AI Course

Dr. Mürsel Taşgın

Support Vector Machines

Support Vector Machines

- Developed by Vapnik et al. (1995) at AT&T Bell laboratories
- Supervised learning model; it can be used for classification and regression analysis
- Linear model

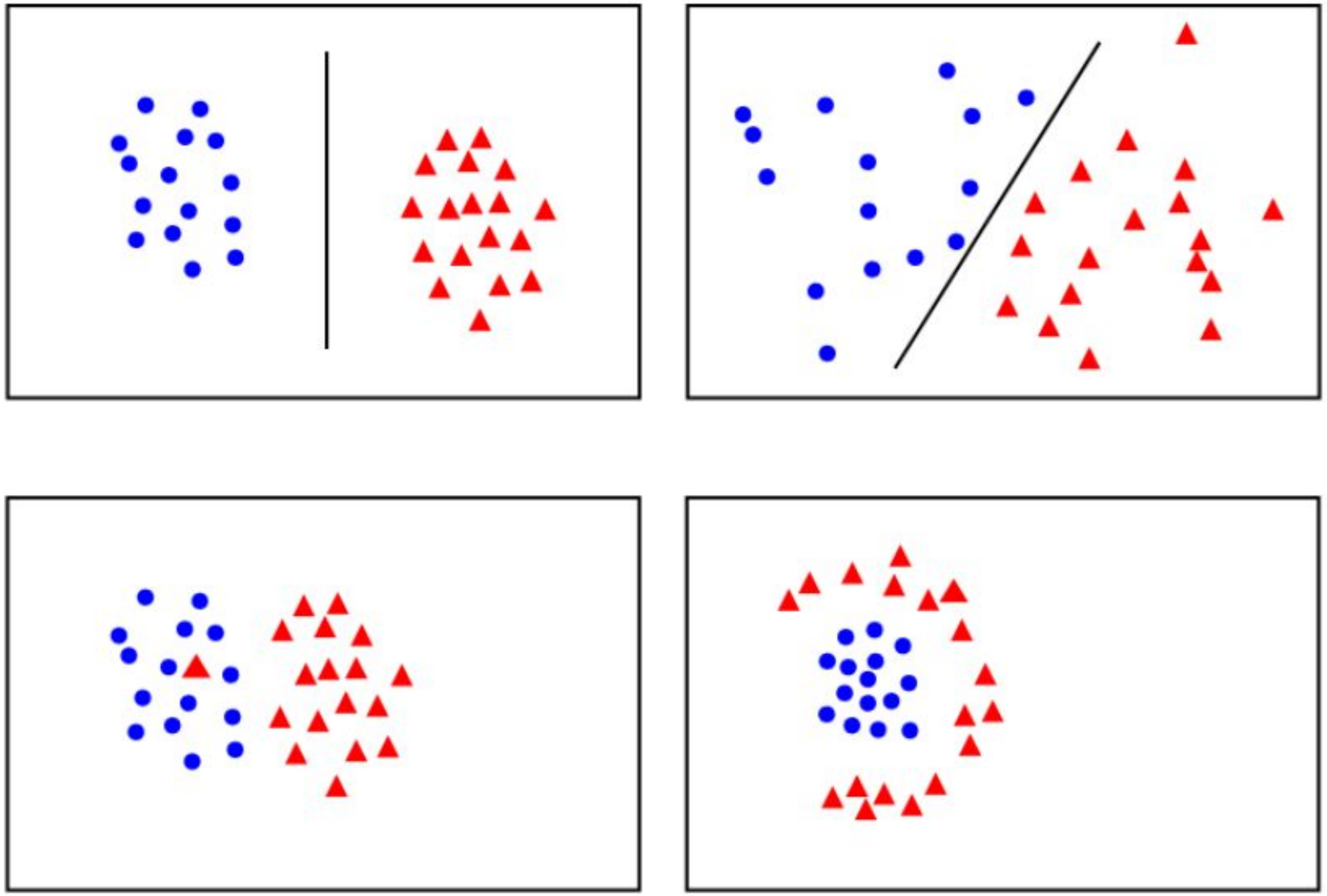
How it works

- Given a set of training examples, an SVM model learns from data to assign new examples to one category or the other
- Hyperplanes
- Margins

Support Vector Machines – *Linear separability*

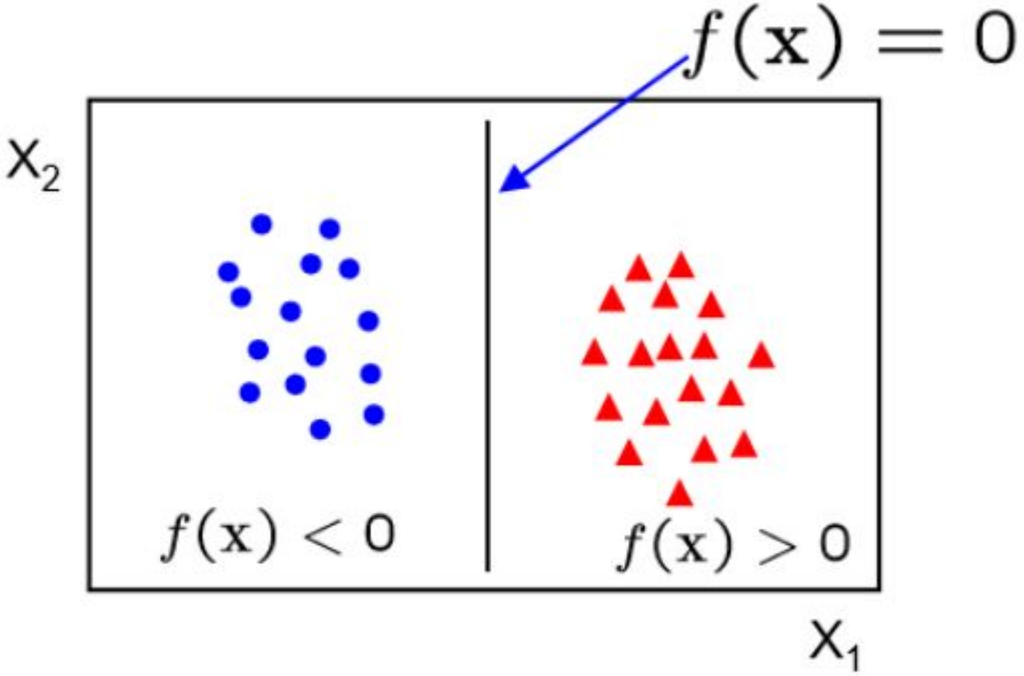
linearly separable

not linearly separable



Support Vector Machines

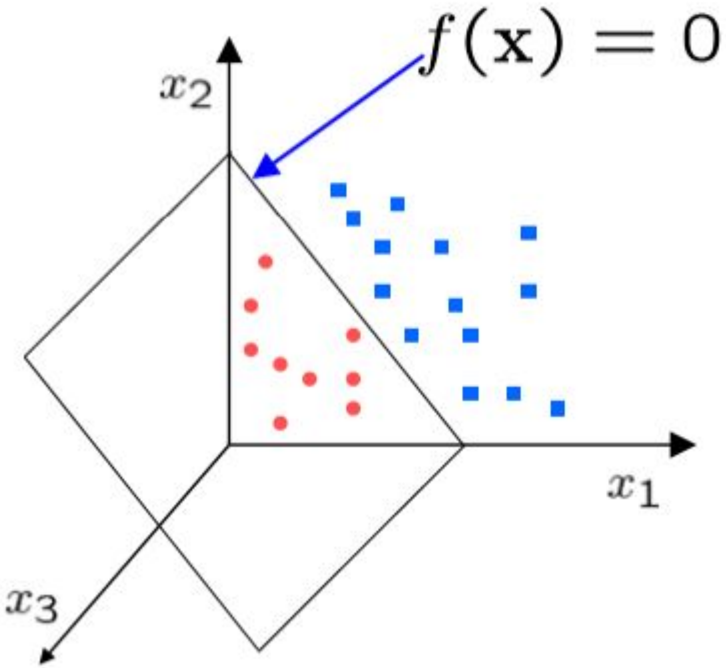
A linear classifier has the form: $f(x) = w^T x + b$



- In 2D, the discriminant is a line
- ***w*** is the normal to the line, and ***b*** is the bias
- ***w*** is known as weight vector

Support Vector Machines

A linear classifier has the form: $\boldsymbol{f(x) = w^T x + b}$



- In 3D, the discriminant is a plane and in upper dimensions (n-D), it is a ***hyperplane***

Support Vector Machines

Given linearly separable data x_i labelled into two categories $y_i = \{-\mathbf{1}, \mathbf{1}\}$,

Find a weight vector w such that the discriminant function

$$\boldsymbol{f(x) = w^T x + b}$$

separates the categories for i=1,..,N

How can we find this separating hyperplane?

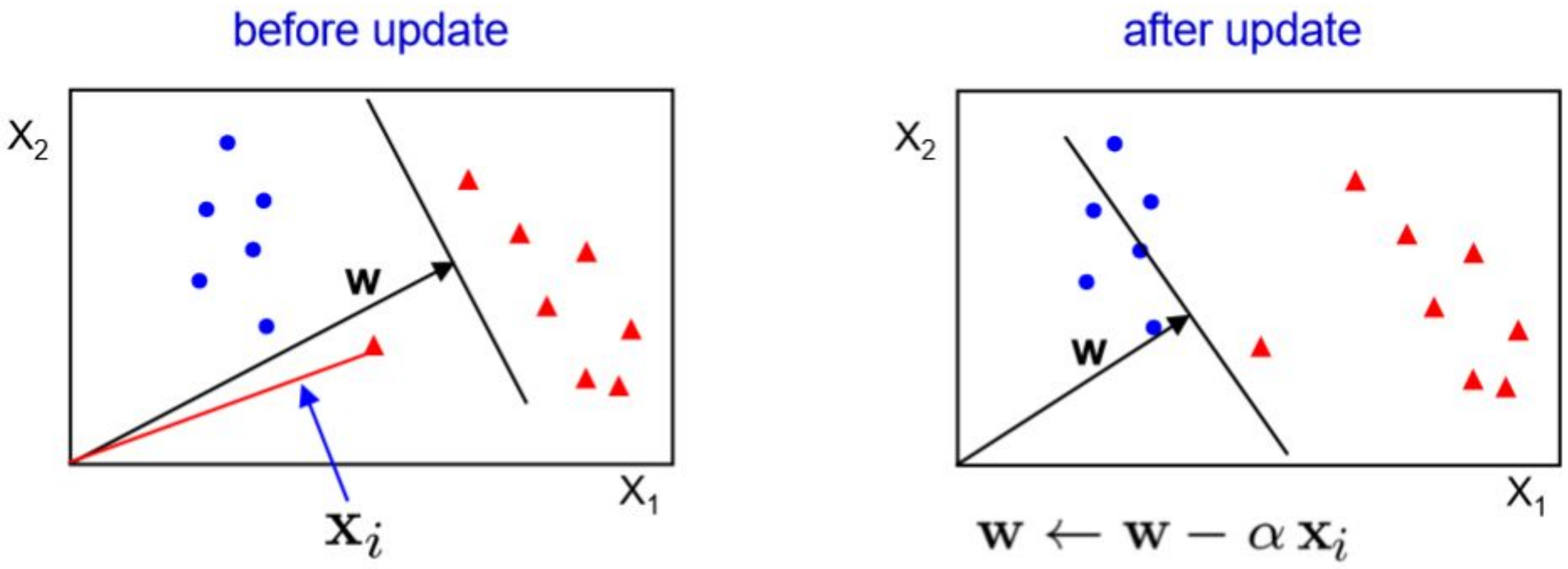
The Perceptron Algorithm

Write classifiers as $\boldsymbol{f(x_i) = w^T x_i + w_0 = w^T x_i}$

- Initialize w = 0
- Cycle through the data points
- If x_i is misclassified, correct labels$\{x_i, y_i\}$
- Loop until all data is correctly classified

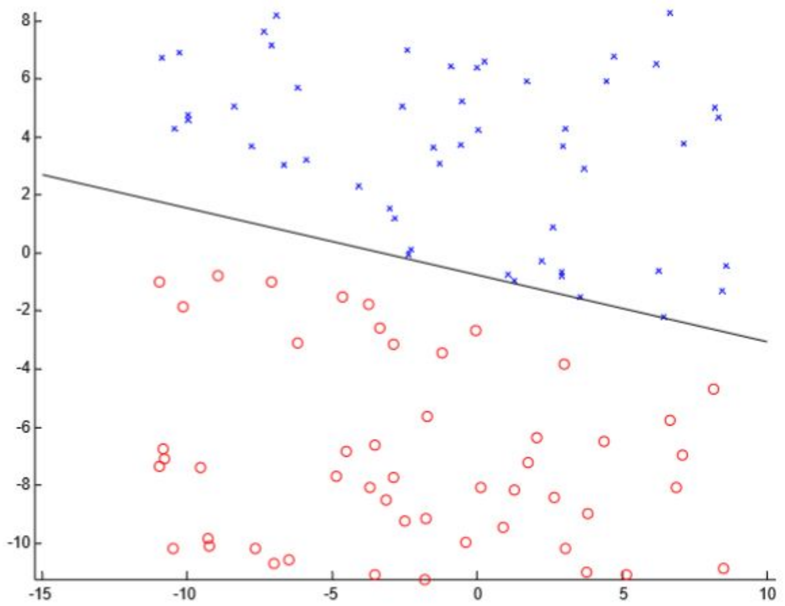
Support Vector Machines

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha\,\mathrm{sign}(f(\mathbf{x}_i))\,\mathbf{x}_i$
- Until all the data is correctly classified



Support Vector Machines

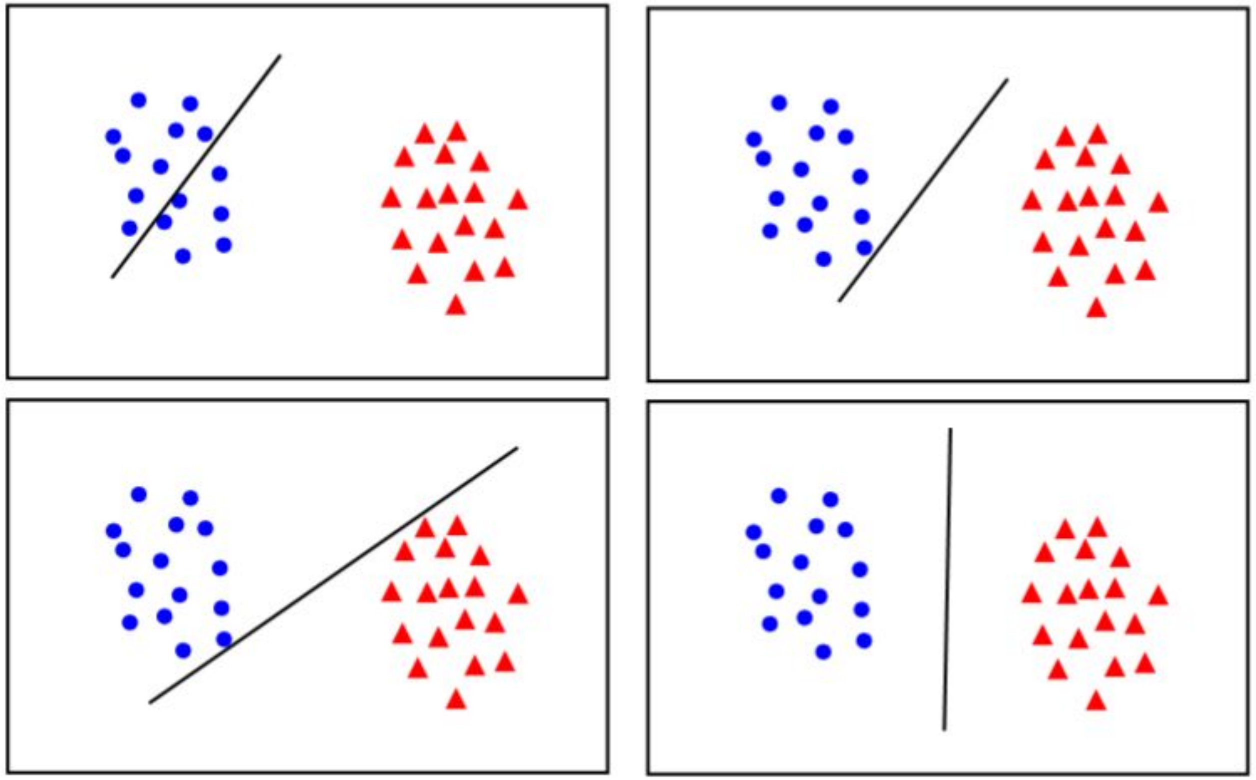
Perceptron example



- if the data is linearly separable, then the algorithm will converge
- Convergence can be slow
- Separating line close to training data
- We would prefer a larger margin for generalization

Support Vector Machines

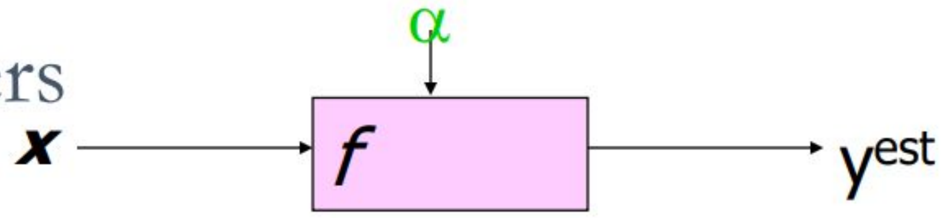
What is the best w?



- maximum margin solution: most stable under perturbations of the inputs

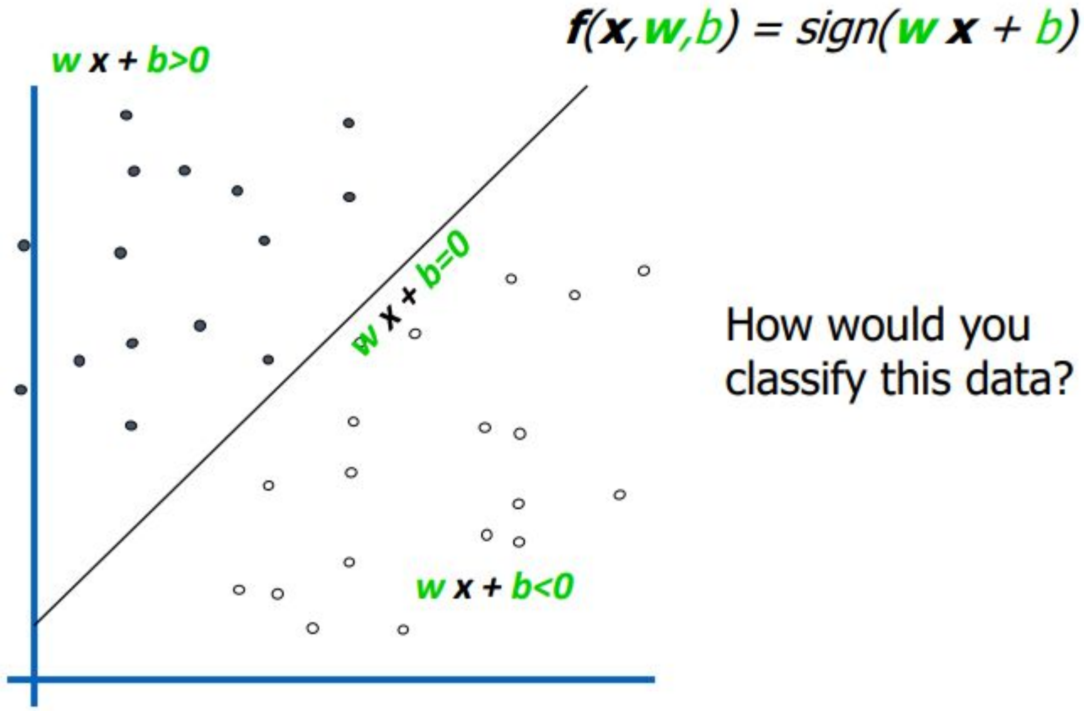
Support Vector Machines

Linear Classifiers



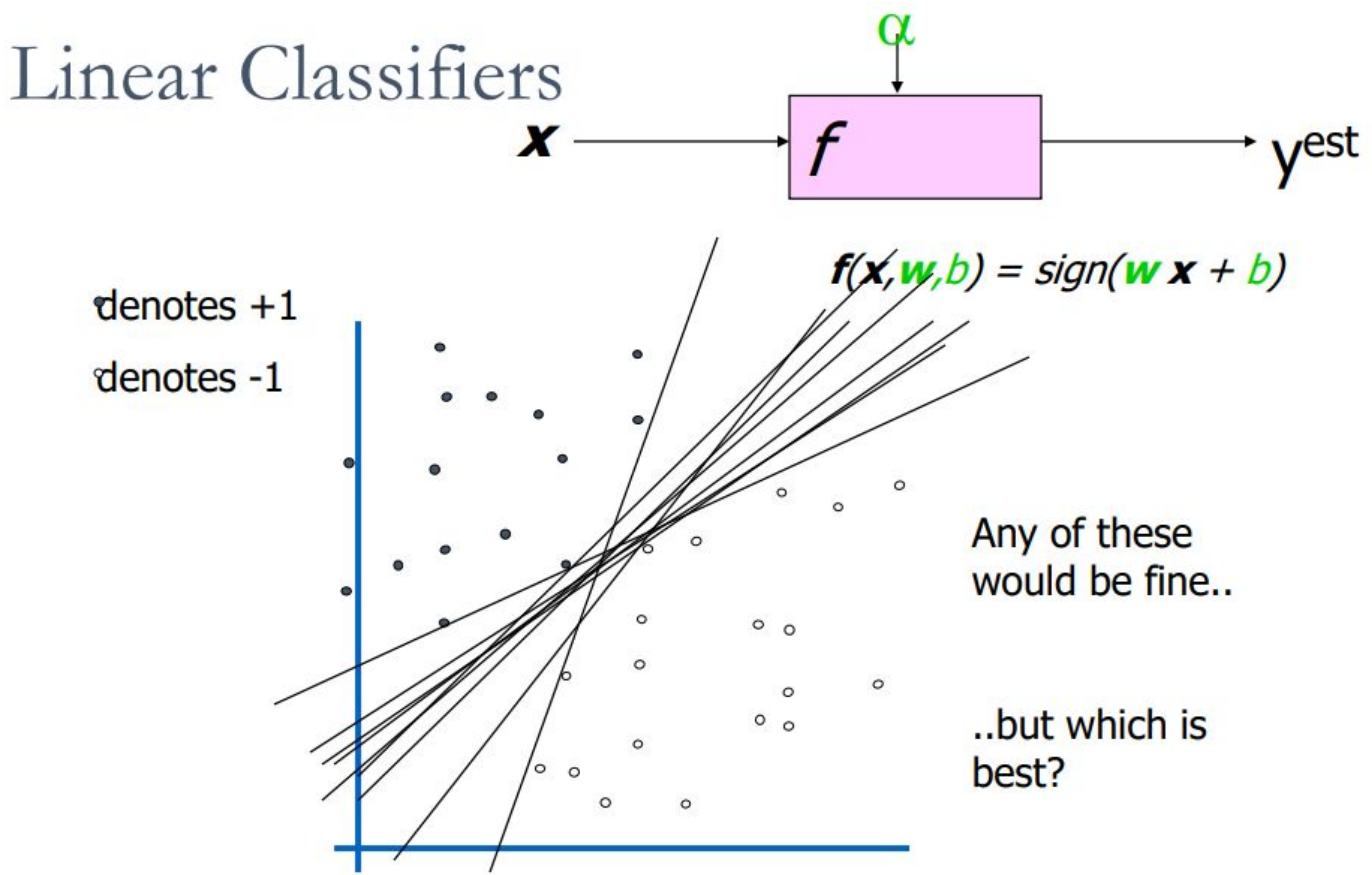
•denotes +1

∘denotes -1

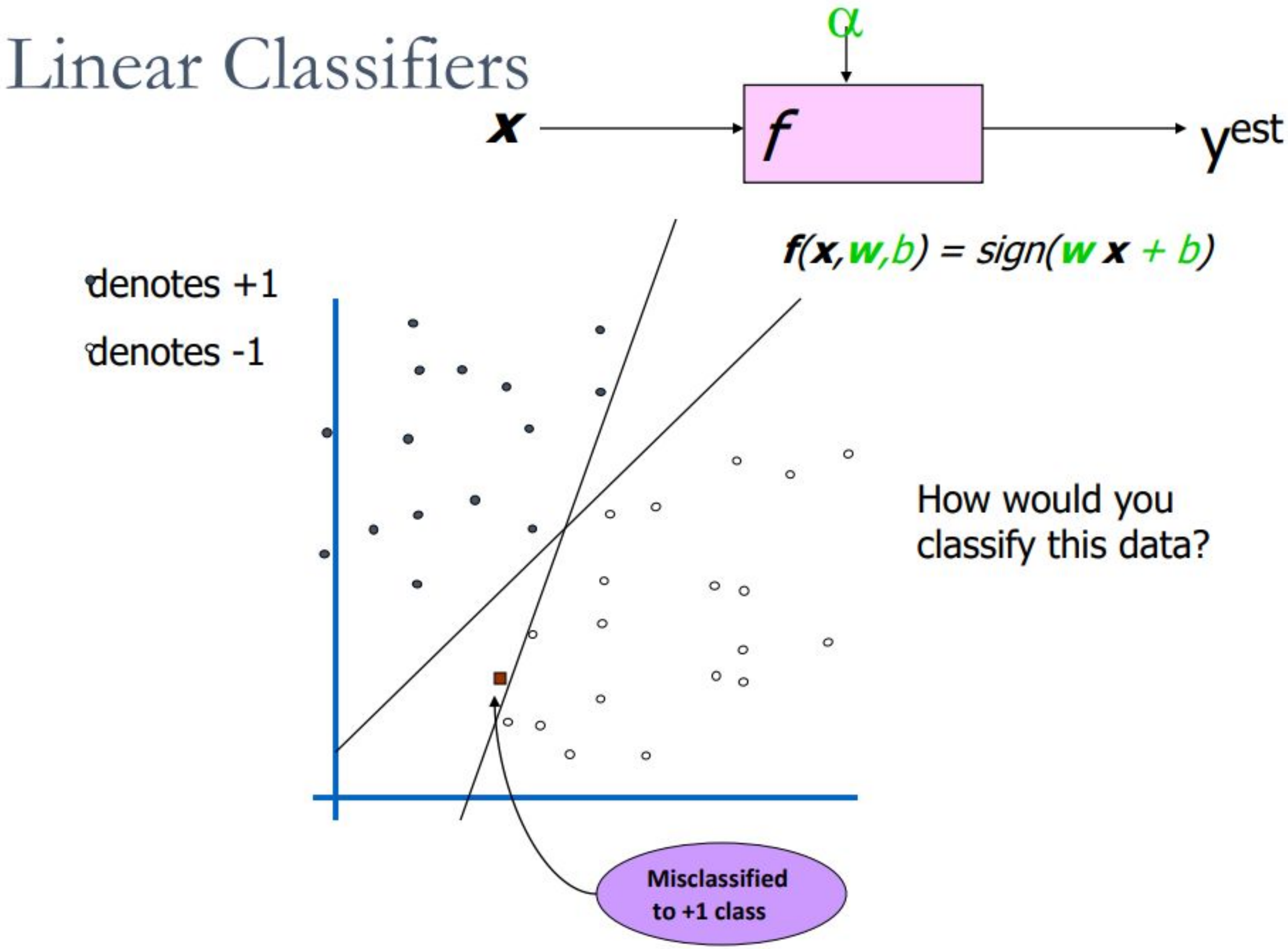


How would you classify this data?

Support Vector Machines



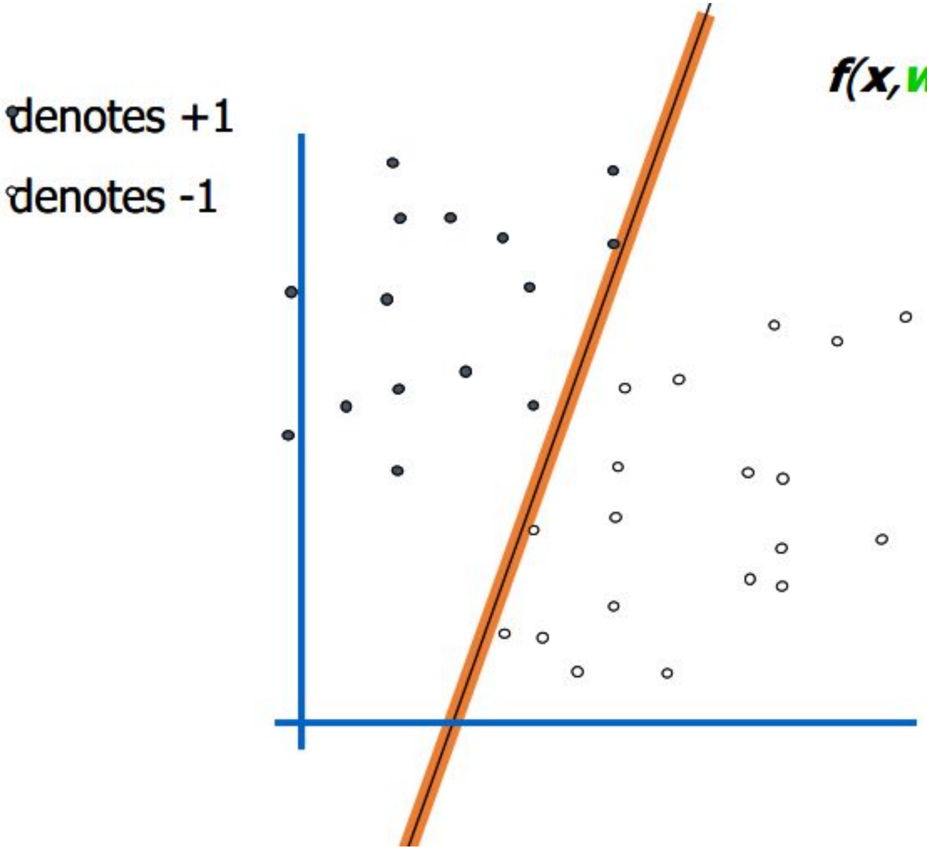
Support Vector Machines



How would you classify this data?

Support Vector Machines - *Margins*

Classifier margin



$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$$

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

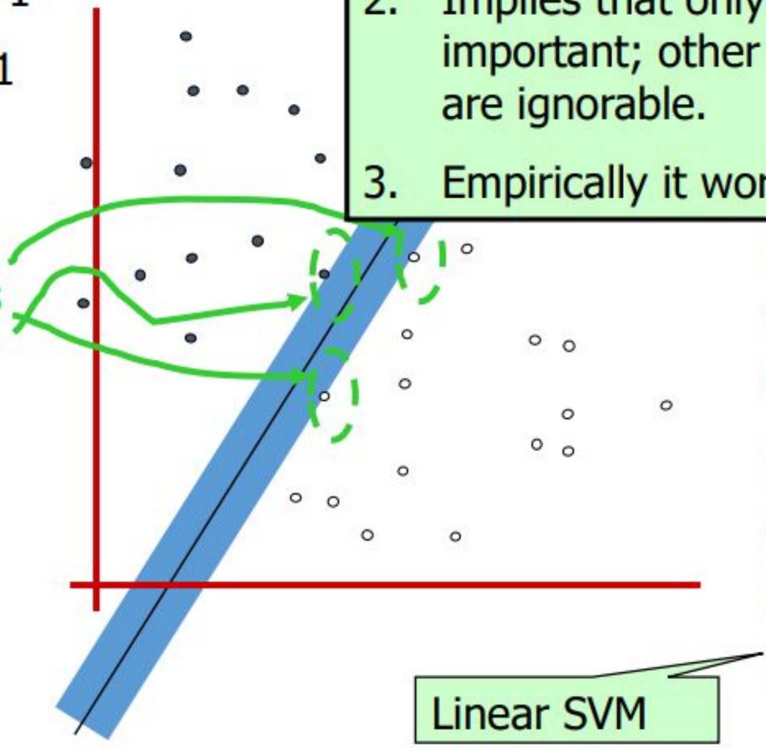
Support Vector Machines – *Support Vectors*

denotes +1

denotes -1

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

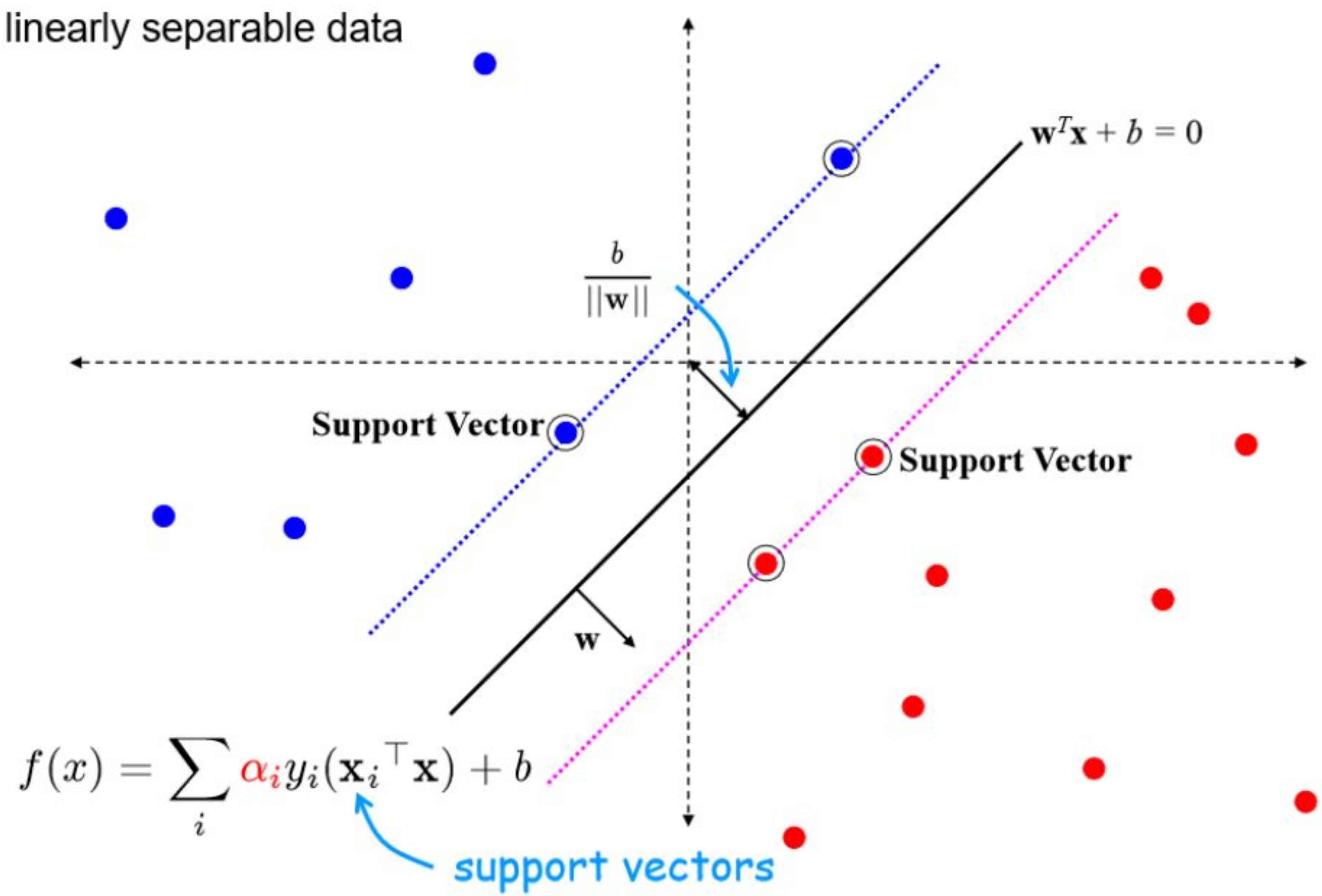
Support Vectors are those datapoints that the margin pushes up against



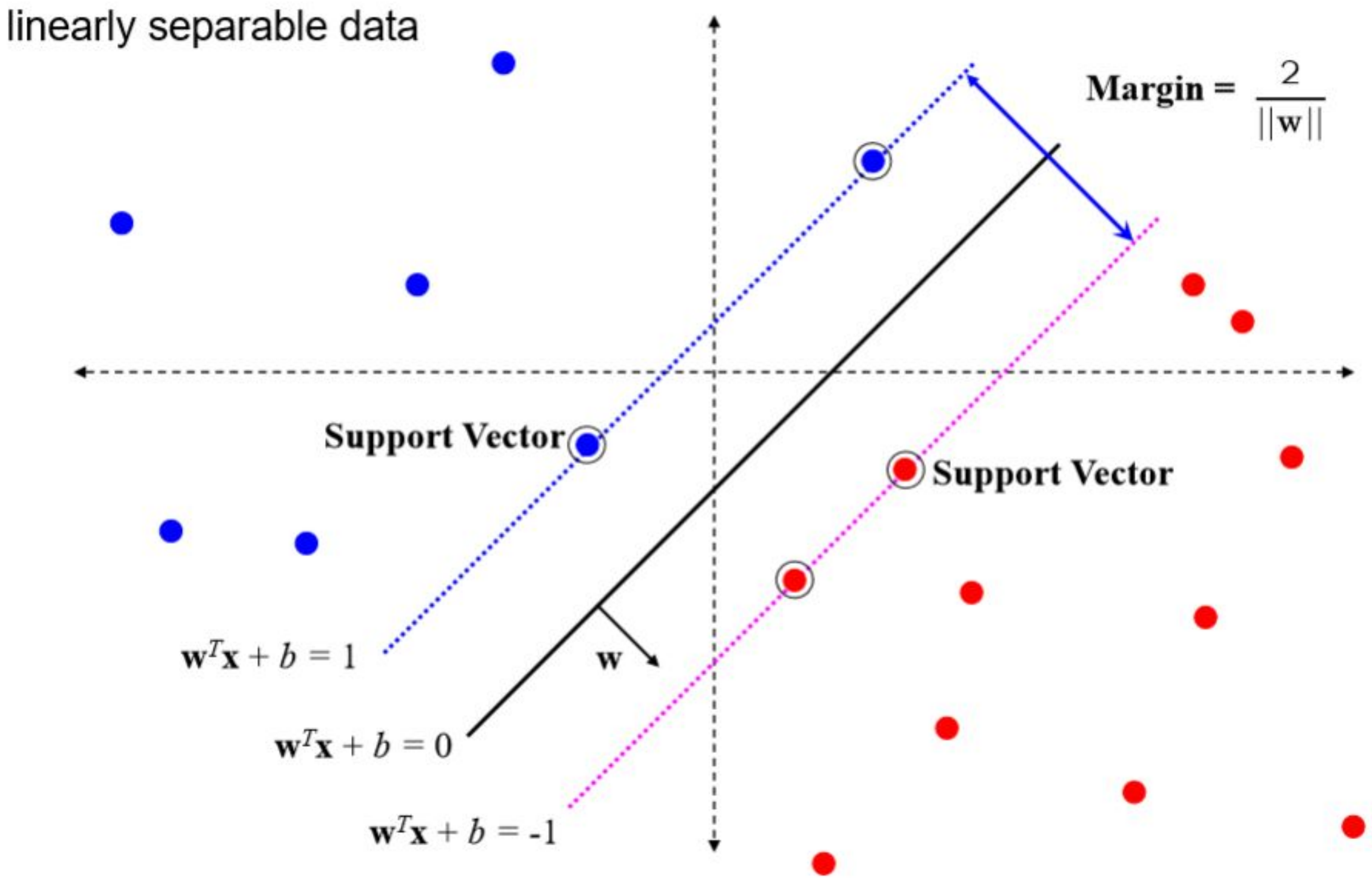
linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Support Vector Machines – *Support Vectors*



Support Vector Machines – *Support Vectors*



Support Vector Machines

- Goal: **1) Correctly classify all training data**

$$wx_i + b \geq 1 \quad \textit{if } y_i = +1$$
$$wx_i + b \leq 1 \quad \textit{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$

2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

Support Vector Machines

SVM – Optimization

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{ll} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{array} \text{ for } i = 1 \ldots N$$

- Or equivalently

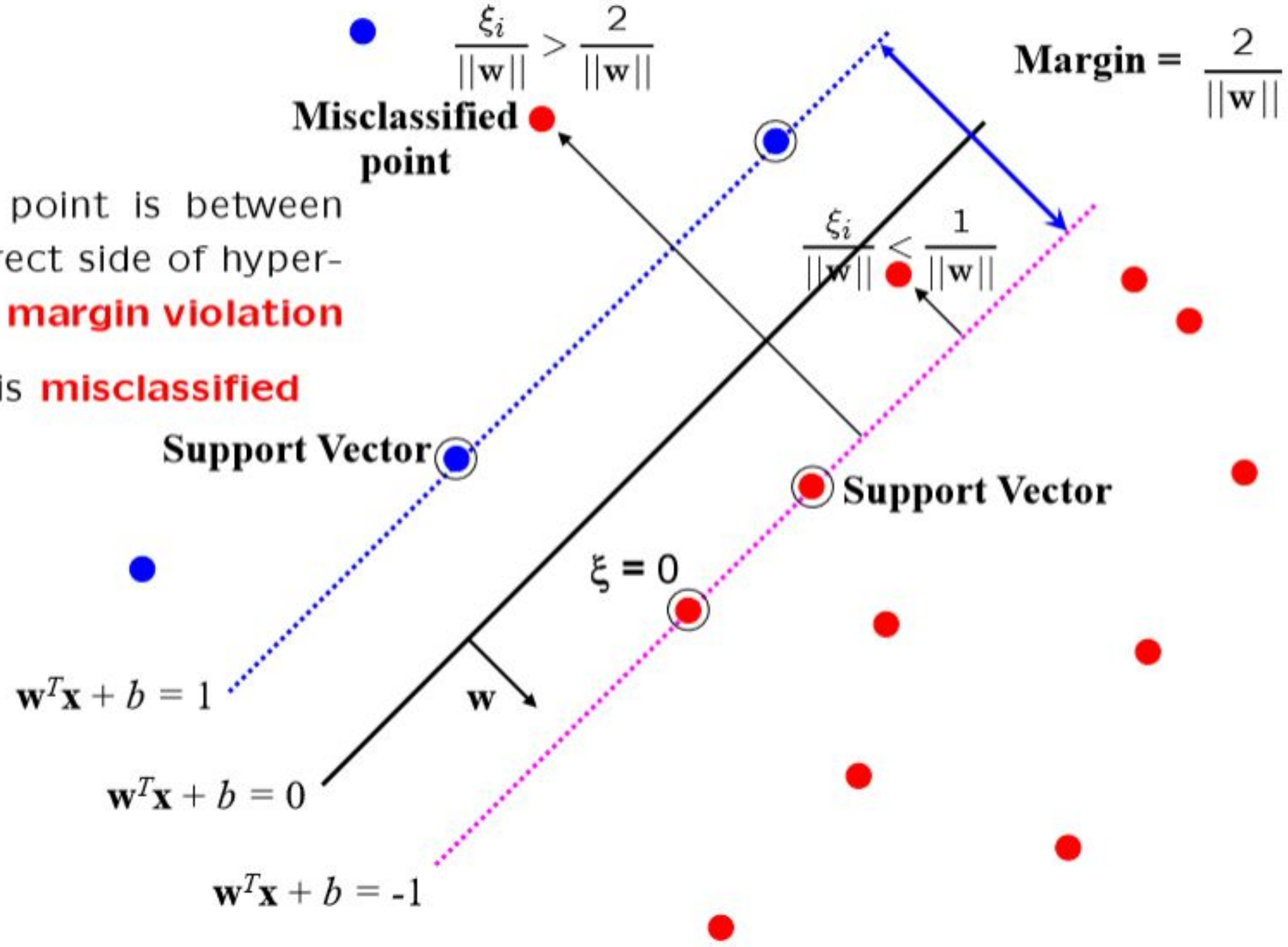
$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 \text{ for } i = 1 \ldots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

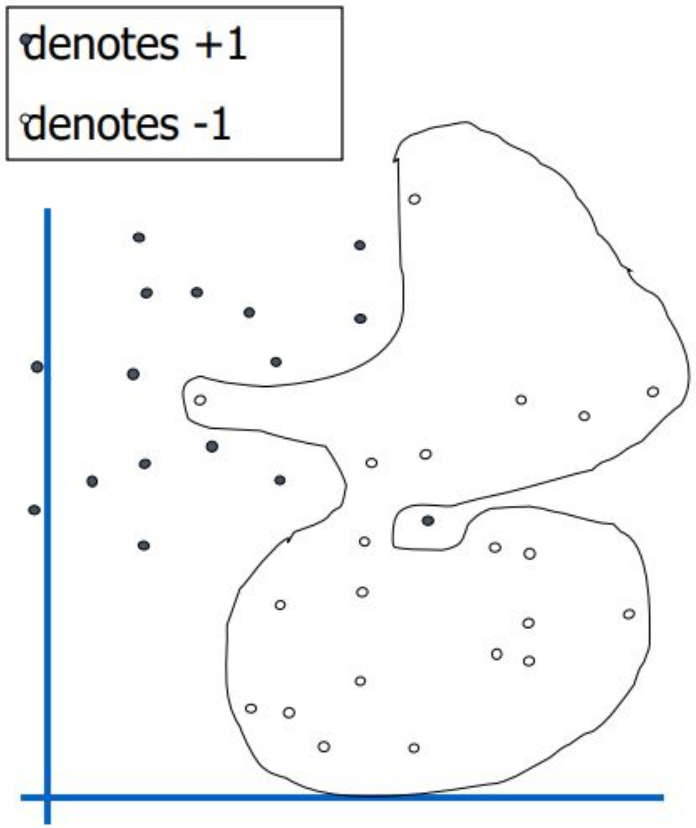
Support Vector Machines – *Slack Variables*

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



Support Vector Machines



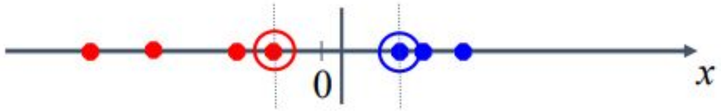
- **Hard Margin:** So far we require all data points be classified correctly
  - No training error
- **What if the training set is noisy?**
  - **Solution 1:** use very powerful kernels

OVERFITTING!

Support Vector Machines

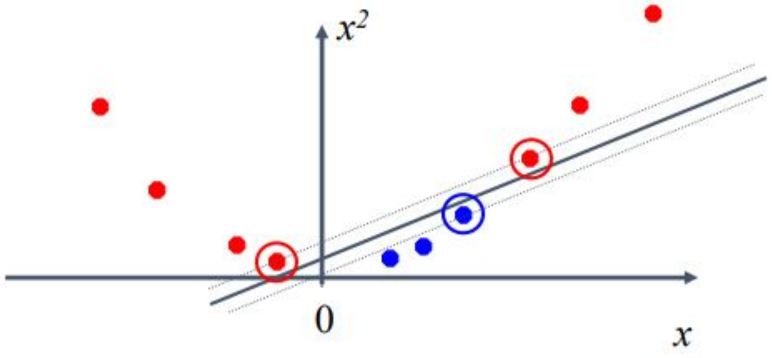
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

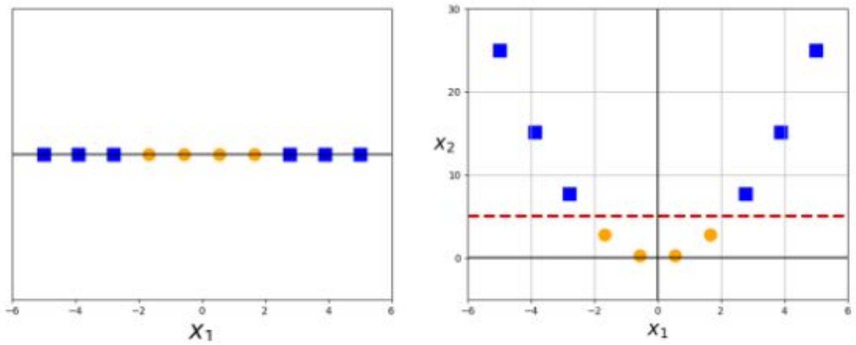


- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:



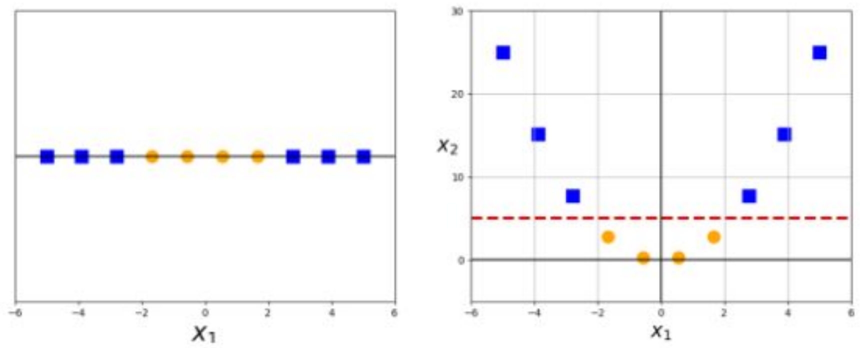
$\phi(x) = x^2$



This data becomes linearly separable after a quadratic transformation to 2-dimensions.

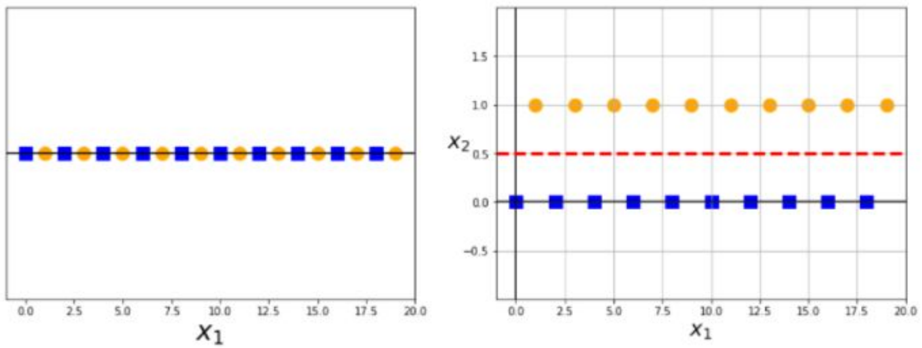
Support Vector Machines

$\phi(x) = x^2$



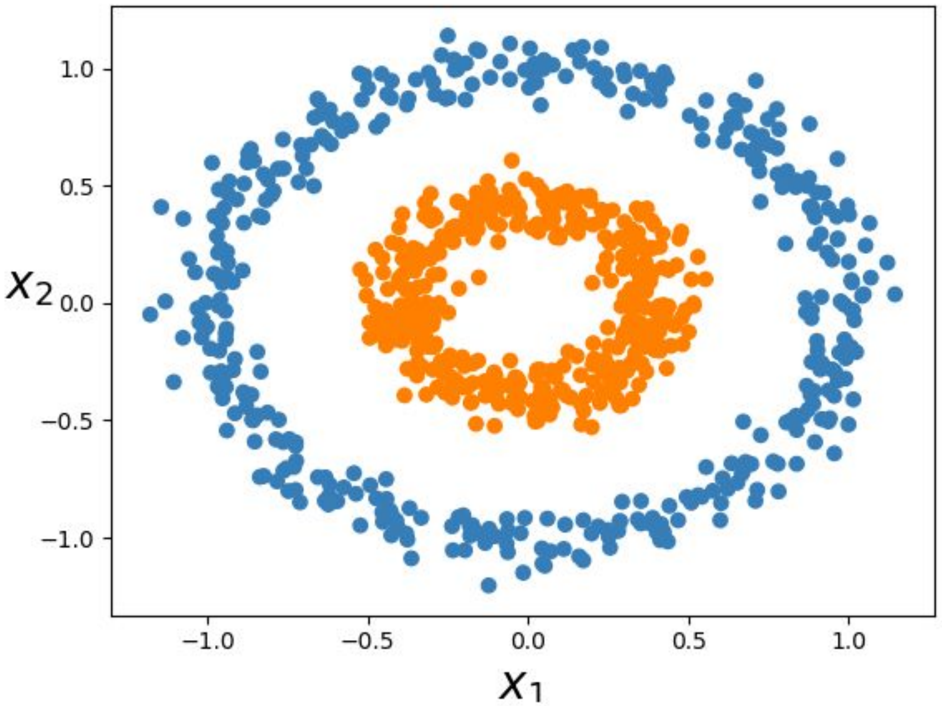
This data becomes linearly separable after a quadratic transformation to 2-dimensions.

$\phi(x) = x \bmod 2$

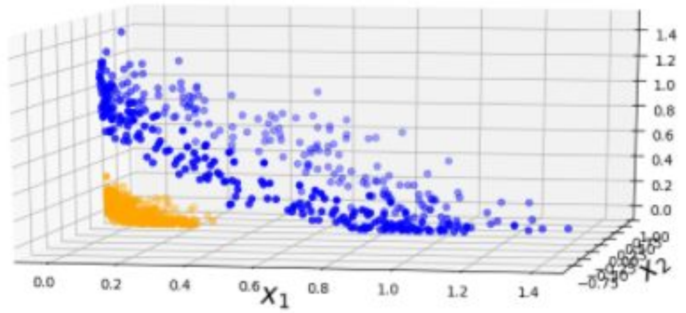


This transformation allows us to linearly separate the even and odd X1 values in 2 dimensions.

Support Vector Machines



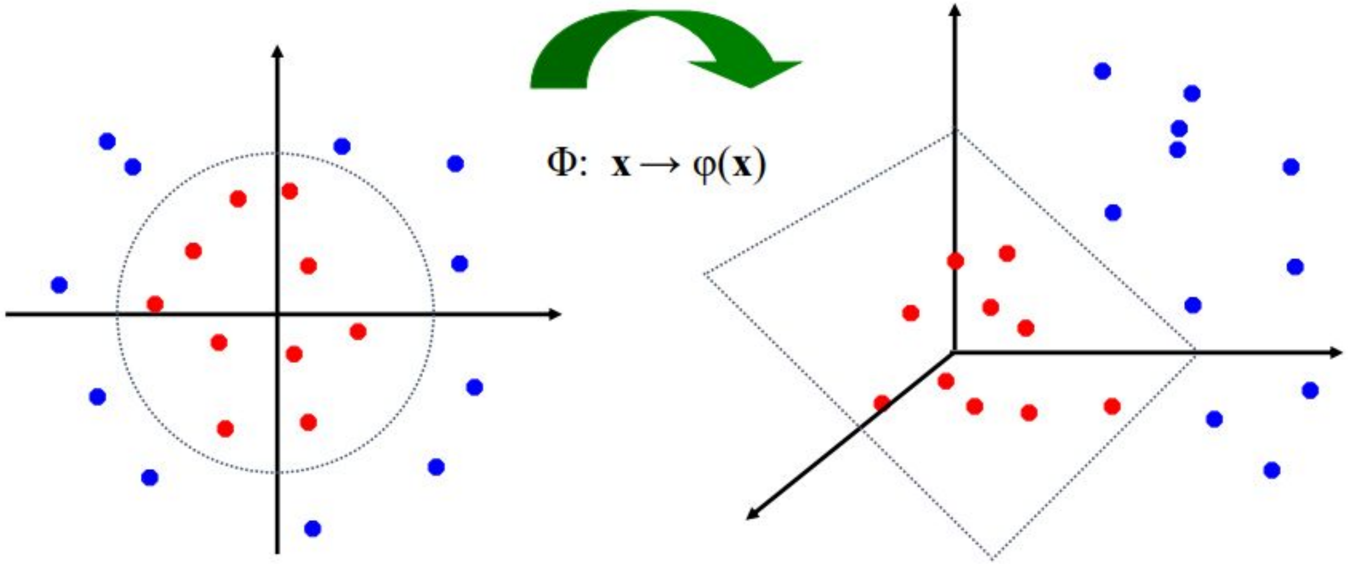
$$\phi(\mathbf{x}) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_1^{\ 2} \\ \sqrt{2}\,x_1 x_2 \\ x_2^{\ 2} \end{pmatrix}$$



Support Vector Machines

Non-linear SVMs: Feature spaces

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

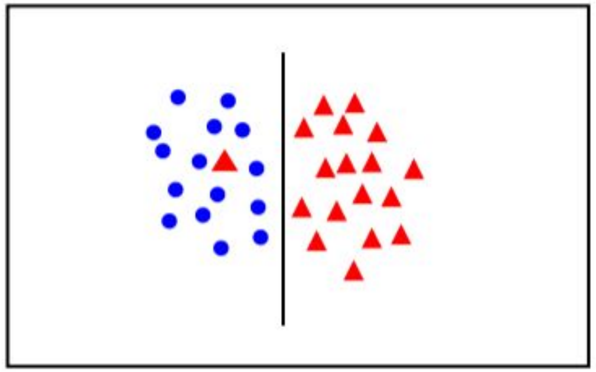


Nonlinear SVM - Overview

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

Support Vector Machines

Handling data that is not linearly separable

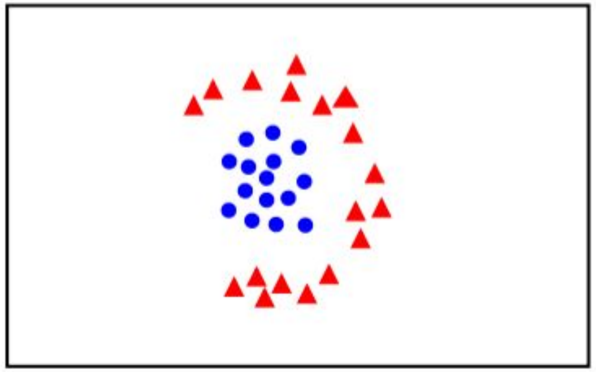


- introduce slack variables

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

$$y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1\ldots N$$

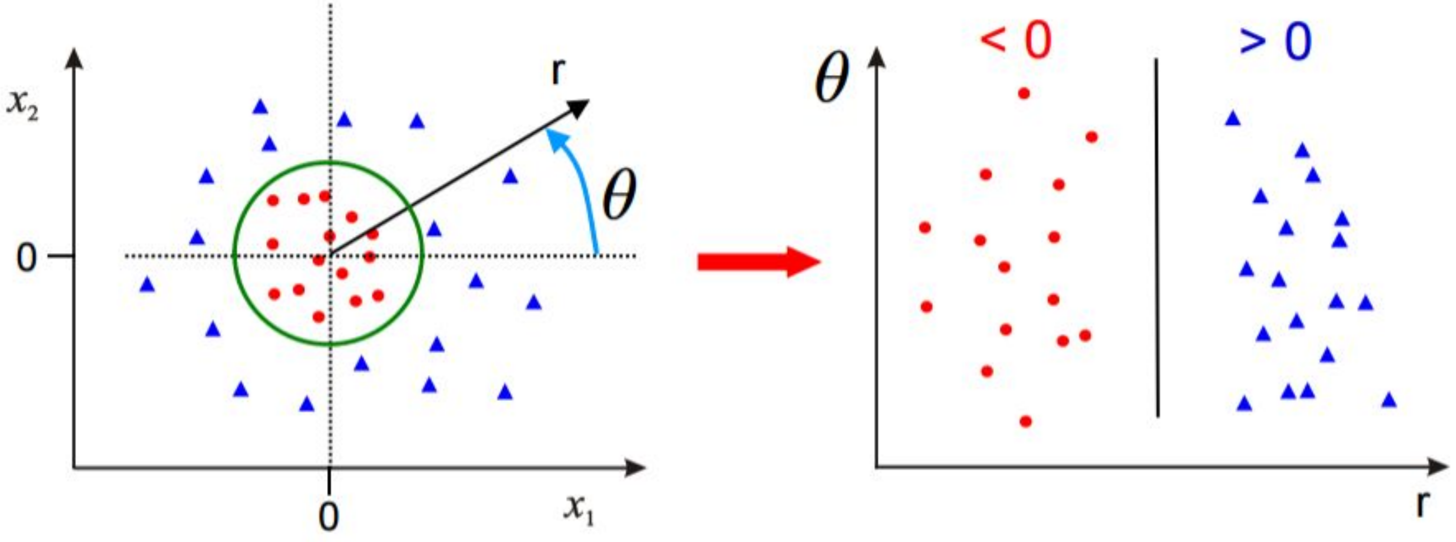


- linear classifier not appropriate

??

Support Vector Machines

Solution 1: use polar coordinates



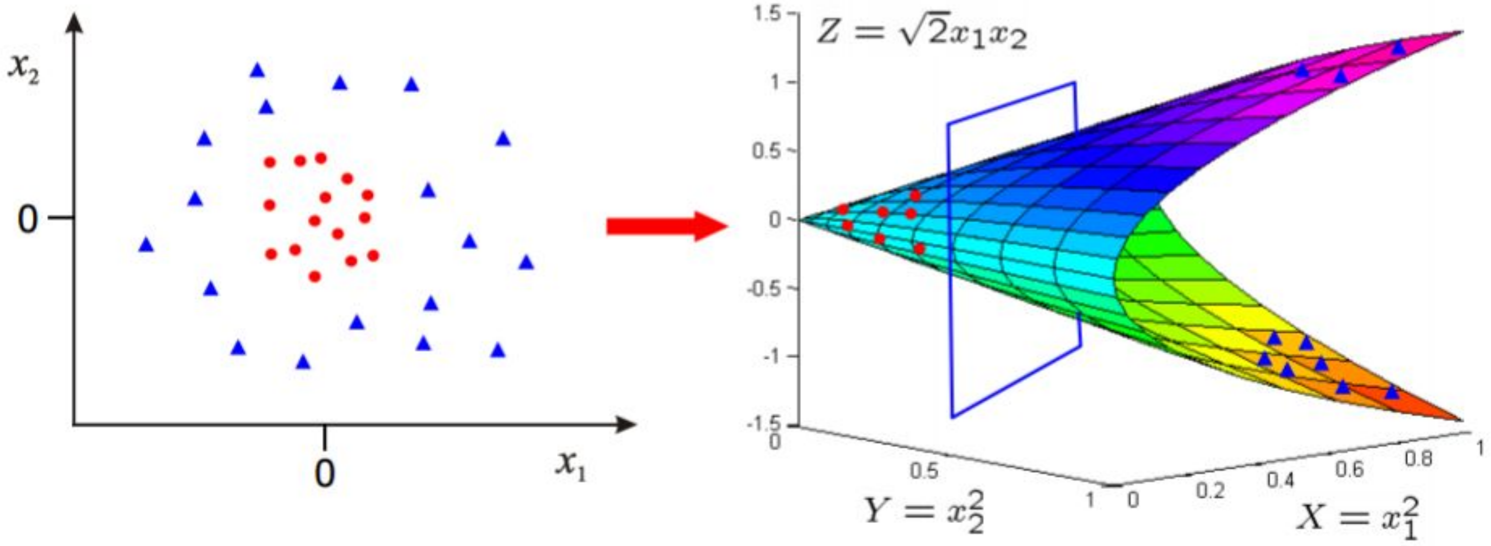
- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Support Vector Machines

Solution 2: map data to higher dimension

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



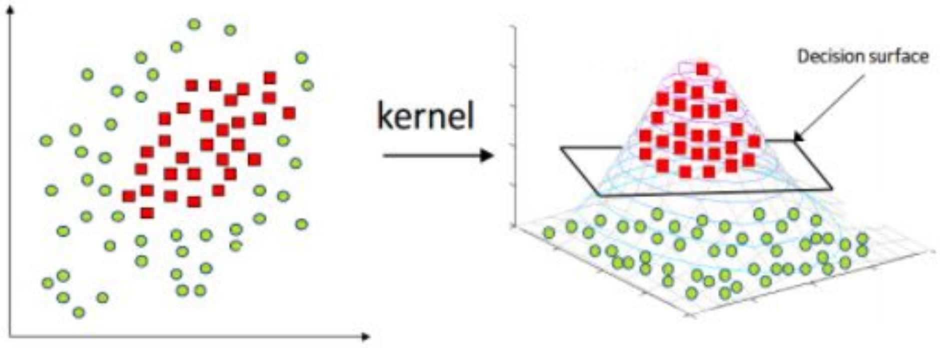
- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

Support Vector Machines – *Kernel Trick*

Kernel Trick

- Classifier can be learnt and applied without explicitly computing $\Phi(\mathbf{x})$
- All that is required is the kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^2$
- Complexity of learning depends on N (typically it is $O(N^3)$) not on D

It allows us to operate in the original feature space without computing the coordinates of the data in a higher dimensional space.

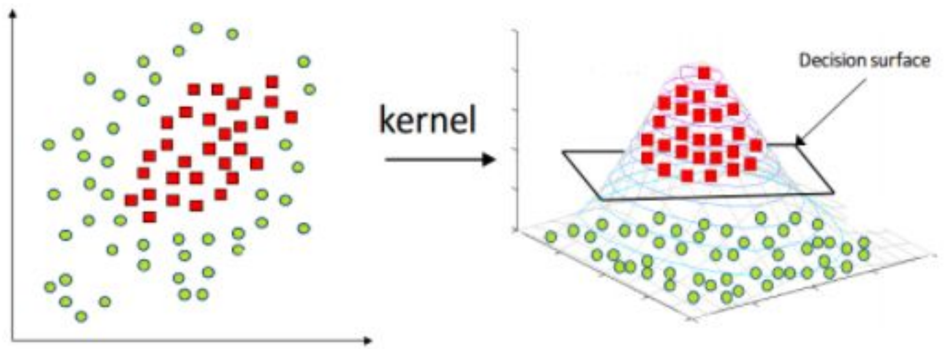


Kernel Definition:

A function that takes as its input vectors in the original space and returns dot product of vectors in the feature space is called a kernel function.

Support Vector Machines – *Kernel Trick*

For non-linear to linear SVM



Naive method:
- We find a kernel function for higher dimension
- Map all the data to higher space
- Find the separation plane
- ** Time-complexity may is too high

Kernel Trick:
- Without computing the coordinates of the data in higher space, simply compute inner products between images of pairs in feature space
- Computationally cheaper

Support Vector Machines

Example kernels

- Linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^\top \mathbf{x}'\right)^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$ for $\sigma > 0$
  - Infinite dimensional feature space

References

- https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f
- https://web.cs.hacettepe.edu.tr/~pinar/courses/VBM687/lectures/SVM.pdf
- https://www.cs.toronto.edu/~hinton/csc2515/notes/lec10svm.ppt