# Lecture 14: Support Vector Machine (SVM)

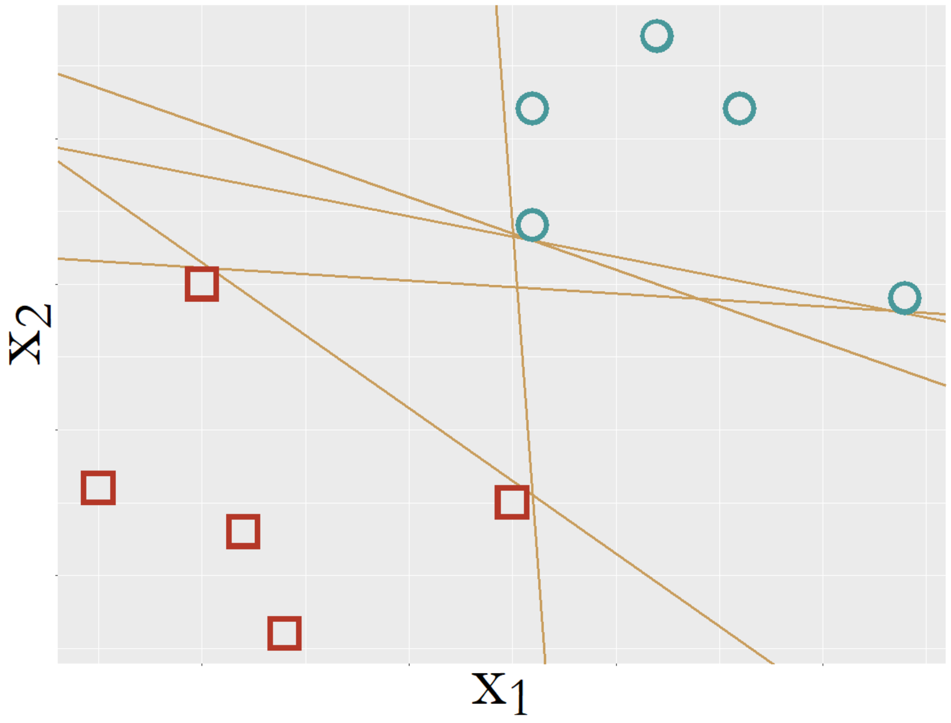
Instructor: Prof. Shuai Huang

Industrial and Systems Engineering

University of Washington

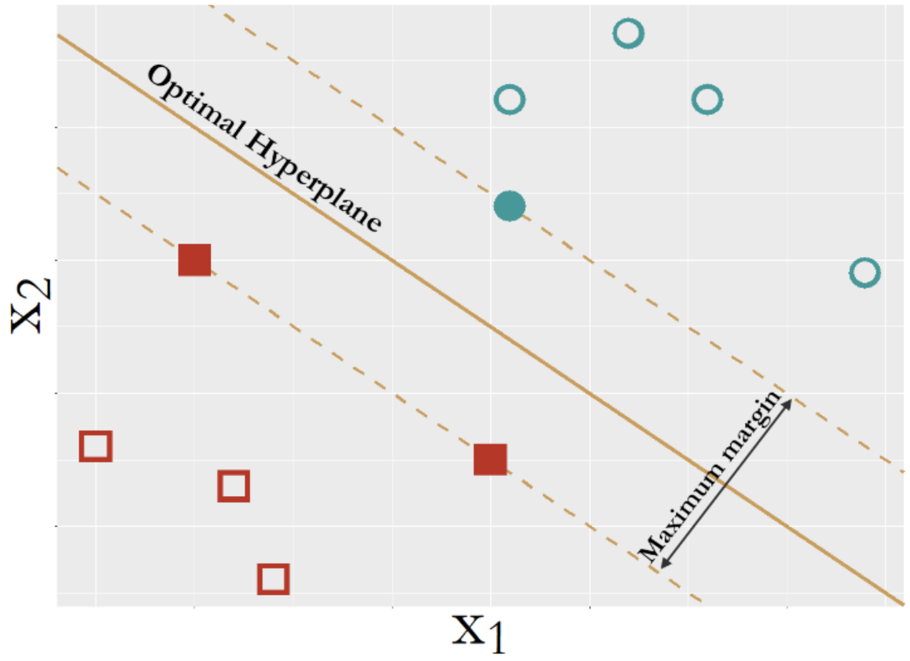
# What ambiguity the SVM ties to solve

- Which model should we use?



# The model with maximum margin

- SVM is essentially a preference over models that have maximum margin



# Can this idea lead to mathematic tractability?

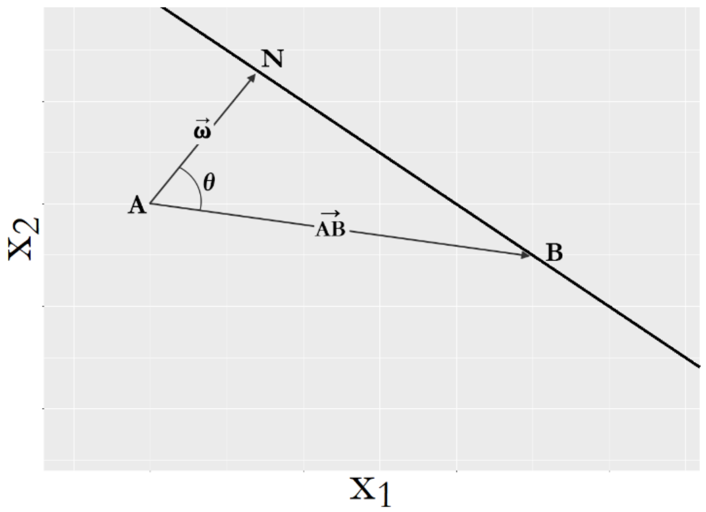
- The goal is to identify a model, $\boldsymbol{w}^T\boldsymbol{x} + b$, using which we can make binary classification:

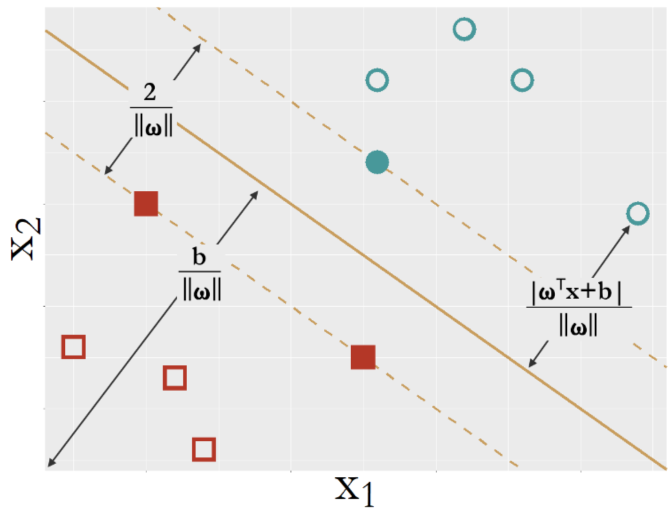
  If $\boldsymbol{w}^T\boldsymbol{x} + b > 0$, then $y = 1$; Otherwise, $y = -1$.

- The final SVM formulation is:

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

Subject to: $y_n\left(\boldsymbol{w}^T\boldsymbol{x}_n + b\right) \geq 1$ for $n = 1,2, \ldots, N$.



# Solve for SVM

- To solve this problem, first, we can use the method of Lagrange multiplier:
$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} \alpha_n \left[y_n\left(\boldsymbol{w}^T \boldsymbol{x}_n + b\right) - 1\right].$$
- This could be rewritten as
$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w} - \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{w}^T \boldsymbol{x}_n - b \sum_{n=1}^{N} \alpha_n y_n + \sum_{n=1}^{N} \alpha_n.$$
- Differentiating $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{w}$ and $b$, and setting to zero yields:
$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n, \ \ \sum_{n=1}^{N} \alpha_n y_n = 0.$$
- Then, we can rewrite $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ as
$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \boldsymbol{x}_n^T \boldsymbol{x}_m.$$
- This is because that:
$$\frac{1}{2}\boldsymbol{w}^T \boldsymbol{w} = \frac{1}{2}\boldsymbol{w}^T \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n = \frac{1}{2}\sum_{n=1}^{N} \alpha_n y_n \boldsymbol{w}^T \boldsymbol{x}_n =$$
$$\frac{1}{2}\sum_{n=1}^{N} \alpha_n y_n \left(\sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n\right)^T \boldsymbol{x}_n = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \boldsymbol{x}_n^T \boldsymbol{x}_m.$$

# The dual form of SVM

- Finally, we can derive the model of SVM by solving its dual form problem:

$$\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \boldsymbol{x}_n^T \boldsymbol{x}_m,$$

Subject to: $\alpha_n \geq 0$ for $n = 1,2,\ldots,N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0.$

- This is a **quadratic programming** problem that can be solved using many existing packages.

# The support points

- The learned model parameters could be represented as:

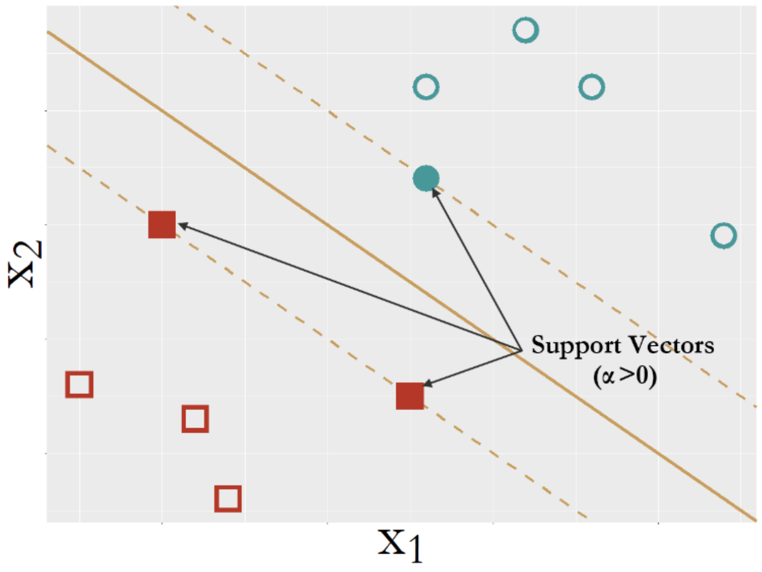
$\hat{\boldsymbol{w}} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$ and $\hat{b} = 1 - \hat{\boldsymbol{w}}^T \boldsymbol{x}_n$ for any $\boldsymbol{x}_n$ whose $\alpha_n > 0$.

- And we know that, based on the KKT condition:

$$\alpha_n\left[y_n\left(\boldsymbol{w}^T \boldsymbol{x}_n + b\right) - 1\right] = 0 \text{ for } n = 1,2, \dots, N.$$

- Thus, for any data point, e.g., the nth data point, it is either

$$\alpha_n = 0 \text{ or } y_n\left(\boldsymbol{w}^T \boldsymbol{x}_n + b\right) - 1 = 0.$$



# A toy example

```r
# For the toy problem
x = matrix(c(5,5,2,2,1,0,0,1,-1,1,-1,0,1,-1), 
nrow = 7, ncol = 2)
y = c(1,1,1,1,-1,-1,-1)
linear.train <- data.frame(x,y)

# Visualize the distribution of data points o
f two classes
require( 'ggplot2' )

p <- qplot( data=linear.train, X1, X2, colour
=factor(y))
p <- p + labs(title = "Scatterplot of data po
ints of two classes")
print(p)
```

| ID | $X_1$ | $X_2$ | $Y$ |
|---|---|---|---|
| 1 | 5 | 1 | 1 |
| 2 | 5 | -1 | 1 |
| 3 | 2 | 1 | 1 |
| 4 | 2 | -1 | 1 |
| 5 | 1 | 0 | -1 |
| 6 | 0 | 1 | -1 |
| 7 | 0 | -1 | -1 |

# A toy example – cont'd

- We can directly identify three support vectors, which are (ID = 3, 4, 5)
- Then, since

$$f(\boldsymbol{x}_*) = \widehat{\boldsymbol{w}}^T \phi(\boldsymbol{x}_*) + b = \sum_{n=1}^{7} \alpha_n y_n \, \phi(\boldsymbol{x}_n)^T \phi(\boldsymbol{x}_*) + b,$$

- and we know that $f(\boldsymbol{x}_3) = 1$, $f(\boldsymbol{x}_4) = 1$,$f(\boldsymbol{x}_5) = -1$.
- We can identify the alpha vector as

$$\alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_3) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_3) + \alpha_5 y_5 \phi(\boldsymbol{x}_5)^T \phi(\boldsymbol{x}_3) + b = 1,$$
$$\alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_4) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_4) + \alpha_5 y_5 \phi(\boldsymbol{x}_5)^T \phi(\boldsymbol{x}_4) + b = 1,$$
$$\alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_5) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_5) + \alpha_5 y_5 \phi(\boldsymbol{x}_5)^T \phi(\boldsymbol{x}_5) + b = -1.$$

# A toy example – cont'd

- Here, since the data is linearly separable, we use linear kernel, i.e., in other words, $\phi(\boldsymbol{x}) = \boldsymbol{x}$. The equations above can be simplified to

$$5\alpha_3 + 3\alpha_4 - 2\alpha_5 + b = 1,$$
$$3\alpha_3 + 5\alpha_4 - 2\alpha_5 + b = 1,$$
$$2\alpha_3 + 2\alpha_4 - \alpha_5 + b = -1.$$

- By solving it, we can get

$$\alpha_3 = 1, \alpha_4 = 1, \alpha_5 = 2, b = -3.$$

- Further, we can know that

$$\widehat{\boldsymbol{w}} = \alpha_3 y_3 \phi(\boldsymbol{x}_3) + \alpha_4 y_4 \phi(\boldsymbol{x}_4) + \alpha_5 y_5 \phi(\boldsymbol{x}_5) = 1\begin{pmatrix}2\\1\end{pmatrix} + 1\begin{pmatrix}2\\-1\end{pmatrix} - 2\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}2\\0\end{pmatrix}.$$

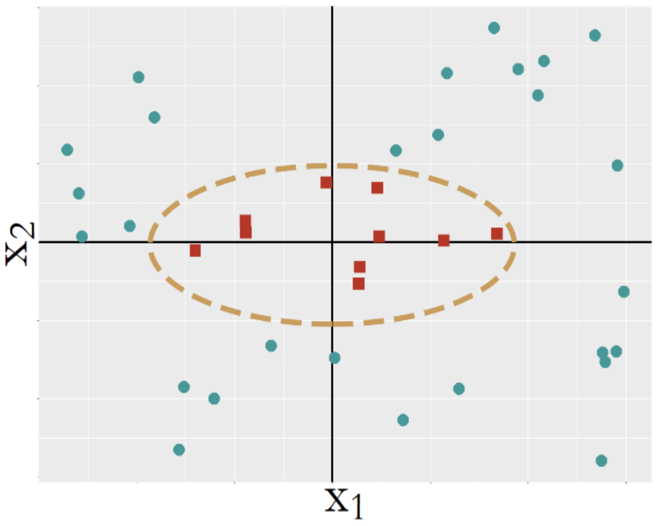
# A toy example – cont'd

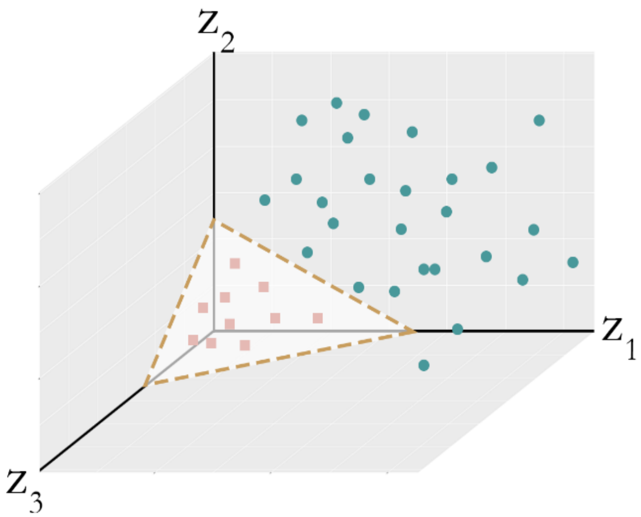
- Validate your calculation using R

```
x = matrix(c(5,5,2,2,1,0,0,1,-1,1,-1,0,1,-1), nrow = 7, ncol = 2)
y = c(1,1,1,1,-1,-1,-1)
linear.train <- data.frame(x,y)
require( 'kernlab' )
linear.svm <- ksvm(y ~ ., data=linear.train, type='C-svc', kernel='vanilladot',
C=10, scale=c(),scaled = FALSE)
alpha(linear.svm) #scaled alpha vector
b(linear.svm)
```

# Extension to nonlinear cases

- Main idea: transformation from $\boldsymbol{x}$ to $\boldsymbol{z}$
- An example: $z_1 = x_1^2, z_2 = \sqrt{2}x_1x_2, z_3 = x_2^2.$
- But not in all times the transformations can be made explicit



# A toy example

- Consider a dataset:

$$\boldsymbol{x}_1 = (-1, -1), y_1 = -1;$$
$$\boldsymbol{x}_2 = (-1, +1), y_2 = +1;$$
$$\boldsymbol{x}_3 = (+1, -1), y_3 = +1;$$
$$\boldsymbol{x}_4 = (+1, +1), y_4 = -1.$$

# A toy example – cont'd

Now, consider the polynomial kernel function with degree of order = 2, i.e., $K(\boldsymbol{x}_n, \boldsymbol{x}_m) = \left(\boldsymbol{x}_n^T \boldsymbol{x}_m + c\right)^2$, which corresponds to the transformation:

$$\phi(\boldsymbol{x}_n) = \left[c^2, \sqrt{2c}x_{n,1}, \sqrt{2c}x_{n,2}, \sqrt{2}x_{n,1}x_{n,2}, x_{n,1}^2, x_{n,2}^2\right]^T.$$

We can identify the alpha vector as

$$\alpha_1 y_1 \phi(\boldsymbol{x}_1)^T \phi(\boldsymbol{x}_1) + \alpha_2 y_2 \phi(\boldsymbol{x}_2)^T \phi(\boldsymbol{x}_1) + \alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_1) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_1) + b = -1,$$

$$\alpha_1 y_1 \phi(\boldsymbol{x}_1)^T \phi(\boldsymbol{x}_2) + \alpha_2 y_2 \phi(\boldsymbol{x}_2)^T \phi(\boldsymbol{x}_2) + \alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_2) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_2) + b = 1,$$

$$\alpha_1 y_1 \phi(\boldsymbol{x}_1)^T \phi(\boldsymbol{x}_3) + \alpha_2 y_2 \phi(\boldsymbol{x}_2)^T \phi(\boldsymbol{x}_3) + \alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_3) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_3) + b = -1,$$

$$\alpha_1 y_1 \phi(\boldsymbol{x}_1)^T \phi(\boldsymbol{x}_4) + \alpha_2 y_2 \phi(\boldsymbol{x}_2)^T \phi(\boldsymbol{x}_4) + \alpha_3 y_3 \phi(\boldsymbol{x}_3)^T \phi(\boldsymbol{x}_4) + \alpha_4 y_4 \phi(\boldsymbol{x}_4)^T \phi(\boldsymbol{x}_4) + b = 1.$$

# A toy example – cont'd

Since $\phi(\boldsymbol{x}_n)^T\phi(\boldsymbol{x}_m) = K(\boldsymbol{x}_n, \boldsymbol{x}_m) = \left(\boldsymbol{x}_n^T\boldsymbol{x}_m + c\right)^2$, here, let's set $c = 0$, we can calculate the kernel matrix as

$$\boldsymbol{K} = \begin{bmatrix} 4 & 0 & 0 & 4 \\ 0 & 4 & 4 & 0 \\ 0 & 4 & 4 & 0 \\ 4 & 0 & 0 & 4 \end{bmatrix}.$$

The equations above can be simplified to

$$-4\alpha_1 - 4\alpha_4 + b = -1,$$
$$4\alpha_2 + 4\alpha_3 + b = 1,$$
$$4\alpha_2 + 4\alpha_3 + b = 1,$$
$$-4\alpha_1 - 4\alpha_4 + b = -1.$$

# A toy example – cont'd

You can note that, here, we actually only have two independent equations. We can identify one solution to be:

$$\alpha_1 = 0.125, \alpha_2 = 0.125, \alpha_3 = 0.125, \alpha_4 = 0.125, b = 0.$$

Further, in this particular case, as we can write up the transformation explicitly, we can write up $\widehat{\boldsymbol{w}}$ explicitly as:

$$\widehat{\boldsymbol{w}} = \sum_{n=1}^{4} \alpha_n y_n \phi(\boldsymbol{x}_n) = \left[0,0,0,1/\sqrt{2}, 0,0\right]^T.$$

Then, we can write up the decision function explicitly as:

$$f(\boldsymbol{x}_*) = \widehat{\boldsymbol{w}}^T \phi(\boldsymbol{x}_*) = x_{*,1} x_{*,2}.$$

# A toy example – cont’d

- Validate your calculation using R

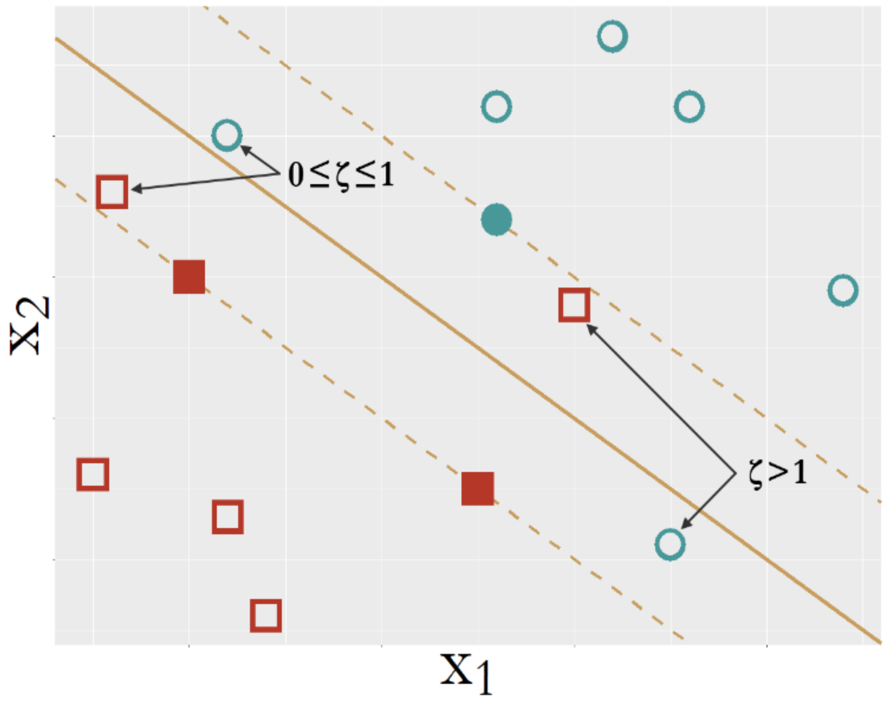
```
x = matrix(c(-1,-1,1,1,-1,1,-1,1), nrow = 4, ncol = 2)
y = c(-1,1,1,-1)
linear.train <- data.frame(x,y)
require( 'kernlab' )
linear.svm <- ksvm(y ~ ., data=linear.train, type='C-svc', kernel='polydot',
kpar=list(degree = 2, offset = 1),  C = 10, scale = c(), scaled = FALSE)
alpha(linear.svm) #scaled alpha vector
b(linear.svm)
coef(linear.svm)
```

# Extension to non-separable cases

- Introduce the slack variables:

  $$y_n(\boldsymbol{w}^T\boldsymbol{x}_n + b) \geq 1 - \xi_n \text{ for } n = 1,2,\ldots,N.$$

- The data points that are within the margins will have the corresponding slack variables as $0 \leq \xi_n \leq 1$
- The data points that are on the wrong side of the decision line have the corresponding slack variables as $\xi_n > 1$.



# The revised SVM formulation

- The corresponding formulation of the SVM model becomes:

$$\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{n=1}^{N} \xi_n,$$

Subject to: $y_n\left(\boldsymbol{w}^T \boldsymbol{x}_n + b\right) \geq 1 - \xi_n$ and $\xi_n \geq 0$, for $n = 1,2, \ldots, N$.

# Assume the transformation exists

- The dual formulation of SVM on the transformed variables is:

$$\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \boldsymbol{z}_n^T \boldsymbol{z}_m,$$

Subject to: $0 \leq \alpha_n \leq C$ for $n = 1,2, \ldots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$.

- What matters here is really the inner product of the transformed vectors
- Thus, we can write it up as $\boldsymbol{z}_n^T \boldsymbol{z}_m = K(\boldsymbol{x}_n, \boldsymbol{x}_m)$. This is called the "**kernel function**". A kernel function is a function that theoretically entails a transformation $\boldsymbol{z} = \phi(\boldsymbol{x})$ such that $K(\boldsymbol{x}_n, \boldsymbol{x}_m)$ implies that it can be written as an inner product $K(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x})$.

# The revised SVM formulation

- With a given kernel function, SVM learns the model by solving the following optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m K(\boldsymbol{x}_n, \boldsymbol{x}_m),$$

Subject to: $0 \leq \alpha_n \leq C$ for $n = 1,2, \dots, N$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$.

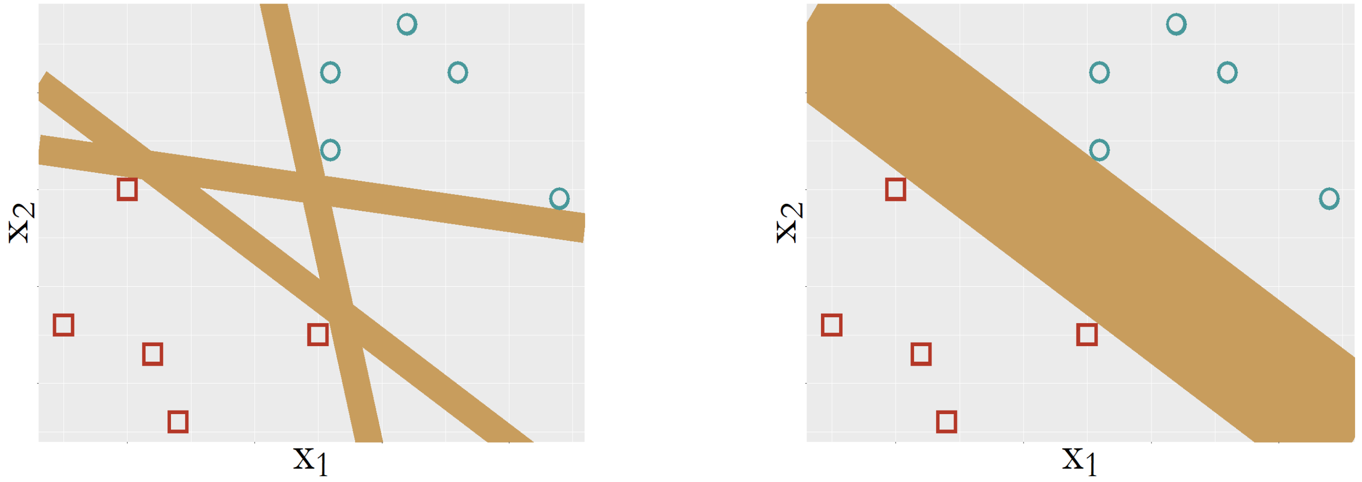
- However, in the kernel space, it will no longer to possible to write up the parameter $\boldsymbol{w}$ the same way as in linear models.
- For any new data point, denoted as $\boldsymbol{x}_*$, the learned SVM model predict on it as

If $\sum_{n=1}^{N} \alpha_n y_n K(\boldsymbol{x}_n, \boldsymbol{x}_*) + b > 0$, then $y = 1$;

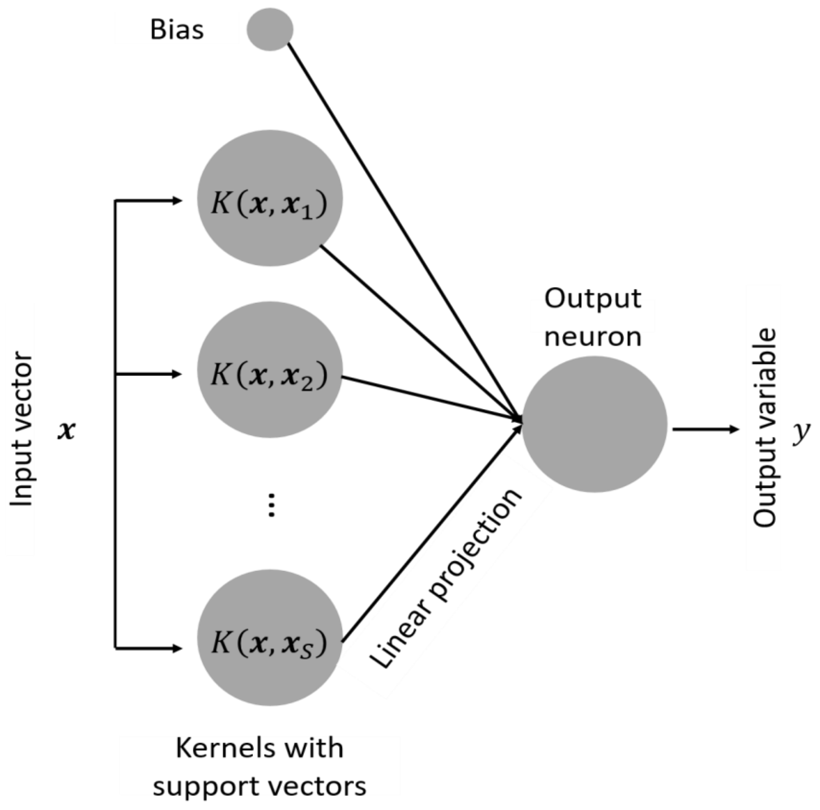
Otherwise, $y = -1$.

# Is SVM a more complex model?

- In statistical learning theory, a more complex model has larger VC-dimension. In intuitive language, that means, a more complex model has more mathematical capacity to encode a richer signal. Thus, it could be very flexible and sensitive to data distributions
- However, for SVM …



# SVM is a neural network model



# R lab

- Download the markdown code from course website
- Conduct the experiments
- Interpret the results
- Repeat the analysis on other datasets