Support Vector Machine

Semester 1, 2021

Ling Luo

Outline

- Nearest Prototype Classification
- SVMs
  - Hyperplane
  - Margins
  - Classification
  - Non-linear SVMs
  - Multi-class SVMs
- Maths behind SVMs

Nearest Prototype Classification

- Calculate the *centroid of each class*, and classify each test instance according to the class of the centroid it is nearest to
- A parametric variant of nearest-neighbour classification
- The **centroid** is calculated simply by averaging the numeric values along each axis:

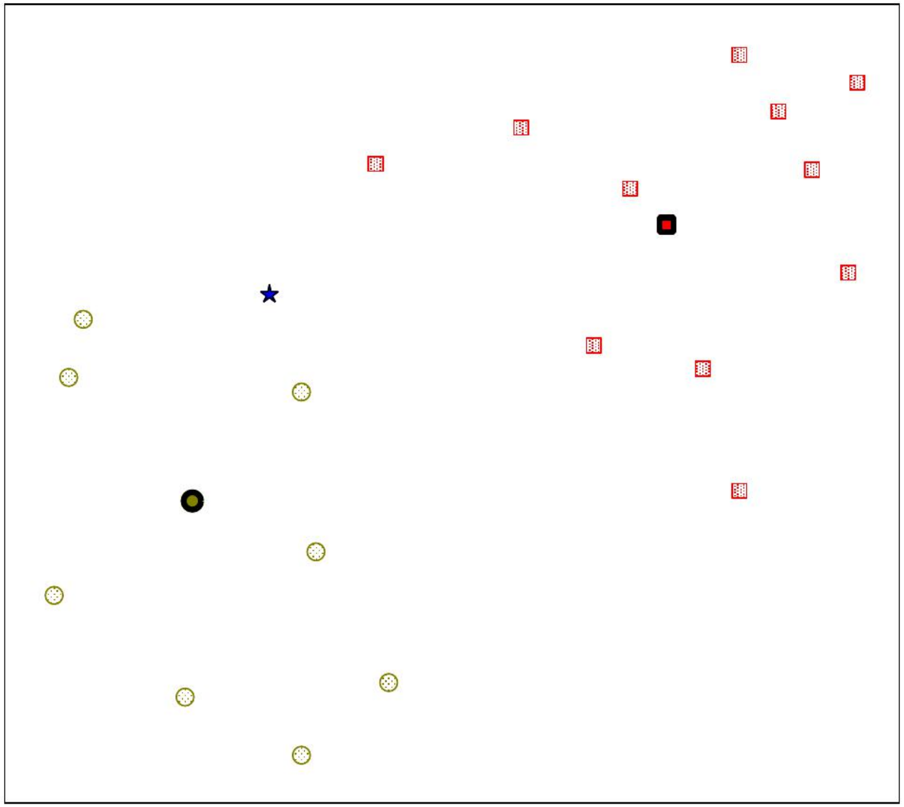
  For a class C_j with M instances $\{\boldsymbol{x}_i: [a_{i,1}, a_{i,2}, \dots, a_{i,D}]\}$

  Prototype $P_j = [a_1^*, a_2^*, \dots, a_D^*]$

  Each $a_k^* = \frac{\sum_{i=1}^{M} a_{i,k}}{M}$

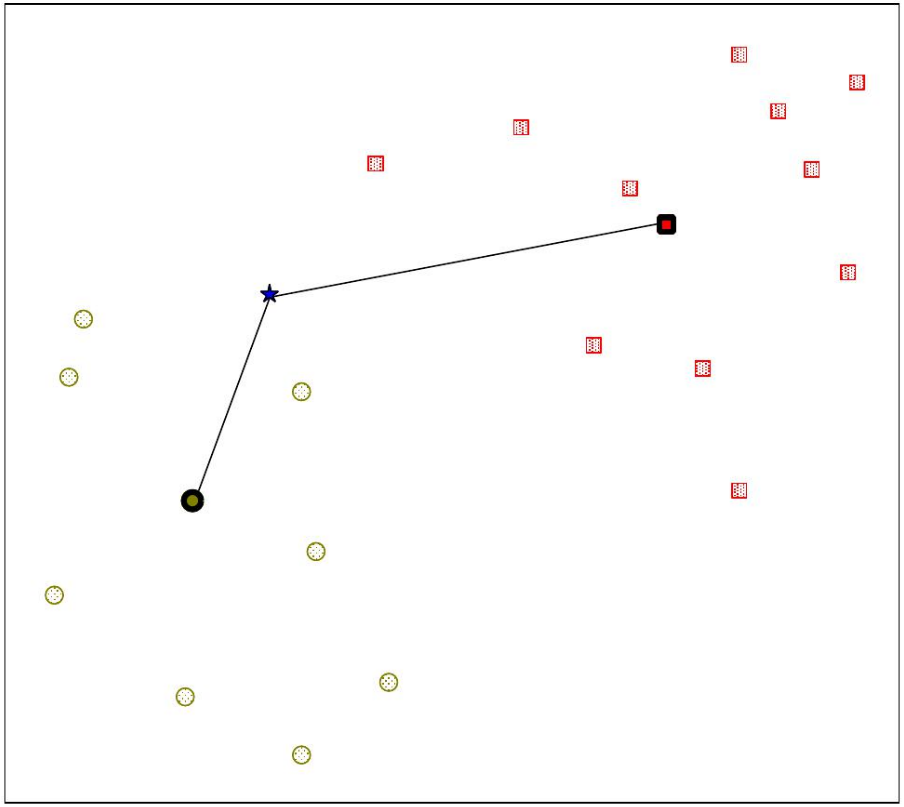
Nearest Prototype Classification

- Classification is based on simple Euclidean distance



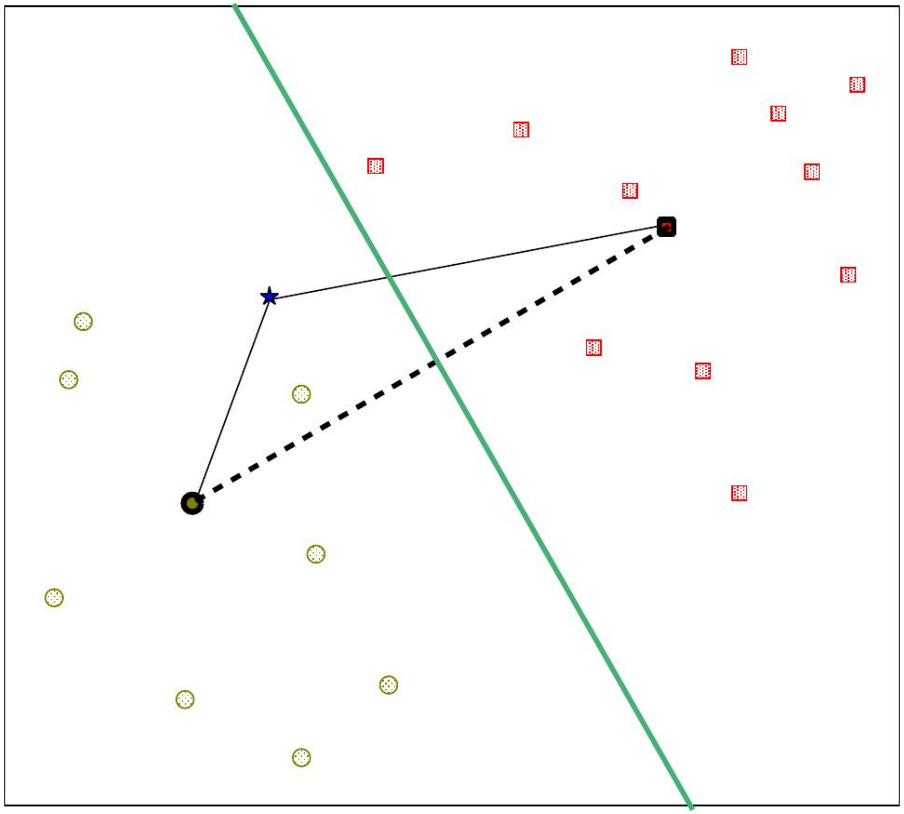
Nearest Prototype Classification

- Classification is based on simple Euclidean distance



Nearest Prototype Classification

- Classification is based on simple Euclidean distance



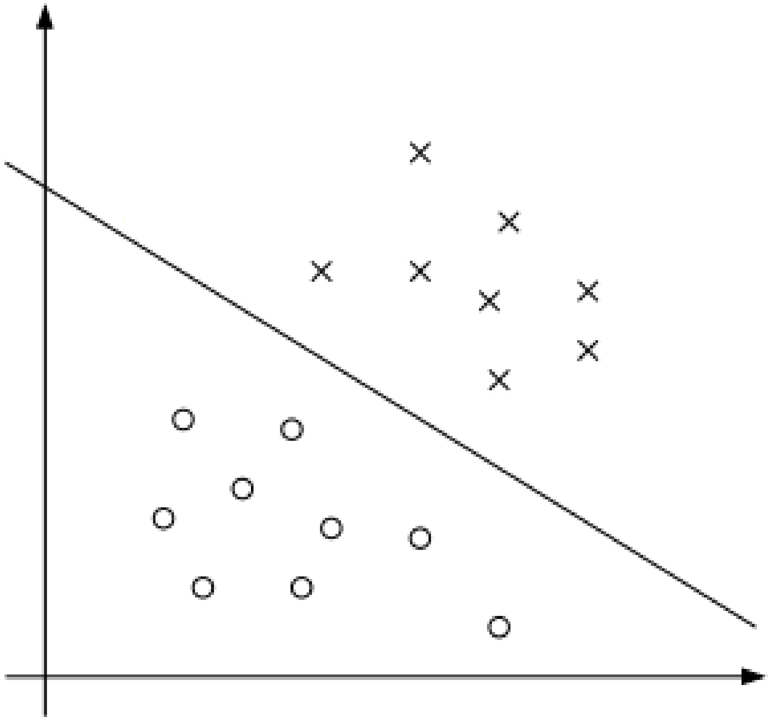
Forms a linear decision boundary

SVM

- Hyperplane
- Margins
- Classification
- Non-linear SVM
- Multi-class SVM

Support Vector Machine

- Goal: find a straight line/hyperplane that separates two classes

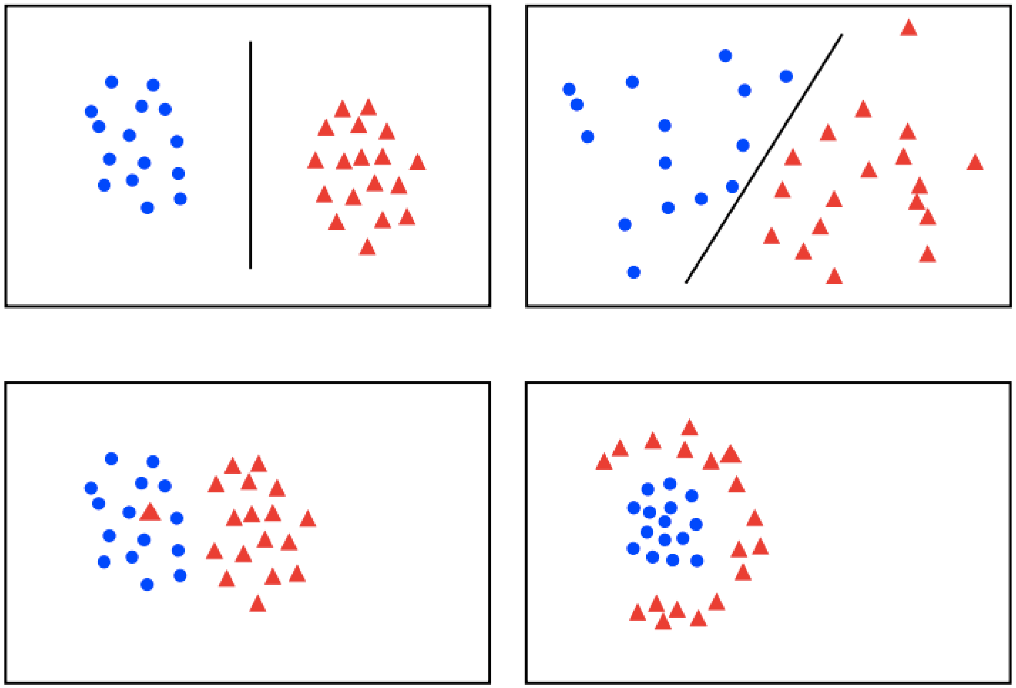


Linear Classifier

- Linear separability

linearly separable

not linearly separable



Linear Classifier

- A separating hyperplane in D dimensions can be defined by a **normal $\boldsymbol{w}$** and an **intercept** b

$$\boldsymbol{w} = [w_1, w_2, \dots, w_D]$$

- The hyperplane passing a point $\boldsymbol{x} = [x_1, x_2, \dots, x_D]$ is:

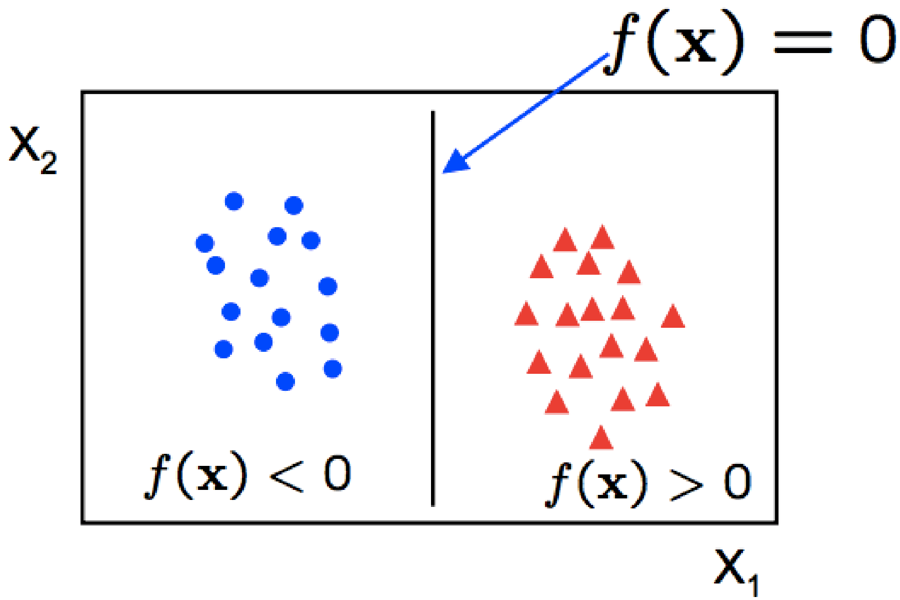
$$w_1 x_1 + \dots + w_D \, x_D + b = 0$$

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

$\boldsymbol{w}$ and $\boldsymbol{x}$ are column vectors, and the dot product $\boldsymbol{w} \cdot \boldsymbol{x}$ is often written as $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

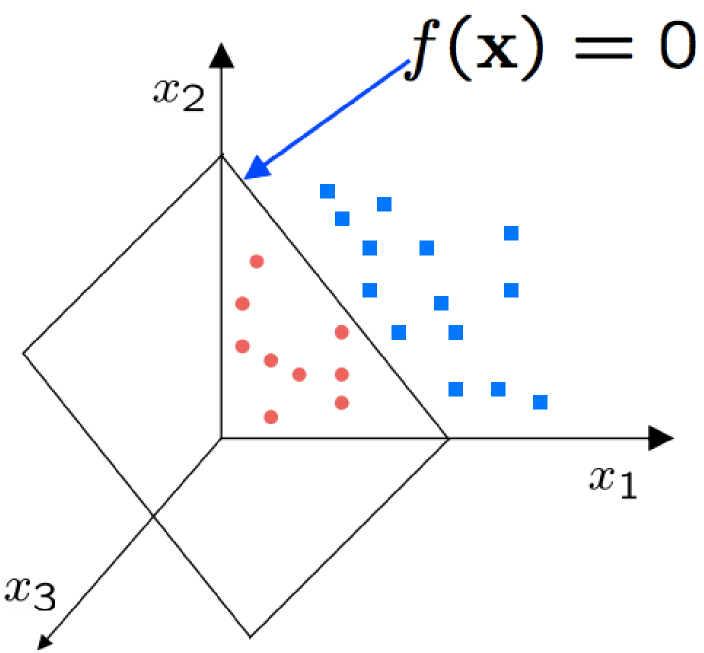
Linear Classifier

- Linear classifier takes the form $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$
- In 2D space, this is a straight line



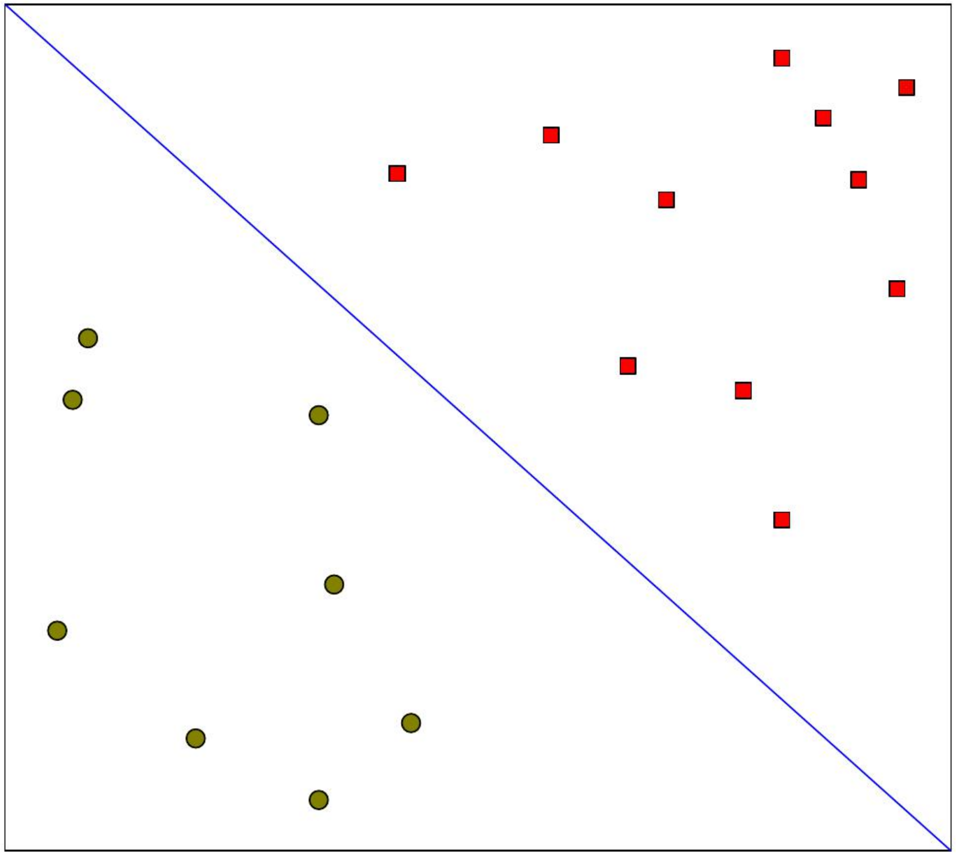
Linear Classifier

- Linear classifier takes the form $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$
- In 3D space, this is a plane



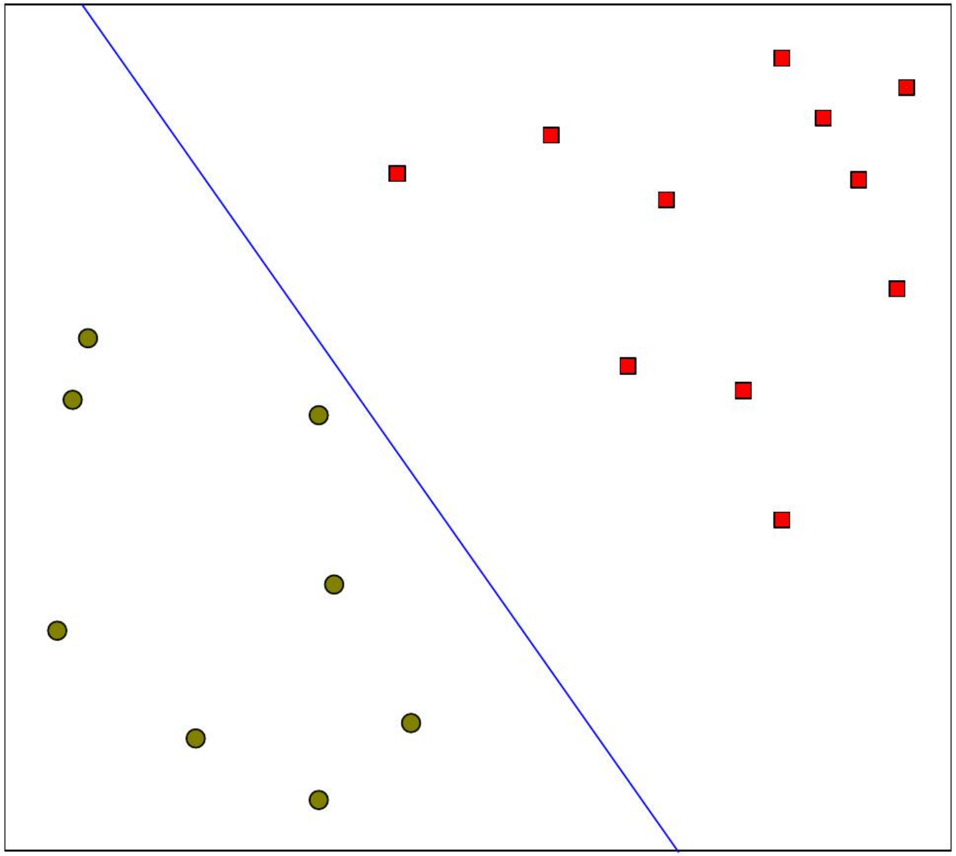
Hyperplanes of SVM

- One solution



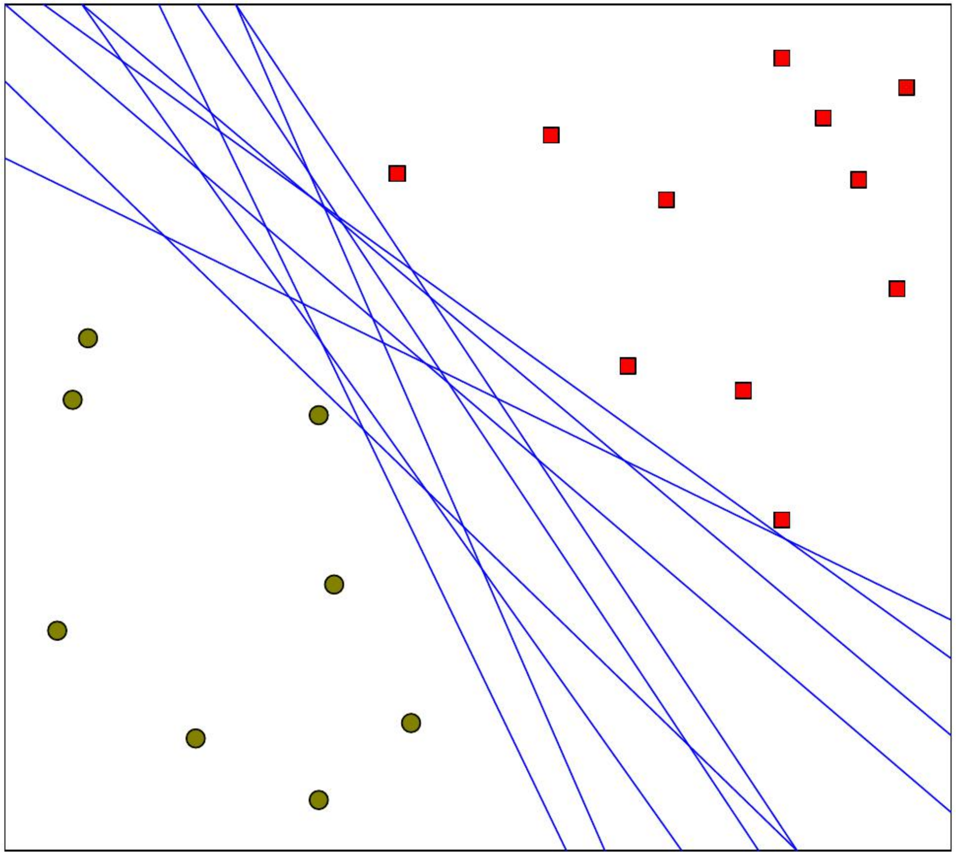
Hyperplanes of SVM

- Another solution



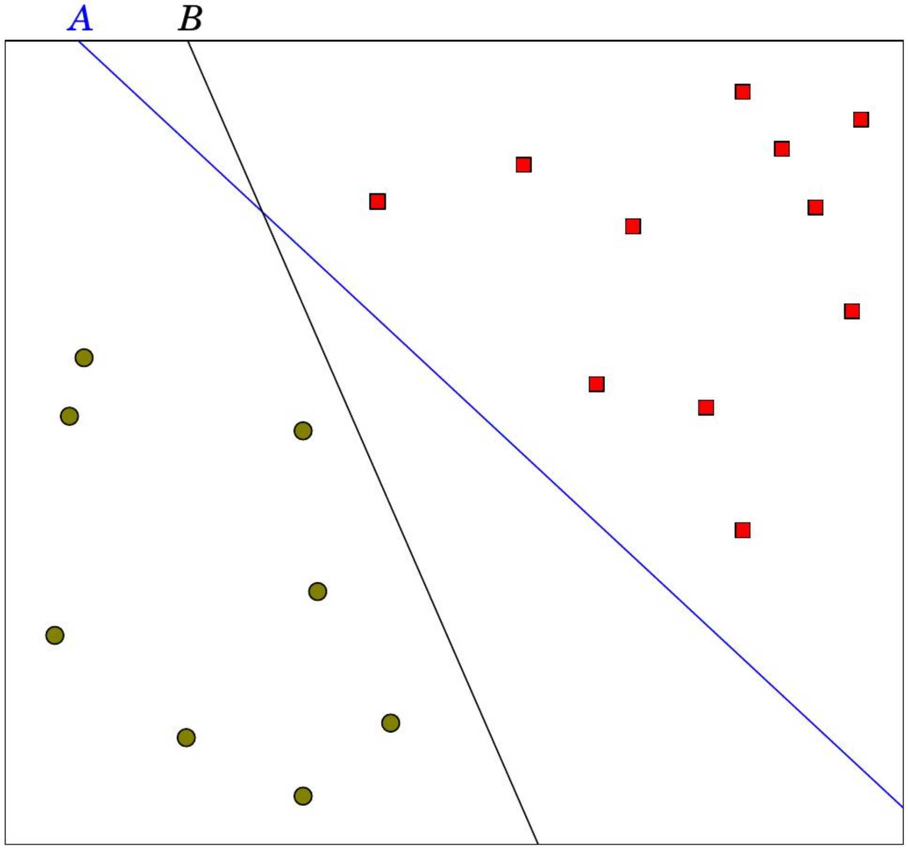
Hyperplanes of SVM

- More solutions



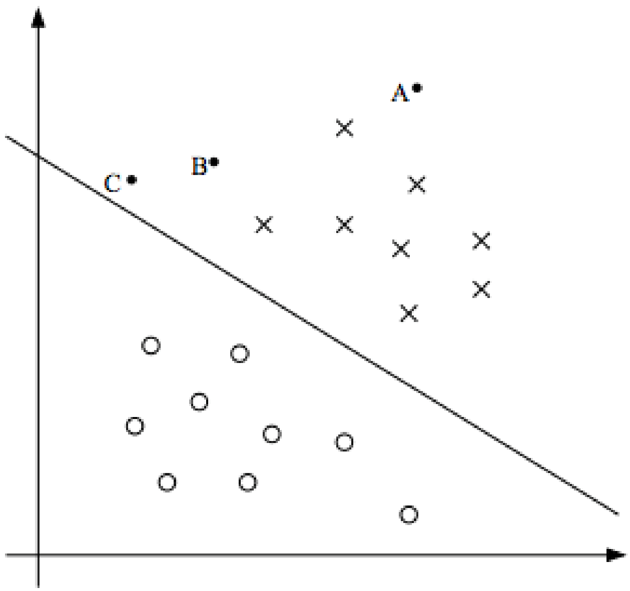
Hyperplanes of SVM

- How can we rate different decision boundaries?



Margins

- Consider the distance from a data point to the boundary



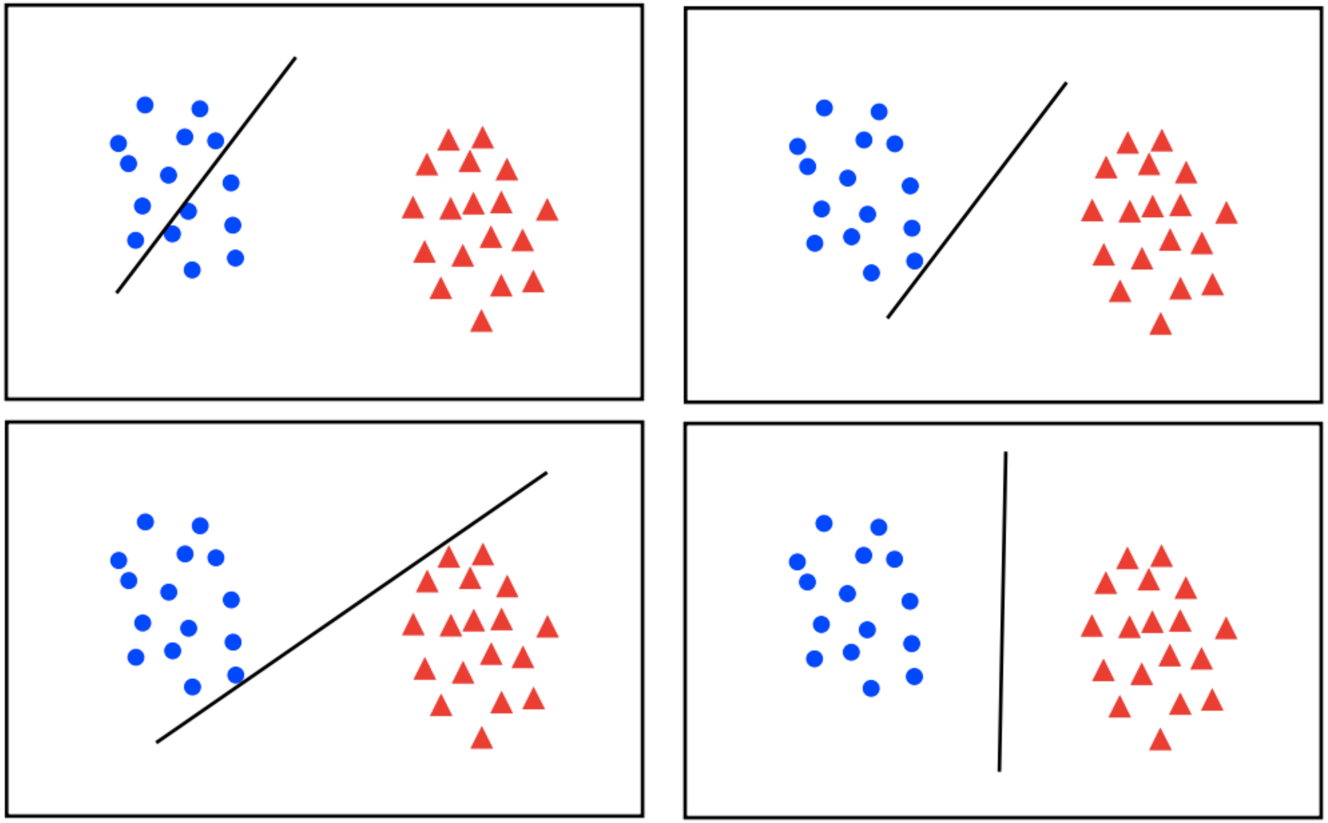
- For point A, we should be quite confident about the prediction of its class.
- For point C, a small change to the decision boundary might change our decision to change; we are less confident in the prediction.

Optimal Solution

- Aim: find a decision boundary that allows to make all correct and confident (far from the decision boundary) predictions for a given training set.
- SVM finds an optimal solution
  - *Maximises the distance* between the hyperplane and the difficult points close to decision boundary
  - Most stable under perturbations of the inputs

Optimal Solution

- What is the best hyperplane?



Classification using SVM

- **Task**: Associate one class as positive (+1), and one as negative (-1)
- **Build the model**: find the best hyperplane $\boldsymbol{w}$ and b, which maximises the margin between the positive and negative training instances

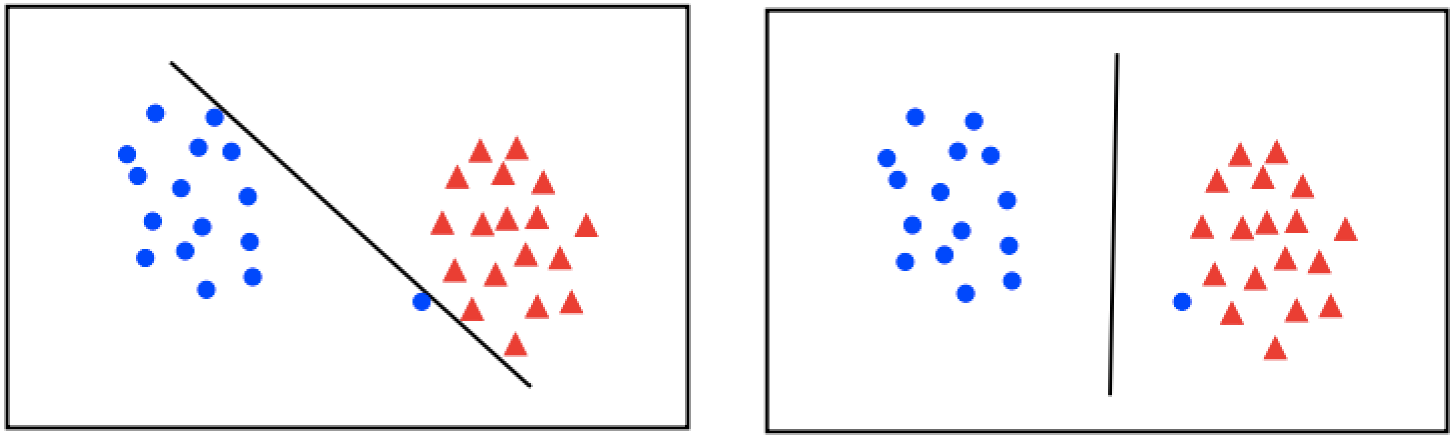
  How to learn $\boldsymbol{w}$ and b? Naïve approach for small training sets:

  1) Pick a plane $\boldsymbol{w}$ and b;
  2) Find the worst classified sample $(\boldsymbol{x}_i, y_i)$;
  3) Move plane (adjust $\boldsymbol{w}$ and b) to improve the classification of $(\boldsymbol{x}_i, y_i)$;
  4) Repeat steps 2-3 until the algorithm converges.

Classification using SVM

- **Make prediction** for a test instance $\boldsymbol{t} = [t_1, t_2, \ldots, t_D]$
  - Find the sign of $f(\boldsymbol{t}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{t} + b$
  - The value of $f(\boldsymbol{t})$ can be transformed into a probability, which shows the confidence.
  - Sometimes we assign "?" to instances within the margin
- **Comparison with KNN**
  - For a linear classifier SVM, the training data is used to learn the weight vector $\boldsymbol{w}$ and intercept b and mostly discarded.
  - For a KNN classifier, the model must memorise all training data

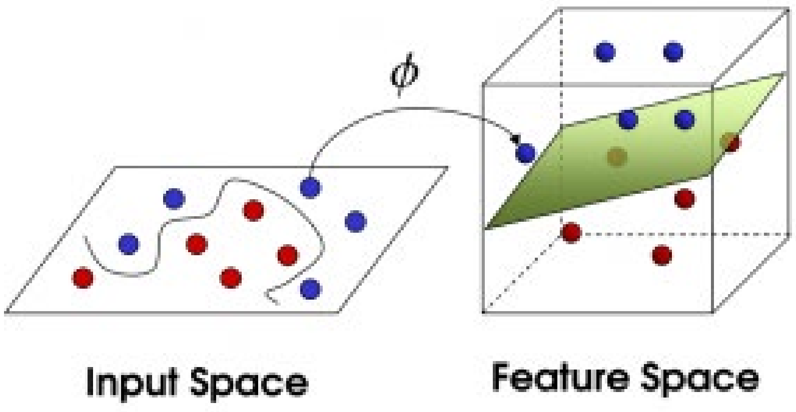
Non-linearly Separable: Soft Margins

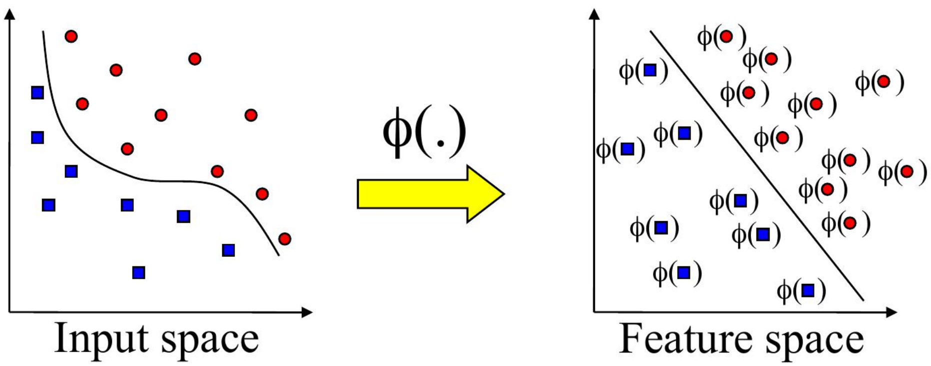


- Possibly large margin solution is better even though one constraint is violated
- Soft margins: trade-off between the margin and the number of mistakes on the training data

Non-linear SVM

- Make non-linearly separable problem separable
- Map data into better representation space





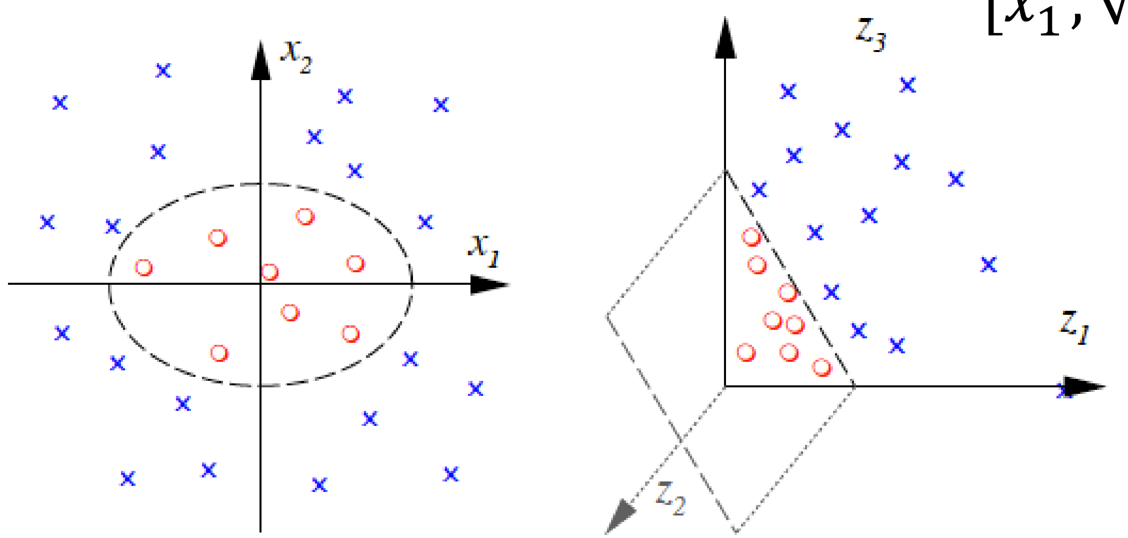
Non-linear SVM

- Solution: transform data by applying a mapping function, and then apply a linear classifier to the new feature vectors.

$$\phi:\ R^2 \rightarrow R^3$$

$$[x_1, x_2] \rightarrow [z_1, z_2, z_3]$$

$$[x_1^2, \sqrt{2}x_1x_2, x_2^2]$$



Multi-Class SVM

- Most common approaches for multiple classes is to convert to two-class problem.
  - **one-versus-all**: one classifier to separate one class from the rest of classes, choose the class which classifies test data point with greatest margin
  - **one-versus-one**: one classifier per pair of classes, choose the class selected by most classifiers
- Training time can be a serious issue, as we need to build many SVMs

Summary

- SVM is a linear hyperplane-based classifier for a two-class classification problem
- SVM selects the hyperplane with *maximum margin*
- *Soft margins* allow some data points to violate the separating hyperplane
- Non-linear SVM *transforms* data to a new feature space and finds a hyperplane separating two classes in the new space

Maths behind SVM

Specification of SVM

- Training set: N examples $(\boldsymbol{x}_1, y_1)$, $(\boldsymbol{x}_2, y_2)$, …, $(\boldsymbol{x}_N, y_N)$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ and $y_i \in \{-1,1\}$. Assume that two classes are linearly separable.
- The hyperplane separating two classes can be represented as:

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

and for training samples,

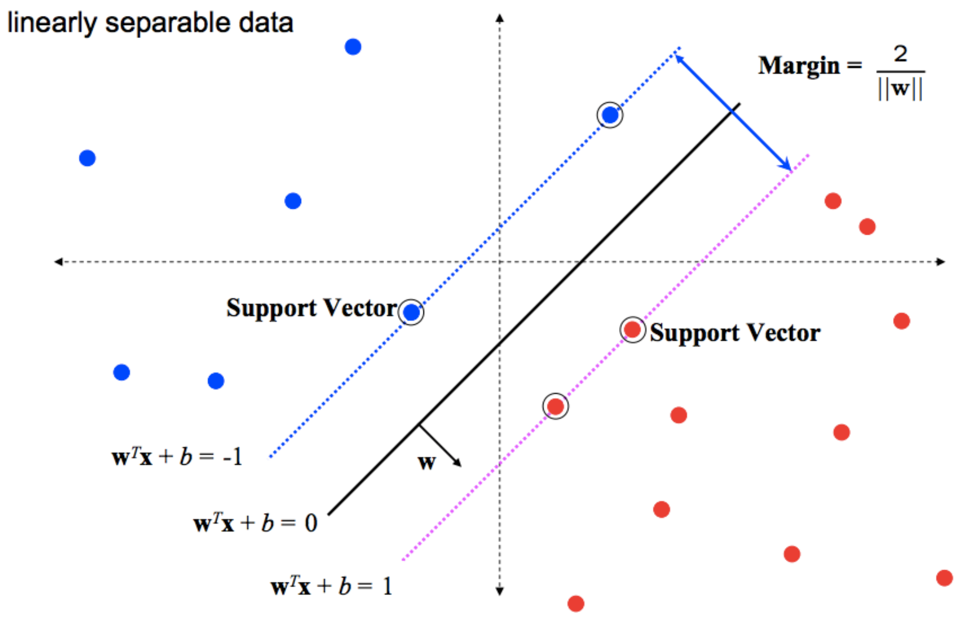
$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geq 1 \text{ for } y_i = +1$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leq -1 \text{ for } y_i = -1$$

$$\Rightarrow \quad y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1 \geq 0$$

Support Vectors

- Objective: find the data points that act as the boundaries of the two classes
- They constrain the margin between the two classes



Optimisation

- Optimisation problem: maximising the margin $\frac{2}{||w||}$
- Constraints: all points are on the correct side of the hyperplane

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 \geq 0$$

- Maximising $\frac{2}{||w||}$ is inconvenient, so we minimise

$$\frac{1}{2}||\boldsymbol{w}||^2 = \frac{1}{2}(w_1^2 + w_2^2 +, \ldots + w_D^2)$$

Optimisation

- Constrained optimisation problem

$$\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w}\|^2$$

$$\text{subject to } y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1 \geq 0, \forall\, i \in \{1,2,\ldots,N\}$$

- Determination of model parameters corresponds to a convex quadratic optimisation problem. Any local solution is also a global optimum.

Soft Margins

- Introduce *slack variables* $\{\xi_1, \xi_2,\ldots,\xi_i,\ldots,\xi_N\}$, which allows few points to be on the "wrong" side of the hyperplane at some cost
- New objective function with slack variables

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to } y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) + \xi_i - 1 \geq 0,$$

$$\xi_i \;\geq 0, \forall\, i \in \{1,2,\ldots,N\}$$

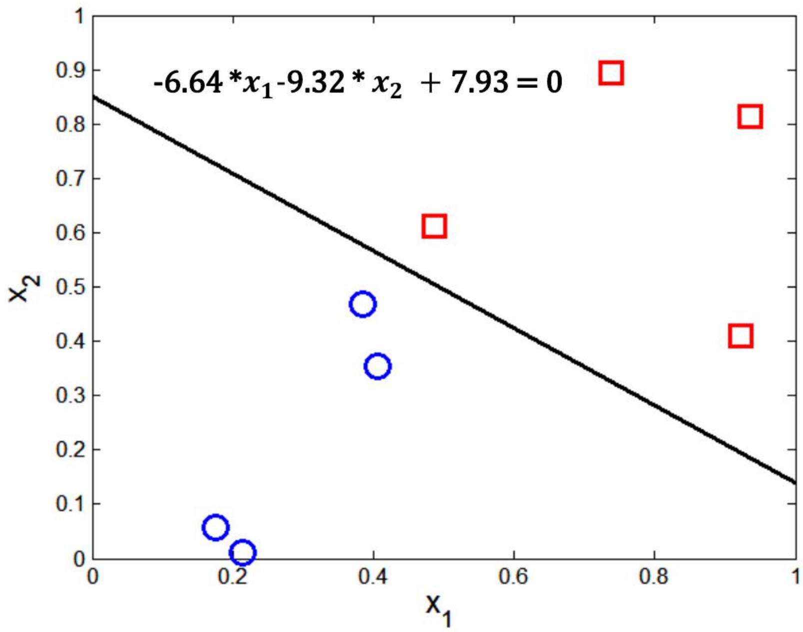
where C makes a trade-off between *maximising the margin* and *minimising the training error*, and it must be tuned.

Solving Optimisation Problem

- Solve constrained optimisation problem using *Lagrange multipliers*, where we introduce a value α_i for each constraint.
  - How many α_i do we need?
  - Solve α_i … the derivation is out of scope
  - Eventually, most α_i are 0, and the non-zero values correspond to support vectors

$$w_d = \sum_i^N \alpha_i y_i x_{id} \qquad b = \frac{1}{N_{sv}} \sum_{j \in N_{sv}} \frac{1 - y_j \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_j}{y_j}$$

Solving Optimisation Problem



Support vectors

x1	x2	y	α
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

$$w_1 = \sum_i \alpha_i y_i x_{i1} = 65.5261*1*0.3858 + 65.5261 * (-1)*0.4871 = -6.64$$

$$w_2 = \sum_i \alpha_i y_i x_{i2} = 65.5261*1*0.4687 + 65.5261 * (-1)*0.611 = -9.32$$

Source: Tan et al. (2018) Chapter 6.9

Classification using SVM

- After solving the optimisation problem, we get all α_i and can ignore training instances that are not support vectors
- Classify a new instance $\boldsymbol{t} = [t_1, t_2, \dots, t_D]$
  - Find the sign of $f(\boldsymbol{t}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{t} + b$

$$f(\boldsymbol{t}) = \sum_{i}^{N} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}}\, \boldsymbol{t} + b$$

  - If $f(\boldsymbol{t}) > 0$, class label is +1; else, class label is -1

Non-linear SVM

- Transform our dataset into a higher-order space
- For example, the *polynomial kernel of order 2*, ϕ_{P2}, transforms a vector of m dimensions into a vector of $C_m^2 + 2m + 1$ $(= \frac{m^2}{2} + \frac{3m}{2} + 1)$ dimensions

$$\boldsymbol{x}: [x_1, x_2, \dots, x_m]$$

$$\phi_{P2}(\boldsymbol{x}): [1, \sqrt{2}x_1, \sqrt{2}x_2, \dots, \sqrt{2}x_m, x_1^2, x_2^2, \dots, x_m^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \dots, \sqrt{2}x_{m-1}x_m]$$

Non-linear SVM

- In training, we need to calculate $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$ of all pairs of training instances when solving α_i
- Computation: $\mathcal{O}(DN^2)$ for D attributes and N training instances
- After transformation using ϕ_{P2} , there are $\mathcal{O}\left(D^2\right)$ attributes

- Solution: kernel trick!

Kernel Trick

- A kernel function acts on the *un-transformed vectors*, but calculates the *dot product of the transformed vectors*
- For example, given 2D vectors and a kernel function K_{P2}

$$\boldsymbol{x}_i : [x_{i1}, x_{i2}] \qquad \boldsymbol{x}_j : [x_{j1}, x_{j2}]$$

$$K_{P2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^2$$

$$\begin{aligned}
(1 + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^2 &= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}]^{\mathrm{T}} \\
&\quad\ [1, x_{j1}^2, \sqrt{2}x_{j1}x_{j2}, x_{j2}^2, \sqrt{2}x_{j1}, \sqrt{2}x_{j2}] \\
&= \phi_{P2}(\boldsymbol{x}_i)^{\mathrm{T}} \phi_{P2}(\boldsymbol{x}_j)
\end{aligned}$$

Kernel Trick

- Using the polynomial kernel function

$$K_{P2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^2$$

Computation:
1) The dot product between the two vectors
2) One extra addition
3) One extra exponentiation

- We get the dot product between the higher-order vectors

$$K_{P2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi_{P2}(\boldsymbol{x}_i)^{\mathrm{T}} \phi_{P2}(\boldsymbol{x}_j)$$

effectively skip the cost of transformation step, plus all of the extra calculations!

Common Kernel Functions

A kernel function K must be continuous, symmetric, and have a positive definite gram matrix

- Linear Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

- Polynomial Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)^d$$

- Radial Basis Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2})$$

Classify with Kernel Function

- In non-linear SVM, we replace our dot product with the corresponding kernel function
- Classify a new instance $\boldsymbol{t} = [t_1, t_2, \ldots, t_D]$

original $$f(\boldsymbol{t}) = \sum_{i}^{N} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{t} + b$$

$$\Rightarrow \quad f(\boldsymbol{t}) = \sum_{i}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{t}) + b$$

  - If $f(\boldsymbol{t}) > 0$, class label is +1; else, class label is -1

Summary and Resources

- Learning SVM models means finding the best separating hyperplanes
- Classification of new instances is efficient
- SVMs can be applied to non-linearly-separable data with an appropriate kernel function

- Resources
  - Tan et al. Introduction to Data Mining (2018, 2nd edition). Section 6.9
  - http://nlp.stanford.edu/IR-book/pdf/15svm.pdf
  - https://www.youtube.com/watch?v=_PwhiWxHK8o
  - http://research.microsoft.com/pubs/67119/svmtutorial.pdf