



ARISTOTLE UNIVERSITY OF THESSALONIKI



FACULTY OF ENGINEERING

Pattern Recognition & Machine Learning

Support Vector Machines and Kernels

Panagiotis C. Petrantonakis

Assistant Professor

Dept. of Electrical and Computer Engineering

ppetrant@ece.auth.gr

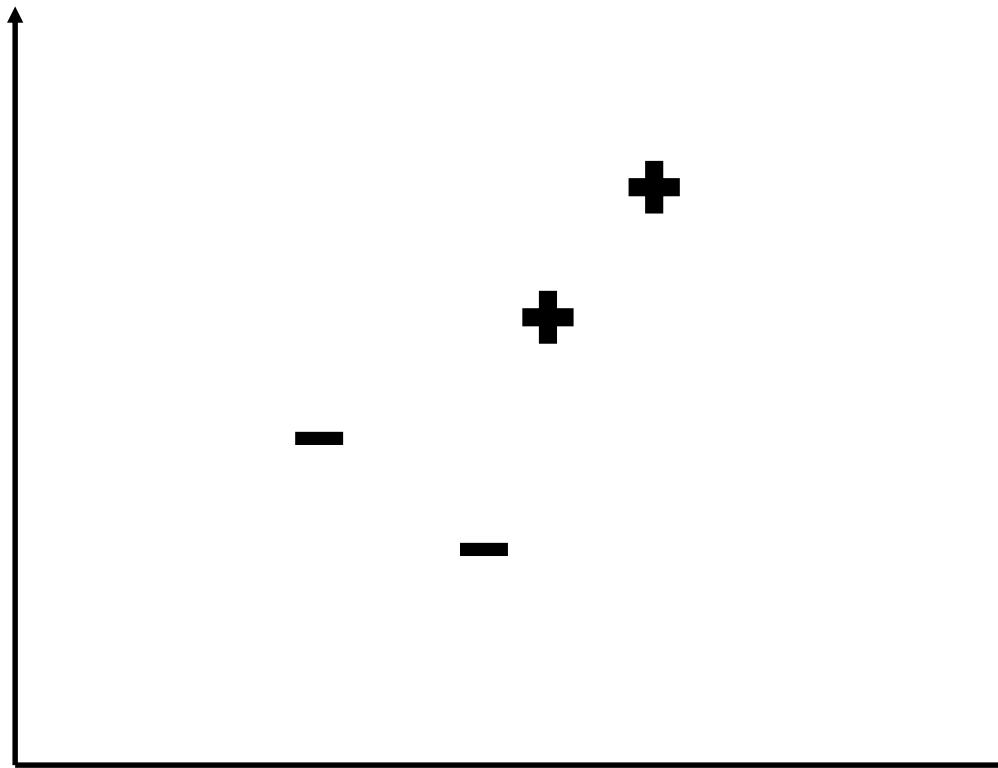
Fall Semester

Until now...

- Linearly separable dataset:
 - Linear classifier: $y(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$
 - Classification: $\begin{cases} t_n = +1 & \text{if } y(\mathbf{x}_n) \geq 0 \\ t_n = -1 & \text{if } y(\mathbf{x}_n) < 0 \end{cases}$
- We saw different algorithms
 - perceptron
 - least squares
 - logistic regression
- All trained using SGD which gives different decision boundaries with, e.g., different weight initialization!

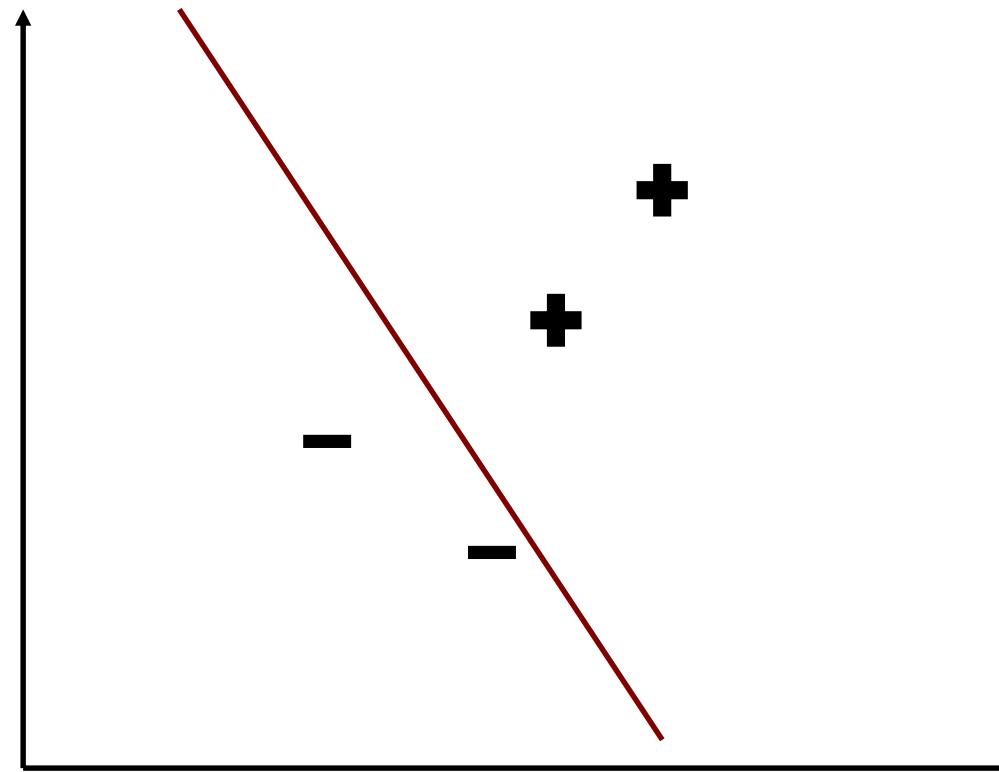
Linearly separable dataset

- Classification:
$$\begin{cases} t_n = +1 \text{ if } y(\mathbf{x}_n) \geq 0 \\ t_n = -1 \text{ if } y(\mathbf{x}_n) < 0 \end{cases}$$



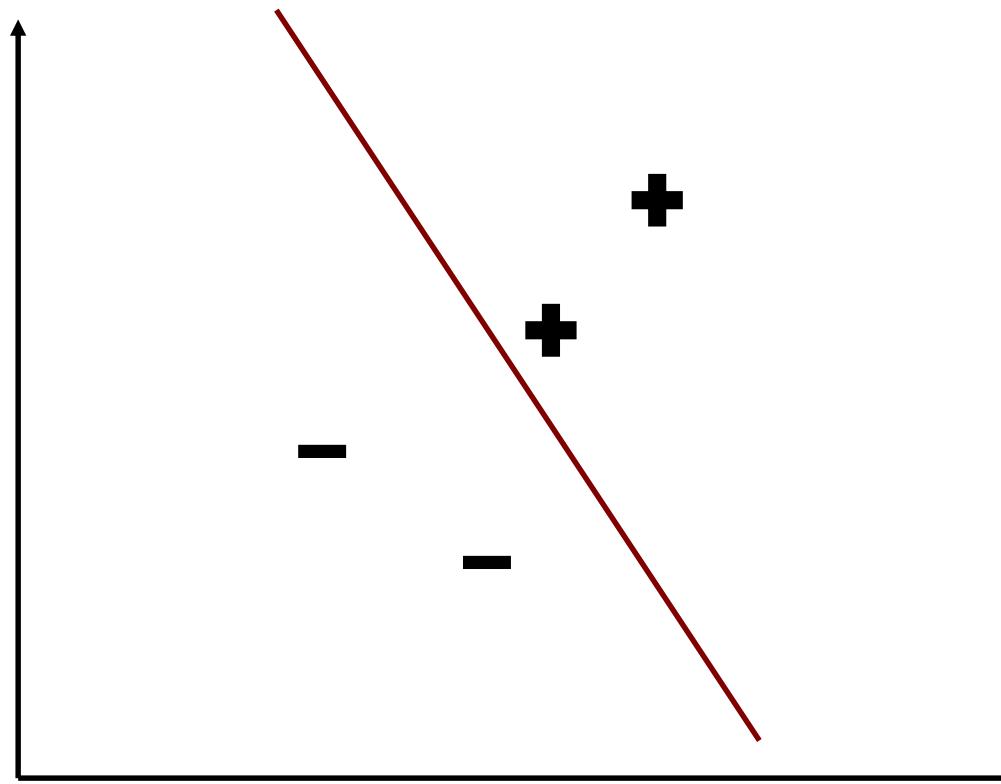
Linearly separable dataset

- Classification:
$$\begin{cases} t_n = +1 \text{ if } y(\mathbf{x}_n) \geq 0 \\ t_n = -1 \text{ if } y(\mathbf{x}_n) < 0 \end{cases}$$



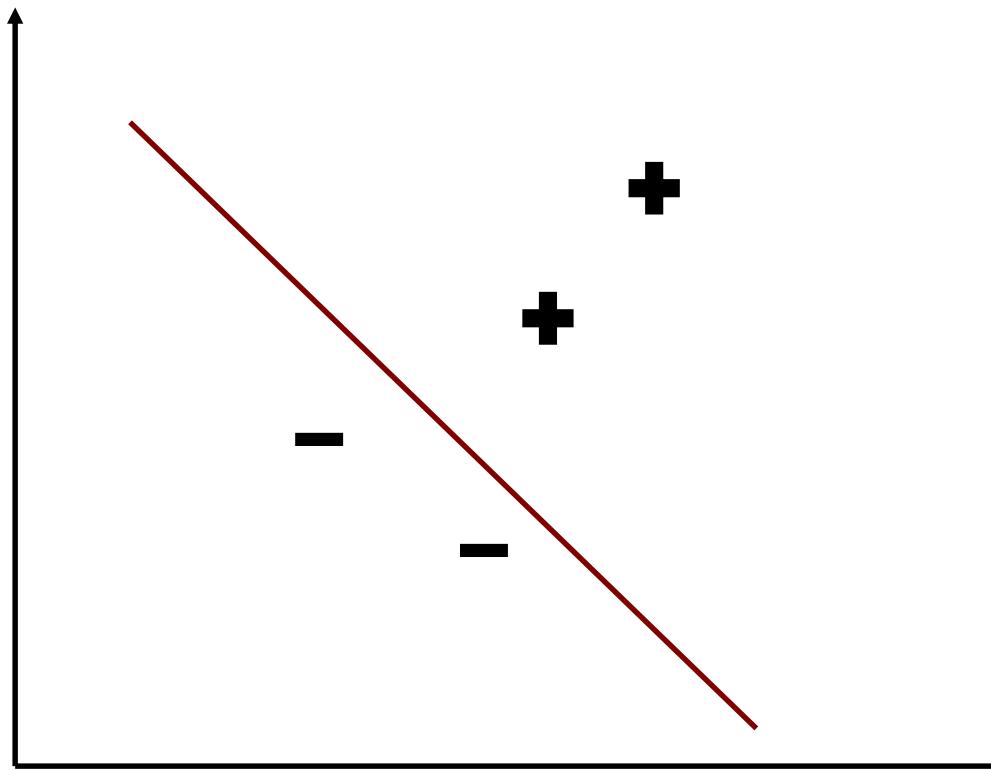
Linearly separable dataset

- Classification:
$$\begin{cases} t_n = +1 \text{ if } y(\mathbf{x}_n) \geq 0 \\ t_n = -1 \text{ if } y(\mathbf{x}_n) < 0 \end{cases}$$



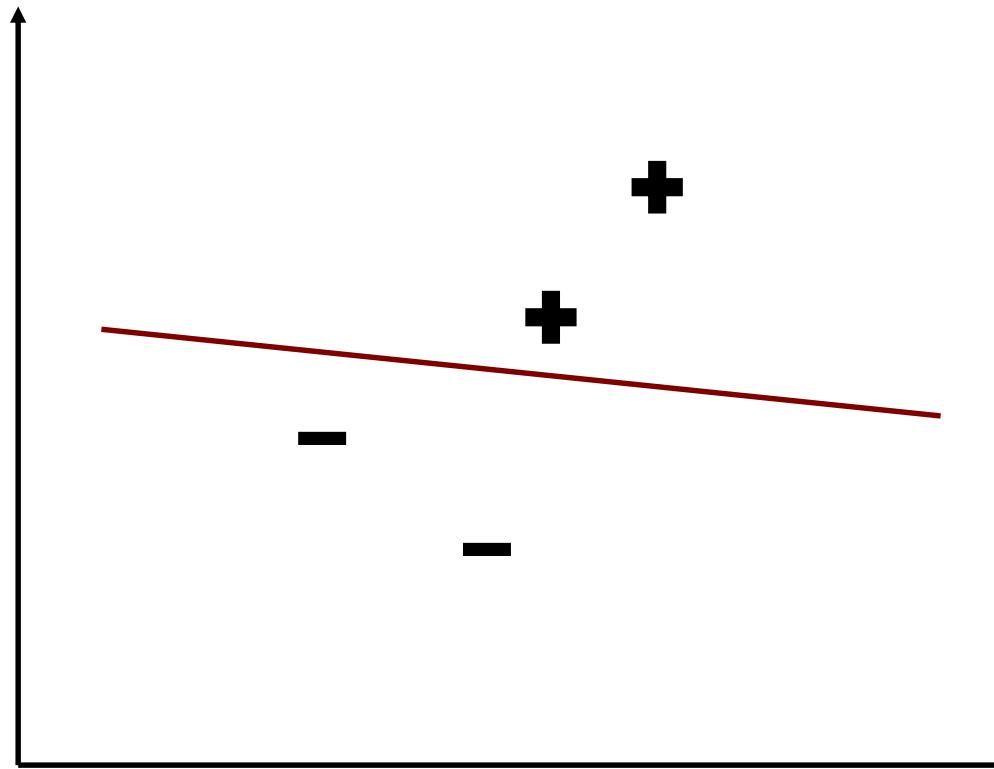
Linearly separable dataset

- Classification:
$$\begin{cases} t_n = +1 \text{ if } y(\mathbf{x}_n) \geq 0 \\ t_n = -1 \text{ if } y(\mathbf{x}_n) < 0 \end{cases}$$

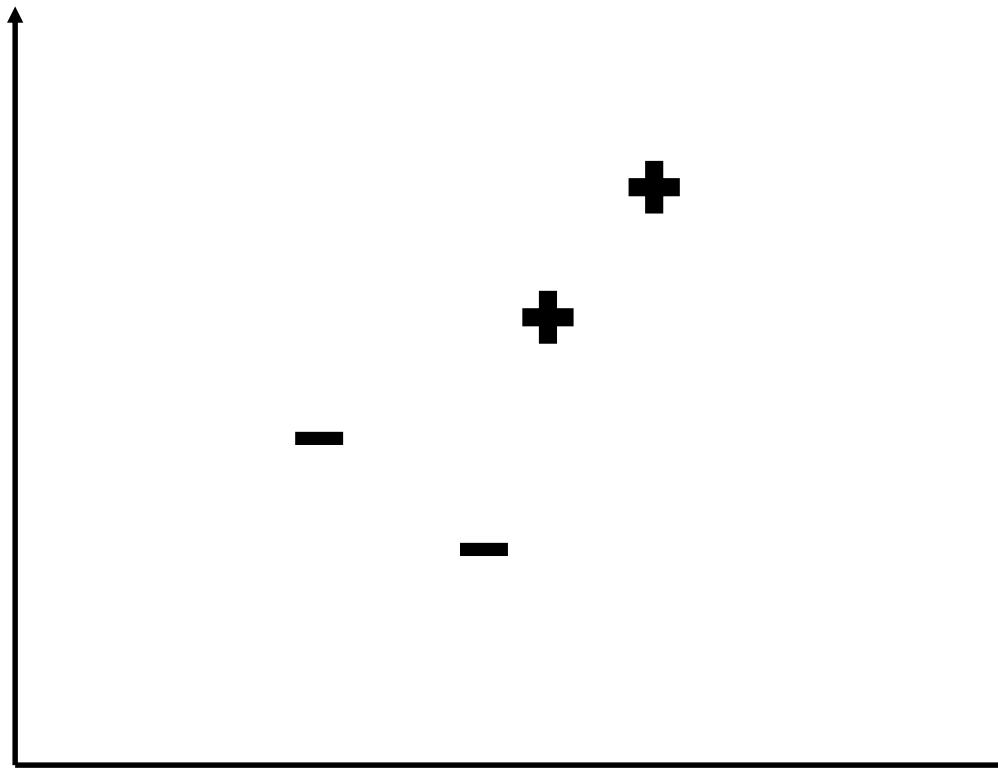


Linearly separable dataset

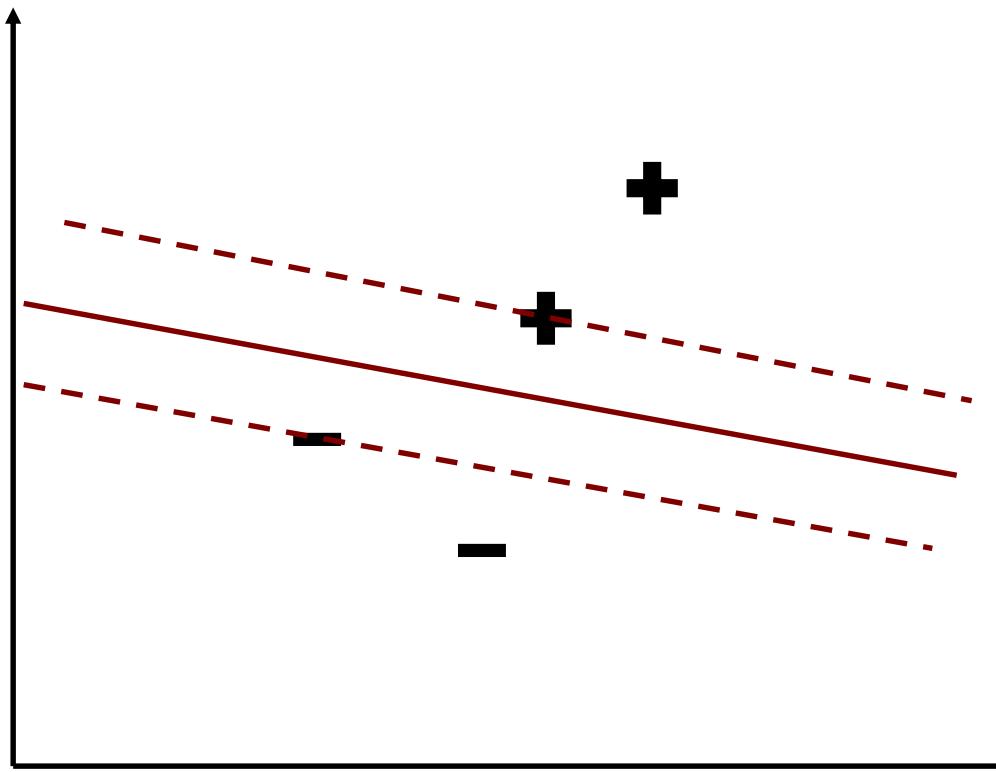
- Classification:
$$\begin{cases} t_n = +1 \text{ if } y(\mathbf{x}_n) \geq 0 \\ t_n = -1 \text{ if } y(\mathbf{x}_n) < 0 \end{cases}$$



How should we choose the boundary?

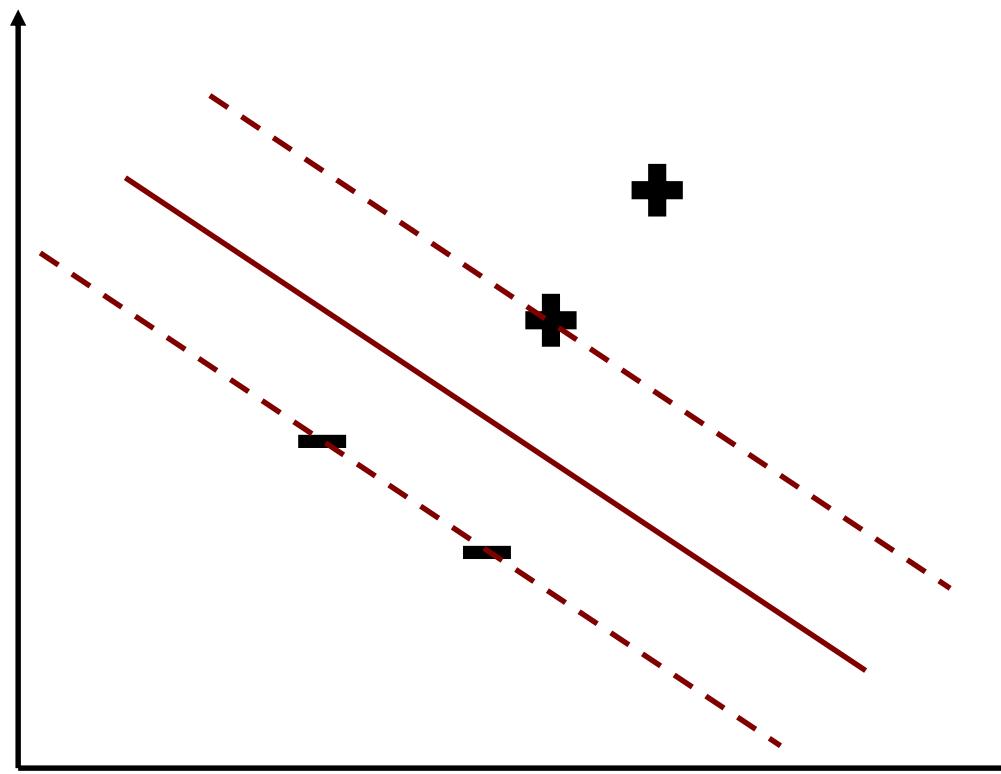


How should we choose the boundary?



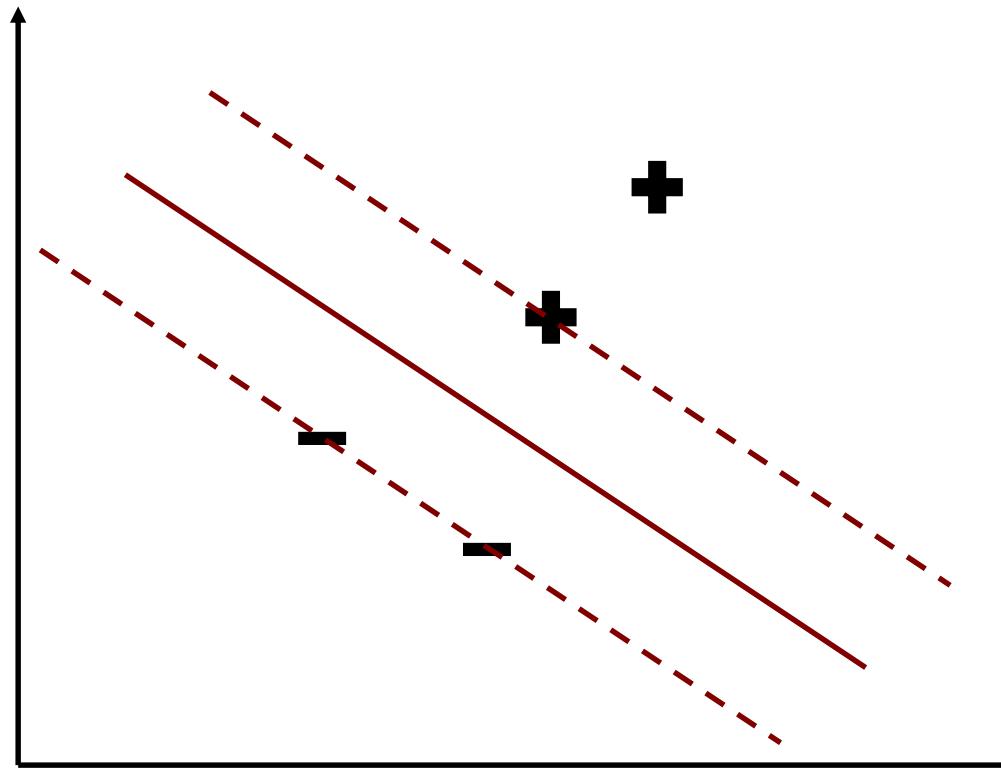
How should we choose the boundary?

- Maximum Margin Classifier



How should we choose the boundary?

- Maximum Margin Classifier
 - Today we will talk about: Support Vector Machines!
- Maximum Margin: most stable under perturbations of the input



Linear Classifier (recap)

$$y(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

- If \mathbf{x}' lies on decision boundary:

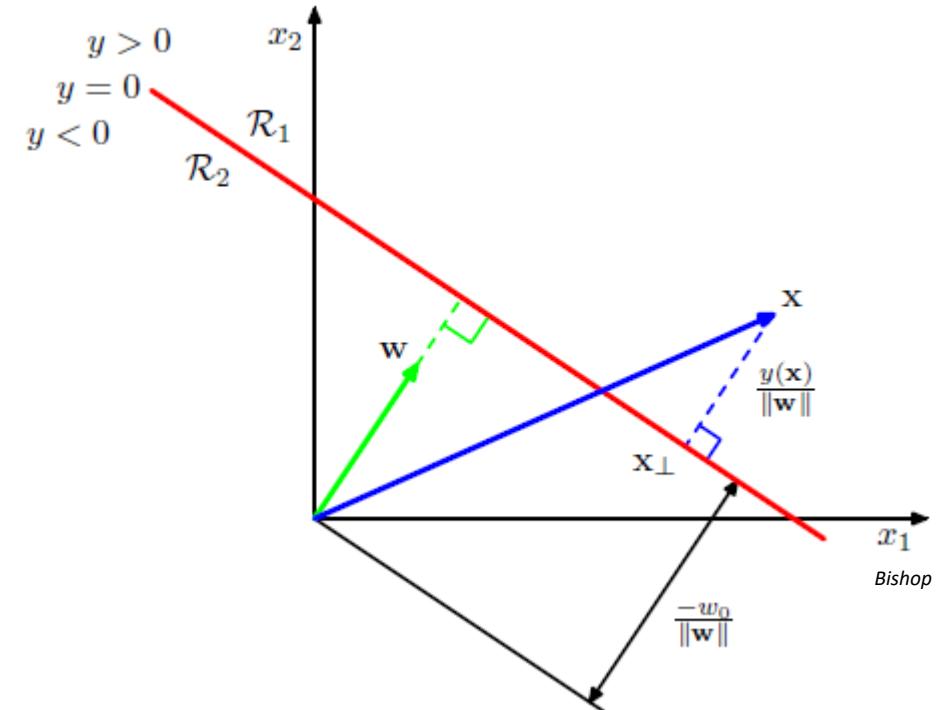
$$y(\mathbf{x}') = \mathbf{w}^t \mathbf{x}' + w_0 = 0$$

- Distance from \mathbf{x} to decision boundary is

$$r = \frac{|y(\mathbf{x})|}{\|\mathbf{w}\|} = \frac{t_n y(\mathbf{x})}{\|\mathbf{w}\|}$$

- For all $n = 1, \dots, N$ we have $t_n y(\mathbf{x}) \geq 0$

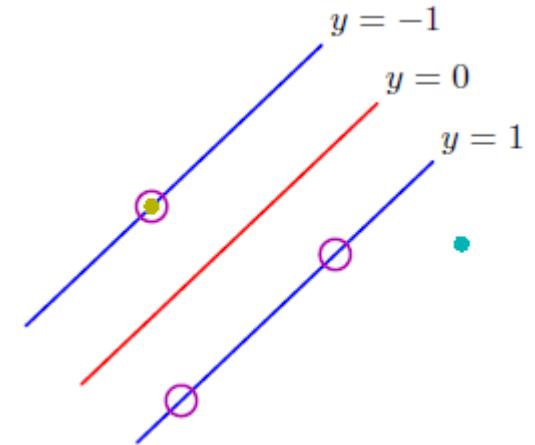
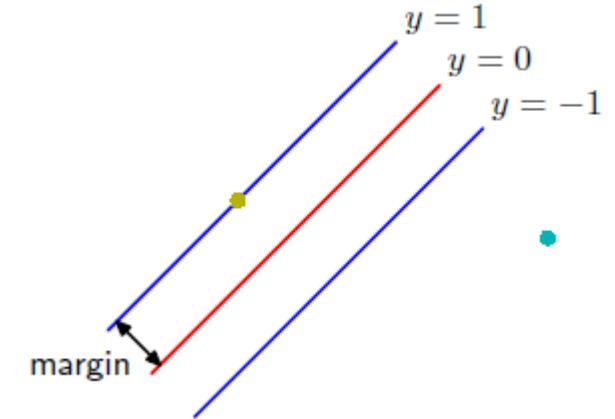
- Classification : $\begin{cases} \omega_1 & \text{if } y(\mathbf{x}_n) \geq 0 \\ \omega_2 & \text{if } y(\mathbf{x}_n) < 0 \end{cases}$



Maximum Margin Classifier (MMC)

- **Definition:** Margin is the perpendicular distance from decision boundary to the closest x_n .
- Distance of a point to the decision boundary is

$$r = \frac{|y(\mathbf{x})|}{\|\mathbf{w}\|} = \frac{\mathbf{t}_n y(\mathbf{x})}{\|\mathbf{w}\|} = \frac{\mathbf{t}_n (\mathbf{w}^T \mathbf{x}_n + w_0)}{\|\mathbf{w}\|}$$



Bishop

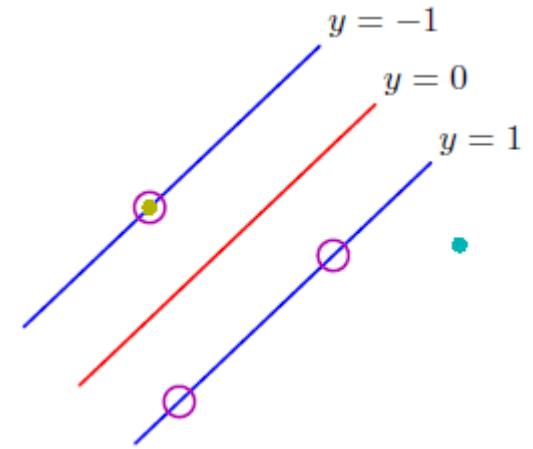
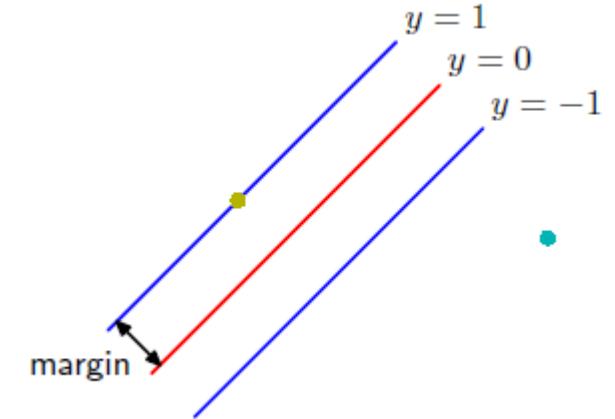
Maximum Margin Classifier (MMC)

- Definition: Margin is the perpendicular distance from decision boundary to the closest x_n .
- Distance of a point to the decision boundary is

$$r = \frac{|y(x)|}{\|w\|} = \frac{t_n y(x)}{\|w\|} = \frac{t_n (\mathbf{w}^T \mathbf{x}_n + w_0)}{\|w\|}$$

- Thus, margin is:

$$\min_n \frac{t_n (\mathbf{w}^T \mathbf{x}_n + w_0)}{\|w\|} = \min_n \frac{t_n (\kappa \mathbf{w}^T \mathbf{x}_n + \kappa w_0)}{\|\kappa w\|}$$



Bishop

Maximum Margin Classifier (MMC)

- Definition: Margin is the perpendicular distance from decision boundary to the closest x_n .
- Distance of a point to the decision boundary is

$$r = \frac{|y(x)|}{\|w\|} = \frac{t_n y(x)}{\|w\|} = \frac{t_n (\mathbf{w}^T \mathbf{x}_n + w_0)}{\|w\|}$$

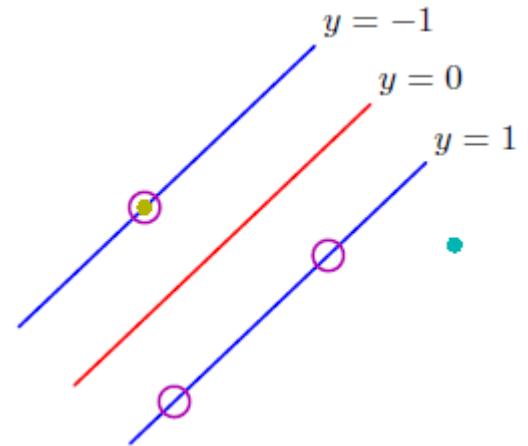
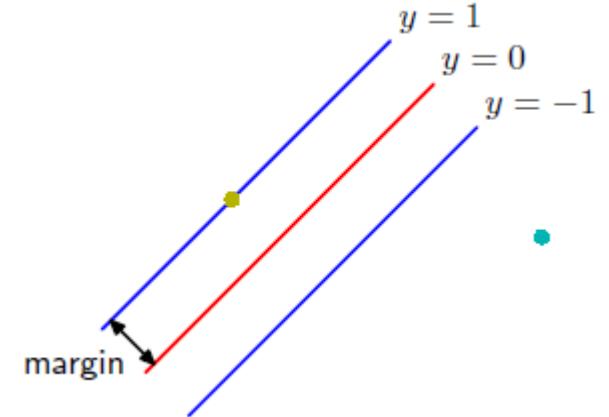
- Thus, margin is:

$$\min_n \frac{t_n (\mathbf{w}^T \mathbf{x}_n + w_0)}{\|w\|} = \min_n \frac{t_n (\kappa \mathbf{w}^T \mathbf{x}_n + \kappa w_0)}{\|\kappa w\|}$$

- Thus we choose κ such that:

$$t_n (\mathbf{w}^T \mathbf{x}_n + w_0) = 1$$

for the point, closest to the decision boundary (points in cycle)



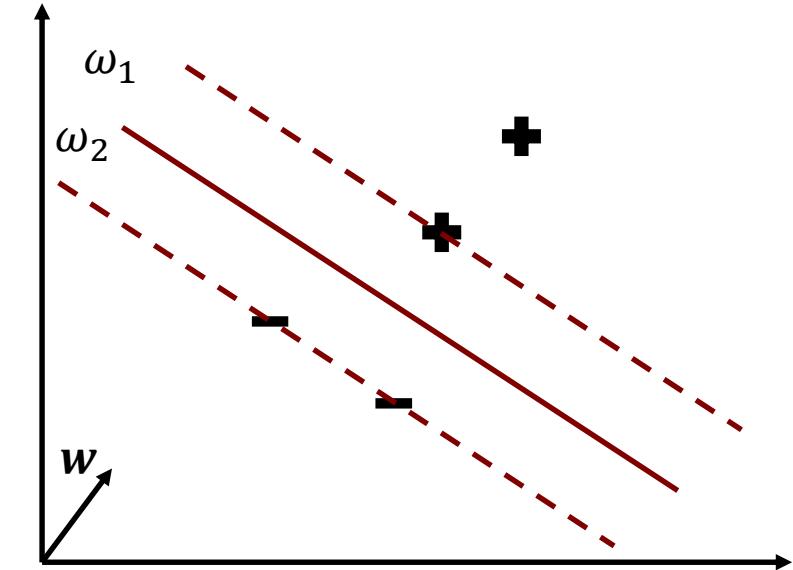
Bishop

Maximum Margin Classifier (MMC)

$$\begin{aligned}(\mathbf{w}^T \mathbf{x}_+ + w_0) &\geq 1 \\(\mathbf{w}^T \mathbf{x}_- + w_0) &< -1\end{aligned}$$

- For all the rest points:

$$t_n(\mathbf{w}^T \mathbf{x}_n + w_0) \geq 1$$



Maximum Margin Classifier (MMC)

$$\begin{aligned}(\mathbf{w}^T \mathbf{x}_+ + w_0) &\geq 1 \\(\mathbf{w}^T \mathbf{x}_- + w_0) &< -1\end{aligned}$$

- For all the rest points:

$$t_n(\mathbf{w}^T \mathbf{x}_n + w_0) \geq 1$$

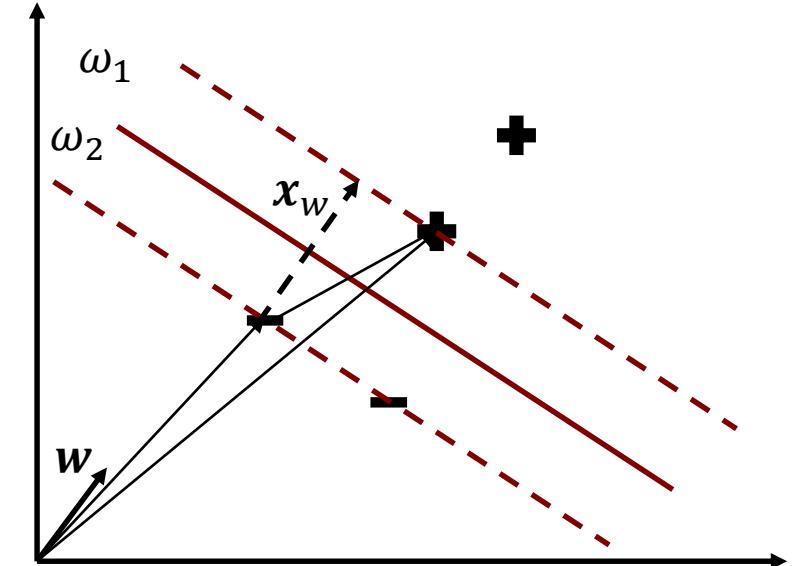
- Size of the margin:

$$\mathbf{x}_w = (\mathbf{x}_+ - \mathbf{x}_-) \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\mathbf{x}_w = (\mathbf{x}_+^T \mathbf{w} - \mathbf{x}_-^T \mathbf{w}) \frac{1}{\|\mathbf{w}\|}$$

$$\mathbf{x}_w = ((1 - w_0) + (1 + w_0)) \frac{1}{\|\mathbf{w}\|}$$

$$\mathbf{x}_w = \frac{2}{\|\mathbf{w}\|}$$



Maximum Margin Classifier (MMC)

$$\begin{aligned}(\mathbf{w}^T \mathbf{x}_+ + w_0) &\geq 1 \\(\mathbf{w}^T \mathbf{x}_- + w_0) &< -1\end{aligned}$$

- For all the rest points:

$$t_n(\mathbf{w}^T \mathbf{x}_n + w_0) \geq 1$$

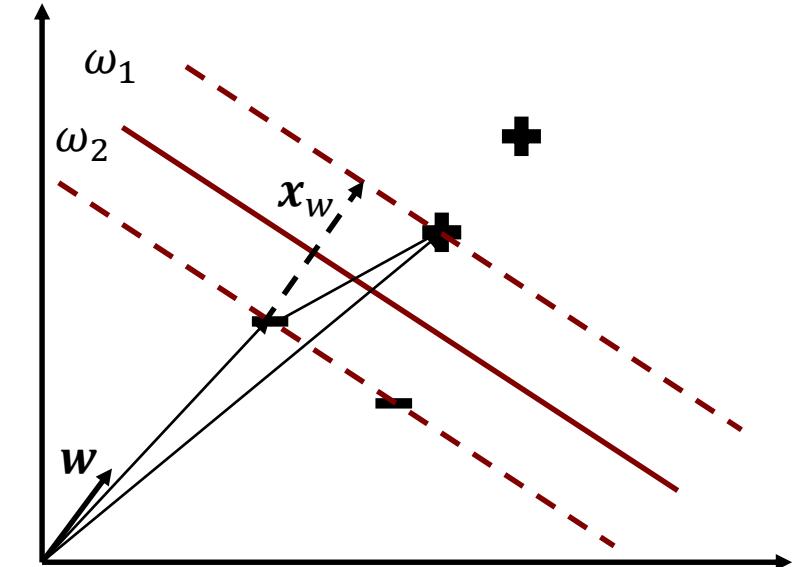
- Size of the margin:

$$\mathbf{x}_w = (\mathbf{x}_+ - \mathbf{x}_-) \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\mathbf{x}_w = (\mathbf{x}_+^T \mathbf{w} - \mathbf{x}_-^T \mathbf{w}) \frac{1}{\|\mathbf{w}\|}$$

$$\mathbf{x}_w = ((1 - w_0) + (1 + w_0)) \frac{1}{\|\mathbf{w}\|}$$

$$\mathbf{x}_w = \frac{2}{\|\mathbf{w}\|}, \text{ thus the size of the margin is } \frac{1}{\|\mathbf{w}\|}.$$



Maximum Margin Classifier (MMC)

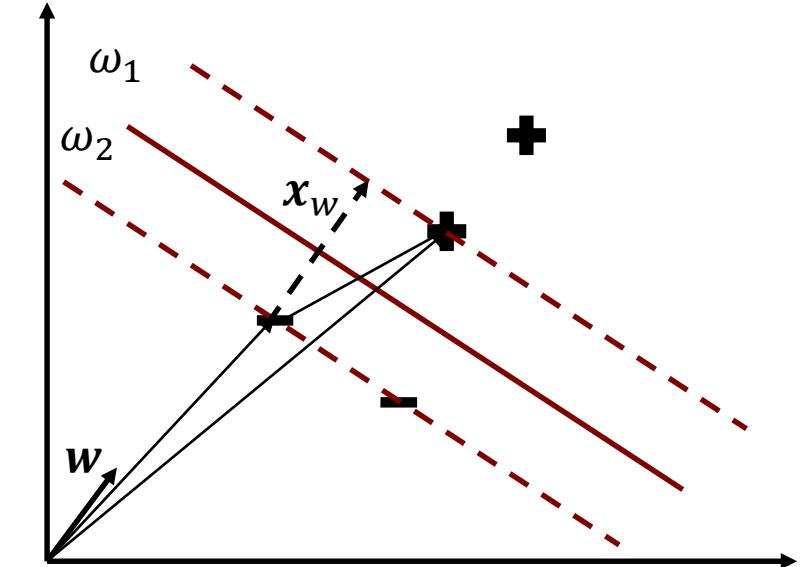
$$\begin{aligned}(\mathbf{w}^T \mathbf{x}_+ + w_0) &\geq 1 \\(\mathbf{w}^T \mathbf{x}_- + w_0) &< -1\end{aligned}$$

- Decision rule:

$$\omega_1 \text{ if } (\mathbf{w}^T \mathbf{x} + w_0) \geq 1$$

- During training with N samples:

$t_n, i = 1, \dots, N$ targets $\in \{-1, 1\}$



Then,

$$t_n(\mathbf{w}^T \mathbf{x}_+ + b) \geq 1$$

$$\rightarrow t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$$

$$t_n(\mathbf{w}^T \mathbf{x}_- + b) \geq 1$$

$$\begin{aligned}(\mathbf{w}^T \mathbf{x}_+ + w_0) &\geq 1 \\(\mathbf{w}^T \mathbf{x}_- + w_0) &< -1\end{aligned}$$

Maximum Margin Classifier (MMC)

- Thus, we want to maximize the margin

$$\frac{1}{\|\mathbf{w}\|}$$

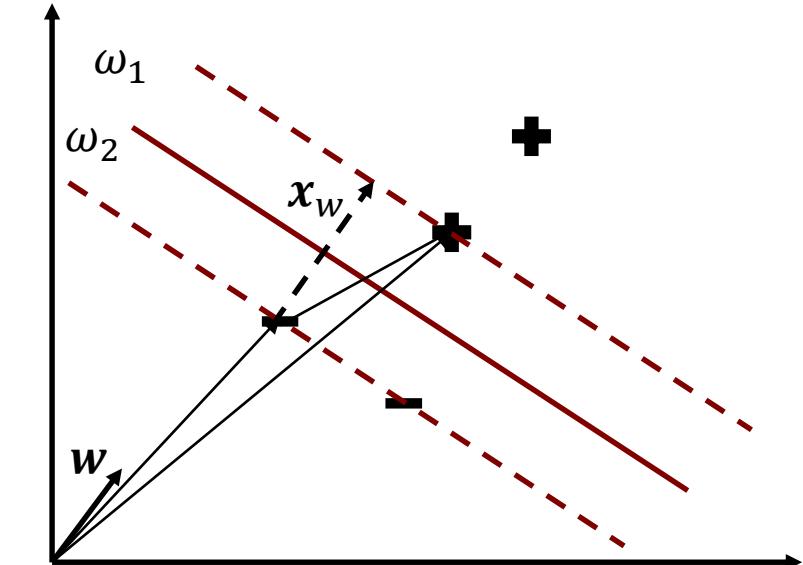
subject to N constraints

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) - 1 \geq 0$$

- Equivalent problem:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$$

- How should we solve this optimization problem?



$$\begin{aligned}(\mathbf{w}^T \mathbf{x}_+ + w_0) &\geq 1 \\(\mathbf{w}^T \mathbf{x}_- + w_0) &< -1\end{aligned}$$

Maximum Margin Classifier (MMC)

- Thus, we want to maximize the margin

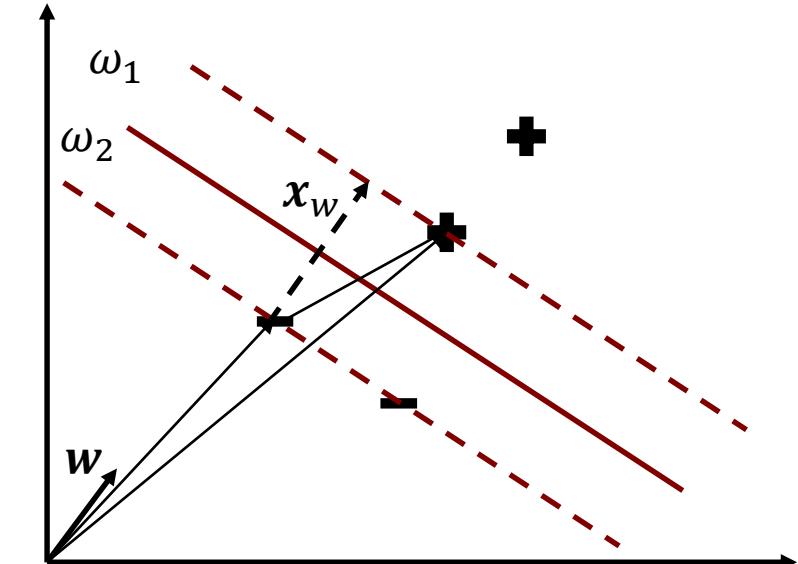
$$\frac{1}{\|\mathbf{w}\|}$$

subject to N constraints

$$t_n(\mathbf{w}^T \mathbf{x}_n + b) - 1 \geq 0$$

- Equivalent problem:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$$

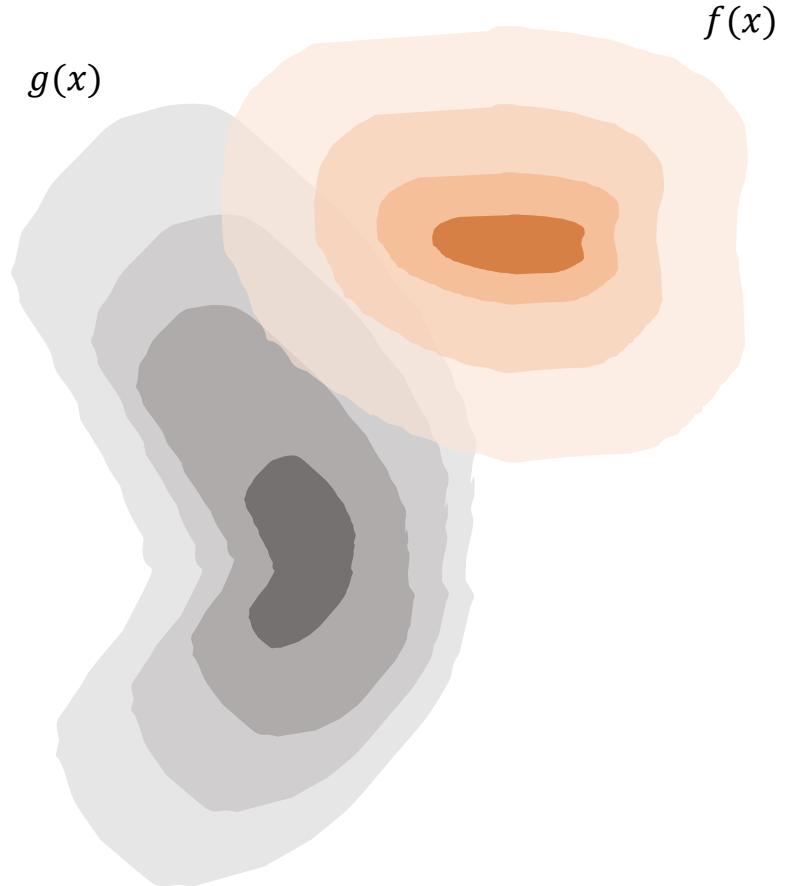


- How should we solve this optimization problem?
 - Lagrange Multipliers!

Lagrange Multipliers

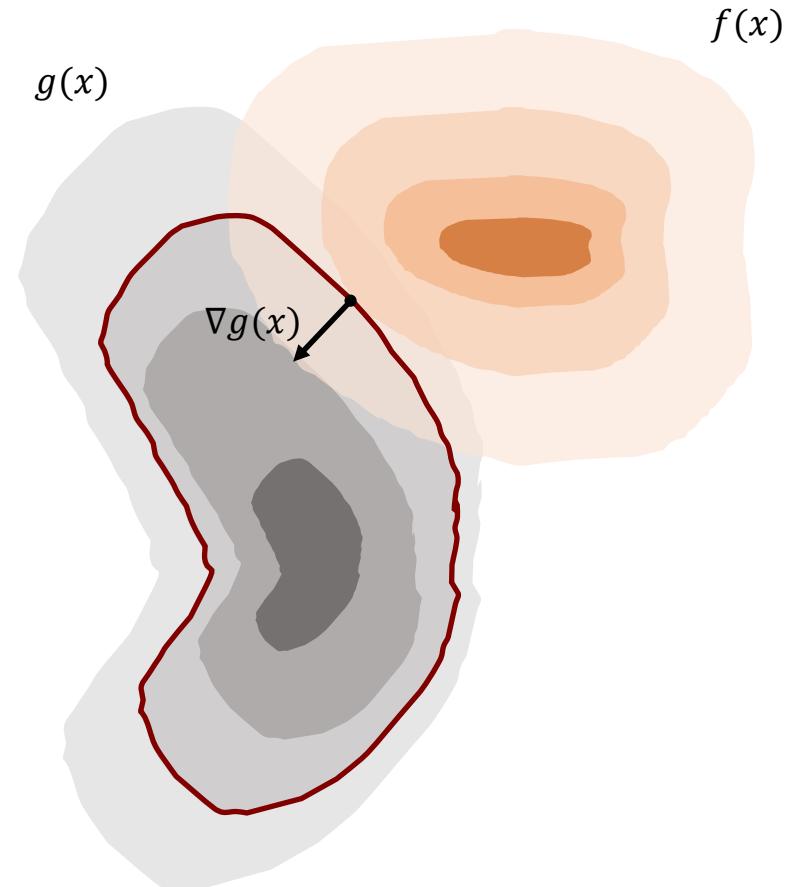
Optimization with equality constraints

- In general: Maximize $f(x)$ subject to $g(x) = 0$.



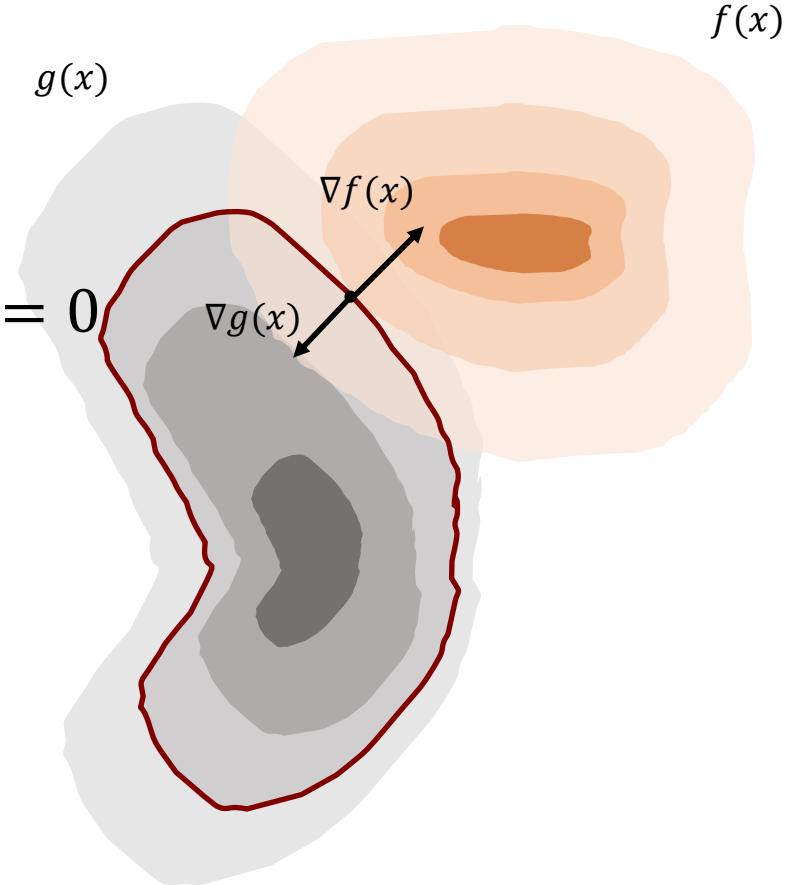
Optimization with equality constraints

- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface



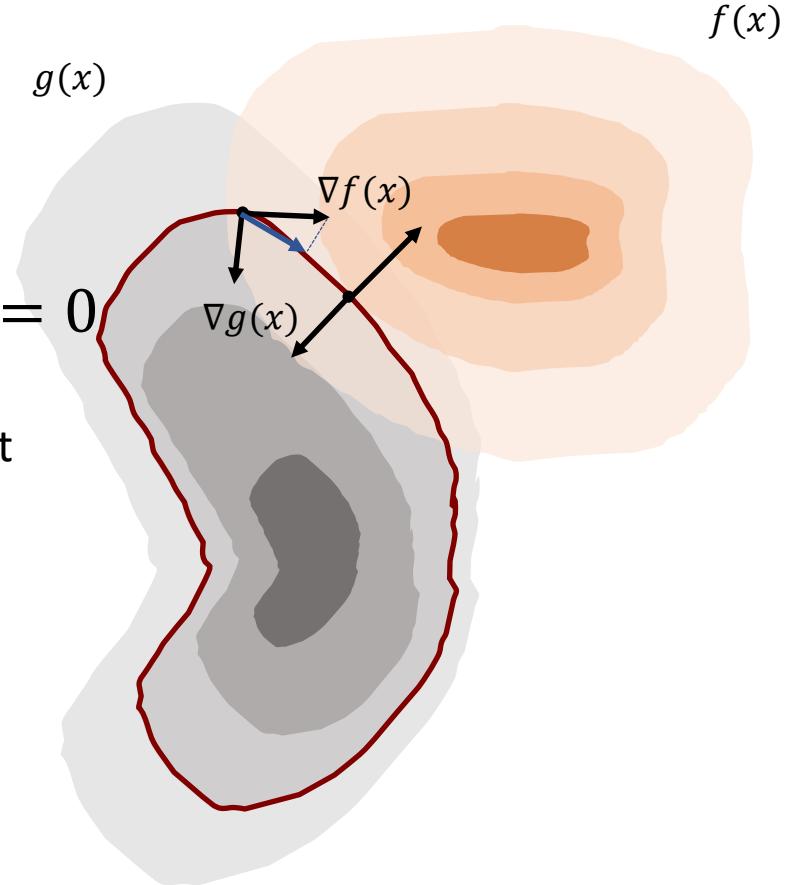
Optimization with equality constraints

- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface
- $\nabla f(x)$ is also perpendicular to the constraint $g(x) = 0$
 - Why?



Optimization with equality constraints

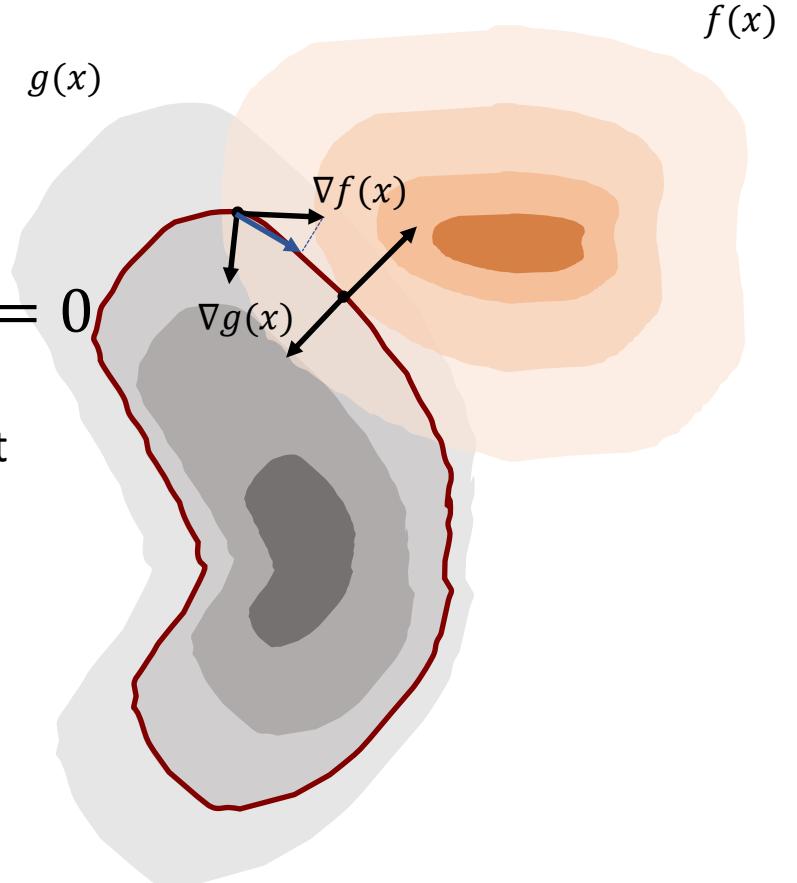
- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface
- $\nabla f(x)$ is also perpendicular to the constraint $g(x) = 0$
 - Why?
 - In every other point on that surface ($g(x) = 0$) $f(x)$ is not in its maximum.



Optimization with equality constraints

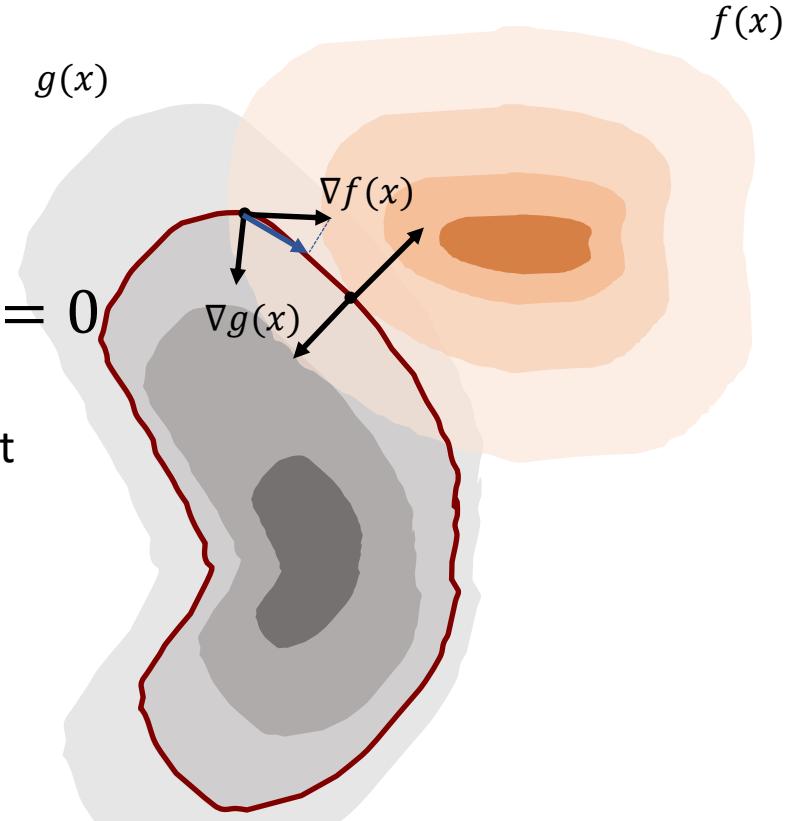
- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface
- $\nabla f(x)$ is also perpendicular to the constraint $g(x) = 0$
 - Why?
 - In every other point on that surface ($g(x) = 0$) $f(x)$ is not in its maximum.
- Therefore we want:

$$\nabla f(x) + \lambda \nabla g(x) = 0$$



Optimization with equality constraints

- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface
- $\nabla f(x)$ is also perpendicular to the constraint $g(x) = 0$
 - Why?
 - In every other point on that surface ($g(x) = 0$) $f(x)$ is not in its maximum.
- Therefore we want:
$$\nabla f(x) + \lambda \nabla g(x) = 0$$
- We introduce the Lagrangian function: $L(x, \lambda) = f(x) + \lambda g(x)$



Optimization with equality constraints

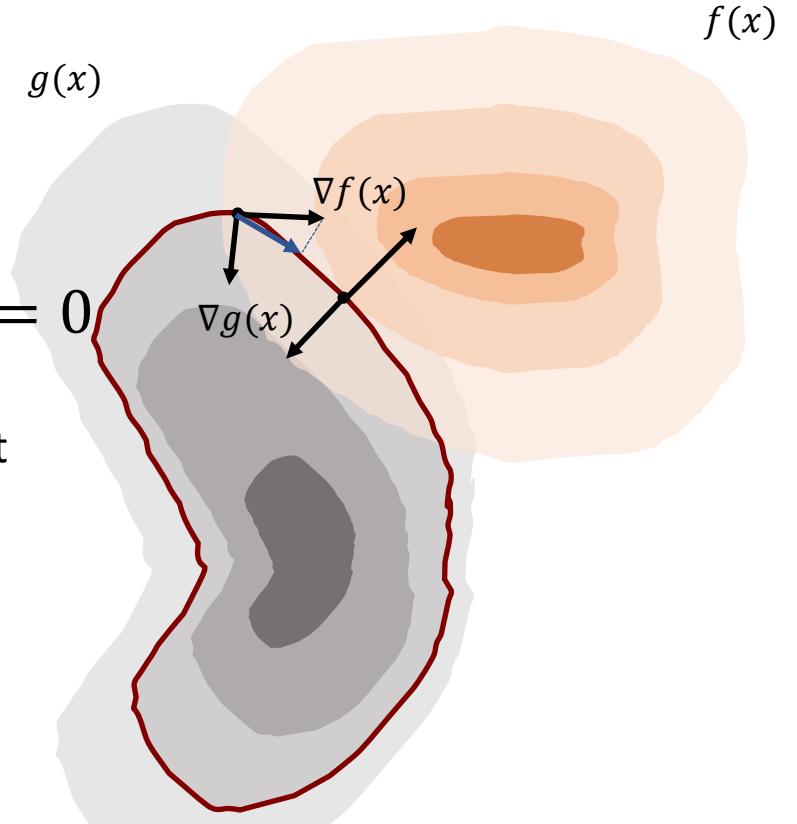
- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface
- $\nabla f(x)$ is also perpendicular to the constraint $g(x) = 0$

- Why?

- In every other point on that surface ($g(x) = 0$) $f(x)$ is not in its maximum.

- Therefore we want:

$$\nabla f(x) + \lambda \nabla g(x) = 0$$



- We introduce the Lagrangian function: $L(x, \lambda) = f(x) + \lambda g(x)$

- Solution: $\frac{\partial L(x, \lambda)}{\partial x} = 0 \rightarrow \nabla f(x) + \lambda \nabla g(x) = 0,$

Optimization with equality constraints

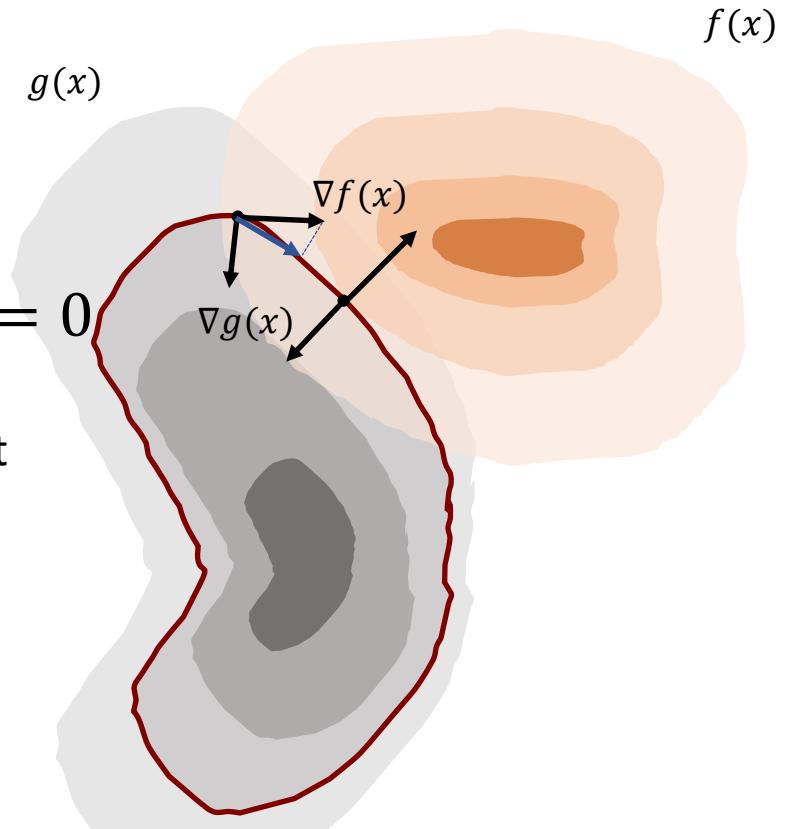
- In general: Maximize $f(x)$ subject to $g(x) = 0$.
- $\nabla g(x)$ is perpendicular to the constraint surface
- $\nabla f(x)$ is also perpendicular to the constraint $g(x) = 0$

• Why?

- In every other point on that surface ($g(x) = 0$) $f(x)$ is not in its maximum.

- Therefore we want:

$$\nabla f(x) + \lambda \nabla g(x) = 0$$



- We introduce the Lagrangian function: $L(x, \lambda) = f(x) + \lambda g(x)$

- Solution: $\frac{\partial L(x, \lambda)}{\partial x} = 0 \rightarrow \nabla f(x) + \lambda \nabla g(x) = 0, \frac{\partial L(x, \lambda)}{\partial \lambda} = 0 \rightarrow g(x) = 0$

Example

- Assume $f(x_1, x_2) = 1 - x_1^2 - x_2^2$ and $g(x_1, x_2) = x_1 + x_2 - 1$
- We need to maximize $f(x_1, x_2)$ s.t. $g(x_1, x_2) = 0$
- $L(x_1, x_2, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$
- Thus,

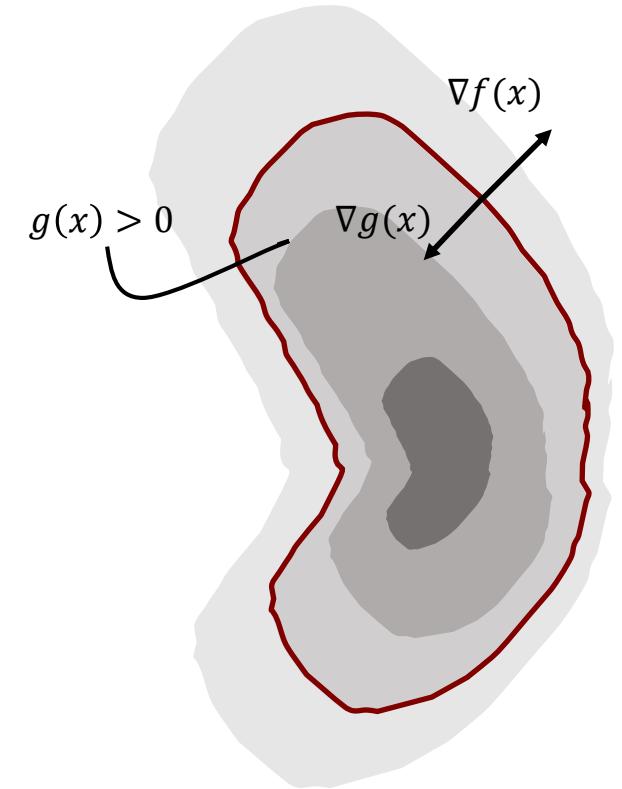
$$\frac{\partial L(x_1, x_2, \lambda)}{\partial x_1} = 0, \frac{\partial L(x_1, x_2, \lambda)}{\partial x_2} = 0, \frac{\partial L(x_1, x_2, \lambda)}{\partial \lambda} = 0$$

- From the system above we get:

$$\lambda = 1, x_1 = x_2 = \frac{1}{2}$$

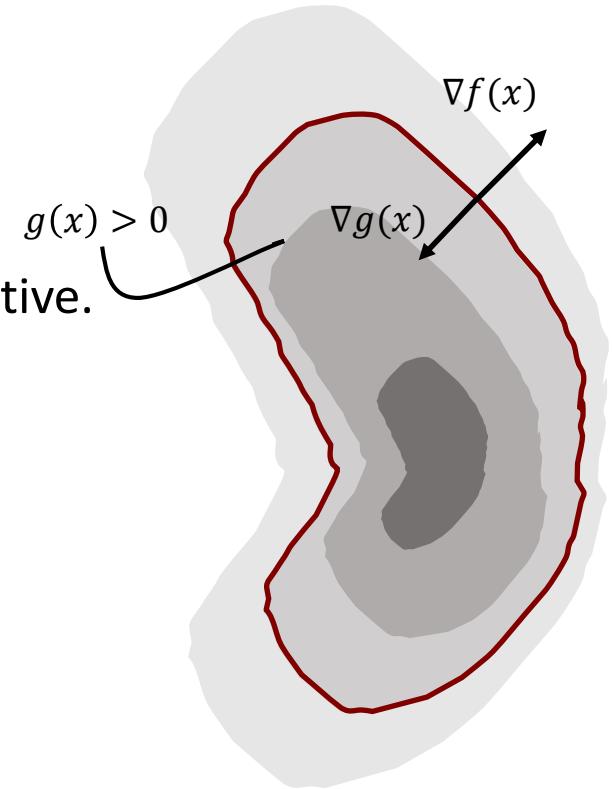
Optimization with inequality constraints

- We want to maximize $f(x)$ subject to $g(x) \geq 0$



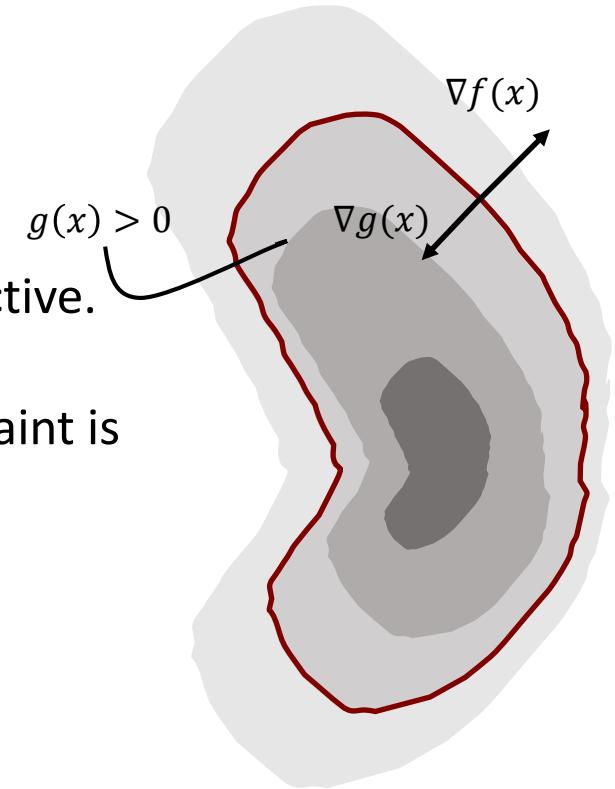
Optimization with inequality constraints

- We want to maximize $f(x)$ subject to $g(x) \geq 0$
- Two kinds of solutions:
 - Stationary point lies in the region $g(x) \geq 0$. Thus the constraint is inactive. Thus, the solution is $\nabla f(x) = 0$ ($\mu = 0$)



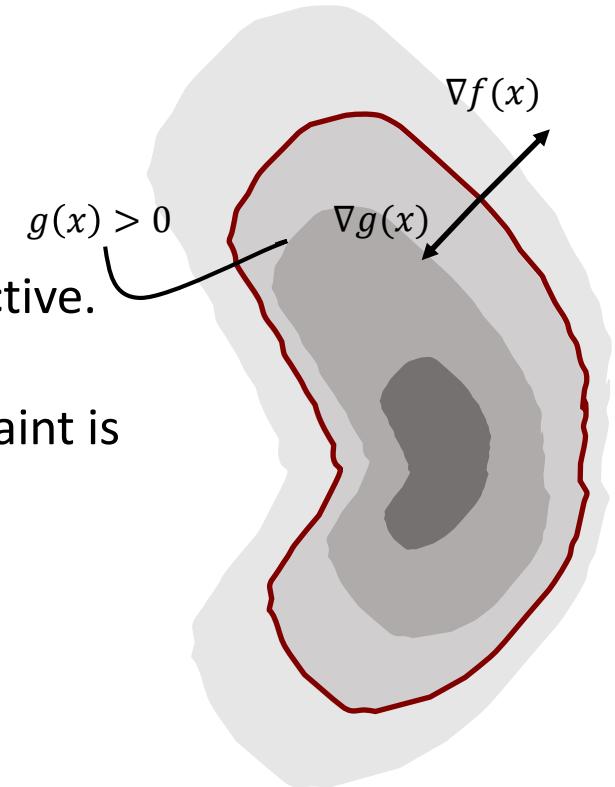
Optimization with inequality constraints

- We want to maximize $f(x)$ subject to $g(x) \geq 0$
- Two kinds of solutions:
 - Stationary point lies in the region $g(x) > 0$. Thus the constraint is inactive. Thus, the solution is $\nabla f(x) = 0$ ($\mu = 0$)
 - Stationary point lies on the boundary $g(x) = 0$. Thus, now the constraint is active. So, the solution is $\nabla f(x) = -\mu \nabla g(x)$ with $\mu \geq 0$.



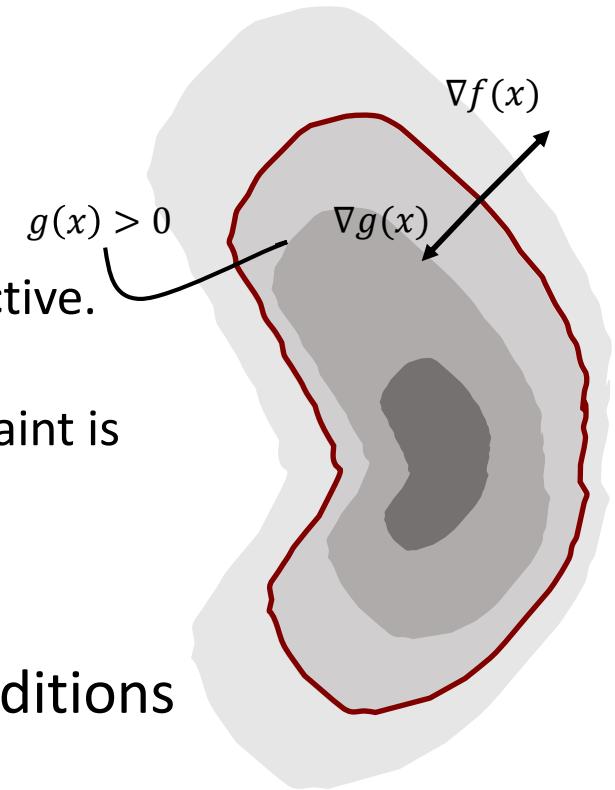
Optimization with inequality constraints

- We want to maximize $f(x)$ subject to $g(x) \geq 0$
- Two kinds of solutions:
 - Stationary point lies in the region $g(x) \geq 0$. Thus the constraint is inactive. Thus, the solution is $\nabla f(x) = 0$ ($\mu = 0$)
 - Stationary point lies on the boundary $g(x) = 0$. Thus, now the constraint is active. So, the solution is $\nabla f(x) = -\mu \nabla g(x)$ with $\mu \geq 0$.
 - in both cases we have: $\mu g(x) = 0$.



Optimization with inequality constraints

- We want to maximize $f(x)$ subject to $g(x) \geq 0$
- Two kinds of solutions:
 - Stationary point lies in the region $g(x) \geq 0$. Thus the constraint is inactive. Thus, the solution is $\nabla f(x) = 0$ ($\mu = 0$)
 - Stationary point lies on the boundary $g(x) = 0$. Thus, now the constraint is active. So, the solution is $\nabla f(x) = -\mu \nabla g(x)$ with $\mu \geq 0$.
 - in both cases we have: $\mu g(x) = 0$.
- Thus, we maximize $L(x, \lambda) = f(x) + \mu g(x)$ subject to the conditions Karush-Kuhn-Tucker conditions (KKT):
 - $g(x) \geq 0, \mu \geq 0, \mu g(x) = 0$
- If we need to minimize instead of maximizing then we minimize the Lagrangian function $L(x, \lambda) = f(x) - \mu g(x)$



Optimization with multiple constraints

- Suppose we want to maximize $f(x)$ subject to $g_j(x) = 0$ for $j = 1, \dots, J$ and $h_k(x) \geq 0$ for $k = 1, \dots, K$.
- We then introduce the Lagrange Multipliers $\{\lambda_j\}$ and $\{\mu_k\}$, and optimize the Lagrangian function given by:

$$L(x, \{\lambda_j\}, \{\mu_k\}) = f(x) + \sum_{j=1}^J \lambda_j g_j(x) + \sum_{k=1}^K \mu_k h_k(x)$$

$$\text{s.t. } \mu_k \geq 0 \text{ and } \mu_k h_k(x) = 0, k = 1, \dots, K$$

- **Note:** notice that only the μ Lagrange multipliers have to be positive!

Maximum Margin Classifier

Maximum Margin Classifier (MMC)

- We want to maximize the margin:

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad N \text{ constraints} \quad t_n(\mathbf{w}^t \mathbf{x}_n + b) \geq 1$$

- Lagrangian function:

$$L(\mathbf{w}, w_0, \mu) = \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{f(\mathbf{x})} - \sum_{n=1}^N \underbrace{\mu_n [t_n(\mathbf{w}^t \mathbf{x}_n + w_0) - 1]}_{g(\mathbf{x})}$$

- With KKT conditions for $n = 1, \dots, N$:

$$\mu_n \geq 0$$

$$t_n(\mathbf{w}^t \mathbf{x}_n + w_0) - 1 \geq 0$$

$$\mu_n [t_n(\mathbf{w}^t \mathbf{x}_n + b) - 1] = 0$$

Maximum Margin Classifier (MMC)

- Stationary points:

$$\frac{\partial L(\mathbf{w}, w_0, \mu)}{\partial \mathbf{w}} = 0, \frac{\partial L(\mathbf{w}, w_0, \mu)}{\partial w_0} = 0$$

- First condition:

$$\frac{\partial L(\mathbf{w}, w_0, \mu)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N \mu_n t_n \mathbf{x}_n = 0 \rightarrow \mathbf{w} = \sum_{n=1}^N \mu_n t_n \mathbf{x}_n$$

- Second condition:

$$\frac{\partial L(\mathbf{w}, w_0, \mu)}{\partial w_0} = - \sum_{n=1}^N \mu_n t_n = 0 \rightarrow \sum_{n=1}^N \mu_n t_n = 0$$

- We can now eliminate \mathbf{w} and w_0 from L
-

$$\mathbf{w} = \sum_{i=1}^N \mu_n t_n \mathbf{x}_n$$
$$\sum_{i=1}^N \mu_n t_n = 0$$

Maximum Margin Classifier (MMC)

- Elimination of \mathbf{w} and w_0 :

$$L(\mathbf{w}, w_0, \mu) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1]$$

$$\mathbf{w} = \sum_{i=1}^N \mu_n t_n \mathbf{x}_n$$
$$\sum_{i=1}^N \mu_n t_n = 0$$

Maximum Margin Classifier (MMC)

- Elimination of \mathbf{w} and w_0 :

$$\begin{aligned} L(\mathbf{w}, w_0, \mu) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\ &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \end{aligned}$$

$$\mathbf{w} = \sum_{i=1}^N \mu_n t_n \mathbf{x}_n$$

$$\sum_{i=1}^N \mu_n t_n = 0$$

Maximum Margin Classifier (MMC)

- Elimination of \mathbf{w} and w_0 :

$$\begin{aligned}
 L(\mathbf{w}, w_0, \mu) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \mathbf{w}^T \left(\frac{1}{2} \mathbf{w} - \sum_{n=1}^N \mu_n t_n \mathbf{x}_n \right) - \sum_{n=1}^N \mu_n t_n w_0 + \sum_{n=1}^N \mu_n
 \end{aligned}$$

→ $\mathbf{w} = \sum_{i=1}^N \mu_n t_n \mathbf{x}_n$

$\sum_{i=1}^N \mu_n t_n = 0$

Maximum Margin Classifier (MMC)

- Elimination of \mathbf{w} and w_0 :

$$\begin{aligned}
 L(\mathbf{w}, w_0, \mu) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \mathbf{w}^T \left(\frac{1}{2} \mathbf{w} - \boxed{\sum_{n=1}^N \mu_n t_n \mathbf{x}_n} \right) - \sum_{n=1}^N \mu_n t_n w_0 + \sum_{n=1}^N \mu_n
 \end{aligned}$$

→ $\mathbf{w} = \sum_{i=1}^N \mu_n t_n \mathbf{x}_n$

$\sum_{i=1}^N \mu_n t_n = 0$

Maximum Margin Classifier (MMC)

- Elimination of \mathbf{w} and w_0 :

$$\begin{aligned}
 L(\mathbf{w}, w_0, \mu) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \mathbf{w}^T \left(\frac{1}{2} \mathbf{w} - \boxed{\sum_{n=1}^N \mu_n t_n \mathbf{x}_n} \right) - \sum_{n=1}^N \mu_n t_n w_0 + \sum_{n=1}^N \mu_n \\
 &= -\frac{1}{2} \mathbf{w}^T \mathbf{w} - w_0 \sum_{n=1}^N \mu_n t_n + \sum_{n=1}^N \mu_n
 \end{aligned}$$

→ $\mathbf{w} = \sum_{i=1}^N \mu_n t_n \mathbf{x}_n$
→ $\sum_{i=1}^N \mu_n t_n = 0$

Maximum Margin Classifier (MMC)

- Elimination of \mathbf{w} and w_0 :

$$\begin{aligned}
 L(\mathbf{w}, w_0, \mu) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1] \\
 &= \mathbf{w}^T \left(\frac{1}{2} \mathbf{w} - \sum_{n=1}^N \mu_n t_n \mathbf{x}_n \right) - \sum_{n=1}^N \mu_n t_n w_0 + \sum_{n=1}^N \mu_n \\
 &= -\frac{1}{2} \mathbf{w}^T \mathbf{w} - w_0 \sum_{n=1}^N \mu_n t_n + \sum_{n=1}^N \mu_n
 \end{aligned}$$

Maximum Margin Classifier (MMC)

- We can now minimize w.r.t. μ the Lagrangian:

$$L(\mathbf{w}, w_0, \mu) = \sum_{n=1}^N \mu_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \mu_n \mu_m t_n t_m \mathbf{x}_n^T \mathbf{x}_m$$

with constraints:

$$\mu \geq 0$$

and

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

- Optimization problem: **Quadratic Programming**
- After solving this optimization problem we must get the decision boundary. How?

Maximum Margin Classifier (MMC)

- We get \mathbf{w} from

$$\mathbf{w} = \sum_{n=1}^N \mu_n t_n \mathbf{x}_n$$

Maximum Margin Classifier (MMC)

- We get \mathbf{w} from

$$\mathbf{w} = \sum_{n=1}^N \mu_n t_n \mathbf{x}_n$$

- Now recall that from KKT conditions I have:

$$\mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1] = 0 \text{ and } \mu_n \geq 0$$

- For which training samples are the Lagrange coefficients greater than 0?

Maximum Margin Classifier (MMC)

- We get \mathbf{w} from

$$\mathbf{w} = \sum_{n=1}^N \mu_n t_n \mathbf{x}_n$$

- Now recall that from KKT conditions I have:

$$\mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1] = 0 \text{ and } \mu_n \geq 0$$

- For which training samples are the Lagrange coefficients greater than 0?
- If the training samples is on the margin then $t_n (\mathbf{w}^t \mathbf{x}_n + w_0) = 1$, and thus from the above KKT condition we get (probably) non-zero μ_n .

Maximum Margin Classifier (MMC)

- We get \mathbf{w} from

$$\mathbf{w} = \sum_{n=1}^N \mu_n t_n \mathbf{x}_n$$

- Now recall that from KKT conditions I have:

$$\mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1] = 0 \text{ and } \mu_n \geq 0$$

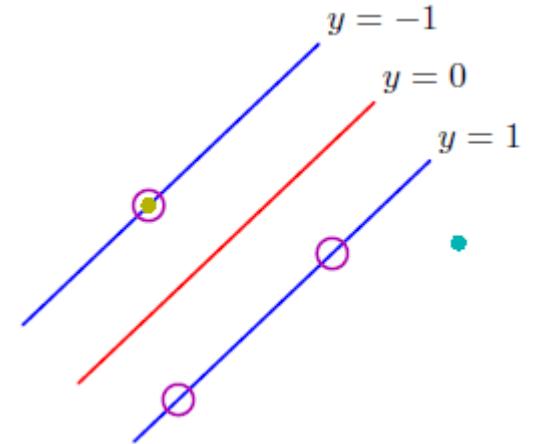
- For which training samples are the Lagrange coefficients greater than 0?
- If the training samples is on the margin then $t_n (\mathbf{w}^t \mathbf{x}_n + w_0) = 1$, and thus from the above KKT condition we get (probably) non-zero μ_n .
- For all the rest training samples $t_n (\mathbf{w}^t \mathbf{x}_n + w_0) \geq 1$ and $\mu_n = 0$.

Support Vectors

- Thus, we get \mathbf{w} from

$$\mathbf{w} = \sum_{n=1}^{N_s} \mu_n t_n \mathbf{x}_n$$

- Vector \mathbf{w} is linear combination of a set of vectors that lie on maximum margin hyperplanes that are called *support vectors!*



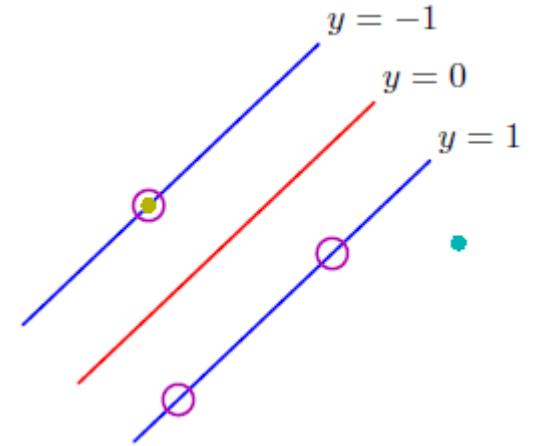
Bishop

Support Vectors

- Thus, we get \mathbf{w} from

$$\mathbf{w} = \sum_{n=1}^{N_s} \mu_n t_n \mathbf{x}_n$$

- Vector \mathbf{w} is linear combination of a set of vectors that lie on maximum margin hyperplanes that are called *support vectors*!



- For the estimation of the hyperplane we still need w_0 .
- We can use the fact that for support vectors we have:
 $t_n(\mathbf{w}^t \mathbf{x}_n + w_0) = 1$.

Support Vectors

- Thus, for a particular support vector \mathbf{x}_n we have:

$$t_n(\mathbf{w}^T \mathbf{x}_n + w_0) = 1 \Rightarrow \\ t_n \left(\sum_{m \in S} \mu_m t_m \mathbf{x}_m^T \mathbf{x}_n + w_0 \right) = 1$$

where S is the set of support vectors. By multiplying both parts with t_n :

$$\sum_{m \in S} \mu_m t_m \mathbf{x}_m^T \mathbf{x}_n + w_0 = t_n \Rightarrow w_0 = t_n - \sum_{m \in S} \mu_m t_m \mathbf{x}_m^T \mathbf{x}_n$$

- We get a more stable result if we average over all support vectors:

$$w_0 = \frac{1}{N_s} \sum_{n \in S} \left(t_n - \sum_{m \in S} \mu_m t_m \mathbf{x}_m^T \mathbf{x}_n \right)$$

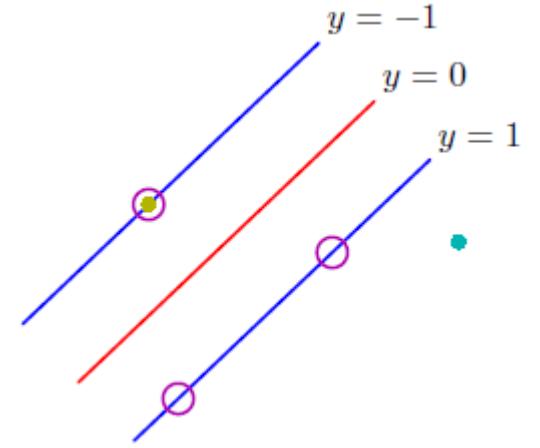
Decision Rule

- We predict the class of a data point x by:

$$y(x) = \mathbf{w}^T \mathbf{x} + w_0$$

or by replacing \mathbf{w} and w_0 :

$$y(x) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + \frac{1}{N_s} \sum_{n \in S} \left(t_n - \sum_{m \in S} \mu_m t_m \mathbf{x}_m^T \mathbf{x}_n \right)$$



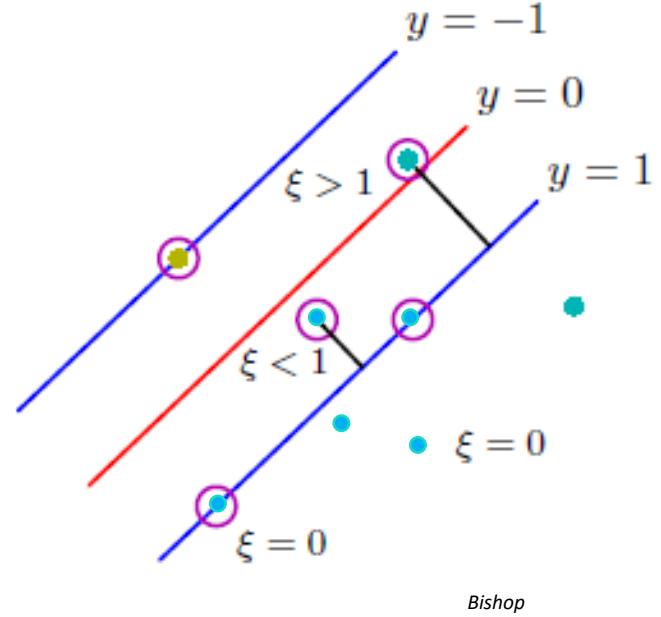
Bishop

Soft (Maximum) Margin Classifier

- So far we have assumed that all data points from the two classes are **perfectly separable** with a linear decision boundary.
 - Hard Maximum Margin Classifier
- Sometimes, though, class conditional distributions overlap with each other!
- Thus, we need to modify MMC to allow for some training points to be misclassified.

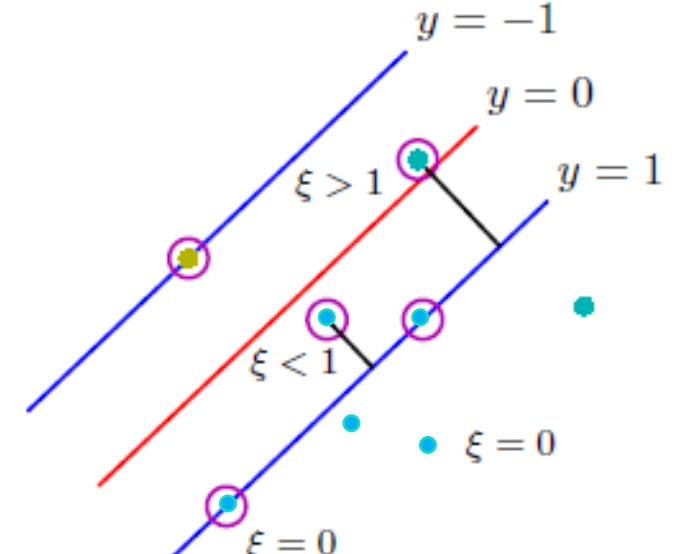
Soft (Maximum) Margin Classifier

- So far we have assumed that all data points from the two classes are **perfectly separable** with a linear decision boundary.
 - Hard Maximum Margin Classifier
- Sometimes, though, class conditional distributions overlap with each other!
- Thus, we need to modify MMC to allow for some training points to be misclassified.
- For each misclassified sample a **penalty** is attributed to it based on the distance to the margin boundary.



Soft (Maximum) Margin Classifier

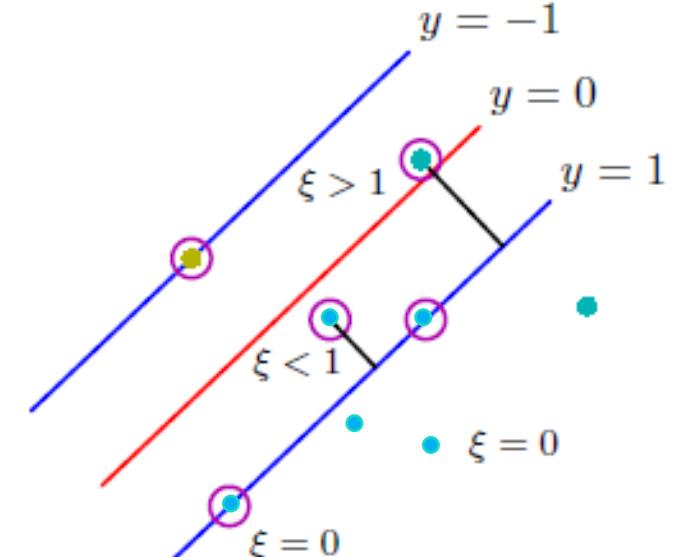
- Introduce slack variables $\xi_n \geq 0, n = 1, \dots, N$, i.e., one slack variable for each sample.



Bishop

Soft (Maximum) Margin Classifier

- Introduce slack variables $\xi_n \geq 0, n = 1, \dots, N$, i.e., one slack variable for each sample.
- If the sample is on the correct side of the margin: $\xi_n = 0$
- If the sample is on the wrong side of the margin
 $\xi_n = |t_n - y(\mathbf{x}_n)|$.
 - Recall: $y(\mathbf{x}_n)$ is proportional to the distance from the boundary.

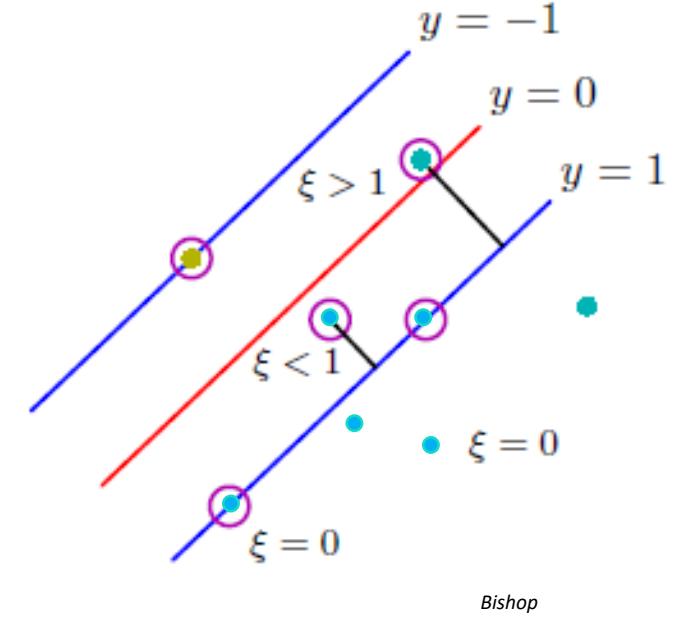


Bishop

Soft (Maximum) Margin Classifier

- Introduce slack variables $\xi_n \geq 0, n = 1, \dots, N$, i.e., one slack variable for each sample.
- If the sample is on the correct side of the margin: $\xi_n = 0$
- If the sample is on the wrong side of the margin
 $\xi_n = |t_n - y(\mathbf{x}_n)|$.
 - Recall: $y(\mathbf{x}_n)$ is proportional to the distance from the boundary.
- Previously we had hard margins:

$$t_n y(\mathbf{x}_n) \geq 1, n = 1, \dots, N$$

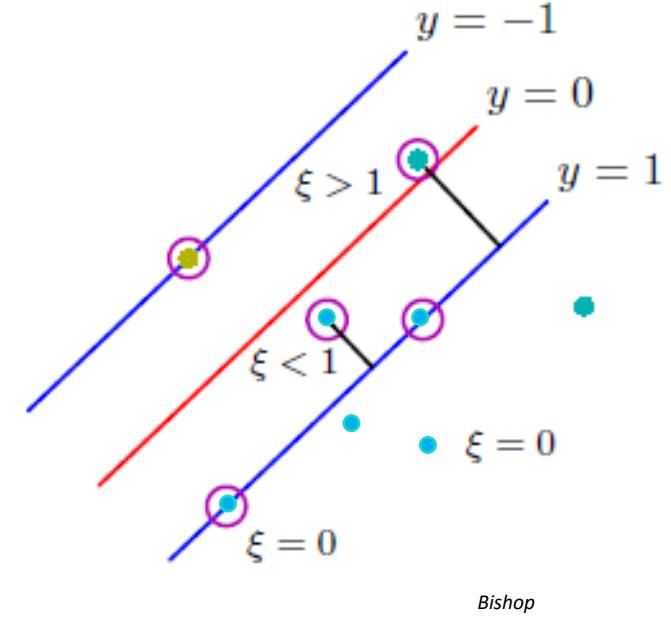


- Now we have soft margins:

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, n = 1, \dots, N$$

Soft (Maximum) Margin Classifier

- Thus we have three kinds of samples:
 - Samples that are the correct side of the margin (or on the margin):
$$\xi_n = 0$$
 - Samples that lie within the margin but on the correct side of the decision boundary:
$$0 < \xi_n \leq 1$$
 - Misclassified samples:
$$\xi_n > 1$$
- This is described as **relaxing the hard margin constraint** and allows for sample training data to be misclassified with some penalty (soft margins).



Bishop

Soft (Maximum) Margin Classifier

- We need to maximize margin but **give penalty** to points that lie on the wrong side of the margin (or decision boundary).
- Thus we need to minimize:

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

subject to constraints for $n = 1, \dots, N$:

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

$$\xi_n \geq 0$$

- Recall: how we define Lagrangian with **multiple** constraints!

Soft (Maximum) Margin Classifier

- Corresponding Lagrangian:

$$\begin{aligned} L(\mathbf{w}, w_0, \xi, \lambda, \mu) &= \\ &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N \lambda_n \xi_n - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1 + \xi_n] \end{aligned}$$

with KKT conditions for $n = 1, \dots, N$:

$$\begin{array}{ll} \mu_n \geq 0 & \lambda_n \geq 0 \\ t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1 + \xi_n \geq 0 & \xi_n \geq 0 \\ \mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1 + \xi_n] = 0 & \lambda_n \xi_n = 0 \end{array}$$

- How many constraints in total?

Soft (Maximum) Margin Classifier

- Corresponding Lagrangian:

$$\begin{aligned} L(\mathbf{w}, w_0, \xi, \lambda, \mu) &= \\ &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N \lambda_n \xi_n - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1 + \xi_n] \end{aligned}$$

with KKT conditions for $n = 1, \dots, N$:

$$\begin{array}{ll} \mu_n \geq 0 & \lambda_n \geq 0 \\ t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1 + \xi_n \geq 0 & \xi_n \geq 0 \\ \mu_n [t_n (\mathbf{w}^t \mathbf{x}_n + w_0) - 1 + \xi_n] = 0 & \lambda_n \xi_n = 0 \end{array}$$

- How many constraints in total? $6N$ constraints

Soft (Maximum) Margin Classifier

- Corresponding Lagrangian:

$$L = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N \lambda_n \xi_n - \sum_{n=1}^N \mu_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0) - 1 + \xi_n]$$

- Same procedure as before: minimize L w.r.t. variables \mathbf{w}, w_0, ξ_n and use the KKT conditions to eliminate those variables from L .
- In particular:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N \mu_n t_n \mathbf{x}_n = 0 \rightarrow \mathbf{w} = \sum_{n=1}^N \mu_n t_n \mathbf{x}_n$$

$$\frac{\partial L}{\partial w_0} = - \sum_{n=1}^N \mu_n t_n = 0 \rightarrow \sum_{n=1}^N \mu_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = C - \mu_n - \lambda_n \rightarrow \mu_n = C - \lambda_n$$

Soft (Maximum) Margin Classifier

- By eliminating \mathbf{w}, w_0, ξ_n from \mathcal{L} , I get:

$$\mathcal{L}(\boldsymbol{\mu}) = \sum_{n=1}^N \mu_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \mu_n \mu_m t_n t_m \mathbf{x}_n^T \mathbf{x}_m$$

with remaining constraints:

$$0 \leq \boldsymbol{\mu} \leq C, \quad \sum_{i=1}^N \mu_i t_i = 0$$

(box constraints)

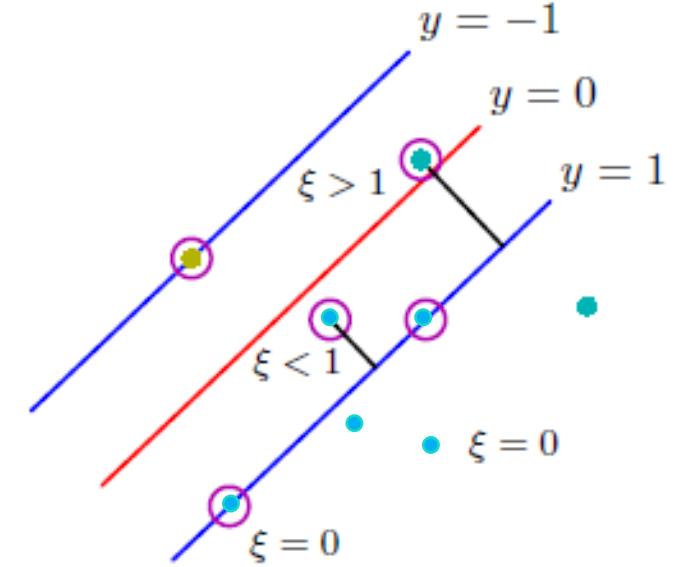
- As soon as we optimize \mathcal{L} w.r.t. $\boldsymbol{\mu}$ we predict the class of a new sample by:

$$y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0$$

Soft (Maximum) Margin Classifier

$$y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0$$

- Now, I have two types of support vectors. For support vectors $\mu_n > 0$, but also:
 - If $\mu_n < C$ then from $\mu_n = C - \lambda_n$ I get $\lambda_n > 0$ and from the condition $\lambda_n \xi_n = 0$ I get $\xi_n = 0$, thus they **lie on the margin** (they cannot be on the correct side of the margin as $\mu_n > 0$).

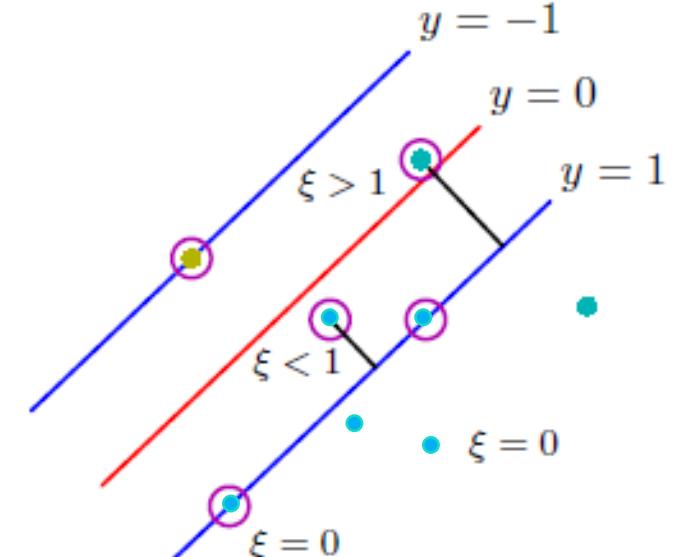


Bishop

Soft (Maximum) Margin Classifier

$$y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0$$

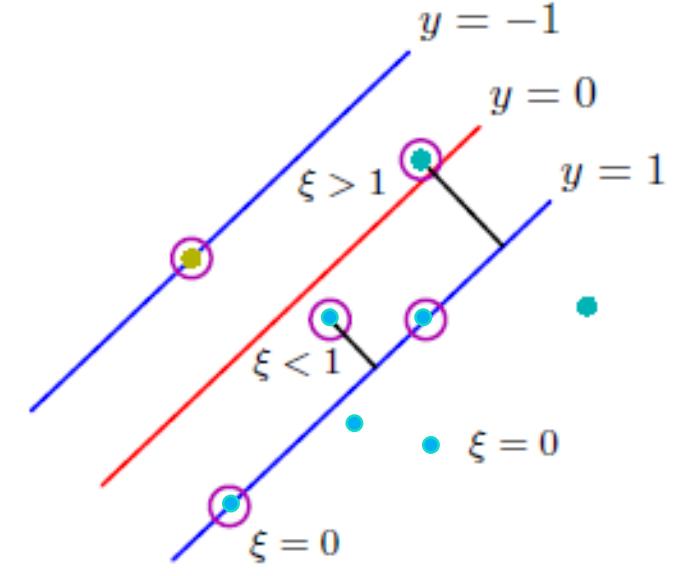
- Now, I have two types of support vectors. For support vectors $\mu_n > 0$, but also:
 - If $\mu_n < C$ then from $\mu_n = C - \lambda_n$ I get $\lambda_n > 0$ and from the condition $\lambda_n \xi_n = 0$ I get $\xi_n = 0$, thus they lie on the margin (they cannot be on the correct side of the margin as $\mu_n > 0$).
 - If $\mu_n = C$, then from $\mu_n = C - \lambda_n$ I get $\lambda_n = 0$ so from the condition $\lambda_n \xi_n = 0$ I get $\xi_n > 0$:
 - correctly classified, within margin, $\xi_n \leq 1$
 - misclassified $\xi_n > 1$



Soft (Maximum) Margin Classifier

$$y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0$$

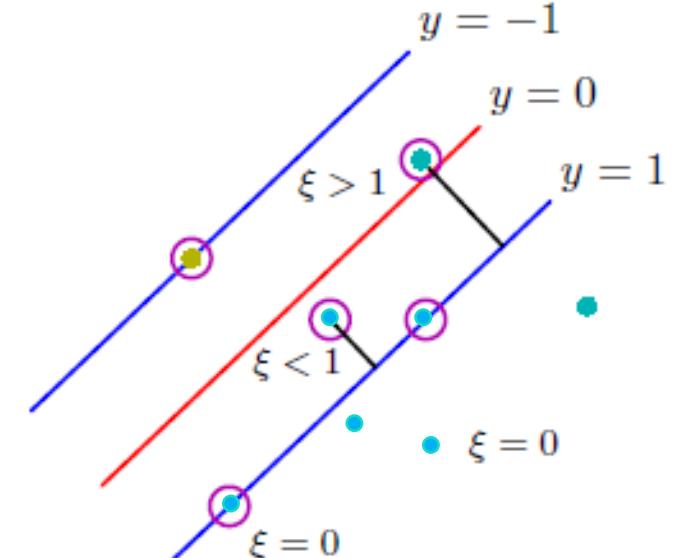
- Now, I have two types of support vectors. For support vectors $\mu_n > 0$, but also:
 - If $\mu_n < C$ then from $\mu_n = C - \lambda_n$ I get $\lambda_n > 0$ and from the condition $\lambda_n \xi_n = 0$ I get $\xi_n = 0$, thus they lie on the margin (they cannot be on the correct side of the margin as $\mu_n > 0$).
 - If $\mu_n = C$, then from $\mu_n = C - \lambda_n$ I get $\lambda_n = 0$ so from the condition $\lambda_n \xi_n = 0$ I get $\xi_n > 0$:
 - correctly classified, within margin, $\xi_n \leq 1$
 - misclassified $\xi_n > 1$
- What happens when $C \rightarrow \infty$



Soft (Maximum) Margin Classifier

$$y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0$$

- Now, I have two types of support vectors. For support vectors $\mu_n > 0$, but also:
 - If $\mu_n < C$ then from $\mu_n = C - \lambda_n$ I get $\lambda_n > 0$ and from the condition $\lambda_n \xi_n = 0$ I get $\xi_n = 0$, thus they lie on the margin (they cannot be on the correct side of the margin as $\mu_n > 0$).
 - If $\mu_n = C$, then from $\mu_n = C - \lambda_n$ I get $\lambda_n = 0$ so from the condition $\lambda_n \xi_n = 0$ I get $\xi_n > 0$:
 - correctly classified, within margin, $\xi_n \leq 1$
 - misclassified $\xi_n > 1$
- What happens when $C \rightarrow \infty$
 - Hard Margin Classifier
- What happens when $C \rightarrow 0$

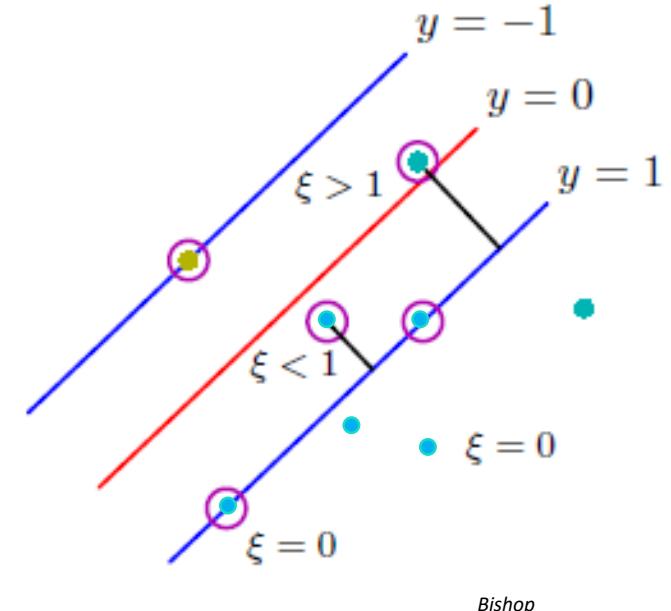


Bishop

Soft (Maximum) Margin Classifier

$$y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0$$

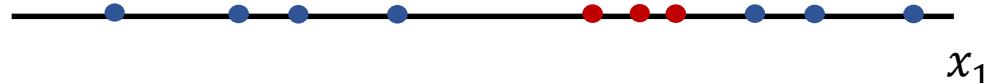
- Now, I have two types of support vectors. For support vectors $\mu_n > 0$, but also:
 - If $\mu_n < C$ then from $\mu_n = C - \lambda_n$ I get $\lambda_n > 0$ and from the condition $\lambda_n \xi_n = 0$ I get $\xi_n = 0$, thus they lie on the margin (they cannot be on the correct side of the margin as $\mu_n > 0$).
 - If $\mu_n = C$, then from $\mu_n = C - \lambda_n$ I get $\lambda_n = 0$ so from the condition $\lambda_n \xi_n = 0$ I get $\xi_n > 0$:
 - correctly classified, within margin, $\xi_n \leq 1$
 - misclassified $\xi_n > 1$
- What happens when $C \rightarrow \infty$
 - Hard Margin Classifier
- What happens when $C \rightarrow 0$
 - Infinite margin (every sample becomes a support vector)



Bishop

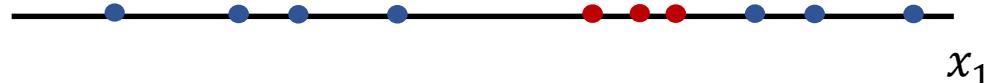
Basis functions

- We have already talked about classification with basis functions.
- Example: with a feature x_1 and corresponding values of two classes (red and blue):



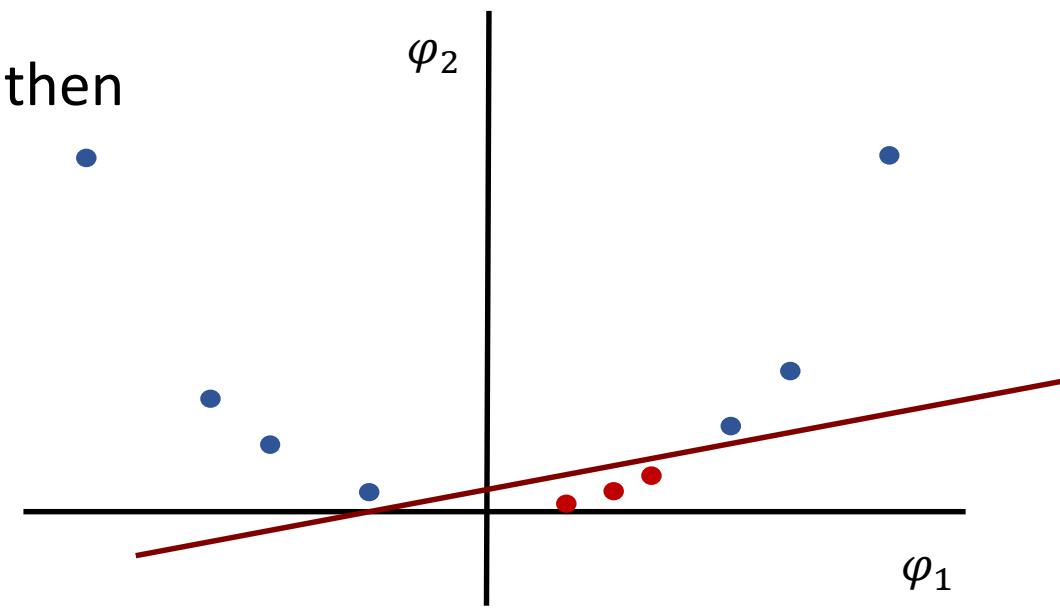
Basis functions

- We have already talked about classification with basis functions.
- Example: with a feature x_1 and corresponding values of two classes (red and blue):



- If I add one more feature: $x_2 = x_1^2$ then

$$\varphi(x_1) = \begin{bmatrix} x_1 \\ x_1^2 \end{bmatrix}$$



Basis functions

- Example: Now consider the mapping of the feature vector $\mathbf{x} \in \mathbb{R}^2 \rightarrow \varphi(\mathbf{x}) \in \mathbb{R}^3$:

$$\varphi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

It can be easily shown that:

$$\varphi_n(\mathbf{x})^T \varphi_m(\mathbf{x}) = (\mathbf{x}_n^T \mathbf{x}_m)^2 = K(\mathbf{x}_n, \mathbf{x}_m)$$

- **In words:** the inner product of the vectors in the **new** high dimensional space is expressed as a function of the inner product of the respective vectors in the low dimensional space!
 - It can be proved that for every symmetric, continuous function $K(\mathbf{x}_n, \mathbf{x}_m)$ there is a high dimensional space in which $K(\mathbf{x}_n, \mathbf{x}_m)$ defines an inner product. (under certain conditions – see Mercer Theorem)
-

Kernel functions

- Functions like $K(\mathbf{x}_n, \mathbf{x}_m)$ are called *kernel functions*.
- For every kernel function there exists a mapping $\varphi: \mathbb{R}^d \rightarrow \mathbb{R}^M$ such that:

$$K(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$$

- Depending on the kernel, M can be **infinite!**
 - In general it is difficult to retrieve the corresponding mapping $\varphi(\mathbf{x})$ for a given kernel.
 - There are certain types of kernels that we often use in Machine Learning.
-

Kernel functions-Examples

- Generalized polynomial kernel:

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^t \mathbf{x}' + c)^k, k > 0$$

- Radial Basis Functions:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}\right)$$

- Tanh kernel:

$$K(\mathbf{x}, \mathbf{x}') = \tanh(\beta \mathbf{x}^t \mathbf{x}' + \gamma)$$

- And many more. We can also construct new kernels from known ones...
-

Kernel functions-Examples

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}'$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)$$

Bishop

The Kernel Trick!

- Formulate your problem in such a way that the input vectors \mathbf{x}_n appear only in the form of scalar products:

$$\mathbf{x}_n^T \mathbf{x}_n$$

- Replace all instances with kernel function: $K(\mathbf{x}_n, \mathbf{x}_m)$.
 - $K(\mathbf{x}_n, \mathbf{x}_m)$ corresponds to a scalar product in some (possibly infinite dimensional) feature space φ !
 - **You do not know this space but you know what $K(\mathbf{x}_n, \mathbf{x}_m)$ represents!**
 - With this trick you **may** have projected the initial feature space to a high dimensional one where classes are linearly separable!
-

Kernel Trick and SVMs

- The decision rule for SVMs:

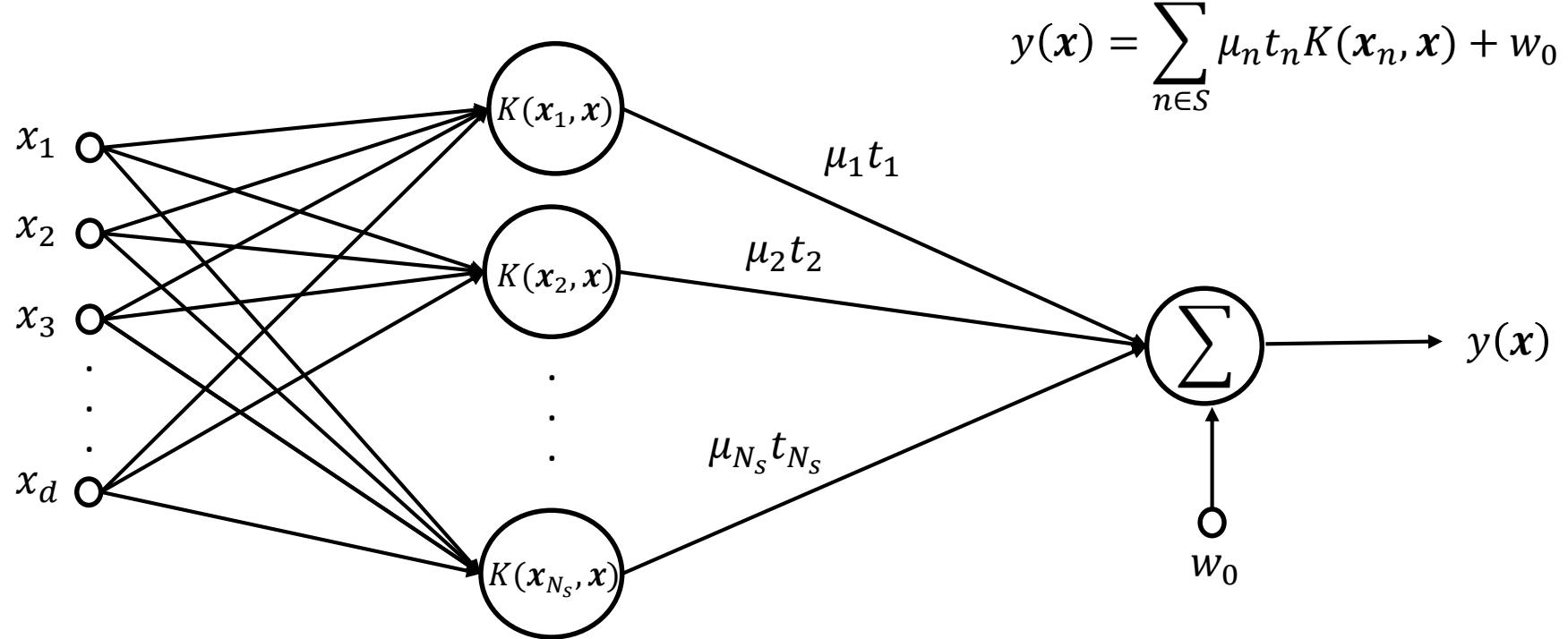
$$\text{decide } \omega_1 \text{ if } y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n \mathbf{x}_n^T \mathbf{x} + w_0 \geq 0$$

- With the kernel trick we can replace the inner product $\mathbf{x}_n^T \mathbf{x}$ with a kernel function leveraging the abovementioned benefits.
- Then, the decision rule becomes:

$$\text{decide } \omega_1 \text{ if } y(\mathbf{x}) = \sum_{n \in S} \mu_n t_n K(\mathbf{x}_n, \mathbf{x}) + w_0 \geq 0$$

Kernel Trick and SVMs

- Which can be modeled as a neural network of the form:



Multiclass SVMs

- SVMs are by definition **two-class classifiers**.
- To extend SVM for multiclass classification ($c > 2$) a number of approaches have been proposed the most common of which are:
 - *One-vs-the rest approach*: I have one SVM model for each class, i.e., $y_i(x), i = 1, \dots, c$. I make a decision based on:
$$y(x) = \max_i y_i(x)$$
 - *One-vs-one approach*: train $\frac{c(c-1)}{2}$ different two-class SVMs (One vs. one) and then classify test points according to which class has the **highest number of ‘votes’**.
 - There is a variety of other approaches in the literature (see for Example 7.1.3 in Bishop’s book).



ARISTOTLE UNIVERSITY OF THESSALONIKI



FACULTY OF ENGINEERING

Questions?

Pattern Recognition & Machine Learning

Support Vector Machines and Kernels