Linear classifiers

CE-717: Machine Learning

Sharif University of Technology

M. Soleymani

Fall 2016

Topics

- Discriminant functions
- Linear classifiers
  - Perceptron
  - Fisher
- Multi-class classification

SVM will be covered in the later lectures

Classification problem

- Given: Training set
  - labeled set of N input-output pairs $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - $y \in \{1, \dots, K\}$

- Goal: Given an input $\boldsymbol{x}$, assign it to one of K classes

- Examples:
  - Spam filter
  - Handwritten digit recognition
  - …

Discriminant functions

- **Discriminant function** can directly assign each vector $\boldsymbol{x}$ to a specific class k

- A popular way of representing a classifier
  - Many classification methods are based on discriminant functions

- Assumption: the classes are taken to be disjoint
  - The input space is thereby divided into **decision regions**
    - boundaries are called **decision boundaries** or decision surfaces.

Discriminant Functions

- **Discriminant functions**: A discriminant function $f_i(\boldsymbol{x})$ for each class $\mathcal{C}_i$ ($i = 1, \ldots, K$):
  - $\boldsymbol{x}$ is assigned to class $\mathcal{C}_i$ if:

$$f_i(\boldsymbol{x}) > f_j(\boldsymbol{x}) \quad \forall j \neq i$$

- Thus, we can easily divide the feature space into K decision regions

$$\forall \boldsymbol{x}, f_i(\boldsymbol{x}) > f_j(\boldsymbol{x}) \quad \forall j \neq i \Rightarrow \boldsymbol{x} \in \mathcal{R}_i$$

$\mathcal{R}_i$: Region of the i-th class

- **Decision surfaces (or boundaries)** can also be found using discriminant functions
  - Boundary of the $\mathcal{R}_i$ and $\mathcal{R}_j$ separating samples of these two categories:

$$\forall \boldsymbol{x}, f_i(\boldsymbol{x}) = f_j(\boldsymbol{x})$$

Discriminant Functions: Two-Category

- Decision surface: $f(\boldsymbol{x}) = 0$

- For two-category problem, we can only find a function $f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}$
  - $f_1(\boldsymbol{x}) = f(\boldsymbol{x})$
  - $f_2(\boldsymbol{x}) = -f(\boldsymbol{x})$

- First, we explain two-category classification problem and then discuss the multi-category problems.
  - Binary classification: a target variable $y \in \{0,1\}$ or $y \in \{-1,1\}$

Linear classifiers

- Decision boundaries are linear in $\boldsymbol{x}$, or linear in some given set of functions of $\boldsymbol{x}$
- Linearly separable data: data points that can be exactly classified by a linear decision surface.
- Why linear classifier?
  - Even when they are not optimal, we can use their simplicity
    - are relatively easy to compute
    - In the absence of information suggesting otherwise, linear classifiers are an attractive candidates for initial, trial classifiers.

Two Category

- $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x} + w_0 = w_0 + w_1x_1 + \ldots w_dx_d$
  - $\boldsymbol{x} = [x_1\ x_2\ \ \ldots x_d]$
  - $\boldsymbol{w} = [w_1\ w_2\ \ \ldots w_d]$
  - w_0: bias

if $\boldsymbol{w}^T\boldsymbol{x} + w_0 \geq 0$ then $\mathcal{C}_1$

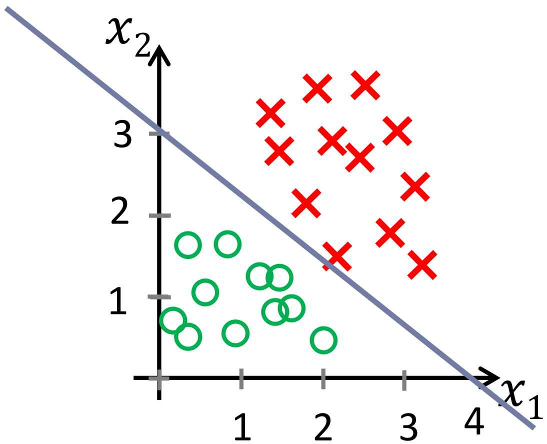
else $\mathcal{C}_2$

Decision surface (boundary): $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$

$\boldsymbol{w}$ is orthogonal to every vector lying within the decision surface

Example

$$3 - \frac{3}{4}x_1 - x_2 = 0$$



if $\boldsymbol{w}^T\boldsymbol{x} + w_0 \geq 0$ then $\mathcal{C}_1$
else $\mathcal{C}_2$

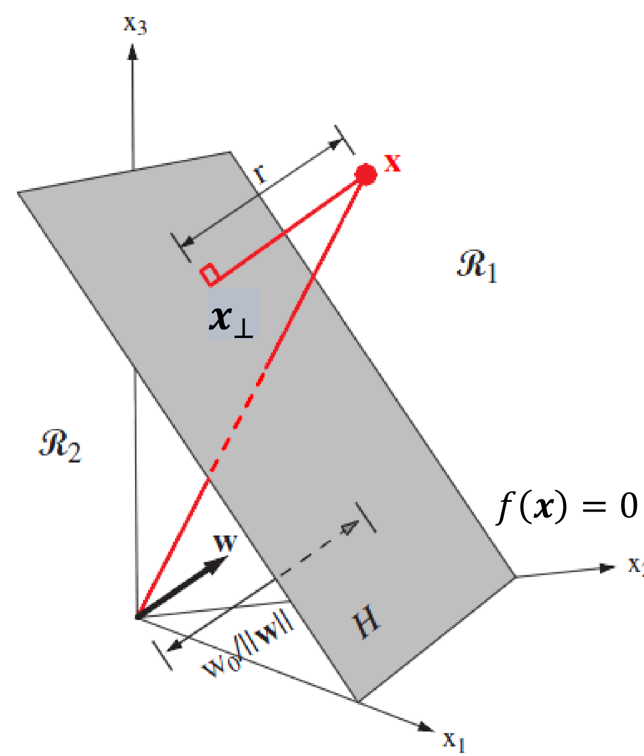
Linear classifier: Two Category

- Decision boundary is a $(d-1)$-dimensional hyperplane H in the d-dimensional feature space
  - The orientation of H is determined by the normal vector $[w_1, \dots, w_d]$
  - w_0 determine the location of the surface.
    - The normal distance from the origin to the decision surface is $\frac{w_0}{\|\boldsymbol{w}\|}$

$$\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = r\|\boldsymbol{w}\| \Rightarrow r = \frac{\boldsymbol{w}^T\boldsymbol{x} + w_0}{\|\boldsymbol{w}\|}$$

gives a signed measure of the perpendicular distance r of the point $\boldsymbol{x}$ from the decision surface

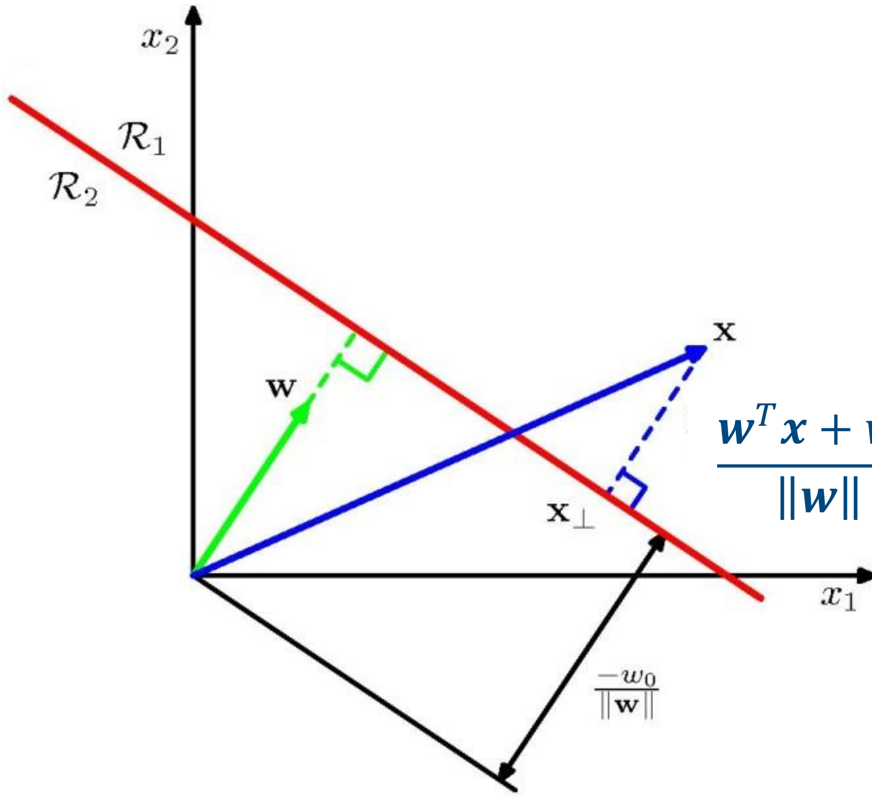


Linear boundary: geometry

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 > 0$$

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$$

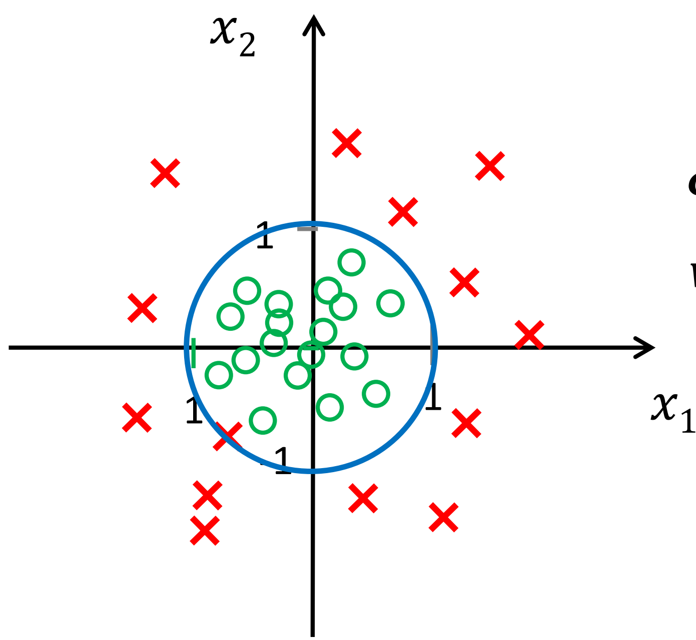
$$\boldsymbol{w}^T\boldsymbol{x} + w_0 < 0$$



$$\frac{\boldsymbol{w}^T\boldsymbol{x} + w_0}{\|\boldsymbol{w}\|}$$

Non-linear decision boundary

- Choose non-linear features
- Classifier still linear in parameters $\boldsymbol{w}$



$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]$

$$-1 + x_1^2 + x_2^2 = 0$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_1 \boldsymbol{x}_2]$$

$$\boldsymbol{w} = [w_0, w_1, \dots, w_m] = [-1, 0, 0, 1, 1, 0]$$

if $\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \geq 0$ then $y = 1$
else $y = -1$

Cost Function for linear classification

- Finding linear classifiers can be formulated as an optimization problem:
  - Select how to measure the prediction loss
    - Based on the training set $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, a cost function $J(\boldsymbol{w})$ is defined
  - Solve the resulting optimization problem to find parameters:
    - Find optimal $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x}; \widehat{\boldsymbol{w}})$ where $\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} J(\boldsymbol{w})$

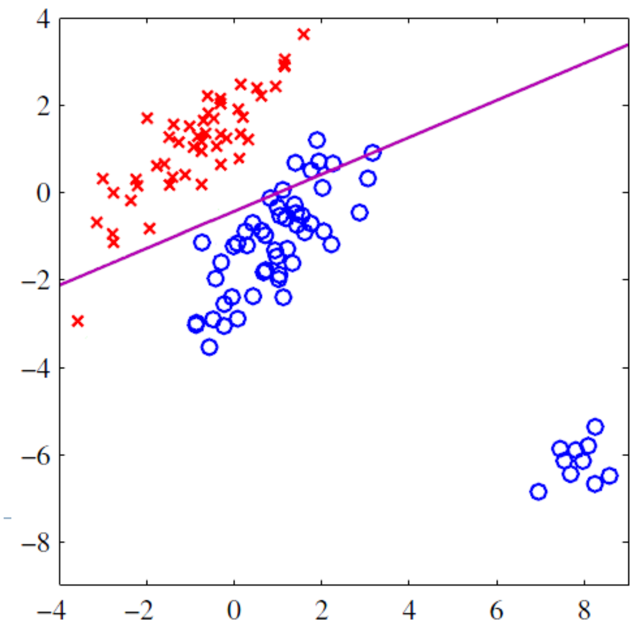
- Criterion or cost functions for classification:
  - We will investigate several cost functions for the classification problem

SSE cost function for classification $K = 2$

SSE cost function is not suitable for classification:

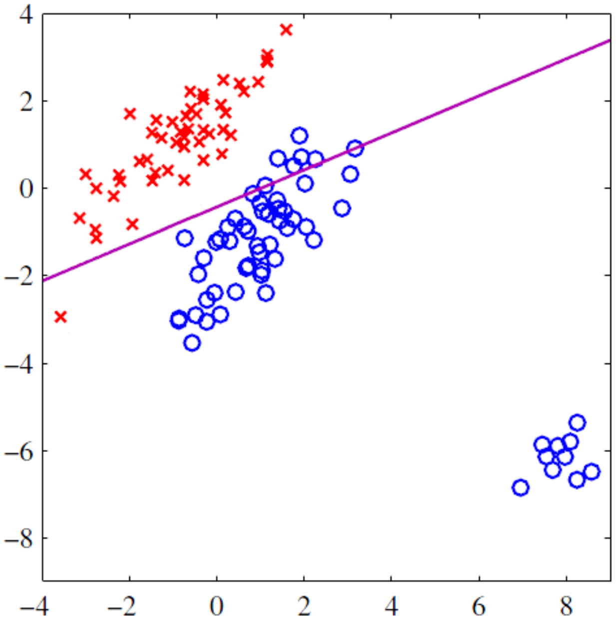
- Least square loss penalizes 'too correct' predictions (that they lie a long way on the correct side of the decision)
- Least square loss also lack robustness to noise

$$J(\boldsymbol{w}) = \sum_{i=1}^{N} \left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

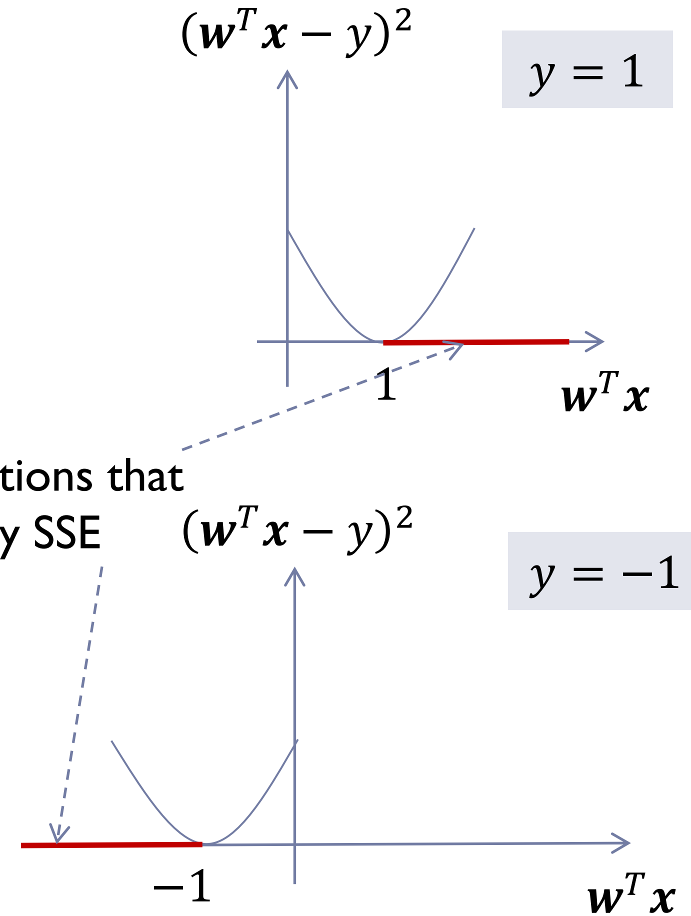


SSE cost function for classification

$K = 2$



[Bishop]



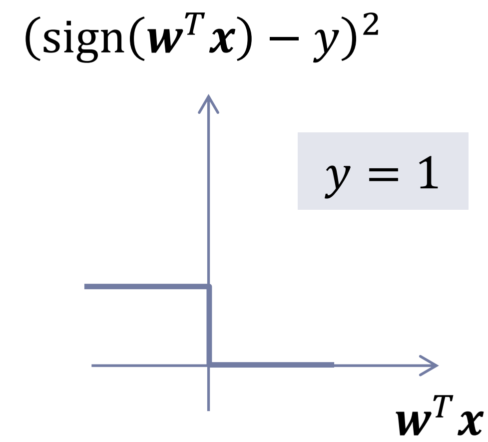
Correct predictions that are penalized by SSE

SSE cost function for classification $K = 2$

- Is it more suitable if we set $f(\boldsymbol{x}; \boldsymbol{w}) = g(\boldsymbol{w}^T \boldsymbol{x})$?

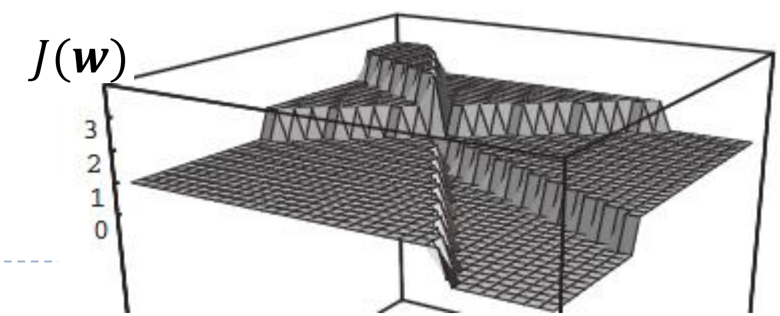
$$J(\boldsymbol{w}) = \sum_{i=1}^{N} \left(\text{sign}\left(\boldsymbol{w}^T \boldsymbol{x}^{(i)}\right) - y^{(i)} \right)^2$$

$$\text{sign}(z) = \begin{cases} -1, & z < 0 \\ 1, & z \geq 0 \end{cases}$$



- $J(\boldsymbol{w})$ is a piecewise constant function shows the number of misclassifications

  Training error incurred in classifying training samples



Perceptron algorithm

- Linear classifier
- Two-class: $y \in \{-1,1\}$
  - $y = -1$ for C_2, $\quad y = 1$ for C_1

- Goal: $\forall i, \boldsymbol{x}^{(i)} \in C_1 \ \Rightarrow \boldsymbol{w}^T\boldsymbol{x}^{(i)} > 0$

  $\forall i, \boldsymbol{x}^{(i)} \in C_2 \ \Rightarrow \boldsymbol{w}^T\boldsymbol{x}^{(i)} < 0$

- $f(\boldsymbol{x}; \boldsymbol{w}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x})$

Perceptron criterion

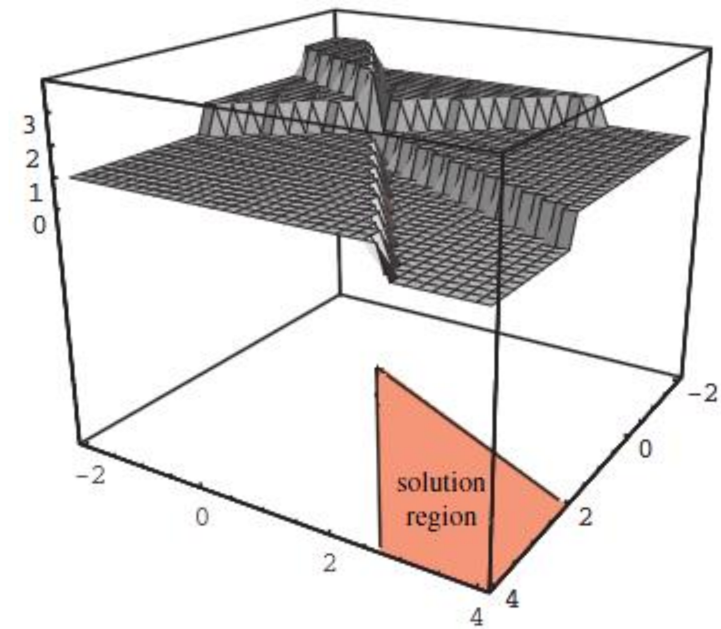
$$J_P(\boldsymbol{w}) = -\sum_{i\in\mathcal{M}} \boldsymbol{w}^T\boldsymbol{x}^{(i)}y^{(i)}$$

$\mathcal{M}$: subset of training data that are misclassified

Many solutions? Which solution among them?

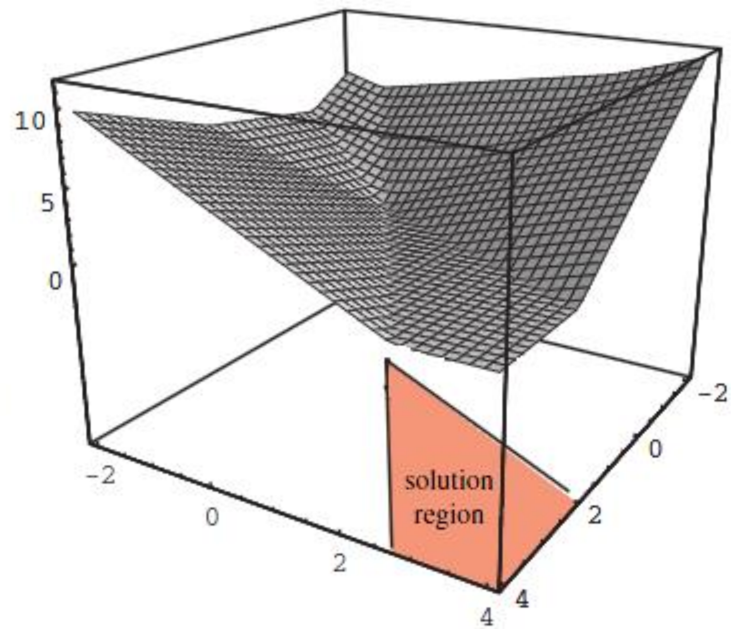
Cost function

$J(\boldsymbol{w})$



# of misclassifications
as a cost function

$J_P(\boldsymbol{w})$



Perceptron's
cost function

There may be many solutions in these cost functions

[Duda, Hart, and Stork, 2002]

Batch Perceptron

"Gradient Descent" to solve the optimization problem:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \nabla_{\boldsymbol{w}} J_P(\boldsymbol{w}^{t})$$

$$\nabla_{\boldsymbol{w}} J_P(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{x}^{(i)} y^{(i)}$$

Batch Perceptron converges in finite number of steps for linearly separable data:

Initialize $\boldsymbol{w}$
Repeat
$\quad \boldsymbol{w} = \boldsymbol{w} + \eta \sum_{i \in \mathcal{M}} \boldsymbol{x}^{(i)} y^{(i)}$
Until $\eta \sum_{i \in \mathcal{M}} \boldsymbol{x}^{(i)} y^{(i)} < \theta$

Stochastic gradient descent for Perceptron

- Single-sample perceptron:
  - If $\boldsymbol{x}^{(i)}$ is misclassified:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} + \eta \boldsymbol{x}^{(i)} y^{(i)}$$

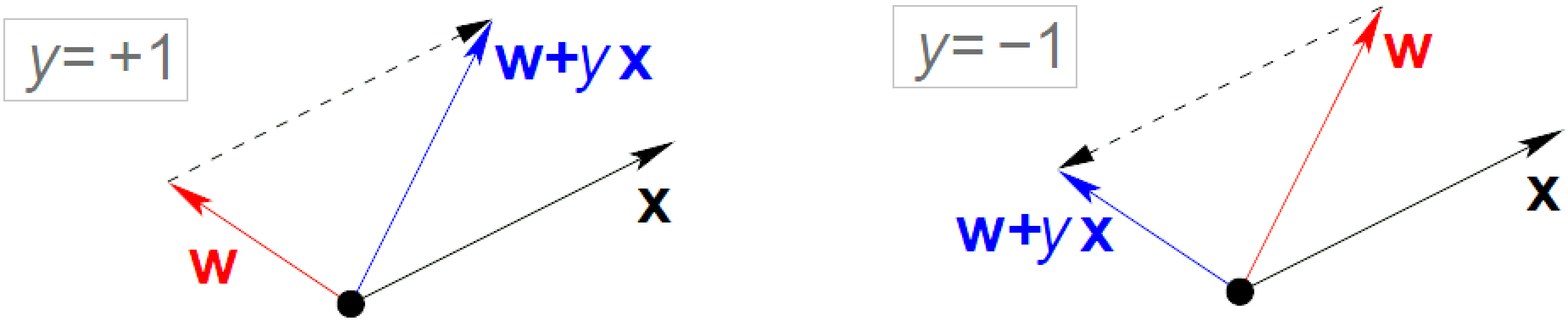
- Perceptron convergence theorem: for linearly separable data
  - If training data are linearly separable, the single-sample perceptron is also guaranteed to find a solution in a finite number of steps

Fixed-Increment single sample Perceptron

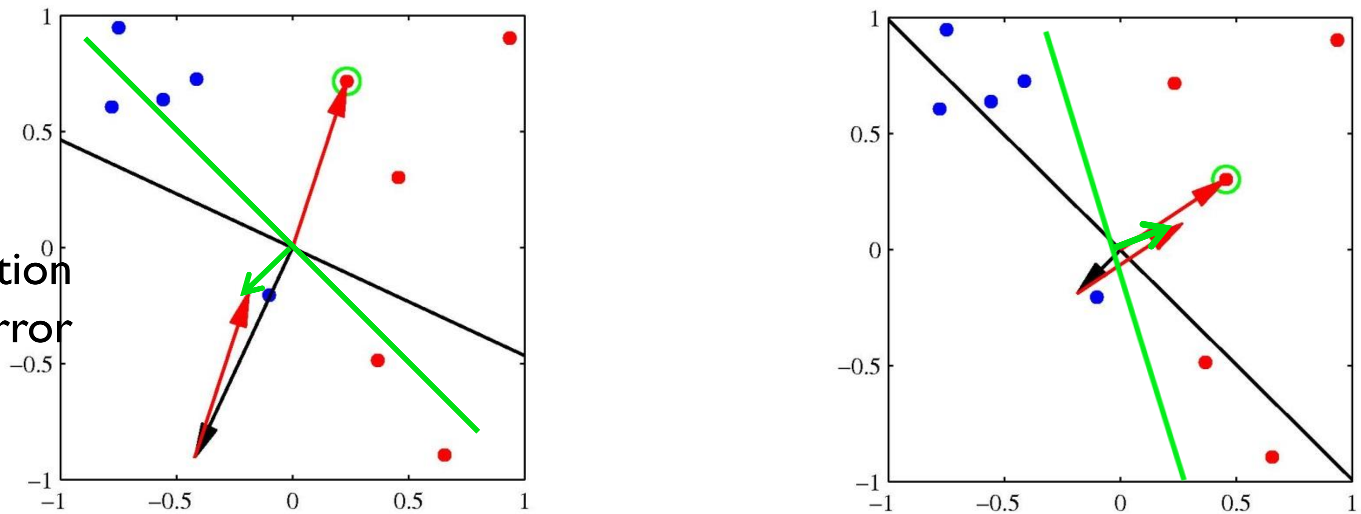
η can be set to 1 and proof still works

Initialize $\boldsymbol{w}, t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  $i \leftarrow t \bmod N$
  if $\boldsymbol{x}^{(i)}$ is misclassified then
    $\boldsymbol{w} = \boldsymbol{w} + \boldsymbol{x}^{(i)} y^{(i)}$
Until all patterns properly classified

Example



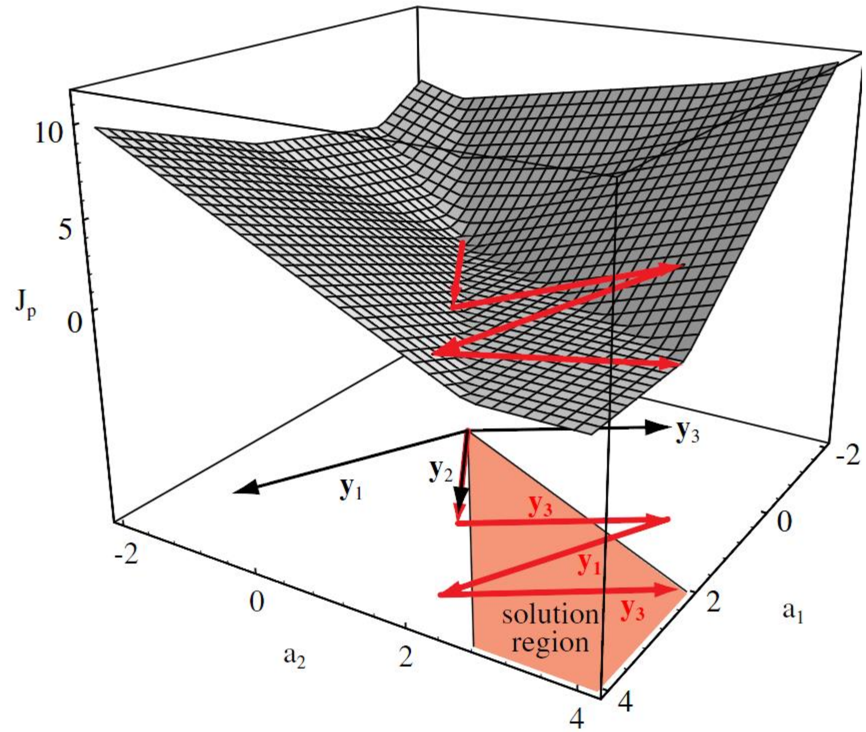
Perceptron: Example



Change $\boldsymbol{w}$ in a direction that corrects the error

[Bishop]

Convergence of Perceptron



[Duda, Hart & Stork, 2002]

- For data sets that are not linearly separable, the single-sample perceptron learning algorithm will never converge

Pocket algorithm

- For the data that are not linearly separable due to noise:
  - Keeps in its pocket the best $\boldsymbol{w}$ encountered up to now.

```
Initialize w
for t = 1, ..., T
    i ← t mod N
    if x^(i) is misclassified then
        w^new = w + x^(i) y^(i)
        if E_train(w^new) < E_train(w) then
            w = w^new
end
```

Initialize $\boldsymbol{w}$
for $t = 1, \dots, T$
$\quad i \leftarrow t \bmod N$
$\quad$ if $\boldsymbol{x}^{(i)}$ is misclassified then
$\qquad \boldsymbol{w}^{new} = \boldsymbol{w} + \boldsymbol{x}^{(i)} y^{(i)}$
$\qquad$ if $E_{train}(\boldsymbol{w}^{new}) < E_{train}(\boldsymbol{w})$ then
$\qquad\quad \boldsymbol{w} = \boldsymbol{w}^{new}$
end

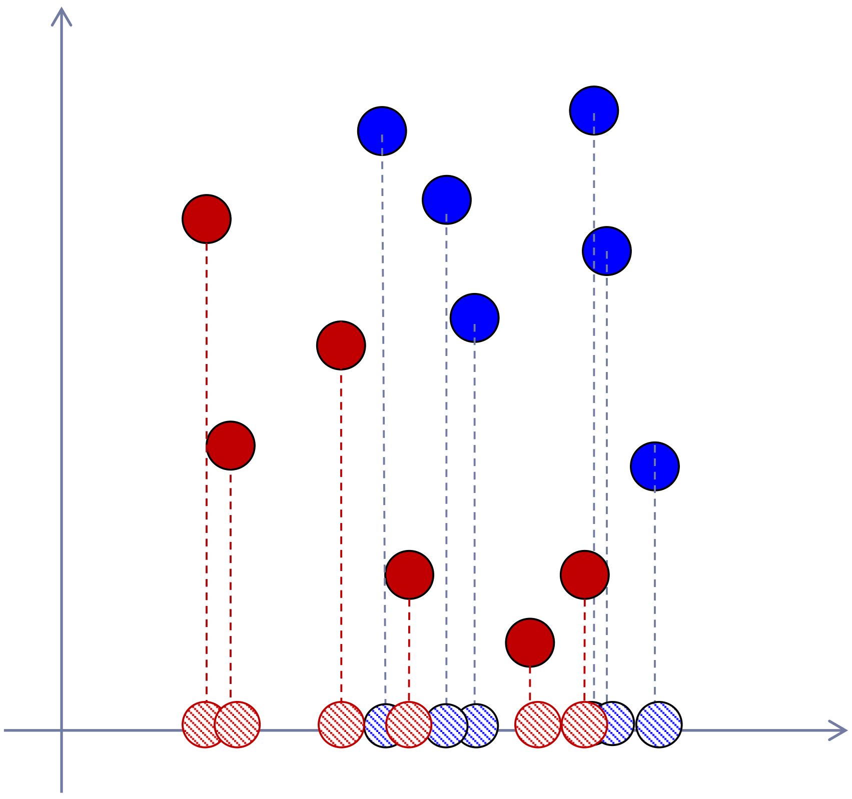
$$E_{train}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \left[sign(\boldsymbol{w}^T \boldsymbol{x}^{(n)}) \neq y^{(n)} \right]$$

Linear Discriminant Analysis (LDA)

- Fisher's Linear Discriminant Analysis :
  - Dimensionality reduction
    - Finds linear combinations of features with large ratios of between-groups scatters to within-groups scatters (as discriminant new variables)
  - Classification
    - Predicts the class of an observation x by first projecting it to the space of discriminant variables and then classifying it in this space

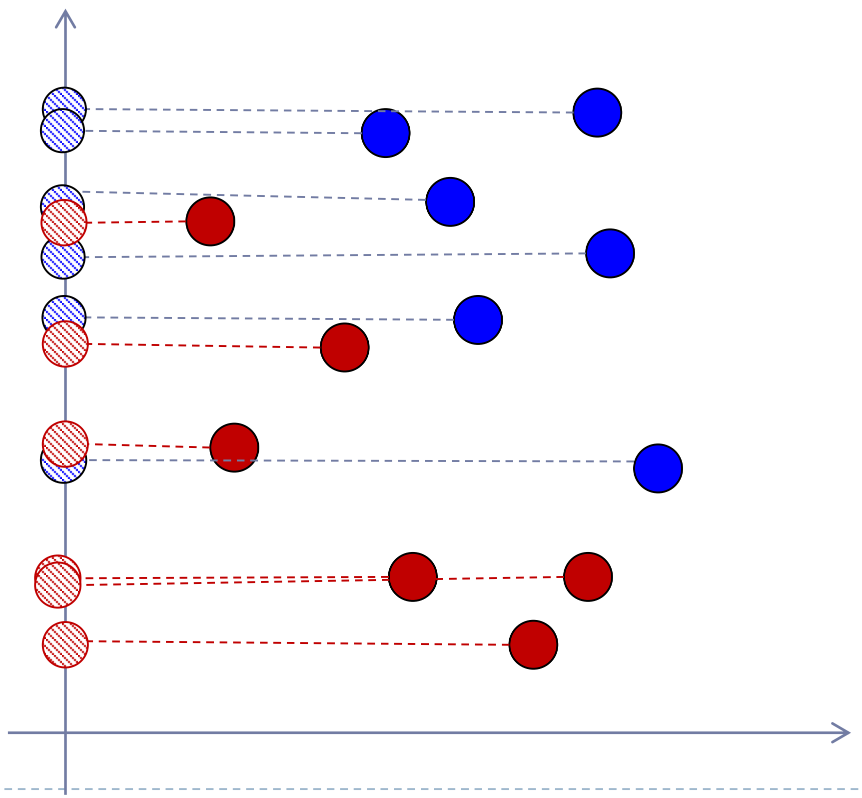
Good Projection for Classification

- What is a good criterion?
  - Separating different classes in the projected space



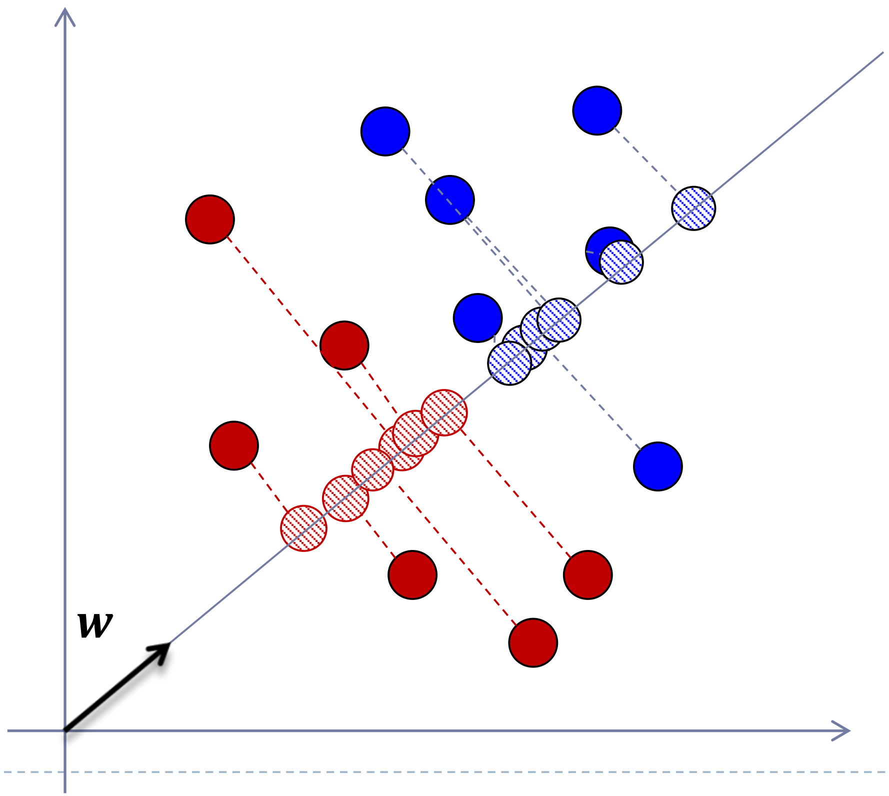
Good Projection for Classification

- What is a good criterion?
  - Separating different classes in the projected space



Good Projection for Classification

- What is a good criterion?
  - Separating different classes in the projected space



LDA Problem

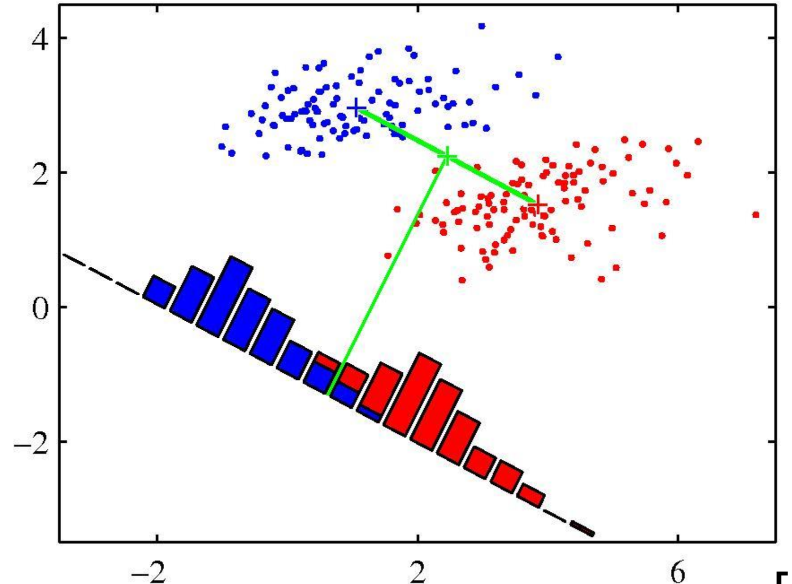
- Problem definition:
  - $C = 2$ classes
  - $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ training samples with N_1 samples from the first class ($\mathcal{C}_1$) and N_2 samples from the second class ($\mathcal{C}_2$)
  - Goal: finding the best direction $\boldsymbol{w}$ that we hope to enable accurate classification

- The projection of sample $\boldsymbol{x}$ onto a line in direction $\boldsymbol{w}$ is $\boldsymbol{w}^T\boldsymbol{x}$

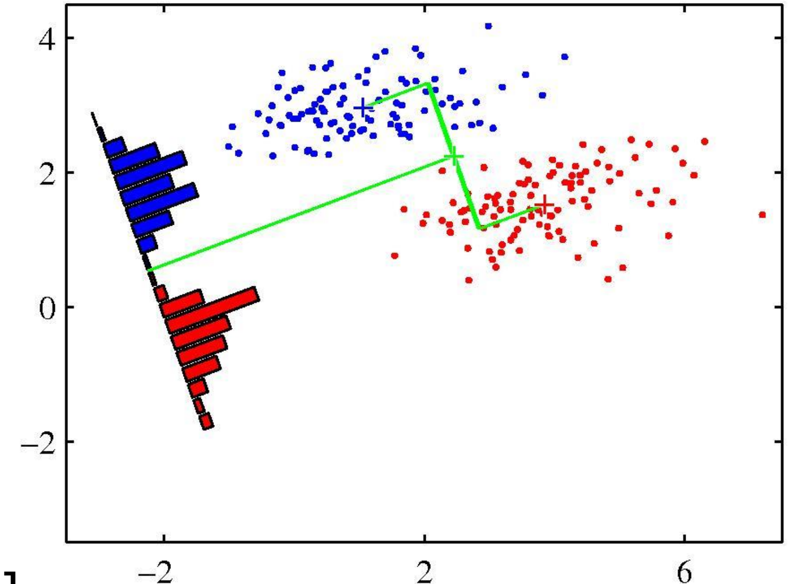
- What is the measure of the separation between the projected points of different classes?

Measure of Separation in the Projected Direction

- Is the direction of the line jointing the class means a good candidate for $\boldsymbol{w}$?



[Bishop]



Measure of Separation in the Projected Direction

- The direction of the line jointing the class means is the solution of the following problem:
  - Maximizes the separation of the projected class means

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = (\mu_1' - \mu_2')^2$$
$$\text{s.t. } \|\boldsymbol{w}\| = 1$$

$$\mu_1' = \boldsymbol{w}^T \boldsymbol{\mu}_1 \qquad \boldsymbol{\mu}_1 = \frac{\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \boldsymbol{x}^{(i)}}{N_1}$$

$$\mu_2' = \boldsymbol{w}^T \boldsymbol{\mu}_2 \qquad \boldsymbol{\mu}_2 = \frac{\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_2} \boldsymbol{x}^{(i)}}{N_2}$$

- What is the problem with the criteria considering only $|\mu_1' - \mu_2'|$?
  - It does not consider the variances of the classes in the projected direction

LDA Criteria

- Fisher idea: maximize a function that will give
  - large separation between the projected class means
  - while also achieving a small variance within each class, thereby minimizing the class overlap.

$$J(\boldsymbol{w}) = \frac{|\mu_1' - \mu_2'|^2}{s_1'^2 + s_2'^2}$$

LDA Criteria

- The scatters of the original data are:

$$s_1^2 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1 \right\|^2$$

$$s_2^2 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_2} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_2 \right\|^2$$

- The scatters of projected data are:

$$s_1'^2 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} - \boldsymbol{w}^T \boldsymbol{\mu}_1 \right)^2$$

$$s_2'^2 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_2} \left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} - \boldsymbol{w}^T \boldsymbol{\mu}_1 \right)^2$$

LDA Criteria

$$J(\boldsymbol{w}) = \frac{|\mu_1' - \mu_2'|^2}{s_1'^2 + s_2'^2}$$

$$|\mu_1' - \mu_2'|^2 = |\boldsymbol{w}^T\boldsymbol{\mu}_1 - \boldsymbol{w}^T\boldsymbol{\mu}_2|^2$$

$$= \boldsymbol{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\boldsymbol{w}$$

$$s_1'^2 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - \boldsymbol{w}^T\boldsymbol{\mu}_1\right)^2$$

$$= \boldsymbol{w}^T\left(\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1\right)^T\right)\boldsymbol{w}$$

LDA Criteria

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$$

Between-class scatter matrix ← $\boldsymbol{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$

Within-class scatter matrix ← $\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$

$$\boldsymbol{S}_1 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1\right)^T$$

$$\boldsymbol{S}_2 = \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_2} \left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_2\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_2\right)^T$$

scatter matrix=N×covariance matrix

LDA Derivation

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\frac{\partial \mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\partial \mathbf{w}} \times \mathbf{w}^T \mathbf{S}_W \mathbf{w} - \frac{\partial \mathbf{w}^T \mathbf{S}_W \mathbf{w}}{\partial \mathbf{w}} \times \mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\left(\mathbf{w}^T \mathbf{S}_W \mathbf{w}\right)^2} = \frac{\left(2\mathbf{S}_B \mathbf{w}\right)\mathbf{w}^T \mathbf{S}_W \mathbf{w} - \left(2\mathbf{S}_W \mathbf{w}\right)\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\left(\mathbf{w}^T \mathbf{S}_W \mathbf{w}\right)^2}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}$$

LDA Derivation

If $\boldsymbol{S}_W$ is full-rank

$$\boldsymbol{S}_B\boldsymbol{w} = \lambda\boldsymbol{S}_W\boldsymbol{w} \Longrightarrow \boldsymbol{S}_W^{-1}\boldsymbol{S}_B\boldsymbol{w} = \lambda\boldsymbol{w}$$

- $\boldsymbol{S}_B\boldsymbol{w}$ (for any vector $\boldsymbol{w}$) points in the same direction as $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$:

$$\boldsymbol{S}_B\boldsymbol{w} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\boldsymbol{w} \propto (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

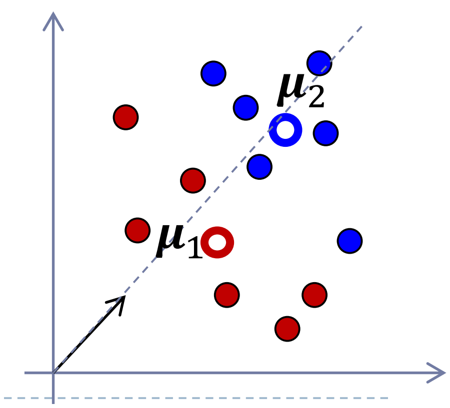
$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

- Thus, we can solve the eigenvalue problem immediately

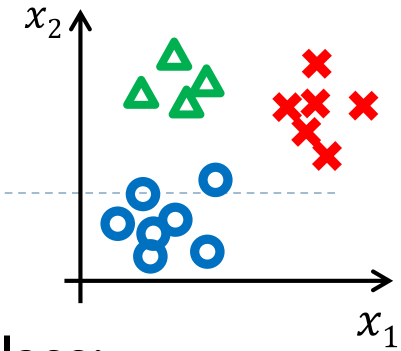
LDA Algorithm

- Find $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ as the mean of class 1 and 2 respectively
- Find $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ as scatter matrix of class 1 and 2 respectively
- $\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$
- $\boldsymbol{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$

- Feature Extraction
  - $\boldsymbol{w} = \boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ as the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_b$
- Classification
  - $\boldsymbol{w} = \boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$
  - Using a threshold on $\boldsymbol{w}^T\boldsymbol{x}$, we can classify $\boldsymbol{x}$



Multi-class classification



- Solutions to multi-category problems:
  - Extend the learning algorithm to support multi-class:
    - A function $f_i(\boldsymbol{x})$ for each class i is found
      - $\hat{y} = \underset{i=1,\dots,c}{\operatorname{argmax}} f_i(\boldsymbol{x})$

$\boldsymbol{x}$ is assigned to class C_i if $f_i(\boldsymbol{x}) > f_j(\boldsymbol{x}) \quad \forall j \neq i$

  - Converting the problem to a set of two-class problems:

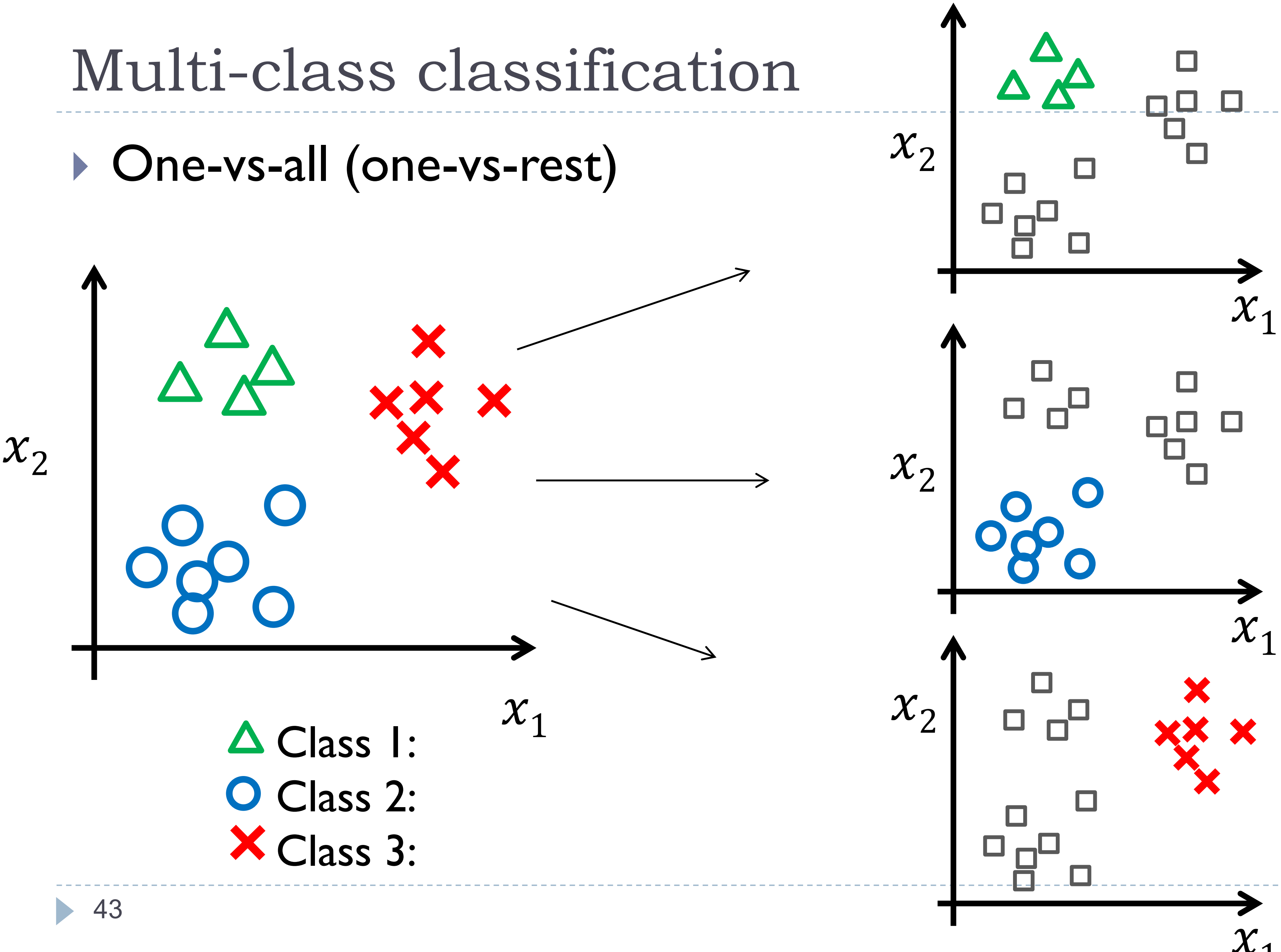
Converting multi-class problem to a set of two-class problems

- "**one versus rest**" or "one against all"
  - For each class C_i, a linear discriminant function that separates samples of C_i from all the other samples is found.
    - Totally linearly separable

- "**one versus one**"
  - $c(c-1)/2$ linear discriminant functions are used, one to separate samples of a pair of classes.
    - Pairwise linearly separable

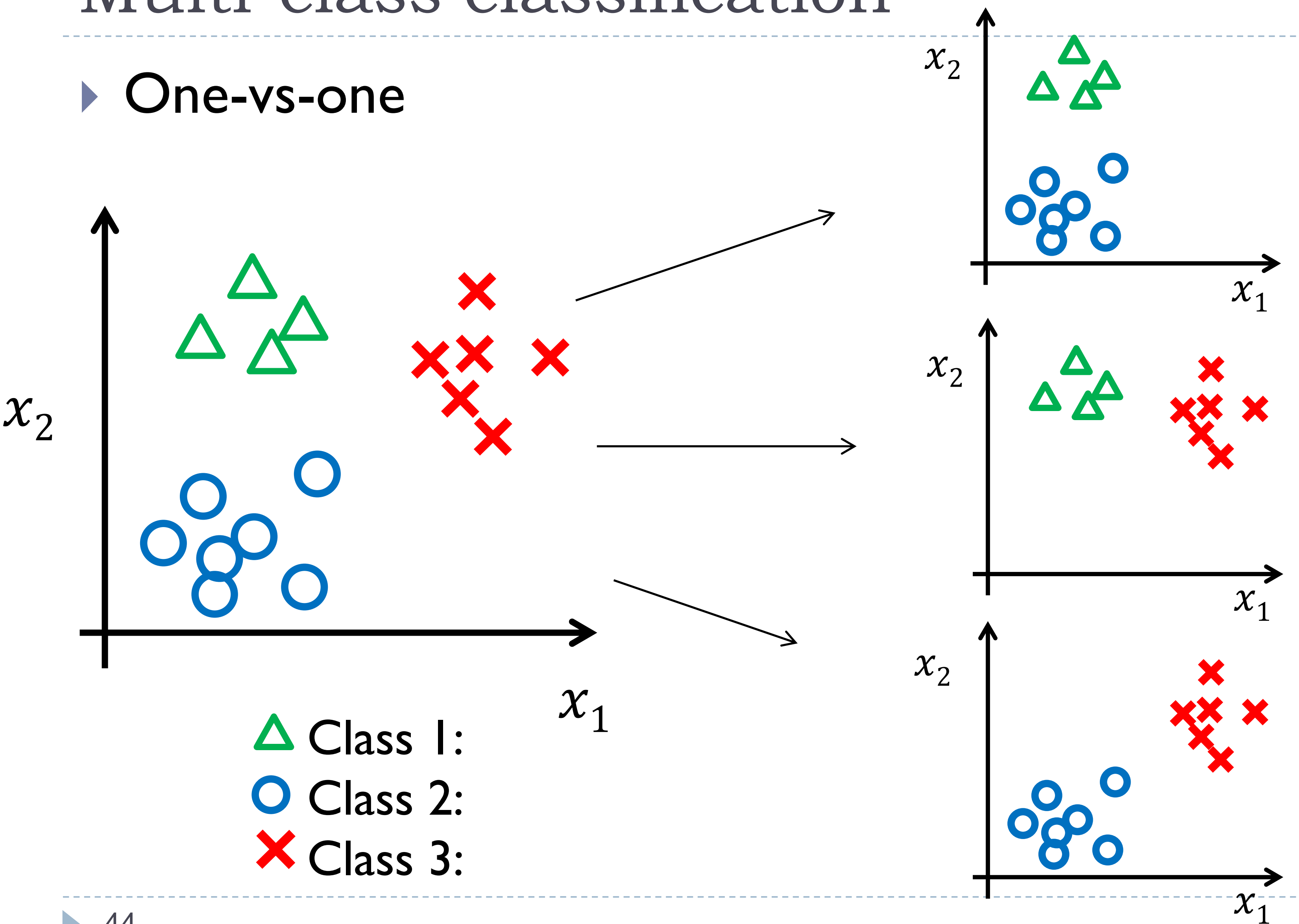
Multi-class classification

- One-vs-all (one-vs-rest)



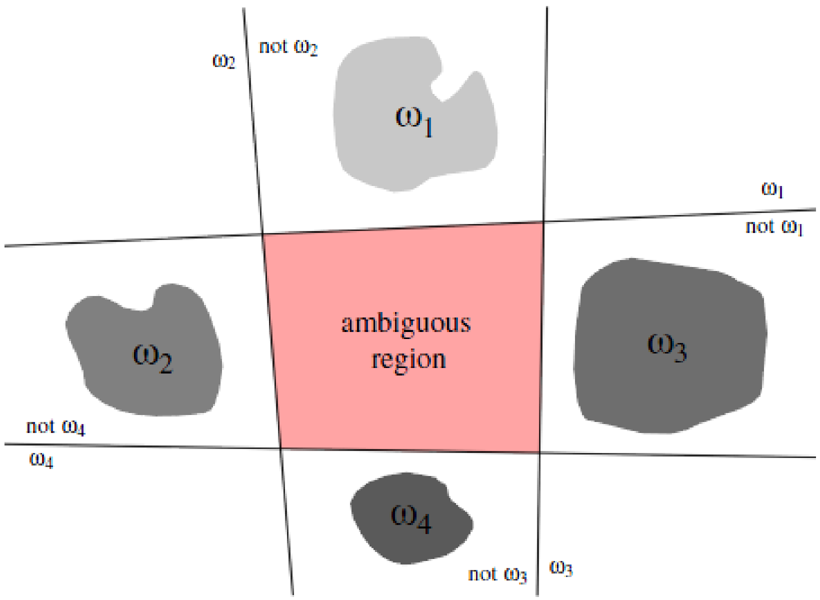
Multi-class classification

- One-vs-one

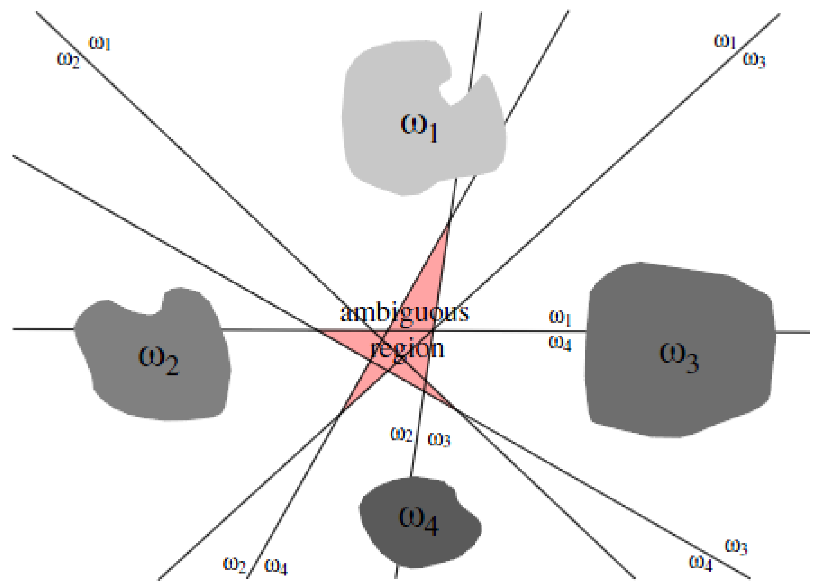


Multi-class classification: ambiguity

- Converting the multi-class problem to a set of two-class problems can lead to **regions in which the classification is undefined**



one versus rest



one versus one

[Duda, Hart & Stork, 2002]

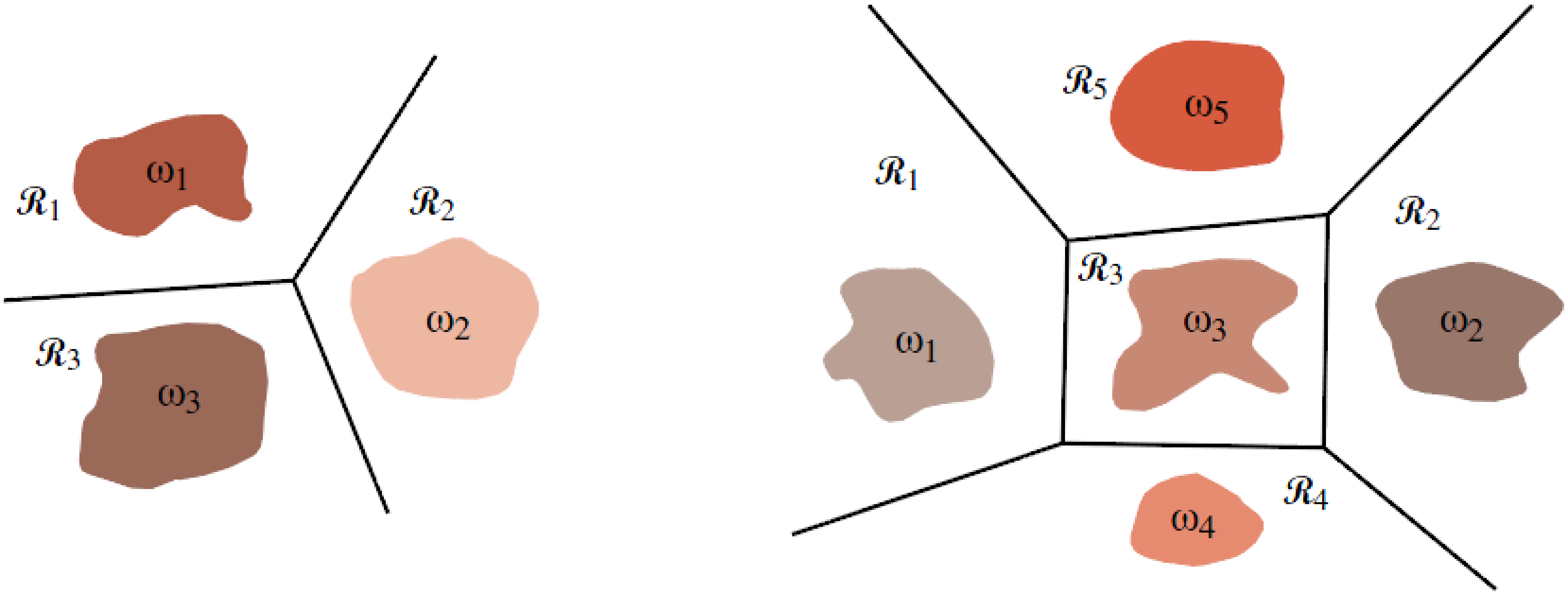
Multi-class classification: linear machine

- A discriminant function $f_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}$ for each class $\mathcal{C}_i$ $(i = 1, \dots, K)$:
  - $\boldsymbol{x}$ is assigned to class $\mathcal{C}_i$ if:

$$f_i(\boldsymbol{x}) > f_j(\boldsymbol{x}) \quad \forall j \neq i$$

- **Decision surfaces** (boundaries) can also be found using discriminant functions
  - Boundary of the contiguous $\mathcal{R}_i$ and $\mathcal{R}_j$: $\forall \boldsymbol{x}, f_i(\boldsymbol{x}) = f_j(\boldsymbol{x})$
  - $(\boldsymbol{w}_i - \boldsymbol{w}_j)^T\boldsymbol{x} + (w_{i0} - w_{j0}) = 0$

Multi-class classification: linear machine



[Duda, Hart & Stork, 2002]

Perceptron: multi-class

$$\hat{y} = \underset{i=1,\dots,c}{\operatorname{argmax}} \boldsymbol{w}_i^T \boldsymbol{x}$$

$$J_P(\boldsymbol{W}) = -\sum_{i \in \mathcal{M}} \left(\boldsymbol{w}_{y^{(i)}} - \boldsymbol{w}_{\hat{y}^{(i)}} \right)^T \boldsymbol{x}^{(i)}$$

$\mathcal{M}$: subset of training data that are misclassified

$\mathcal{M} = \left\{ i | \hat{y}^{(i)} \neq y^{(i)} \right\}$

Initialize $\boldsymbol{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_c], k \leftarrow 0$
repeat
  $k \leftarrow (k+1) \bmod N$
  if $\boldsymbol{x}^{(i)}$ is misclassified then
    $\boldsymbol{w}_{\hat{y}^{(i)}} = \boldsymbol{w}_{\hat{y}^{(i)}} - \boldsymbol{x}^{(i)}$
    $\boldsymbol{w}_{y^{(i)}} = \boldsymbol{w}_{y^{(i)}} + \boldsymbol{x}^{(i)}$
Until all patterns properly classified

Resources

- C. Bishop, “Pattern Recognition and Machine Learning”, Chapter 4.1.