Winter 2019

CSEN1022: Machine Learning

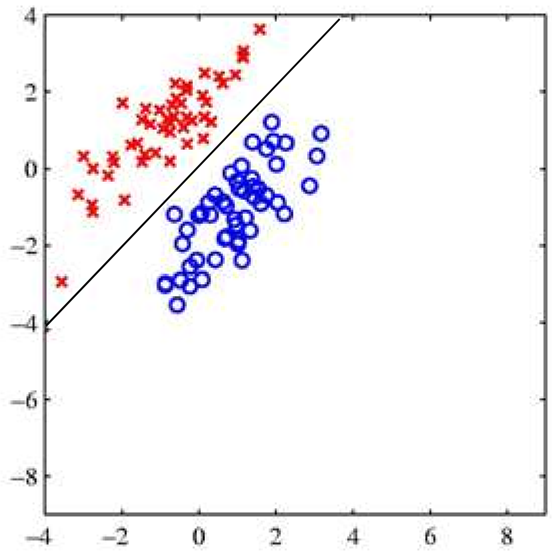
Non-linear Classifiers

Seif Eldawlatly

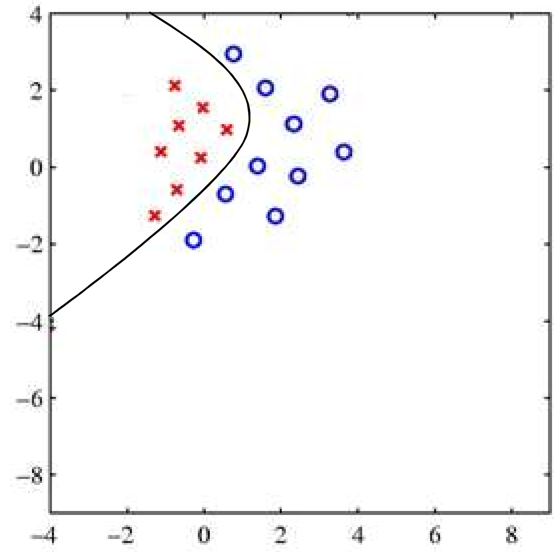
Linear vs. Non-linear

- Decision Boundary

Linearly Separable



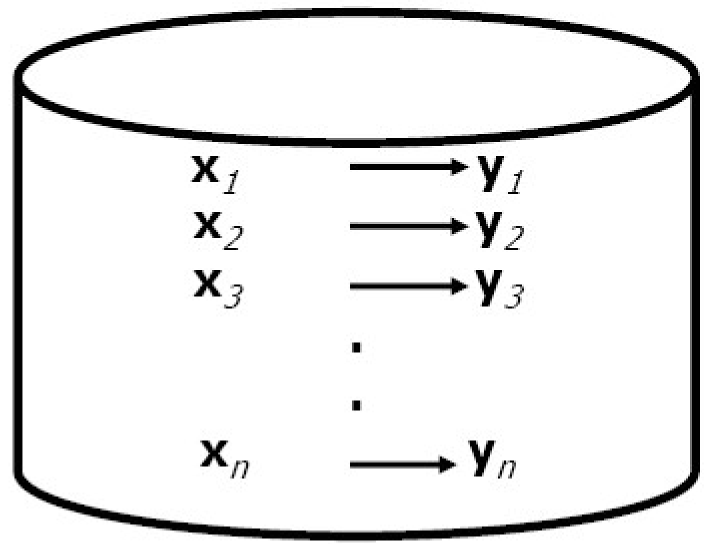
Non-linearly Separable



Instance-based Learning

- Each time a new instance is encountered, its relationship to previously stored instances is examined

- Disadvantage: Computation cost is high
  - To classify a new point, search database for similar points and fit with local points



K-nearest Neighbor (KNN) Classifier

- Most basic instance-based method

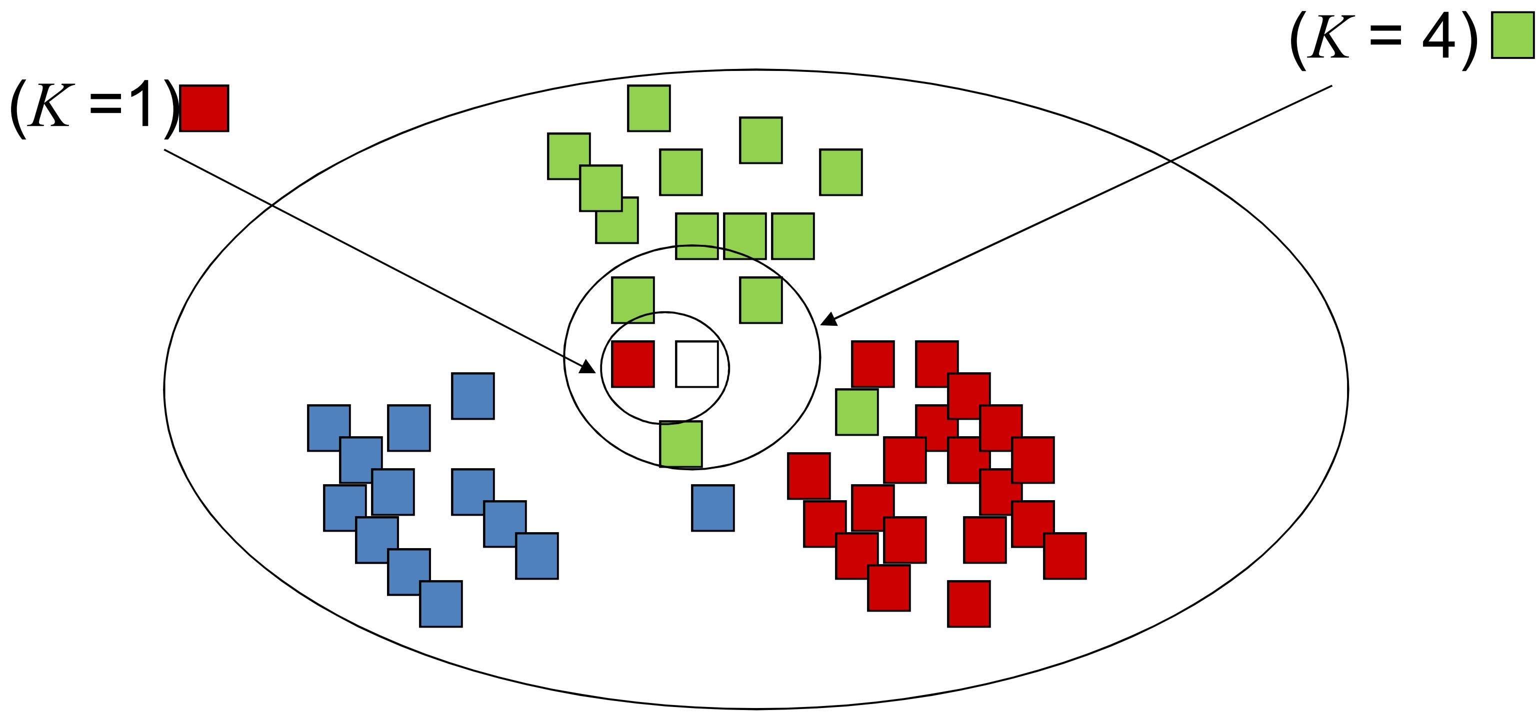
- Uses Euclidean distance to determine how dissimilar a pair of points are

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{r=1}^{n} (x_{ir} - x_{jr})^2}$$

- For any new input vector, the nearest K points are considered

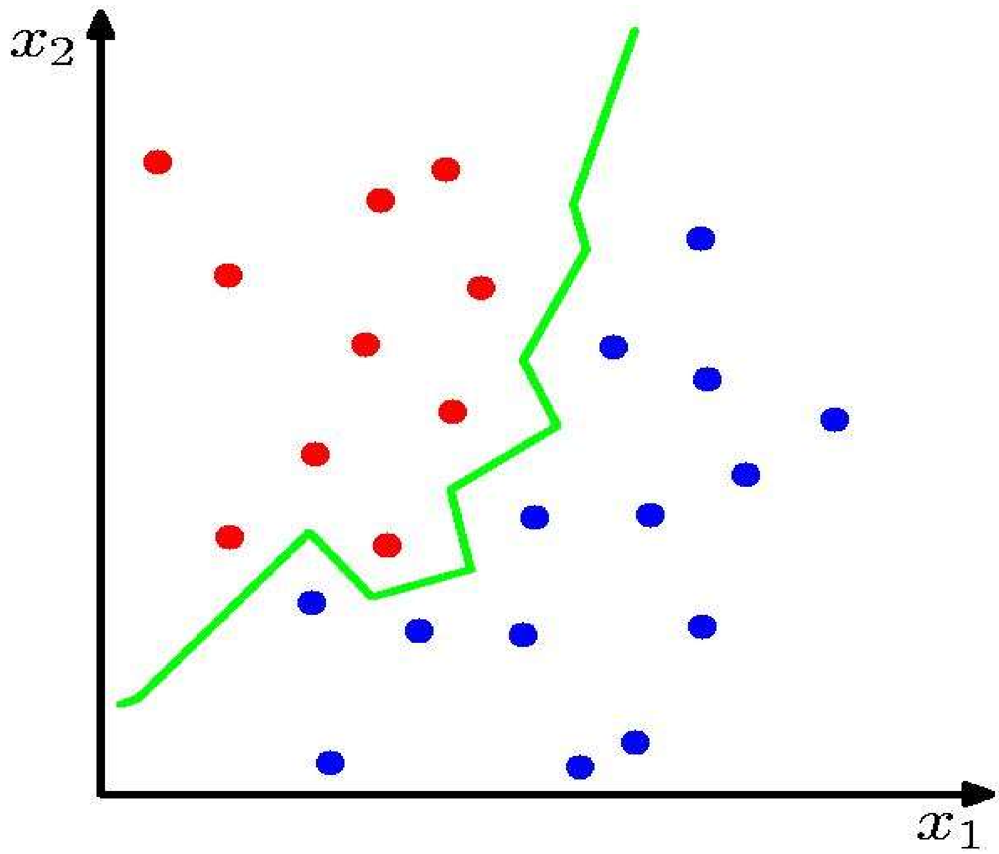
- A majority voting scheme is used to classify the new input vector

K-nearest Neighbor (KNN) Classifier



K-nearest Neighbor (KNN) Classifier

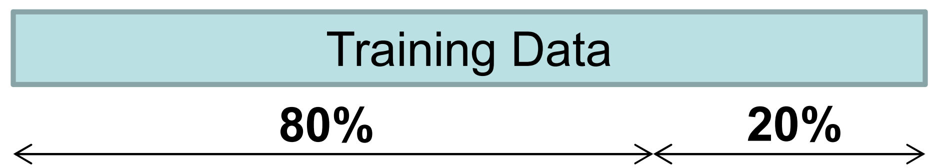
- A non-linear classifier



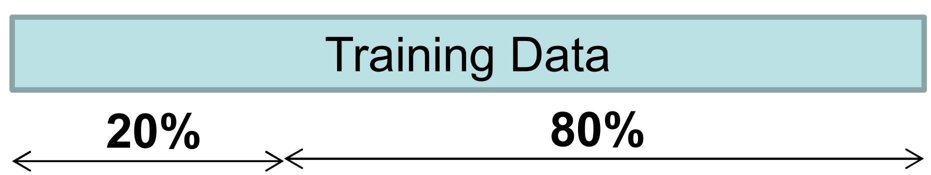
How to Choose *K*?

a) Cross-validation:

- 80% of training data for training and 20% for validation
- Find target value of the 20% part using the 80% and compute the corresponding error



$$\left\{\begin{array}{l} K = 1 \rightarrow \text{Error} = E_1 \\ K = 2 \rightarrow \text{Error} = E_2 \\ \quad\vdots \\ K = M \rightarrow \text{Error} = E_M \end{array}\right.$$



$$\left\{\begin{array}{l} K = 1 \rightarrow \text{Error} = E_1 \\ K = 2 \rightarrow \text{Error} = E_2 \\ \quad\vdots \\ K = M \rightarrow \text{Error} = E_M \end{array}\right.$$

The partitioning and validation process is repeated a number of times (for example 10 times) with different partitioning

How to Choose *K*?

a) Cross-validation:

- Find $K = k^*$ that minimizes the average error for the validation data

$$k^* = \arg\min_{k} \overline{E_k} \quad \text{, where} \quad \overline{E_k} = \frac{1}{L}\sum_{l=1}^{L} E_l$$

k = 1, 2, …, M, where M is the maximum number of neighbors

L is the total number of partitionings examined

- The obtained K is then used to classify the test data

How to Choose *K*?

b) Leave-one-out method

This method is equivalent to the previous cross-validation but with 1 validation point at a time

- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$

Weighted KNN Classifier

- Weight the contribution of each of the *K* neighbors according to their distance from the tested point

$$w(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{d(\mathbf{x}_i, \mathbf{x}_j)}$$

- Weighted posterior probability

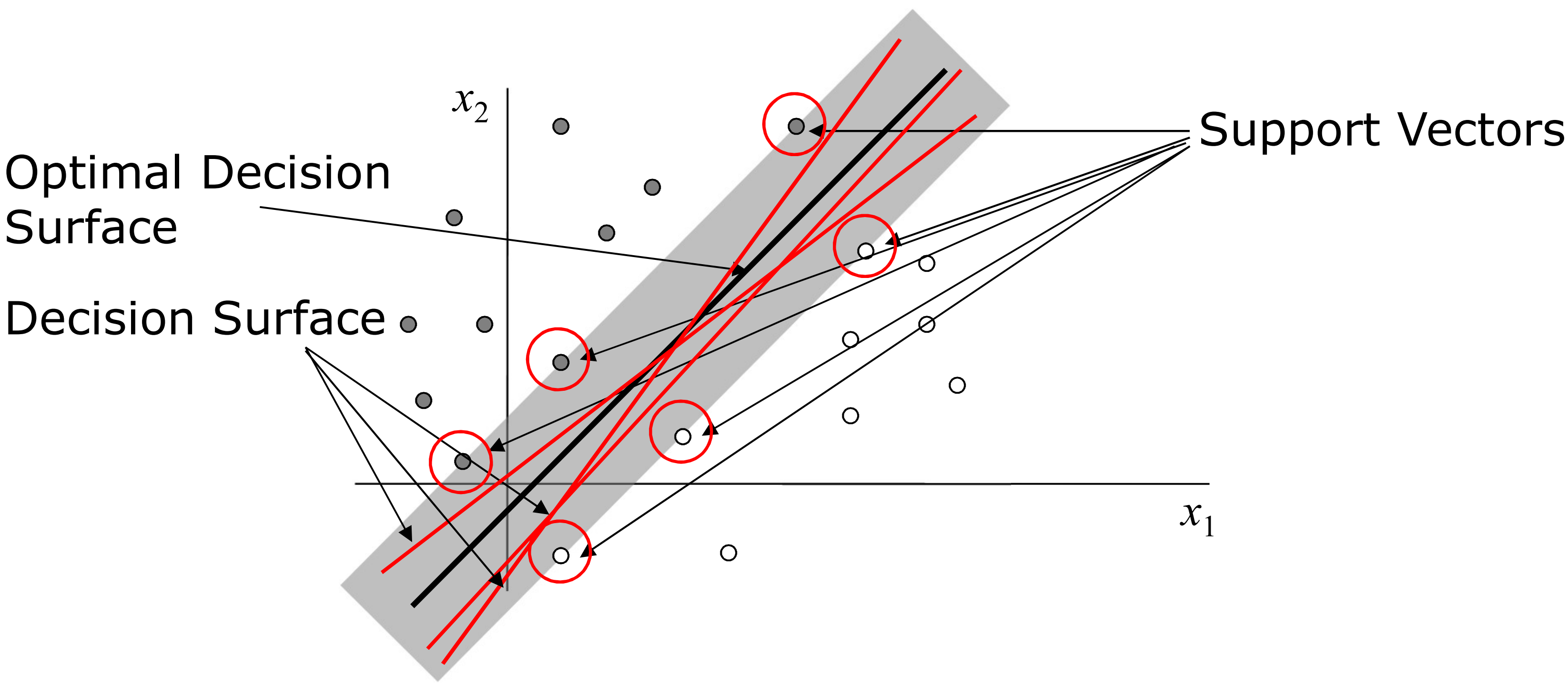
$$p(C_k | \mathbf{x}_i) = \frac{\sum_{j=1}^{K} w(\mathbf{x}_i, \mathbf{x}_j)\delta(C_j, C_k)}{\sum_{j=1}^{K} w(\mathbf{x}_i, \mathbf{x}_j)}$$

where $$\delta(C_j, C_k) = \begin{cases} 1 & , C_j = C_k \\ 0 & , C_j \neq C_k \end{cases}$$

Support Vector Machine (SVM)

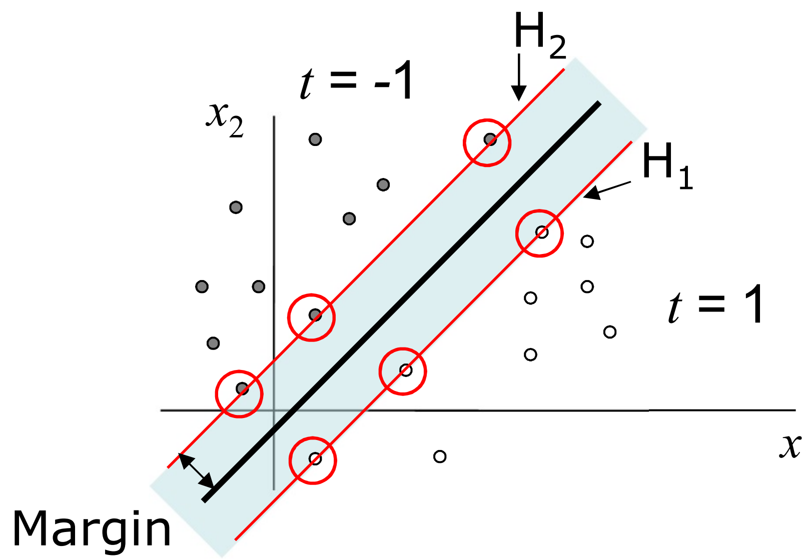
- Finds the optimal decision boundary

- For linearly-separable data



Support Vector Machine (SVM)

- SVM is a maximum margin classifier
- The margin: the smallest distance between the decision boundary and any of the input vectors



- Let

$$H_1: \quad \mathbf{w}^T\mathbf{x} + w_o = +1$$

$$H_2: \quad \mathbf{w}^T\mathbf{x} + w_o = -1$$

For t = 1 $\quad \mathbf{w}^T\mathbf{x}_i + w_o \geq +1$

For t = -1 $\quad \mathbf{w}^T\mathbf{x}_i + w_o \leq -1$

- Can be re-written as

$$t_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) \geq 1$$

- Distance between the decision boundary and any point

$$r = \frac{\mathbf{w}^T\mathbf{x}_i + w_o}{\|\mathbf{w}\|}$$

- For points on H_1 and H_2 $\quad r = \dfrac{1}{\|\mathbf{w}\|}$

Support Vector Machine (SVM)

- Maximizing the margin $\frac{1}{\|\mathbf{w}\|}$ is the same as minimizing $\|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$
- Finding the maximum margin can be formulated as

$$\text{Minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{subject to the constraints} \quad t_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) \geq 1$$

- To minimize a function subject to some constraints, we include the constraints in the objective function using Lagrange multipliers
- Using Lagrange multipliers, the objective function can be formulated as

$$J = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\alpha_i\left[t_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) - 1\right]$$

- Taking derivative w.r.t. parameters and equate with zero

$$\mathbf{w} \quad \rightarrow \quad \mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i \mathbf{x}_i$$

$$w_o \quad \rightarrow \quad \sum_{i=1}^{N}\alpha_i t_i = 0$$

Support Vector Machine (SVM)

- Substitute in J, the problem can be re-written as: Find the Lagrange multipliers that maximize

$$J = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i t_i = 0$$

$$\alpha_i \geq 0, \text{ for } i = 1,2,\ldots,N$$

- This can be solved using convex quadratic programming optimization to find α_i and so find the decision boundary

Support Vector Machine (SVM)

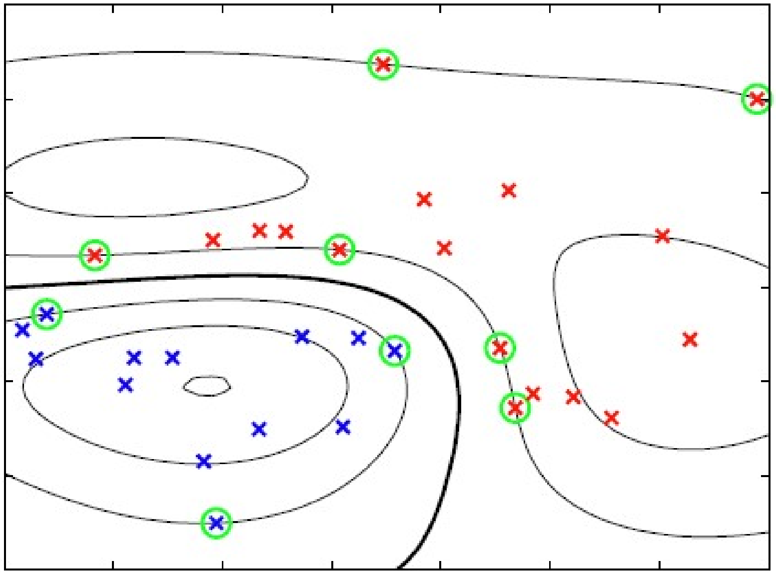
- For non-linearly separable data

$$J = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i t_i = 0$$

$$C \geq \alpha_i \geq 0, \text{ for } i = 1,2,\ldots,N$$



- Kernel function for non-linear transformation

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) \quad \begin{cases} K(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \mathbf{x}_j + 1\right)^p \\ K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\dfrac{-1}{2\sigma^2} \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \\ K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\beta_o \mathbf{x}_i^T \mathbf{x}_j + \beta_1\right) \end{cases}$$

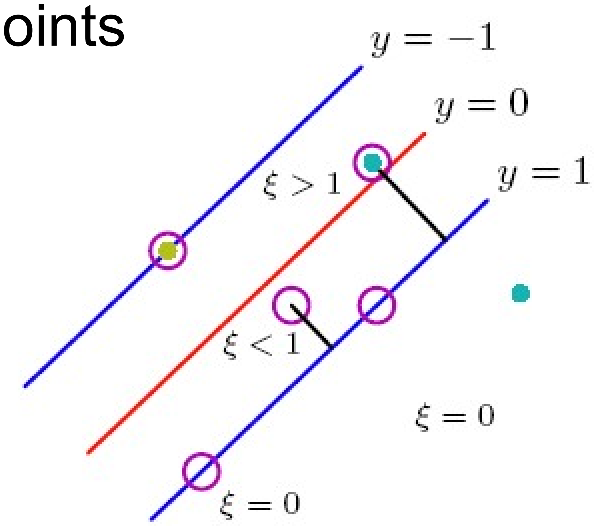
Support Vector Machine (SVM)

- What is the constant C ?
  - Define slack variables ξ to allow points to be miss-classified

$$t_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) \geq 1 - \xi_i$$

  - Minimize the objective function

$$J = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$



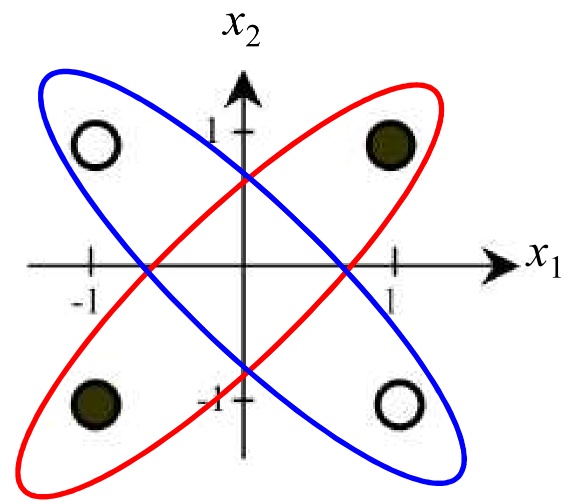
- This makes SVM a non-linear classifier
- For the non-linear case, the solution of SVM is similar to the linear case and given by

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i \phi(\mathbf{x}_i)$$

SVM for XOR Problem

- Consider designing a classifier to identify two classes in the non-linearly separable case of the XOR gate

Input Vector $\mathbf{x} = [x_1 \;\; x_2]^T$	Target Value t
$[-1\ -1]^T$	-1
$[-1\ 1]^T$	1
$[1\ -1]^T$	1
$[1\ 1]^T$	-1



Using the polynomial kernel with p = 2 $K(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \mathbf{x}_j + 1\right)^2$

SVM for XOR Problem

- The polynomial kernel can be re-expressed as

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= \left(\mathbf{x}_i^T \mathbf{x}_j + 1\right)^2 \\
&= \left(\begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} x_{j1} \\ x_{j2} \end{bmatrix} + 1 \right)^2 \\
&= \left(x_{i1} x_{j1} + x_{i2} x_{j2} + 1 \right)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2}
\end{aligned}
$$

$$
= \begin{bmatrix} 1 & x_{i1}^2 & \sqrt{2} x_{i1} x_{i2} & x_{i2}^2 & \sqrt{2} x_{i1} & \sqrt{2} x_{i2} \end{bmatrix} \begin{bmatrix} 1 \\ x_{j1}^2 \\ \sqrt{2} x_{j1} x_{j2} \\ x_{j2}^2 \\ \sqrt{2} x_{j1} \\ \sqrt{2} x_{j2} \end{bmatrix}
$$

$$
= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
$$

SVM for XOR Problem

- For the XOR problem, the kernel matrix is given by $K(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \mathbf{x}_j + 1\right)^2$

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & K(\mathbf{x}_1, \mathbf{x}_3) & K(\mathbf{x}_1, \mathbf{x}_4) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & K(\mathbf{x}_2, \mathbf{x}_3) & K(\mathbf{x}_2, \mathbf{x}_4) \\ K(\mathbf{x}_3, \mathbf{x}_1) & K(\mathbf{x}_3, \mathbf{x}_2) & K(\mathbf{x}_3, \mathbf{x}_3) & K(\mathbf{x}_3, \mathbf{x}_4) \\ K(\mathbf{x}_4, \mathbf{x}_1) & K(\mathbf{x}_4, \mathbf{x}_2) & K(\mathbf{x}_4, \mathbf{x}_3) & K(\mathbf{x}_4, \mathbf{x}_4) \end{bmatrix}$$

	Input Vector $\mathbf{x} = [x_1 \ \ x_2]^T$	Target Value t
$\mathbf{x}_1$	$[-1\ -1]^T$	-1
$\mathbf{x}_2$	$[-1\ 1]^T$	1
$\mathbf{x}_3$	$[1\ -1]^T$	1
$\mathbf{x}_4$	$[1\ 1]^T$	-1

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

$$J = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j t_i t_j K\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

$$= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2)$$

SVM for XOR Problem

- Taking the derivative of J with respect to each α_i and equate with zero

$$
\begin{aligned}
9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\
-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\
-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\
\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1
\end{aligned}
$$

- By solving these equations simultaneously, we get

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$$

SVM for XOR Problem

- Since the weight vector $\mathbf{w}$ is given by

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \phi(\mathbf{x}_i)$$

where

$$\phi(\mathbf{x}_i) = \begin{bmatrix} 1 & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} \end{bmatrix}^T$$

Therefore

$$\mathbf{w} = \frac{1}{8}[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)]$$

$$= \frac{1}{8}\left[-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ \frac{-1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

	Input Vector $\mathbf{x} = [x_1 \; x_2]^T$	Target Value t
$\mathbf{x}_1$	$[-1 \; -1]^T$	-1
$\mathbf{x}_2$	$[-1 \; 1]^T$	1
$\mathbf{x}_3$	$[1 \; -1]^T$	1
$\mathbf{x}_4$	$[1 \; 1]^T$	-1

SVM for XOR Problem

- Therefore for any new input vector $\mathbf{x} = [x_1, x_2]^T$, the decision boundary is defined as

$$\mathbf{w}^T \phi(\mathbf{x}) = \begin{bmatrix} 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0$$

$$\Rightarrow -x_1 x_2 = 0$$

	Input Vector $\mathbf{x} = [x_1 \ \ x_2]^T$	Target Value t	$-x_1 x_2$
$\mathbf{x}_1$	[-1 -1]^T	-1	-1
$\mathbf{x}_2$	[-1 1]^T	1	1
$\mathbf{x}_3$	[1 -1]^T	1	1
$\mathbf{x}_4$	[1 1]^T	-1	-1