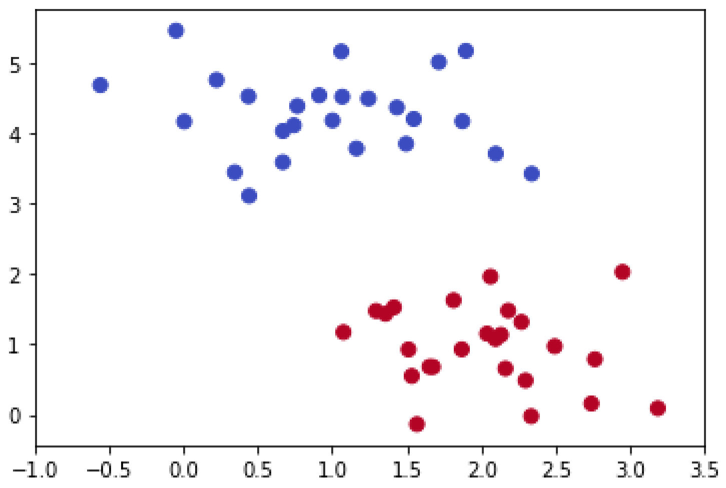
UNIVERSITY of SOUTH FLORIDA

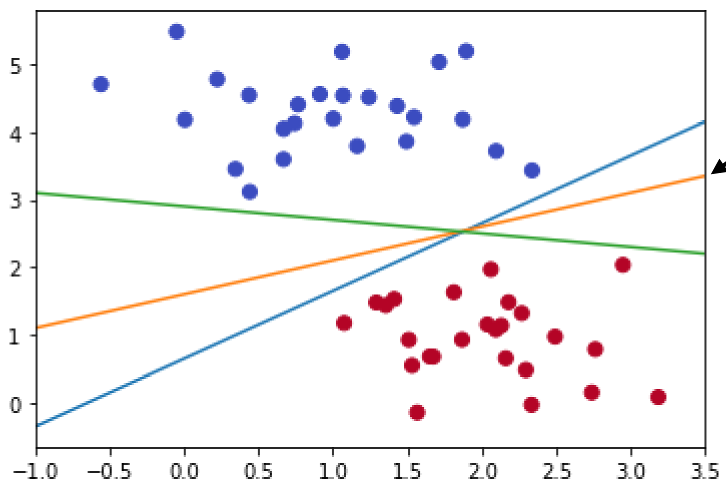
# Nonlinear Models

**Spring 2020**
**Instructor: Ankit Shah, Ph.D.**

## Classifiers with Linear Decision Boundaries

# Maximal Margin Classifier





Decision boundary

Optimal separating hyperplane:

$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0$

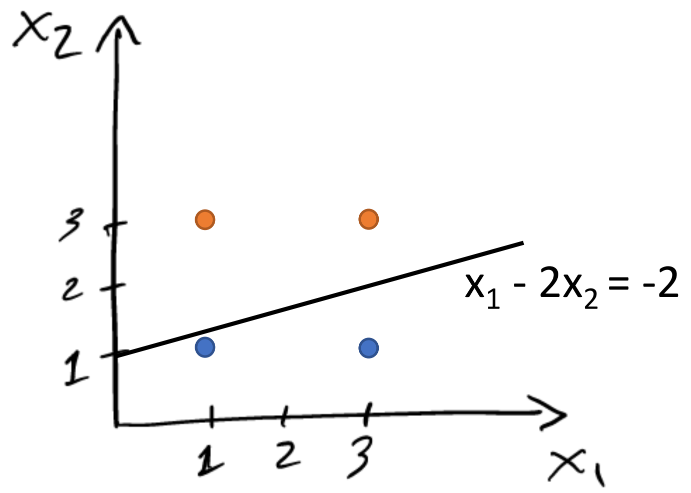
One that is the farthest away from any of the samples in the training data set

Used when classes can be separated well by a linear boundary
i.e., the classes can be separated by a (p-1)-dimensional hyperplane

*Code to obtain the above figure in Python:*

```
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=50, centers=2,random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='coolwarm')
```

# Maximal Margin Classifier



- Say, we have the following data points ($x_1$, $x_2$) in the training data:
  (1,3), (3,1), (3,3), (1,1)
  Classify them in 2 different classes: -1 and +1
- *Hyperplane:* $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$
  $2 + x_1 - 2x_2 = 0$
- (1,3) -> -ve value -> class -1
- (3,1) -> +ve value -> class +1
- (3,3) -> -ve value -> class -1
- (1,1) -> +ve value -> class +1

- If vector X satisfies $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p > 0$, then it belongs to one side of the p-dimensional space: y = 1
- If vector X satisfies $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p < 0$, then it belongs to another side of the p-dimensional space: y = -1
- Generalizing the above: a separating hyperplane has the following property
  $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$ for (i = 1, …, n)

# Classification using a Separating Hyperplane

- If a separating hyperplane exists, then there will be an infinite collection of them
- We can adjust the hyperplane by a little and still separate the classes
- We multiply the $\beta_j$ by a non-zero constant such that the equation still holds true: $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$

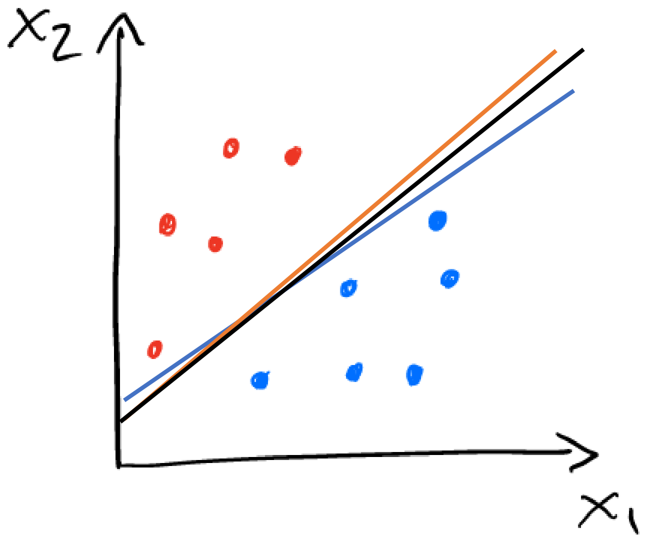
Objective:
- We want to construct a classifier that optimally separates the classes such that the separating hyperplane is the farthest distance from the training observations

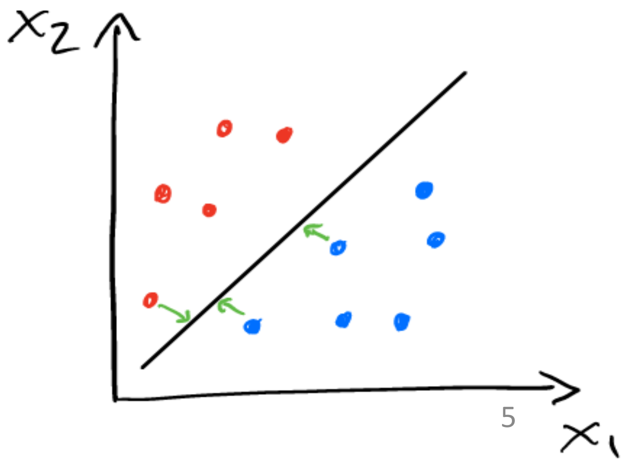
How do we do that?:
  - We compute the perpendicular distance from each training data point to the given separating hyperplane
  - The smallest such distance is the minimal distance from the observations to the hyperplane – Margin
  - Our objective is to maximize the margin  Max. min. distance
- Mathematical Formulation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

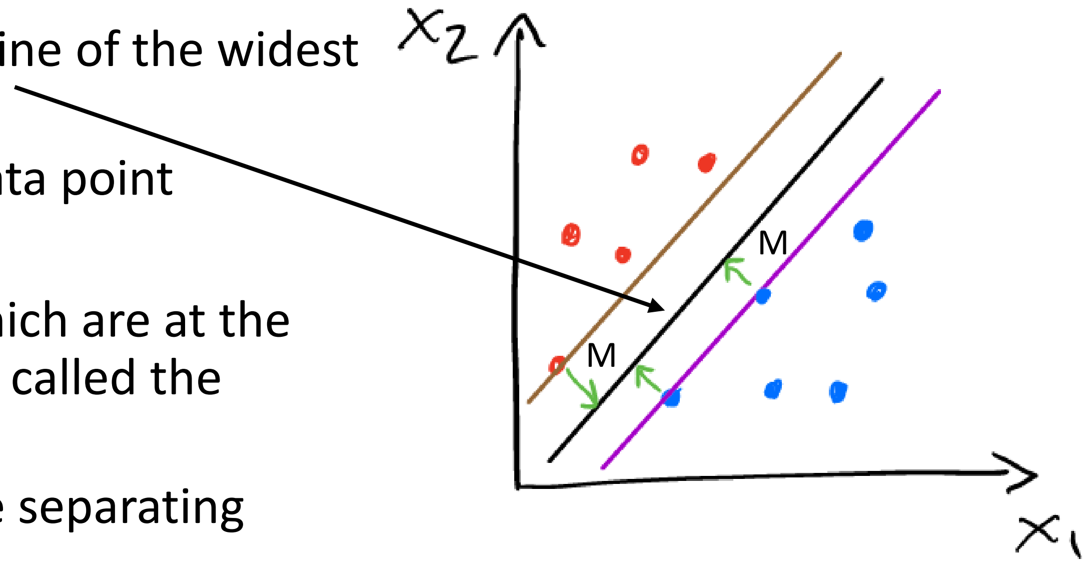
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M$$



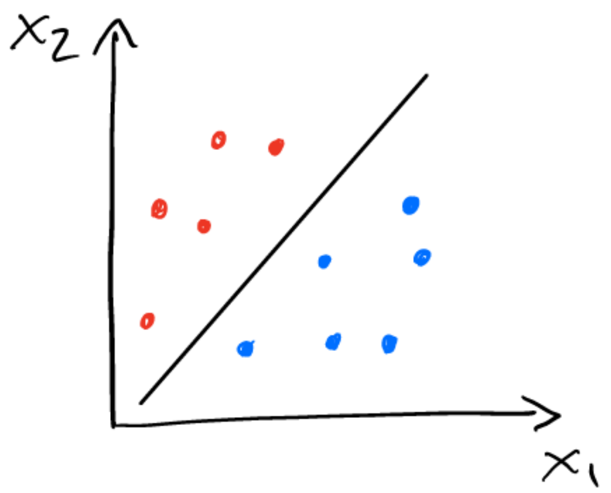


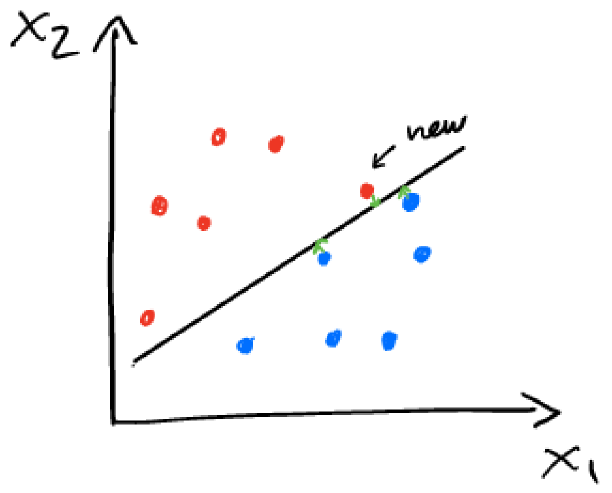
# Classification using a Separating Hyperplane

- Think of the maximal margin hyperplane as the mid-line of the widest "slab" that we can insert between two classes
- The value M (smallest distance between a training data point and the hyperplane) is called the margin
- Observations (data points in the training data set) which are at the exact distance M from the separating hyperplane are called the support vectors
- Each data point is at least at a distance of M from the separating hyperplane



## Maximal Margin Classifier

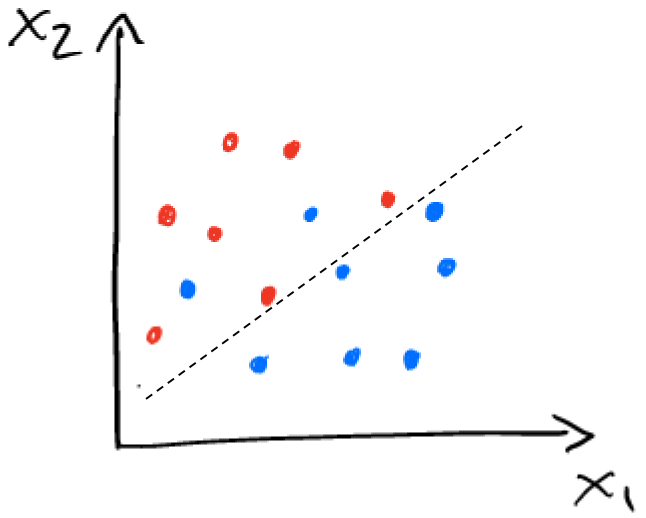


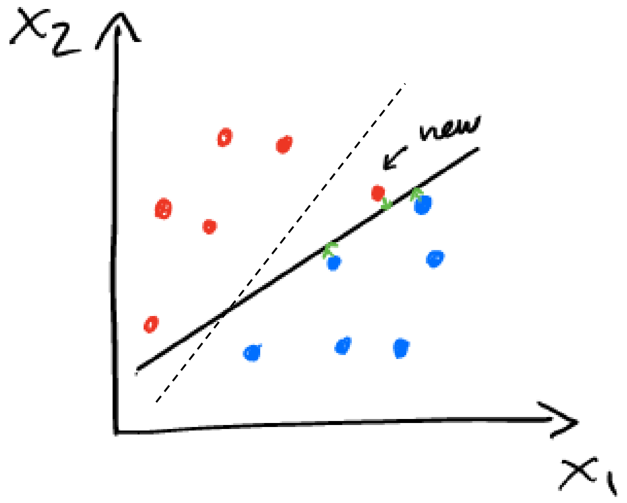


- A small change in the data set can result in a fairly large change in the decision boundary of the maximal margin classifier
- Such sensitivity suggests that the maximal margin classifier overfits the training data
  - i.e., the model is prone to a higher misclassification rate on the testing (unseen) data set
- This motivates a need for a classifier that can be used to produce a more appealing decision boundary

# Support Vector Classifier

- Support vector classifier is also called the “soft” margin classifier
- Used when the classes of the training data cannot be perfectly separated by a hyperplane and also when a more appealing decision boundary is needed in which separation is possible

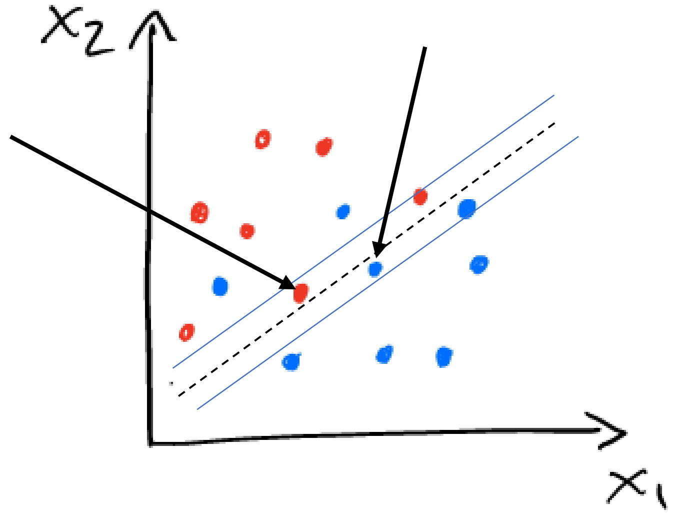




- In the above cases, we may consider a hyperplane that does not separate the 2 classes perfectly
- Our objective is to provide 1) robustness to the model and 2) better classification for majority of the data points

# Support Vector Classifier

- As shown in the previous example, we allowed some data points to be on the incorrect side of the hyperplane and also closer to the hyperplane on the correct side but with a value less than the margin
- Hence, we call it a soft margin
    - i.e., it can be violated by some data points to obtain a more robust model



- The support vector classifier has a (p-1)-dimensional hyperplane for the decision boundary and has a soft margin that could be violated

# Support Vector Classifier

- Mathematical Formulation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

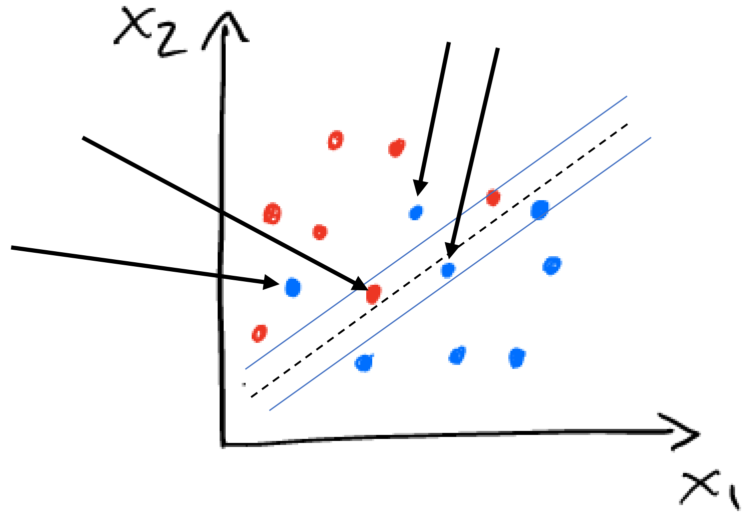
$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

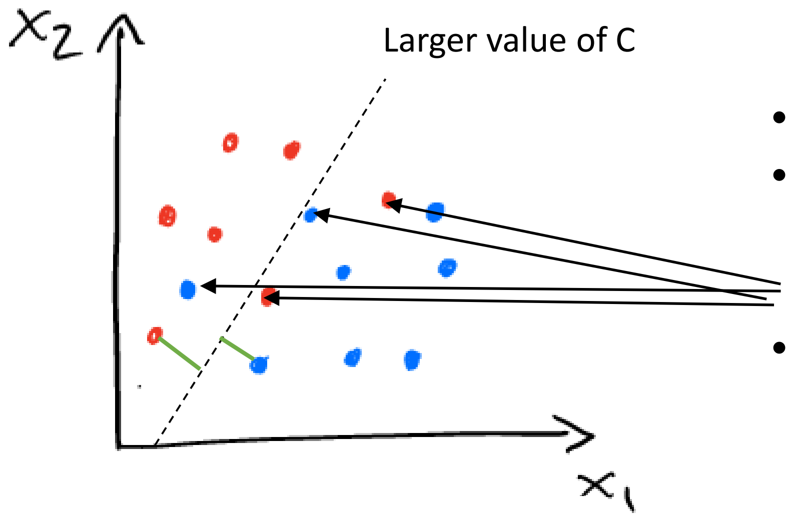
Slack variable

Tuning parameter

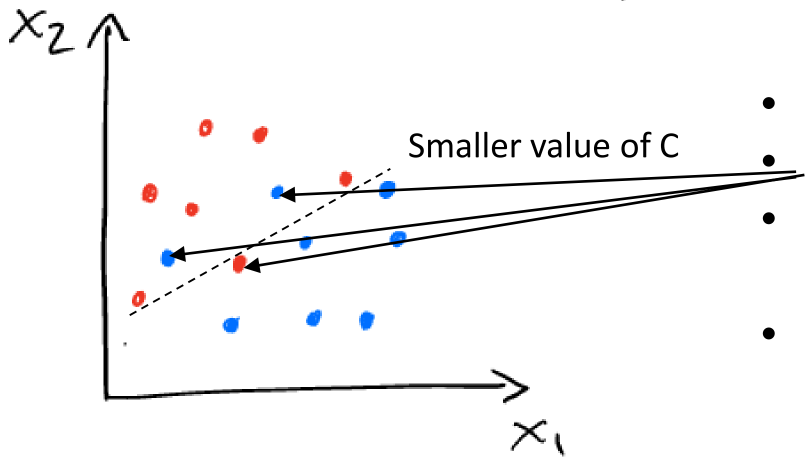


- $\varepsilon_i$ allows the data point *i* to be on the wrong side of the margin or the hyperplane
- C determines the number and severity of the violations to the margin and the hyperplane
- Support vectors: All data points with $\varepsilon_i > 0$

# Relationship of C with M



- Larger value of C -> larger value of M
- More slack variables come into play and hence more violators (support vectors)
- This is a condition with
  Low variance – High bias



- Smaller value of C -> smaller value of M
- Less number of support vectors
- This is a condition where we are chasing the data points
- Overfit -> Low bias

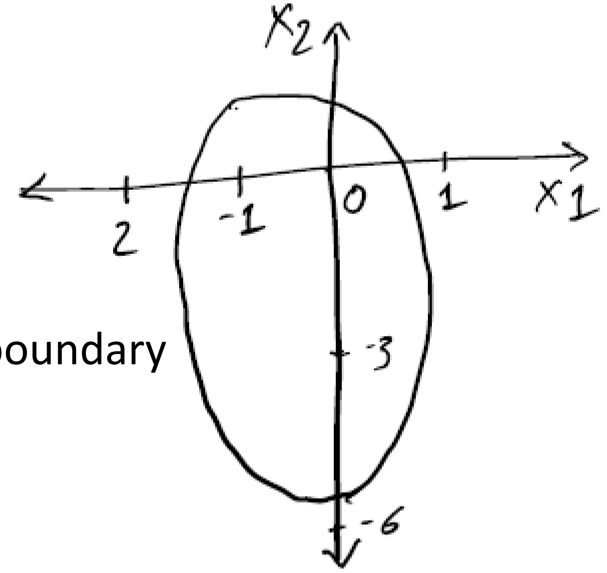
## Support Vector Machines

# Support Vector Machines

- Suppose we have 2 predictor variables $x_1$and $x_2$ and we add 2 more predictor variables $x_3 = x_1^2$ and $x_4 = x_2^2$
- Next, we want to apply the support vector classifier and we obtain the decision boundary that satisfies the condition:
- $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_\mathrm{p} x_p = 0$

$\big\downarrow$ Obtained after fitting on a training data set

- -0.13 + 0.7 $x_1$ + 0.5 $x_2$ + 0.5 $x_1^2$ + 0.1 $x_2^2$ = 0 Note that $\sum_j \beta_j^2 = 1$



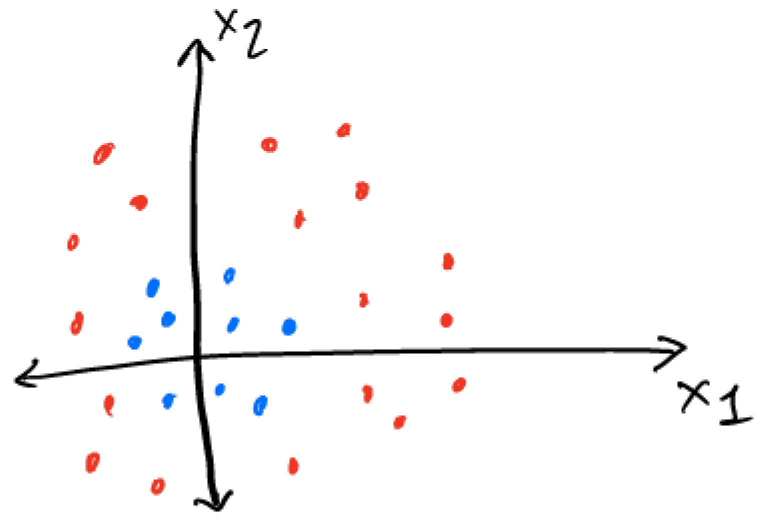
Elliptical decision boundary

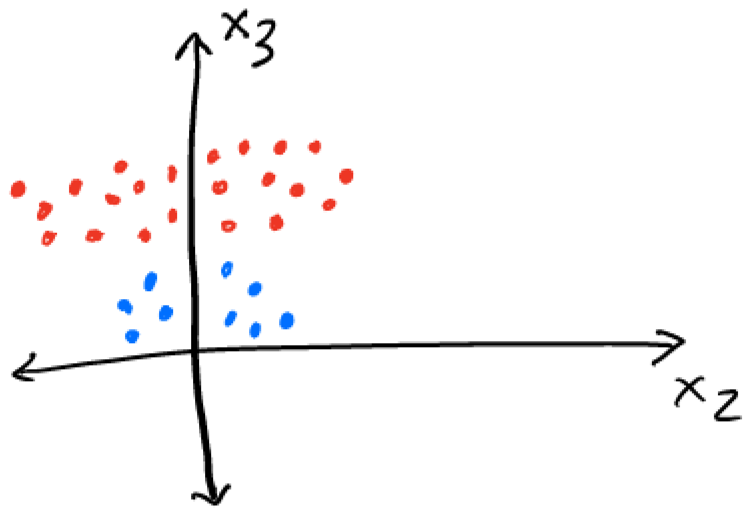
- From the above we can arrive at:
- 0.5($x_1^2$+1.4 $x_1$) + 0.1($x_2^2$+5 $x_2$) = (1-0.87)
- 0.5 ($x_1^2$+1.4 $x_1$+0.49) + 0.1 ($x_2^2$+5 $x_2$+6.25) = 1
- 0.5($x_1$+0.7)$^2$ + 0.1($x_2$+2.5)$^2$ = 1 $\Leftarrow$ Equation of an ellipse

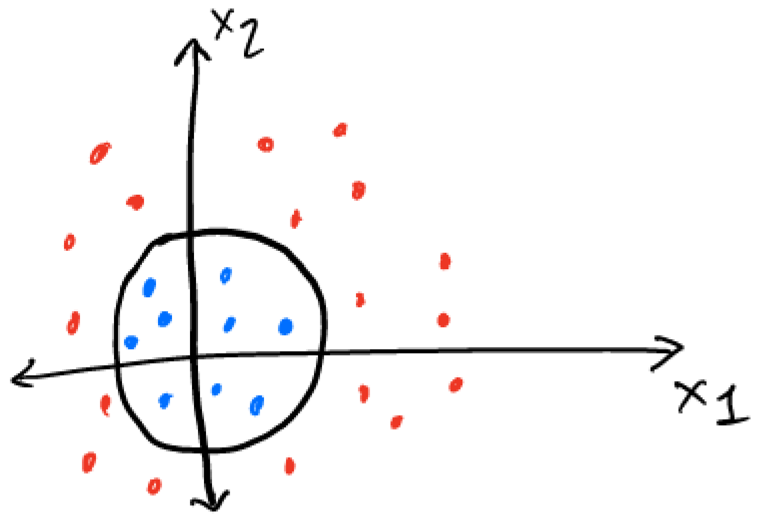
**The Support Vector Machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way (using *kernels*)**

- Kernel method is an efficient computational approach for achieving the above

# Support Vector Machines







- There is no separating hyperplane that can segregate the 2 classes
- Apply a transformation (kernel function) that maps the original data points into a higher-dimensional space in which they become separable
- Upon transforming it back to the original plane, we obtain the nonlinear circular boundary as shown above

# Support Vector Machines

- We can obtain the solution to the support vector classifier by simply taking inner products of the data points
- The inner product of vectors x$_1$ and x$_2$ can be represented as $\langle x_1, x_2 \rangle$ = $\sum_{j=1:p} x_{1\mathrm{j}} x_{2\mathrm{j}}$
- The decision rule for the support vector classifier can then be expressed as:

$$f(x) = \beta_{0+} \sum_{i=1}^{n} \alpha_i \langle x_, x_{\mathrm{i}} \rangle$$

$\alpha_i$ are the parameters that need to be estimated along with $\beta_0$

- $\alpha_i$ values are non-zero only for the support vectors

  What does n represent?

- In order to obtain the value of $f(x)$, we need to compute the inner product between the new point x and each of the training points $x_{\mathrm{i}}$ Measure of similarity
- We then classify the new point x as class 1 if $f(x) > 0$ and class -1 if $f(x) < 0$

# Support Vector Machines

- The inner product of vectors $x_1$ and $x_2$: $\langle x_1, x_2 \rangle = \sum_{j=1:p} x_{1j} x_{2j}$
- We can generalize the above as $K(x_i, x_{i'})$, where K is called a *kernel*

For example, a linear kernel will be:

$$K(x_i, x_{i'}) = \sum_{j=1:p} x_{ij} x_{i'j}$$

And a polynomial kernel of degree d will be:

$$K(x_i, x_{i'}) = \left(\sum_{j=1:p} x_{ij} x_{i'j}\right)^d$$

When the Support Vector Classifier is used with such a nonlinear kernel, then the resulting classifier is called a Support Vector Machine (SVM)

One of the popular kernels is the radial kernel

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1:p} (x_{ij} - x_{i'j})^2\right)$$

The tuning parameter $\gamma$ controls how quickly a training data point's influence decreases with increasing distance to the test data point

# Support Vector Machines

$p = 3$

$x_1 = (x_{11}, x_{12}, x_{13})$

$x_2 = (x_{21}, x_{22}, x_{23})$

polynomial degree $d = 2$

$f_1 =$
$(x_{11}x_{11}, x_{11}x_{12}, x_{11}x_{13},$
$x_{12}x_{11}, x_{12}x_{12}, x_{12}x_{13},$
$x_{13}x_{11}, x_{13}x_{12}, x_{13}x_{13})$

$f_2 =$
$(x_{21}x_{21}, x_{21}x_{22}, x_{21}x_{23},$
$x_{22}x_{21}, x_{22}x_{22}, x_{22}x_{23},$
$x_{23}x_{21}, x_{23}x_{22}, x_{23}x_{23})$

If we had used the kernel with $d = 2$,

$$K(x_1, x_2) = \left((1, 2, 3)\begin{pmatrix}4,\\5,\\6\end{pmatrix}\right)^2$$
$$= (4 + 10 + 18)^2$$
$$= (32)^2$$
$$= 1024$$

$x_1 = (1, 2, 3)$

$x_2 = (4, 5, 6)$

$f_1 = (1, 2, 3,$
$2, 4, 6,$
$3, 6, 9)$

$f_2 = (16, 20, 24,$
$20, 25, 30,$
$24, 30, 36)$

Inner product

$$= 16 + 40 + 72 + 40 + 100$$
$$+ \text{- - -} + 324$$
$$= 1024$$

Please go through the coding examples (Python) from the recording