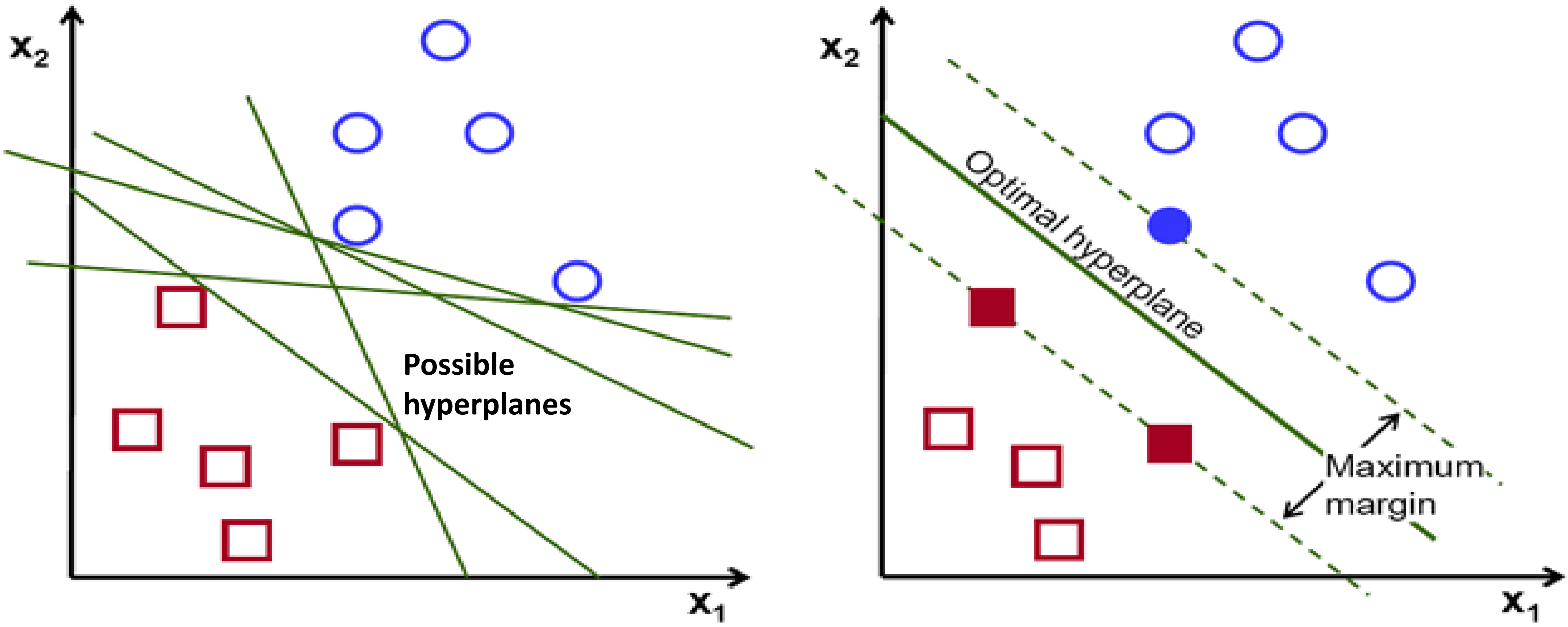
SUPPORT VECTOR MACHINE

The idea of support vectors and its importance

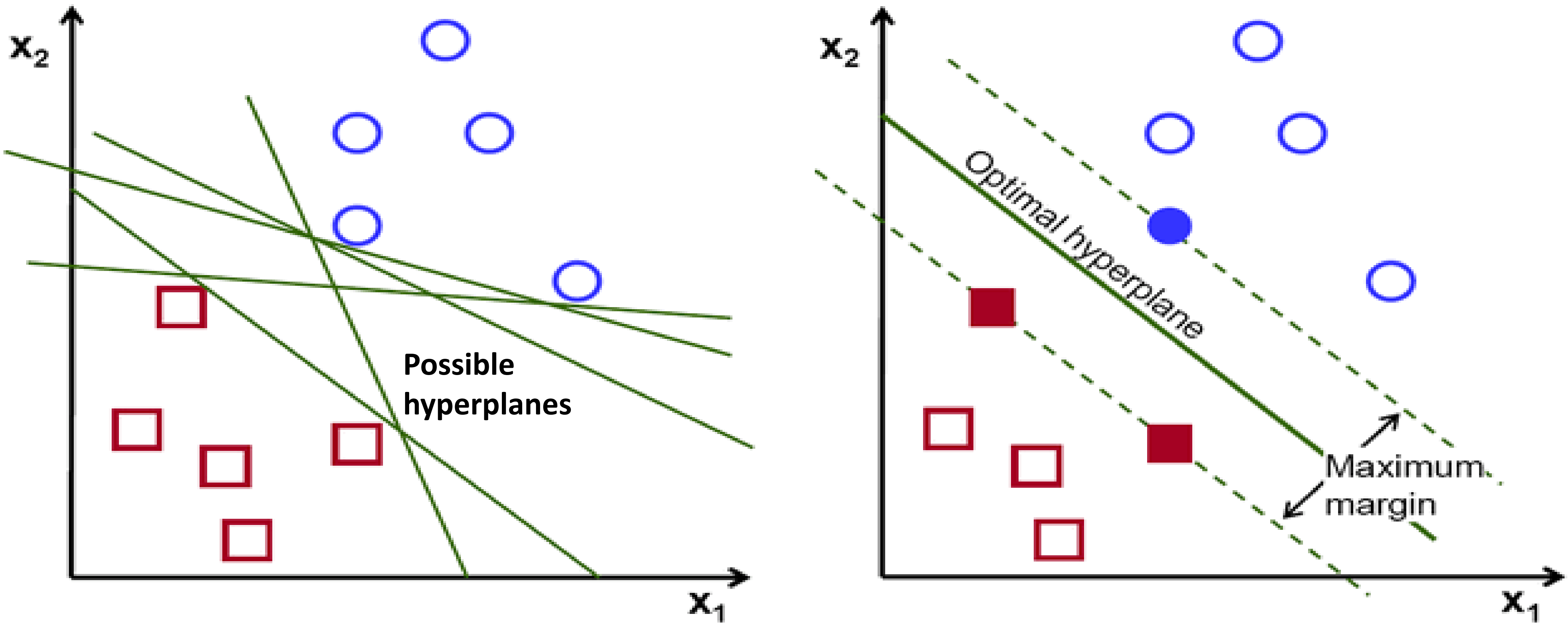
Introduction

- The support vector machine is currently considered to be the best off-the-shelf learning algorithm and has been applied successfully in various domains.
- Support vector machines were originally designed for binary classification.
- Then, it is extended to solve multi-class and regression problems.
- But, it is widely used in classification objectives.
- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

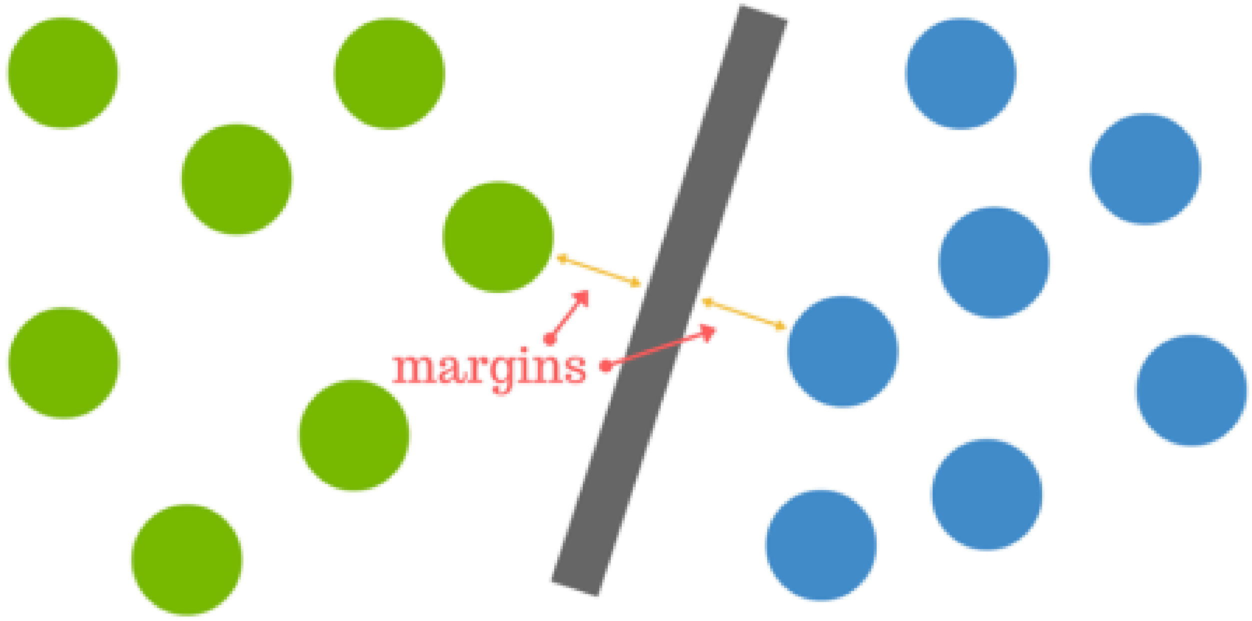
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes.



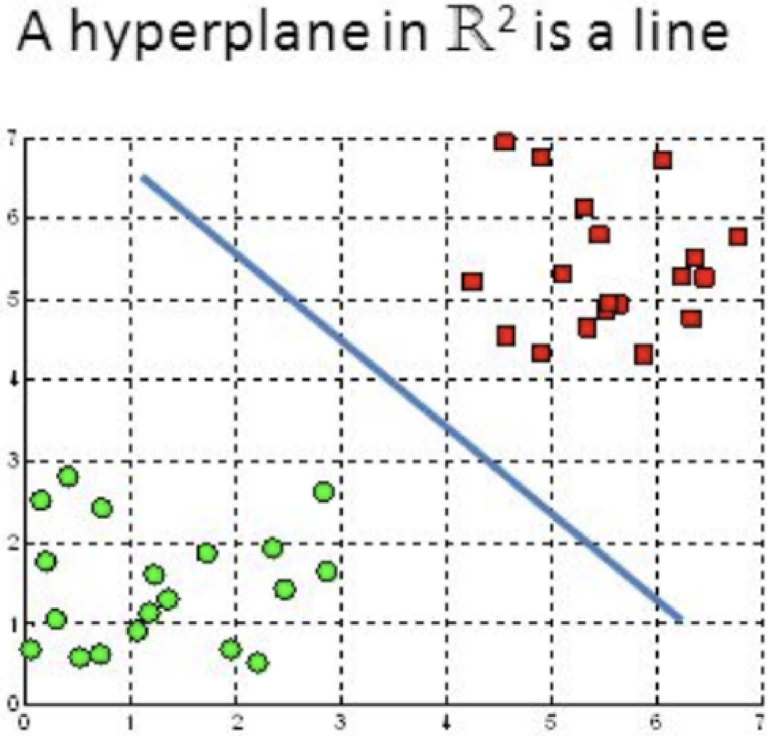
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

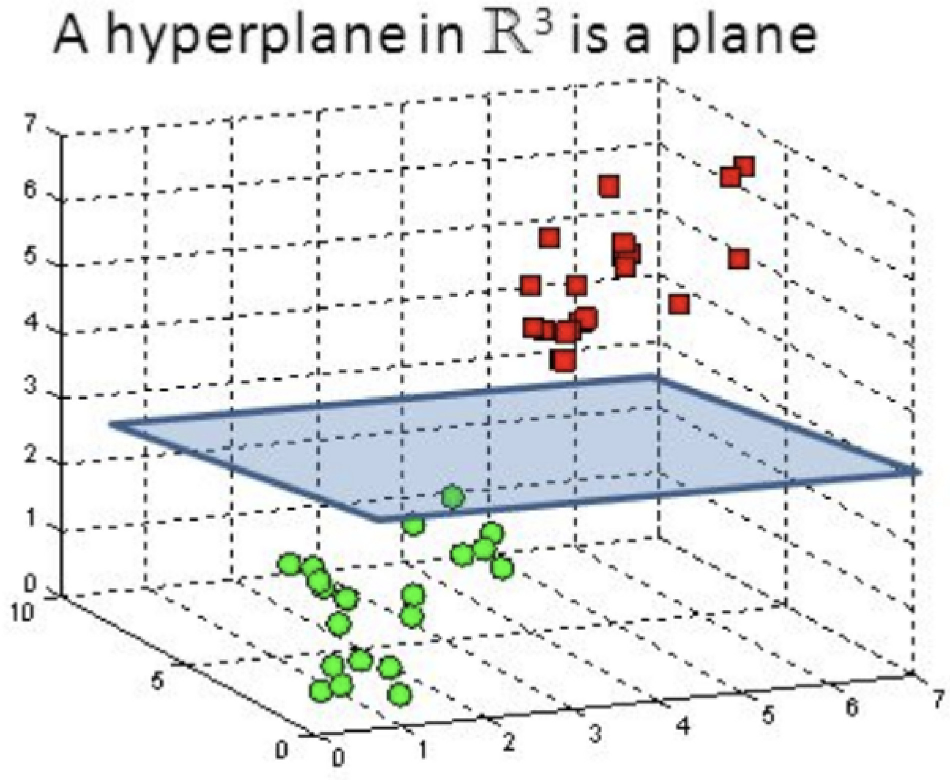


- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.
- The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.

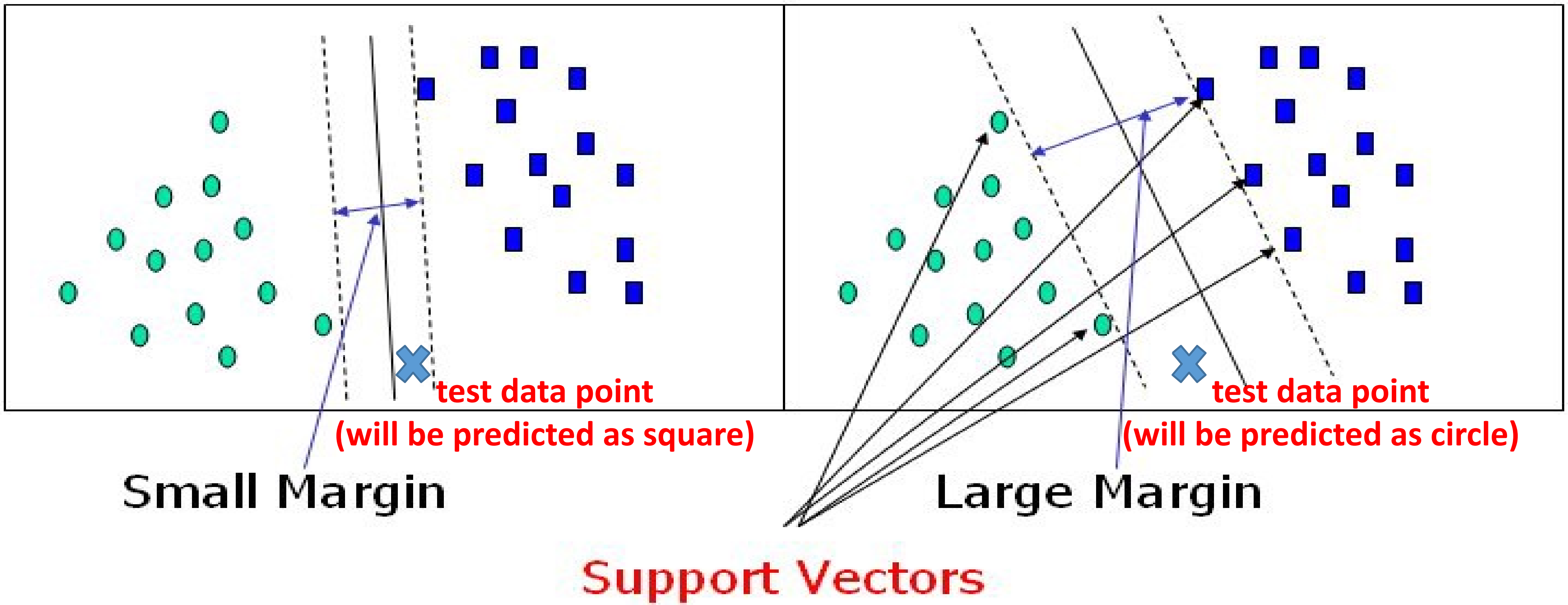


- Hyperplanes are decision boundaries that help classify the data points.
- Data points falling on either side of the hyperplane can be attributed to different classes.
- Also, the dimension of the hyperplane depends upon the number of features.
- It becomes difficult to imagine when the number of features exceeds 3.

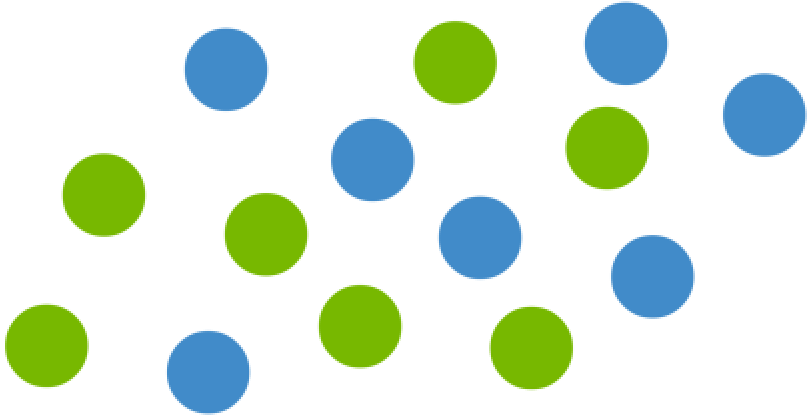




- Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.
- Using these support vectors, we maximize the margin of the classifier.
- Deleting the support vectors will change the position of the hyperplane.
- These are the points that help us build our SVM (that works for a new/test data)

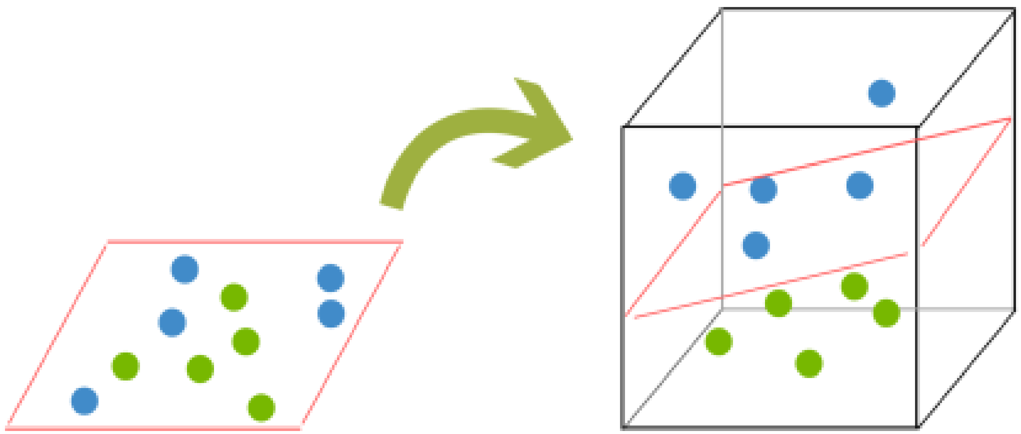


- But what happens when there is no clear hyperplane?
- A dataset will often look more like the jumbled balls below which represent a linearly non separable dataset.



- In order to classify a dataset like the one above it's necessary to move away from a 2d view of the data to a 3d view.

- Explaining this is easiest with another simplified example.
- Imagine that our two sets of colored balls above are sitting on a sheet and this sheet is lifted suddenly, launching the balls into the air.
- While the balls are up in the air, you use the sheet to separate them.
- This 'lifting' of the balls represents the mapping of data into a higher dimension.



- This is known as kernelling.

Pros & Cons of Support Vector Machines

Pros

- Accuracy
- Works well on smaller cleaner datasets
- It can be more efficient because it uses a subset of training points

Cons

- Isn’t suited to larger datasets as the training time with SVMs can be high
- Less effective on noisier datasets with overlapping classes

Applications

- SVM is used for text classification tasks such as category assignment, detecting spam and sentiment analysis.
- It is also commonly used for image recognition challenges, performing particularly well in aspect-based recognition and color-based classification.
- SVM also plays a vital role in many areas of handwritten digit recognition, such as postal automation services.

Derivation of Support Vector Equation

Comparison with logistic regression

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$

$x_0 = 1, y \in \{0, 1\}$

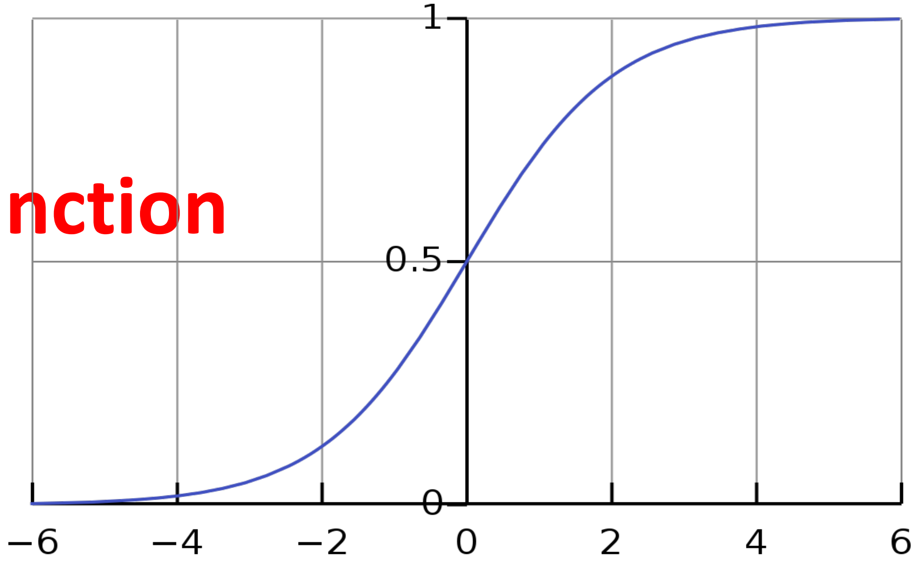
$$\log(\text{odds}) = \log \frac{P(\text{Class } 1|\mathbf{x})}{1 - P(\text{Class } 1|\mathbf{x})} = w_0 + w_1 x_1 + \cdots + w_n x_n$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ?

Max Likelihood Estimation (**already discussed**)

Sigmoid function



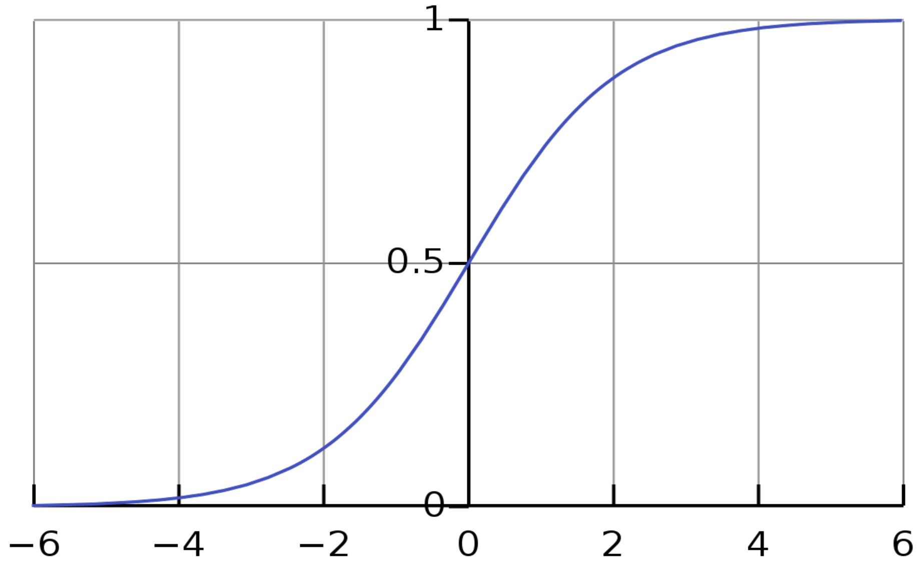
Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

Comparison with logistic regression

- In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is less than -1, we identify it with another class.
- Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values ([-1,1]) which acts as margin.



Sigmoid function

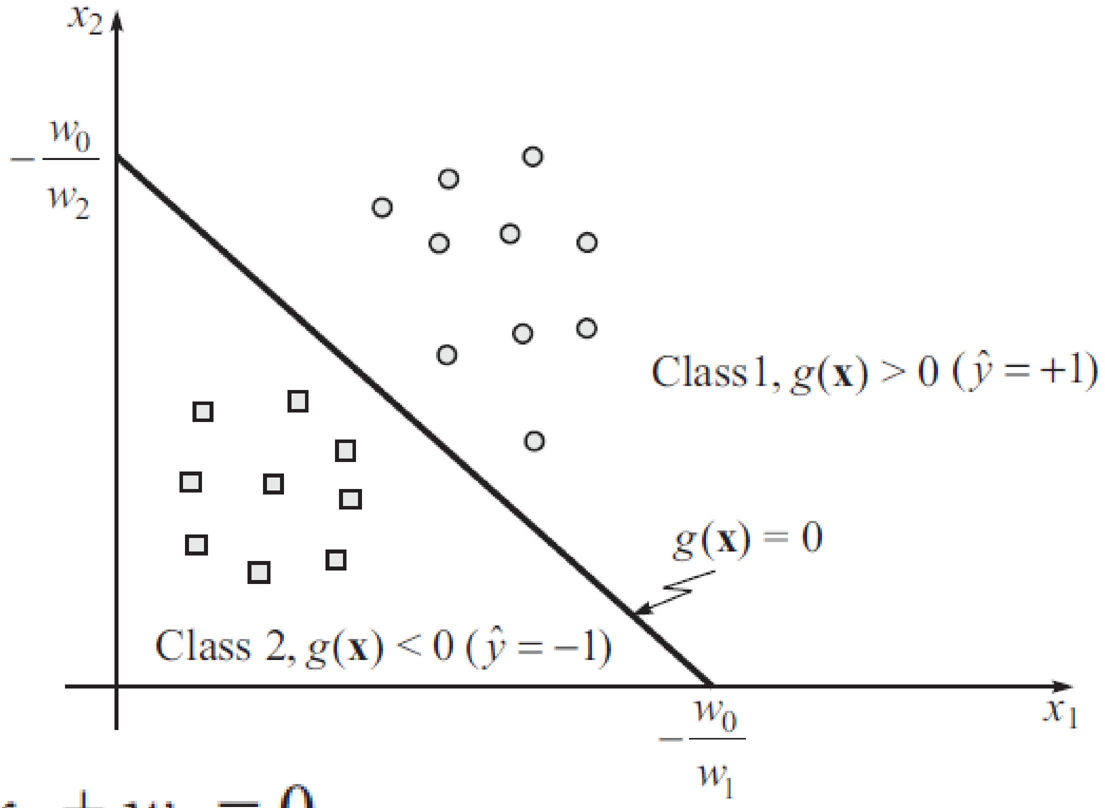
Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

Comparison with logistic regression

- In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is less than -1, we identify it with another class.
- Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values ([-1,1]) which acts as margin.



$$g(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_0 = 0$$

- $g(x)$ is a linear discriminant function that divides (categorizes) $\Re^2$ into two decision regions.

- The generalization of the linear discriminant function for an n-dimensional feature space in $\Re^n$ is straight forward:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$$

$\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$ is the feature vector

$\mathbf{w} = [w_1\ w_2\ \ldots\ w_n]^T$ is a *weight vector*

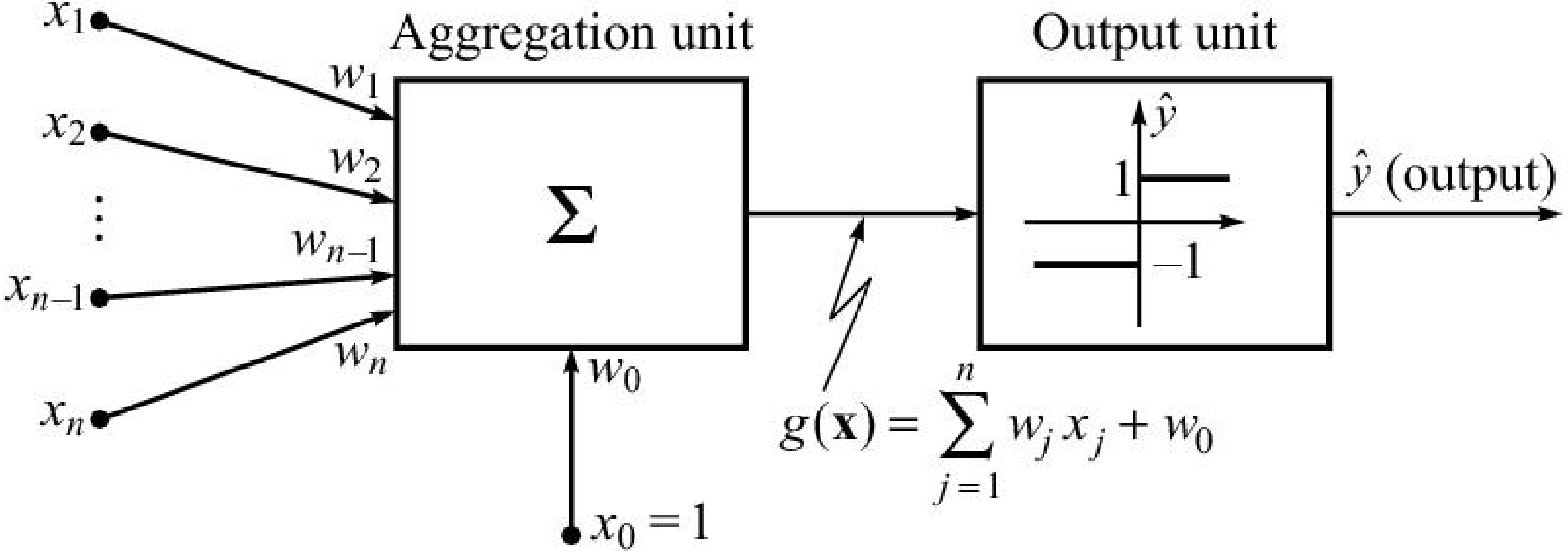
$w_0 =$ *bias* parameter

- The discriminant function is now a linear n-dimensional surface, called a hyperplane; symbolized as $\mathcal{H}$
- A two-category classifier implements the following decision rule:

Decide Class 1 if $g(x) > 0$ and Class 2 if $g(x) < 0$

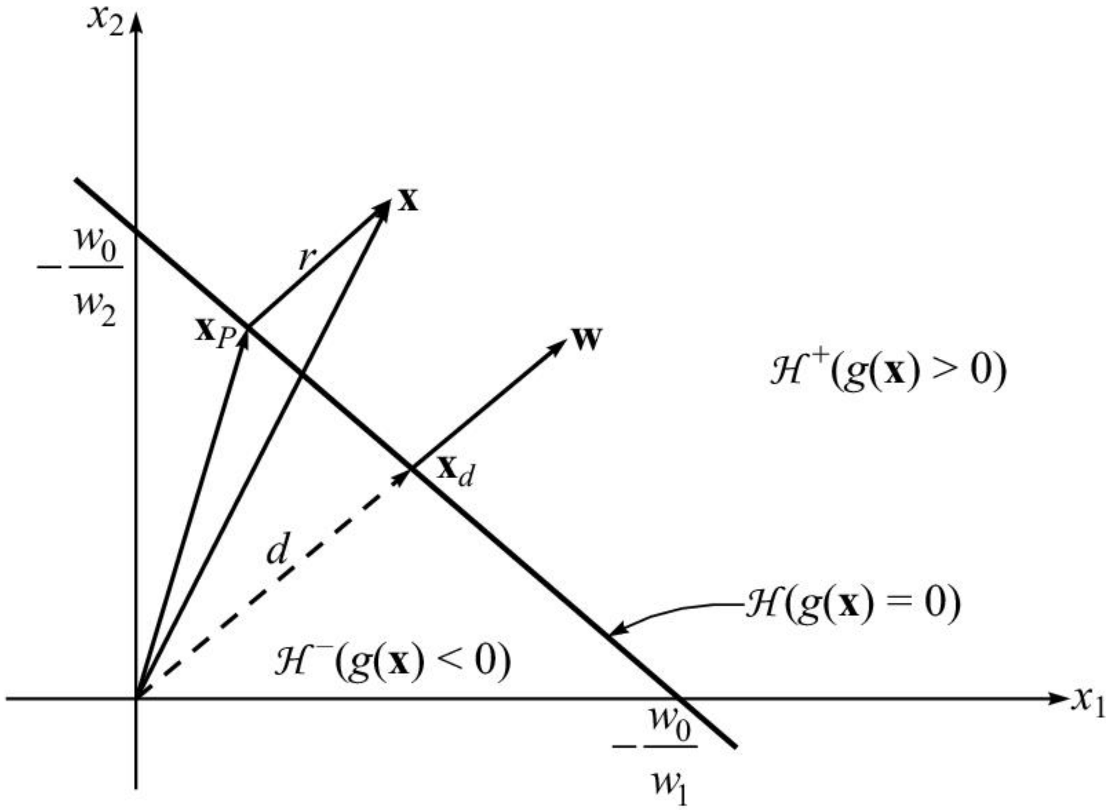
- Thus, x is assigned to Class 1 if the inner product $\mathrm{w}^T\mathrm{x}$ exceeds the threshold (bias) $-w_0$, and to Class 2 otherwise.

- Figure shows the architecture of a typical implementation of the linear classifier.
- It consists of two computational units: an aggregation unit and an output unit.



A simple linear classifier

- Geometry for n = 2 with $w_1 > 0$, $w_2 > 0$ and $w_0 < 0$ is shown in Figure below.
- The origin is on the negative side of $\mathscr{H}$ if $w_0 < 0$, and if $w_0 > 0$, the origin is on the positive side of $\mathscr{H}$.
- If $w_0 = 0$, the hyperplane passes through the origin.



Linear decision boundary between two classes

Location of any point $\mathbf{x}$ may be considered relative to $\mathscr{H}$.

Defining $\boldsymbol{x}_p$ as the normal projection of $\mathbf{x}$ onto $\mathscr{H}$,

$$\mathbf{x} = \mathbf{x}_P + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

where $\|\mathbf{w}\|$ is the Euclidean norm of $\mathbf{w}$ and $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ is a unit vector.

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \mathbf{w}^T\left(\mathbf{x}_P + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$$

$$= \mathbf{w}^T\mathbf{x}_P + w_0 + \frac{\mathbf{w}^T r\,\mathbf{w}}{\|\mathbf{w}\|}$$

$$= r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} = r\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} = r\|\mathbf{w}\|$$

Algebraic measure of the distance from x to the hyperplane

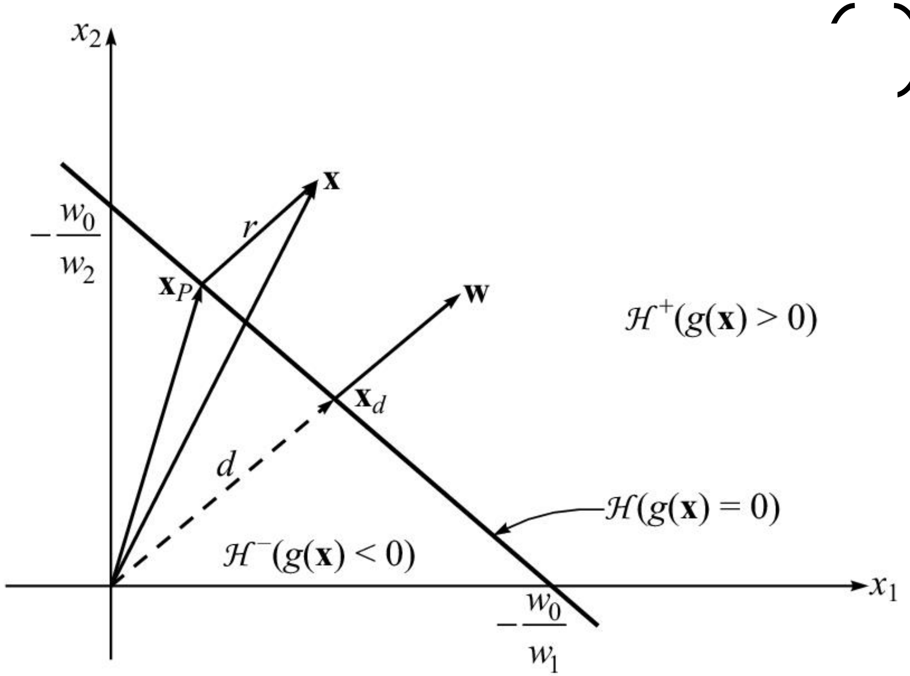
$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

As

$$g(\mathbf{x}) = r.\|\mathbf{w}\|$$

$|g(\mathbf{x})|$ is a measure of the Euclidean distance of the point $\mathbf{x}$ from the decision hyperplane $\mathscr{H}$.

$$\;) = \mathbf{w}^T\mathbf{x} + w_0 \begin{cases} > 0 \text{ if } \mathbf{x} \in \mathscr{H}^+ \\ = 0 \text{ if } \mathbf{x} \in \mathscr{H} \end{cases}$$

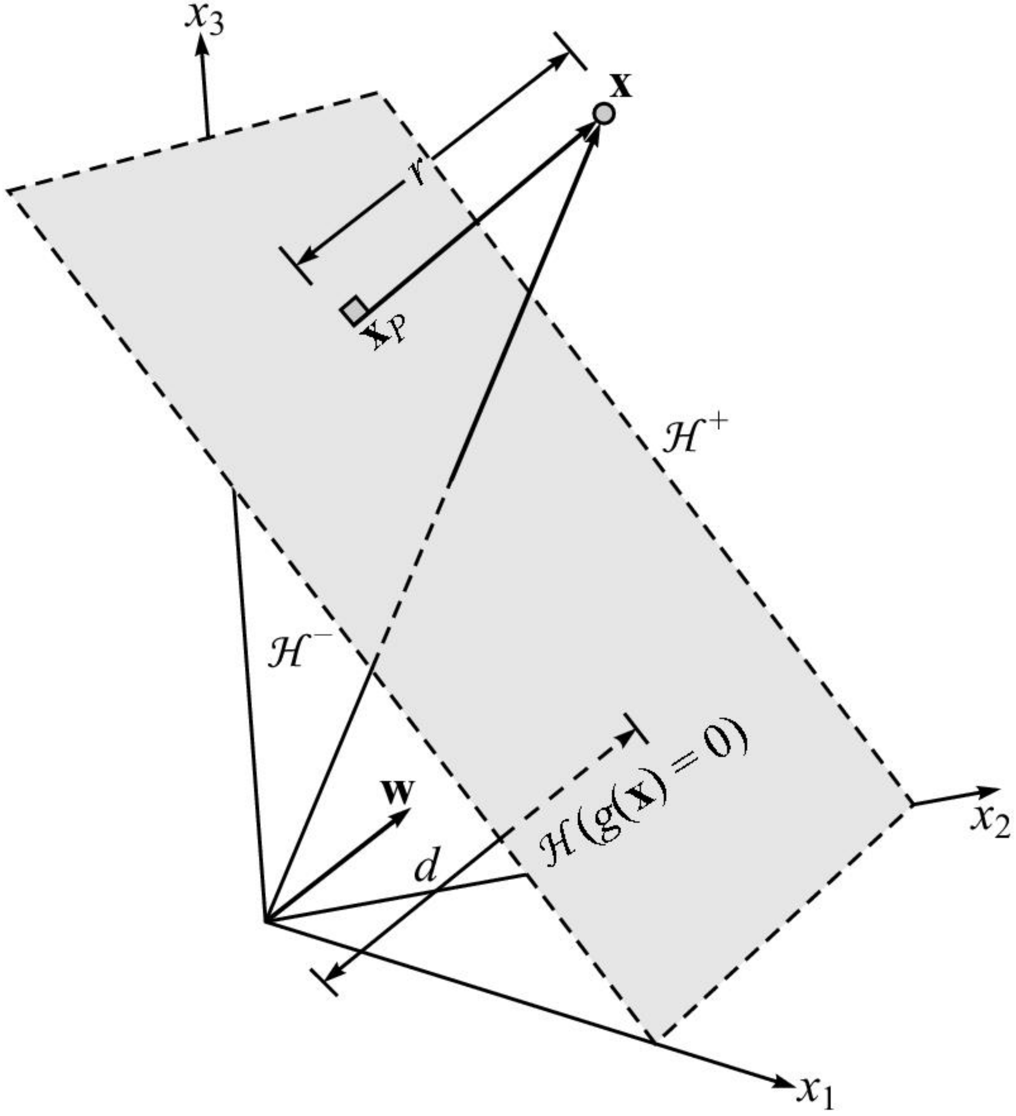


$$g(\mathbf{x}_d) = \mathbf{w}^T\mathbf{x}_d + w_0 = 0;\ \mathbf{x}_d = d\,\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$d\,\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} + w_0 = 0;\ d\,\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} = -w_0;\ d = \frac{-w_0}{\|\mathbf{w}\|}$$

Perpendicular distance d from coordinate origin to $\mathscr{H} = w_0/\|\mathbf{w}\|$

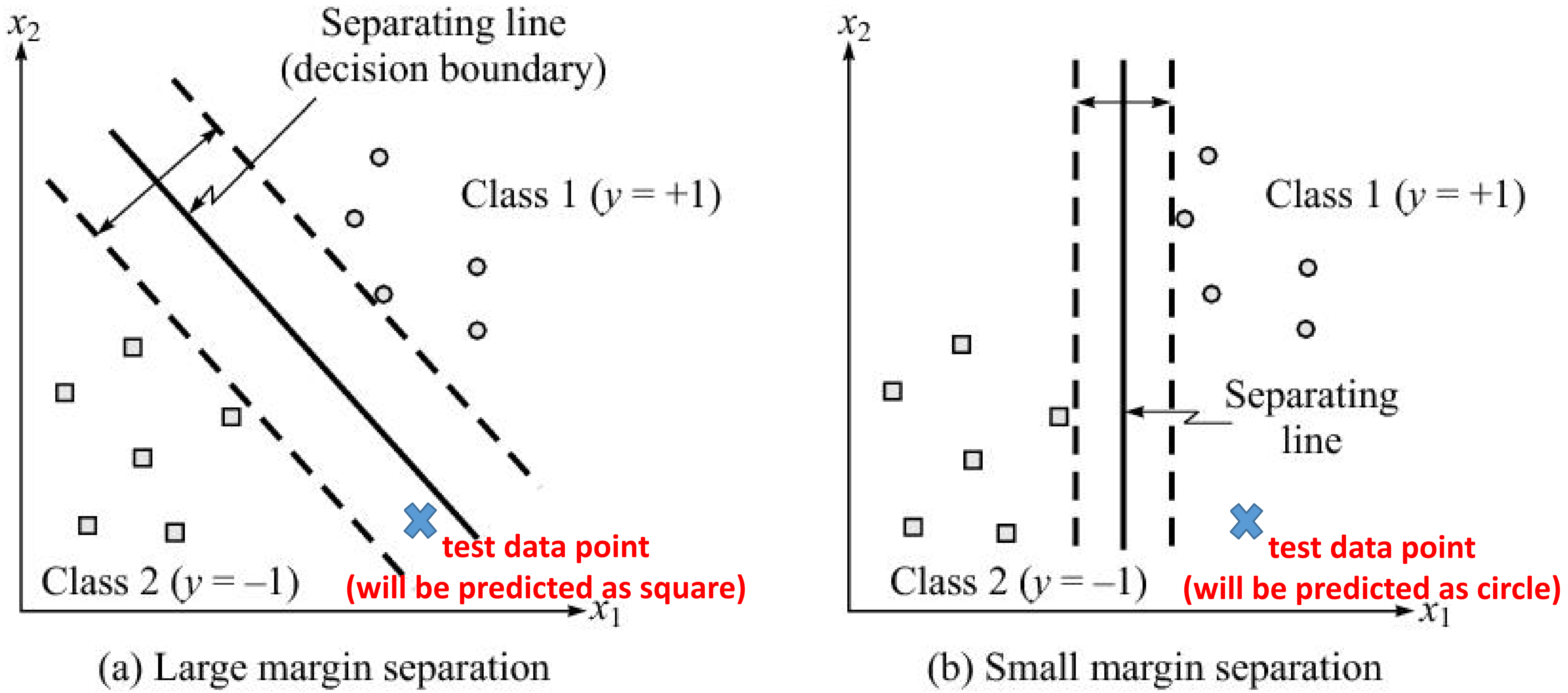
Geometry for 3-dimensions (n=3)



Hyperplane $\mathscr{H}$ separates the feature space into two half space $\mathscr{H}^+$ and $\mathscr{H}^-$

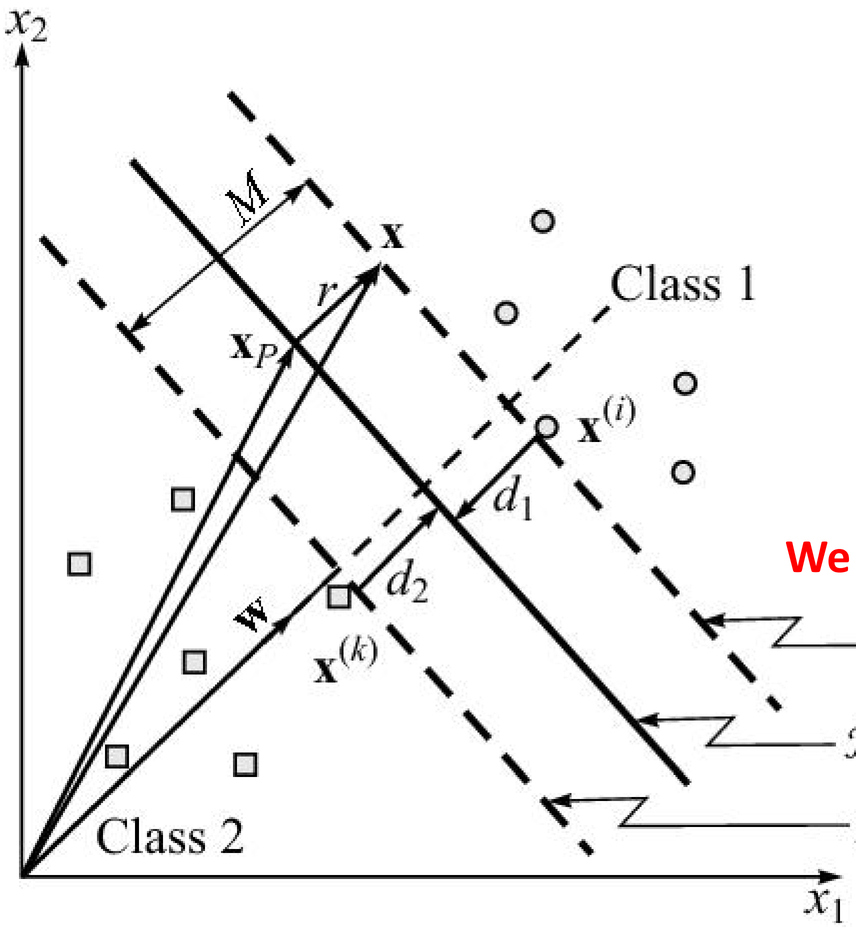
Linear Maximal Margin Classifier for Linearly Separable Data

- For linearly separable, many hyperplanes exist to perfrom separation.
- SVM framework tells which hyperplane is best.
- Hyperplane with the largest *margin which* minimizes training error.
- Select the decision boundary that is far away from both the classes.
- Large margin separation is expected to yield good generalization.
- in $w^T x + w_0 = 0$, w defines a direction perpendicular to the hyperplane.
- w is called the normal vector (or simply normal) of the hyperplane.
- Without changing the normal vector w, varying w_0 moves the hyperplane parallel to itself.



Large margin and small margin separation

Two parallel hyperplanes $\mathcal{H}_1$ and $\mathcal{H}_2$ that pass through $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(k)}$ respectively.



$$d_1 = \frac{g(\mathbf{x}^{(i)})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x}^{(i)} + w_0}{\|\mathbf{w}\|}$$

$$d_2 = \frac{g(\mathbf{x}^{(k)})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x}^{(k)} + w_0}{\|\mathbf{w}\|}$$

We can rescale w and w_0 to obtain $\mathcal{H}_1$ and $\mathcal{H}_2$ as:

$\mathcal{H}_1: \mathbf{w}^T\mathbf{x} + w_0 = +1\,(g(\mathbf{x}) > 0)$

$\mathcal{H}: \mathbf{w}^T\mathbf{x} + w_0 = 0\,(g(\mathbf{x}) = 0)$

$\mathcal{H}$ divides the input space into two half spaces

$\mathcal{H}_2: \mathbf{w}^T\mathbf{x} + w_0 = -1\,(g(\mathbf{x}) < 0)$

Geometric interpretation of algebraic distances of points to a hyperplane for two-dimensional case

$\mathscr{H}_1$ and $\mathscr{H}_2$ are parallel to the hyperplane $\mathbf{w}^T\mathbf{x} + w_0 = 0$.

$$\mathscr{H}_1: \mathbf{w}^T\mathbf{x} + w_0 = +1$$

$$\mathscr{H}_2: \mathbf{w}^T\mathbf{x} + w_0 = -1$$

such that

$$\mathbf{w}^T\mathbf{x}^{(i)} + w_0 \geq 1 \quad \text{if } y^{(i)} = +1$$

$$\mathbf{w}^T\mathbf{x}^{(i)} + w_0 \leq -1 \text{ if } y^{(i)} = -1$$

or equivalently,

$$d_1 = \frac{1}{||\mathbf{w}||}; d_2 = \frac{-1}{||\mathbf{w}||} \qquad y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + w_0\right) \geq 1$$

distance between the two hyperplanes = margin M

$$M = \frac{2}{||\mathbf{w}||}$$

This equation states that maximizing the margin of separation between

KKT Condition

Learning problem in SVM

- Linearly separable training examples,

$$\mathcal{D} = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)} \right), \left(\mathbf{x}^{(2)}, y^{(2)} \right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)} \right) \right\}$$

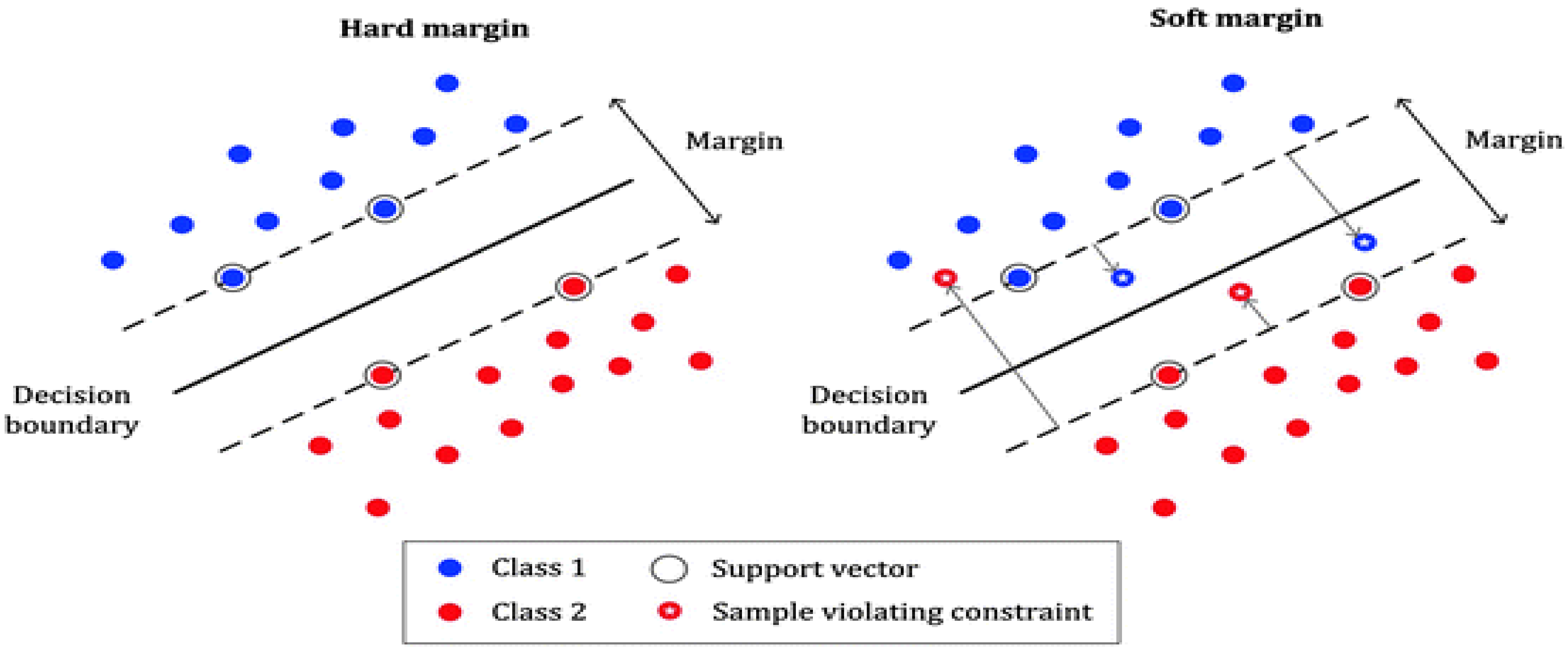
Problem: Solve the following constrained minimization problem:

$$\text{minimize } f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{subject to } \; y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + w_0 \right) \geq 1; i = 1, \ldots, N$$

This is the formulation of ***hard-margin*** SVM.

Hard margin svm Vs Soft margin svm



Dual formulation of constrained optimization problem:

Lagrangian is constructed:

$$L(\mathbf{w}, w_0, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \lambda_i \left[y^{(i)} \left(\mathbf{w}^T \mathbf{x}^{(i)} + w_0 \right) - 1 \right]$$

The Karush-Kuhn-Tucker (**KKT**) conditions are as follows:

i. $$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y^{(i)} \mathbf{x}^{(i)}$$

$$\frac{\partial L}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y^{(i)} = 0$$

ii. $y^{(i)} \left(\mathbf{w}^T \mathbf{x}^{(i)} + w_0 \right) - 1 \geq 0; i = 1, \ldots, N$

iii. $\lambda_i \geq 0; i = 1, \ldots, N$

- $\mathbf{w}$ is computed using condition (i) of KKT conditions

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y^{(i)} \mathbf{x}^{(i)}$$

- and w_0 is computed using condition (iv) of KKT conditions

$$\lambda_i [y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + w_0) - 1] = 0;\ i = 1, \ldots, N$$

- from, condition (iii), it can be said that Very small percentage have $\lambda_i > 0$
  - most among N, vanish with $\lambda_i = 0$.
- $\mathbf{x}^{(i)}$ whose $\lambda_i > 0$ are the *support vectors* and they lie on the margin.

- w is the weighted sum of these training instances that are selected as the support vectors:

$$\mathbf{w} = \sum_{i \,\epsilon\, svindex} \lambda_i y^{(i)} \mathbf{x}^{(i)}$$

- where $svindex$ is the set of indices of support vectors
- All support vectors are used to compute w_0, and then their average is taken for the final value of w_0

$$w_0 = \frac{1}{|svindex|} \sum_{i \,\epsilon\, svindex} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})$$

- where $|svindex|$ is the total number of indices in $svindex$, i.e., total number of support vectors.

The majority of λ_i are 0, for which $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) > 1$. These are the $\mathbf{x}^{(i)}$ points that exist more than adequately away from the discriminant, and have zero effect on the hyperplane. The instances that are not support vectors have no information; the same solution will be obtained on removing any subset from them. From this viewpoint, the SVM algorithm can be said to be similar to the k-NN algorithm (Section 3.4) which stores only the instances neighboring the class discriminant.

During testing, we do not enforce a margin. We calculate

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

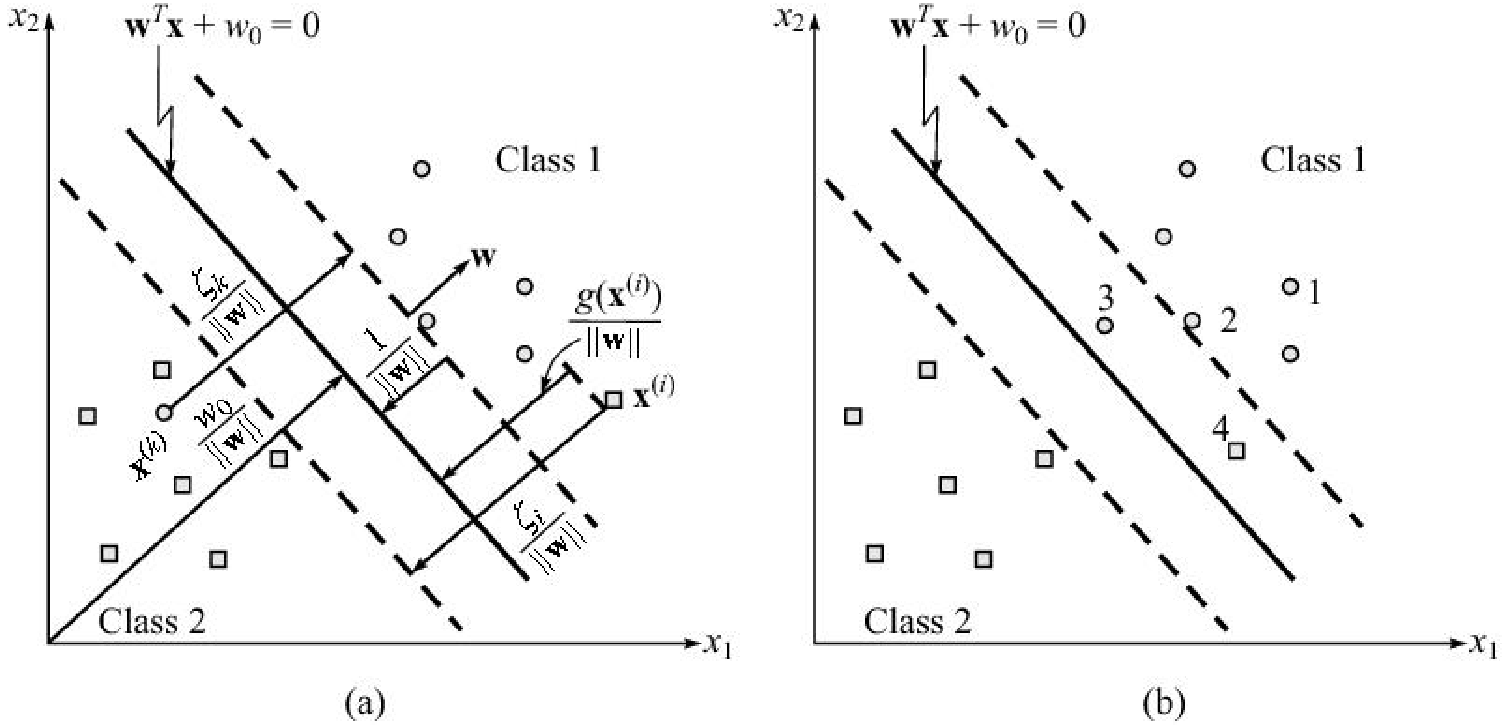
and choose the class according to the sign of $g(\mathbf{x})$: $sgn\,(g(\mathbf{x}))$ which we call the *indicator function* i_F,

$$i_F = \hat{y} = sgn\,(\mathbf{w}^T\mathbf{x} + w_0)$$

Choose Class 1 ($\hat{y} = +1$) if $\mathbf{w}^T\mathbf{x} + w_0 > 0$, and Class 2 ($\hat{y} = -1$) otherwise.

Linear Soft Margin Classifier for Overlapping Classes

- To generalize SVM, allow noise in the training data.
- Hard margin linear SVM algorithm will not work.



Soft decision boundary

- To allow error in data, relax margin constraints by invoking *slack* variables $\zeta_i (\geq 0)$:

$$\mathbf{w}^T\mathbf{x}^{(i)} + w_0 \geq 1 - \zeta_i \text{ for } y^{(i)} = +1$$

$$\mathbf{w}^T\mathbf{x}^{(i)} + w_0 \leq -1 + \zeta_i \text{ for } y^{(i)} = -1$$

Thus, new constraints:

$$y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + w_0\right) \geq 1 - \zeta_i; i = 1, \ldots, N$$

$$\zeta_i \geq 0$$

Penalize the errors by assigning extra cost and change the objective function to

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_{i=1}^{N}\zeta_i\right); C \geq 0$$

- Hence it all boils down to optimization problem

$$\text{minimize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i$$

$$\text{subject to} \quad y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + w_0\right) \geq 1 - \zeta_i; i = 1,\ldots,N$$

$$\zeta_i \geq 0; \mathrm{i} = 1,\ldots,N$$

This formulation is the *soft margin* SVM.

Lagrangian

$$L\left(\mathbf{w}, w_0, \zeta, \lambda, \mu\right) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\lambda_i\left[y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + w_0\right) - 1 + \zeta_i\right]$$

$$- \sum_{i=1}^{N}\mu_i\zeta_i$$

Using KKT conditions, the dual formulation of the *soft-margin SVM* is reduced to

$$\text{maximize } L_{*}(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \lambda_i \lambda_k y^{(i)} y^{(k)} \mathbf{x}^{(i)T} \mathbf{x}^{(k)}$$

$$\text{subject to } \sum_{i=1}^{N} \lambda_i y^{(i)} = 0$$

$$0 \le \lambda_i \le C; i = 1, \ldots, N$$

- ζ_i and μ_i- not in the dual objective function.
- The objective function is identical to that for separable case.
- Only difference - constraint $\lambda_i \le C$

λ_i can have values in the interval $0 \le \lambda_i \le$ C. thus three cases are there:

Case 1: $\lambda_i = 0$

Don’t contribute to the optimum value of **w**.

Case 2: $0 < \lambda_i < C$

Corresponding patterns are on the margin.

Case 3: $\lambda_i = C$

Corresponding pattern is misclassified or lies inside the margin.

- support vectors define **w** ($\lambda_i > 0$); $\mathbf{w} = \sum_{i \,\epsilon\, svindex} \lambda_i y^{(i)} \mathbf{x}^{(i)}$

$svindex$ is the set of indices of support vectors

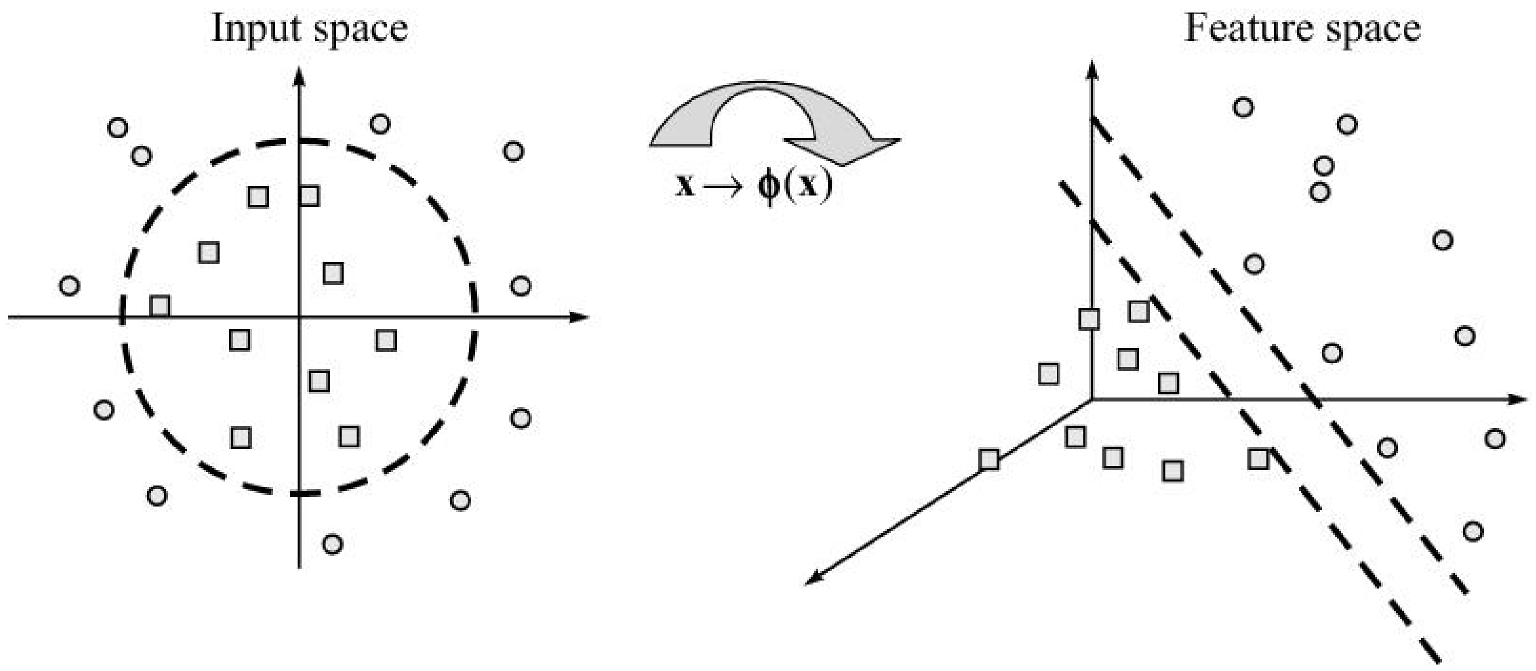
Kernel Function: Dealing with non linearity

Non-linear classifiers

- for several real-life datasets, the decision boundaries are nonlinear.
- To deal with nonlinear case, the formulation and solution methods employed for the linear case are still applicable.
- Only input data is transformed from its original space into another space (higher dimensional space) so that a linear decision boundary can separate Class 1 examples from Class 2.
- The transformed space is called the feature space.
- The original data space is known as the input space.

Non-linear classifiers

- For training examples which cannot be linearly separated.
- In the feature space, they can be separated linearly with some transformations.



Transformation from input space to feature space

The new optimization problem becomes

$$\text{minimize} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\zeta_i$$

$$\text{subject to} \quad y^{(i)}\left(\mathbf{w}^T\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right) + w_0\right) \geq 1 - \zeta_i; i = 1, \ldots, N$$

$$\zeta_i \geq 0; \mathrm{i} = 1, \ldots, N$$

The corresponding dual is

$$\text{minimize}\ L_*(\lambda) = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\lambda_i\lambda_k y^{(i)}\left[\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)\right]^T\boldsymbol{\phi}\left(\mathbf{x}^{(k)}\right)$$

$$\text{subject to} \sum_{i=1}^{N}\lambda_i y^{(i)} = 0$$

The decision boundary becomes:

$$\sum_{i=1}^{N} \lambda_i y^{(i)} \left[\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)\right]^T \boldsymbol{\phi}(\mathbf{x}) + w_0 = 0$$

- Is there a need to know the mapping of $\boldsymbol{\phi}$? No.

In SVM, this is done through the use of *kernel function*, denoted by *K*.

$$K\left(\mathbf{x}^{(i)}, \mathbf{x}\right) = \left[\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)\right]^T \boldsymbol{\phi}(\mathbf{x})$$

There is no explicit need to know what $\boldsymbol{\phi}$ is.

Constructing Kernels:

- Does any kernel work? No, only valid kernel functions work. Identification of $\boldsymbol{\phi}$ is not needed if it can be shown whether the function is a kernel or not without the need of mapping.

- Function satisfying Mercer's theorem can work as kernel function.

- Mercer's theorem, which provides a test whether a function $K\left(\mathbf{x}^{(i)},\mathbf{x}^{(k)}\right)$ constitutes a valid kernel without having to construct the function $\boldsymbol{\phi}(\mathbf{x})$.

$$K\left(\mathbf{x}^{(i)},\mathbf{x}^{(k)}\right) = \left[\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)\right]^T \boldsymbol{\phi}\left(\mathbf{x}^{(k)}\right)$$

Mercer’s theorem:

Every semi-positive definite symmetric function is a kernel

$K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}\right)$ is a kernel function if and only if the matrix K is positive semidefinite.

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & K(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) & \cdots & K(\mathbf{x}^{(1)}, \mathbf{x}^{(N)}) \\ \vdots & \vdots & K(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) & \vdots \\ K(\mathbf{x}^{(N)}, \mathbf{x}^{(1)}) & K(\mathbf{x}^{(N)}, \mathbf{x}^{(2)}) & \cdots & K(\mathbf{x}^{(N)}, \mathbf{x}^{(N)}) \end{bmatrix}$$

A positive semidefinite matrix is a Hermitian matrix (a complex square matrix that is equal to its own conjugate transpose) all of whose eigenvalues are nonnegative.

Polynomial and Radial Basis Kernel

Common kernel functions used:

Polynomial kernel of degree d

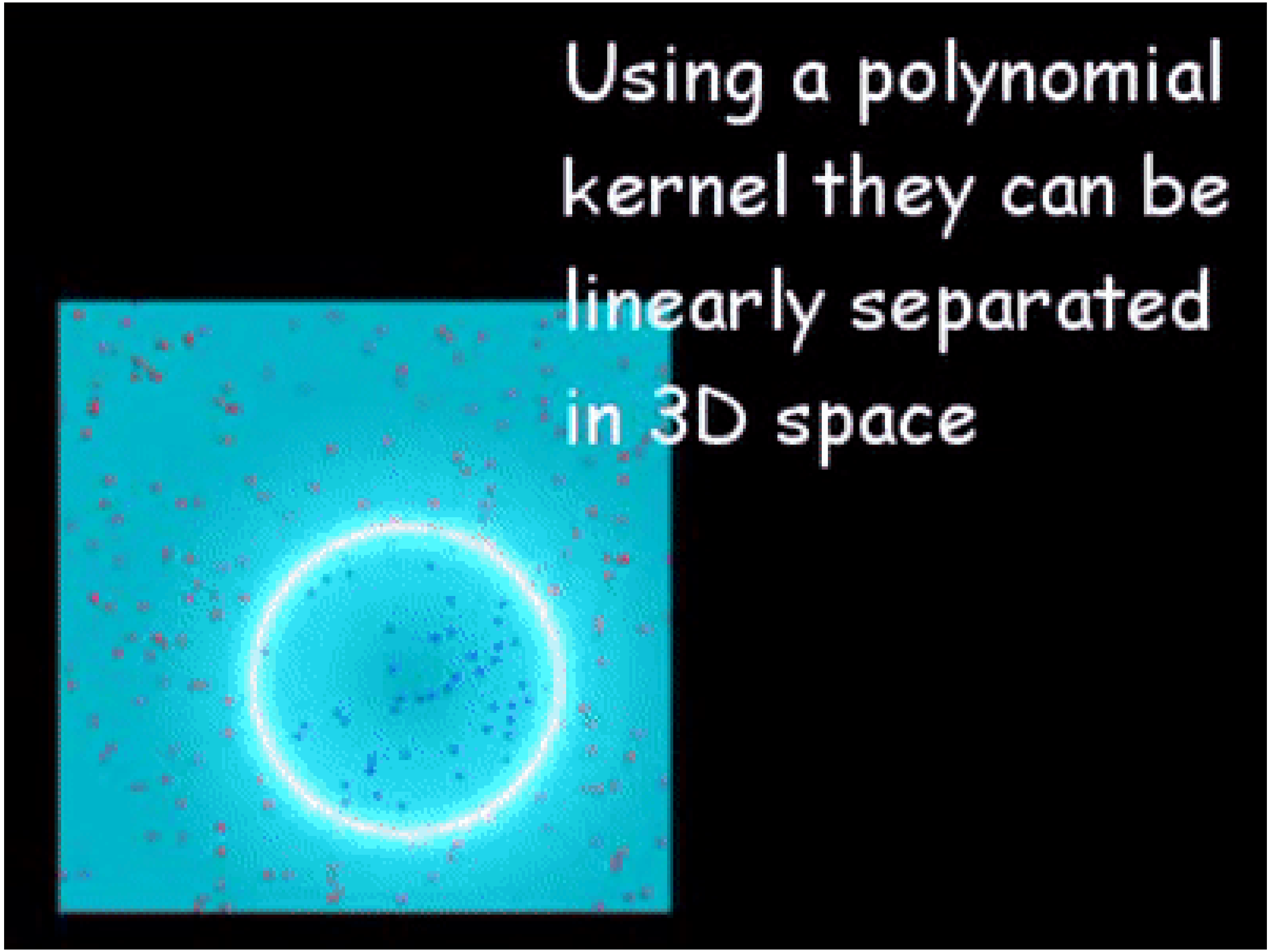
$$K\left(\mathbf{x}^{(i)},\mathbf{x}^{(k)}\right) = \left(\mathbf{x}^{(i)T}\mathbf{x}^{(k)} + c\right)^{d}; c > 0,\ d \geq 2$$

Gaussian radial basis function kernel (RBF)

$$K\left(\mathbf{x}^{(i)},\mathbf{x}^{(k)}\right) = \exp\left(-\frac{\left|\left|\mathbf{x}^{(i)} - \mathbf{x}^{(k)}\right|\right|^{2}}{2\sigma^{2}}\right); \sigma > 0$$

Each of these results in a different nonlinear classifier in (the original) input space.

Polynomial Kernel



Polynomial Kernel

- The polynomial kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.
- It looks not only at the given features of input samples to determine their similarity, but also combinations of these (interaction features).
- Quite popular in natural language processing (NLP).
- The most common degree is d = 2 (quadratic), since larger degrees tend to overfit on NLP problems.
- One problem with the polynomial kernel is that it may suffer from numerical instability: (result ranges from 0 to infinity)

Radial Basis Kernel

- RBF kernels are the most generalized form of kernelization.
- It is one of the most widely used kernels due to its similarity to the Gaussian distribution.
- The RBF kernel function for two points X_1 and X_2 computes the similarity or how close they are to each other.

$$K\left(\mathbf{x}^{(i)},\mathbf{x}^{(k)}\right) = \exp\left(-\frac{\left|\left|\mathbf{x}^{(i)} - \mathbf{x}^{(k)}\right|\right|^2}{2\sigma^2}\right); \sigma > 0$$

where,

'σ' is the variance and our hyperparameter

$||X_1 - X_2||$ is the Euclidean (L_2-norm) Distance between two points X_1 and X_2

Radial Basis Kernel

- The maximum value that the RBF kernel can get is 1 and occurs when d_{12} is 0 which is when the points are the same, i.e. $X_1 = X_2$.
- When the points are the same, there is no distance between them and therefore they are extremely similar.
- When the points are separated by a large distance, then the kernel value is less than 1 and close to 0 which would mean that the points are dissimilar.

- There are no golden rules for determining which admissible kernel will result in the most accurate SVM.
- In practice, the kernel chosen does not generally make a large difference in resulting accuracy.
- SVM training always finds a global solution, unlike neural networks (to be discussed in the next chapter) where many local minima usually exist.