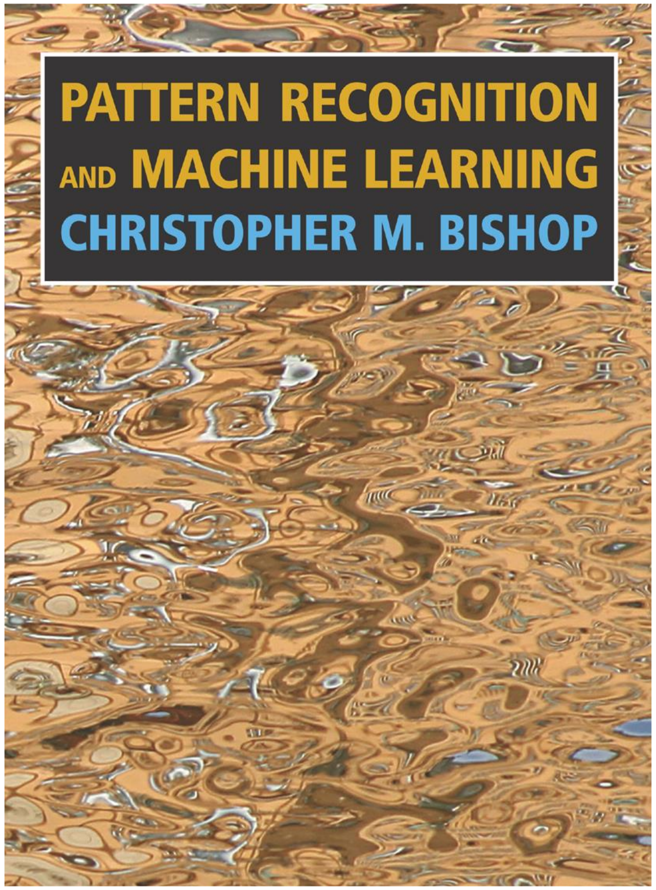
Linear Classifiers

Song Liu (song.liu@bristol.ac.uk)

Reference



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

Pattern Recognition and Machine Learning

Christopher Bishop, 2006

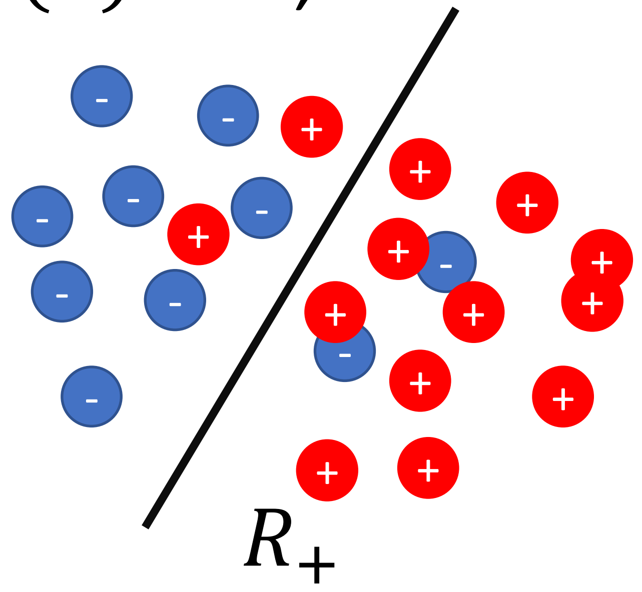
Outline

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

Binary Classification

- **Input**: $\boldsymbol{x} \in R^d$
- **Output**: $y \in \{-1, +1\}$
- A decision boundary is defined by a function $f(\boldsymbol{x})$

$f(\boldsymbol{x}) = 0$, decision boundary



$f(\boldsymbol{x}) \leq 0$
Prediction "-"

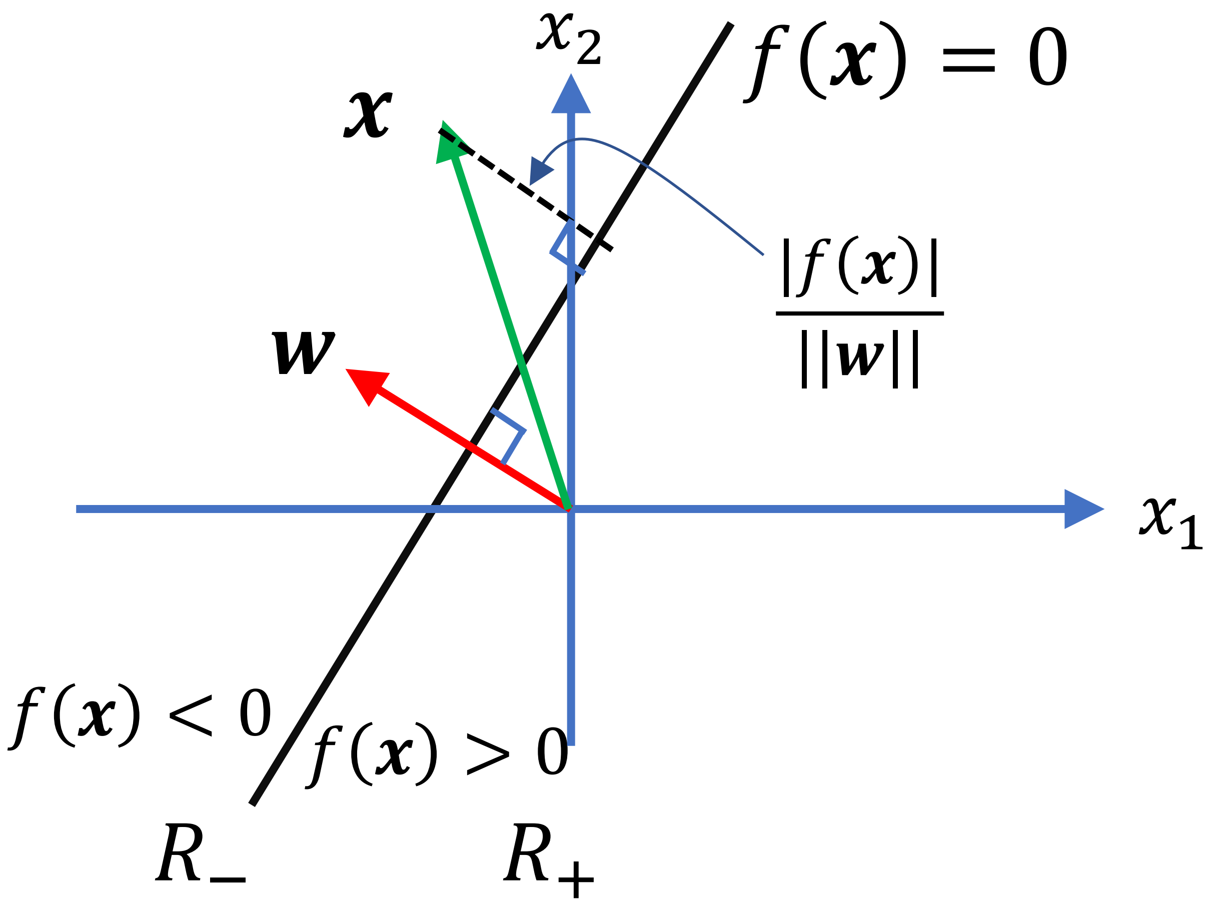
$f(\boldsymbol{x}) \geq 0$
Prediction "+"

R_-

R_+

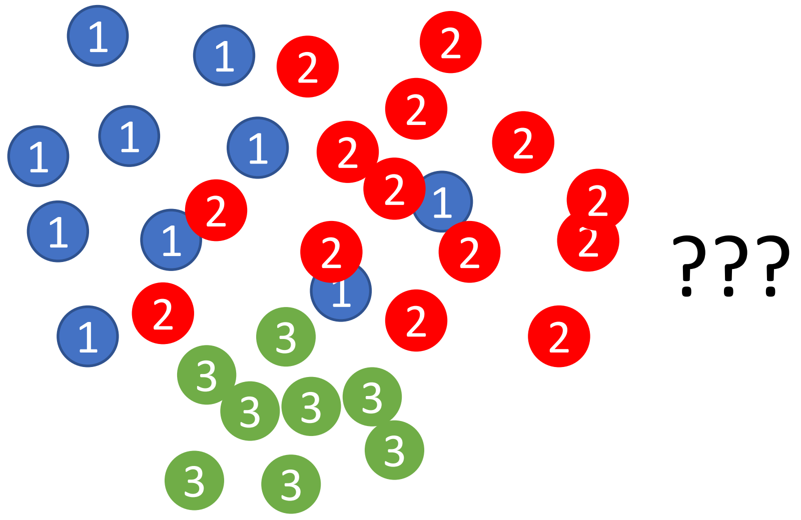
Geometry of Binary Classification

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



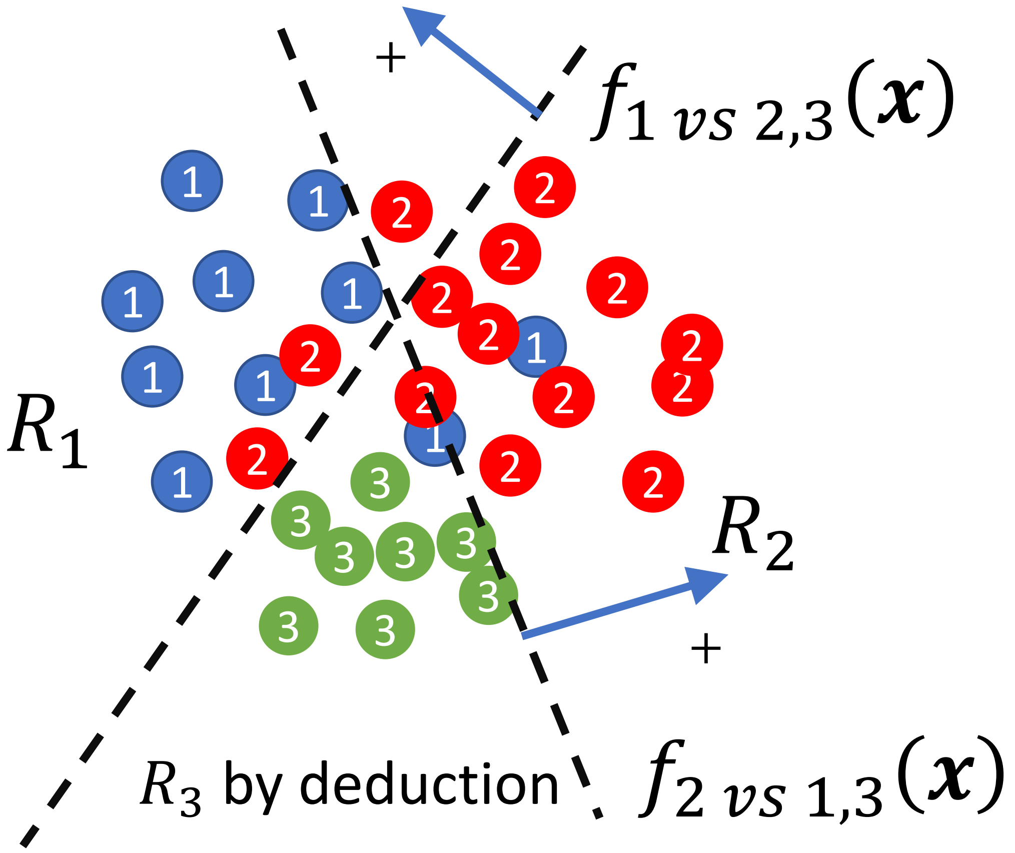
Multi-class Classification

- **Input**: $\boldsymbol{x} \in R^d$
- **Output**: $y \in \{1 \ldots K\}$
- The geometry gets a lot more complicated…
  - Cannot simply check the sign of a single $f(\boldsymbol{x})$.



One versus The Other

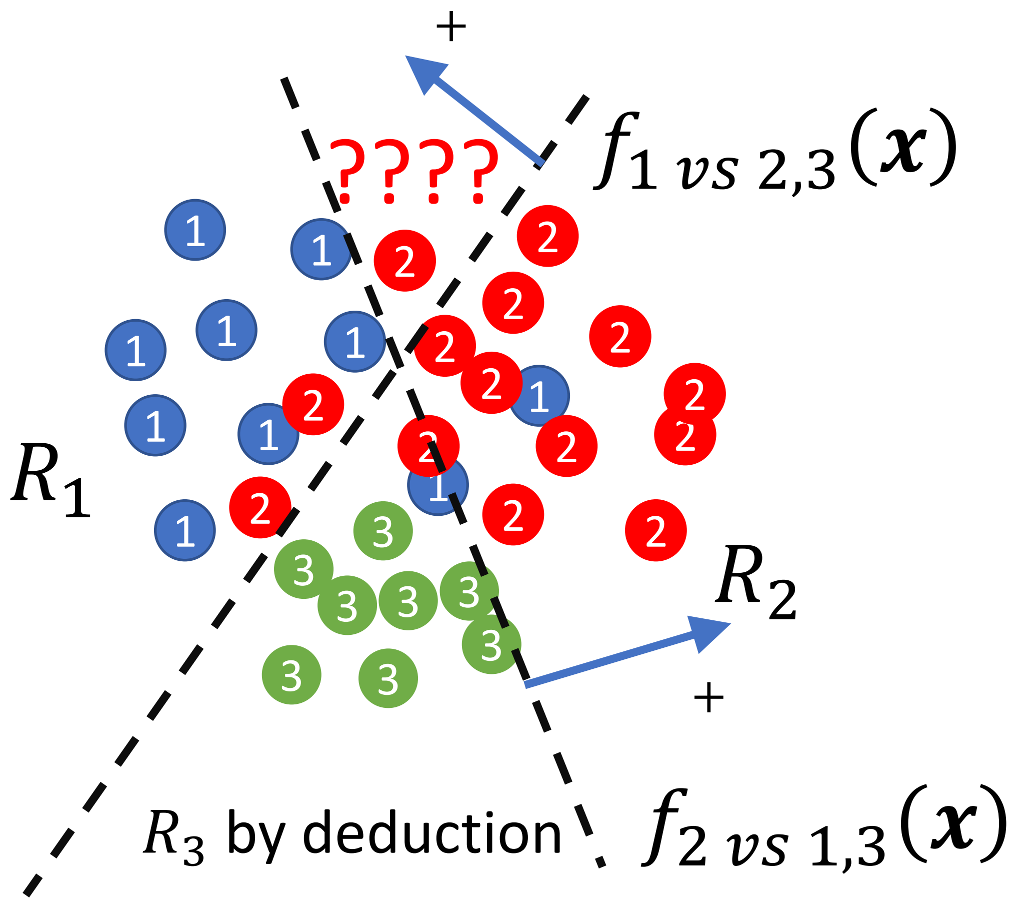
- Construct $K-1$ binary classifiers
- Classify Class k vs. the rest of classes



$f_{1\ vs\ 2,3}(\boldsymbol{x})$
$f_{2\ vs\ 1,3}(\boldsymbol{x})$
$+$
$+$
R_1
R_2
R_3 by deduction
1
2
3

One versus The Other

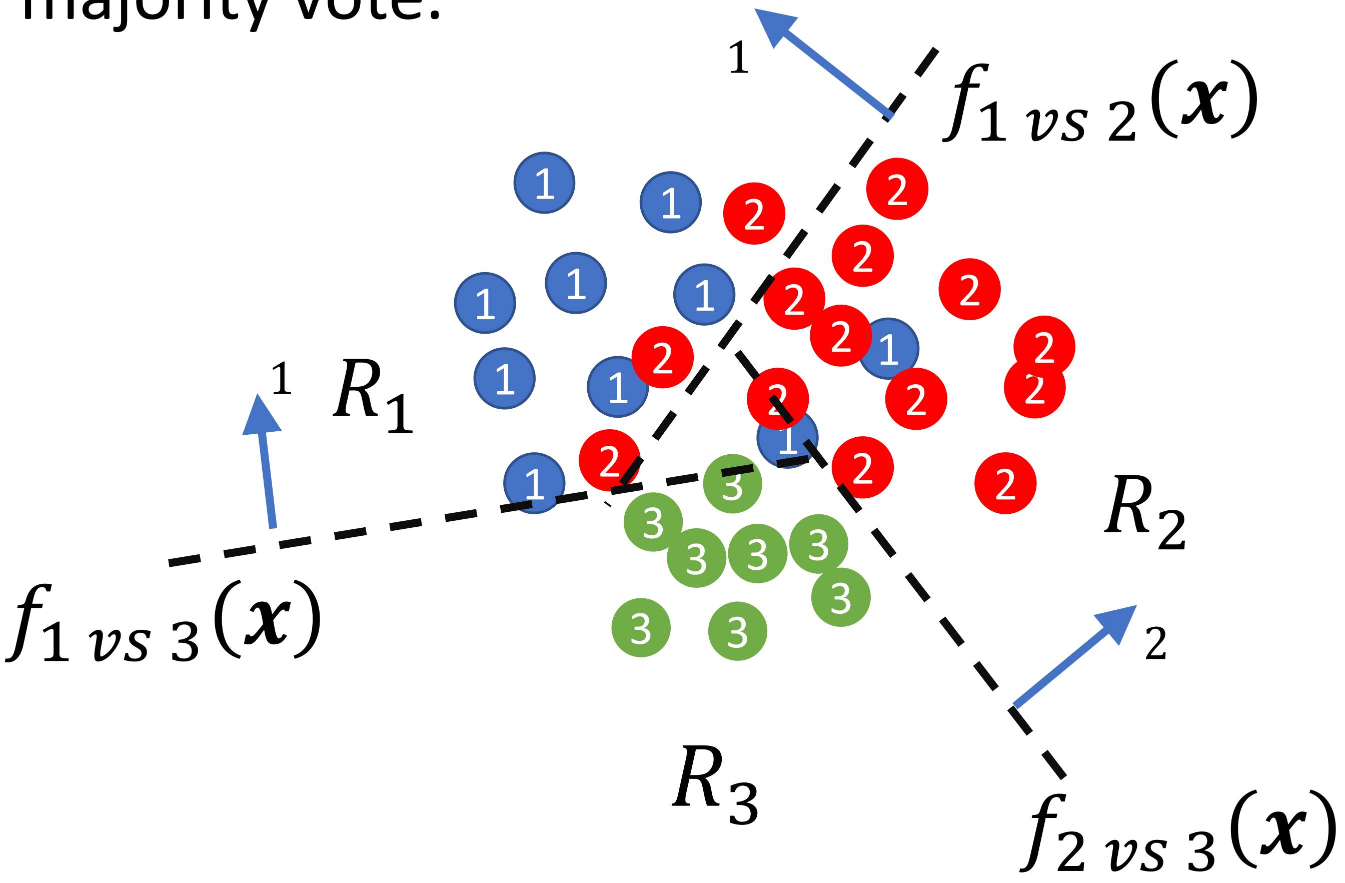
- One versus the other also creates confusion!



+
????
$f_{1\ vs\ 2,3}(\boldsymbol{x})$
R_1
R_2
+
R_3 by deduction
$f_{2\ vs\ 1,3}(\boldsymbol{x})$

One versus One

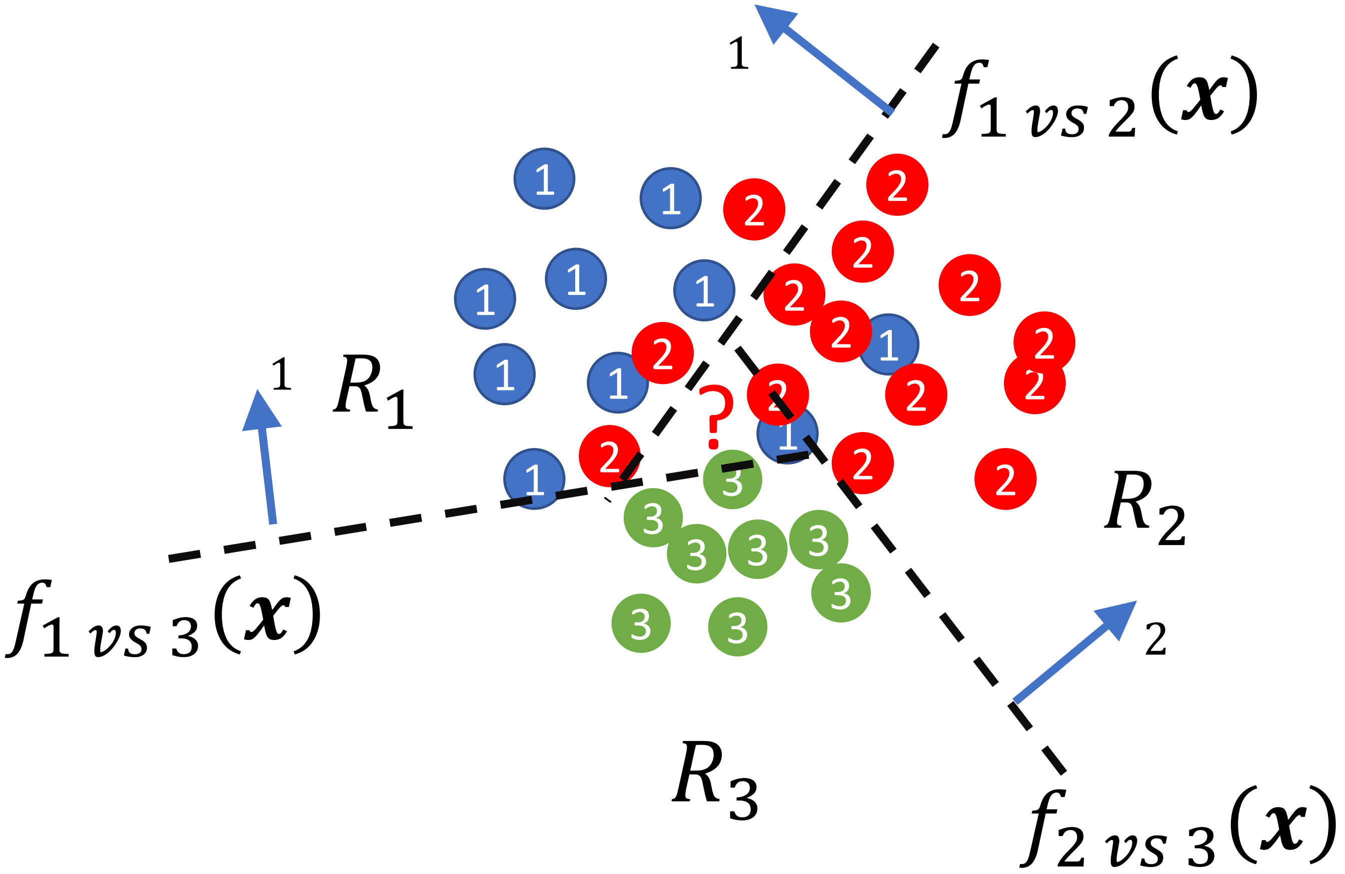
- We can create pairwise binary classifiers and check majority vote.



1
$f_{1\ vs\ 2}(\boldsymbol{x})$
1
R_1
$f_{1\ vs\ 3}(\boldsymbol{x})$
R_2
2
R_3
$f_{2\ vs\ 3}(\boldsymbol{x})$

One versus One

- One versus one creates confusion as well…



1
$f_{1\ vs\ 2}(\boldsymbol{x})$
1
R_1
$?$
R_2
$f_{1\ vs\ 3}(\boldsymbol{x})$
2
R_3
$f_{2\ vs\ 3}(\boldsymbol{x})$

Multi-class Classification

- Or…
- rather than relying on sign of f to make predictions, we can fit a vector valued function $\boldsymbol{f}: R^d \to R^K$:

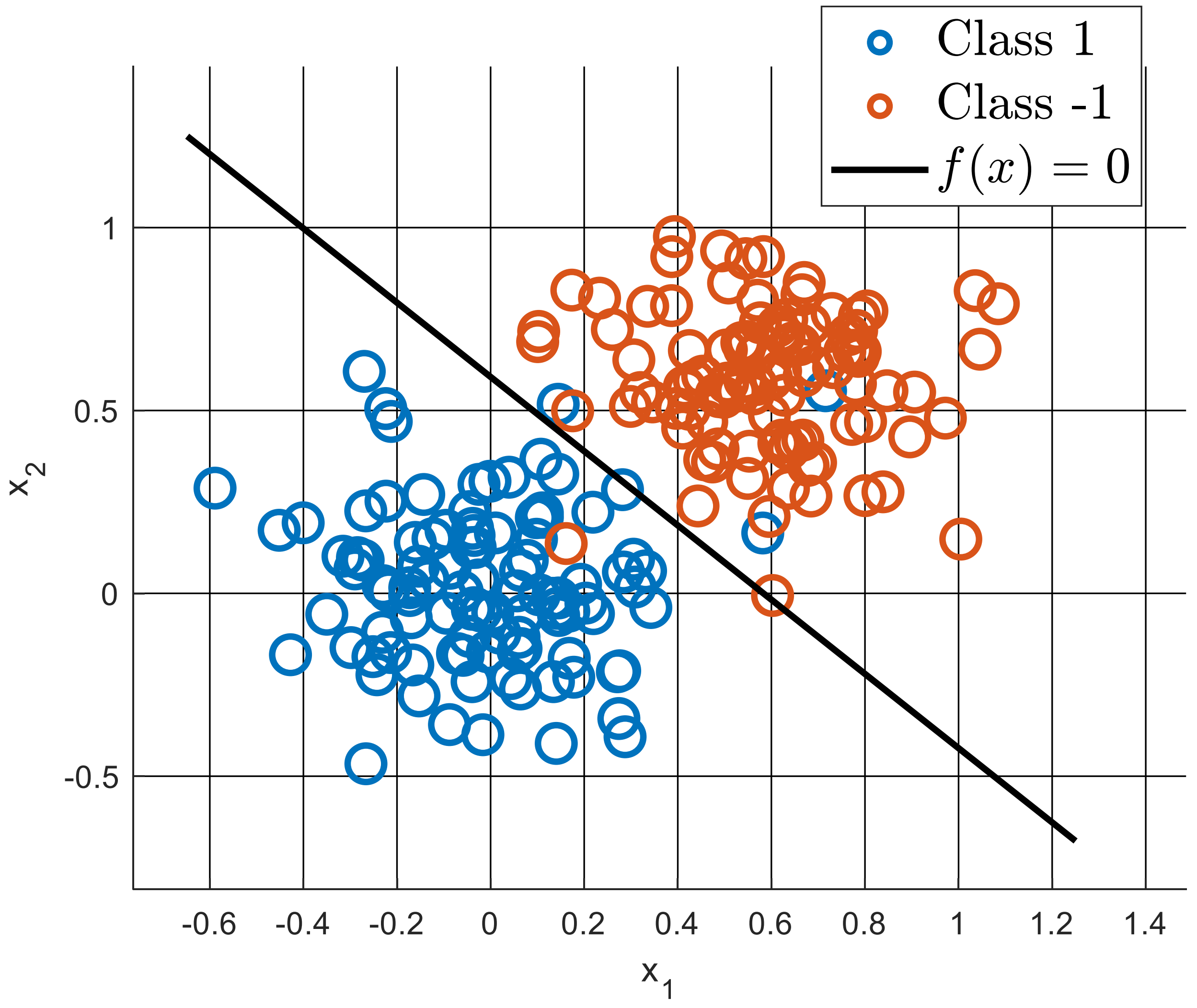
- Given an $\boldsymbol{x}$, prediction is $\hat{k} = \underset{k}{\operatorname{argmax}} f^{(k)}(\boldsymbol{x})$,
  - The classification does not have a simple geometry interpretation anymore.
  - We will see an example soon.

Least Squares Classifier

- For binary classification, perform LS on D.
- $\boldsymbol{w}_{\mathrm{LS}} \coloneqq \operatorname*{argmin}_{\boldsymbol{w}} \sum_{i \in D} [y_i - f(\boldsymbol{x}_i; \boldsymbol{w})]^2$
  - Now y_i takes binary value 1 or -1

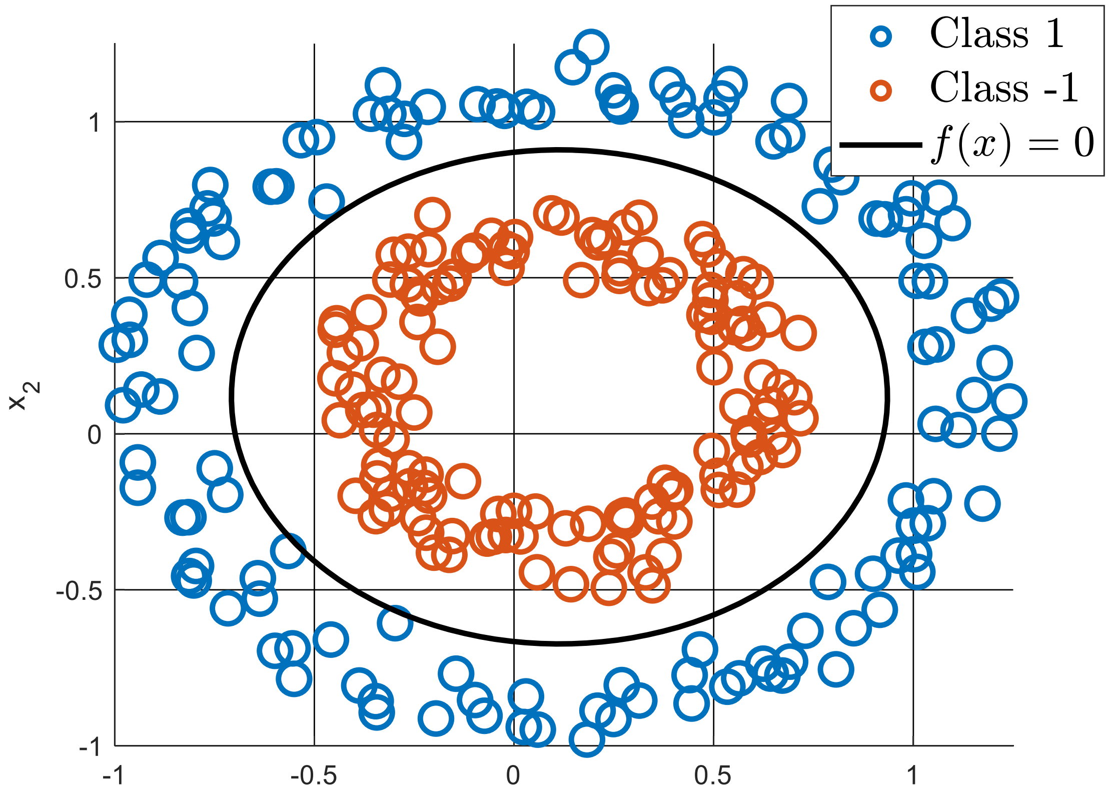
- Prediction function $f(\boldsymbol{x}_i; \boldsymbol{w}_{\mathrm{LS}})$.
- The predicted label $\hat{y} \coloneqq \operatorname{sign}\left(f(\boldsymbol{x}_i; \boldsymbol{w}_{\mathrm{LS}})\right)$

Least Square Classifier



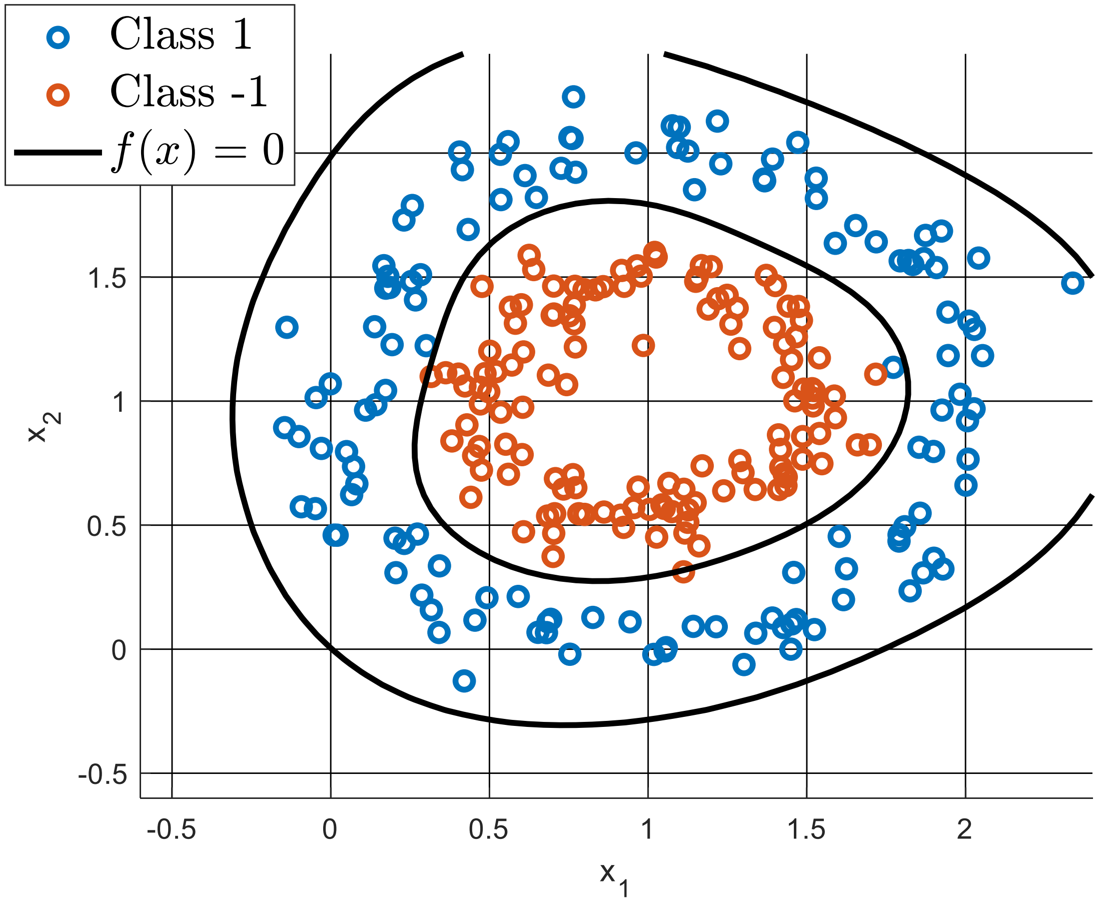
Least Square Classifier

- You can use feature transform $\boldsymbol{\phi}$ for f as well.
- $f(\boldsymbol{x}; \boldsymbol{w}) \coloneqq \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$,
- 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 $\boldsymbol{\phi}$!



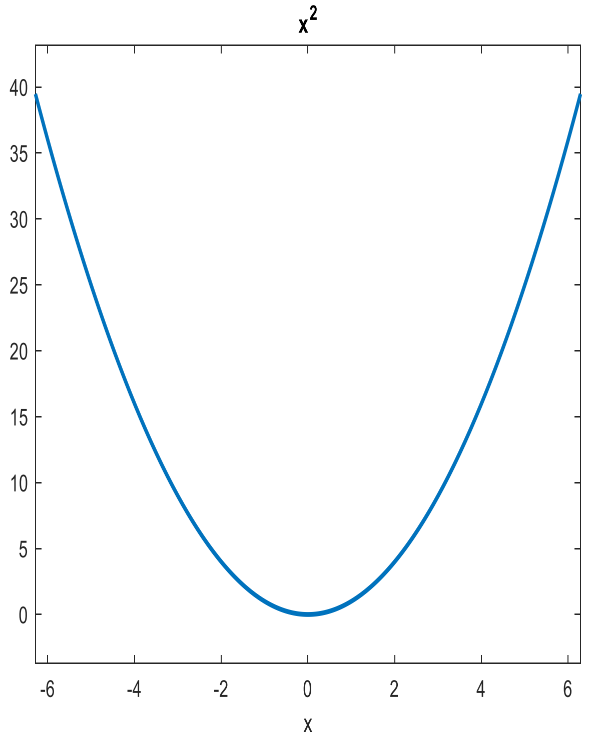
Multi-class LS classification

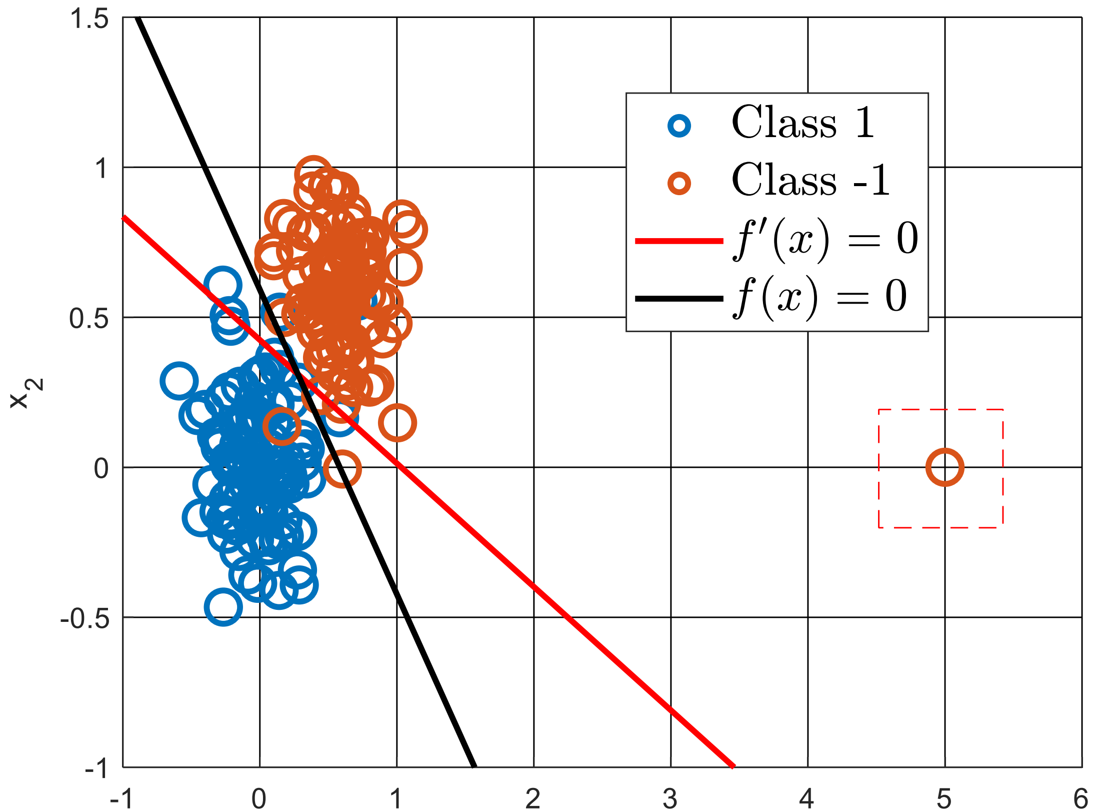
- LS can be adapted to multi-class classification.
- Suppose output $y \in \{1 \dots K\}$
- Replace $y_i = k$ in D with $\boldsymbol{t}_i \in \{0,1\}^K$.
    - $t_i^{(k)} = 1$.
    - $t_i^{(j)} = 0, \forall j \neq k$
- "**One-hot encoding**"

- $\boldsymbol{W}_{\mathrm{LS}} := \mathrm{argmin}_{\boldsymbol{W}} \sum_{i \in D} ||\boldsymbol{t}_i - \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{W})||^2$
- $\boldsymbol{W} \in R^{(d+1) \times K}, \widetilde{\boldsymbol{x}}_i := \left[\boldsymbol{x}_i^\top, 1\right]^\top \in R^d, \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{W}) = \boldsymbol{W}^\top \widetilde{\boldsymbol{x}}$
- Prediction: $\hat{k} = \underset{k}{arg\mathrm{max}} f^{(k)}(\boldsymbol{x}; \boldsymbol{W}) = \underset{k}{arg\mathrm{max}} \left(\boldsymbol{w}_{\mathrm{LS}}^{(k)}\right)^\top \widetilde{\boldsymbol{x}}$
    - where $\boldsymbol{w}^{(k)}$ is the k-th column of $\boldsymbol{W}$.

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.





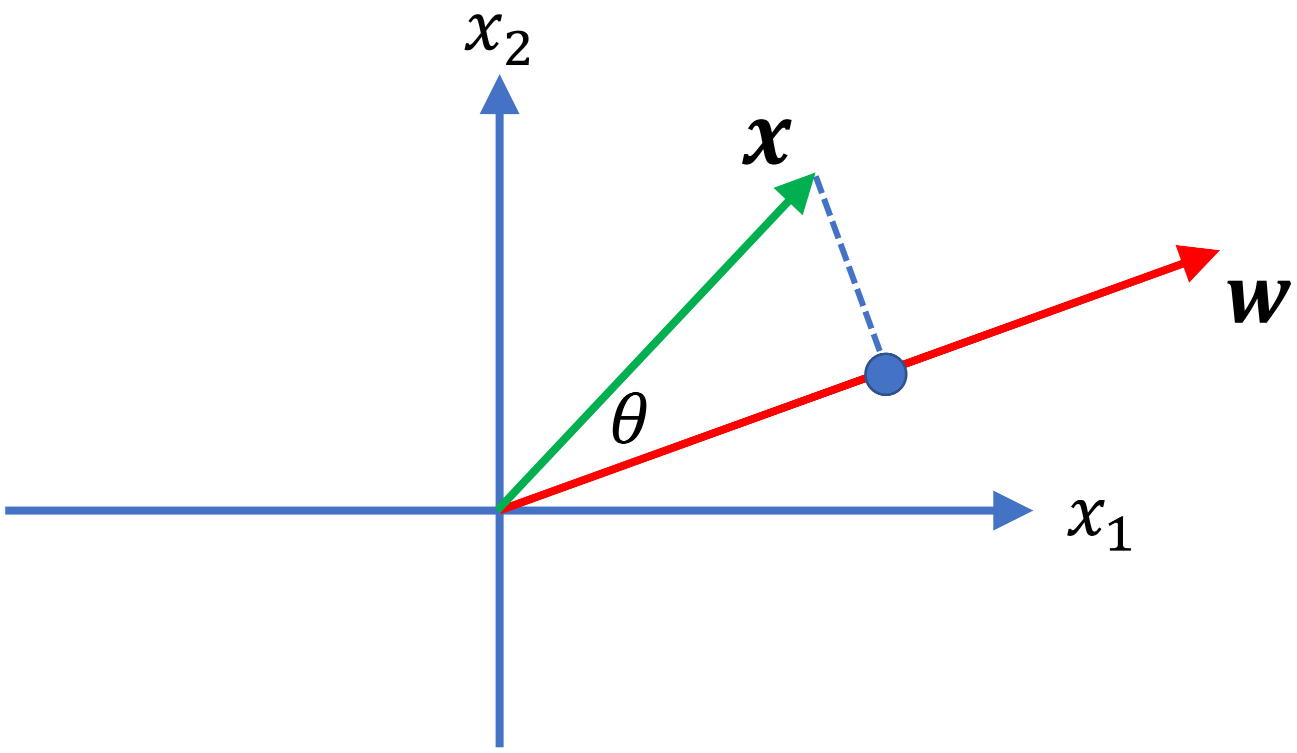
Why not to use LS Classifier?

- 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)

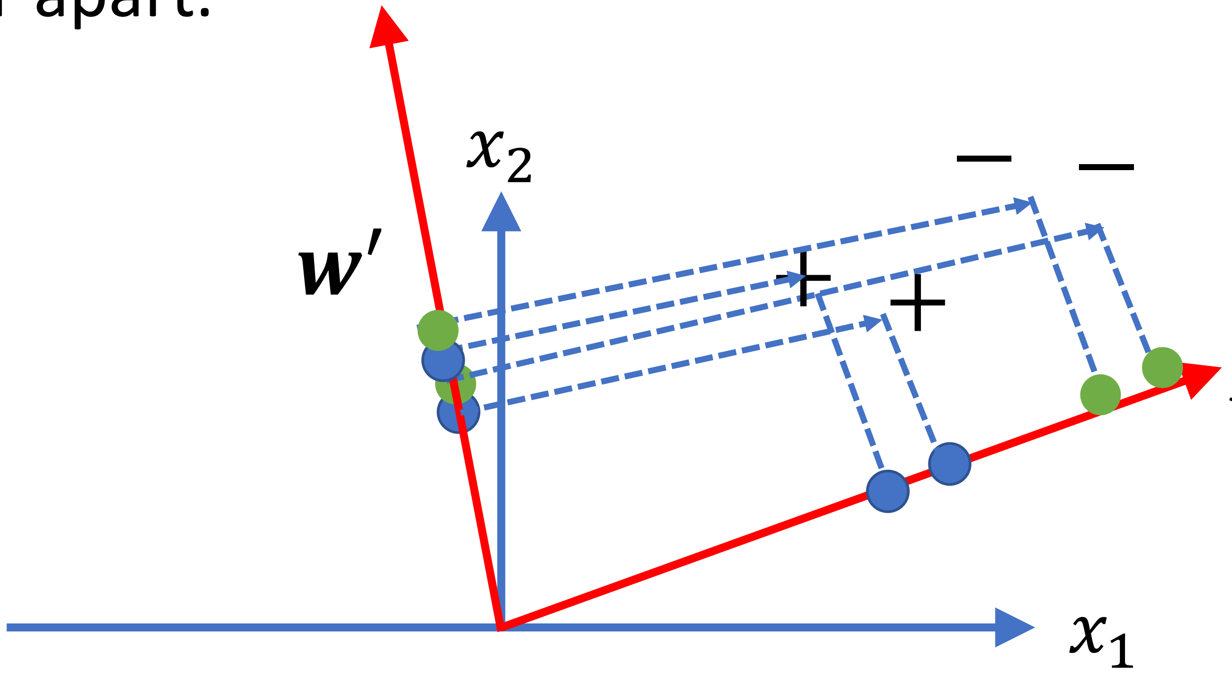
Embedding by Inner Product

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



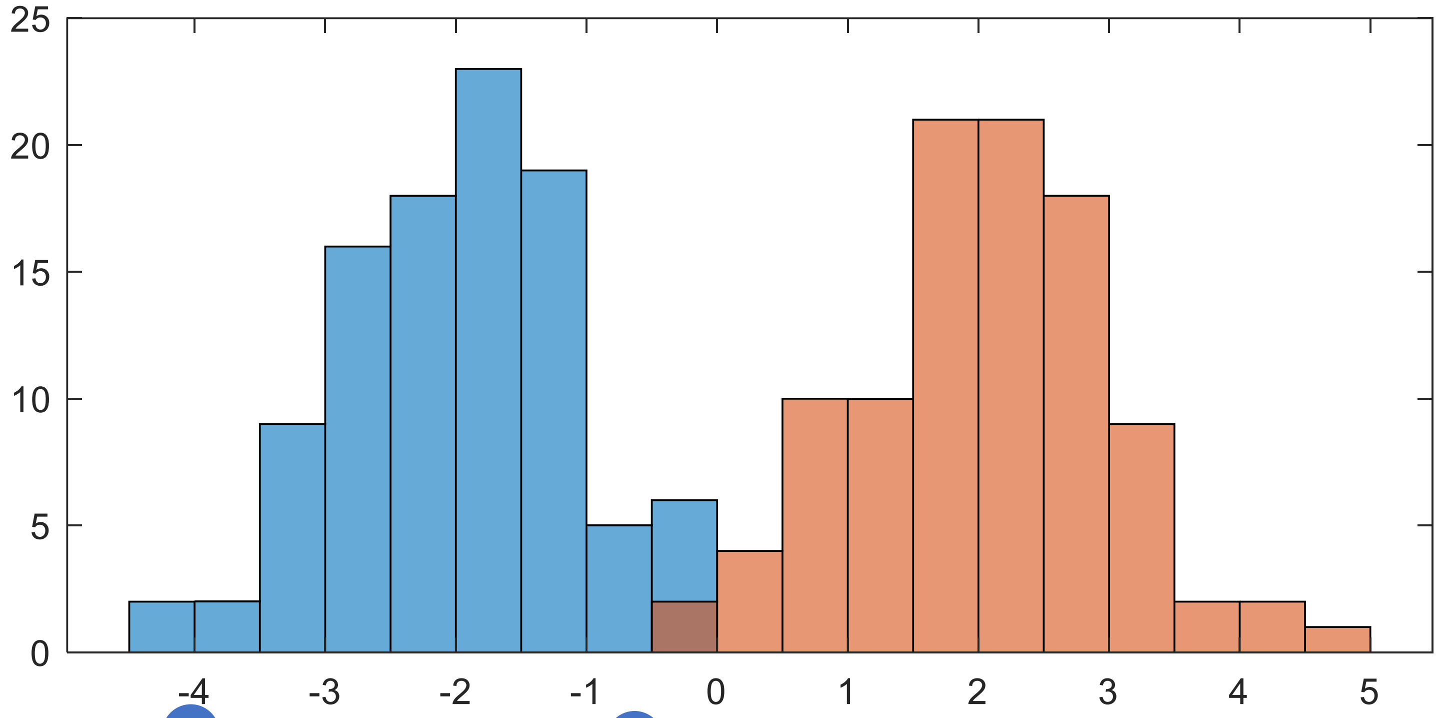
Embedding by Inner Product

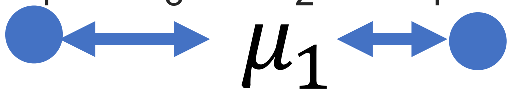
- What would be a good embedding?
- Clearly, we prefer $\boldsymbol{w}$ to $\boldsymbol{w}'$, as the embedding is **more separated** between $+$ and $-$.
- We want points within the class close, but points between two classes far apart.



Within Class and Between Class Scatterness

$$f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$$





$$\mu_1 \longleftrightarrow \mu \longleftrightarrow \mu_2$$

Within-class Scatterness

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

Between-class Scatterness

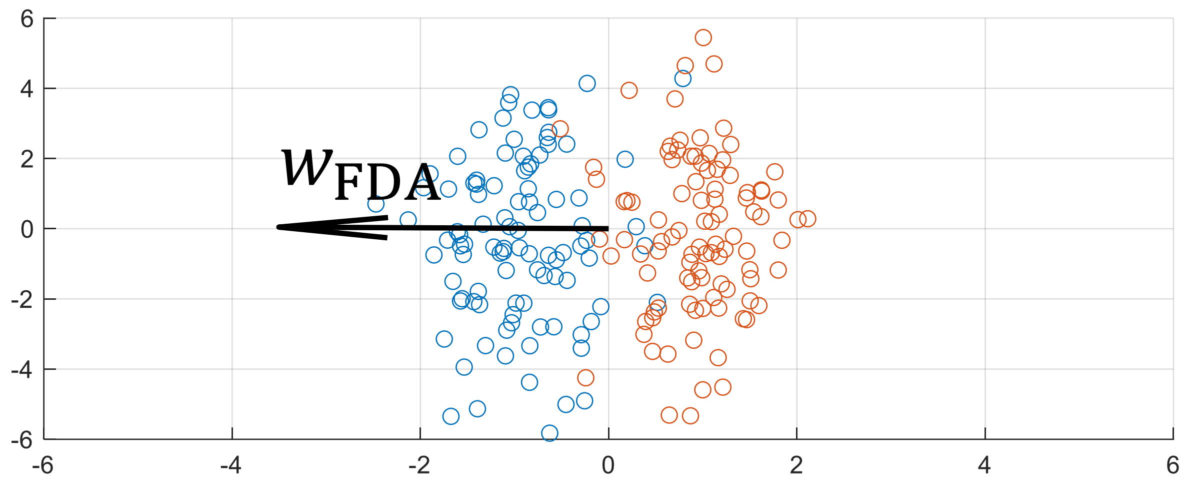
- Embedded dataset center:
  - $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{w}^{\top}\boldsymbol{x}_i$
- Between-class scatterness
  - $s_{\mathrm{b},k} = n_k(\hat{\mu}_k - \hat{\mu})^2$
    - n_k is needed to make $s_{\mathrm{b},k}$ at the same scale with $s_{\mathrm{w},k}$.

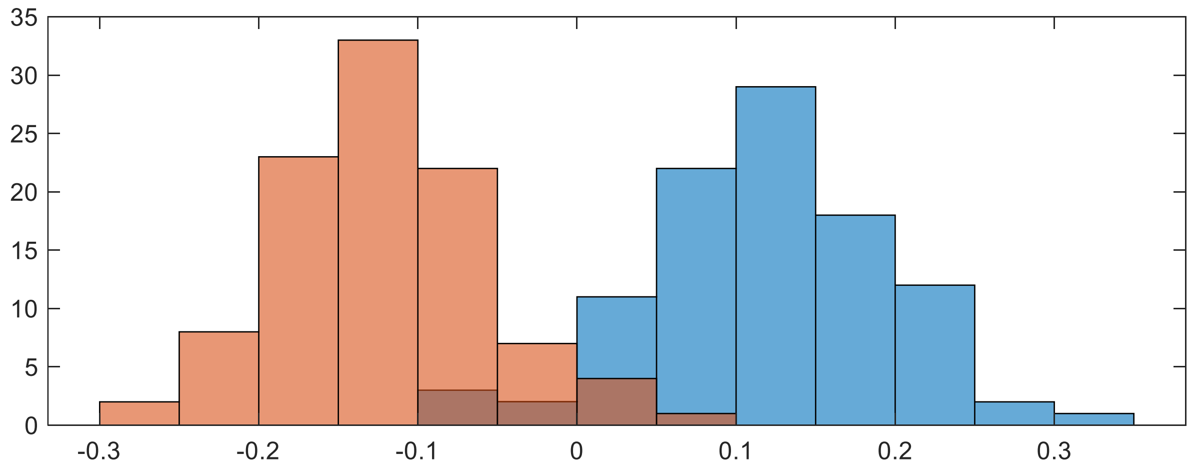
Fisher Discriminant Analysis

- **Maximizing** between class scatterness $\forall_k$.
- **Minimize** within class scatterness $\forall_k$.
- $\max_{\boldsymbol{w}} \boxed{\sum_k s_{\mathrm{b},k}} / \boxed{\sum_k s_{\mathrm{w},k}}$

- If $K = 2$, this has a simple solution that
- $\boldsymbol{w} := \boldsymbol{S}_{\boldsymbol{w}}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-), \boldsymbol{S}_{\boldsymbol{w}} := \sum_{k=1}^{K} \boldsymbol{S}_k$
- $\boldsymbol{S}_k$ **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.

- $f(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{FDA}}) = \langle \boldsymbol{w}_{\mathrm{FDA}}, \boldsymbol{x} \rangle$ obtained by FDA cannot be directly used for making a prediction:

- In general, $f(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{FDA}}) > 0$ does not mean $\boldsymbol{x}$ 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**:

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

- We need: $p(y|\boldsymbol{x}), y \in \{1, \dots, K\}$

- Discriminative: Infer $p(y|\boldsymbol{x})$ directly.
- **Generative**: Infer $p(y|\boldsymbol{x}) \propto p(\boldsymbol{x}|y)p(y)$, infer $p(\boldsymbol{x}|y)$!

Continuous Input Variable

- To infer $p(\boldsymbol{x}|y)$, we need a model.
- If $\boldsymbol{x}$ is continuous, MVN is a natural choice for $p(\boldsymbol{x}|y)$.

- **Model** $p(\boldsymbol{x}|y = k; \boldsymbol{w}) \coloneqq N_{\boldsymbol{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Assuming IID, and all classes have shared covariance $\boldsymbol{\Sigma}$
- **Write down the likelihood** over D:
- $p(D|\boldsymbol{w}) = \prod_{i \in D} p(\boldsymbol{x}_i, y_i | \boldsymbol{w}) = \prod_{i \in D} p(\boldsymbol{x}_i | y_i; \boldsymbol{w}) p(y_i)$

$$= \prod_{i \in D} N_{\boldsymbol{x}_i}\left(\boldsymbol{\mu}_{y_i}; \boldsymbol{\Sigma}\right) p(y_i)$$

Continuous Input Variable

- $\hat{\boldsymbol{\mu}}_{1\ldots K}, \hat{\boldsymbol{\Sigma}} \quad := \arg\max_{\boldsymbol{\mu}_{1\ldots k}, \boldsymbol{\Sigma}} \; \sum_{i \in D} \log[N_{\boldsymbol{x}_i}(\boldsymbol{\mu}_{y_i}; \boldsymbol{\Sigma}) p(y_i)]$

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

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

Linear Decision Boundary

- **Prediction**: $\hat{y} \coloneqq \operatorname{argmax}_y p(y|\boldsymbol{x}; \hat{\boldsymbol{w}}) \propto p(\boldsymbol{x}|y; \hat{\boldsymbol{w}})p(y)$

- **Prove**: when using shared covariance matrix MVN model, the decision boundary is piecewise-linear.
- The decision boundary is

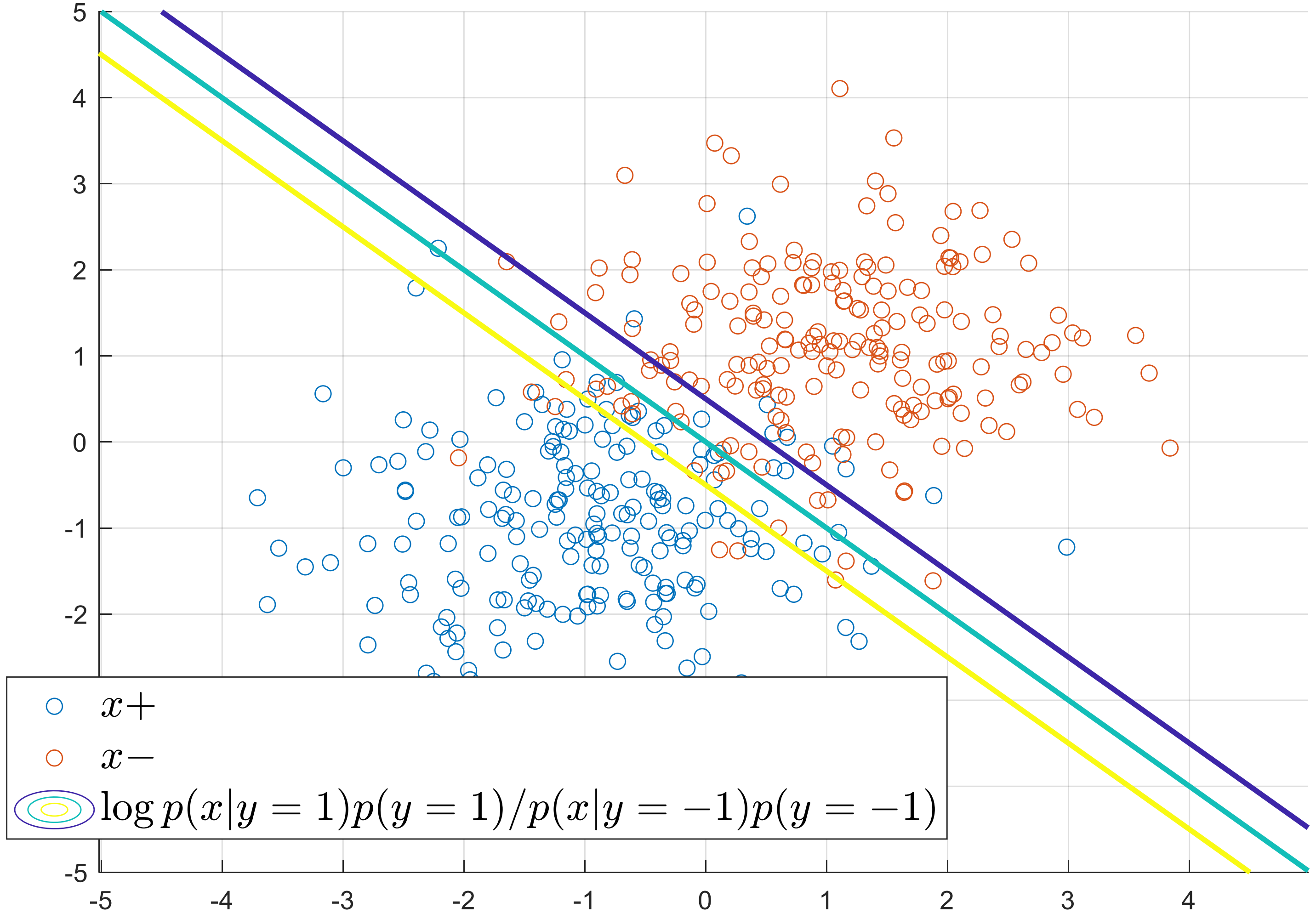
$$\{\boldsymbol{x}|p(y=k|\boldsymbol{x};\hat{\boldsymbol{w}}) = p(y=k'|\boldsymbol{x};\hat{\boldsymbol{w}})\}$$
$$\forall k \neq k'$$

Which is the same as the set

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

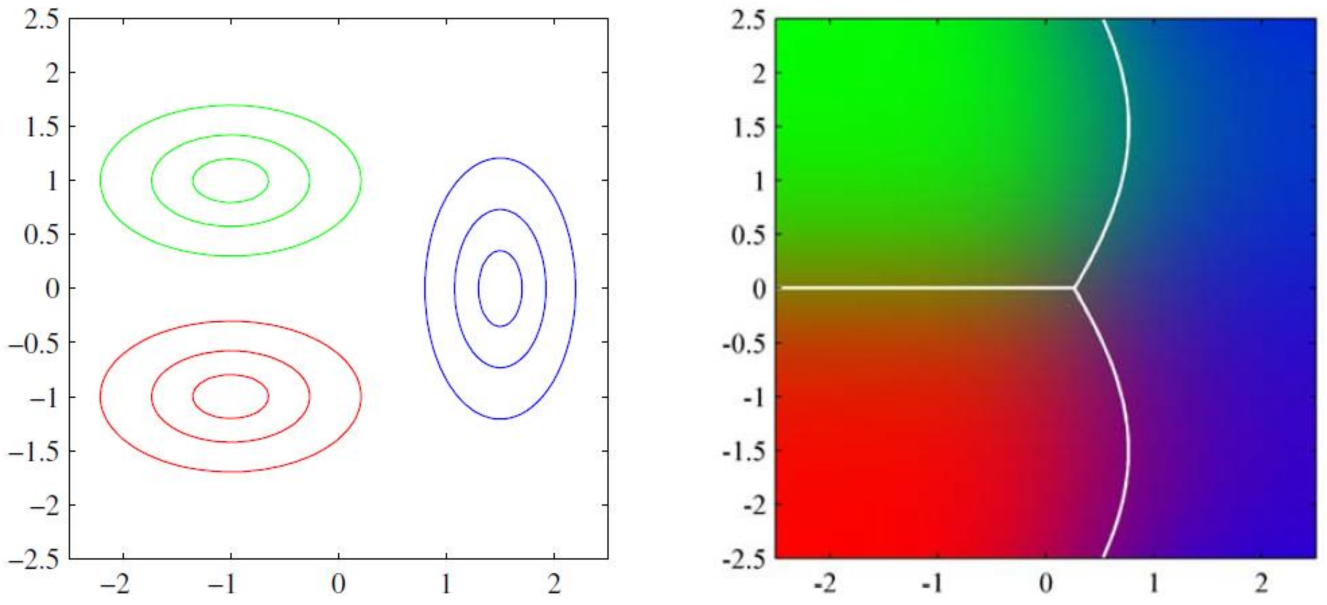
Hint: take log on both sides of the equality.

Linear Decision Boundary



Continuous Input Variable

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



PRML, Figure 4.11

Discrete Input Variable $\boldsymbol{x}$

- In many classification tasks, we are dealing with discrete variables as $\boldsymbol{x}$. For example, in a spam filter,
- $\boldsymbol{x} \coloneqq \left[x^{(1)}, \dots, x^{(d)}\right]^{\top}$ are frequencies of words in a document. This is called "bag of words" representation.
- $y \in \{\mathrm{spam}, \mathrm{ham}\}$.

- For example, the document "to be or not to be"
- $\boldsymbol{x} \coloneqq [\mathrm{to} = 2, \mathrm{be} = 2, \mathrm{or} = 1, \mathrm{not} = 1, \mathrm{question} = 0]^{\top}$.
- $x^{(i)} \in N_0$

Naïve Bayes

- Assume $x^{(1)} \dots x^{(d)}$ follows multinomial distribution
- $p(\boldsymbol{x} = \boldsymbol{x}_0 | y) \propto \prod_{i=1\dots d} \beta(i|y)^{x_0^{(i)}}$ 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:

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

- $\beta(\mathrm{to}|y = \mathrm{spam})$ 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: $\hat{y} \coloneqq \operatorname{argmax}_y p(\boldsymbol{x} = \boldsymbol{x}_0 | y) p(y)$

- $p(y = k)$: $\frac{n_k}{n}$
- $p(\boldsymbol{x} = \boldsymbol{x}_0 | y) \propto \prod_{i=1\ldots d} \beta(i|y)^{x_0^{(i)}}$
  - $\beta(i|y)$ has been obtained by previous counting.

- $p(\boldsymbol{x} = \text{"}\boldsymbol{to\ be\ or\ not\ to\ be}\text{"} | y = \text{spam}) \propto$
  $\beta(to|\text{spam})^2 \beta(be|\text{spam})^2 \beta(or|\text{spam}) \beta(not|\text{spam})$

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.