

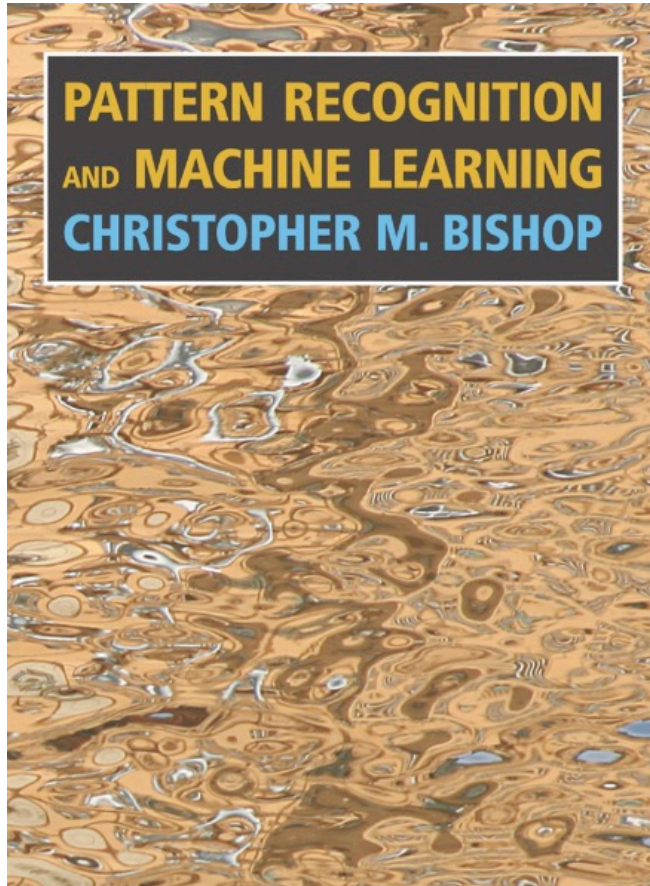
# Linear Classifiers

---

Song Liu ([song.liu@bristol.ac.uk](mailto:song.liu@bristol.ac.uk))

# Reference

---



Today's class *roughly* follows Chapter 4-4.2.

Pattern Recognition and  
Machine Learning

Christopher Bishop, 2006

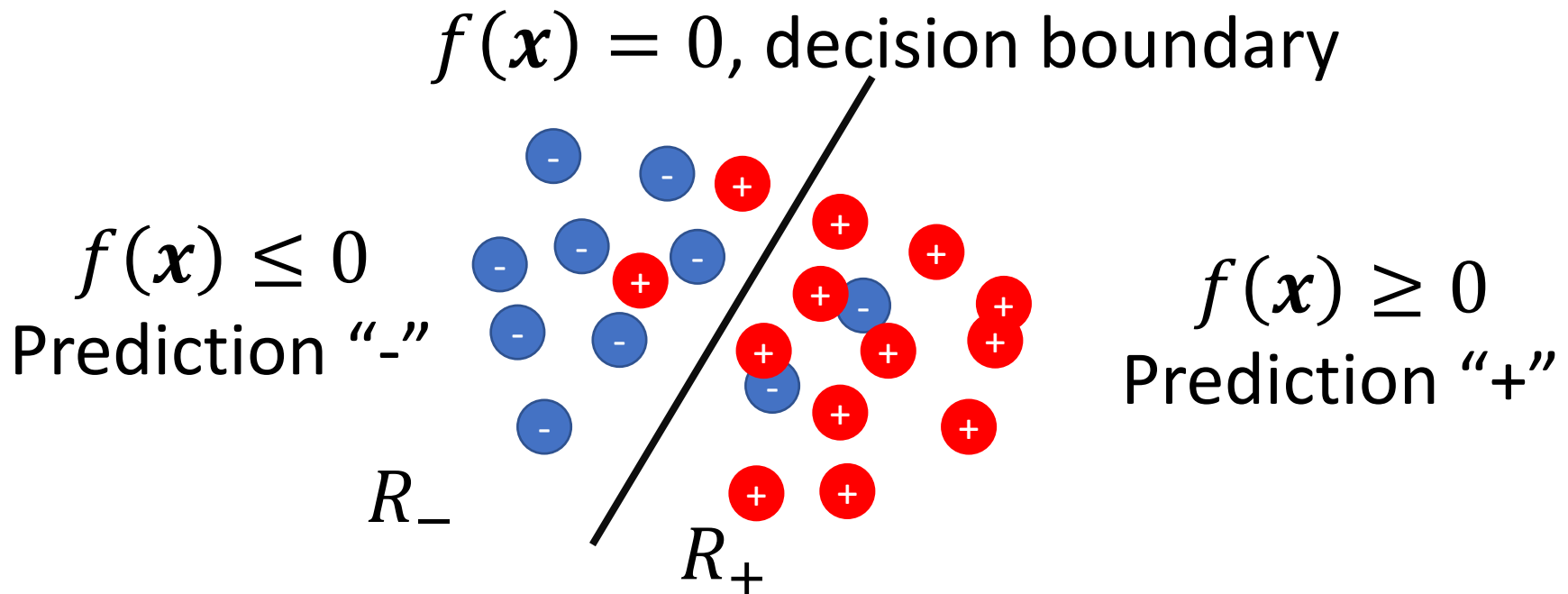
# Outline

---

- Geometry of decision function
- **Non probabilistic classifiers**
  - Least square classifier
  - Fisher discriminant analysis
- **Probabilistic classifiers**
  - Generative Classifiers

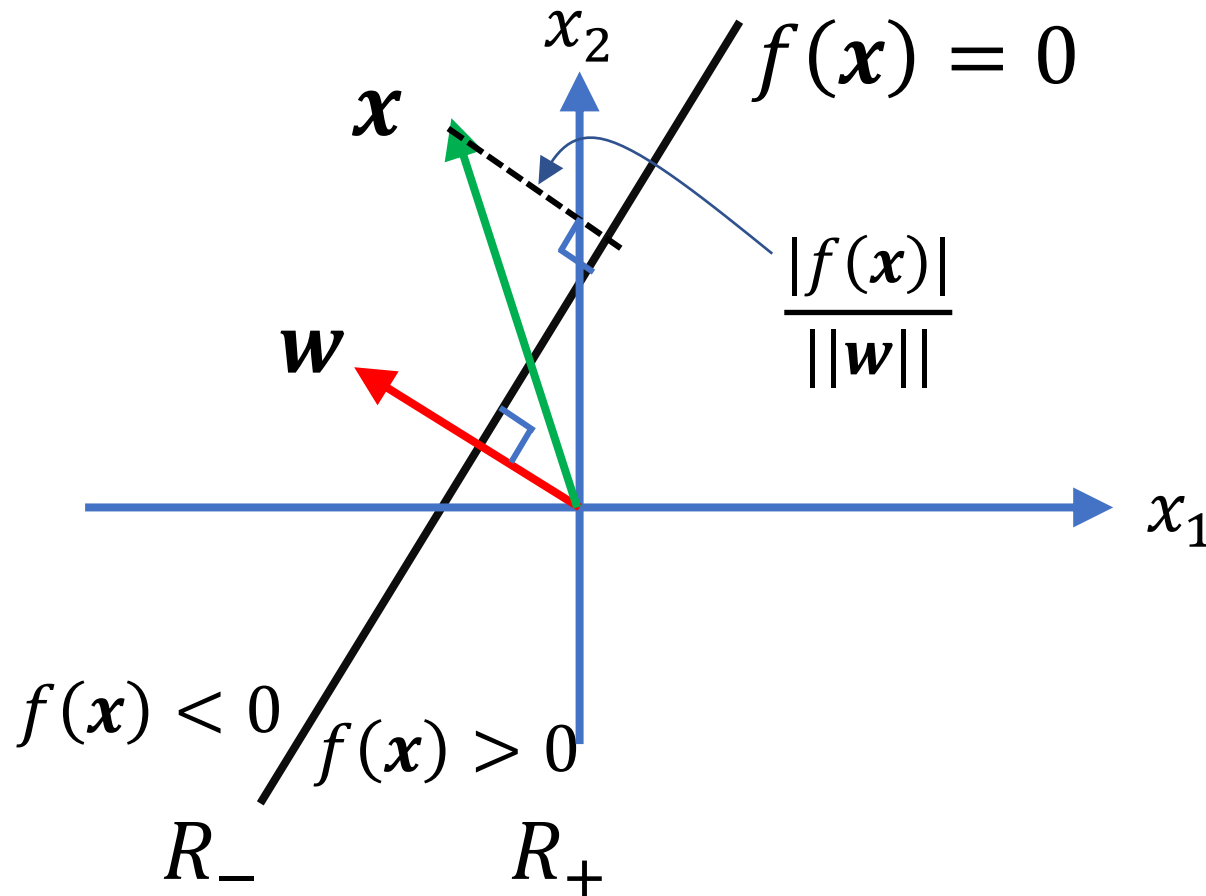
# Binary Classification

- **Input:**  $\{\mathbf{x}_i\}_{i=1}^n$
- **Output:**  $y \in \{-1, +1\}$
- A decision boundary is defined by a function  $f(\mathbf{x})$



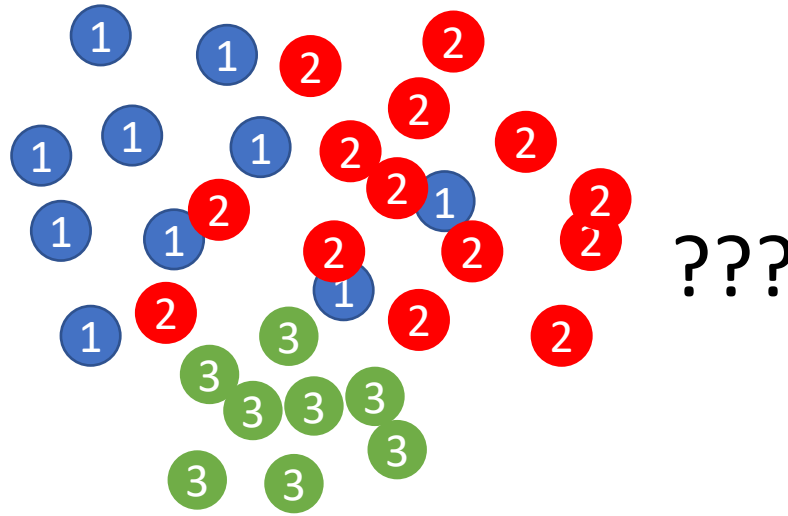
# Geometry of Binary Classification

Suppose  $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + w_0$



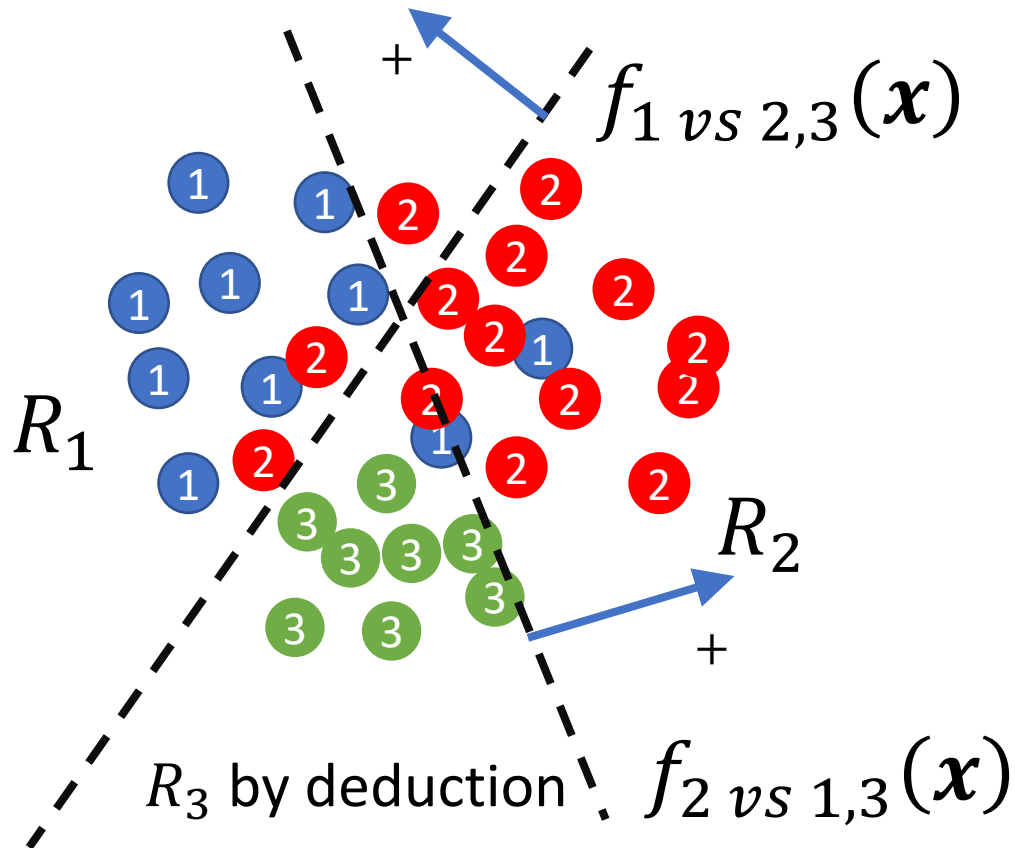
# Multi-class Classification

- **Input:**  $\{\mathbf{x}_i\}_{i=1}^n$
- **Output:**  $y \in \{1 \dots K\}$
- The geometry gets a lot more complicated...
  - Cannot simply check the sign of a single  $f(\mathbf{x})$ .



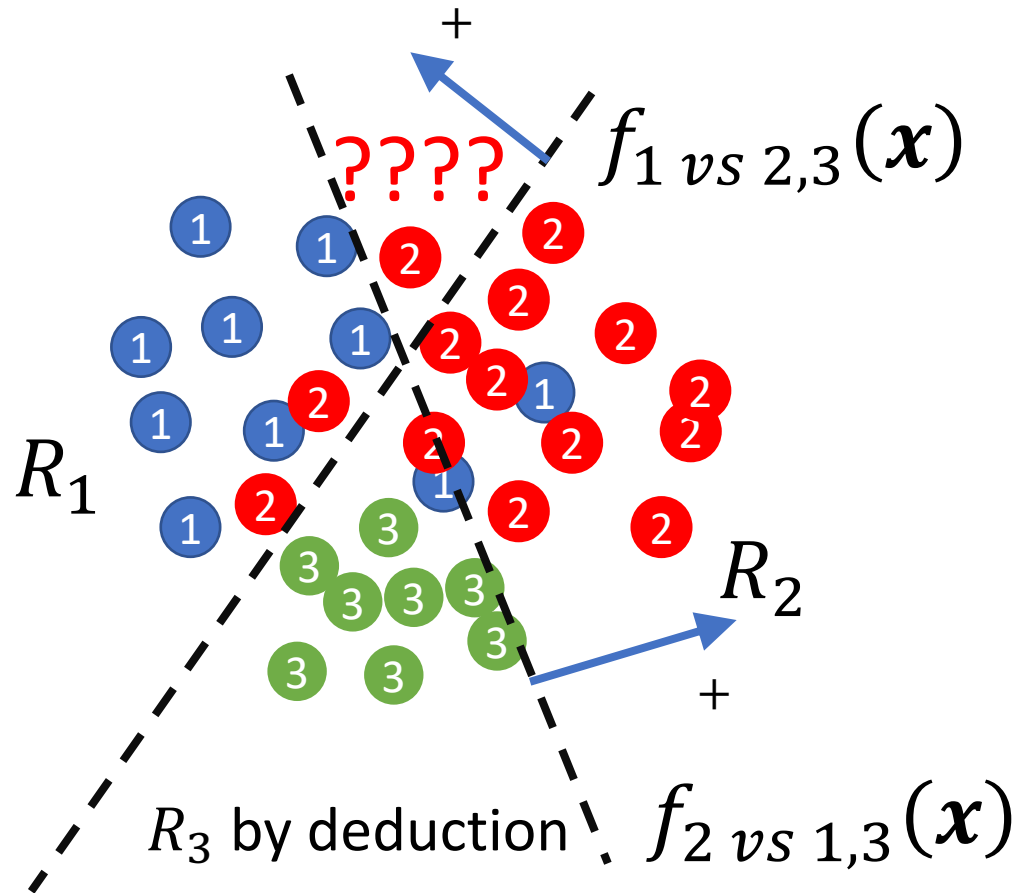
# One versus The Other

- Construct  $K - 1$  classifiers
- Classify Class  $k$  vs. the rest of classes



# One versus The Other

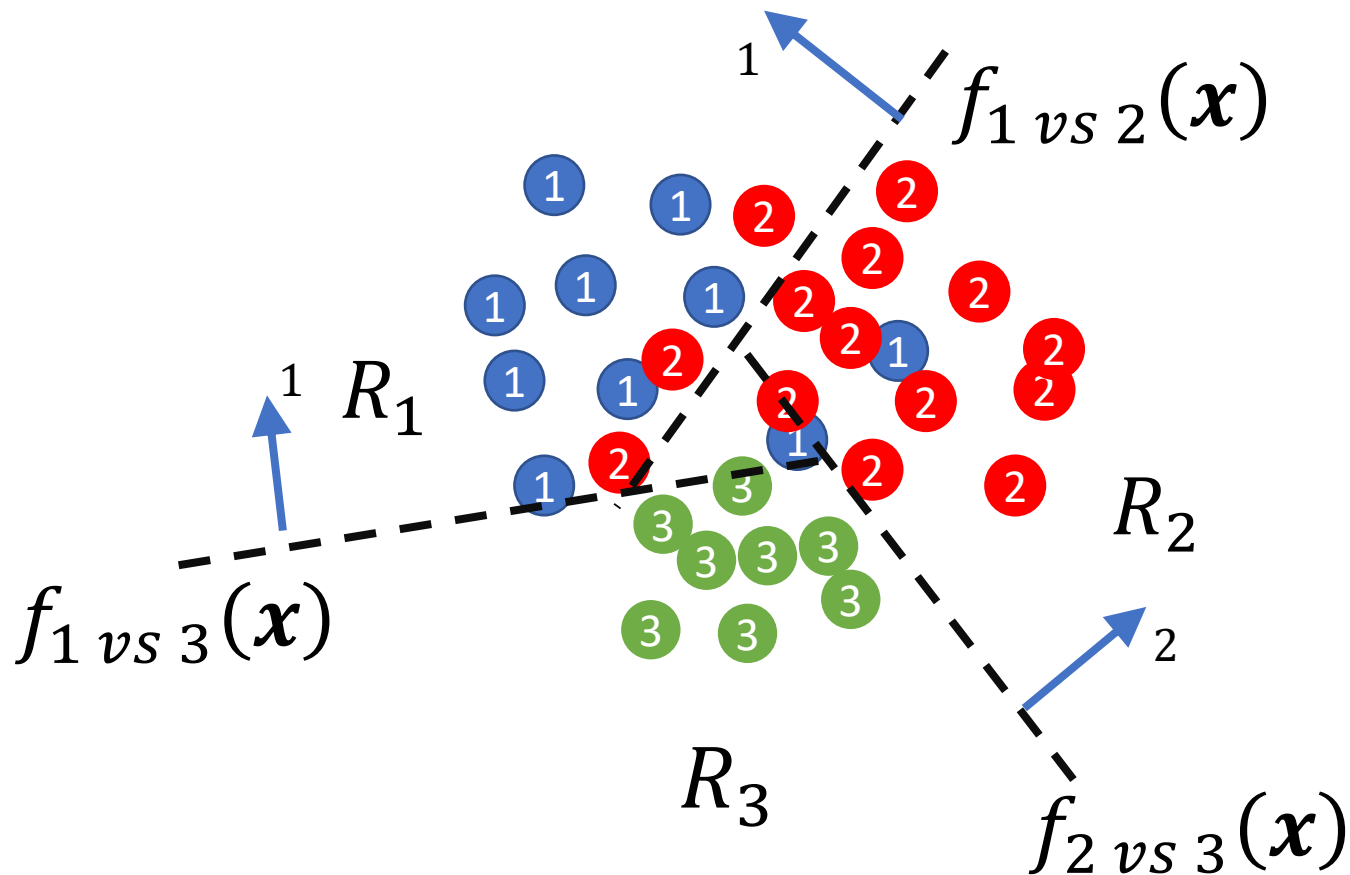
- One versus the other also creates confusion!





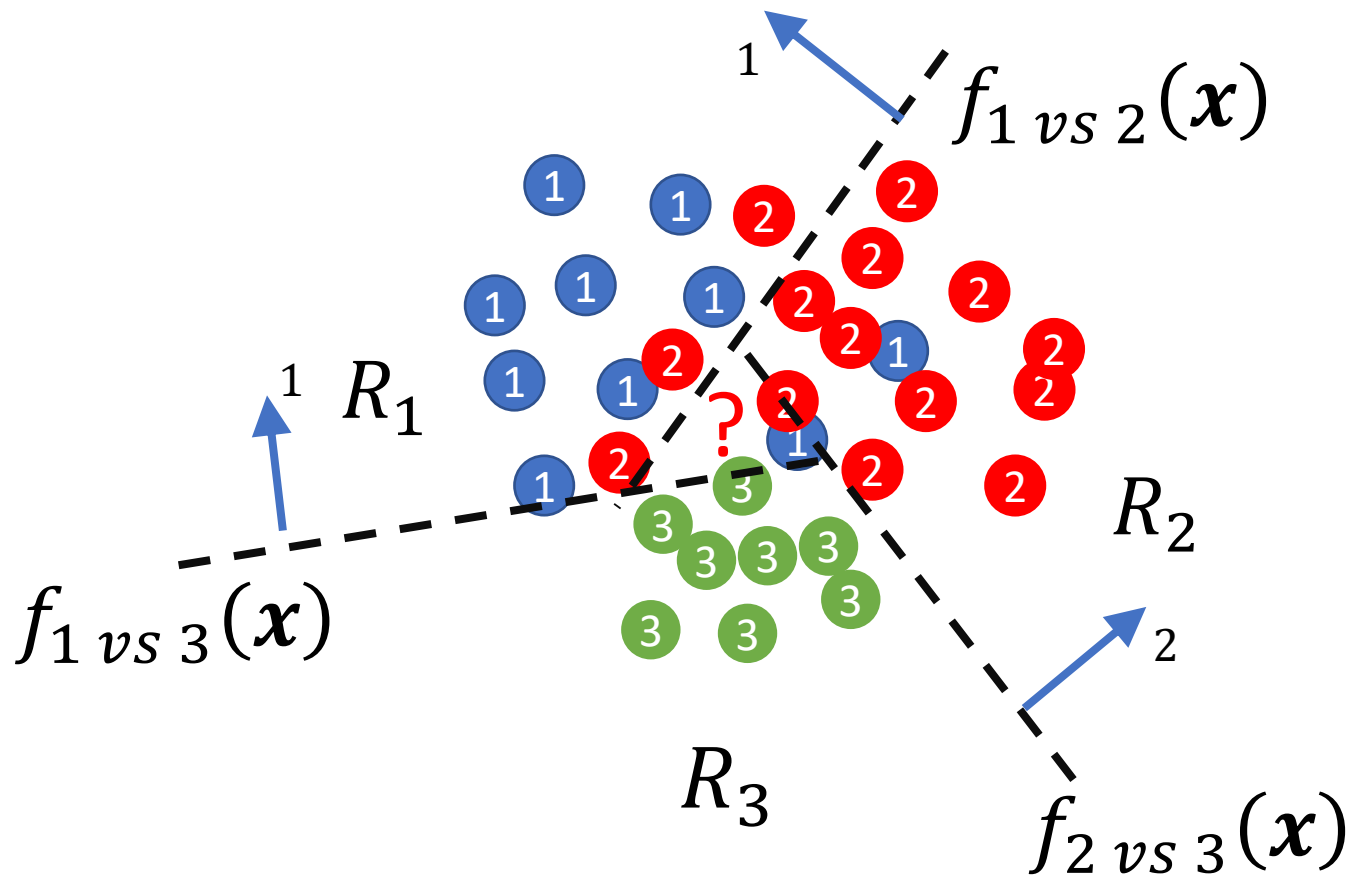
# One versus One

- We can create pairwise classifiers and check majority vote.



# One versus One

- One versus one creates confusion as well...



# Multi-class Classification

---

- Rather than relying on sign of  $f$  to make predictions, we estimate  $K$  functions:
- $\{f_k(\mathbf{x}; \mathbf{w}_k)\}_{k=1}^K$
- Given an  $\mathbf{x}$ , prediction is  $\hat{k} = \max_k f_k(\mathbf{x}; \mathbf{w}_j)$ ,
- **Problem:**  $f_k$  does not have a simple geometry interpretation anymore.
- However,  $f_k$  does have probabilistic interpretation.

How do we obtain the  
prediction function  $f$ ?

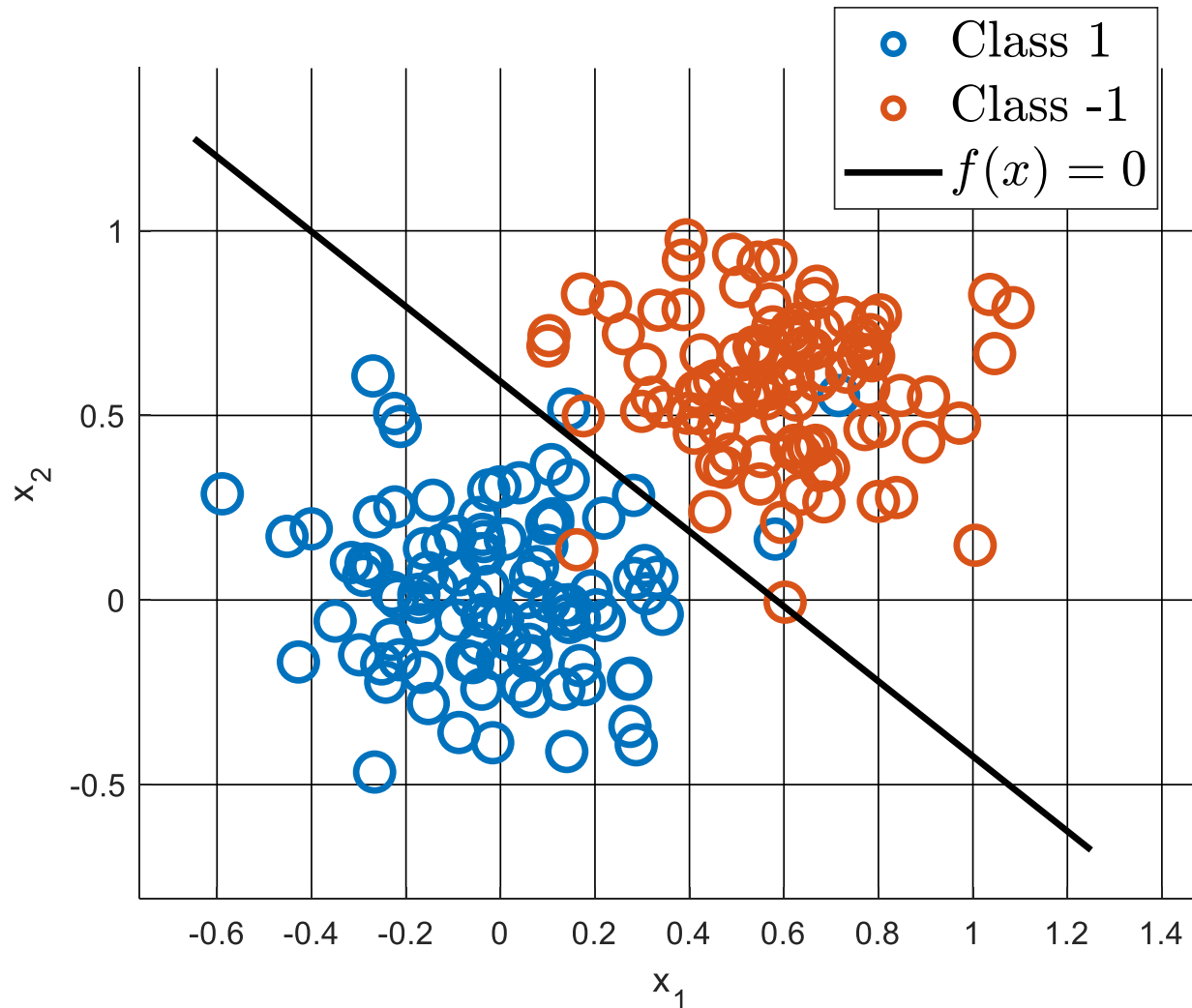
Can we use least squares?

# Least Square Classifier

---

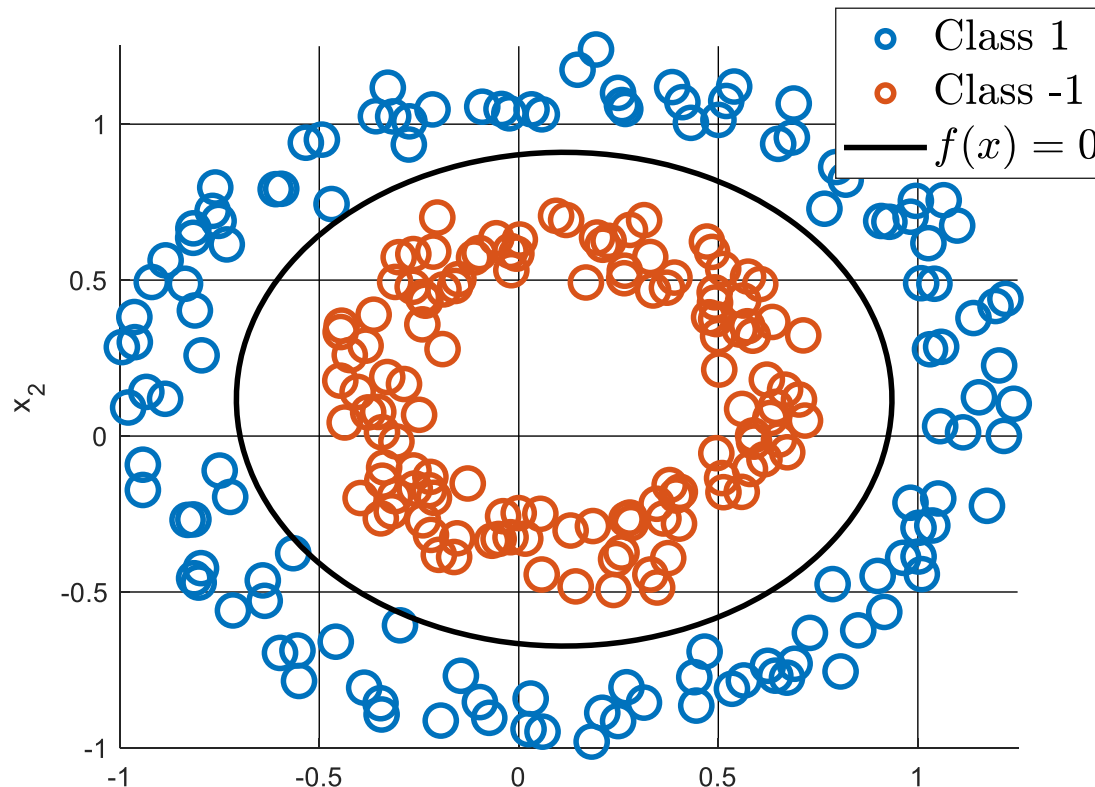
- For binary classification, import  $D$  in LS.
- $\mathbf{w}_{LS} := \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i \in D} [y_i - f(\mathbf{x}_i; \mathbf{w})]^2$ 
  - Now  $y_i$  takes binary value 1 or  $-1$
- Prediction function  $f(\mathbf{x}_i; \mathbf{w}_{LS})$ .
- The predicted label  $\hat{y} := \operatorname{sign}(f(\mathbf{x}_i; \mathbf{w}_{LS}))$

# Least Square Classifier



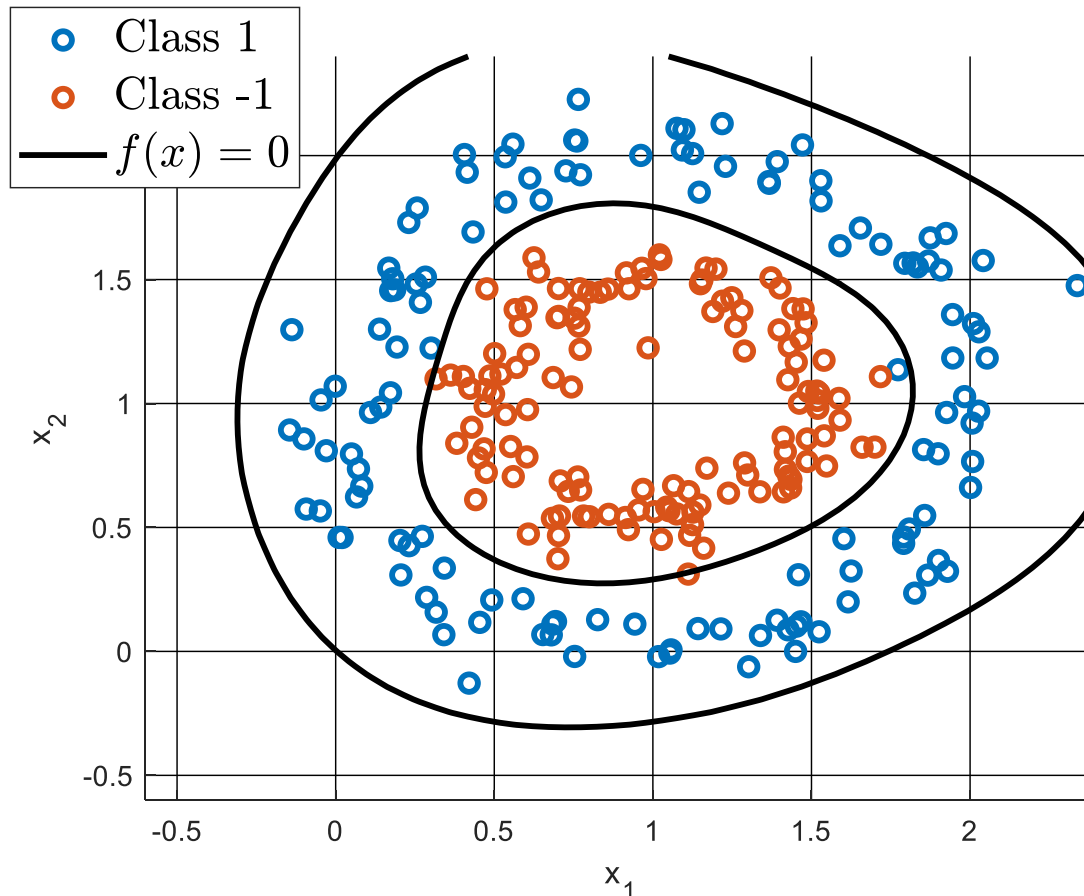
# Least Square Classifier

- You can use feature transform  $\phi$  for  $f$  as well.
- $f(x; \mathbf{w}) := \langle \mathbf{w}, \phi(x) \rangle$ ,
- e.g. poly., trigonometric, RBF, kernel.



# Least Square Classifier

Data may not be separable in the original space but can be separable in the **feature space** created by  $\phi$ !



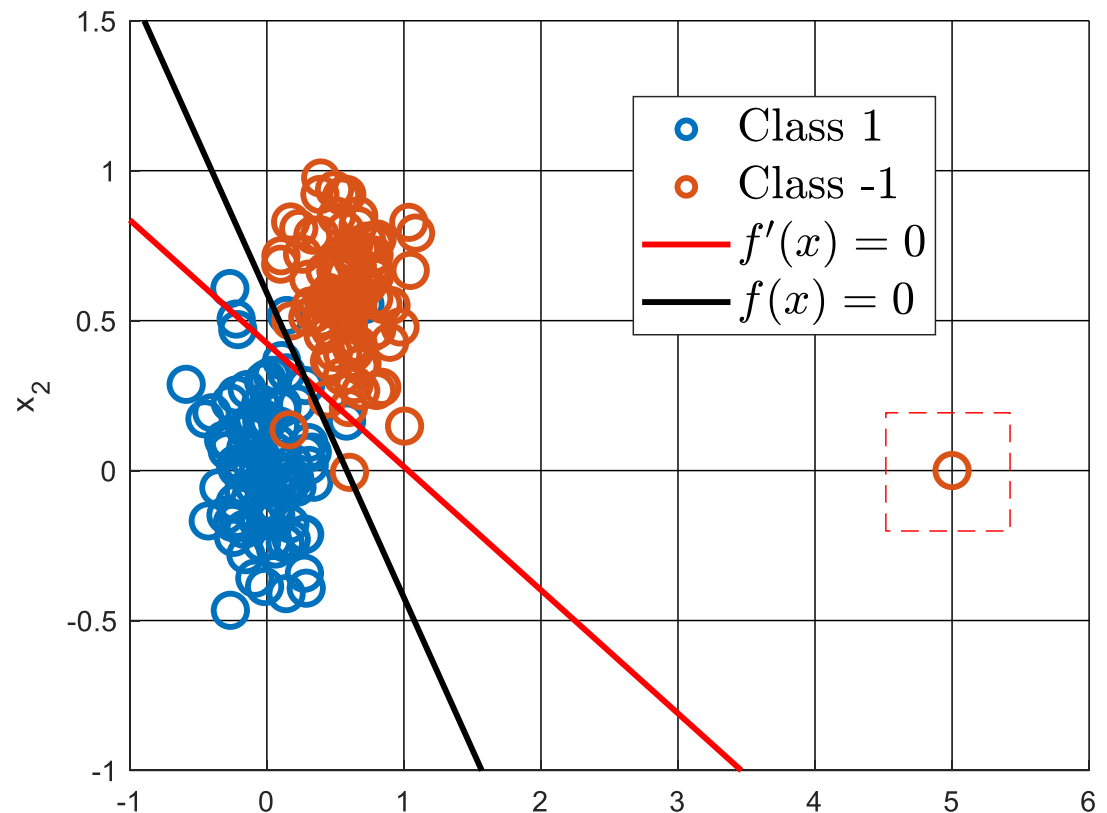
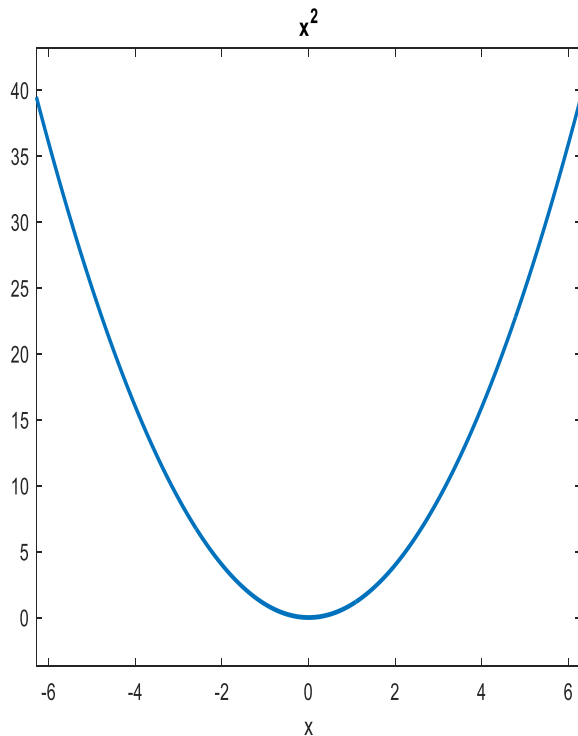


# Multi-class LS classification

- LS can be adapted to multi-class classification.
- Suppose output  $y \in \{1 \dots K\}$
- Replace  $y_i = k$  in  $D$  with  $\mathbf{t}_i \in \{0,1\}^K$ .
  - $t_i^{(k)} = 1$ .
  - $t_i^{(j)} = 0, \forall j \neq k$
- **“One-hot encoding”**
- $\mathbf{W}_{\text{LS}} := \operatorname{argmin}_{\mathbf{W}} \sum_{i \in D} \|\mathbf{t}_i - \mathbf{W}^\top \tilde{\mathbf{x}}_i\|^2$
- $\mathbf{W}_{\text{LS}} \in R^{(d+1) \times K}, \tilde{\mathbf{x}}_i := [\mathbf{x}_i^\top, 1]^\top, \in R^d$ .

# Why not to use LS Classifier?

- Square loss does not make sense in classification tasks.
- Data point far away from decision boundary can influence the decision boundary by a lot.



# Why not to use LS Classifier?

---

- Unlike LS regression,
- LS classification lacks a probabilistic interpretation.
- It cannot be interpreted as Maximum Likelihood of some probabilistic model on  $D$ .

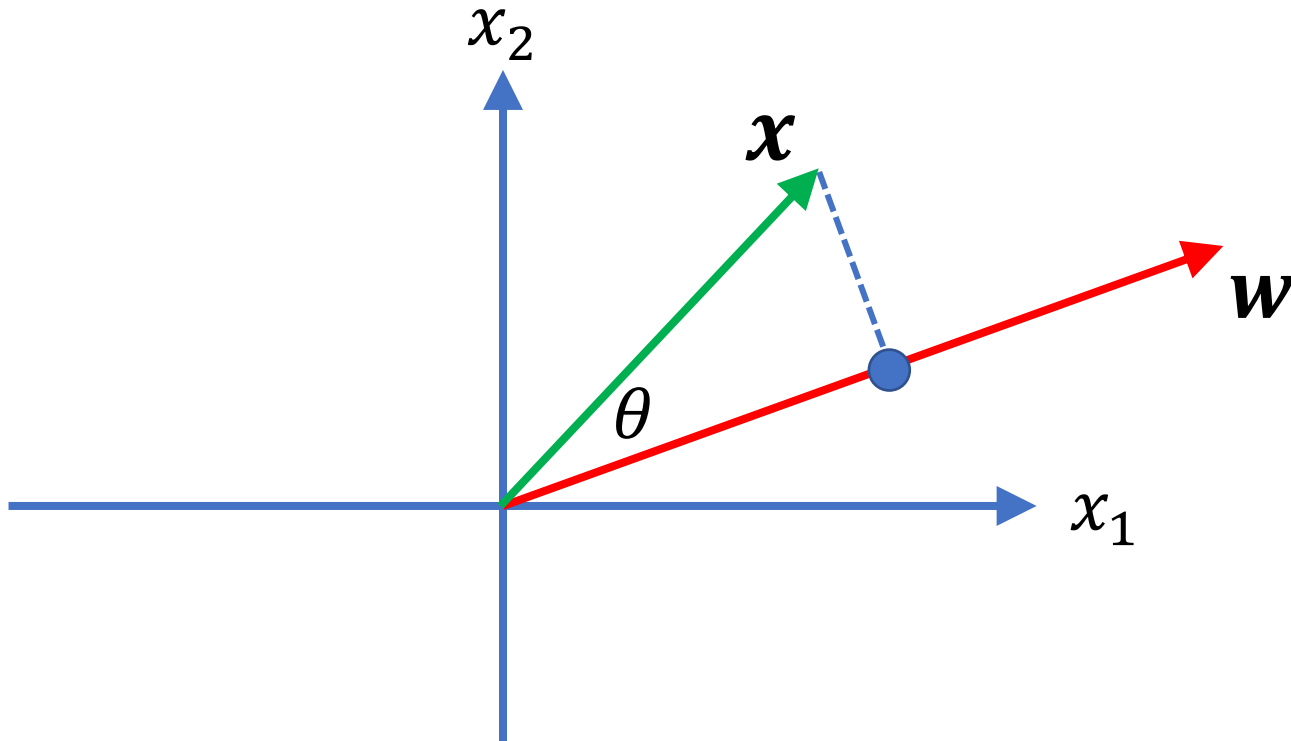
# Fisher Discriminant Analysis (FDA)

---



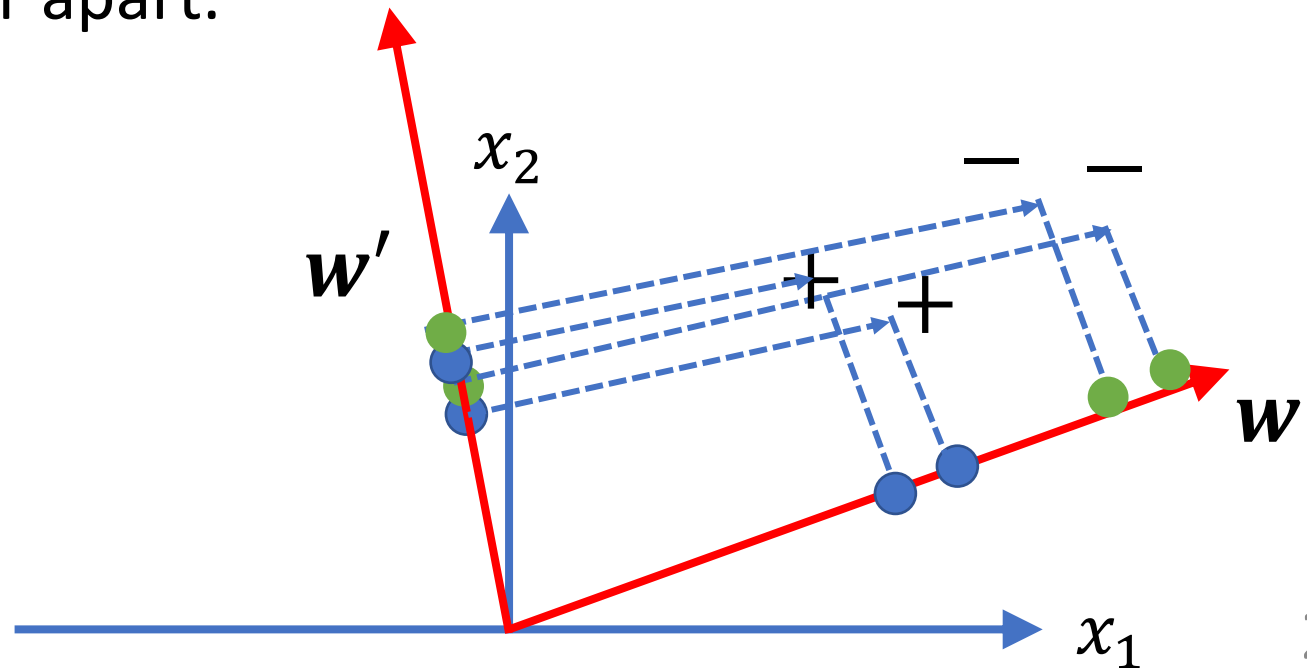
# Embedding by Inner Product

- The inner product  $\langle \mathbf{w}, \mathbf{x} \rangle$  “embeds”  $\mathbf{x}$ , onto a one-dimensional line along  $\mathbf{w}$  direction.

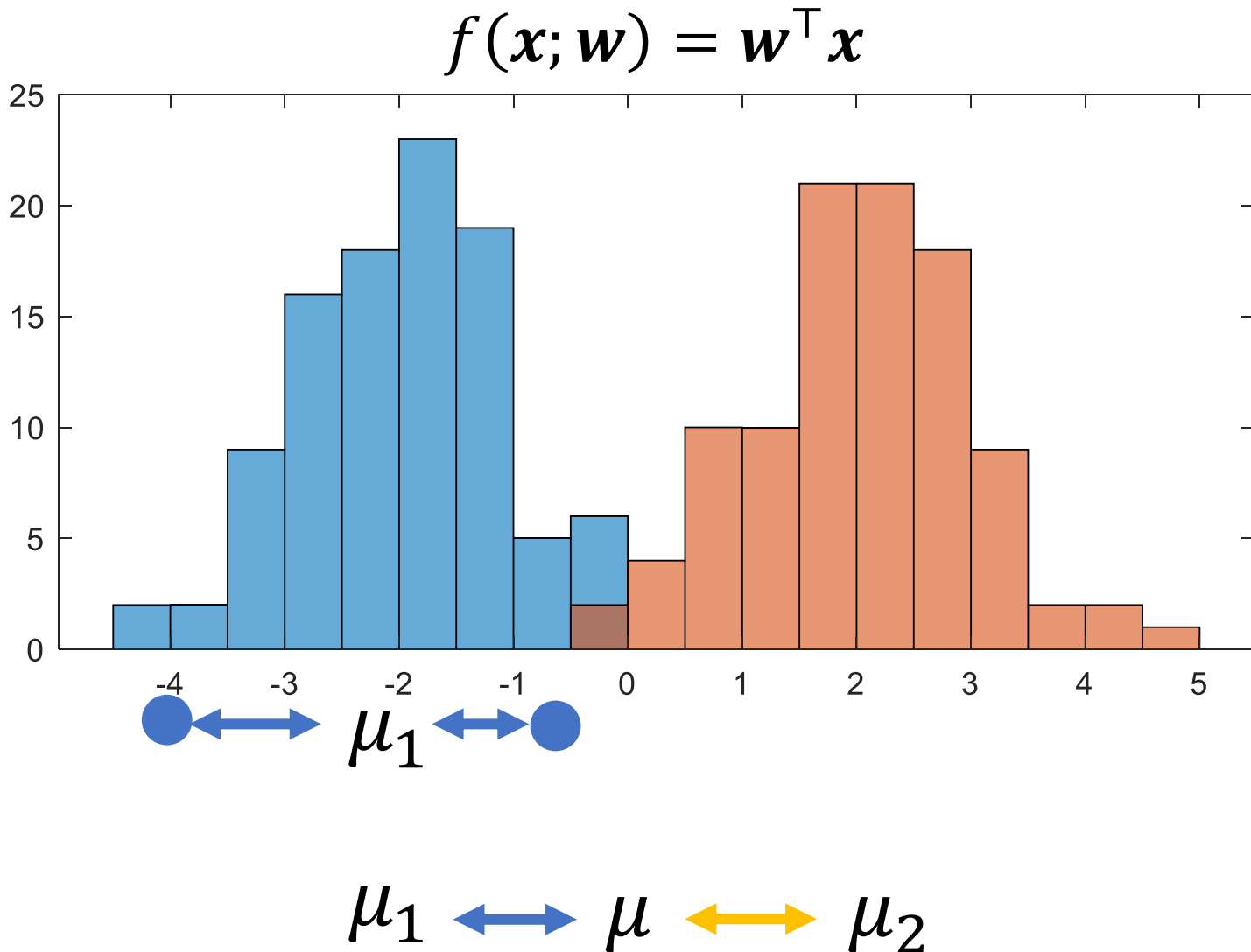


# Embedding by Inner Product

- What would be a good embedding?
- Clearly, we prefer  $\mathbf{w}$  to  $\mathbf{w}'$ , as the embedding is **more separated** between  $+$  and  $-$ .
- We want points within the class close, but points between two classes far apart.



# Within Class and Between Class Scatterness



# Within-class Scatterness

---

- Embedding is  $\mathbf{w}^\top \mathbf{x}$ .
- Embedded center for class  $k$ :
  - $\hat{\mu}_k = \frac{1}{n_k} \sum_{i, \mathbf{y}_i=k} \mathbf{w}^\top \mathbf{x}_i$
- Within class scatterness of class  $k$ :
- $S_{\mathbf{w},k} = \sum_{i, \mathbf{y}_i=k} (\mathbf{w}^\top \mathbf{x}_i - \hat{\mu}_k)^2$ 
  - Sum over points in **individual** classes.



# Between-class Scatterness

---

- Embedded dataset center:

- $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{w}^\top \mathbf{x}_i$

- Between-class scatterness

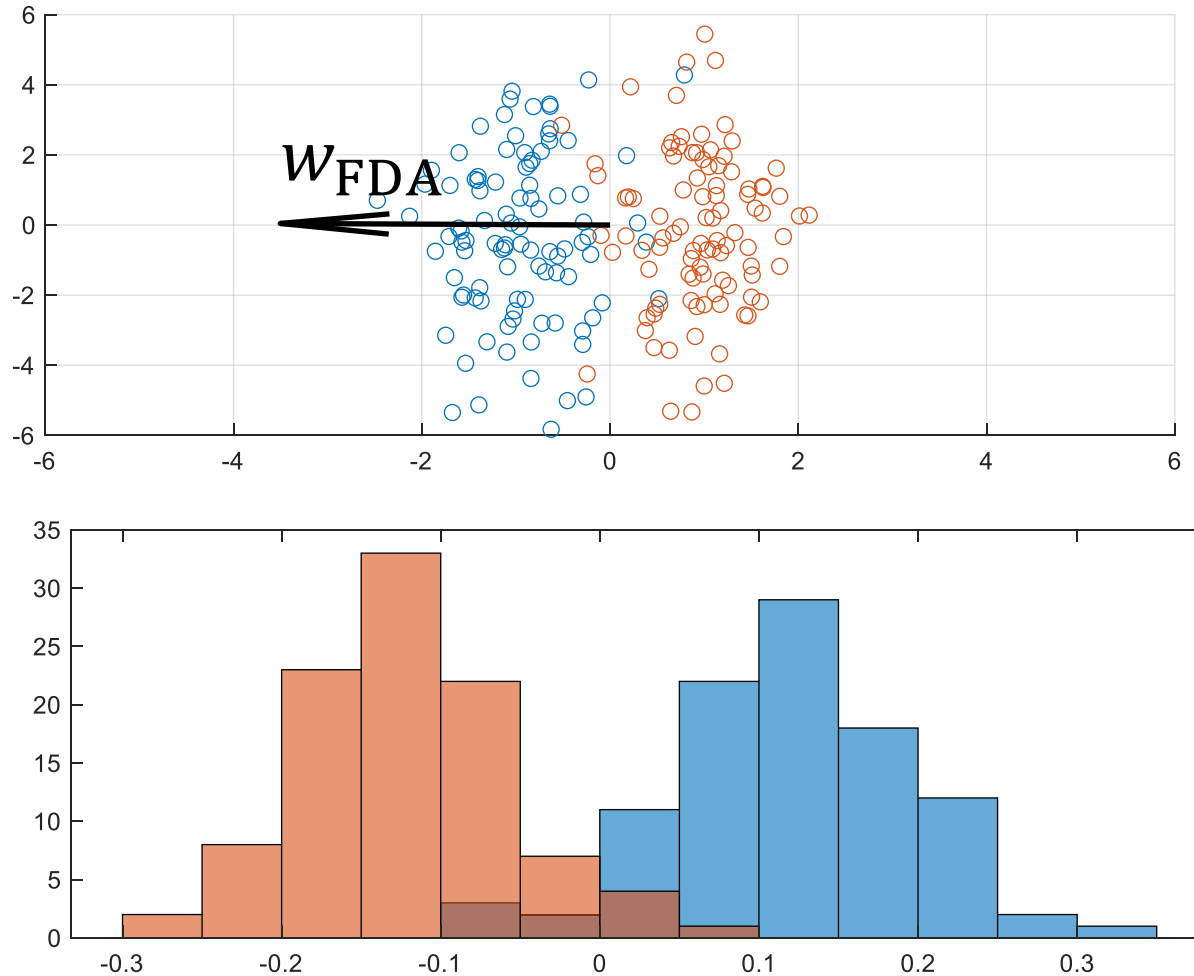
- $s_{b,k} = n_k (\hat{\mu}_k - \hat{\mu})^2$

- $n_k$  is needed to make  $s_{b,k}$  at the same scale with  $s_{w,k}$ .

# Fisher Discriminant Analysis

- **Maximizing** between class scatterness  $\forall_k$ .
- **Minimize** within class scatterness  $\forall_k$ .
- $\max_w \boxed{\sum_k s_{b,k}} / \boxed{\sum_k s_{w,k}}$
- If  $K = 2$ , this has a simple solution that
- $\mathbf{w} := \mathbf{S}_w^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$ ,  $\mathbf{S}_w := \sum_{k=1}^K \mathbf{S}_k$
- $\mathbf{S}_k$  is **sample covariance** of class  $k$  times  $n_k$ .
- Read PRML 4.14 for its derivation

# Example of FDA



# Fisher Discriminant Analysis

---

- However, FDA does not learn a decision function  $f$ .
- $f(\mathbf{x}; \mathbf{w}_{\text{FDA}}) = \langle \mathbf{w}_{\text{FDA}}, \mathbf{x} \rangle$  obtained by FDA cannot be directly used for making a prediction:
- In general,  $f(\mathbf{x}; \mathbf{w}_{\text{FDA}}) > 0$  does not mean  $\mathbf{x}$  is predicted as positive or negative data point: FDA does not care about classification accuracy, a.k.a., minimizing FP or FN.

# Probabilistic Generative Classifiers

# Probabilistic Classification

- How to put classification problem under a prob. framework?

- **Minimize Expected Loss:**

$$\hat{y} := \operatorname{argmin}_{y_0} \mathbb{E}_{p(y|\mathbf{x})} [L(y, y_0) | \mathbf{x}]$$

- We need:  $p(y|\mathbf{x})$ ,  $y \in \{1, \dots, K\}$
- **Discriminative:** Infer  $p(y|\mathbf{x})$  directly.
- **Generative:** Infer  $p(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(y)$ , infer  $p(\mathbf{x}|y)$ !

# Continuous Input Variable

- To infer  $p(\mathbf{x}|y)$ , we need a model.
- If  $\mathbf{x}$  is continuous, MVN is a natural choice for  $p(\mathbf{x}|y)$ .
- **Model**  $p(\mathbf{x}|y = k; \mathbf{w}) := N_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Assuming IID-ness over all  $\mathbf{x}_i$ , and shared covariance  $\boldsymbol{\Sigma}$
- **Write down the likelihood** over  $D$ :
- $p(D|\mathbf{w}) = \prod_{i \in D} p(\mathbf{x}_i, y_i|\mathbf{w}) = \prod_{i \in D} p(\mathbf{x}_i|y_i; \mathbf{w})p(y_i)$   
$$= \prod_{i \in D} N_{\mathbf{x}_i}(\boldsymbol{\mu}_{y_i}; \boldsymbol{\Sigma}) p(y_i)$$

# Continuous Input Variable

$$\bullet \hat{\boldsymbol{\mu}}_{1\dots K}, \hat{\boldsymbol{\Sigma}} := \arg \max_{\boldsymbol{\mu}_{1\dots K}, \boldsymbol{\Sigma}} \sum_{i \in D} \log N_{\mathbf{x}_i}(\boldsymbol{\mu}_{y_i}; \boldsymbol{\Sigma}) p(y_i)$$

1. Plug in estimates for  $p(y_i = k)$ , which is  $\frac{n_k}{n}$ .
2. Now work out the MLE for  $\hat{\boldsymbol{\mu}}_k := \frac{1}{n_k} \sum_{i \in D, y_i = k} \mathbf{x}_i$
3. Plug in  $\hat{\boldsymbol{\mu}}_k$  to work out

$$\hat{\boldsymbol{\Sigma}} := \sum_{k=1\dots K} \frac{n_k}{n} \underbrace{\frac{1}{n_k} \sum_{i \in D, y_i = k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top}_{\text{MLE of covariance of individual classes!}}$$

MLE of covariance of individual classes!



# Linear Decision Boundary

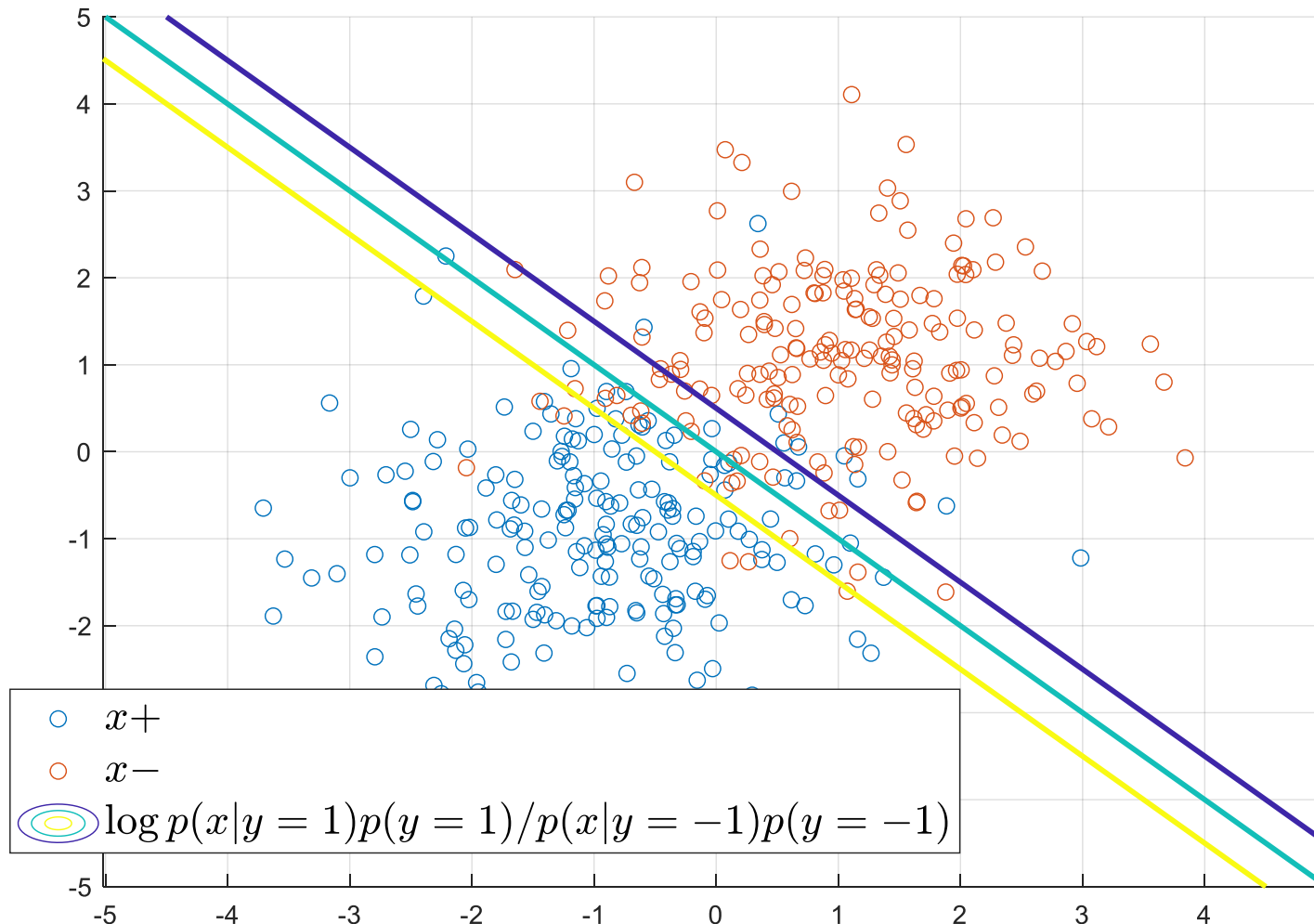
- **Prediction:**  $\hat{y} := \operatorname{argmax}_y p(y|\mathbf{x}; \hat{\mathbf{w}}) \propto p(\mathbf{x}|y; \hat{\mathbf{w}})p(y)$
- **Prove:** when using shared covariance matrix MVN model, the decision boundary is piecewise-linear.
- The decision boundary is
$$\{\mathbf{x} | p(y = k | \mathbf{x}; \hat{\mathbf{w}}) = p(y = k' | \mathbf{x}; \hat{\mathbf{w}})\}$$
$$\forall k \neq k'$$

Which is the same as the set

$$\left\{ \mathbf{x} \left| \frac{p(\mathbf{x}|y = k; \hat{\mathbf{w}})p(y = k)}{p(\mathbf{x}|y = k'; \hat{\mathbf{w}})p(y = k')} = 1 \right. \right\}$$
$$\forall k \neq k'$$

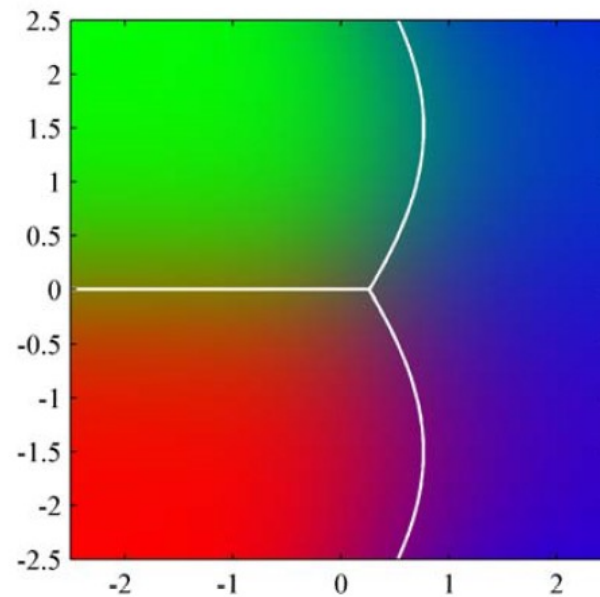
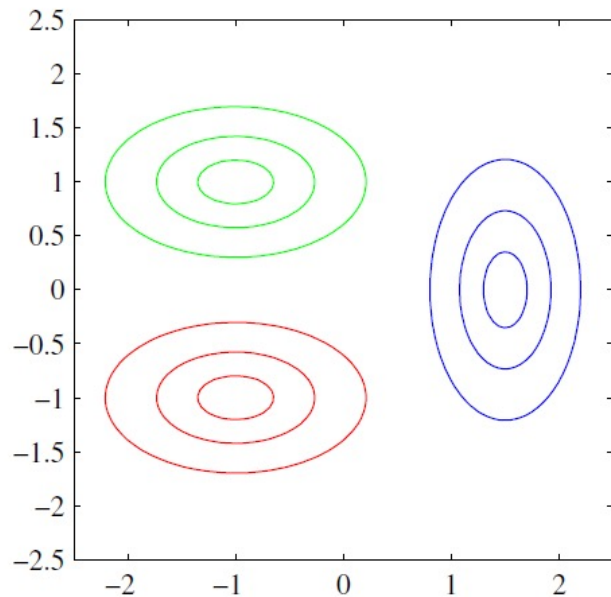
**Hint:** take log on both sides of the equality.

# Linear Decision Boundary



# Continuous Input Variable

- You can also assume for each class  $k$ , there are different covariance matrices  $\Sigma_k$ .
- The MLE reduces to estimating individual  $\mu_k$  and  $\Sigma_k$ .
- **The decision boundary is no longer linear.**



# Discrete Input Variable $x$

- In many classification tasks, we are dealing with discrete variables as  $x$ . For example, in a spam filter,
- $x := [x^{(1)}, \dots, x^{(d)}]^T$  are frequencies of words in a document. This is called “bag of words” representation.
- $y \in \{\text{spam}, \text{ham}\}$ .
- For example, the document “to be or not to be”
- $x := [\text{to} = 2, \text{be} = 2, \text{or} = 1, \text{not} = 1, \text{question} = 0]^T$ .
- $x^{(i)} \in N_0$

# Naïve Bayes

- Assume  $x^{(1)} \dots x^{(d)}$  follows multinomial distribution
- $p(\mathbf{x} = \mathbf{x}_0 | y) \propto \prod_{i=1 \dots d} \beta(i|y)^{x_0^{(i)}}$  up to constant does not depend on  $y$ .

- $\beta(i|y = k)$  is the probability of word  $i$  occurs in class  $k$ .
- It is easy to estimate:

$$\beta(i|y = k) \approx \frac{\sum_{j \in D, y_j = k} x_j^{(i)}}{\sum_{j \in D, y_j = k} \sum_{i=1}^d x_j^{(i)}}$$

- $\beta(\text{to} | y = \text{spam})$  is occurrences of the word “to” in “spam” emails divided by total number of words in “spam” emails in our training dataset.

# Naïve Bayes

- Prediction:  $\hat{y} := \operatorname{argmax}_y p(\mathbf{x} = \mathbf{x}_0 | y) p(y)$
- $p(y = k): \frac{n_k}{n}$
- $p(\mathbf{x} = \mathbf{x}_0 | y) \propto \prod_{i=1 \dots d} \beta(i | y)^{x_0^{(i)}}$ 
  - $\beta(i | y)$  has been obtained by previous counting.
  - Power  $x_0^{(i)}$  is used assuming occur. of words are independent.
- $p(\mathbf{x} = \text{"*to be or not to be*" | } y):$   
 $\beta(\text{to} | y)^2 \beta(\text{be} | y)^2 \beta(\text{or} | y) \beta(\text{not} | y)$

# Conclusion

---

- We have studied classification problem:
- Geometry of decision function
- Least square classifier
- Fisher discriminant analysis
  - Within and between scatterness
- Generative Classifiers:
  - MVN for continuous input variable
  - Naïve Bayes for discrete input variable

# Homework

---

- Prove the statement on page 33.
- (1) Derive the maximum likelihood estimation for parameters in multinomial distribution. (2) Explain the Naïve Bayes classifier using a Maximum Likelihood framework.



# Computing Lab

---

- Implement a version of Perception classifier: “Simplitron”
- Demo.