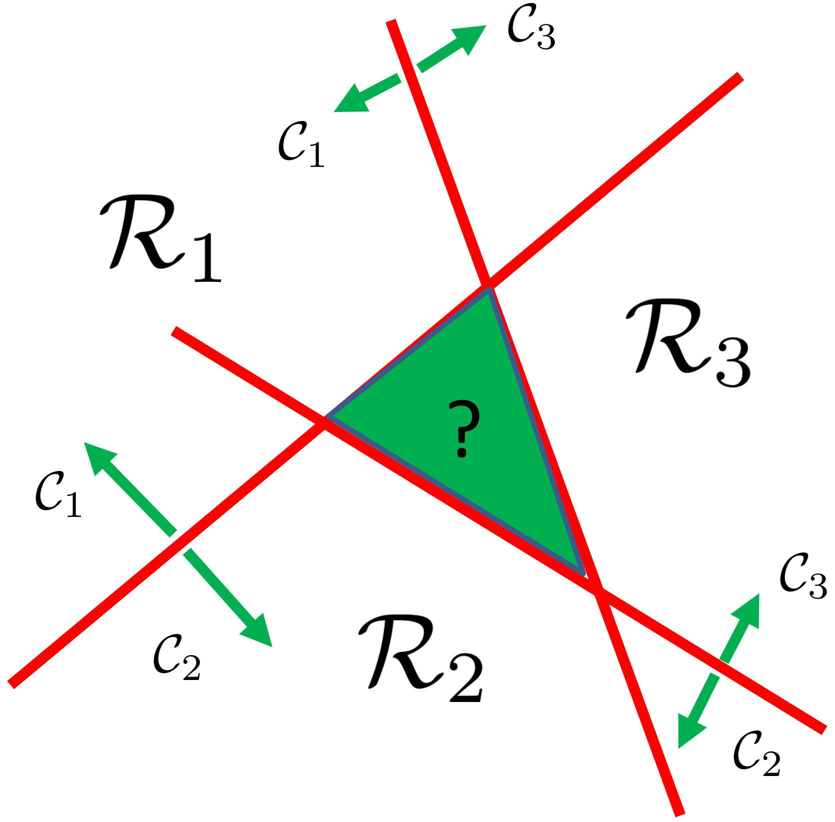
    - The two ways could lead to some issues

Example: Three Classes

One-versus-the-rest classifier



One-versus-one classifier



A Single K-Class Discriminant

- Comprise K linear functions, each for a class

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

- A data point **x** is assigned to class C_k if

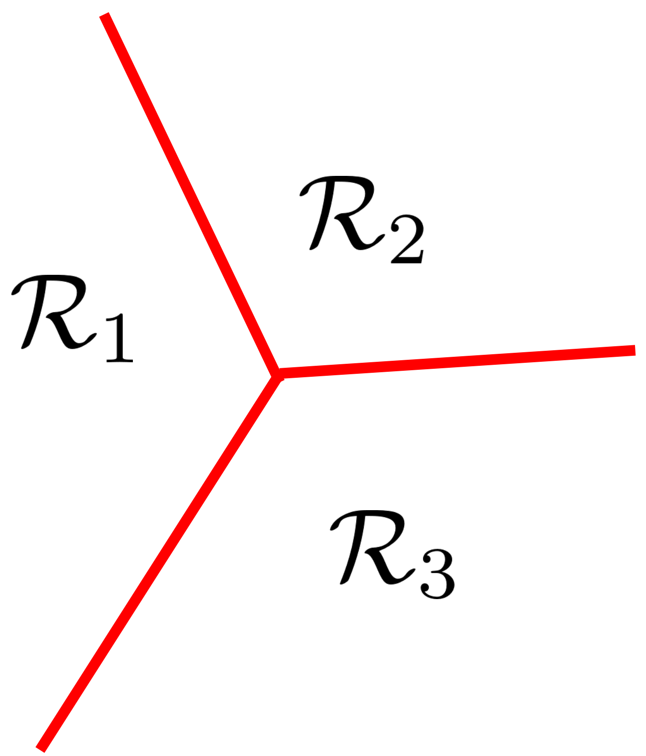
$$y_k(\mathbf{x}) > y_j(\mathbf{x}), \forall j \neq k$$

- The decision boundary between class C_k and class C_j is given by

$$y_k(\mathbf{x}) = y_j(\mathbf{x})$$

  - Which corresponds to a (D−1)-dimensional hyperplane

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$



Least Square for Classification

- Each class C_k is described by its own linear model so that

$$\tilde{\mathbf{w}}_k = [\mathbf{w}_k; w_{k0}] \quad y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

$$\tilde{\mathbf{x}} = [\mathbf{x}; 1] \qquad \mathbf{y}(\tilde{\mathbf{x}}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$$

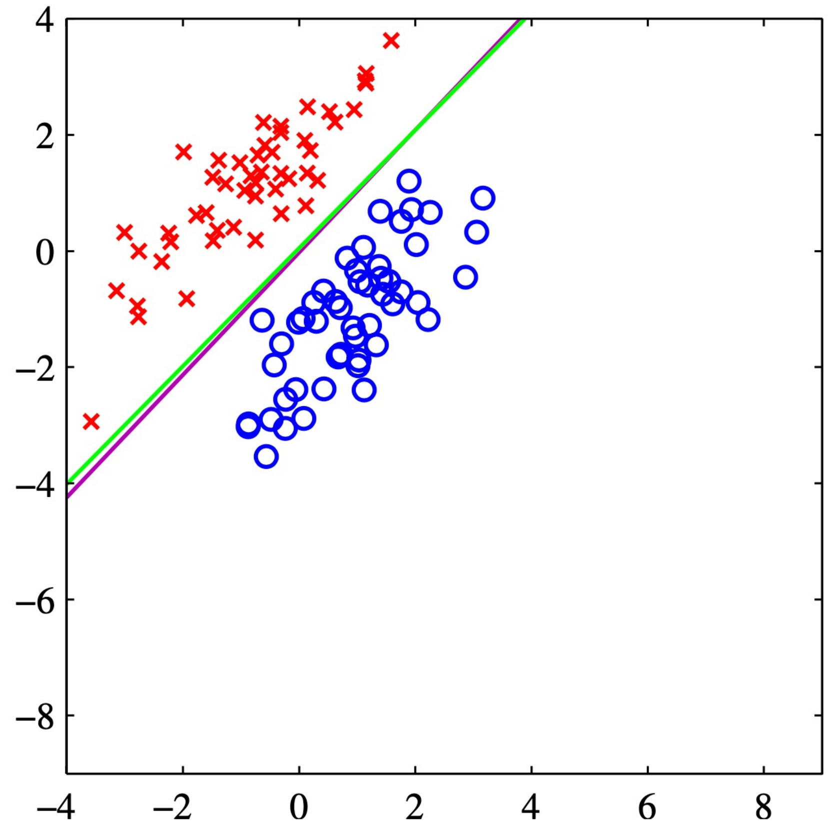
- Consider a training dataset with N data points $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}$
  - Label **t**: one-hot encoding (1-of-K binary coding)
    - #Class = 10 (e.g., in digit recognition).
    - One-hot encoding of $t_n = 3$ is $\mathbf{t_n} = [0,0,0,1,0,0,0,0,0,0] \in \{0,1\}^{10}$

- Least square loss

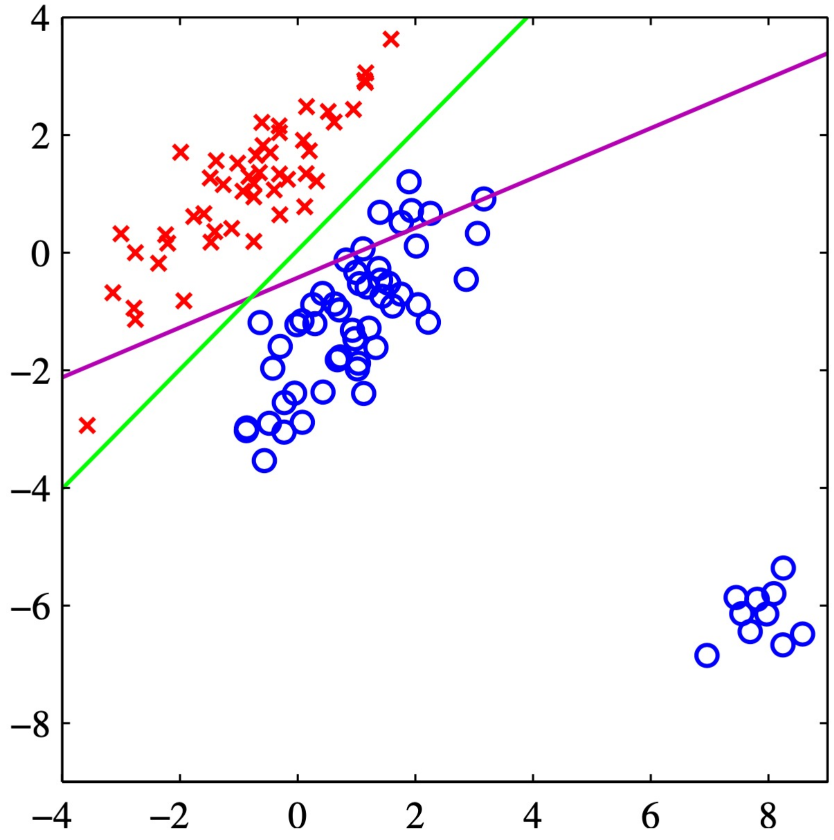
$$\min_{\tilde{\mathbf{W}}} \sum_{n=1}^{N} \left(\mathbf{t}_n - \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}_n\right)^2$$

Normal equation/(Stochastic) Gradient descent

Least Square for Classification

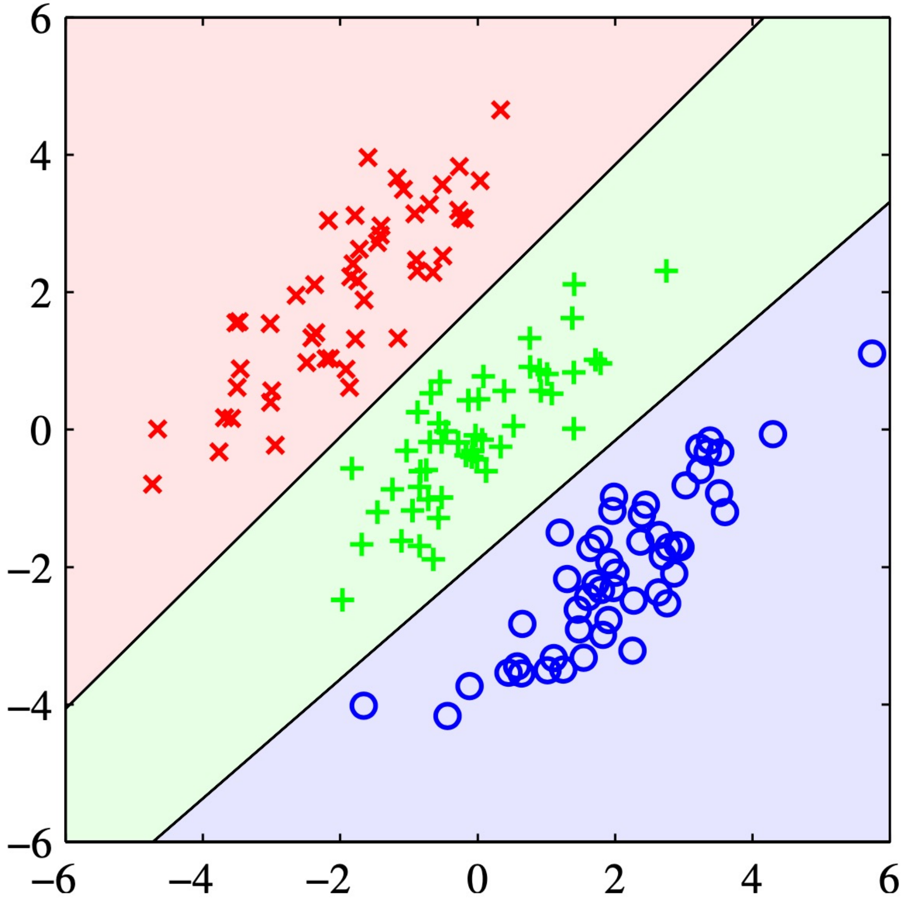


Magenta line is the decision boundary from least squares

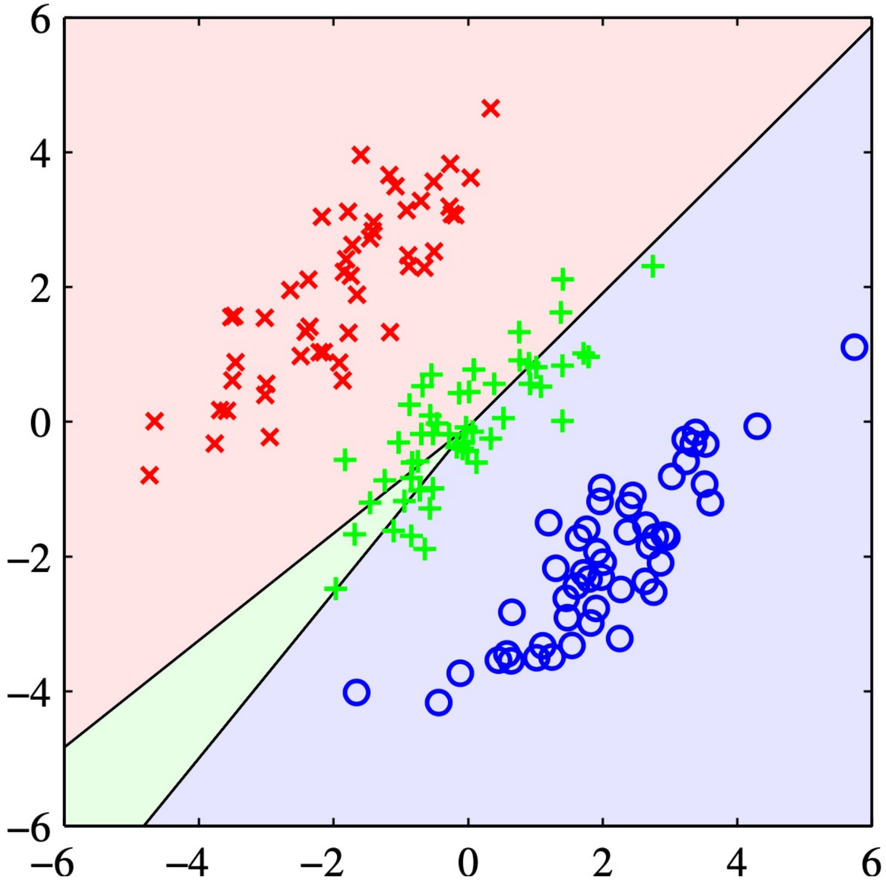


Sensitive to outliers, which lead to large changes in the location of the decision boundary

Least Square for Classification



Linear decision boundaries
could separate classes well



Least square has poor performance

Fisher’s Linear Discriminant Analysis

Linear classification from the viewpoint of dimensionality reduction

Essentially, not a discriminant

Main Idea (Binary Classification)

- Project high-dimensional data into a low-dimensional space such that
  - Projected data points from different classes in low-dim space are separated

- Project a data point ***x*** to 1 dimension with a projection vector **w** is

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

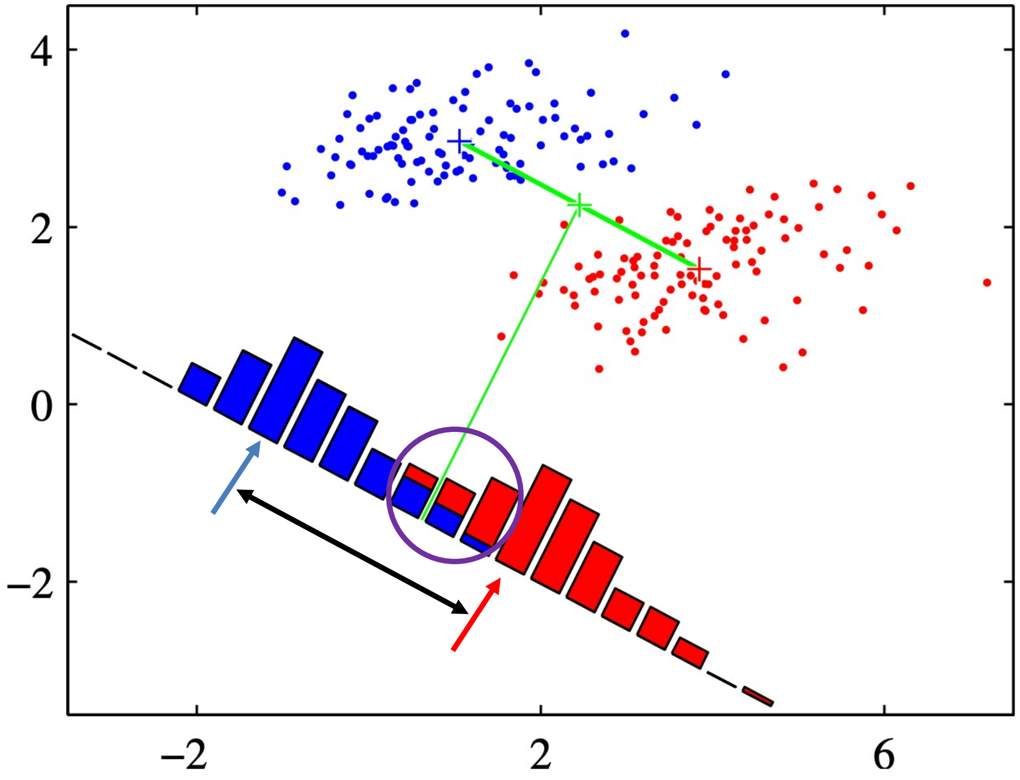
- Goal: maximize the separation of the projected means between classes

$$\mathcal{C}_1 \qquad \mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \qquad m_1 = \mathbf{w}^T \mathbf{m}_1$$

$$\mathcal{C}_2 \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n \qquad m_2 = \mathbf{w}^T \mathbf{m}_2$$

$$\max_{\mathbf{w}} \, (m_2 - m_1)^2 = (\mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1))^2$$

Issue: Between-Class Overlap



Considerable overlap between classes
when projected onto the 1D line

Fisher's Linear Discriminant

- Further minimize within-class variance, thus minimize between-class overlap

$$s_1^2 = \sum_{n \in \mathcal{C}_1} (y_n - m_1)^2 \qquad s_2^2 = \sum_{n \in \mathcal{C}_2} (y_n - m_2)^2$$

$$\min_{\mathbf{w}} \; s_1^2 + s_2^2$$

- Fisher's ratio

$$\max_{\mathbf{w}} (m_2 - m_1)^2$$

$$\min_{\mathbf{w}} (s_1^2 + w_2^2) \quad \Longrightarrow \quad \max_{\mathbf{w}} \frac{(m_2 - m_1)^2}{(s_1^2 + w_2^2)}$$

Fisher's Linear Discriminant

$$m_k = \mathbf{w}^T \mathbf{m}_k \qquad s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \qquad y_n = \mathbf{w}^T \mathbf{x}_n$$

Between-class covariance matrix $\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \Longrightarrow \max_{\mathbf{w}} (m_2 - m_1)^2 = \max_{\mathbf{w}} \mathbf{w}^T \mathbf{S}_B \mathbf{w}$

Total within-class covariance matrix $\mathbf{S}_W = \sum_{k \in \{1,2\}} \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \Longrightarrow \min_{\mathbf{w}} (s_1^2 + w_2^2) = \min_{\mathbf{w}} \mathbf{w}^T \mathbf{S}_W \mathbf{w}$

$$\max_{\mathbf{w}} \frac{(m_2 - m_1)^2}{(s_1^2 + w_2^2)} \Longrightarrow \max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

Differentiate w.r.t w and set it to be **0** $\Longrightarrow \cancel{(\mathbf{w}^T \mathbf{S}_B \mathbf{w})} \mathbf{S}_W \mathbf{w} = \cancel{(\mathbf{w}^T \mathbf{S}_W \mathbf{w})} \mathbf{S}_B \mathbf{w}$

$$\mathbf{S}_B \mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)((\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}) \propto \mathbf{m}_2 - \mathbf{m}_1 \Longrightarrow \mathbf{w} \propto \mathbf{S}_{\mathbf{w}}^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

Fisher's Linear Discriminant

$$\mathbf{w} \propto \mathbf{S}_{\mathbf{w}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

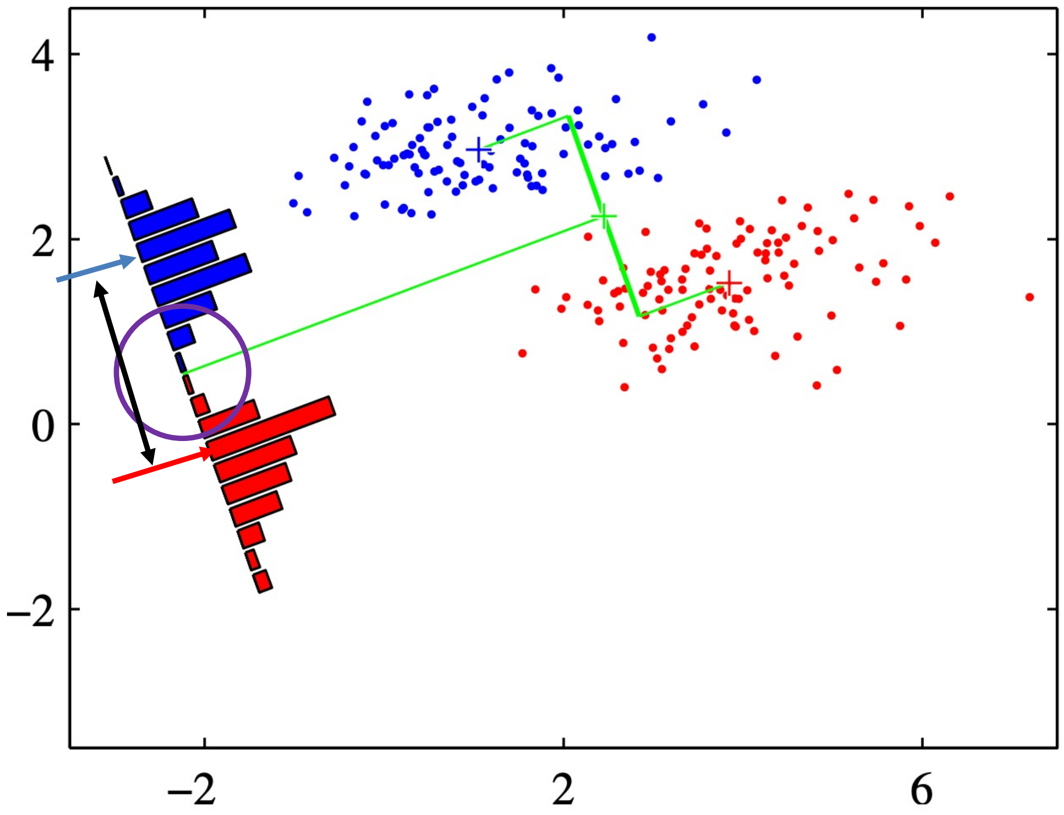
Classification $\quad y(\mathbf{x}_t) = \mathbf{w}^T \mathbf{x}_t$

$$= (\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{S}_{\mathbf{w}}^{-1} \mathbf{x}_t \qquad \mathcal{C}_1$$

$\geq y_0$ Need to determine y_0

It is essentially not a discriminant

A Two-Class Example



Between-class overlap is significantly reduced

Within-class data points are close (small variance)

Generalize to Multi-Classes (K<D)

$$\mathbf{w} \in \mathbb{R}^D \quad y_n = \mathbf{w}^T \mathbf{x}_n \quad \Longrightarrow \quad \mathbf{y}_n = \mathbf{W}^T \mathbf{x}_n \quad \mathbf{W} \in \mathbb{R}^{D \times d}$$

Per-class mean $\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$ $\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k$ All-class mean

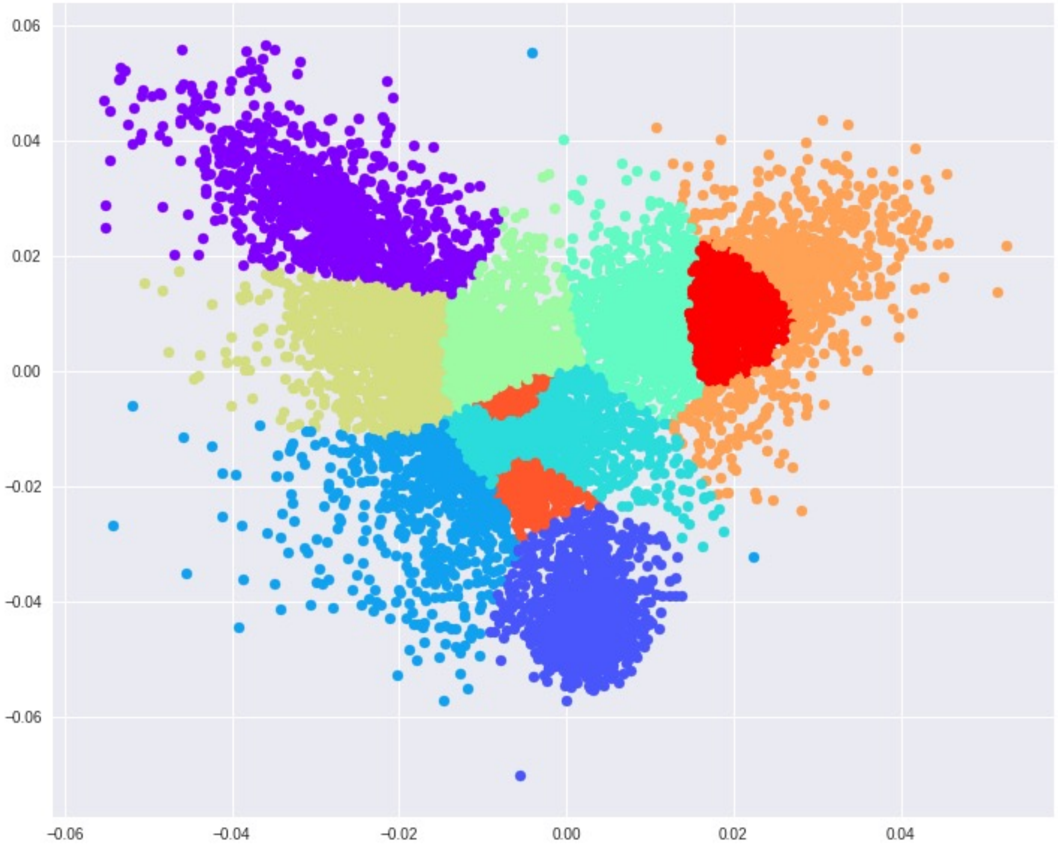
Between-class covariance matrix $\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \Longrightarrow \mathbf{S}_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$

Total within-class covariance matrix $\mathbf{S}_W = \sum_{k \in \{1,2\}} \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \Longrightarrow \mathbf{S}_W = \sum_{k=1}^{K} \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T$

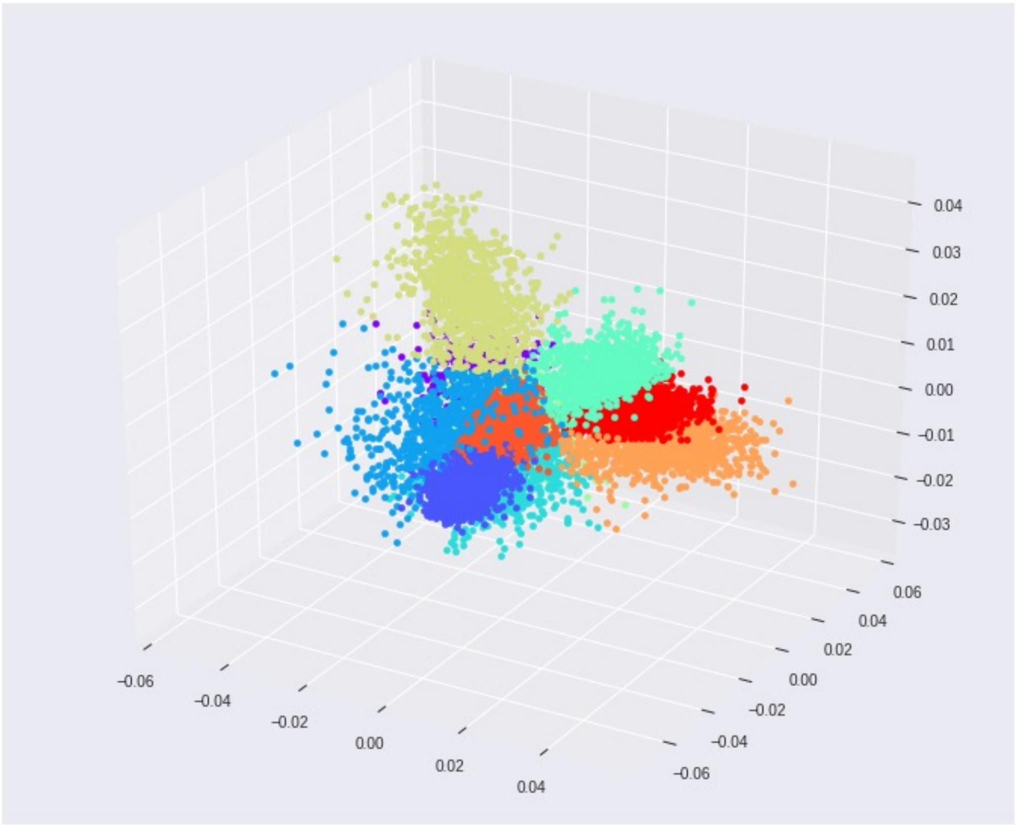
Fisher's ratio $\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \Longrightarrow \max_{\mathbf{w}} \mathrm{Tr}\{(\mathbf{W}^T \mathbf{S}_W \mathbf{W})^{-1} (\mathbf{W}^T \mathbf{S}_B \mathbf{W})\}$

To create a discriminant, we model a Gaussian distribution over the D-dim data x for each class k

10-Classes MNIST Digit Classification



d=2



d=3

Perceptron Algorithm
(Only for Binary Classification)

Perceptron Algorithm

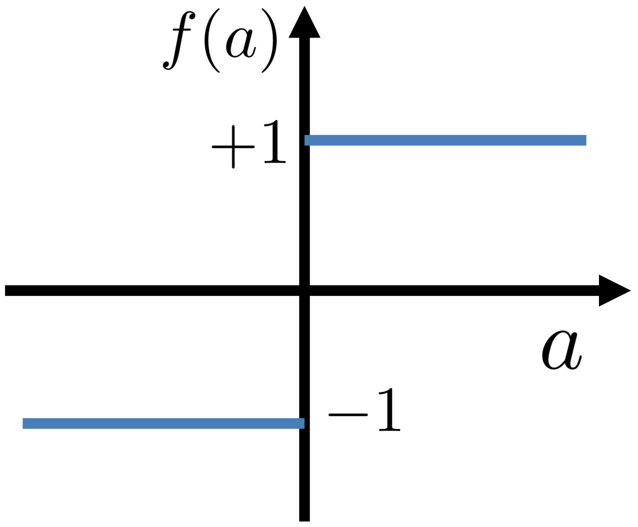
- Another example of a linear discriminant function
  - An important place in the history of pattern recognition (Rosenblatt, 1962)

- Data point x is first transformed using a nonlinear transformation ϕ to give a feature vector ϕ(x);
- Then apply a nonlinear activation function (step function) *f to classify data*

$$y = f(\mathbf{w}^T\phi(\mathbf{x})) \qquad f(a) = \begin{cases} +1, & \text{if } a > 0; \\ -1, & \text{if } a < 0. \end{cases}$$

$$\mathbf{w}^T\phi(\mathbf{x}) > 0 \qquad +1$$

$$\mathbf{w}^T\phi(\mathbf{x}) < 0 \qquad -1$$



Perceptron Algorithm

- Binary classification: label $t \in \{-1, +1\}$

$$\mathbf{x}_n \in \mathcal{C}_1 : t_n = +1 \implies \mathbf{w}^T\phi(\mathbf{x}_n) > 0$$
$$\mathbf{x}_n \in \mathcal{C}_2 : t_n = -1 \implies \mathbf{w}^T\phi(\mathbf{x}_n) < 0$$
$$\implies \mathbf{w}^T\phi(\mathbf{x}_n) \cdot t_n > 0$$

- The perceptron has zero error with any data point correctly classified

$$t_n = +1\,(\mathrm{OR} - 1) \qquad y_n = f(\mathbf{w}^T\phi(\mathbf{x}_n)) = +1\,(\mathrm{OR} - 1)$$

- Whereas a misclassified data x_n it incurs an error

$$-\mathbf{w}^T\phi(\mathbf{x}_n)t_n$$

- The total error of perceptron

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

***M** is the set of all misclassified data*

Solution: Stochastic Gradient Descent

- Per sample gradient descent
  - For a misclassified data point xn

$$E(\mathbf{w};\mathbf{x}_n) = -\mathbf{w}^T\phi(\mathbf{x}_n)t_n \Longrightarrow \nabla_{\mathbf{w}}E(\mathbf{w};\mathbf{x}_n) = -\phi(\mathbf{x}_n)t_n$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nabla_{\mathbf{w}^{(t)}}E(\mathbf{w}^{(t)};\mathbf{x}_n) = \mathbf{w}^{(t)} + \phi(\mathbf{x}_n)t_n$$

- Simple interpretation
  - If a data is correctly classified, then the weight vector remains unchanged
  - If it is incorrectly classified, there is a penalty $|\phi(\mathbf{x}_n)|$

- Error from a data point is reduced with a single update

$$-\left(\mathbf{w}^{(t+1)}\right)^T\phi(\mathbf{x}_n)t_n = -\left(\mathbf{w}^{(t)}\right)^T\phi(\mathbf{x}_n)t_n - \|\phi(\mathbf{x}_n)t_n\|^2 < -\left(\mathbf{w}^{(t)}\right)^T\phi(\mathbf{x}_n)t_n$$

Perceptron Convergence & Correctness

- Convergence
  - If training data is linearly separable, then perceptron learning is guaranteed to find an exact solution in a finite number of iterations

- Correctness
  - Assume the length of all data points is bounded by D, i.e.,

$$\|\mathbf{x}_n\|_2 \leq D, \ \forall n$$

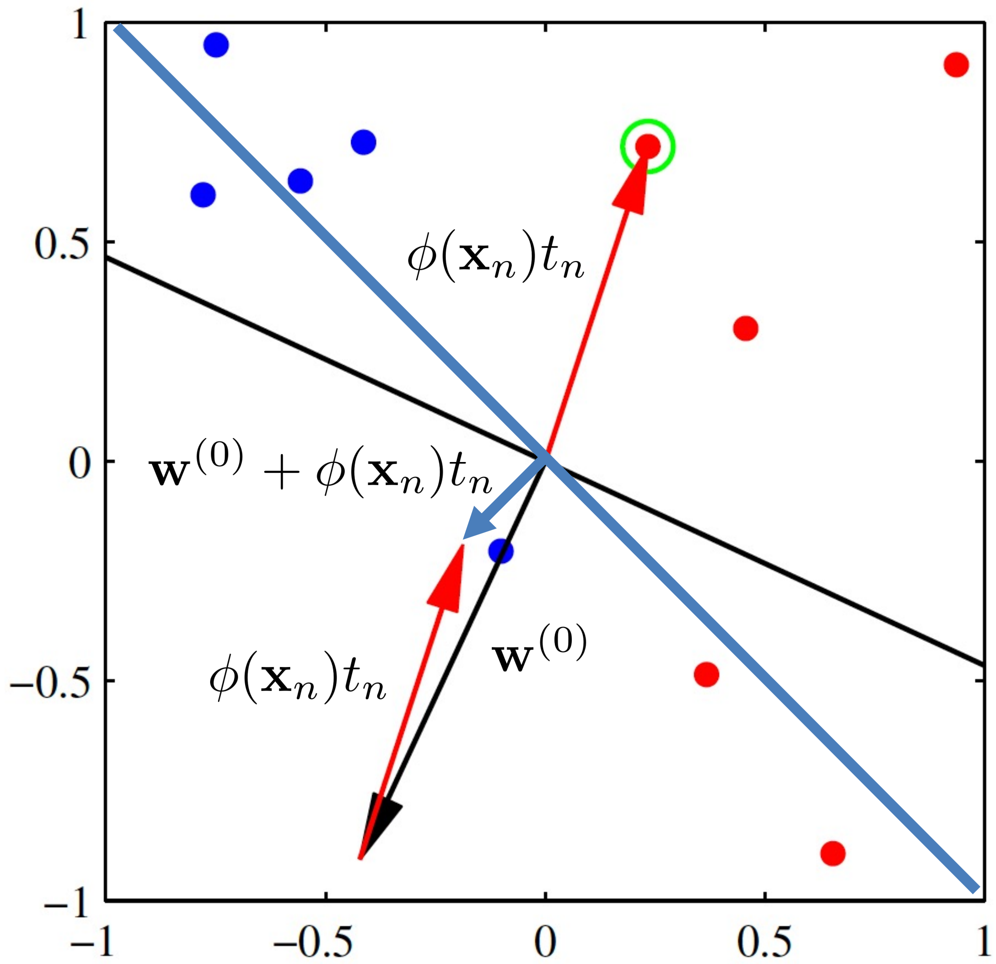
  - Assume there exists unit length w and some γ > 0 such that

$$\mathbf{w}^T \phi(\mathbf{x}_n) \cdot t_n > \gamma, \forall n$$

  - The total number of mistakes the perceptron algorithm makes is at most

$$(D/\gamma)^2$$

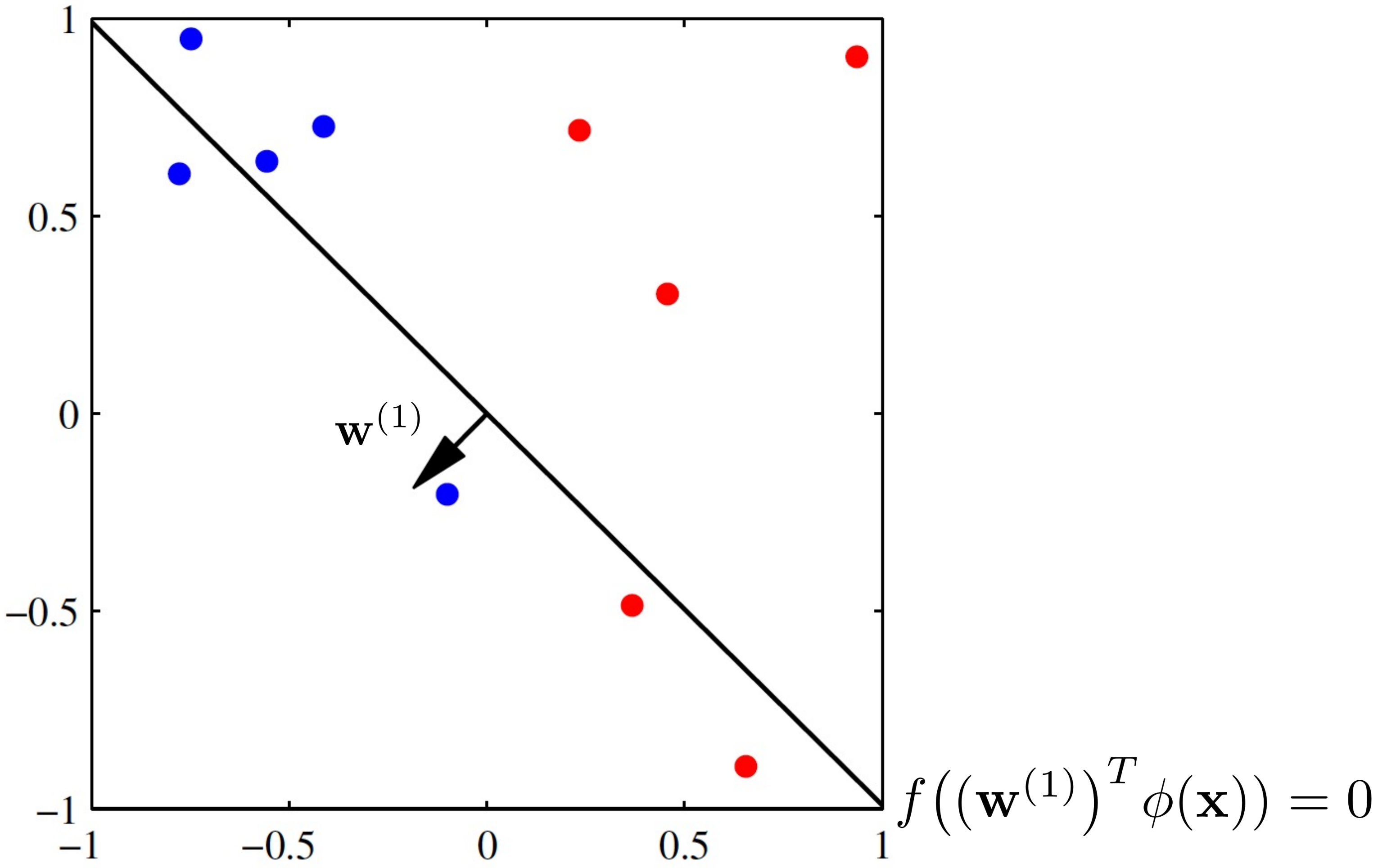
Perceptron Algorithm: Example



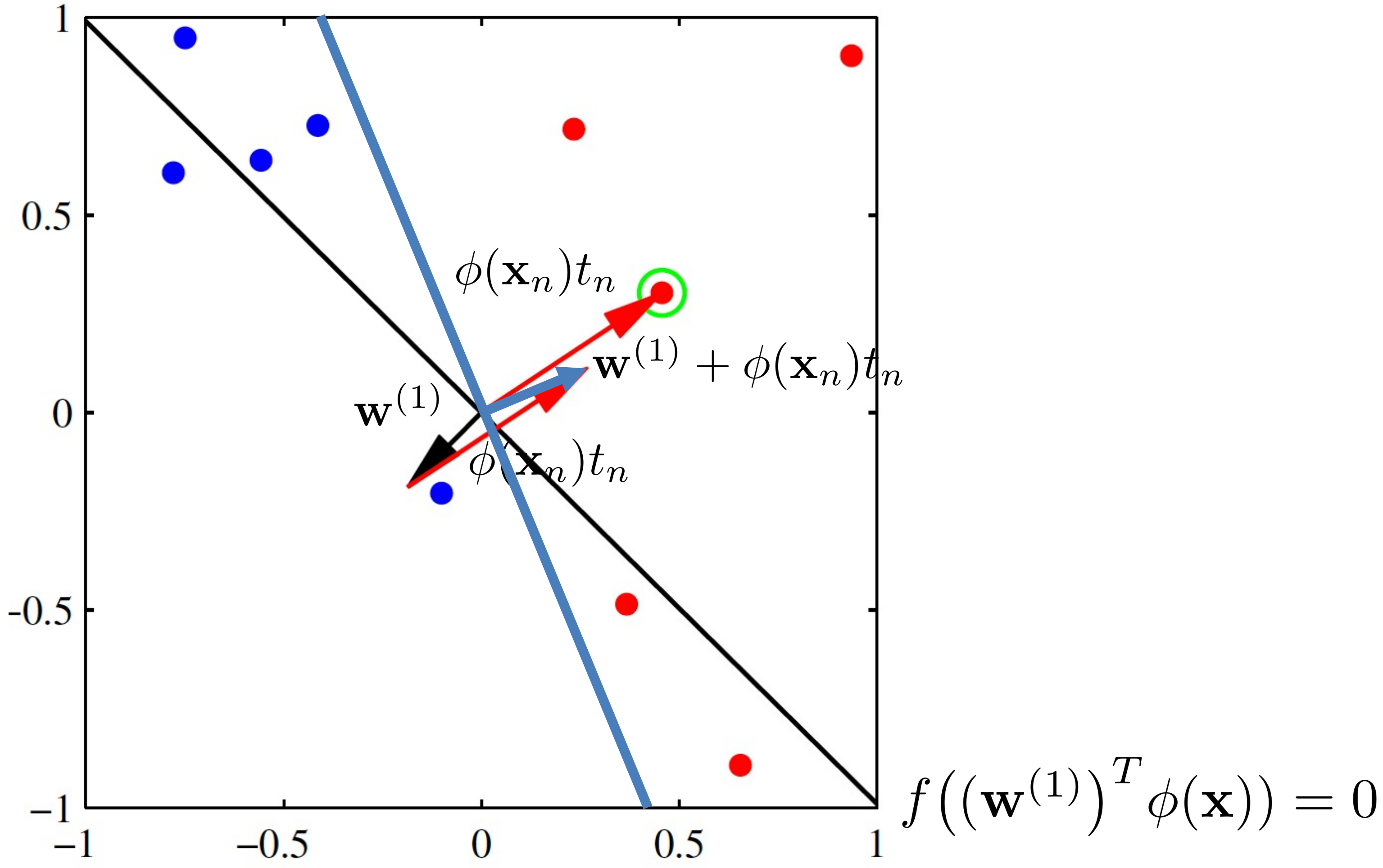
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \phi(\mathbf{x}_n)t_n$$

$$f\left((\mathbf{w}^{(0)})^T\phi(\mathbf{x})\right) = 0$$

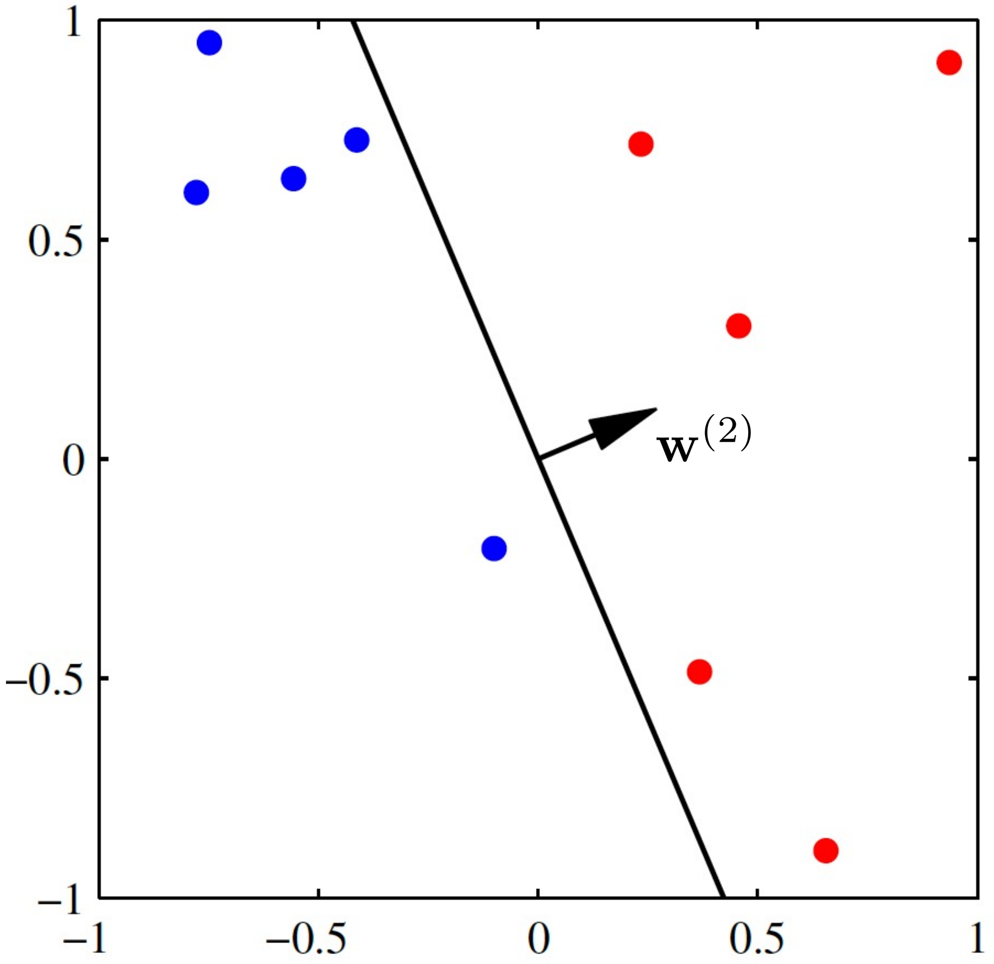
Perceptron Algorithm: Example



Perceptron Algorithm: Example



Perceptron Algorithm: Example



$$f\left(\left(\mathbf{w}^{(2)}\right)^T \phi(\mathbf{x})\right) = 0$$

Perceptron: Pros & Cons

- Pros
  - Easy to implement
  - Time/memory efficient
  - Guaranteed performance when data points are linearly separable

- Cons
  - Sensitive to initialized parameter vector
  - Only applicable to binary classification
  - NEVER converge when data points are not linearly separable