Support Vector Machines

Based on the book titled Machine Learning & Pattern Recognition authored by C. Bishop and the slides prepared by A. Zisserman

Lecture outline

1. Linear classification
   a. Binary classifiers
   b. Perceptron
2. SVM
   a. SVM derivation
   b. SVM optimization
   c. Primal & dual formulation
   d. Hard margin & soft margin
3. SVM and kernels - Nonlinear SVM
   a. Linearly non-separable cases
   b. Kernels
4. Multiclass SVM
5. Advantages and disadvantages of SVM

Linear classification

Binary classification

Given training data (x_i, y_i), for i=1,2,...,N, with

$x_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$,

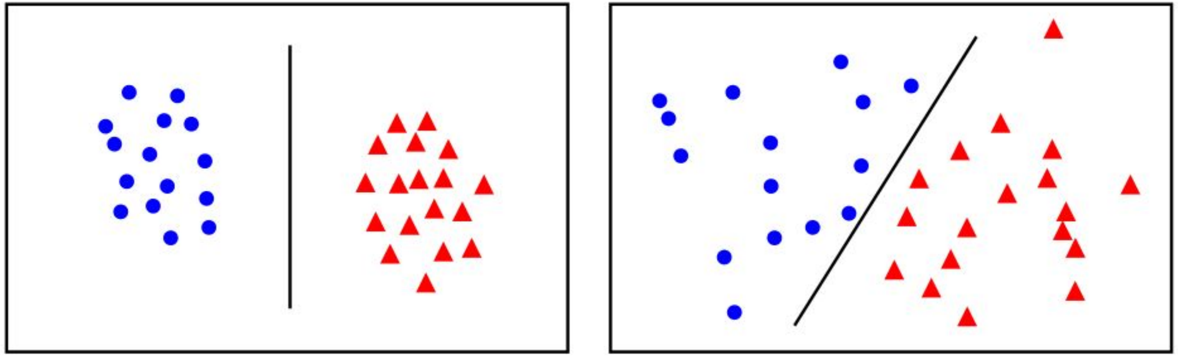
Learn a classifier f(x) such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

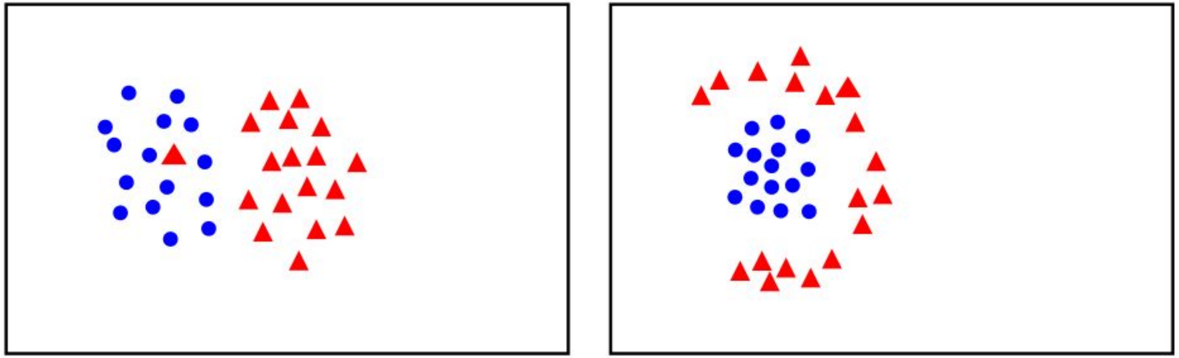
=> $y_i f(x_i) > 0$ for a correct classification

Linear Separability

Linearly separable



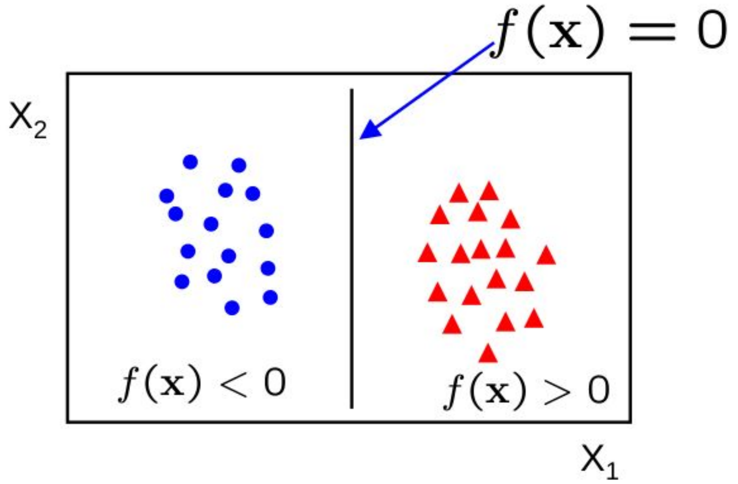
Not linearly separable



Linear classifiers

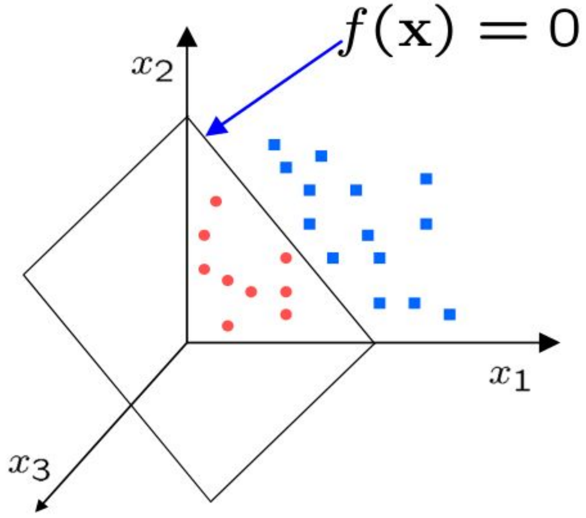
A linear classifier has the following form:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- The **2D** discriminant is a **line**
- **w** is the normal to the line.
  - It's called the **weight vector**
- **b** is the bias
- Both **w** and **b** are learned from the training data
  - To classify new data, we only need **w** and **b**

Linear classifiers (2)



A linear classifier has the following form:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

- In **3D**, the discriminant is a **plane**
- In **4D** and above, the discriminant is a **hyperplane**

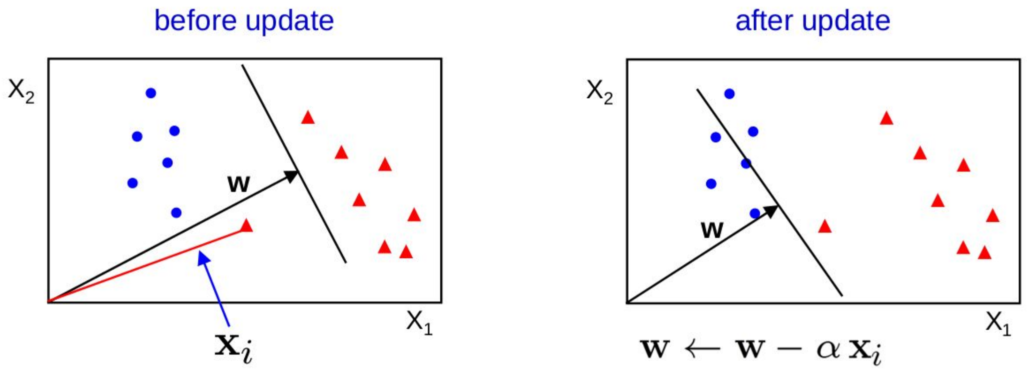
Perceptron classifier

The Perceptron algorithms is one way of finding the separating line/hyperplane $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$

Perceptron algorithm

- Define the classifier as: $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^\top \mathbf{x}_i$
  - Where $\mathbf{w} = (\tilde{\mathbf{w}}, w_0), \mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$
- Initialize **w** = 0
- Cycle through the data points $\{x_i, y_i\}$
  - If x_i is misclassified then, $w \leftarrow w + \alpha \mathrm{sign}(f(x_i))x_i$
- Until all data points are correctly classified

Perceptron classifier (2)

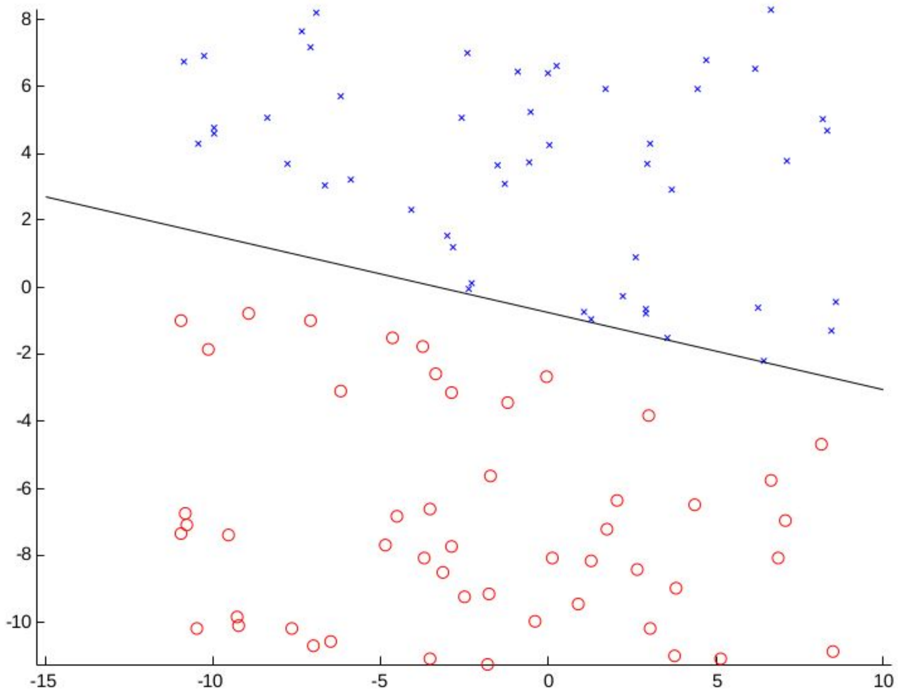


Example in 2D

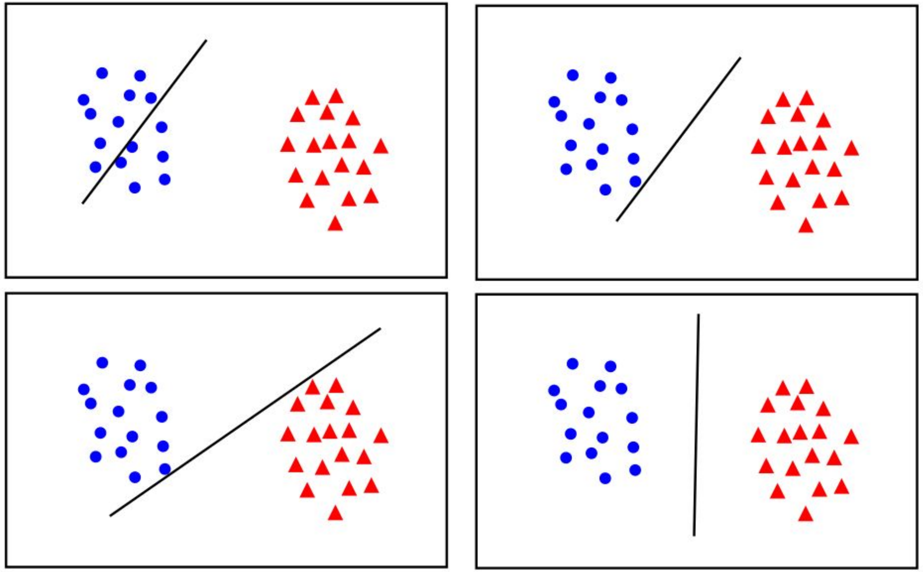
- Initialize **w** = 0
- Cycle through the data points $\{x_i, y_i\}$
  - If xi is misclassified then, $w \leftarrow w + \alpha \mathrm{sign}(f(x_i)) x_i$
- Until all data points are correctly classified

Perceptron classifier (3)

- If the data is linearly separable, then the algorithm will converge
- Convergence can be slow
- Separating line can be very close to training data
- For better generalizability, we would prefer a bigger margin between training data and the separating line



What is the best "w"? The maximum margin solution

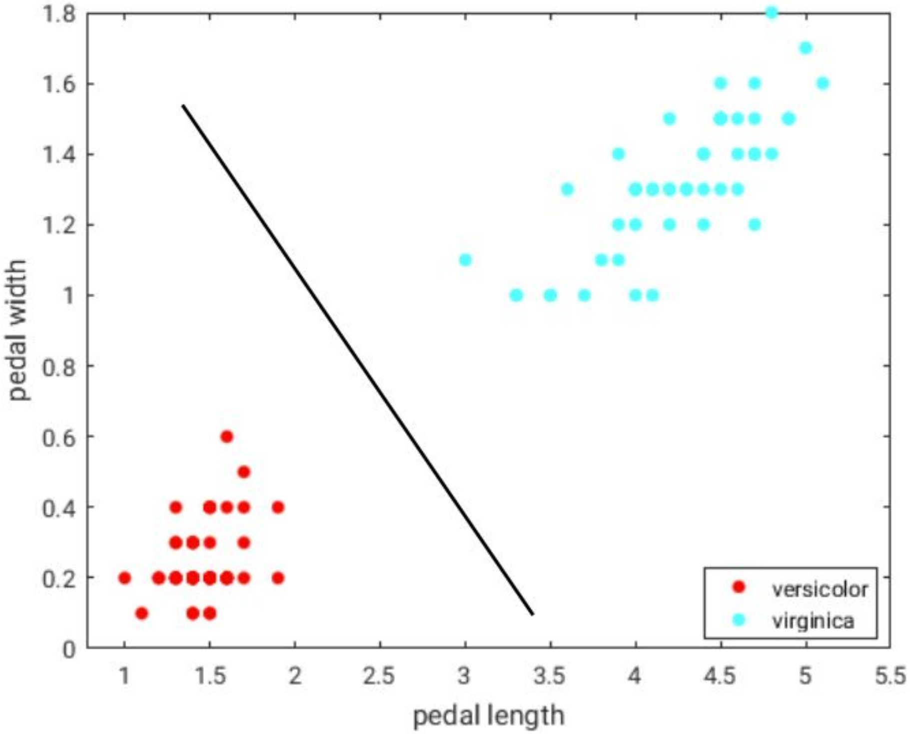


Maximum margin solution: most suitable under perturbations of the inputs

Support Vector Machine

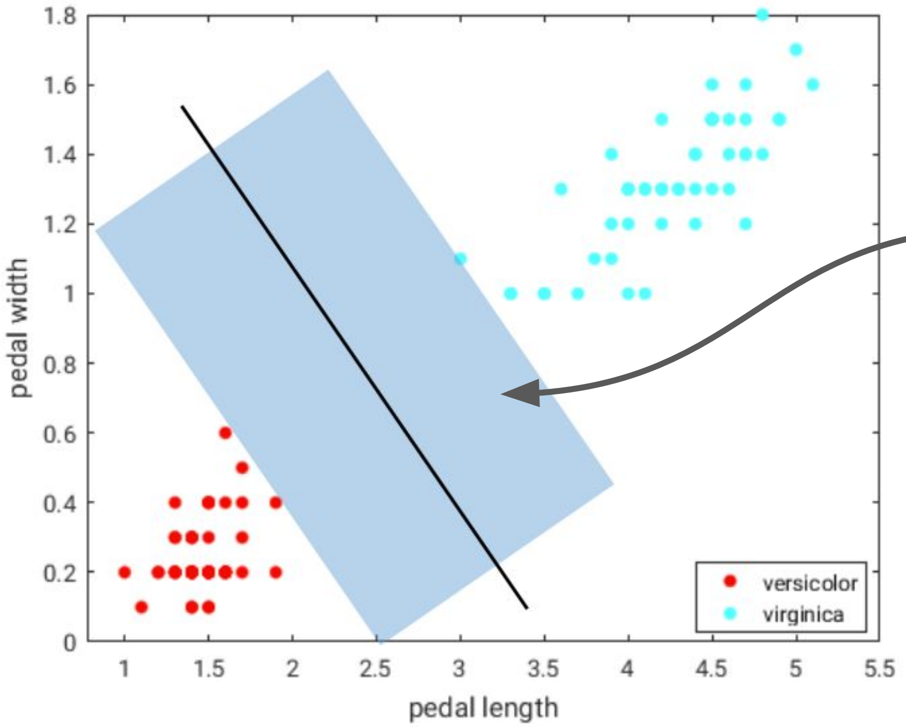
A maximum margin classifier

SVM



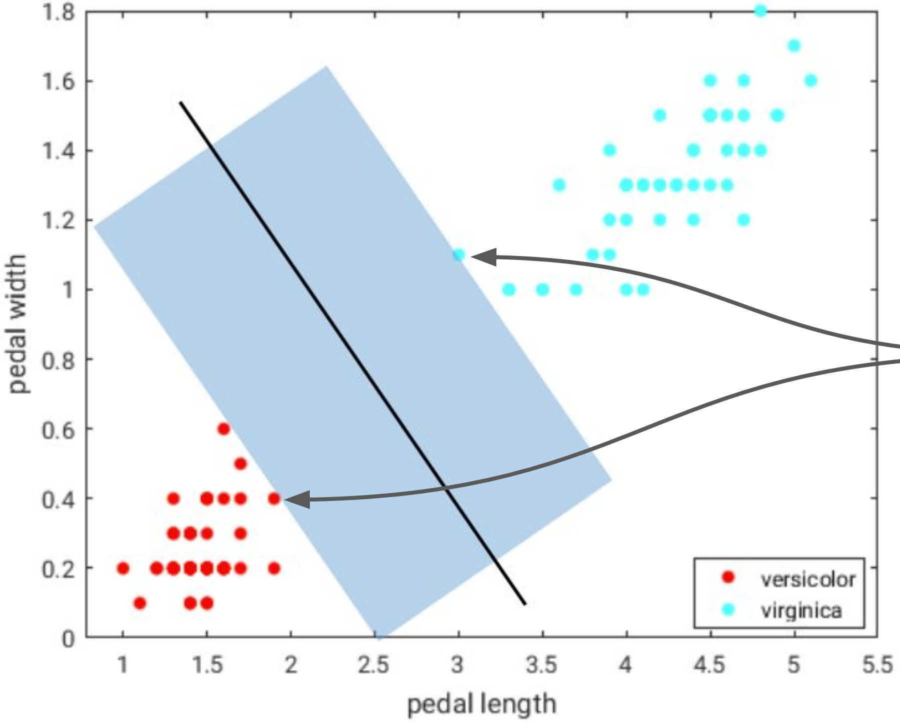
- Find the separating hyperplane that maximizes the margin between the two classes

SVM



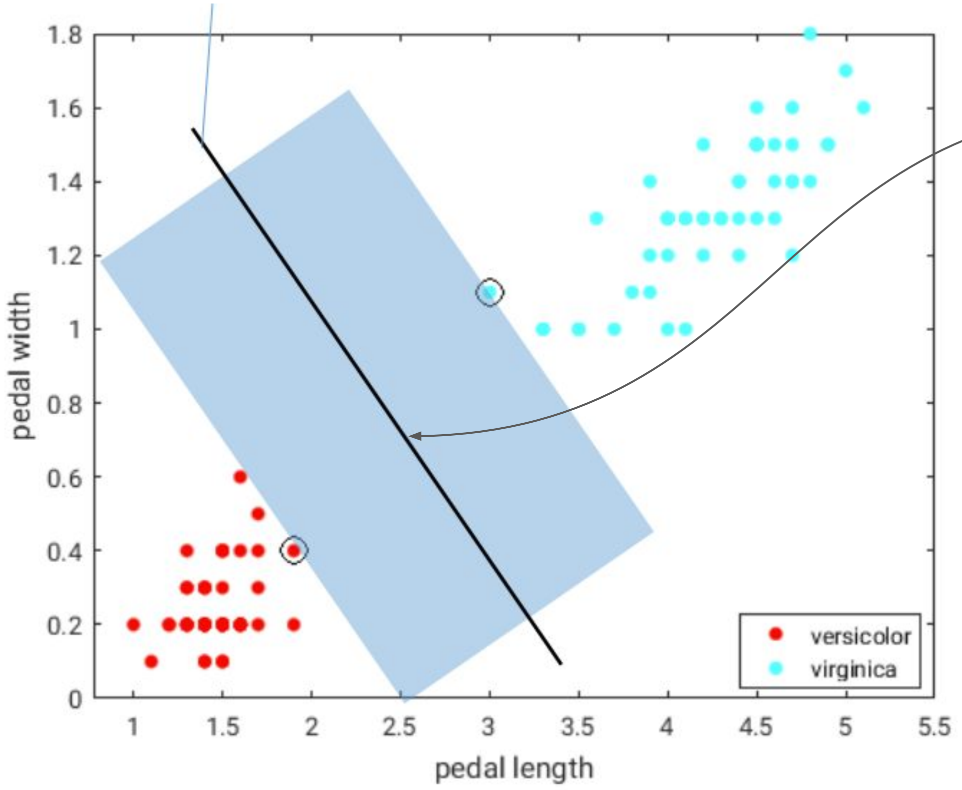
- Find the separating hyperplane that maximizes the margin between the two classes
- Margin = width of the boundary before hitting any data points

SVM



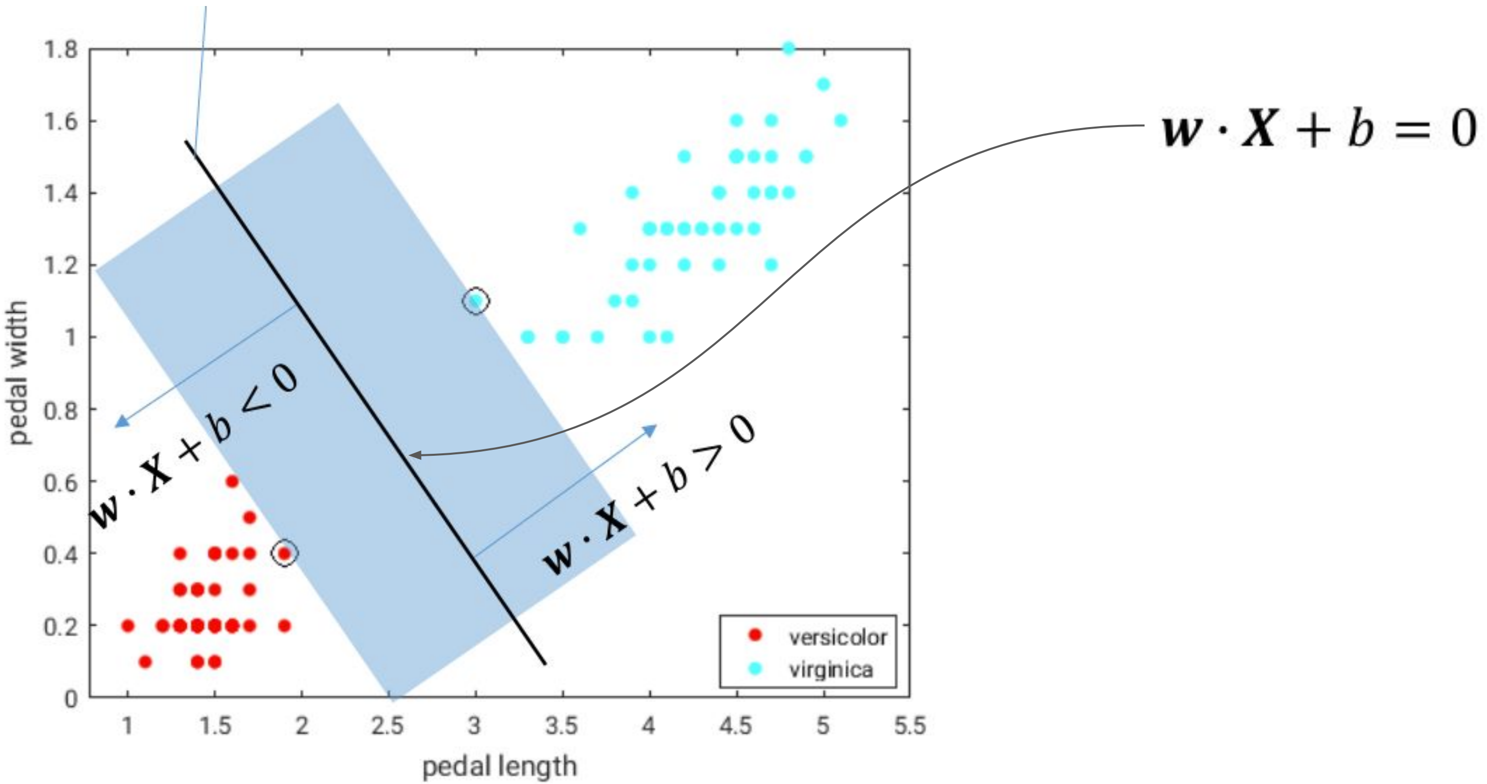
- Find the separating hyperplane that maximizes the margin between the two classes
- Margin = width of the boundary before hitting any data points
- Only the support vectors (points on the boundary) matter when learning the separating hyperplane

SVM: The separating hyperplane

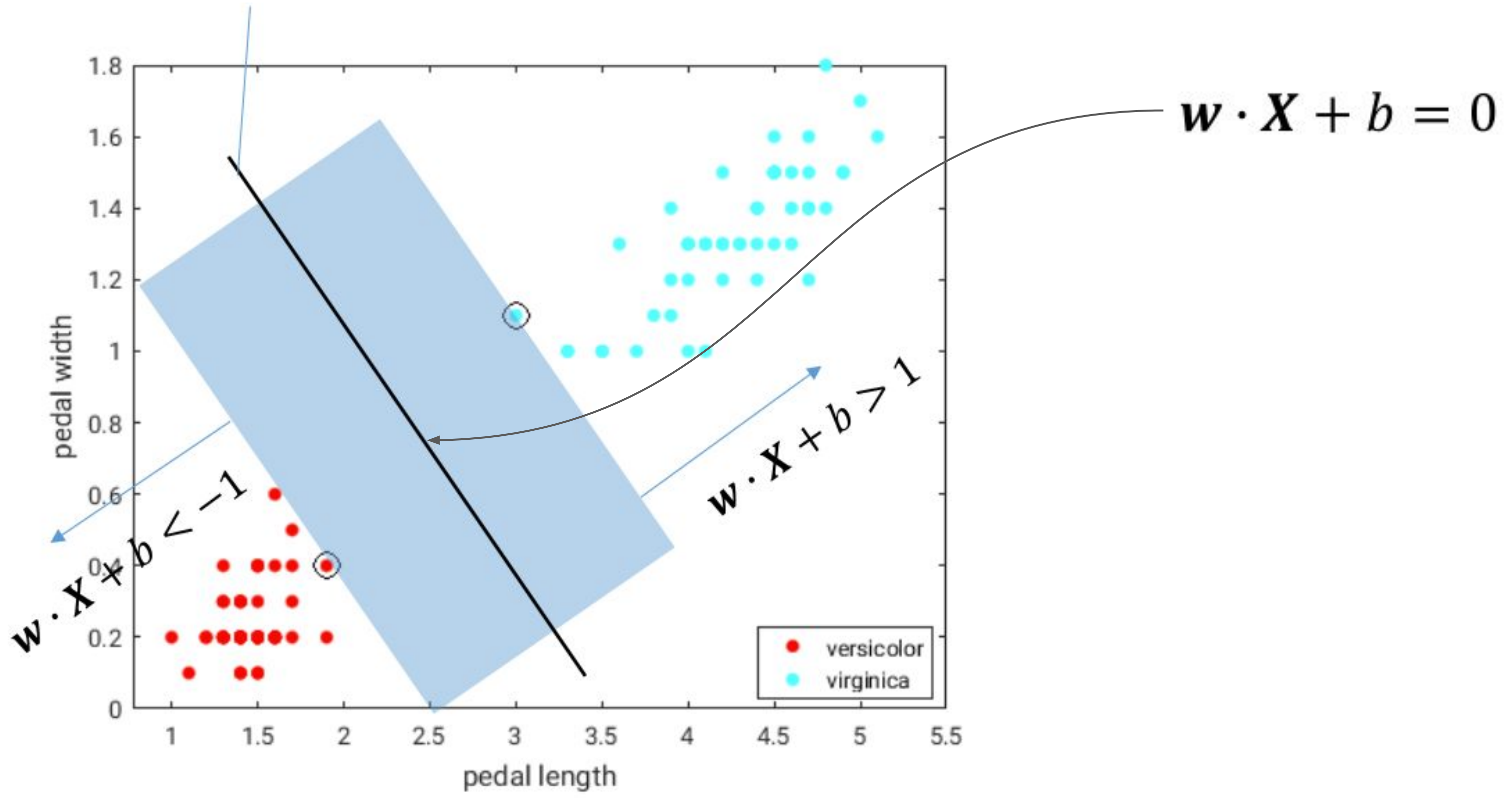


$$\boldsymbol{w} \cdot \boldsymbol{X} + b = 0$$

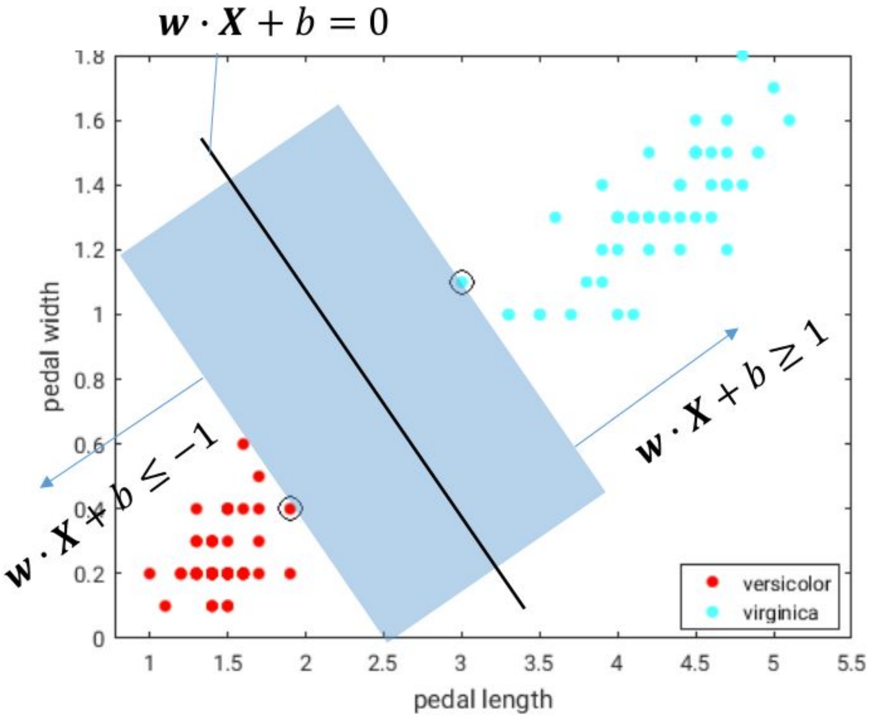
SVM: The separating hyperplane



SVM: The separating hyperplane



SVM Decision Rule



- If we set labels y_i of two classes as +1 and -1
- We have

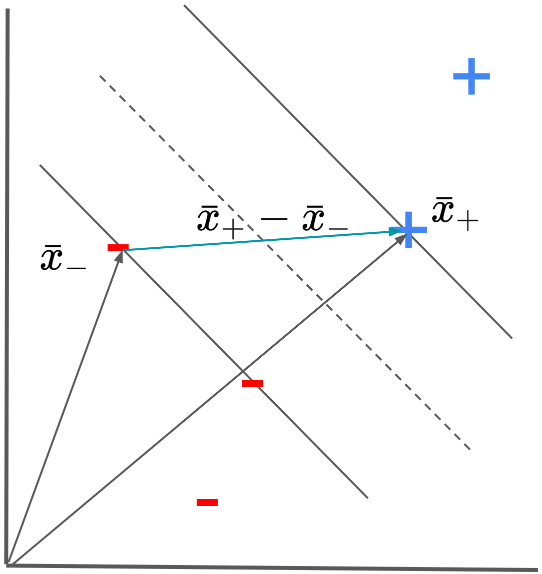
$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x_i} + b \geq 1, & if\ y_i = 1 \\ \boldsymbol{w} \cdot \boldsymbol{x_i} + b \leq -1, & if\ y_i = -1 \end{cases}$$

which is equivalent to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1$$

- This is the constraint the separating hyperplane needs to satisfy

Maximize the margin

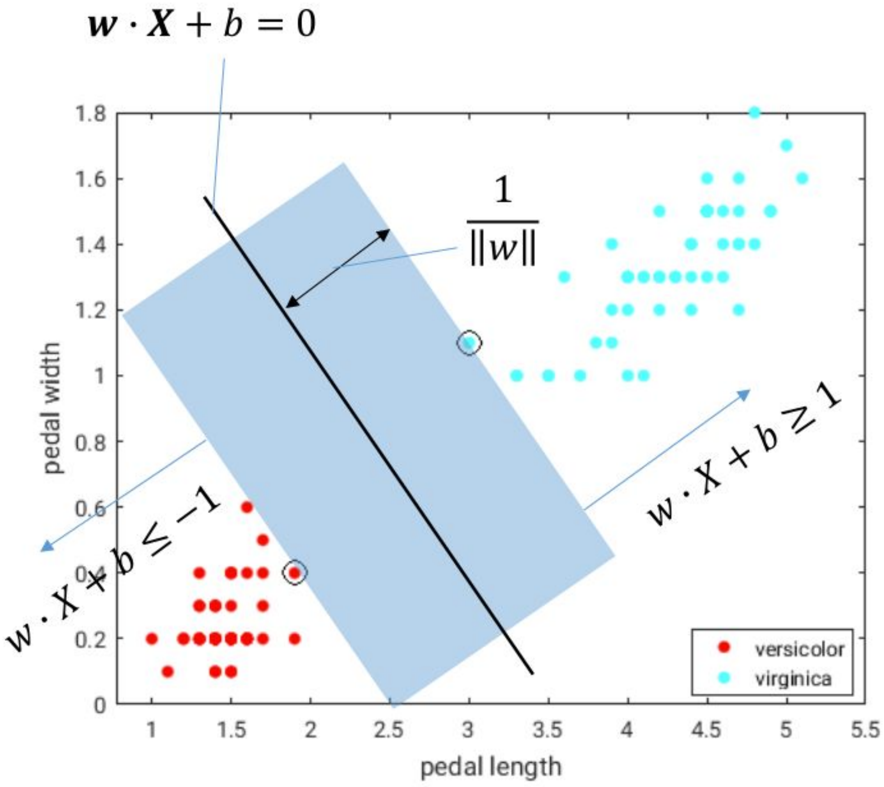


$$\bar{w} \cdot \bar{x}_+ + b = 1$$

$$\bar{w} \cdot \bar{x}_- + b = -1$$

$$width = \underbrace{(\bar{x}_+ - \bar{x}_-)}_{\frac{2}{\bar{w}}} \cdot \frac{\bar{w}}{\|w\|} = \frac{2}{\|w\|}$$

SVM: Optimization problem to find the separating hyperplane



- Margin is $\frac{2}{\|\boldsymbol{w}\|}$

- SVM : maximize the margin, so that it becomes the optimization below

Minimize

$$\Phi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$$

s.t.

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 \;\; for\ all\ i$$

Lagrangian constrained optimization

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to } & f_i(x) \leq 0, \qquad i = 1, \ldots, m \end{aligned}$$

Lagrangian function:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x)$$

Solution: Lagrange method of optimization

Minimize

$$\Phi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$$

s.t.

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 \ \ for\ all\ i$$

- Lagrange Method

Minimize $$L_p(b, \boldsymbol{w}, a_i) = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} a_i\{y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1\}$$

where $a_i \geq 0$

Solution: Lagrangian dual formulation

Primal variables

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} a_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{l} a_i y_i = 0$$

Dual variables

$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{l} a_i y_i = 0$$

This derivation won't be tested in the exams

Solution: Lagrangian dual formulation (2)

- Substitute w and b in the original equation
- Dual problem: find a_i to maximize:

$$L_D(a_i) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} a_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} a_i a_j y_i y_j \boldsymbol{x_i x_j}$$

$$s.t. \quad \sum_{i=1}^{n} a_i y_i = \mathbf{0} \qquad \text{and} \qquad a_i \geq 0$$

This derivation won’t be tested in the exams

Solution

- Final solution

$$\boldsymbol{w} = \sum_{i=1}^{n} a_i y_i \boldsymbol{x_i}$$

- a_i are non-zero only when $\boldsymbol{x}_i$ are support vectors!

$$\boldsymbol{w} = \sum_{\boldsymbol{x_i} \in SV} a_i y_i \boldsymbol{x_i}$$

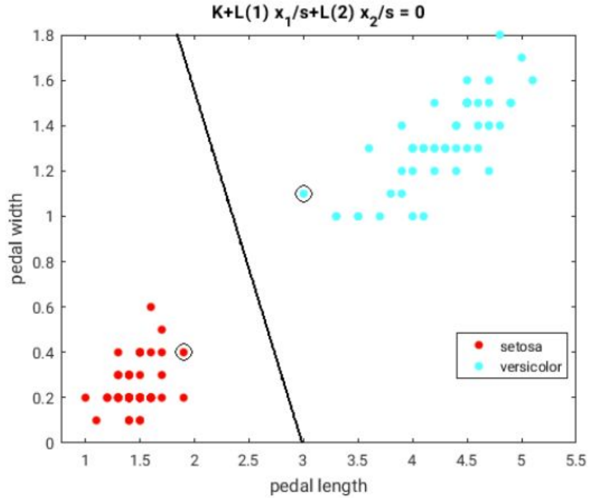
- For a new observation x, we can classify it by the sign of

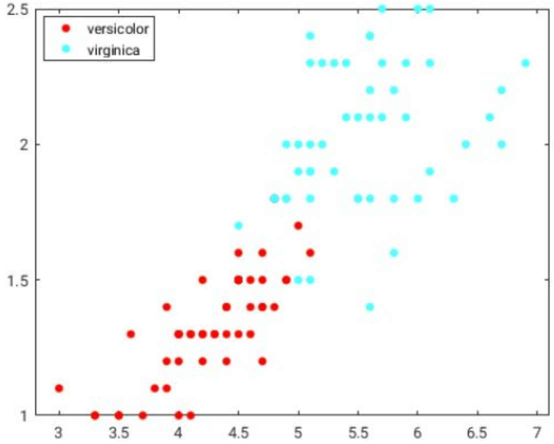
$$f(\boldsymbol{x}) = \sum_{\boldsymbol{x_i} \in SV} a_i y_i \boldsymbol{x_i}\, \boldsymbol{x} + b$$

Soft margin classifier

Hard margin vs soft margin

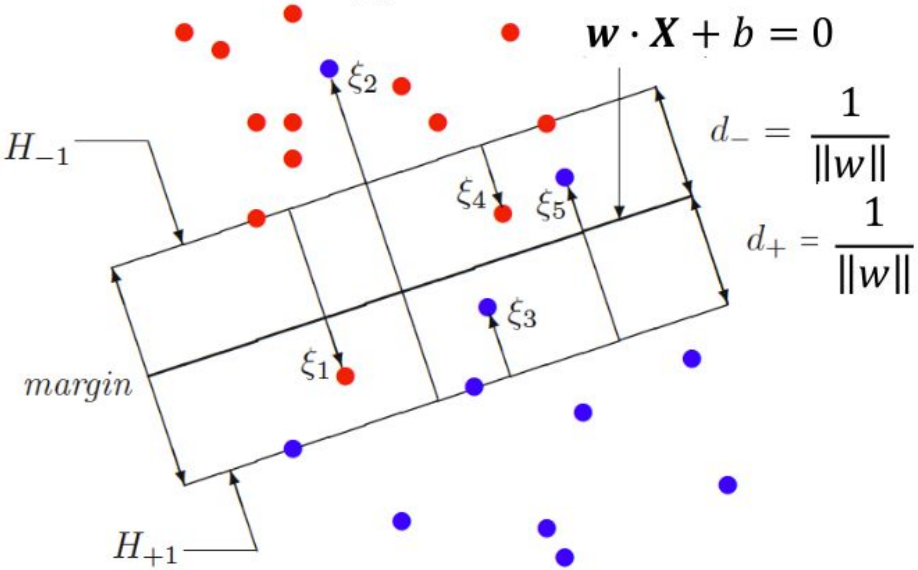
- So far we focus on the case where a separating hyperplane can be found to separate two classes correctly (Hard Margin Problem)
- What if there is no such a hyperplane that can separate two classes completely correct? (Soft Margin Problem?)



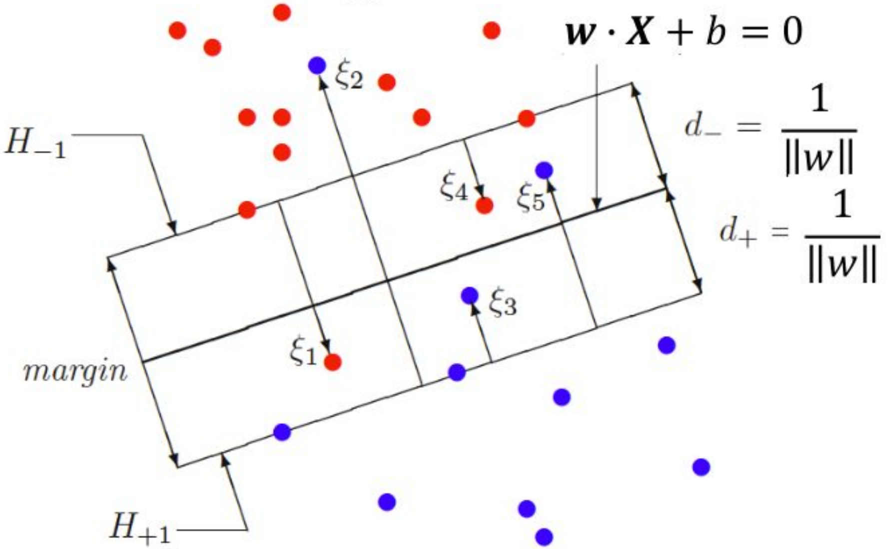


Soft margin classifier with “slack variables”

- Introduce slack variables ξ_i to describe additional offsets



- Introduce slack variables ξ_i to describe additional offsets



- Low c -> low training error, but can overfit.
- High c -> high training error but more generalizability
- C affects the number of support vectors chosen

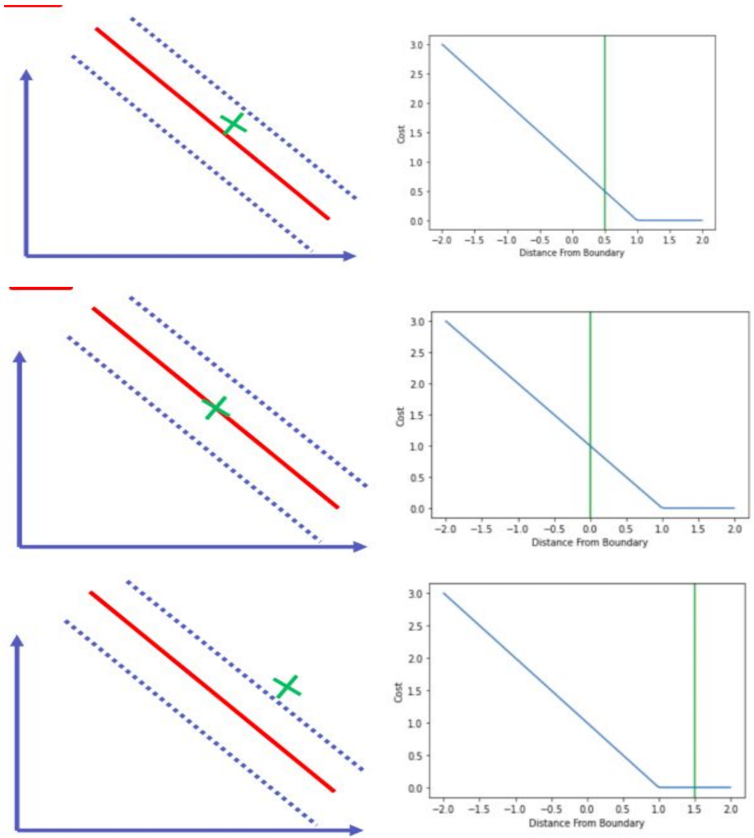
Minimize

$$L_p(b, \boldsymbol{w}, a_i) = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} - \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} a_i\{y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1\} + C\sum_{i=1}^{n} \xi_i$$

where $\quad a_i \geq 0 \quad and \quad \xi_i \geq 0$

$C \geq 0$ is the regularization parameter

Linear SVM with hinge loss



$$\min_{w,b} \frac{1}{2} w^T w + \boxed{C \sum_{i=1} \max(0, 1 - y_i(w^T \phi(x_i) + b))}$$

- The hinge loss term is applicable only to the Primal Formulation of the linear soft margin SVM. Not the Lagrangian Dual Formulation (up next!)
- Incorporates a margin or distance from the classification boundary into the cost calculation.
- Even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough. The hinge loss increases linearly.

Soft margin: Solution

- find a's to maximize

$$L_D(a_i) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} a_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} a_i a_j y_i y_j \boldsymbol{x_i x_j}$$

$$s.t. \quad \sum_{i=1}^{n} a_i y_i = \boldsymbol{0} \qquad \text{and} \qquad C \geq a_i \geq 0$$

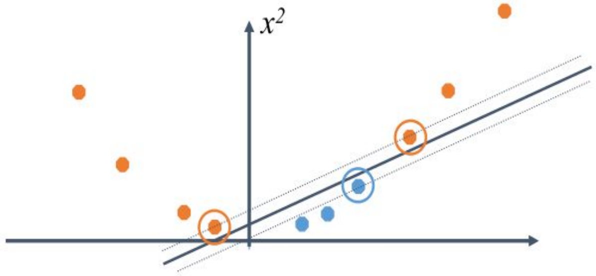
- The format of the solution is the same

$$\boldsymbol{w} = \sum_{\boldsymbol{x_i} \in SV} a_i y_i \boldsymbol{x_i}$$

Nonlinear SVM

Not linearly separable case

1-D example

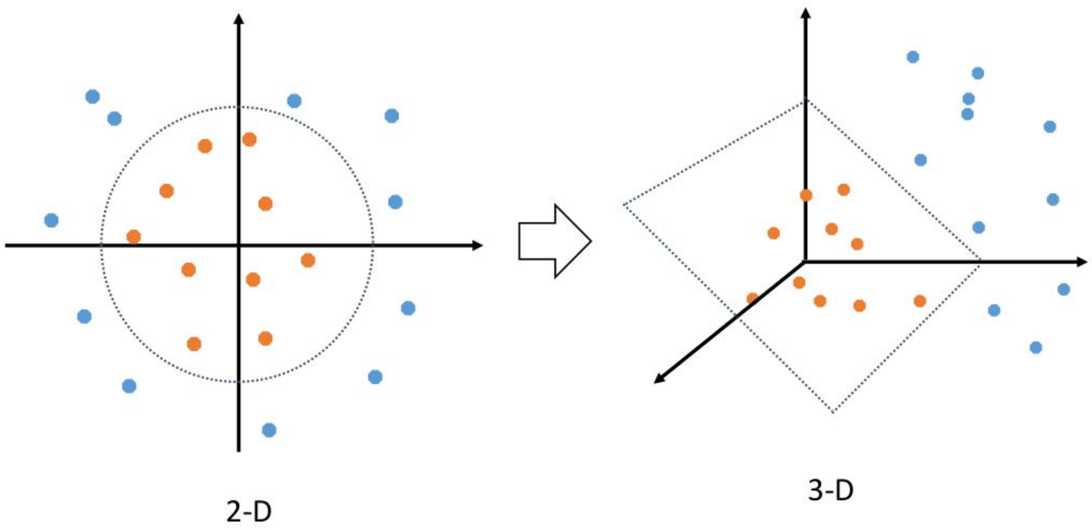


$$\phi(x) = (x^2, x)$$

Solution: map the data to a higher dimensional space

Not linearly separable case

General idea: transfer the data x, typically to a higher dimensional space $\phi(x)$ where they are linearly separable



Nonlinear SVM

- General idea: transfer the data $\phi(x)$, typically to a higher dimensional space where they are linearly separable
- Major issue: Calculating $\phi(x)$ is often computationally expensive and time consuming.
- Do we really need to explicitly know $\phi(x)$？

$$L_D(a_i) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} a_i a_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

Spoiler alert: No

Nonlinear SVM

$$L_D(a_i) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} a_i a_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j$$

- Do we really need explicitly know $\phi(x)$?
- If we have a kernel function

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$$

Then we don't actually need to know $\phi(x)$

This is the basis of the **Kernel Trick**

Kernel Trick

- Now we maximize

$$L_D(a_i) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} a_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} a_i a_j y_i y_j \boldsymbol{K(x_i, x_j)}$$

- Examples of Kernels
  - Polynomial Kernel $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (1 + \boldsymbol{x_i} \cdot \boldsymbol{x_j})^p$
  - Gaussian (RBF) Kernel $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\sigma^2}\right)^p$
  - Sigmoid $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = tanh(\beta_0 \boldsymbol{x_i} \cdot \boldsymbol{x_j} + \beta_1)$

$\gamma = \frac{1}{2\sigma^2}$ Decides the curvature allowed for the decision boundary

Every positive semi-definite symmetric function is a kernel!

> **Definition**. Let X be a $\mathbf{R}$-vector space. A bilinear map $K: X \times X \to \mathbf{R}$ is called *positive semi-definite*, iff we have $K(x, x) \geq 0$ for all $x \in X$. If moreover $K(x, x) = 0 \iff x = 0$, K is called *positive definite*.

SVM for more than 2 classes

- SVM doesn’t support multiclass classification natively. It supports binary classification
- For multiple classes, we have to break down the problem into multiple binary classification problems
- There are two approaches:
    a. One-to-one approach
    b. One-to-rest approach

SVM for more than 2 classes (2)

- One-to-one approach: breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes. Multiple classification results are combined using a majority vote
  - One vs One (OvO)
- One-to-rest approach: N number of classifiers for N number of classes. The classifier for class i is a binary classifier for class i vs the rest. The classifier which gives the most confident result (furthest from the decision boundary) is taken as the correct one.
  - One vs All (OvA)

SVM summary and recommended reading

- Flexibility in choosing a similarity function (kernel)
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property
  - a simple convex optimization problem which is guaranteed to converge to a single global solution
- Recommended video with python demo
- Online demo1, demo2, demo3
- Tutorial: Multi-class classification (refer next slide for an example)
- SVM with Scikit learn
- RBF SVM parameters (includes the tuning of c and γ)

Demo: Multiclass classification - One vs One (OvO)

- 3 classes A, B, C.
- Consider all the binary classifiers among them:
- For N classes the number of such binary classifiers will be:

$$ {}_{n}\mathrm{C}_{2} = \binom{n}{2} = \frac{n \cdot (n-1)}{2} $$

- A vs B, B vs C and A vs C. (3 classifiers)

A v B	B v C	C v A	Majority vote
60% A	51% B	42% C	A
45% B	55 % B	50% C	B