

Support Vector Machine Classifiers



Prof Dharmendra Sharma

Unit Codes: 11482, 11512

Outline

Linear discriminant

Vapnik (VC) model

Support Vector machine - SVM

Problem of maximizing a margin

Mathematics behind SVM

How does the SVM work?

SVM Kernels

Application of SVM

Linear Discriminant Functions

Generative vs Discriminant Approach

- **Generative** approaches estimate the **discriminant function** by first estimating the probability distribution of the data belonging to each class.
- **Discriminative** approaches estimate the **discriminant function** explicitly, without assuming a probability distribution.

Linear Discriminants (case of **two** categories)

- A **linear discriminant** has the following form:

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0 = \sum_{i=1}^d w_i x_i + w_0$$

Decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$

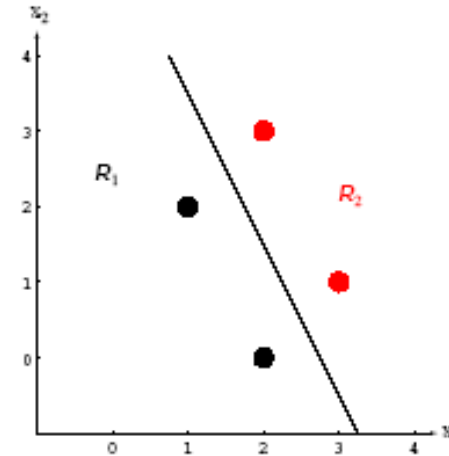
If $g(\mathbf{x})=0$, then \mathbf{x} lies on the **decision boundary** and can be assigned to either class.

Decision Boundary

- The **decision boundary** $g(\mathbf{x})=0$ is a **hyperplane**.
- The orientation of the hyperplane is determined by \mathbf{w} and its location by w_0 .

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

- \mathbf{w} is the normal to the hyperplane.
- If $w_0=0$, it passes through the origin



Decision Boundary Estimation

- Use “learning” algorithms to estimate \mathbf{w} and w_0 from training data \mathbf{x}_k .
- Let us suppose that:

true class label

$$z_k = \begin{cases} +1 & \text{if } \mathbf{x}_k \in \omega_1 \\ -1 & \text{if } \mathbf{x}_k \in \omega_2 \end{cases}$$

predicted class label:

$$\hat{z}_k = \begin{cases} +1 & \text{if } g(\mathbf{x}_k) > 0 \\ -1 & \text{if } g(\mathbf{x}_k) < 0 \end{cases}$$

- The solution can be found by minimizing an error function, e.g., the “training error” or “empirical risk”:

$$J(\mathbf{w}, w_0) = \frac{1}{n} \sum_{k=1}^n [z_k - \hat{z}_k]^2$$

Learning through “empirical risk” minimization

- Typically, a discriminant function $\mathbf{g}(\mathbf{x})$ is estimated from a finite set of examples by **minimizing** an error function (e.g., the **training error**):

$$J(\mathbf{w}, w_0) = \frac{1}{n} \sum_{k=1}^n [z_k - \hat{z}_k]^2$$

empirical risk
minimization

true class label:

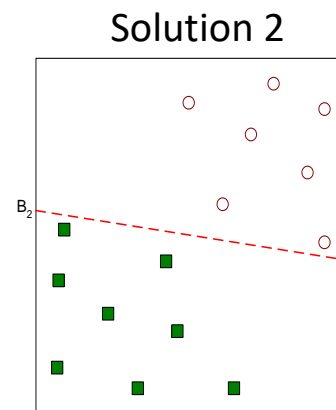
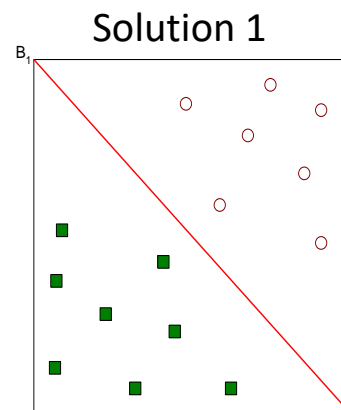
$$z_k = \begin{cases} +1 & \text{if } \mathbf{x}_k \in \omega_1 \\ -1 & \text{if } \mathbf{x}_k \in \omega_2 \end{cases}$$

predicted class label:

$$\hat{z}_k = \begin{cases} +1 & \text{if } g(\mathbf{x}_k) > 0 \\ -1 & \text{if } g(\mathbf{x}_k) < 0 \end{cases}$$

Learning through “empirical risk” minimization

- Conventional **empirical risk** minimization does **not** imply good **generalization** performance.
 - Several different functions $g(x)$ might separate the **training data** set well.

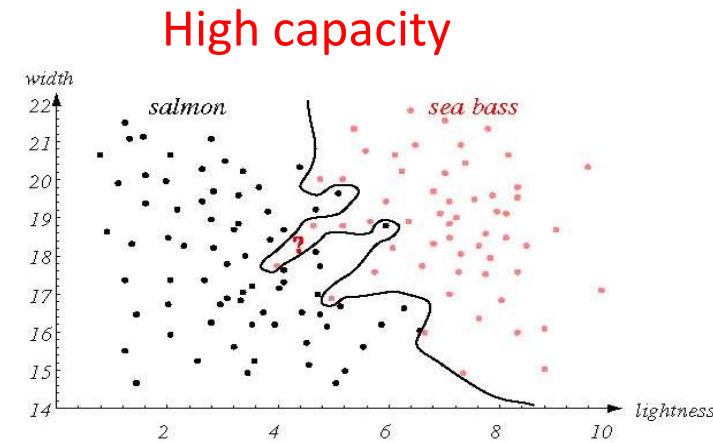
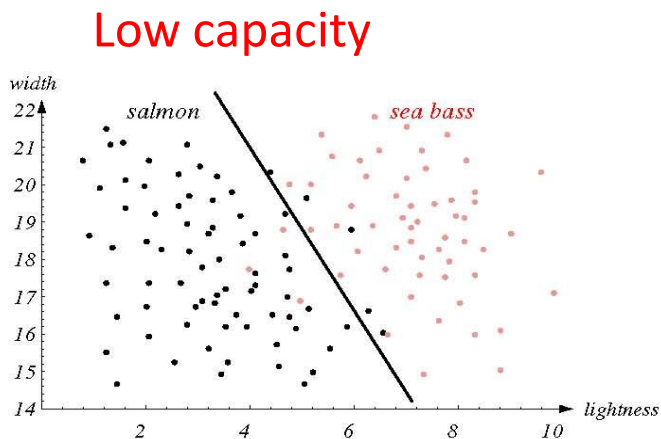


Which solution
generalizes best?

Statistical Learning:

Capacity and VC (Vapnik-Chervonenkis) dimension

- To guarantee good **generalization** performance, the **complexity** (or **capacity** in statistical learning) of the learnt functions must be controlled.
 - Functions with high **capacity** are more complex (i.e., have more degrees of freedom or parameters).



Statistical Learning: Capacity and VC dimension

- In statistical learning, the **Vapnik-Chervonenkis (VC) dimension** is a popular measure of the **capacity** of a classifier.
- The **VC dimension** can be used to predict a **probabilistic** upper bound on the **generalization error** of a classifier.

$$err_{true} \leq err_{train} + \sqrt{\frac{h(\log(2n / h) + 1) - \log(\delta / 4)}{n}}$$

(h: **VC dimension**)

with **probability** $(1-\delta)$;

(n: # of training examples)

(Vapnik, 1995, "*Structural Risk Minimization Principle*")

SVMs

- Primarily a **two-class** classifier but can be extended to **multiple** classes.
- It performs **structural risk minimization** to achieve good generalization performance.
- Training is equivalent to solving a **quadratic programming** problem with **linear constraints**.

(**Not** iterative as in the case of Gradient Descent and Newton's method!)

SVMs

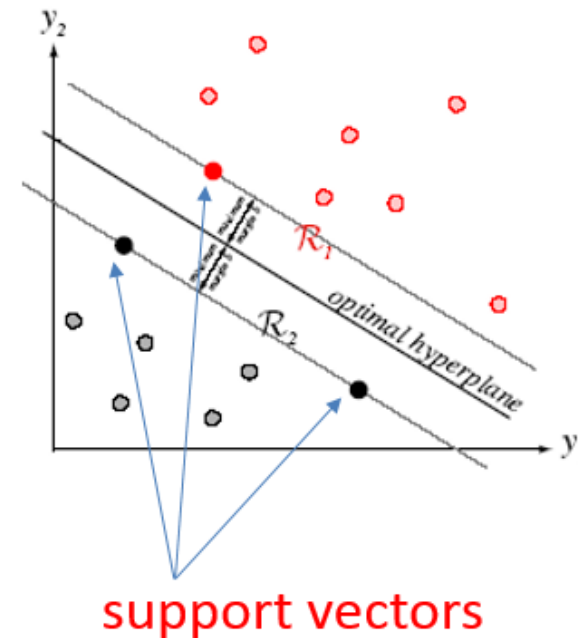
- The original SVM algorithm was first invented by Vladimir N.Vapnik and Alexey Ya. Chervonenkis in 1963
- SVM was developed at AT&T Bell laboratories by V. Vapnik and colleagues (Boser et al., 1992, Guyon et al., 1993, Vapnik et al., 1997)
- SVM is one of the most popular and widely used supervised machine learning algorithms.
- SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees.
- It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.

SVM

- SVM is an algorithm of much promise, and the concepts are relatively simple.
- The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier.
- SVM finds an optimal hyperplane which helps in classifying new data points.

Margin of separation and support vectors

- How is the margin defined?
 - The margin is defined by the distance of the **nearest training samples** from the hyperplane.
 - Intuitively speaking, these are the most difficult samples to classify.
 - We refer to these samples as **support vectors**.

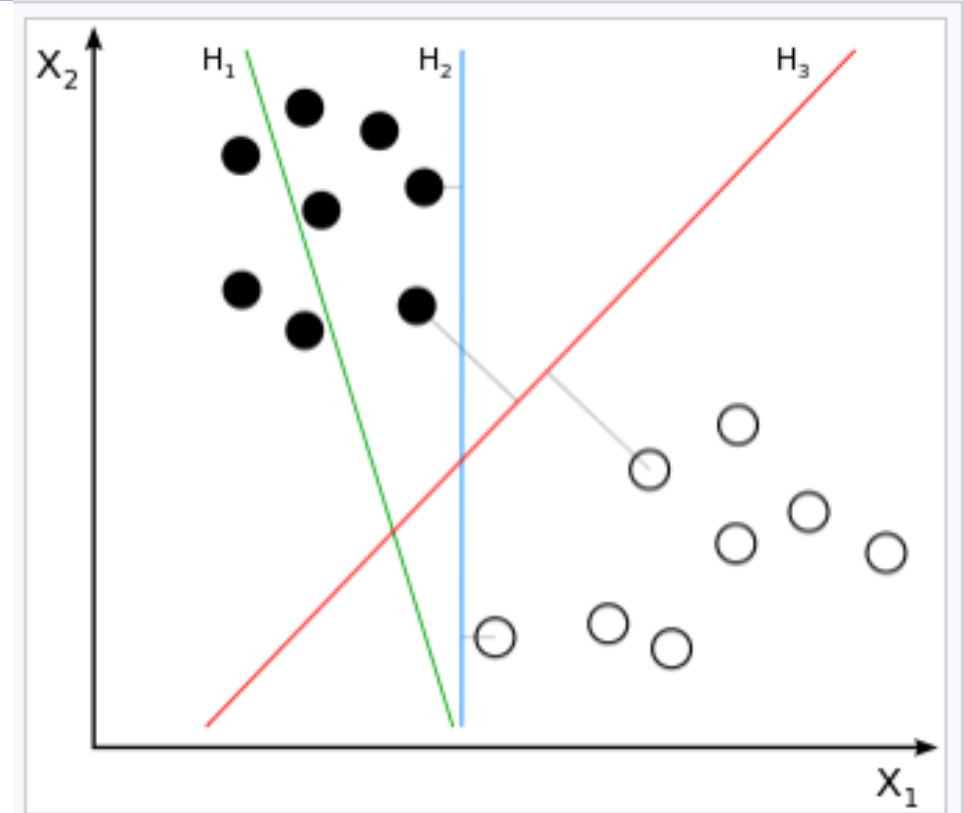



Why SVM?

- Solve the data points that are not linearly separable
- Effective in a higher dimensions
- Suitable for small data set: effective when the number of features is **more than** training examples.

How does SVM work?

- SVM constructs a hyperplane in multidimensional space to separate different classes.
- SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error.
- The core idea of SVM is to find a maximum marginal hyperplane that best divides the dataset into classes.



H_1 does not separate the classes. 
 H_2 does, but only with a small margin.
 H_3 separates them with the maximal margin.

What is a hyperplane?

- In 2 dimensions, a hyperplane is a line:

$$y = b + w_1x_1 + w_2x_2$$

- In n-dimensions, a hyperplane is a n-dimensional space:

$$y = b + w_1x_1 + w_2x_2 + \dots + w_nx_n = b + w^T X$$

- A hyperplane aims to divide a space into two parts

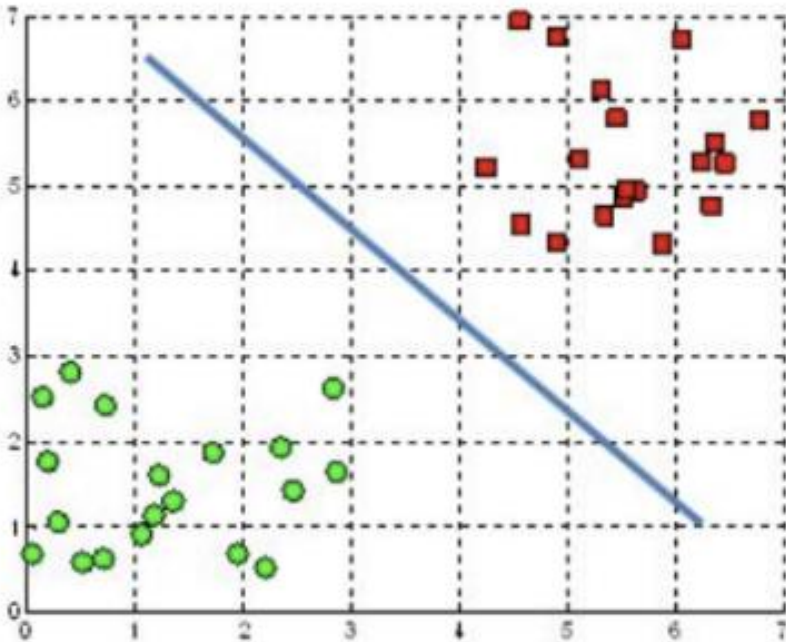
$$b + w^T X > 0 \text{ and}$$

$$b + w^T X < 0$$

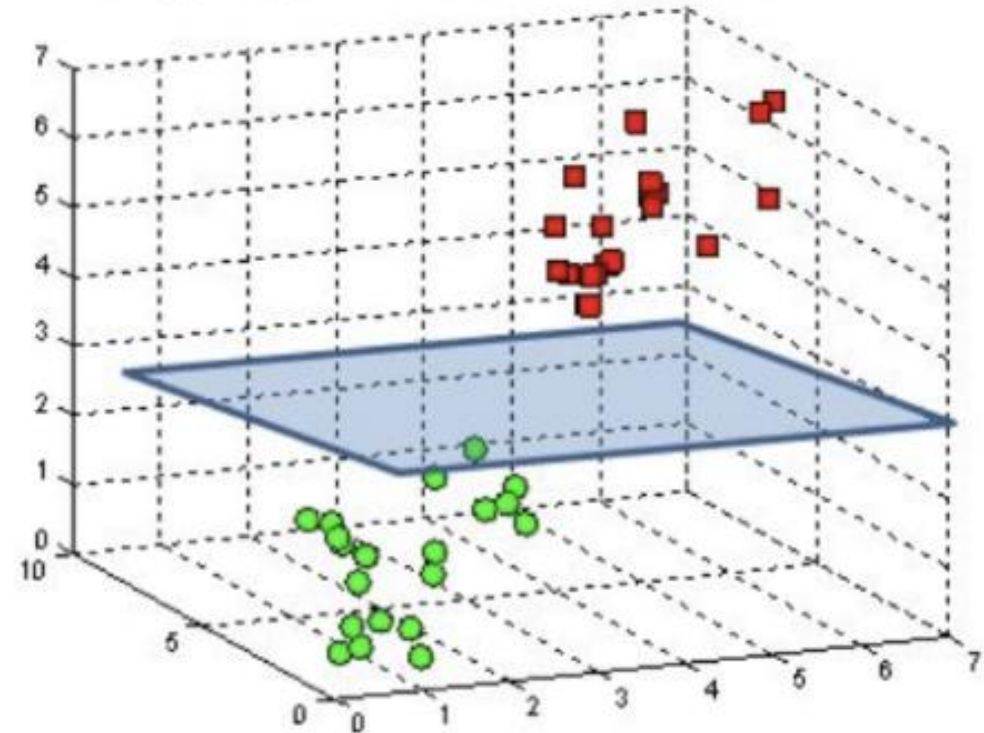
Hyperplanes

A hyperplane is a decision plane which separates between a set of objects having different class memberships.

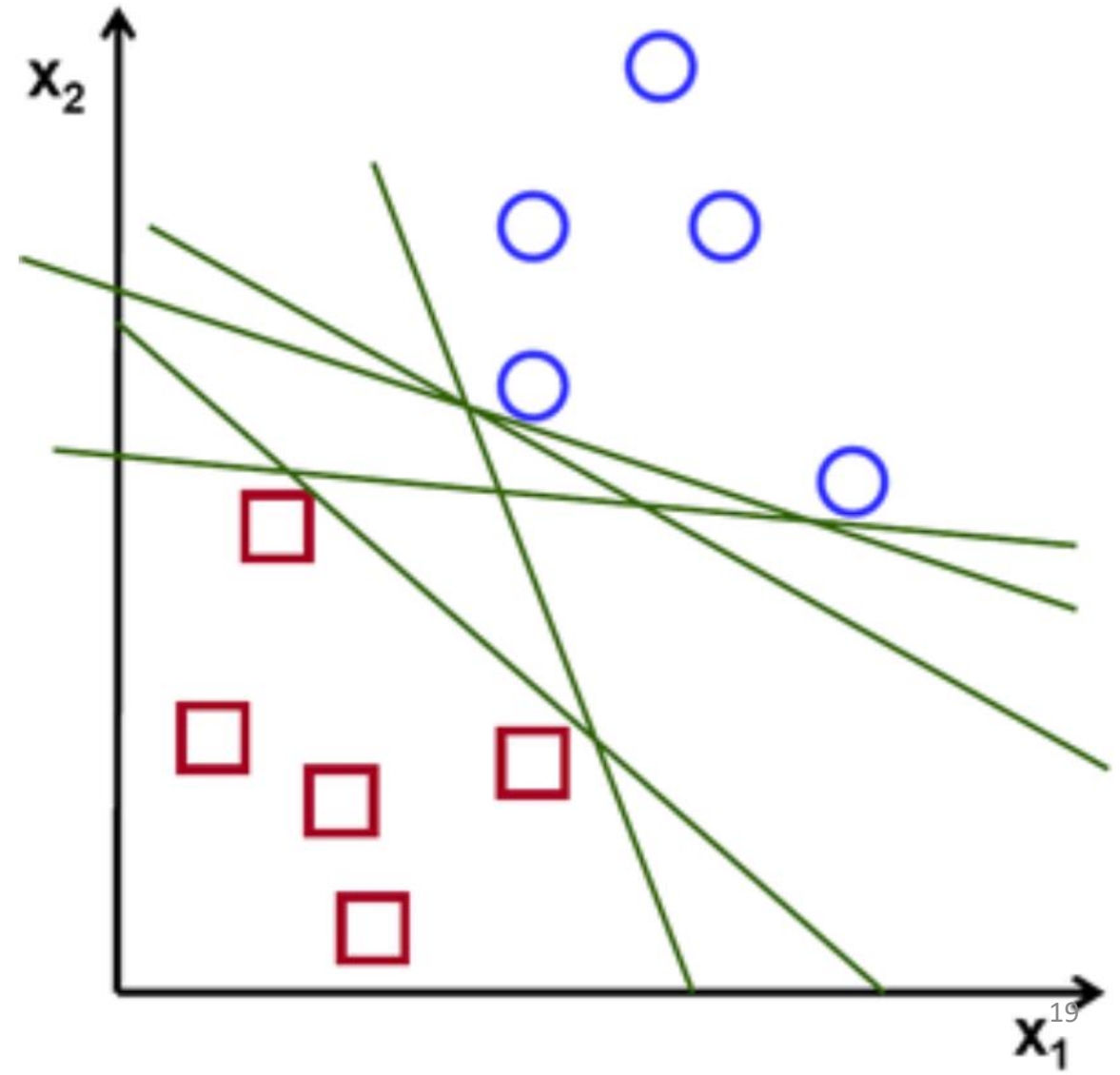
A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane



Possible Hyperplanes



Linear SVM: separable case

(i.e., data **is** linearly separable)

- A linear discriminant is given by:

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

“Decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$ ”

$$z_k = \begin{cases} +1 & \text{if } \mathbf{x}_k \in \omega_1 \\ -1 & \text{if } \mathbf{x}_k \in \omega_2 \end{cases}$$

- Consider the **equivalent** problem:

$$z_k g(\mathbf{x}_k) > 0 \quad \text{or} \quad z_k (\mathbf{w}^t \mathbf{x}_k + w_0) > 0, \quad \text{for } k = 1, 2, \dots, n$$

Linear SVM: separable case

- The solution of the “dual” problem is given by:

$$\mathbf{w} = \sum_{k=1}^n z_k \lambda_k \mathbf{x}_k, \quad w_0 = z_k - \mathbf{w}^t \mathbf{x}_k$$

(use any \mathbf{x}_k to compute w_0)

- The SVM discriminant is given by:

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

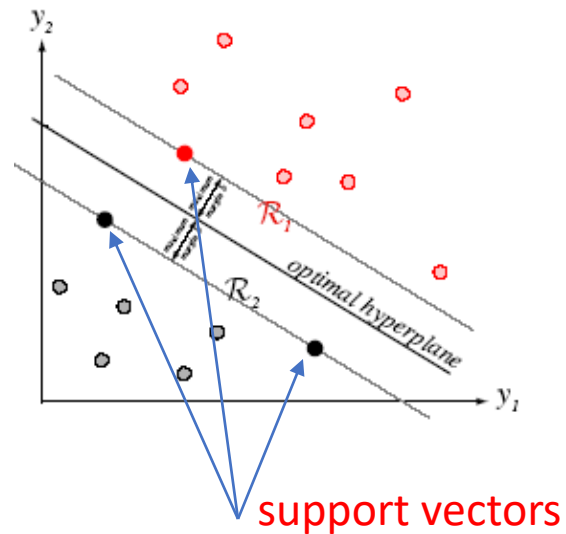


dot product

$$g(\mathbf{x}) = \sum_{k=1}^n z_k \lambda_k (\mathbf{x}_k^t \mathbf{x}) + w_0 = \sum_{k=1}^n z_k \lambda_k (\mathbf{x} \cdot \mathbf{x}_k) + w_0$$

Linear SVM: separable case

- It turns out that if \mathbf{x}_k is **not** a support vector, then $\lambda_k=0$.
- The SVM discriminant function depends **only** on the support vectors!

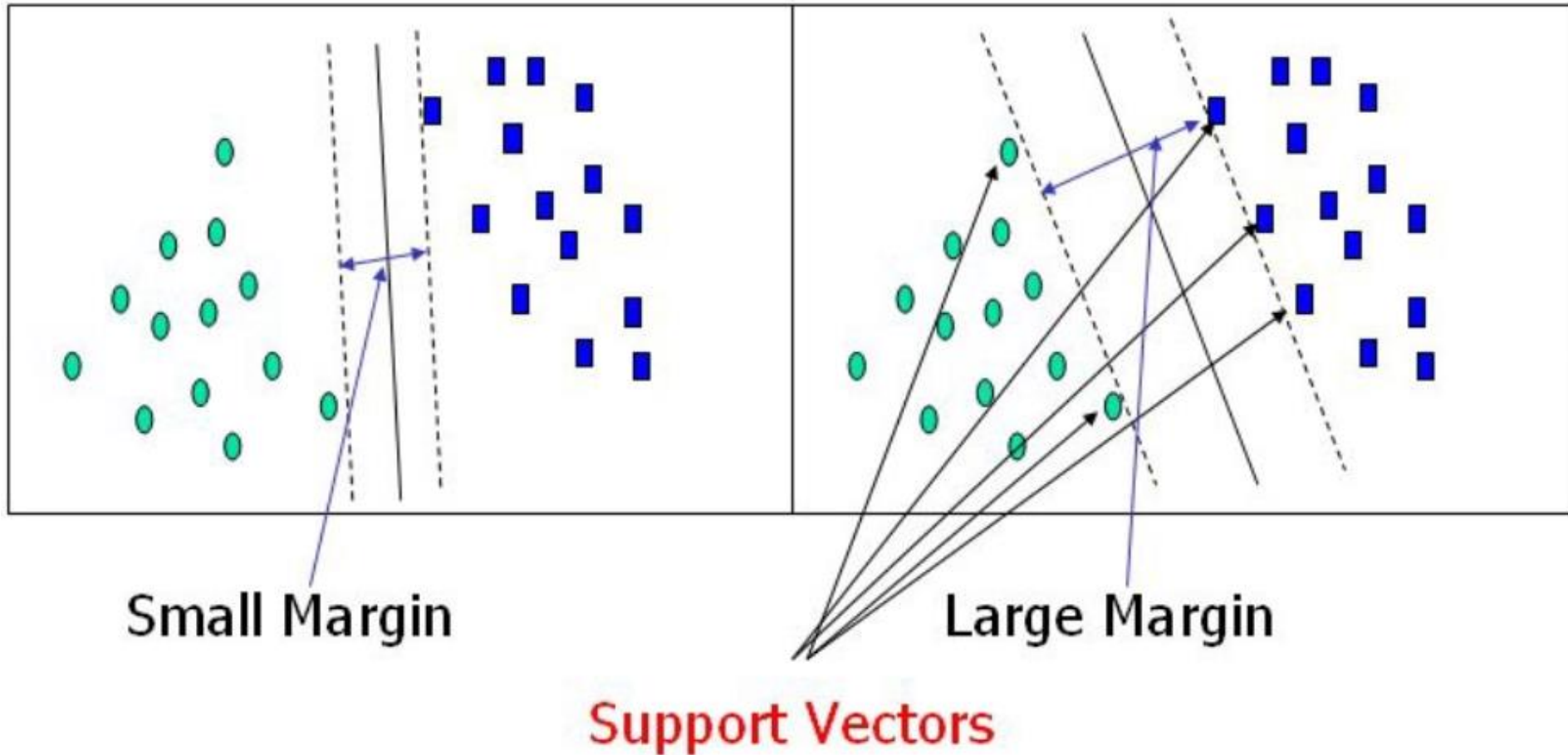


$$g(\mathbf{x}) = \sum_{k=1}^n z_k \lambda_k (\mathbf{x} \cdot \mathbf{x}_k) + w_0$$

Support Vectors

- Support vectors are the data points, which are closest to the hyperplane.
- These points will define the separating line better by calculating margins.
- These points are more relevant to the construction of the classifier.
- Using these support vectors, we maximize the margin of the classifier.
- Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

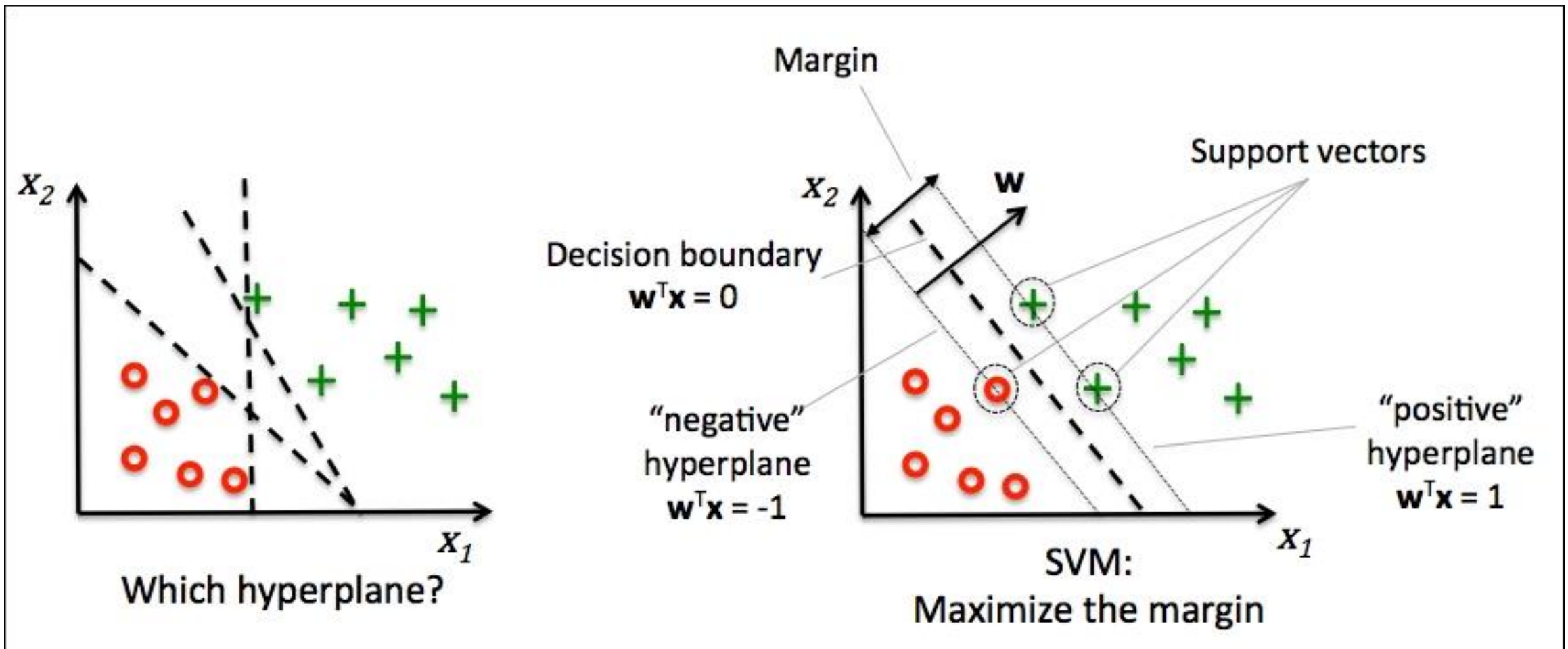
Support Vectors



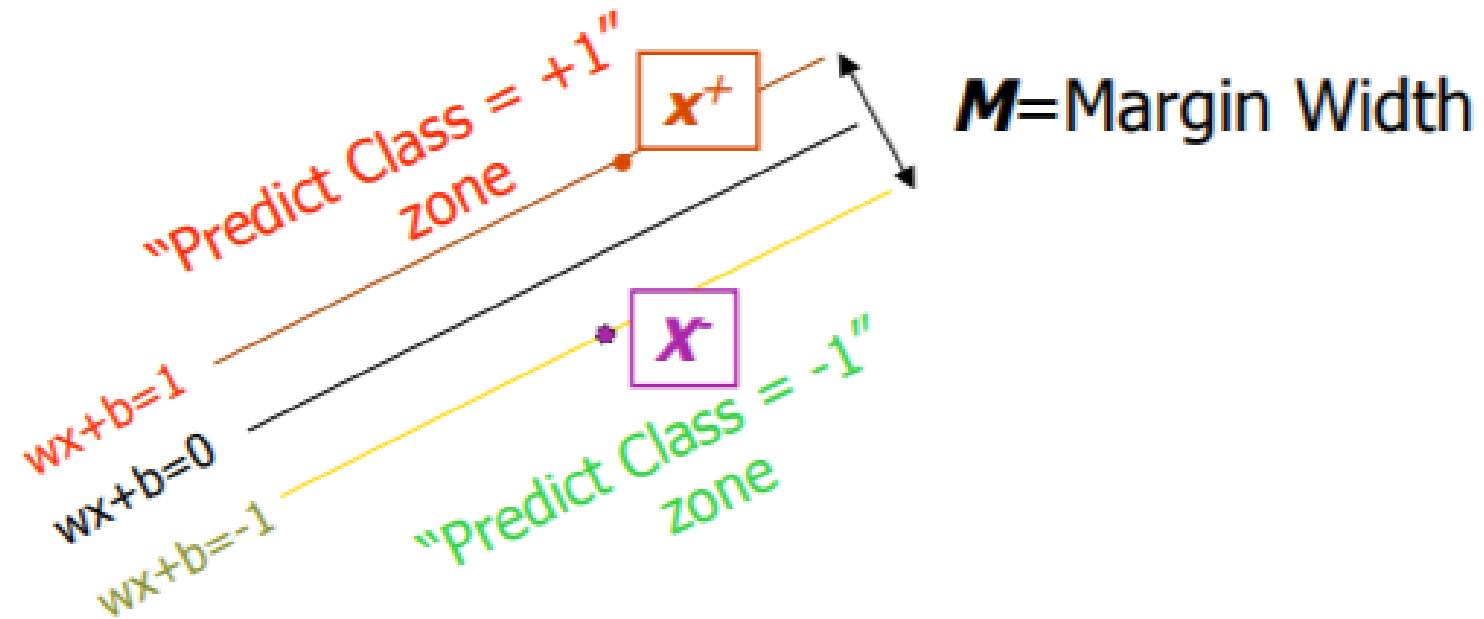
Margin

- A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest points.
- If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.
- The objective of SVM is to find a plane that has the maximum margin, ie a maximum distance between data points of both classes.
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Maximizing the margin



Finding the Boundary



What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^- + b = -1$
- $w \cdot (x^+ - x^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

How does a SVM work?

Given a guess for w and b

- Compute whether all data points in correct half-planes
- Compute the width of the margin

Search the space of w 's and b 's to find the widest margin that matches all data points

Correctly classify all training data

$$\begin{array}{ll} wx_i + b \geq 1 & \text{if } y_i = +1 \\ wx_i + b \leq 1 & \text{if } y_i = -1 \\ y_i(wx_i + b) \geq 1 & \text{for all } i \end{array}$$



$$\text{Maximize: } M = \frac{2}{|w|} \quad \longrightarrow \quad \text{Minimize: } \frac{1}{2} w^t w$$

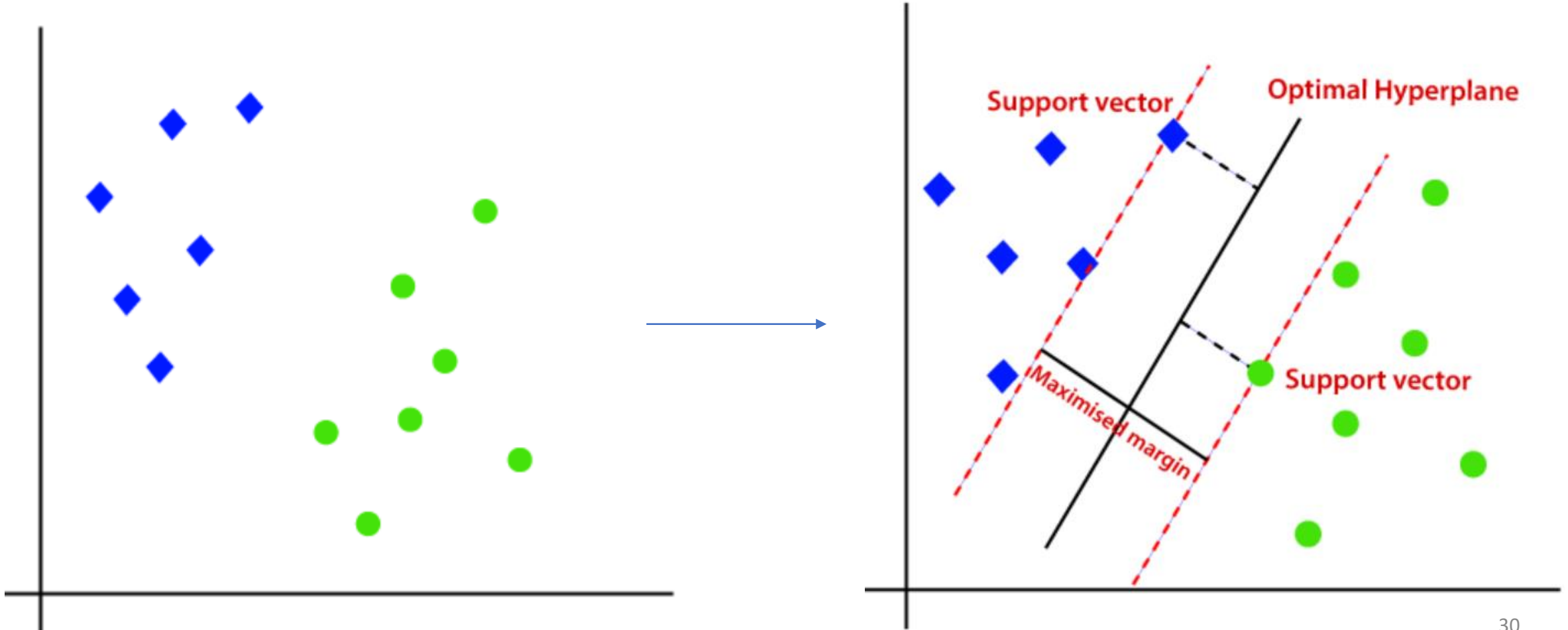
Solving the Optimization Problem

Find w and b such that

$$\frac{1}{2}w^T w \text{ is minimized}$$

$$\text{for all } (x_i, y_i): y_i(w^T x_i + b) \geq 1$$

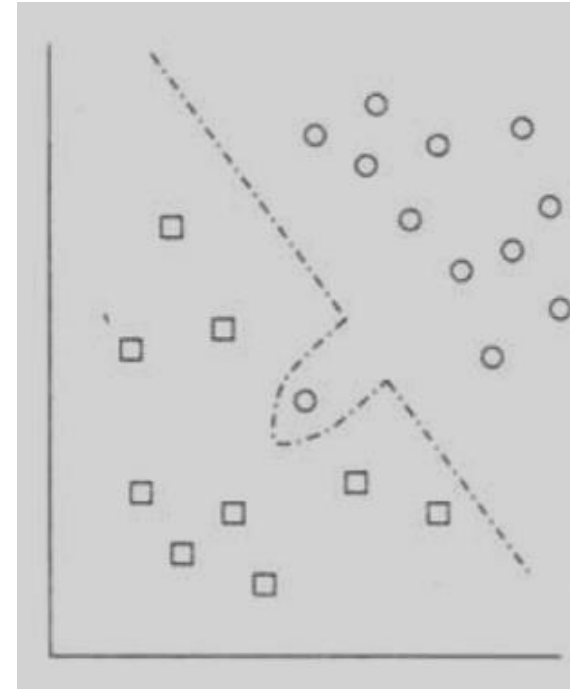
Linear SVM



Linear SVM: non-separable case

(i.e., data **is not** linearly separable)

- Most effective when the data is “almost” linearly separable.
- Aims to prevent **outliers** from affecting the optimal hyperplane.
- Also referred to as **soft margin** classifier.



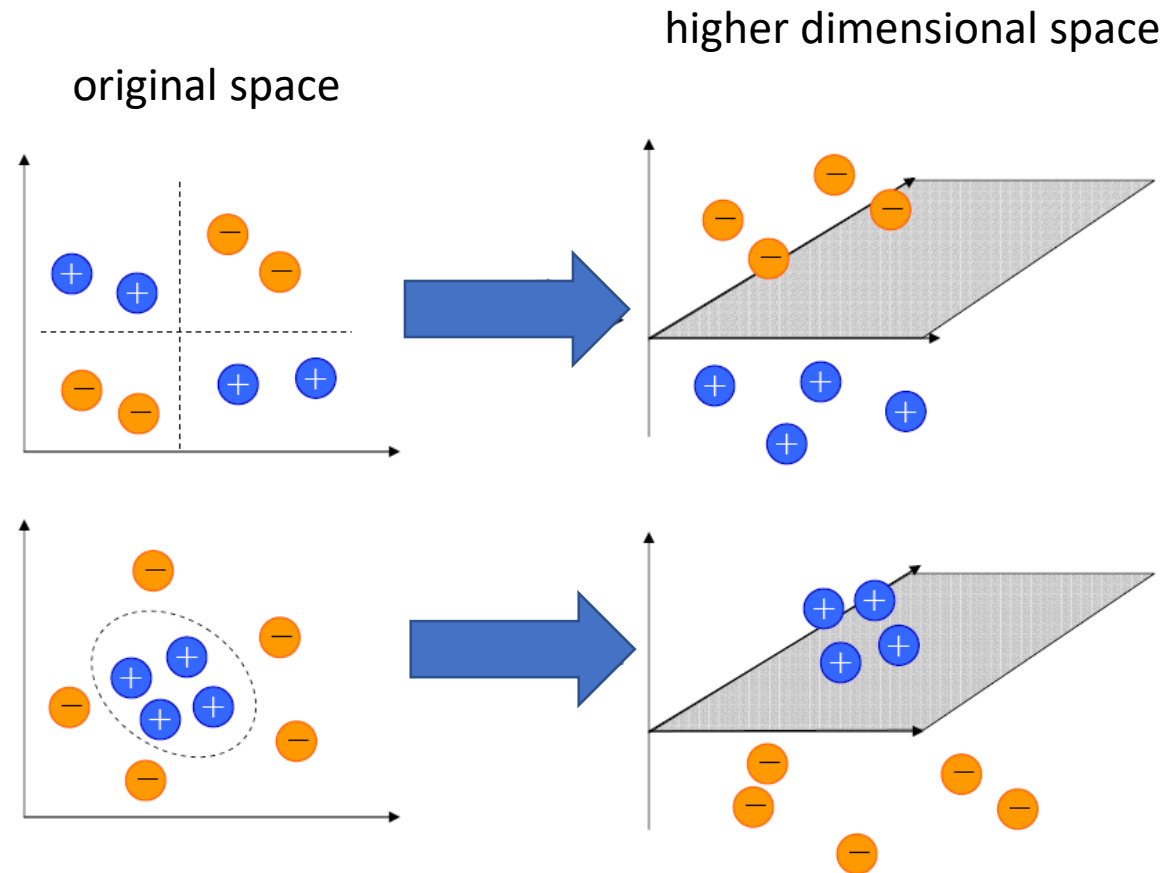
Nonlinear SVM

- Extensions to the non-linear case relate to the idea of mapping the data to a space of much higher dimensionality h :

$$\mathbf{x}_k \rightarrow \Phi(\mathbf{x}_k) = \begin{bmatrix} \varphi_1(\mathbf{x}_k) \\ \varphi_2(\mathbf{x}_k) \\ \dots \\ \varphi_h(\mathbf{x}_k) \end{bmatrix}$$

- This is likely to cast the data **linearly separable** in h -dimensional space.

Examples

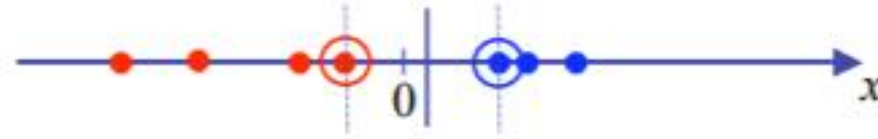


Non-linear SVM

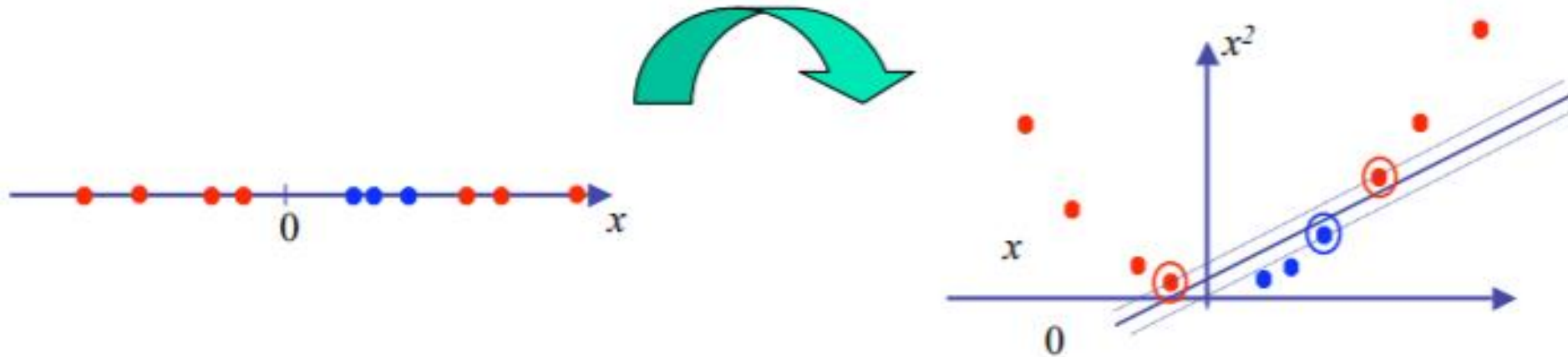
- SVM locates a separating hyperplane in the feature space and classifies points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function (called a *kernel trick*)
- The kernel function plays the role of the dot product in the feature space

Non-linear SVM

Linearly separable data:



Map data to a higher-dimensional space:



Nonlinear SVM

linear SVM: $g(\mathbf{x}) = \sum_{k=1}^n z_k \lambda_k (\mathbf{x} \cdot \mathbf{x}_k) + w_0$



$$\mathbf{x}_k \rightarrow \Phi(\mathbf{x}_k) = \begin{bmatrix} \varphi_1(\mathbf{x}_k) \\ \varphi_2(\mathbf{x}_k) \\ \dots \\ \varphi_h(\mathbf{x}_k) \end{bmatrix}$$

dot product

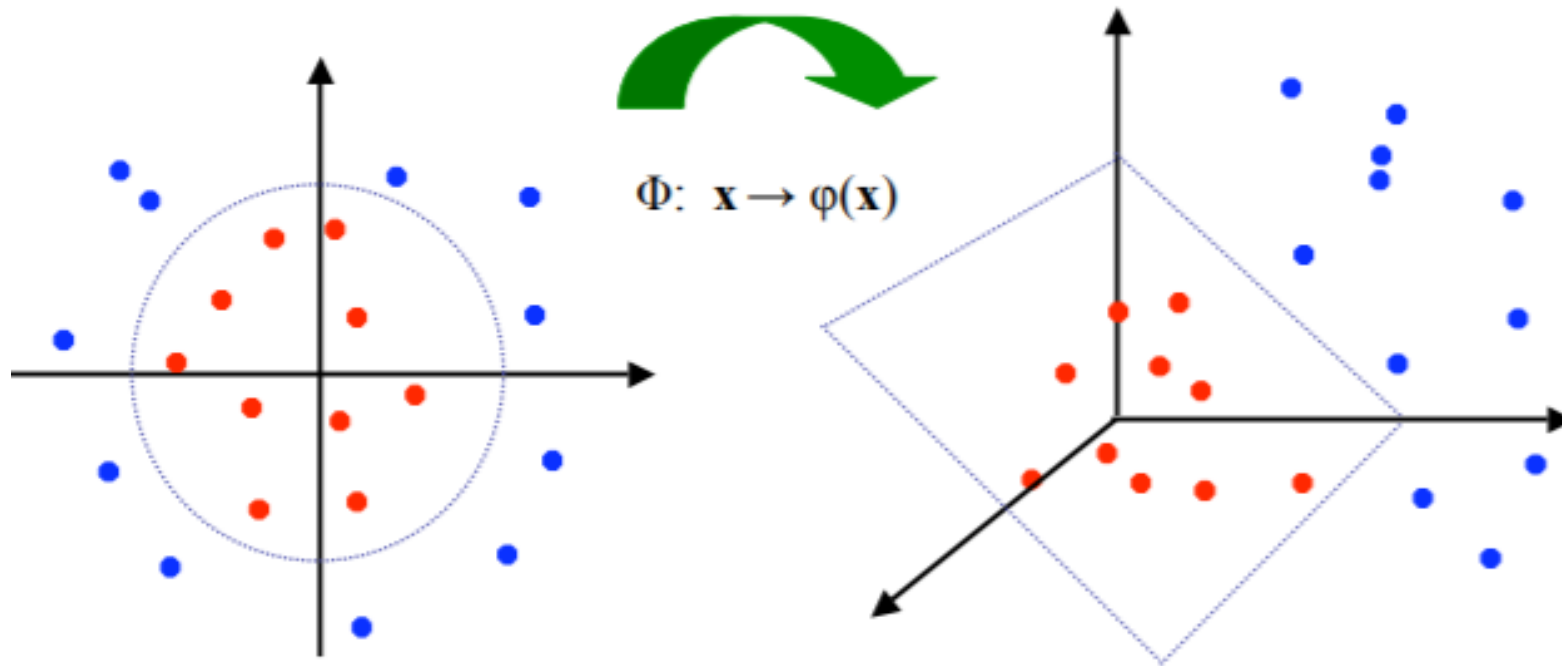
non-linear SVM: $g(\mathbf{x}) = \sum_{k=1}^n z_k \lambda_k (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_k)) + w_0$

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

linear classifier
in h -space

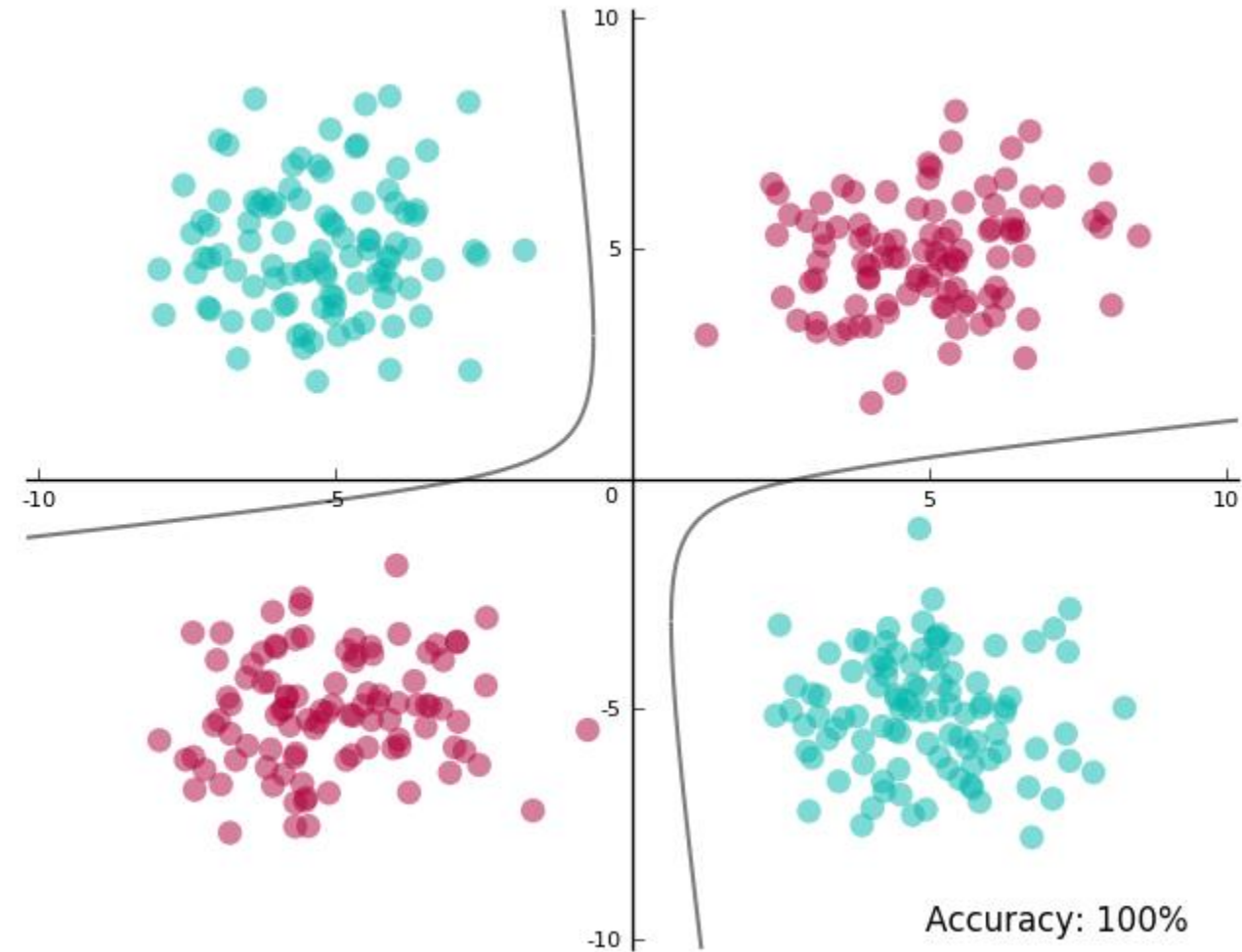
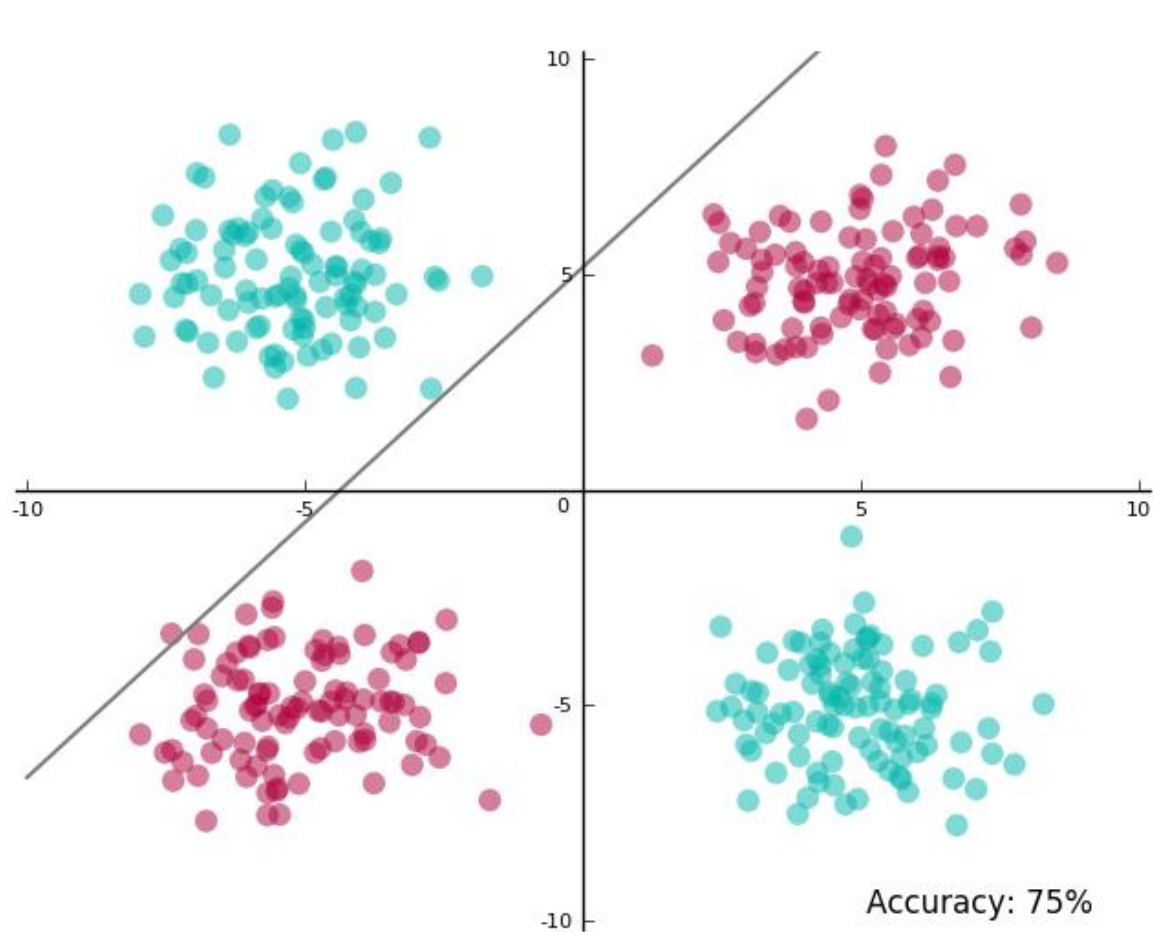
$$\mathbf{w} = \sum_{k=1}^n z_k \lambda_k \Phi(\mathbf{x}_k), \quad w_0 = z_k - \mathbf{w}^t \Phi(\mathbf{x}_k)$$

Non-linear SVM



The original input space can always be mapped to some higher-dimensional feature space where the training set is separable


Non-linear SVM




Kernel trick

- Compute dot products using a **kernel** function in the original space:

$$g(\mathbf{x}) = \sum_{k=1}^n z_k \lambda_k (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_k)) + w_0$$



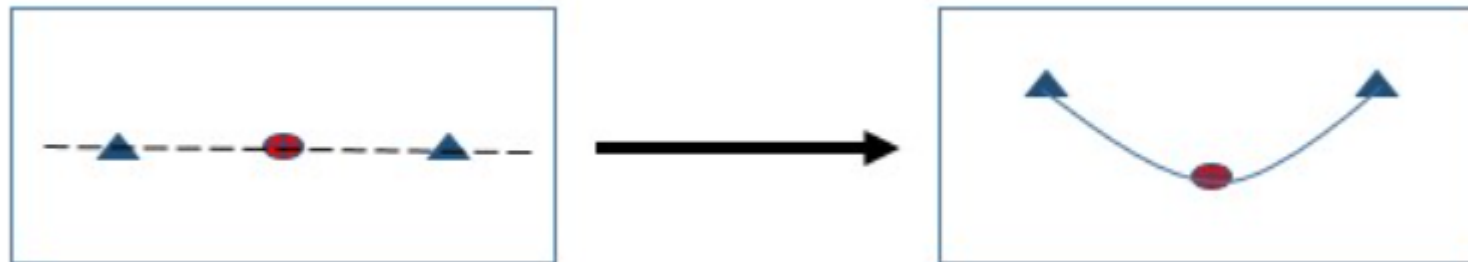
$$g(\mathbf{x}) = \sum_{k=1}^n z_k \lambda_k K(\mathbf{x}, \mathbf{x}_k) + w_0$$

 **kernel function**

- Advantages of kernel trick
 - No need to know $\Phi()$!
 - Computations remain feasible even if **h** is very high.

SVM Kernels

- A kernel transforms an input data space into the required form. SVM uses a technique called the *kernel trick*.
- The kernel takes a low-dimensional input space and transforms it into a higher dimensional space.
- In other words, you can say that it converts non-separable problem to separable problems by adding more dimensions to it.
- It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.



Linear Kernel

A linear kernel can be used as normal dot product of any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

$$k(x, y) = x^T y + c$$


Polynomial kernel


The polynomial kernel is popular in image processing. The polynomial kernel is defined as


$$k(x, y) = (\alpha x^T y + c)^d$$

Adjustable parameters are the slope **alpha**, the constant term **c** and the polynomial degree **d**.

Common Kernel functions

Polynomial: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$  parameter

Sigmoidal: $K(\mathbf{x}, \mathbf{y}) = \tanh(v(\mathbf{x} \cdot \mathbf{y}) + c)$  parameters

Gaussian: $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$  parameter

Radial basis function kernel (RBF)

Radial Basis Function (Gaussian) Kernel is another popular Kernel method used in SVM models for more. RBF kernel is a function whose value depends on the distance from the origin or from some point. Gaussian Kernel is of the following format with a parameter σ :

$$k(x, y) = \exp \left(-\frac{\|x - y\|^2}{2\sigma^2} \right)$$

Computational Complexity

Assume n number of instances

Complexity: $O(n^3)$

Applications of SVM

- Face recognition
- Facial Expression Classification
- Text classification
- Image classification
- Handwriting recognition
- Speech recognition
- Cancer Diagnosis detection from images

Advantages of SVM

- SVM Classifiers offer good accuracy and perform faster prediction compared to Naïve Bayes algorithm.
- They also use less memory because they use a subset of training points in the decision phase.
- It is really effective in the higher dimension.
- Effective when the number of features are more than training examples.
- Best algorithm when classes are separable
- The hyperplane is affected by only the support vectors thus outliers have less impact.
- SVM is suited for extreme case binary classification.

Disadvantages

- SVM is not suitable for large datasets because of its high training time, and it also takes more time in training compared to Naïve Bayes.
- It works poorly with overlapping classes and is also sensitive to the type of kernel used.
- Selecting, appropriately hyperparameters of the SVM that will allow for sufficient generalization performance.
- Selecting the appropriate kernel function can be tricky.

Concluding remarks..

- SVM is based on exact optimization, not on approximate methods (i.e., global optimization method, **no local optima**)
- Appears to avoid **overfitting** in high dimensional spaces and **generalize** well using a small training set.
- Performance depends on the choice of the **kernel** and its parameters.
- Its **complexity** depends on the number of support vectors, **not** on the number of training data.