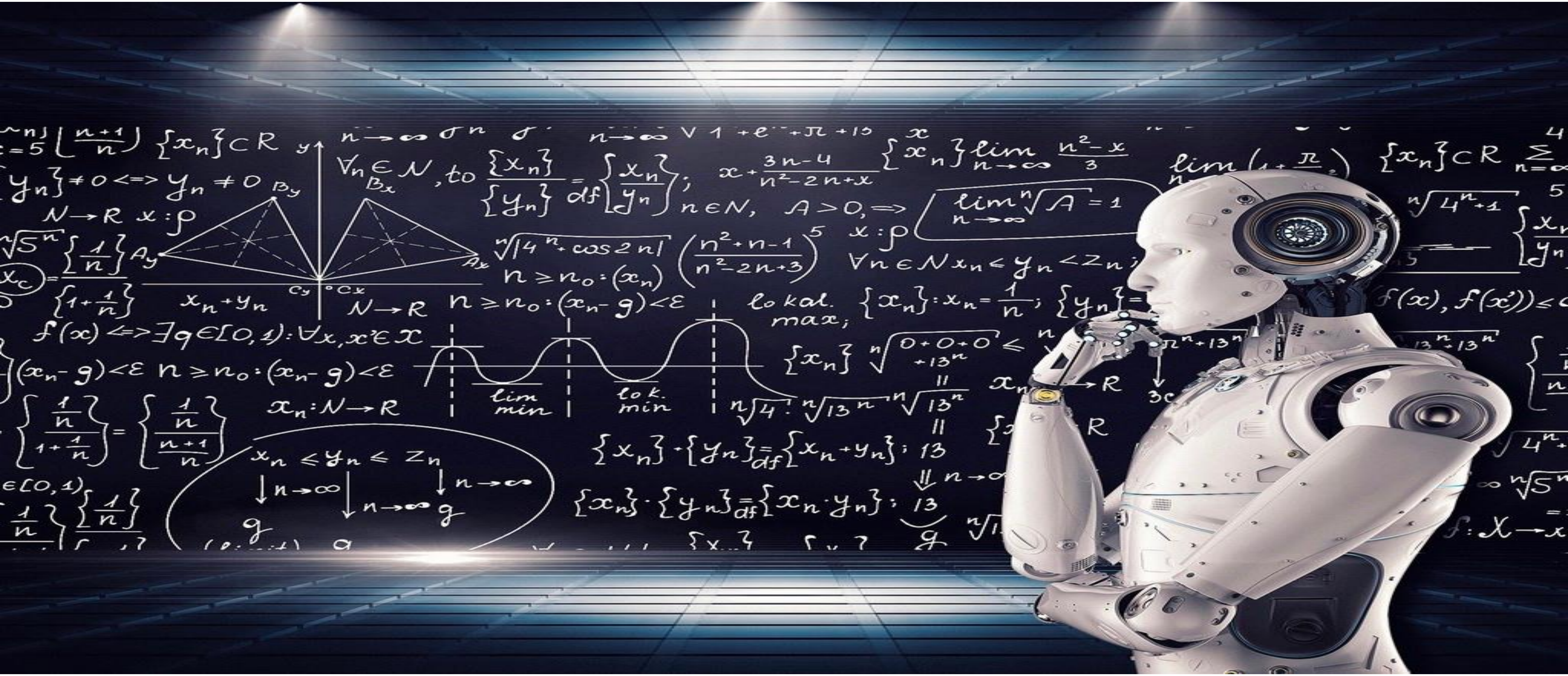
SVM – Support Vector Machine



The goal

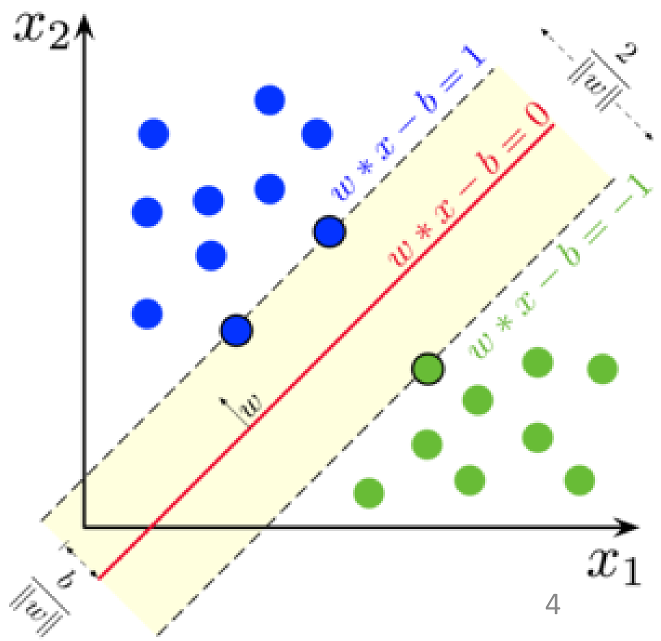
- Find a linear classifier that can separate the data set (we will talk only on 2 classes)
- SVM is based on 3 ideas:
  - The Kernel trick – map data to high dimensional space where it is easier to classify with linear decision surfaces
  - Max Margin – for linearly separable problem, the maximal margin hyperplane is the optimal linear classifier
  - Soft Margin and Regularization – extend the above definition for non-linearly separable problems. introduce term for misclassifications

Definitions

- Linear classifier:
    - A linear function (a hyperplane plane in the features space) that can separate d dimensional data set

$$f(\vec{x}, \vec{w}, b) = sign(\vec{w} \cdot \vec{x} + b)$$

- Margin(x_i) = the distance between the decision boundary and x_i



- Margin$(\vec{w}, b)$ = $\min$ Margin(x_i)
- Maximal margin classifier – $\vec{w}, b = arg\max$ Margin$(\vec{w}, b)$

Non-linearly separable problems

- What can we do with nonlinear separable data?
  - Instance based?
- Transformation \ Mapping?

Mapping

- We want to map the instances to a different feature space where the data is linearly separable
- Then, we will be able to separate the data with a linear hyper-plane

Mapping

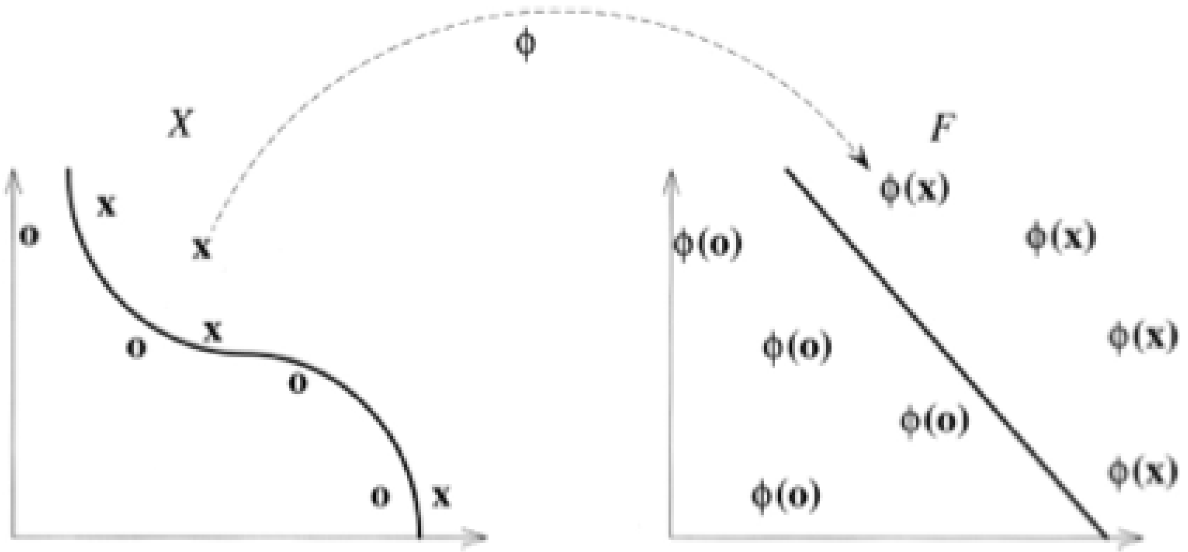
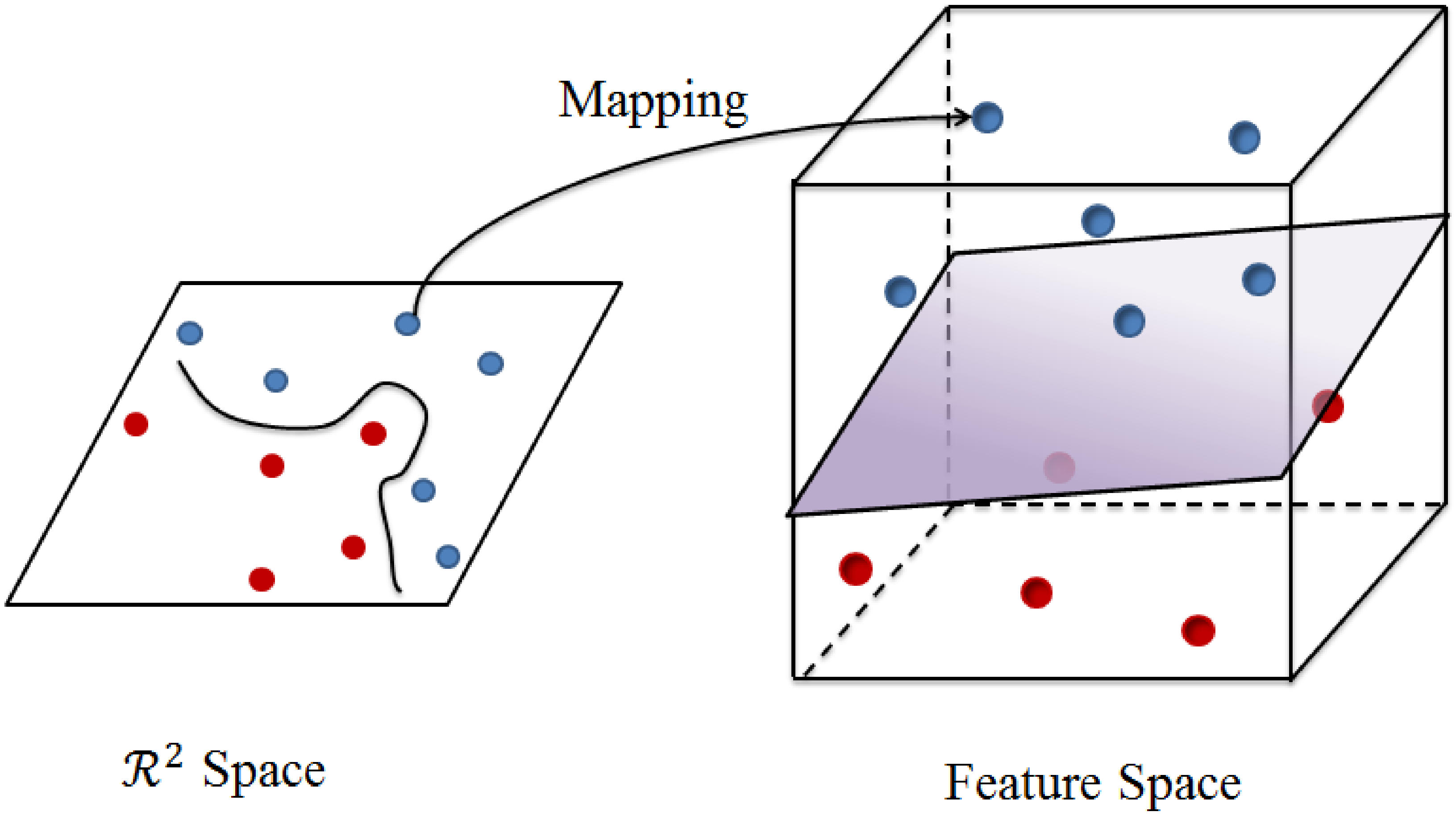


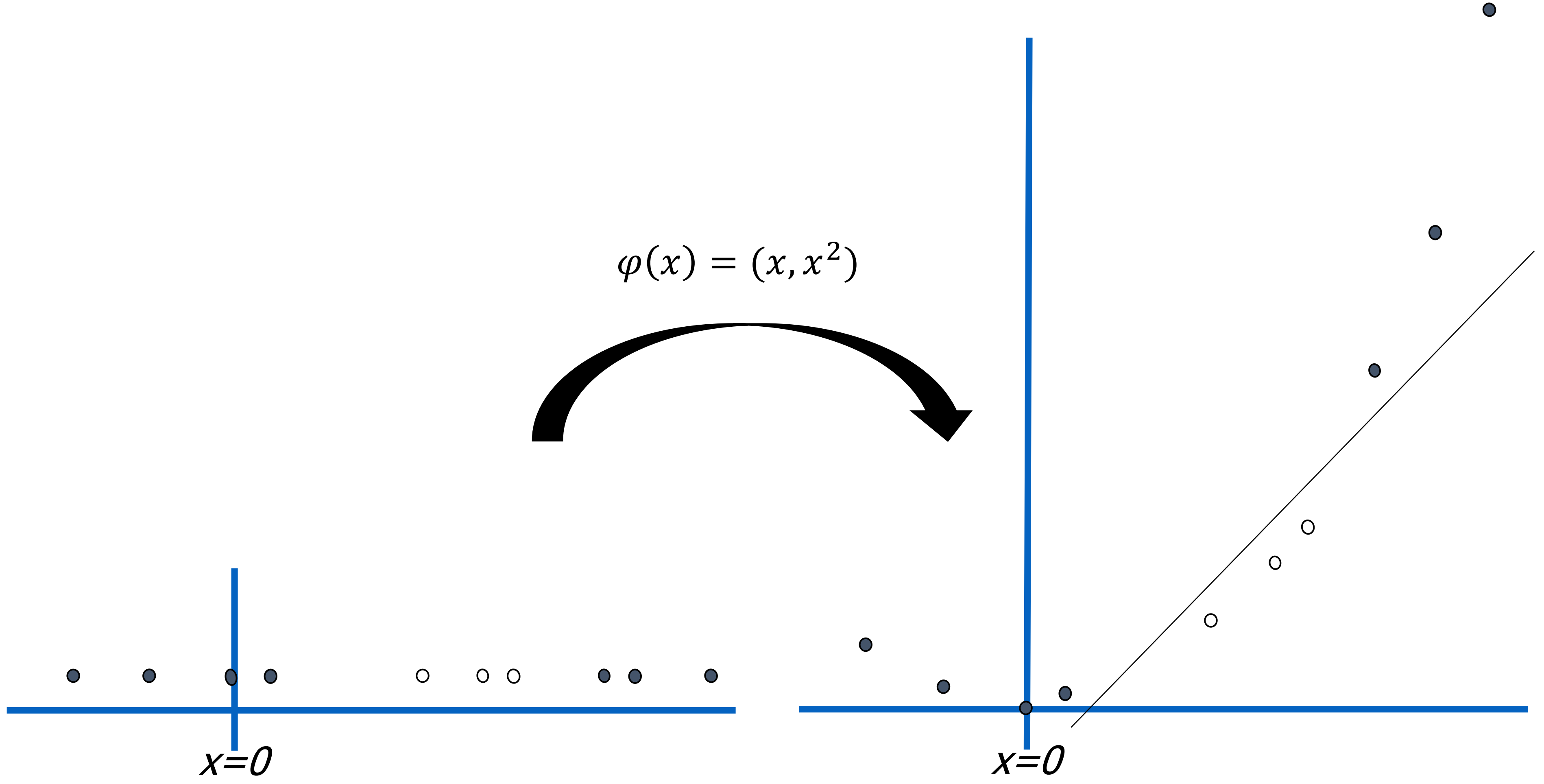
Figure 3: Moving a dataset into a different dimension where instances are separable

Mapping



Mapping

$$\varphi(x) = (x, x^2)$$



Mapping

- What is the problem with mapping?

large time complexity

- The time that takes to map the instances to the new space is not efficient (and therefore not practical)

- We need to find a way to avoid the mapping and still save the benefit of it

Kernel Trick

- Let's assume that we need only the inner product in the mapping space (we will see later that this assumption is correct)
- Meaning, we only want the result of

$$\varphi(x) \cdot \varphi(y)$$

* Where φ is the mapping

- If we can find a function that get the same result “without” the mapping, we will reduce the time complexity
- This function called Kernel, and the Kernel Trick is to avoid the mapping

Kernel Trick – example

- 2-D vectors – $\mathrm{x} = (x_1, x_2)$
- $\varphi(x) = (x_1^2, \sqrt{2} \cdot x_1 x_2, x_2^2)$
- $\varphi(x) \cdot \varphi(y) = ?$

$$x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2$$

$$= (x_1 y_1 + x_2 y_2)^2$$

$$= (x \cdot y)^2 = K(x, y)$$

Kernel Trick – example 2

- 4-D vectors –$\mathrm{x} = (x_1, x_2, x_3, x_4)$
- $\varphi(x) = (1,$

$$\sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2, \sqrt{2} \cdot x_3, \sqrt{2} \cdot x_4,$$
$$x_1^2, x_2^2, x_3^2, x_4^2,$$
$$\sqrt{2} \cdot x_1 x_2, \sqrt{2} \cdot x_1 x_3, \sqrt{2} \cdot x_1 x_4,$$
$$\sqrt{2} \cdot x_2 x_3, \sqrt{2} \cdot x_2 x_4, \sqrt{2} \cdot x_3 x_4)$$

Kernel Trick – example 2

- $\varphi(x) \cdot \varphi(y) =?$

$$\varphi(x) = (1,$$
$$\sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2, \sqrt{2} \cdot x_3, \sqrt{2} \cdot x_4,$$
$$x_1^2, x_2^2, x_3^2, x_4^2,$$
$$\sqrt{2} \cdot x_1 x_2, \sqrt{2} \cdot x_1 x_3, \sqrt{2} \cdot x_1 x_4,$$
$$\sqrt{2} \cdot x_2 x_3, \sqrt{2} \cdot x_2 x_4, \sqrt{2} \cdot x_3 x_4)$$

- 1

Kernel Trick – example 2

- $\varphi(x) \cdot \varphi(y) =$

$$1 + \sum_{i=1}^{4} 2x_i y_i + \sum_{i=1}^{4} x_i^2 y_i^2 + \sum_{i=1}^{3} \sum_{j=i+1}^{4} 2x_i x_j y_i y_j$$

- Lets look at the function $(x \cdot y + 1)^2 = (x \cdot y)^2 + 2x \cdot y + 1$

$$= 1 + \sum_{i=1}^{4} 2x_i y_i + \left(\sum_{i=1}^{4} x_i y_i\right)^2$$

$$= 1 + \sum_{i=1}^{4} 2x_i y_i + \sum_{i=1}^{4} \sum_{j=1}^{4} x_i y_i x_j y_j$$

$$= 1 + \sum_{i=1}^{4} 2x_i y_i + \sum_{i=1}^{4} x_i^2 y_i^2 + \sum_{i=1}^{3} \sum_{j=i+1}^{4} 2x_i x_j y_i y_j$$

- Time complexity – O(d) (We need to calculate only $(x \cdot y + 1)^2$)

Kernel Trick – example 2

- We saw that for

$$
\begin{aligned}
\varphi(x) = (1,&\\
&\sqrt{2}\cdot x_1, \sqrt{2}\cdot x_2, \sqrt{2}\cdot x_3, \sqrt{2}\cdot x_4,\\
&x_1^2, x_2^2, x_3^2, x_4^2,\\
&\sqrt{2}\cdot x_1x_2, \sqrt{2}\cdot x_1x_3, \sqrt{2}\cdot x_1x_4,\\
&\sqrt{2}\cdot x_2x_3, \sqrt{2}\cdot x_2x_4,\\
&\sqrt{2}\cdot x_3x_4)
\end{aligned}
$$

We have the kernel function $(x\cdot y+1)^2$

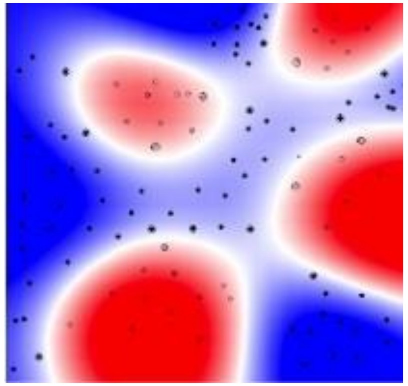
Kernel functions

- There are some known Kernel function
- We don’t need to know the space that the kernel is mapping to

Kernel functions

- Polynomial Kernel with degree d:

$$K(x,y) = (\alpha x^T y + \beta)^d$$



- Radial Basis Function (RBF):

$$K(x,y) = \exp(\frac{-\|x-y\|^2}{2\sigma^2})$$

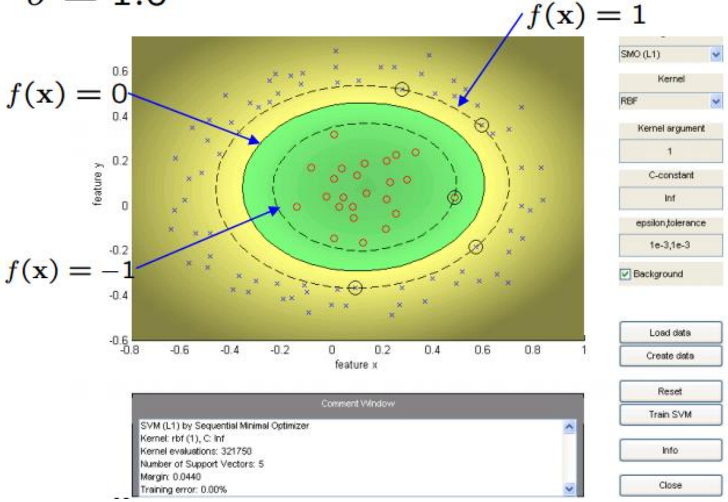
  - Where σ is a parameter
  - We can replace $\frac{1}{2\sigma^2}$ with γ -> $\exp(-\gamma\|x-y\|^2)$

  - The radius of the "balls" is determined by the parameter $\gamma = \frac{1}{2\sigma^2}$
    - A smaller γ means a larger radius, a lower "model complexity"
    - A larger γ means a smaller radius, a finer grain coverage but may lead to an overfitt

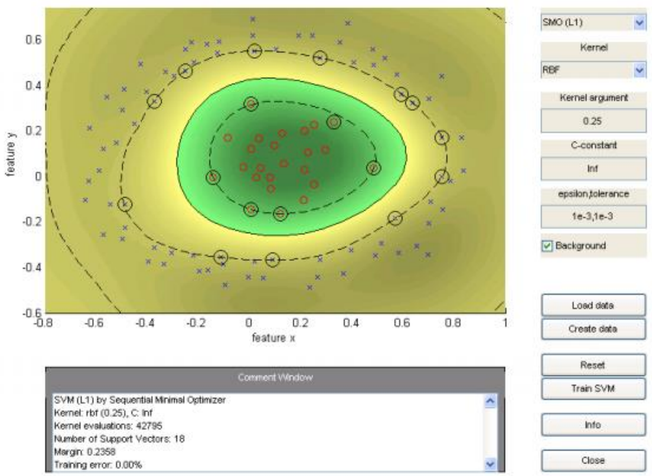
* You can find more function in this [link]

RBF Kernel SVM Example

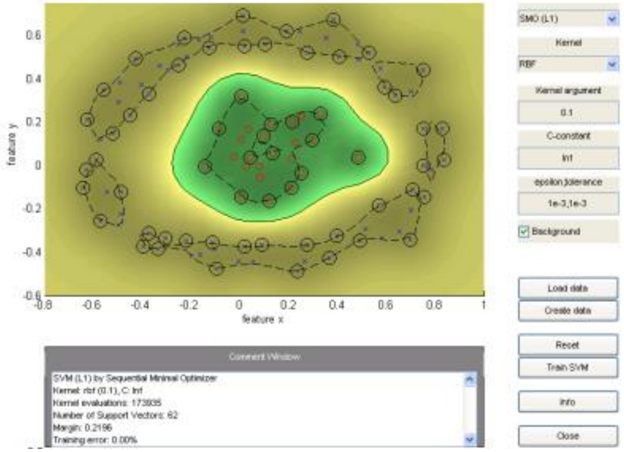
$\sigma = 1.0$



$\sigma = 0.25$



$\sigma = 0.1$



Notice that:
Decreasing sigma, moves towards nearest neighbor classifier

Kernel functions - RBF

- What is the dimension for the space in RBF kernel?
- Let's assume $\sigma = 1/\sqrt{2}$ and that the dimension of the original vectors is 2

$$K(x, y) = exp(-\|x - y\|^2)$$

$$= exp(-(x_1 - y_1)^2 - (x_2 - y_2)^2)$$

$$= exp(-x_1^2 + 2x_1y_1 - y_1^2 - x_2^2 + 2x_2y_2 - y_2^2)$$

$$= exp(-\|x\|^2)exp(-\|y\|^2)exp(2x^Ty)$$

Using Taylor series (you can check if you want…)

$$= exp(-\|x\|^2)exp(-\|y\|^2)\sum_{n=0}^{\infty}\frac{(2x^Ty)^n}{n!}$$

- Now, what is the dimension?

Kernel Trick – summarize

- We can check if non separate data is separate in higher dimension
- Mapping to higher dimension is not efficient
- We could calculate the result of the $\varphi(x) \cdot \varphi(y)$ without calculating the mapping itself if we had a function that give the same result

  * even if the mapping space is infinite
- This called the Kernel Trick
- **We still have to see why we only need the result of the dot product $\boldsymbol{\varphi(x) \cdot \varphi(y)}$**

From Perceptron to Kernel Perceptron

- Regular Perceptron algorithm:
  - Initialize weights to some small random number
  - Repeat until convergence (no error = no weight update):
    - For each x_d in D compute:
      - $o_d = sgn(w \cdot x_d)$
      - For each w_i do:
        - $\Delta w_i = -\eta(o_d - t_d)x_{id}$
        - $w_i = w_i + \Delta w_i$

Which perceptron is it?

From Perceptron to Kernel Perceptron

- Non-Linear Perceptron algorithm:
  - Initialize weights to some small random number
  - Repeat until convergence (no error = no weight update):
    - For each x_d in D compute:
      - $o_d = sgn\left(w \cdot \varphi(x_d)\right)$
      - For each w_i do:
        - $\Delta w_i = -\eta(o_d - t_d)\varphi(x_d)_i$
        - $w_i = w_i + \Delta w_i$

Problem?

From Perceptron to Kernel Perceptron

- Going back to regular Perceptron algorithm:
  - Initialize weights to some small random number
  - Repeat until convergence (no error = no weight update):
    - For each x_d in D compute:
      - $o_d = sgn(w \cdot x_d)$
      - For each w_i do:
        - $\Delta w_i = -\eta(o_d - t_d)x_{id}$
        - Update $w_i = w_i + \Delta w_i$

We add a small
part of x_d
if it is misclassified

From Perceptron to Kernel Perceptron

- In practice we only need some of the instances
- Why?

$$\Delta w_i = -\eta(o_d - t_d)x_{id}$$

0	t	o-t	x_i	Δw_i	$x_i \cdot w_i$
-1	+1	<0	>0	>0	increased
-1	+1	<0	<0	<0	increased
+1	-1	>0	>0	<0	decreased
+1	-1	>0	<0	>0	decreased

From Perceptron to Kernel Perceptron

- Hence, we always add a fraction of $t_d x_d$ (if its misclassified)
- We get in the end, that the weights are linear combination of some training examples

$$w = \sum_{d} \alpha_d t_d x_d$$

- Where $\alpha_d \geq 0$

From Perceptron to Kernel Perceptron

- Now, we can convert the decision function:

$$f(x) = \vec{w} \cdot \vec{x} = \left(\sum_{d} \alpha_d t_d x_d \right) \cdot \vec{x} = \sum_{d} \alpha_d t_d (\vec{x}_d \cdot \vec{x})$$

From Perceptron to Kernel Perceptron

$$f(x) = \sum_d \alpha_d t_d (\vec{x}_d \cdot \vec{x})$$

- It is look like instance base…
- But we only need the instances $\vec{x}_d$ whose $\alpha_d \neq 0$
- These are called **“support vectors”**
- In order to use this form we need to rewrite the update function:

$$\text{if} \ \left(t_i \sum_d \alpha_d t_d (\vec{x}_d \cdot \vec{x}_i) \right) < 0:$$
$$\alpha_i = \alpha_i + \eta$$

From Perceptron to Kernel Perceptron

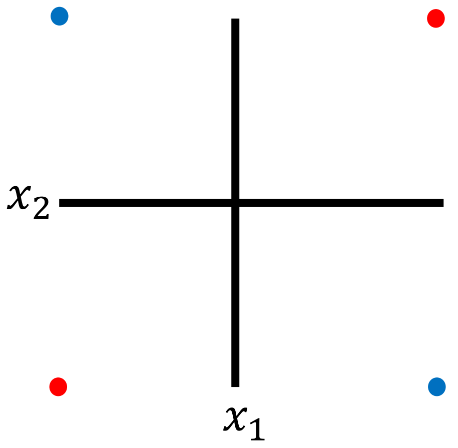
- The Dual Perceptron algorithm:
  - Initialize each α_i to zero
  - Repeat until convergence (no error):
    - For each x_i in D compute:
      - $o_i = \sum_{d \in D} \alpha_d t_d (\vec{x}_d \cdot \vec{x}_i)$
      - If $t_i o_i < 0$
        - $\alpha_i = \alpha_i + \eta$

From Perceptron to Kernel Perceptron

- The Kernel Perceptron algorithm:
  - Initialize each α_i to zero
  - Repeat until convergence (no error):
    - For each x_i in D compute:
      - $o_i = \sum_{d \in D} \alpha_d t_d \big(\varphi(\vec{x}_d) \cdot \varphi(\vec{x}_i)\big) = \sum_{d \in D} \alpha_d t_d K(\vec{x}_d, \vec{x}_i)$
      - If $t_i o_i < 0$
        - $\alpha_i = \alpha_i + \eta$
- This is the first step toward SVM

Kernel Perceptron – Example

x_1	x_2	t
1	1	1
-1	1	-1
-1	-1	1
1	-1	-1



- $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j) = (x_i \cdot x_j)^2$

K	x^1	x^2	x^3	x^4
x^1	4	0	4	0
x^2	0	4	0	4
x^3	4	0	4	0
x^4	0	4	0	4

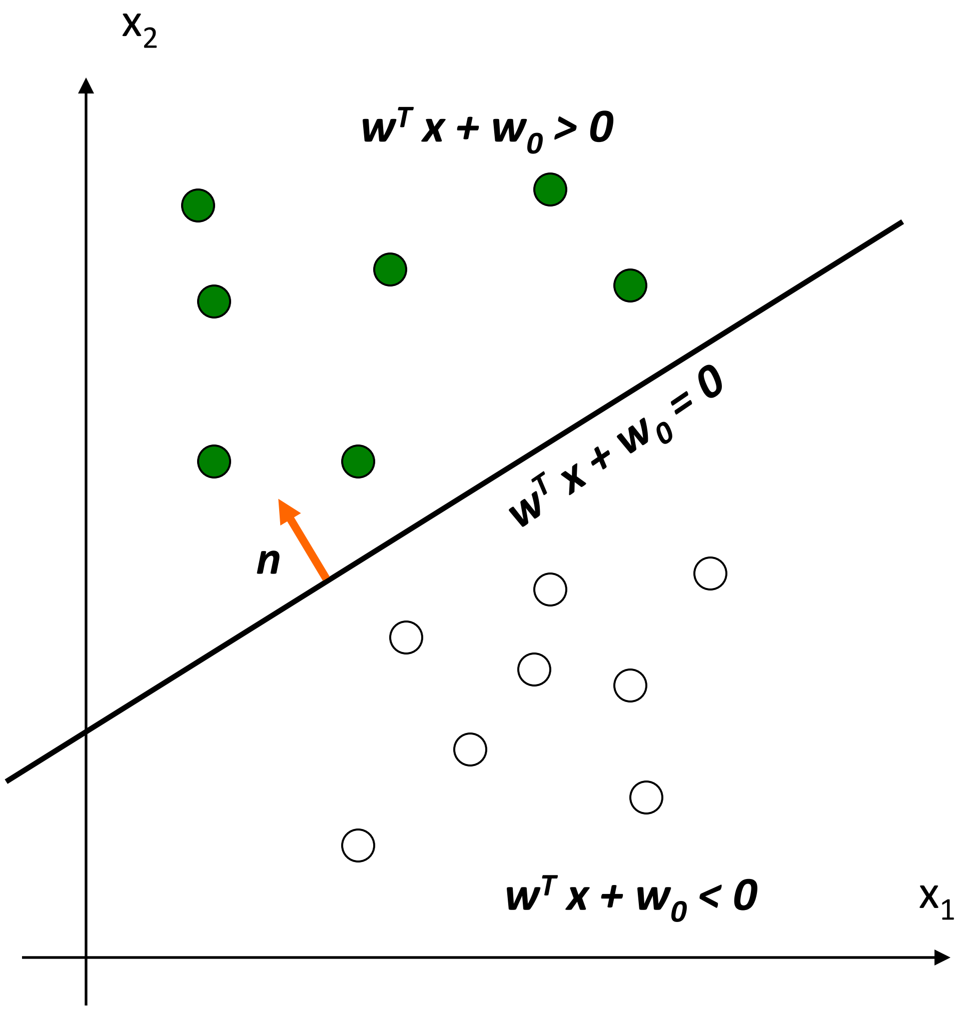
- Init ($\eta = 1$):
  - $\alpha = [\alpha^1, \alpha^2, \alpha^3, \alpha^4] = [0,0,0,0]$
- i=1
  - $\sum_{d \in D} \alpha_d t_d K(\vec{x}_d, \vec{x}_i) =$
    $0 * 4 - 0 * 0 + 0 * 4 - 0 * 0 = 0$
    $sgn(0) = -1 \rightarrow \alpha^1 \mathrel{+}= 1$
- i=2
  - $\sum_{d \in D} \alpha_d t_d K(\vec{x}_d, \vec{x}_i) =$
    $1 * 0 - 0 * 4 + 0 * 0 - 0 * 4 = 0$
    $sgn(0) = -1$
- i=3
  - $\sum_{d \in D} \alpha_d t_d K(\vec{x}_d, \vec{x}_i) =$
    $1 * 4 - 0 * 0 + 0 * 4 - 0 * 0 = 4$
    $sgn(4) = 1$
- i=4
  - $\sum_{d \in D} \alpha_d t_d K(\vec{x}_d, \vec{x}_i) =$
    $1 * 0 - 0 * 4 + 0 * 0 - 0 * 4 = 0$
    $sgn(0) = -1$
- i=1
  - $\sum_{d \in D} \alpha_d t_d K(\vec{x}_d, \vec{x}_i) =$
    $1 * 4 - 0 * 0 + 0 * 4 - 0 * 0 = 4$
    $sgn(4) = 1$

The goal

- Find linear classifier that can separate the data set
- SVM based on 3 ideas :
  - The Kernel trick – map data to high dimensional space where it is easier to classify with linear decision surfaces √
  - Max Margin – for linearly separable problem, the maximal margin hyperplane is the optimal linear classifier X
  - Soft Margin and Regularization – extend the above definition for non-linearly separable problems. introduce term for misclassifications X

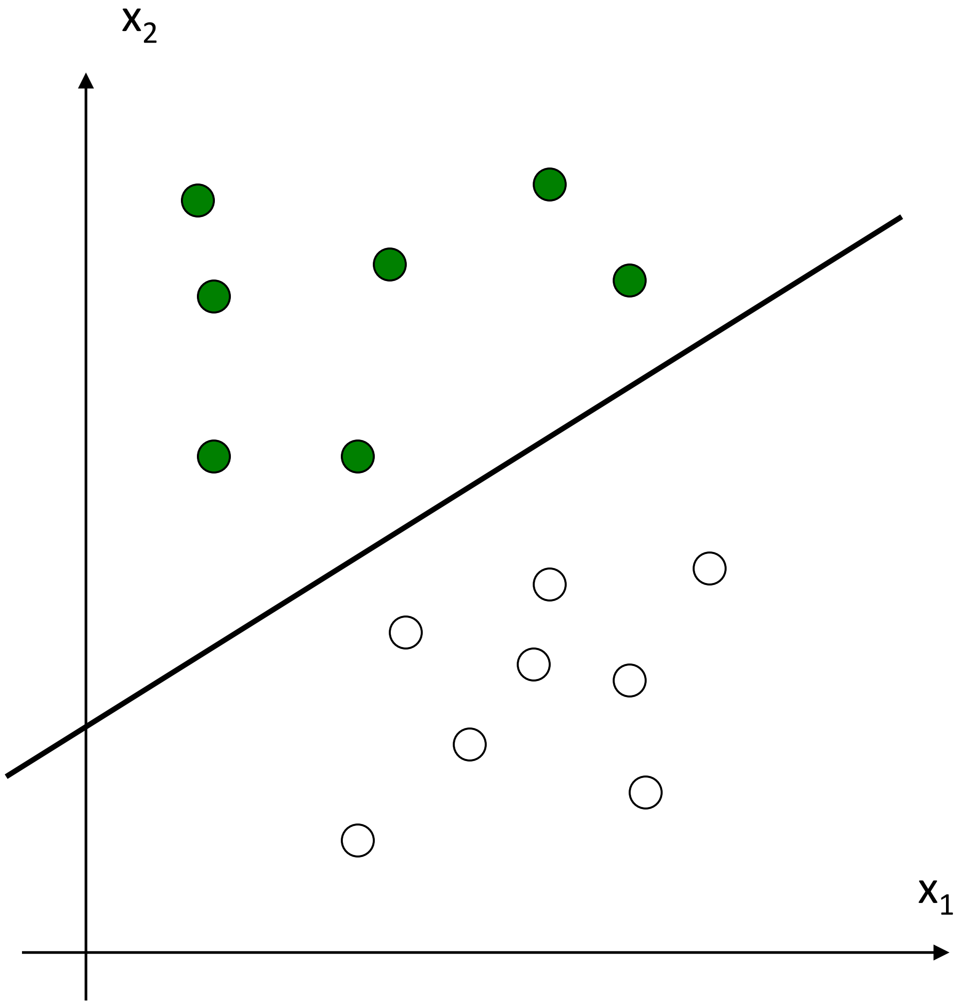
How does it look like?

- f(x) is a linear function
$$f(x) = w^T x + w_0$$
- A hyper-plane in the feature space
- (Unit-length) normal vector of the hyper-plane:
$$n = \frac{w}{||w||}$$



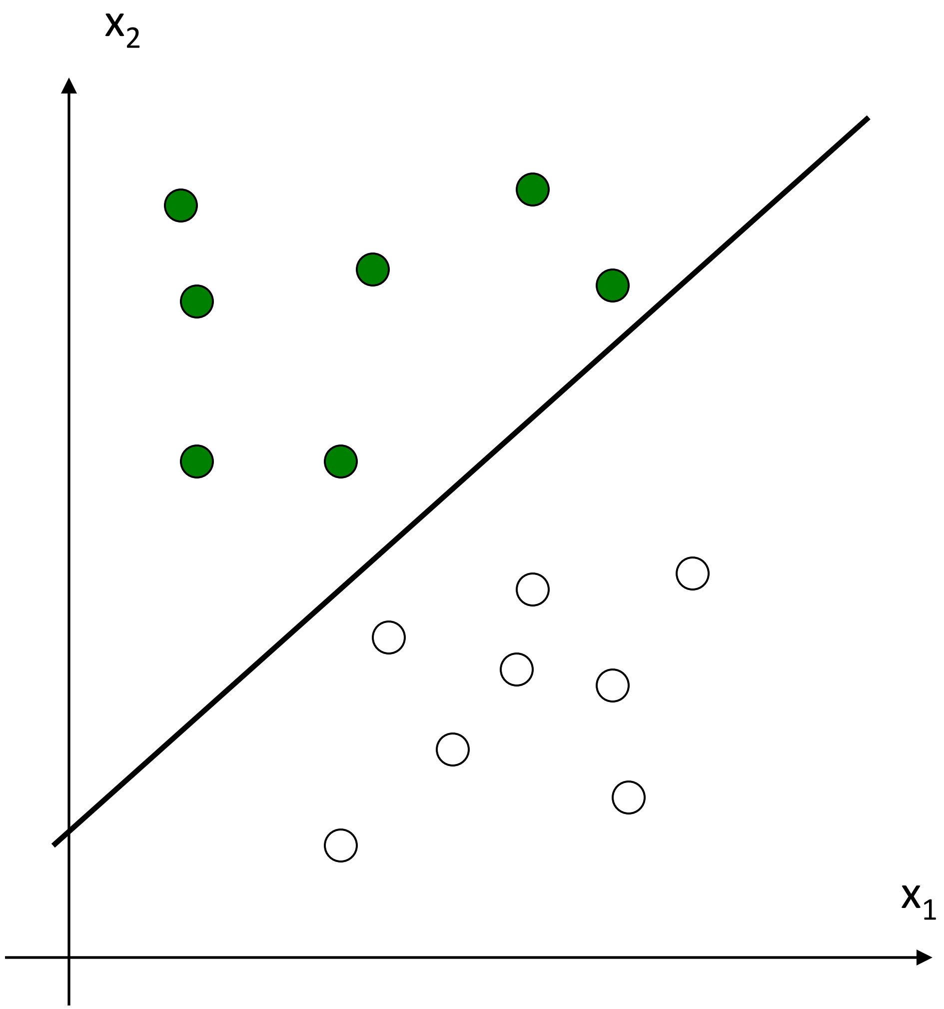
How does it look like?

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



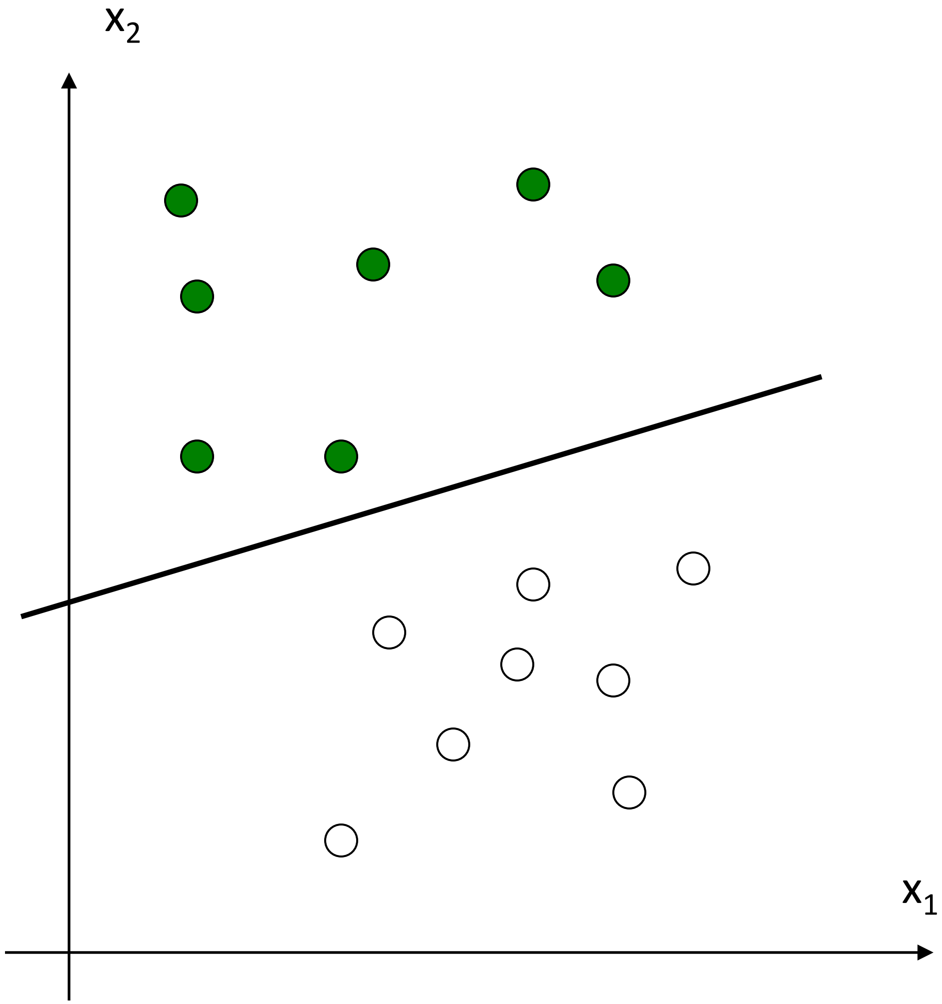
How does it look like?

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



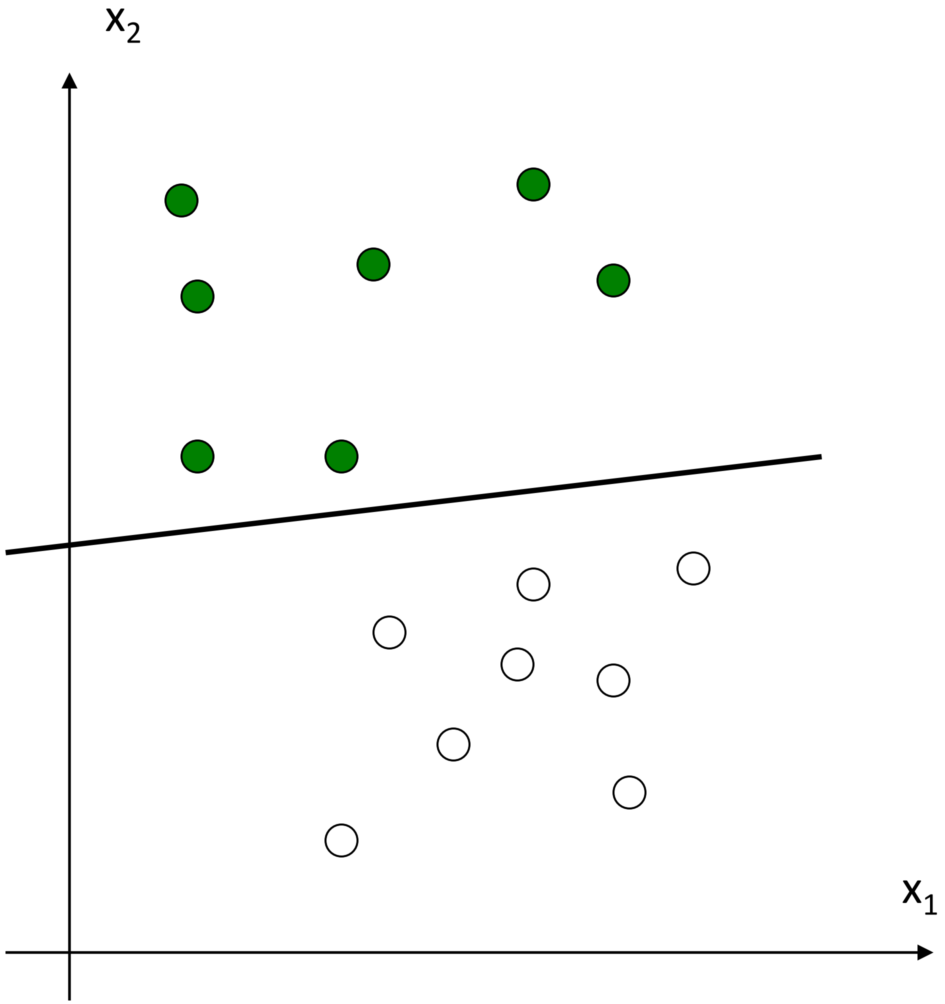
How does it look like?

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



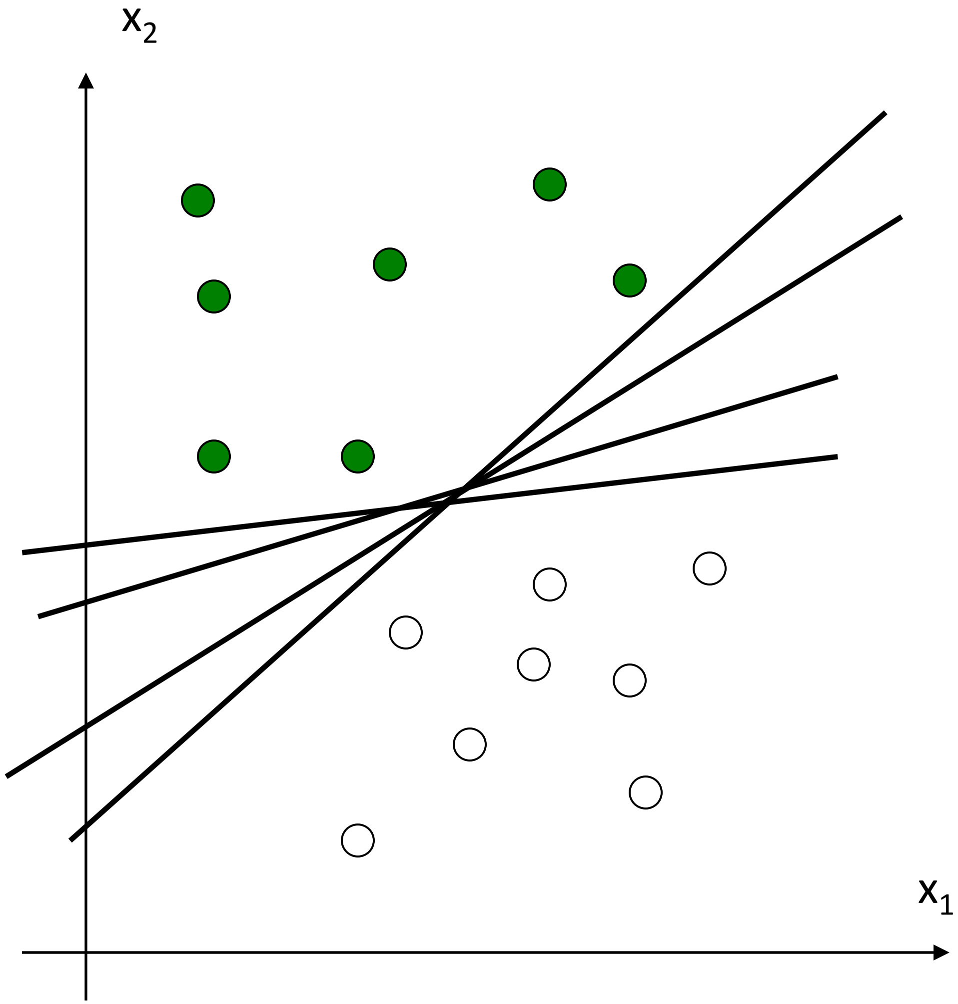
How does it look like?

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



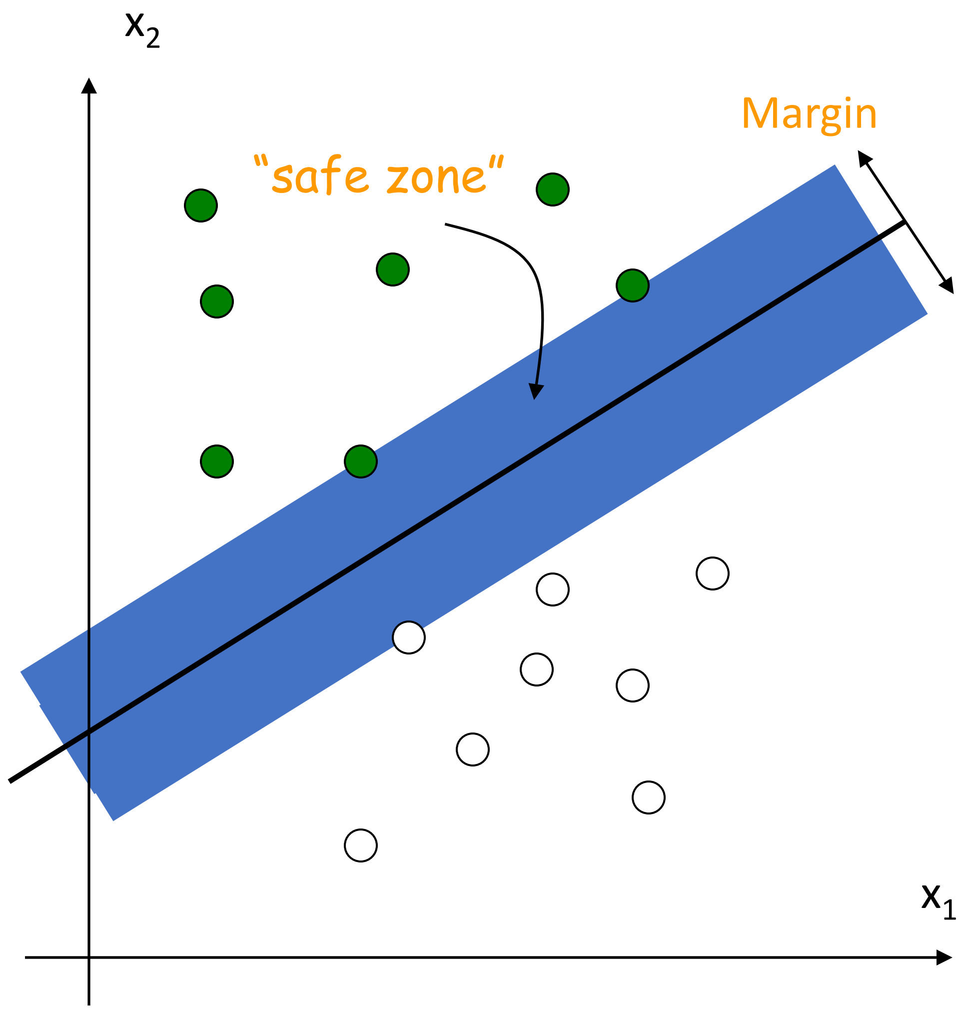
How does it look like?

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!
- Which one is the best?



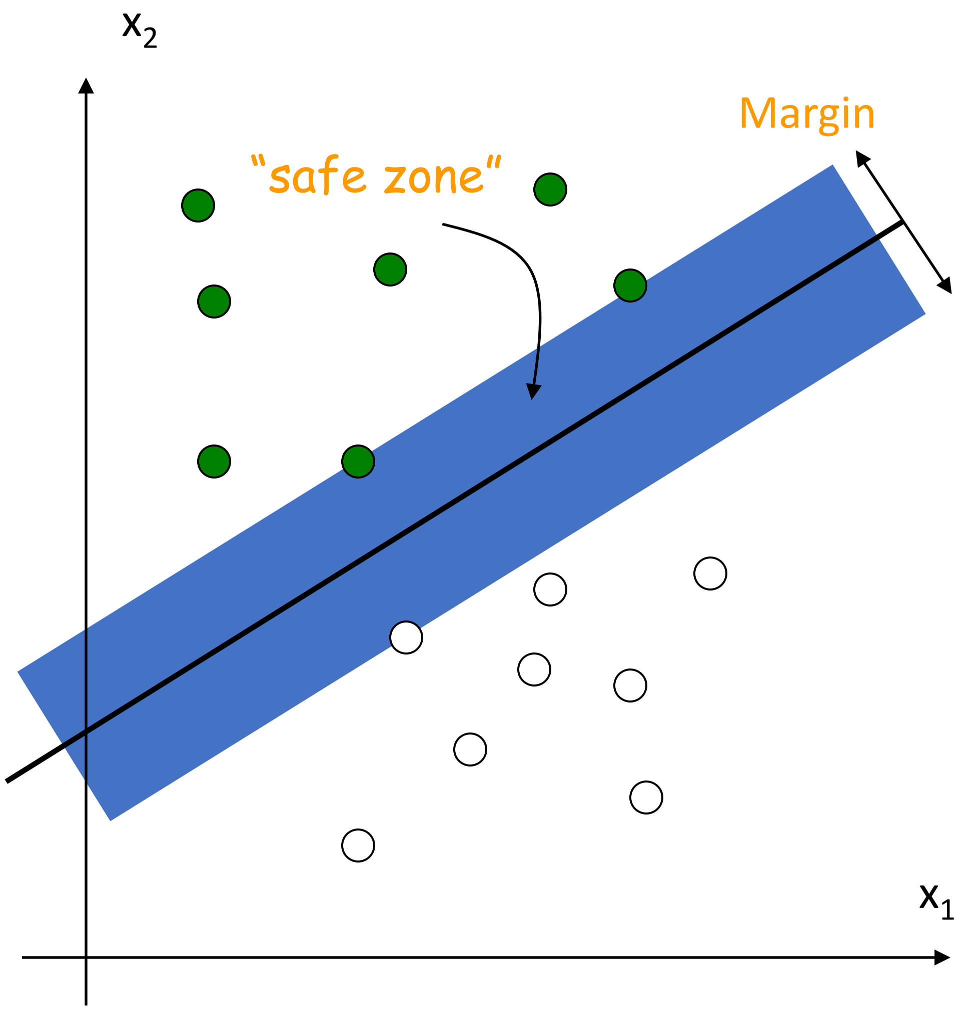
How does it look like?

- The linear classifier with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point



How does it look like?

- Why is this the best?
  - Robust to outliers and thus strong generalization ability
  - If there are no points near the decision surface, then there are no uncertain classification decisions



How does it look like?

- Given a set of data points:

$$\{(x_d, t_d)\}, d = 1,2,\cdots,n$$

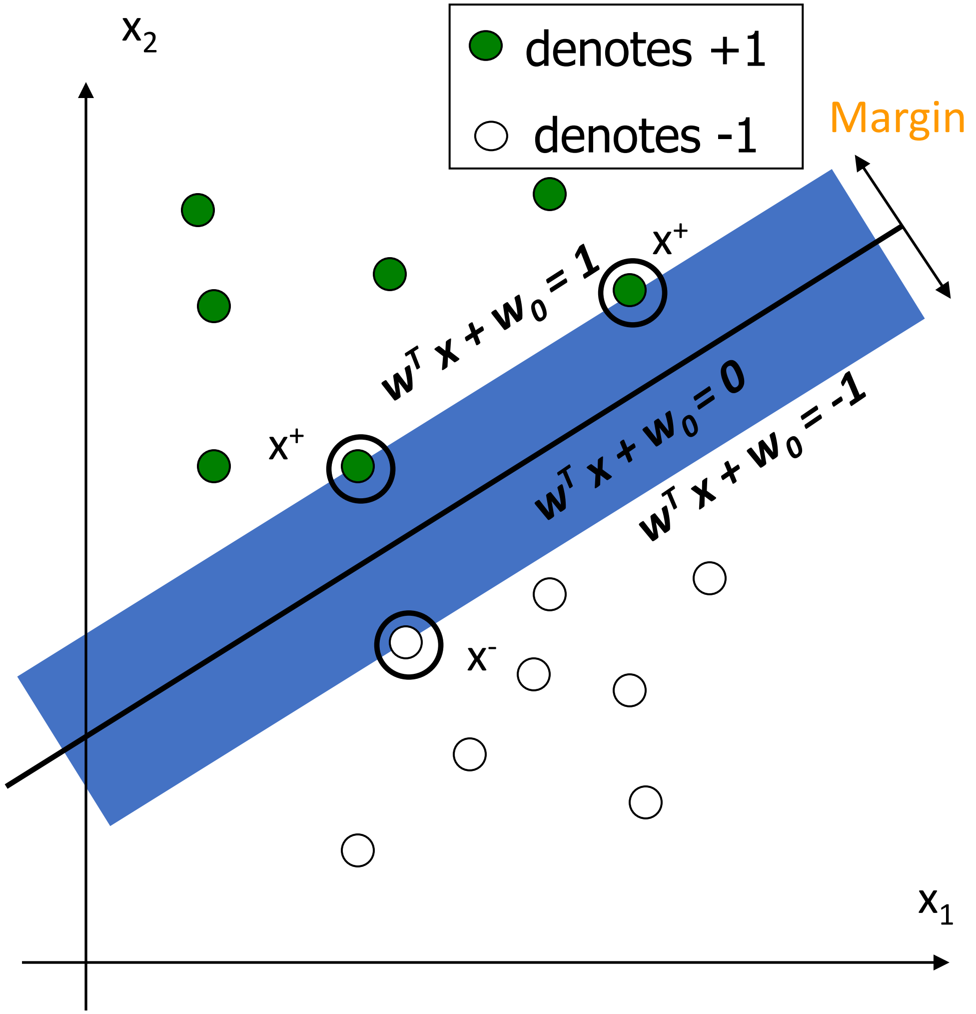
For $t_d = +1,\ w^T x_d + w_0 > 0$

For $t_d = -1,\ w^T x_d + w_0 < 0$

- With a scale transformation on both w and w_0, the above is equivalent to

For $t_d = +1,\ w^T x_d + w_0 \geq 1$

For $t_d = -1,\ w^T x_d + w_0 \leq -1$



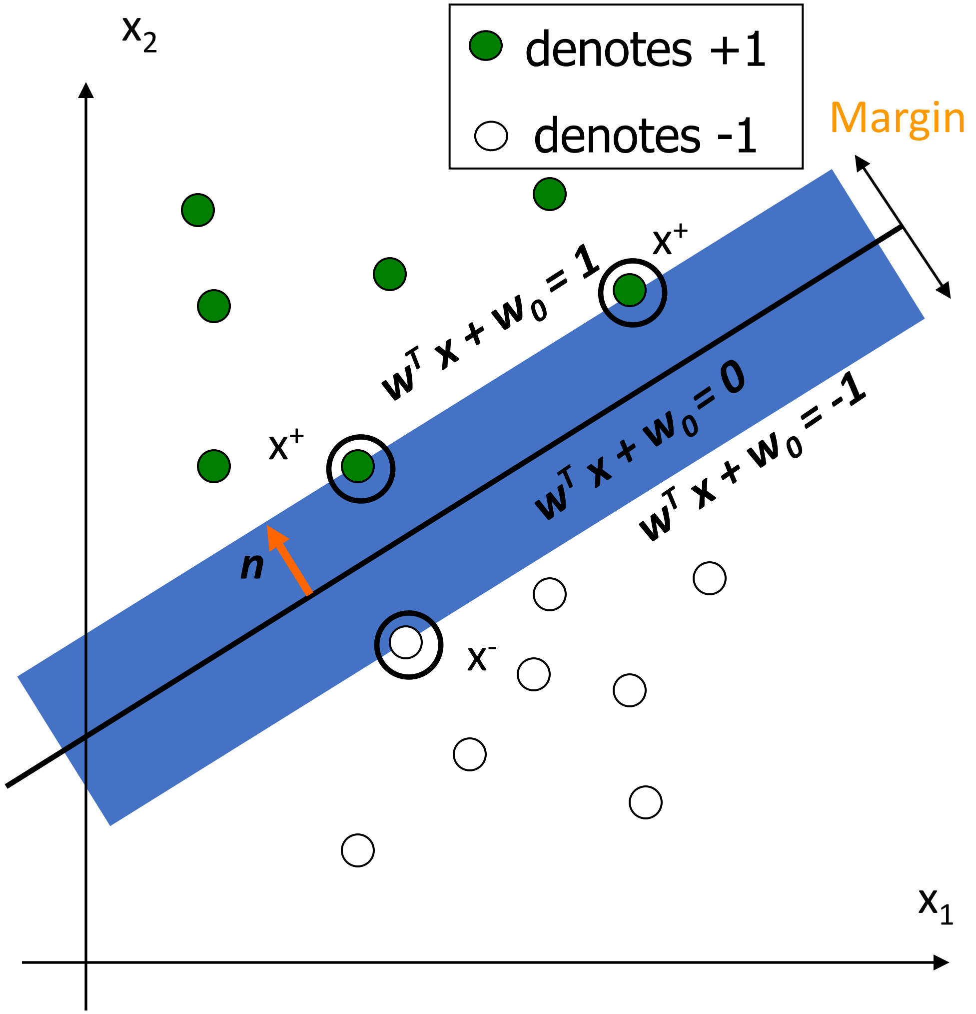
How does it look like?

- We know that:

$$w^T x^+ + w_0 = +1$$
$$w^T x^- + w_0 = -1$$

- The margin width is:

$$\mathrm{M} = (x^+ - x^-) \cdot n$$
$$= (x^+ - x^-) \cdot \frac{w}{\|w\|}$$
$$= \frac{2}{\|w\|}$$



How does it look like?

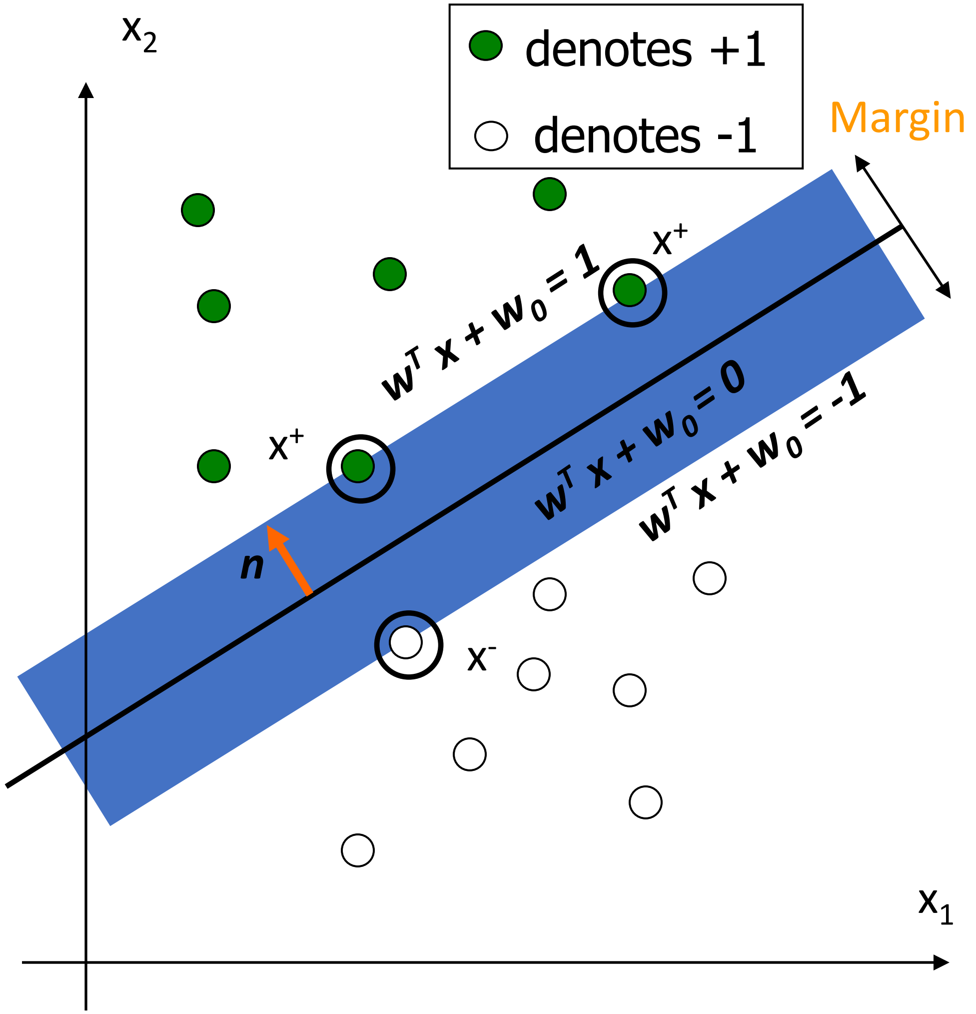
- Our goal:

$$\text{Maximize } \frac{2}{\|w\|}$$

- Subject to:

$$\text{For } t_d = +1, w^T x_d + w_0 \geq 1$$

$$\text{For } t_d = -1, w^T x_d + w_0 \leq -1$$



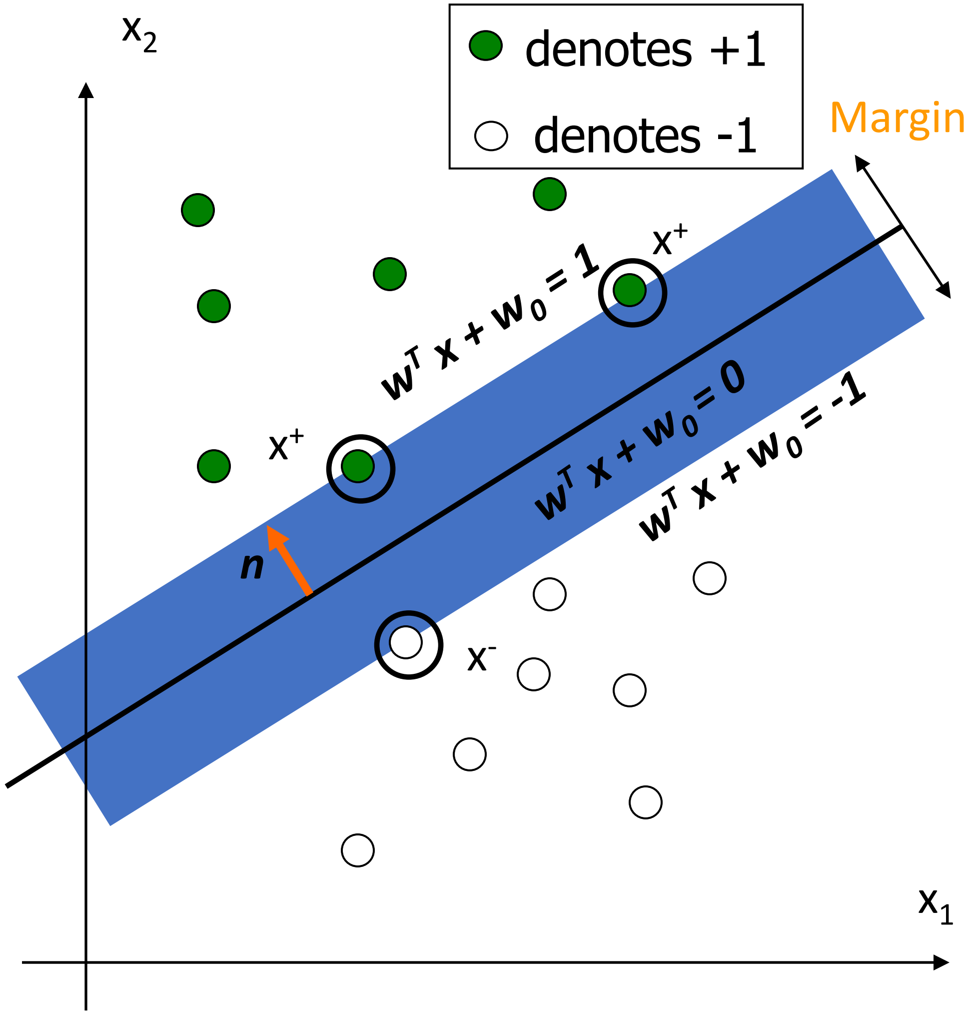
How does it look like?

- Our goal:

$$\text{Maximize } \frac{2}{\|w\|}$$

- Subject to:

$$t_d(w^T x_d + w_0) \geq 1$$



How does it look like?

- Our goal:

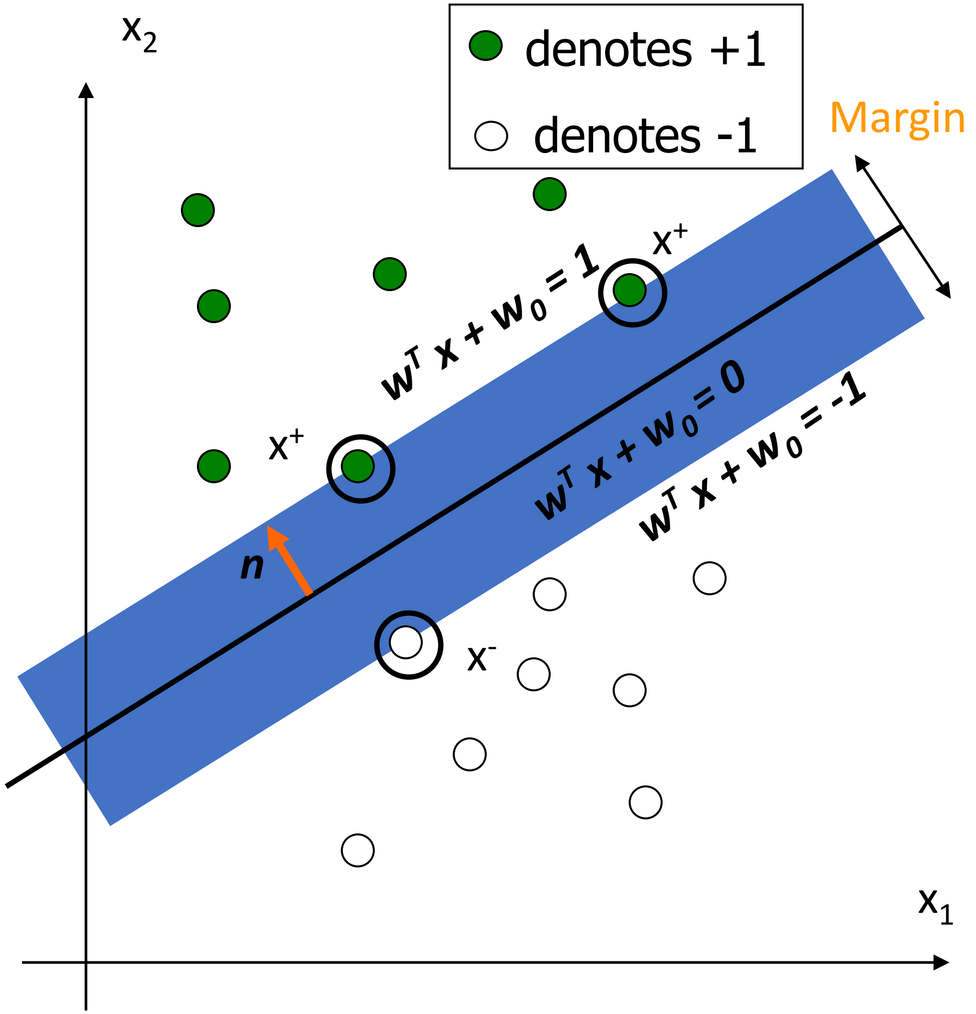
$$\text{Maximize } \frac{2}{\|w\|}$$

Is equivalent to find:

$$\text{Minimize } \frac{1}{2} \cdot \|w\|^2$$

- Subject to:

$$t_d(w^T x_d + w_0) \geq 1$$



Optimization problem

- Minimize

$$\frac{1}{2} \cdot \|w\|^2$$

- Subject to:

$$t_d(w^T x_d + w_0) \geq 1$$

- This is an optimization problem can be solved with Quadratic Programing

Optimization problem

- Minimize

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

- Subject to:

$$t_d(w^T x_d + w_0) \geq 1$$

$$\Downarrow$$

- Minimize

$$\min_{w,w_o} \max_{\alpha_d} L(w, w_0, \alpha_d) = \min_{w,w_o} \max_{\alpha_d} \frac{1}{2}\|w\|^2 - \sum_d \alpha_d(t_d(w^T x_d + w_0) - 1)$$

- Subject to:

$$\alpha_d \geq 0$$

Optimization problem

- Minimize

$$\min_{w,w_o} \max_{\alpha_d} L(w, w_0, \alpha_d) = \min_{w,w_o} \max_{\alpha_d} \frac{1}{2}\|w\|^2 - \sum_d \alpha_d (t_d(w^T x_d + w_0) - 1)$$

- Subject to:

$$\alpha_d \geq 0$$

- Find $\nabla w, \nabla w_0$:

$$\nabla w = 0 \; \rightarrow w = \sum_d \alpha_d t_d x_d$$

$$\nabla w_0 = 0 \; \rightarrow \sum_d \alpha_d t_d = 0$$

Optimization problem

- Minimize

$$\min_{w,w_o}\max_{\alpha_d} L(w,w_0,\alpha_d)=\min_{w,w_o}\max_{\alpha_d}\frac{1}{2}\|w\|^2-\sum_d \alpha_d(t_d(w^T x_d+w_0)-1)$$

- Subject to:

$$\alpha_d \ge 0$$

- Dual - maximize

$$\sum_d \alpha_d - 1/2\sum_d\sum_e \alpha_d\alpha_e t_d t_e x_d^T x_e$$

$$\sum_d \alpha_d t_d = 0, \alpha_d \ge 0$$

Optimization problem

- Given a solution $\alpha_1 \ldots \alpha_n$ to the dual problem, the solution to the primal is:

$$w = \sum_d \alpha_d t_d x_d$$

$$w_0 = t_k - \sum_d \alpha_d t_d x_d^T x_k$$

- For any $\alpha_k > 0$

Optimization problem

- What can we achieve from the duality that wasn't in the primal?

$$\sum_d \alpha_d - 1/2 \sum_d \sum_e \alpha_d \alpha_e t_d t_e x_d^T x_e$$

- We can switch to the mapping space "The Kernel Trick"

$$\sum_d \alpha_d - 1/2 \sum_d \sum_e \alpha_d \alpha_e t_d t_e \, K(x_d, x_e)$$

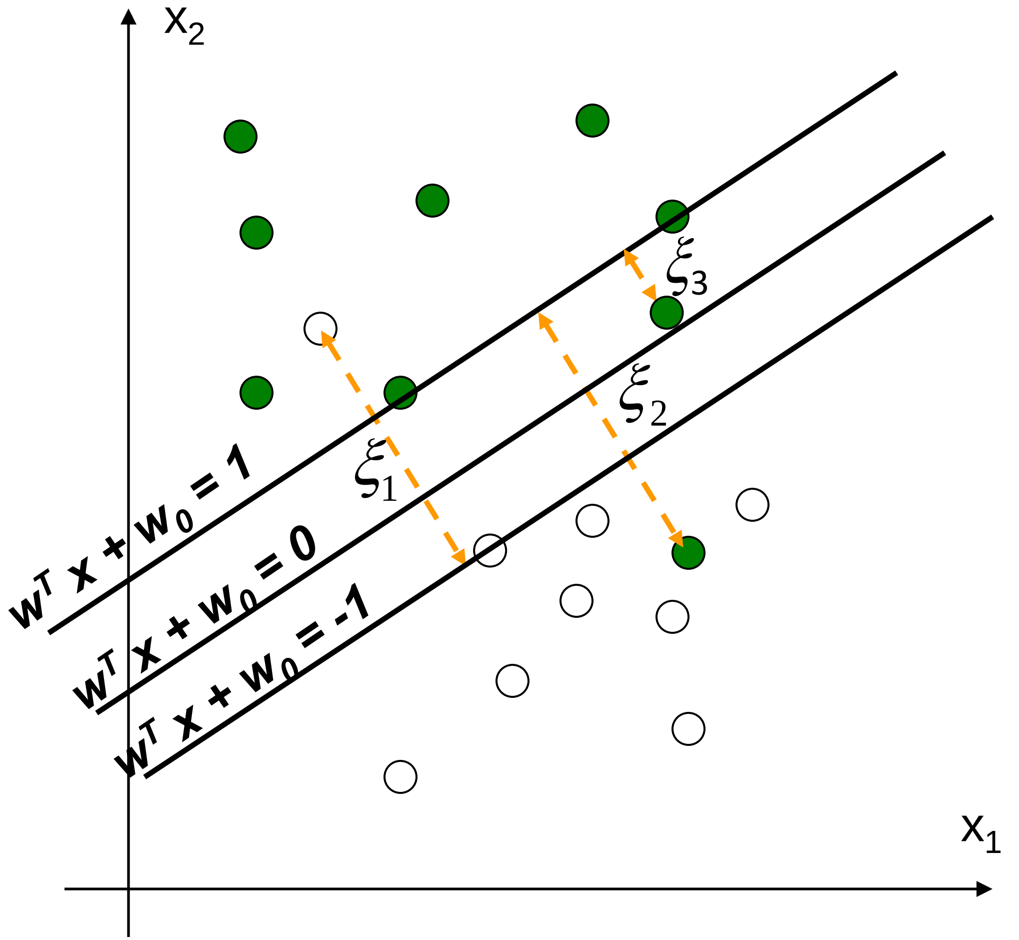
The goal

- Find linear classifier that can separate the data set
- SVM based on 3 ideas :
  - The Kernel trick – map data to high dimensional space where it is easier to classify with linear decision surfaces √
  - Max Margin – for linearly separable problem, the maximal margin hyperplane is the optimal linear classifier √
  - Soft Margin and Regularization – extend the above definition for non-linearly separable problems. introduce term for misclassifications X

Soft Margin

- What if the data is not linear separable? (noisy data, outliers, etc.)

- Slack variables ξ_d can be added to allow margin violations (not necessarily misclassification) of difficult or noisy data points



Soft Margin – optimization problem

- Minimize

$$\frac{1}{2}\|w\|^2 + \gamma \sum_d \xi_d$$

- Subject to:

$$t_d(w^T x_d + w_0) \geq 1 - \xi_d \quad \xi_d \geq 0$$

⬇

- Minimize

$$\min_{w,w_o,\xi_d} \max_{\alpha_d,\mu_d} L(w, w_0, \xi_d, \alpha_d, \mu_d) =$$

$$\min_{w,w_o,\xi_d} \max_{\alpha_d,\mu_d} \frac{1}{2}\|w\|^2 + \gamma \sum_d \xi_d - \sum_d \alpha_d(t_d(w^T x_d + w_0) - 1 + \xi_d) - \sum_d \mu_d \xi_d$$

- Subject to:

$$\alpha_d \geq 0 \quad \mu_d \geq 0$$

Soft Margin – optimization problem

- Minimize

$$\min_{w,w_o,\xi_d} \max_{\alpha_d,\mu_d} L(w, w_0, \xi_d, \alpha_d, \mu_d) =$$

$$\min_{w,w_o,\xi_d} \max_{\alpha_d,\mu_d} \frac{1}{2}\|w\|^2 + \gamma \sum_d \xi_d - \sum_d \alpha_d (t_d (w^T x_d + w_0) - 1 + \xi_d) - \sum_d \mu_d \xi_d$$

- Subject to:

$$\alpha_d \geq 0 \quad \mu_d \geq 0$$

- Dual - maximize

$$\sum_d \alpha_d - 1/2 \sum_d \sum_e \alpha_d \alpha_e t_d t_e \varphi(x_d)^T \varphi(x_e)$$

$$\sum_d \alpha_d t_d = 0 \quad 0 \leq \alpha_d \leq \gamma$$

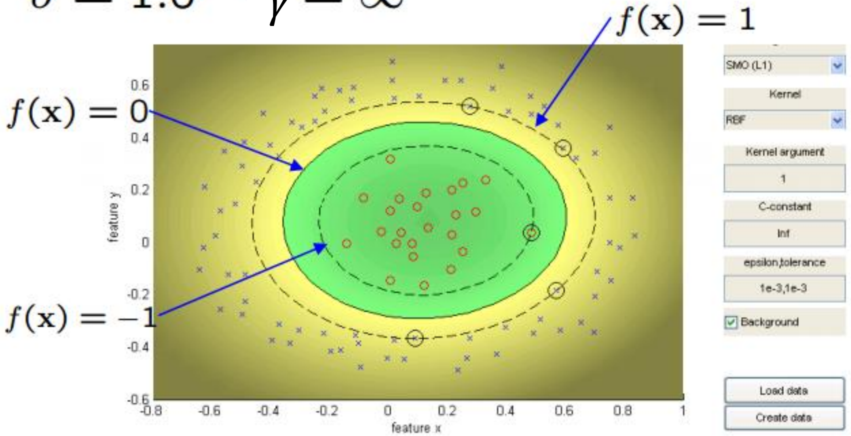
Soft Margin – optimization problem

- The parameter γ balance between the violation penalty and $\frac{1}{2}||w||^2$
- A smaller γ?
  - Means larger margin, a lower "model complexity"
- A larger γ?
  - Means less tolerance to violations, but may lead to an overfit
- As $\gamma \rightarrow \infty$?

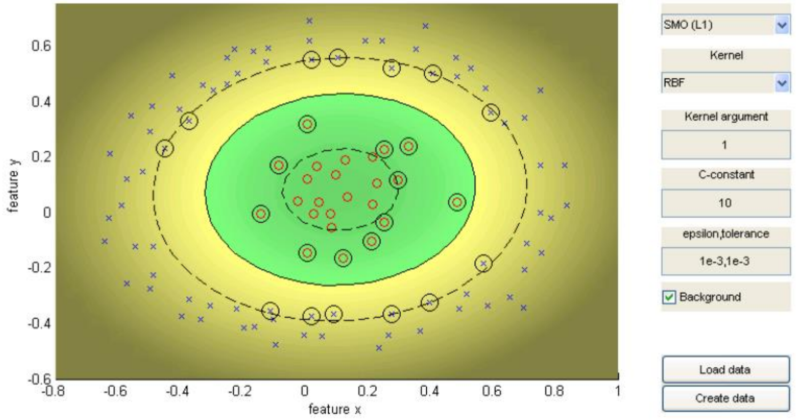
we get closer to the hard-margin solution

RBF Kernel SVM Example

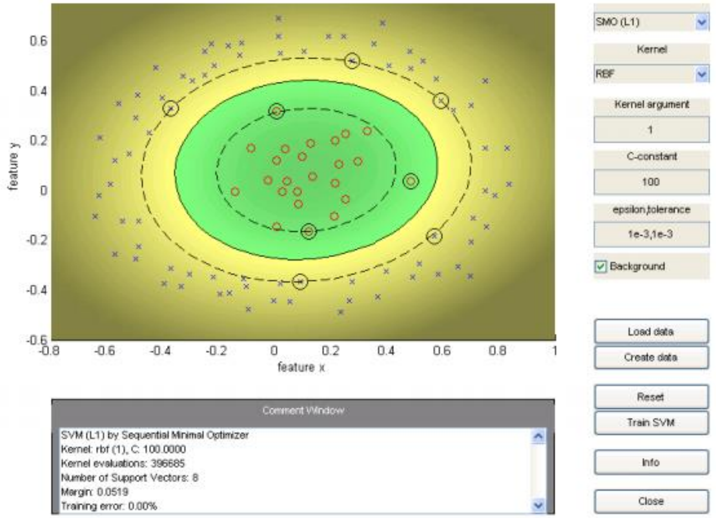
$\sigma = 1.0 \qquad \gamma = \infty$



$\sigma = 1.0 \qquad \gamma = 10$



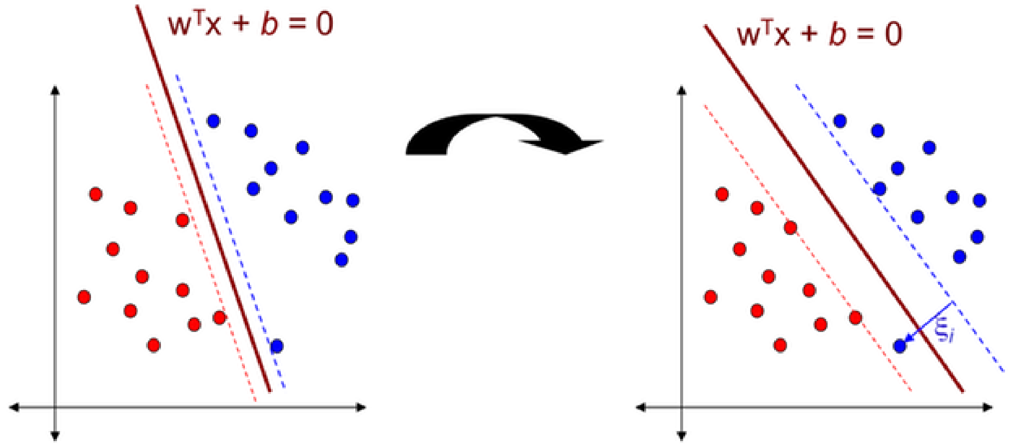
$\sigma = 1.0 \qquad \gamma = 100$

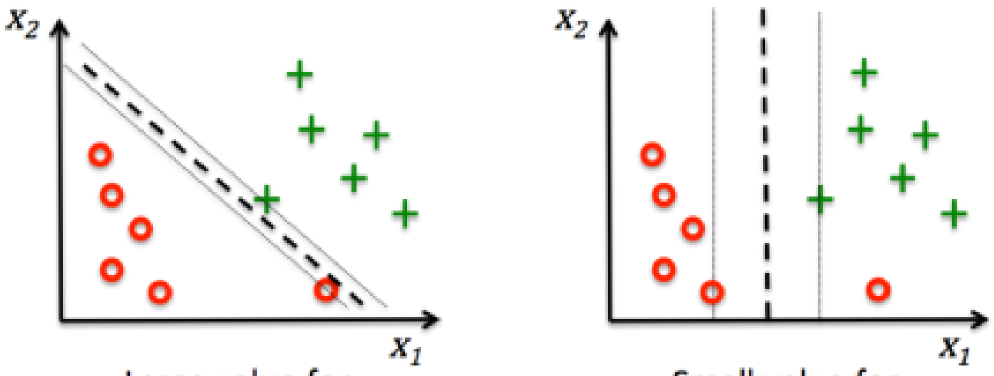


Notice that:

Decrease γ, gives wider (soft) margin

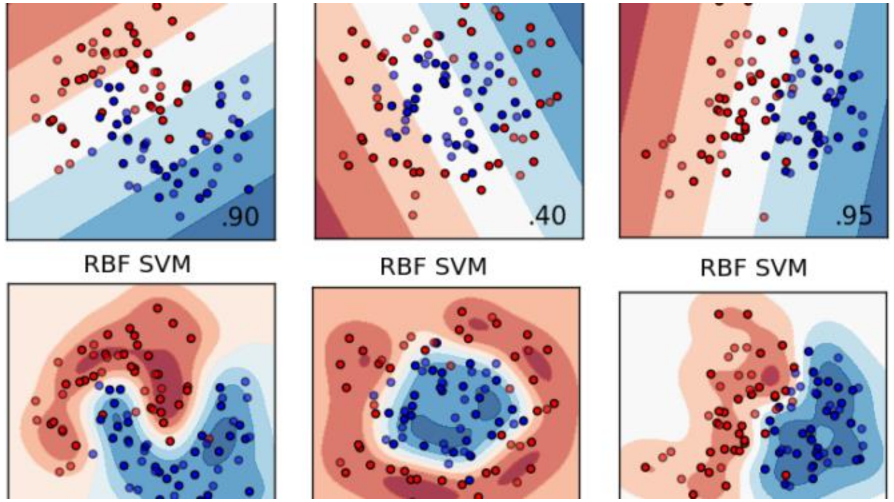
Examples





Large value for parameter *C*

Small value for parameter *C*



The goal

- Find linear classifier that can separate the data set
- SVM based on 3 ideas :
  - The Kernel trick – map data to high dimensional space where it is easier to classify with linear decision surfaces √
  - Max Margin – for linearly separable problem, the maximal margin hyperplane is the optimal linear classifier √
  - Soft Margin and Regularization – extend the above definition for non-linearly separable problems. introduce term for misclassifications √

Questions

?