Support Vector Machines (SVM)

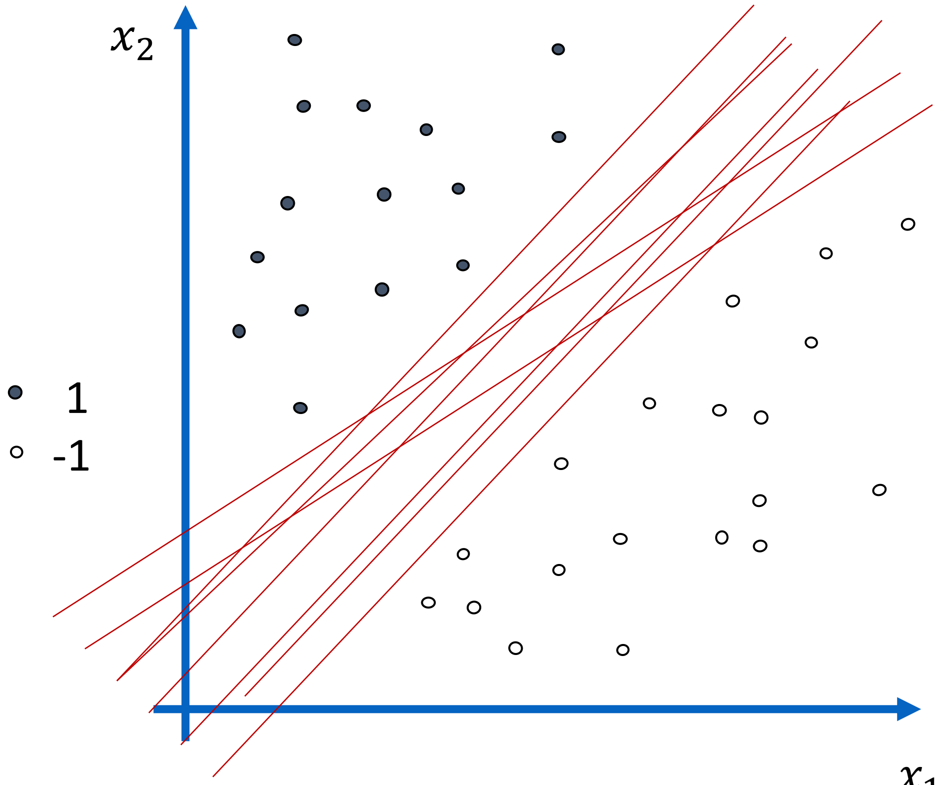
Data Analytics Lab – Week 9

Department of Electrical Engineering
Indian Institute of Technology, Madras

Outline

1. Linear Classifiers
2. Max Margin Classifiers
3. SVMs
4. Linear SVM Classifier
5. Non-Linear SVM Classifier
6. Kernel Trick
7. Kernel Functions

Linear Classifiers

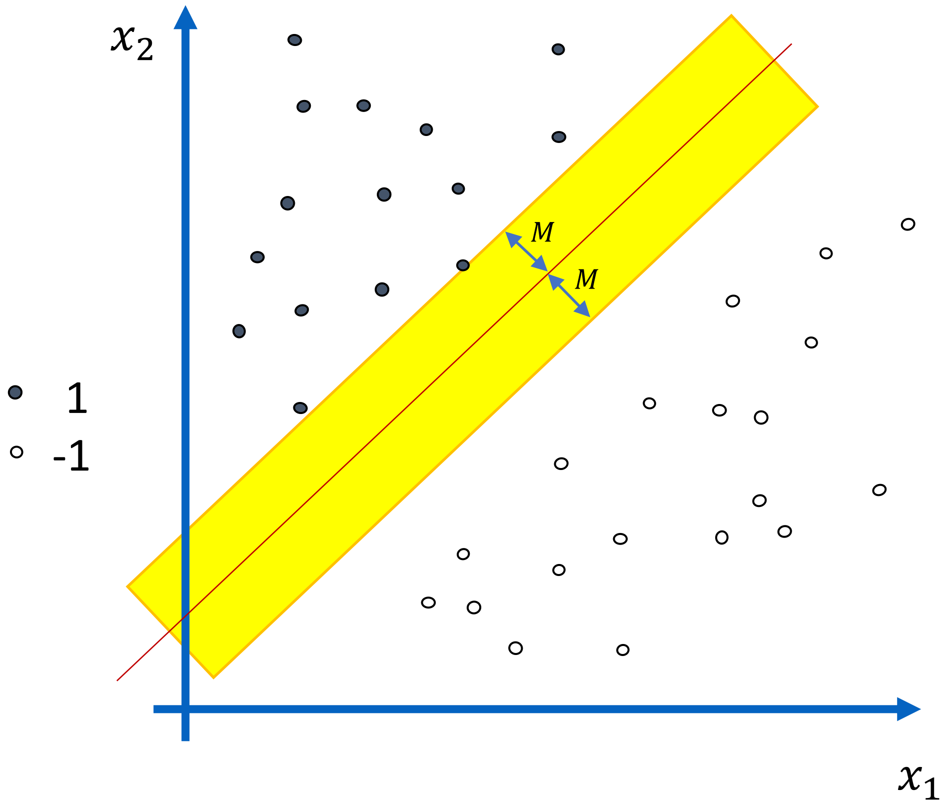


$$y = f(\boldsymbol{x}, \boldsymbol{\theta})$$

How to classify this data using a hyperplane ?

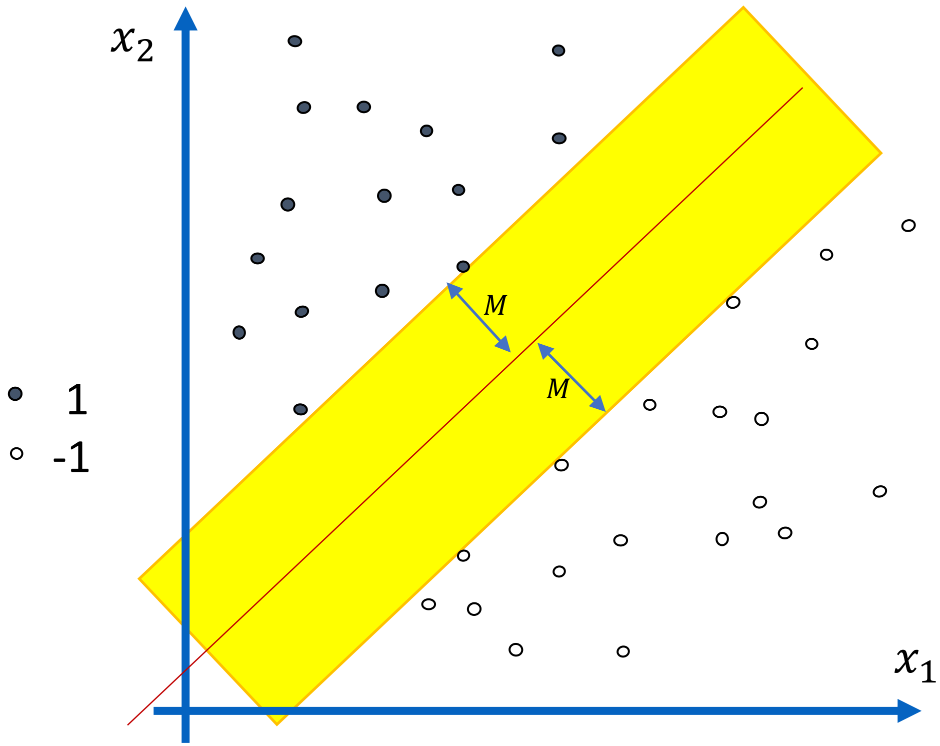
Which of these decision boundaries (hyperplanes) is best ?

Margin of a Linear Classifier



- Margin (M): Perpendicular distance between the separating hyperplane and the closest data point
- Equal margin is considered on both sides of the hyperplane

Max Margin Classifier



- In a max margin classifier, the best separating hyperplane is the one that separates the classes with maximum margin
- Max margin not only ensures good classification but also ensures good generalisation
- Data points which determine the max margin are referred to as support vectors

What if the data is not perfectly separable ?

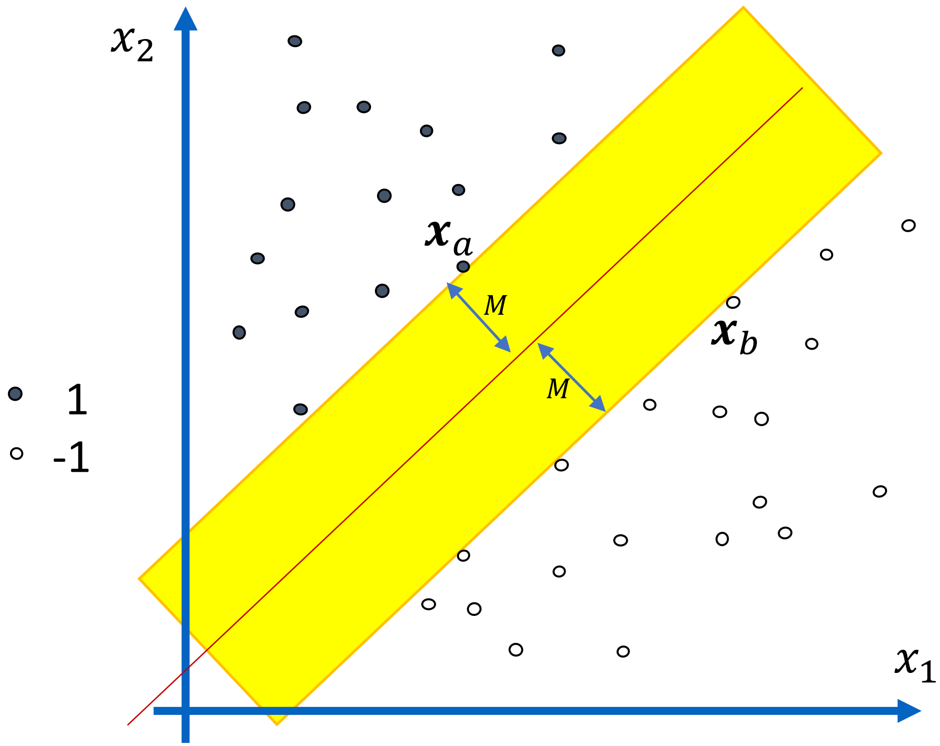
Support Vector Machines

- Support Vector Machines work on the max margin principle and are a generalisation of max margin classifiers
- Developed by Vapnik and others in 1992
- SVMs can be even used to build classifiers to classify data which are not perfectly separable (soft margin)
- SVMs can be used to build both linear as well as non-linear classifiers by using the kernel trick (discussed later)

- SVMs can also be used for regression tasks – Support Vector Regression

  (Refer to this link for more details on Support Vector Regression: https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2)

Linear Support Vector Classifier – Separable Case



Eq. of separating hyperplane:

$$\boldsymbol{\theta}^T \boldsymbol{x} + \boldsymbol{\theta_0} = \mathbf{0}$$

Assume that support vectors are at a distance of 1 from the separating hyperplane

i.e., $\boldsymbol{\theta}^T \boldsymbol{x_a} + \boldsymbol{\theta_0} = \mathbf{1}$

$\boldsymbol{\theta}^T \boldsymbol{x_b} + \boldsymbol{\theta_0} = -\mathbf{1}$

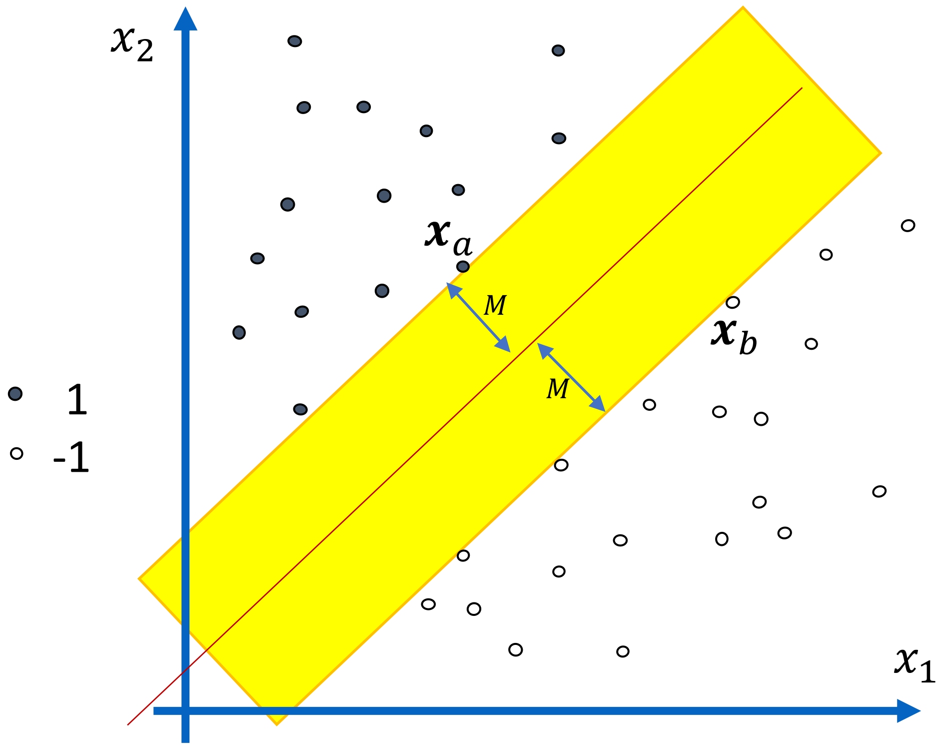
Margin: $2M = \|x_a - x_b\|_2 = \frac{2}{\|\boldsymbol{\theta}\|_2}$

Optimization formulation: $\max\limits_{\boldsymbol{\theta},\theta_0} \frac{2}{\|\theta\|_2}$ s.t. $\boldsymbol{\theta}^{\boldsymbol{T}} x_i + \theta_0 \geq 1 \;\; if \; y_i = 1$

$\boldsymbol{\theta}^{\boldsymbol{T}} x_i + \theta_0 \geq 1 \;\; if \; y_i = -1$

Alternately, $\|\theta\|$ can be minimised

Linear SVC – Separable Case



- Generally, the dual problem of the minimization problem is solved
- Solution is given by:

$$\boldsymbol{\theta} = \sum \alpha_i y_i \boldsymbol{x_i}$$

$$\theta_0 = y_k - \boldsymbol{\theta}^{\boldsymbol{T}} \boldsymbol{x}_k$$

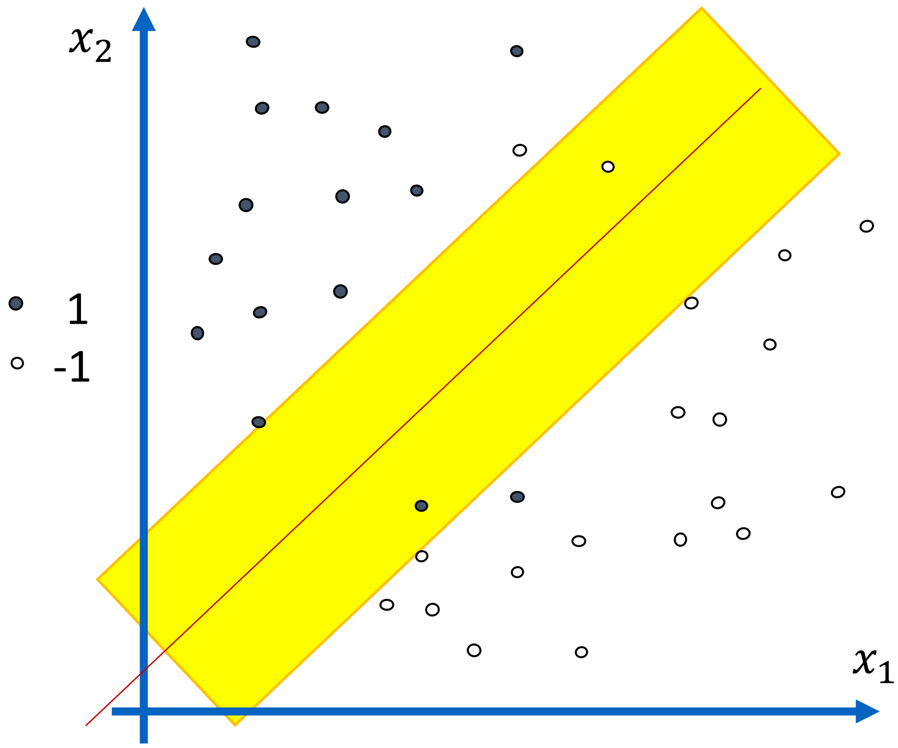
α_i: Langrange multipliers which are non-zero for Support vectors
k: Any support vector

Classifier: $f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta} = \sum \alpha_i y_i \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{T}} \boldsymbol{x} + b$

$\boldsymbol{x}$: Test point to be classified

It relies on inner product between $\boldsymbol{x}_i$ and $\boldsymbol{x}$

Linear SVC – Non Separable Case



- In this case, some mis-classification is allowed
- The error is measured by introducing slack variables into the optimization cost function
- A tunable parameter is also introduced to control the mis-classification error to be allowed

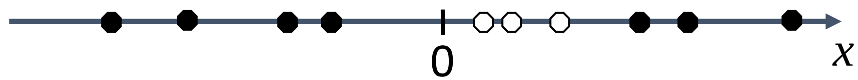
$$\min_{\boldsymbol{\theta},\theta_0} \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + C\sum \xi_i$$

$$\text{s.t } y_i\left(\boldsymbol{\theta}^T x_i + \theta_0\right) \geq 1 - \xi_i$$

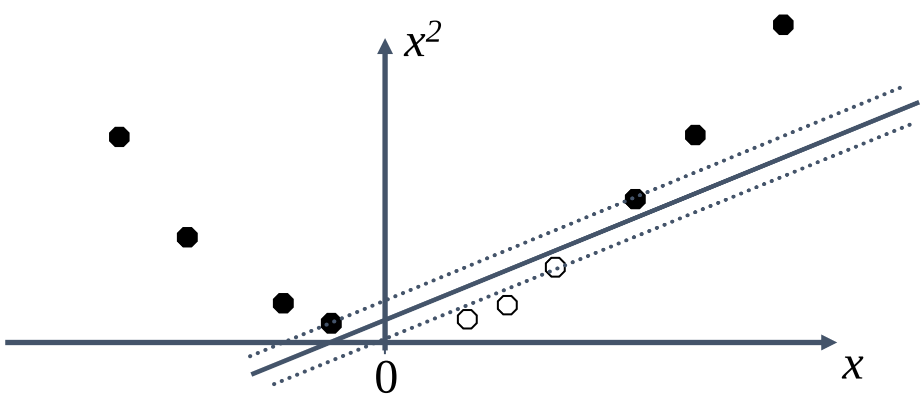
$$\xi_i \geq 0$$

Non-Linear SVC

- Consider 1D data shown in fig:



- No line can separate the two classes of points
- How about mapping them to a higher dimension space ?
- Say $x_1 = x\,; x_2 = x^2$
- In general, original features can be mapped to a higher dimension space : $\boldsymbol{x} \to \phi(\boldsymbol{x})$ where they are linearly separable



Now they are linearly separable

Kernel Trick

- Transforming data to higher dimensional space and solving the optimization problem with transformed features is complicated and very expensive
- Instead we use something called '**kernel trick**'
- Recall that inner product between pairs of data points is what is required to solve the optimization problem
- **Kernel is a function** which takes two data points as input and gives their inner product after they are transformed into a high-dimensional space

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >$$

- So kernel trick refers to an efficient and less expensive way to transform data into higher dimensions

Kernel Example

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >$$

- Consider two points from a dataset with 2 features:

$$\boldsymbol{x_i} = [x_{i1} \quad x_{i2}]$$
$$\boldsymbol{x_j} = [x_{j1} \quad x_{j2}]$$

- They are transformed to a 3-dimensional feature space as follows:

$$\phi(\boldsymbol{x_i}) = [x_{i1}^2 \quad \sqrt{2}x_{i1}x_{i2} \quad x_{i2}^2]$$
$$\phi(\boldsymbol{x_j}) = [x_{j1}^2 \quad \sqrt{2}x_{j1}x_{j2} \quad x_{j2}^2]$$

- Their dot product is given by:

$$x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2 = (\boldsymbol{x}_i^T \boldsymbol{x}_j)^2$$

- Therefore, kernel $k(\boldsymbol{x}_j \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j)^2$ directly gives the dot product of transformed features in 3-D space

Different Types of Kernels

- Some of the most popular kernels used in SVMs are:
  1. Linear Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
  2. Polynomial Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^{\mathrm{d}}$
  3. Gaussian Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{\frac{-\| x_i - x_j \|^2}{2\sigma}}$
  4. RBF Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2}; \gamma > 0$
  5. Sigmoid Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(a x_i^T x_j + b)$

Selection of Kernels

- There are no exact rules for selection of kernels
- Finding the best kernel for SVM is mostly done empirically by trial and error
- If the data is known to have linear relations, then linear SVM works best
- For non-linear SVM, Gaussian or RBF kernels are most popular
- Also every kernel has certain parameters which can be tuned to obtain a good SVM classifier
- Nevertheless, the selection of a kernel and its parameters depends on the distribution of data