Foundations of Machine Learning
CentraleSupélec — Fall 2017

10. Support Vector Machines

Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Inserm
Institut national
de la santé et de la recherche médicale

institutCurie

PSL
RESEARCH UNIVERSITY PARIS

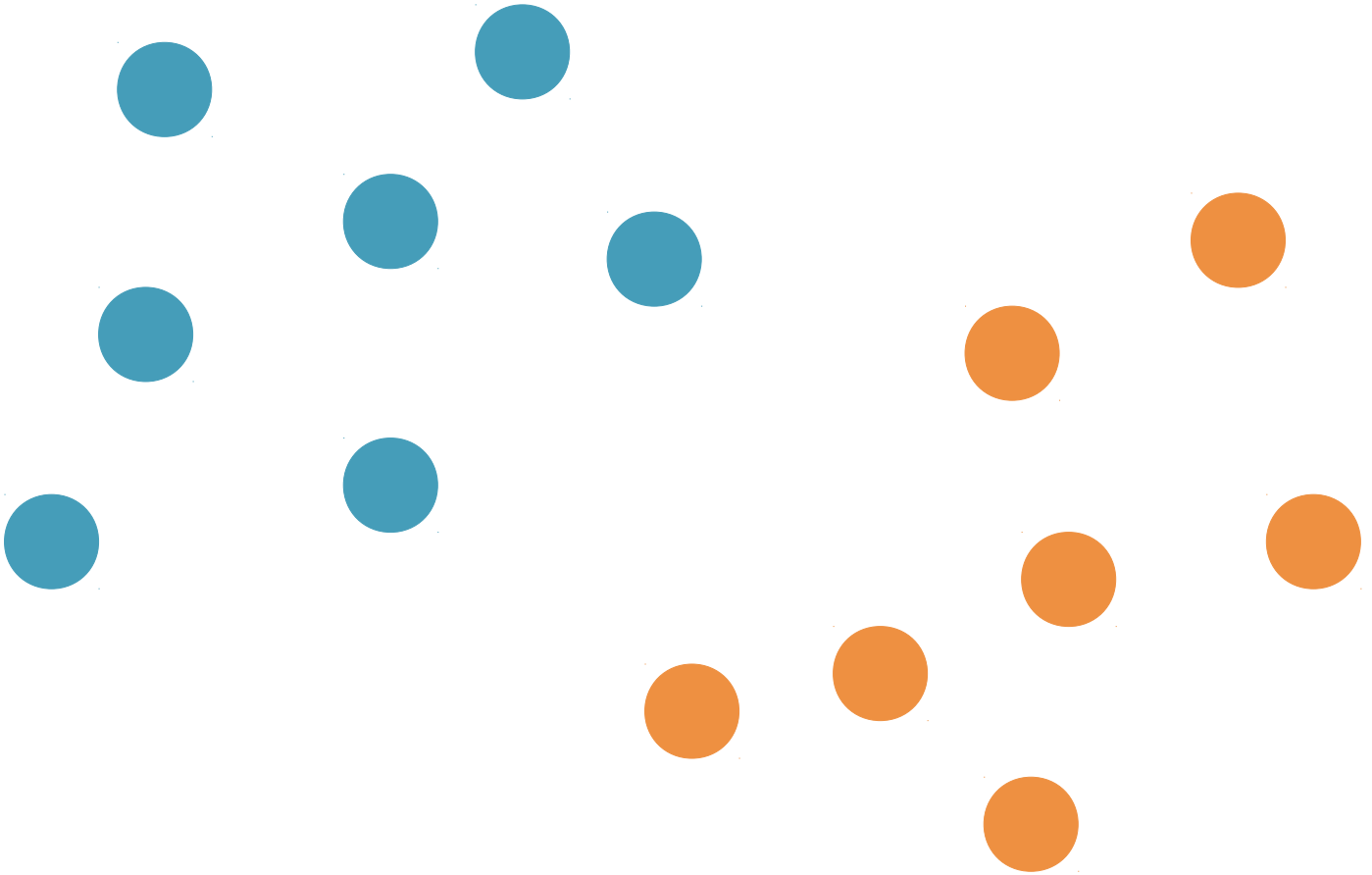
MINES
ParisTech

Learning objectives

- Define a **large-margin classifier** in the separable case.
- Write the corresponding **primal** and **dual** optimization problems.
- Re-write the optimization problem in the case of **non-separable data.**
- Use the **kernel trick** to apply soft-margin SVMs to **non-linear** cases.
- Define kernels for **real-valued data, strings, and graphs.**

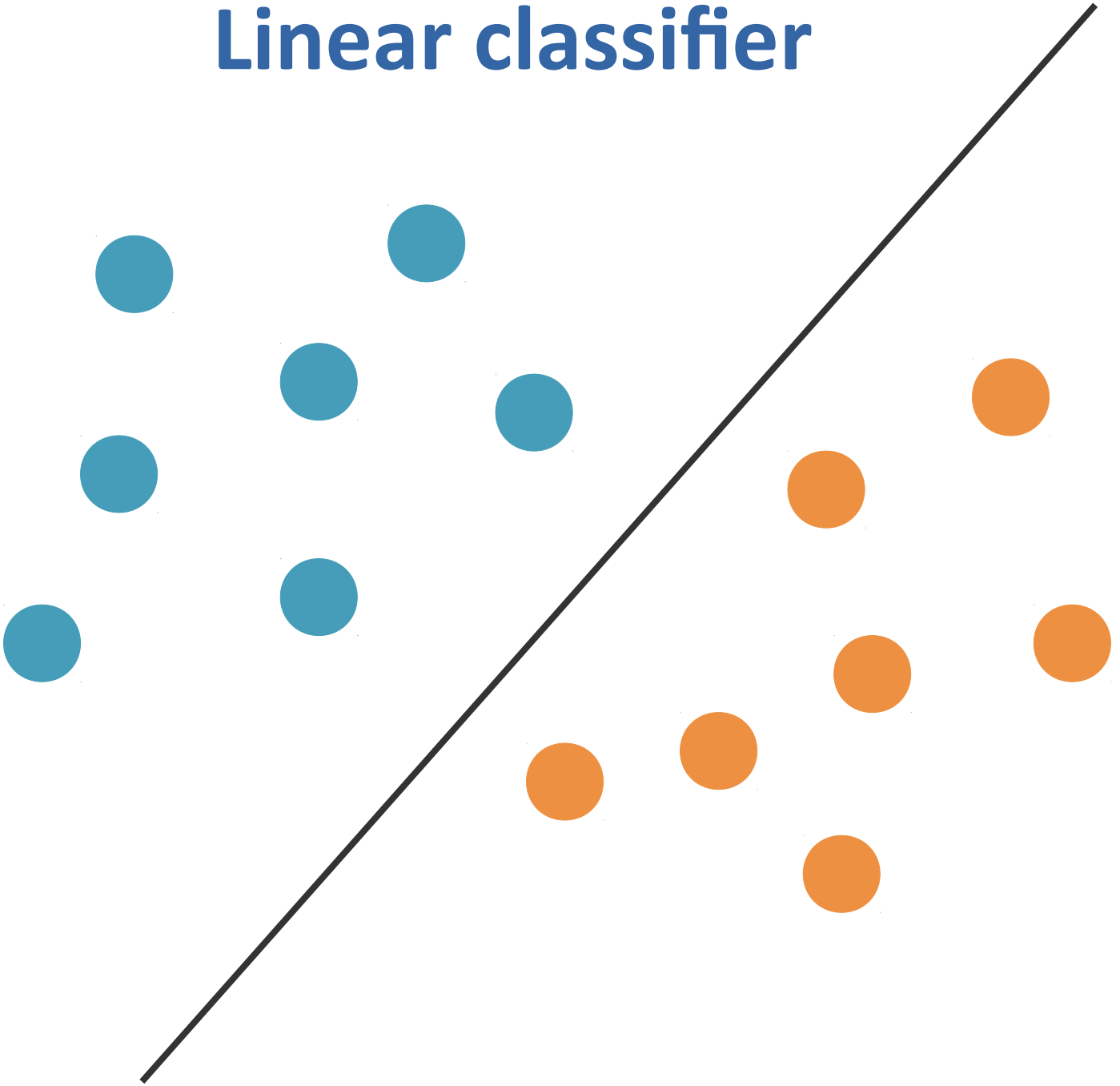
The linearly separable case: hard-margin SVMs

Linear classifier

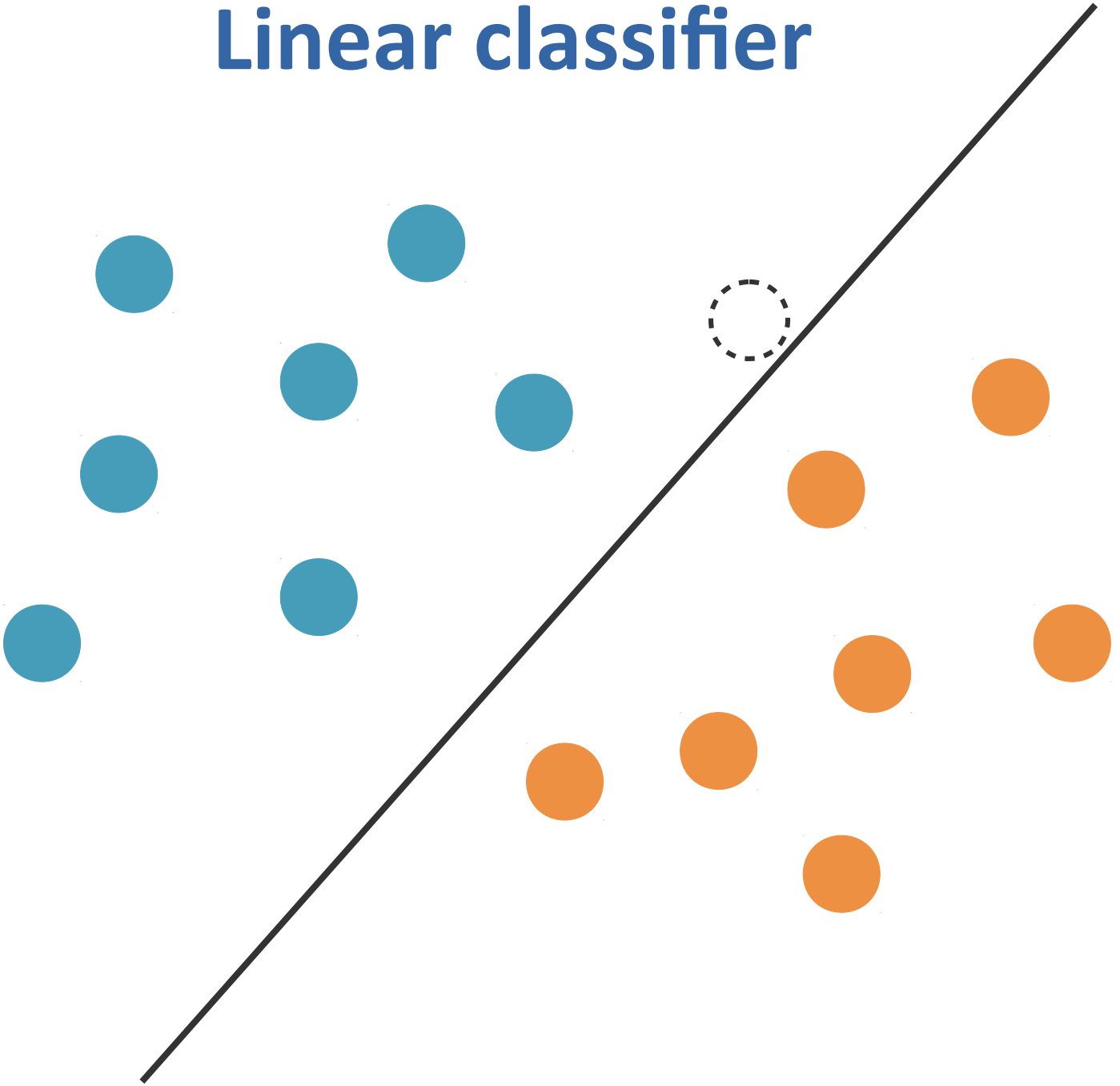


Assume data is **linearly separable**:
there exists a line that separates + from -

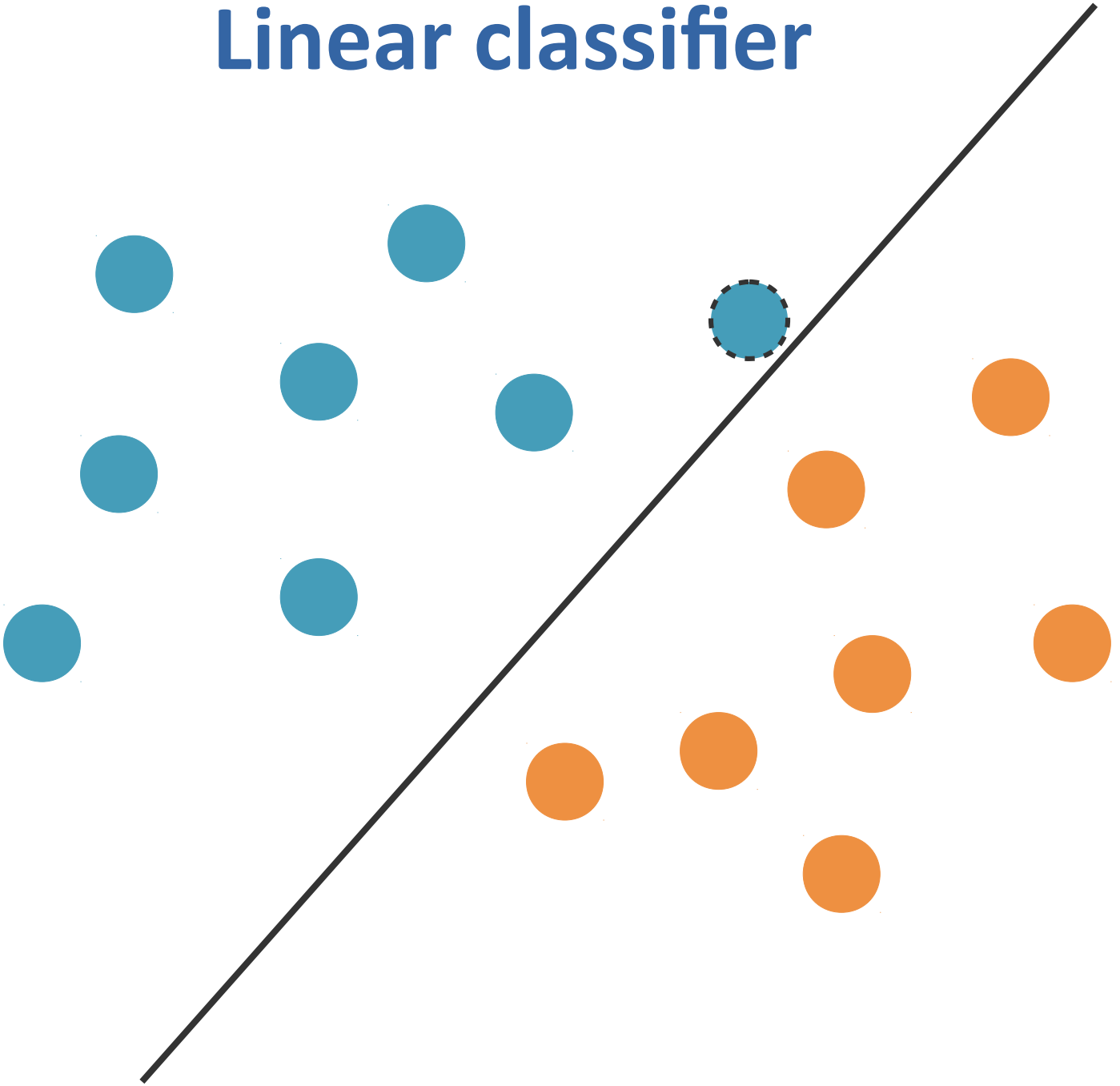
Linear classifier



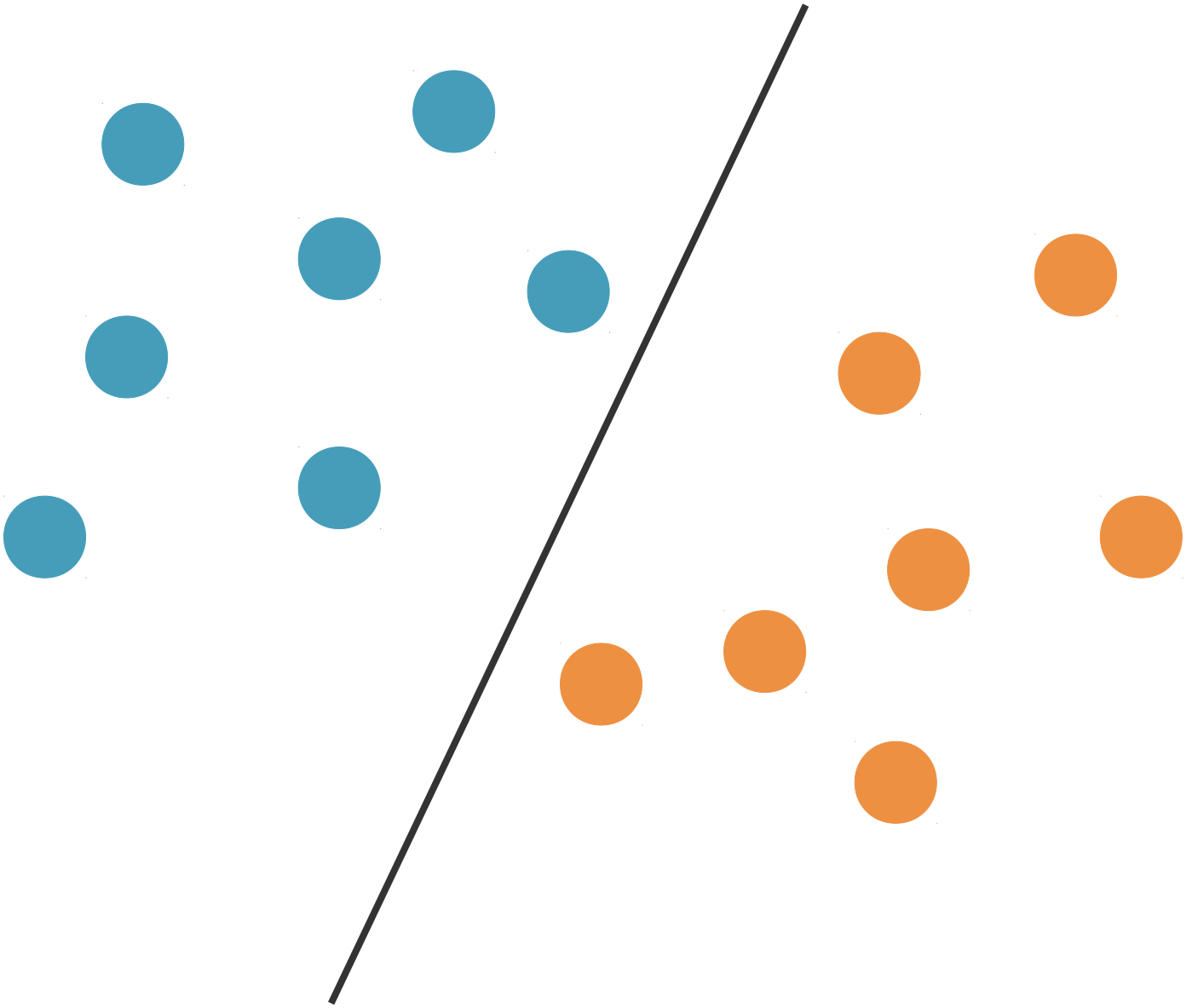
Linear classifier



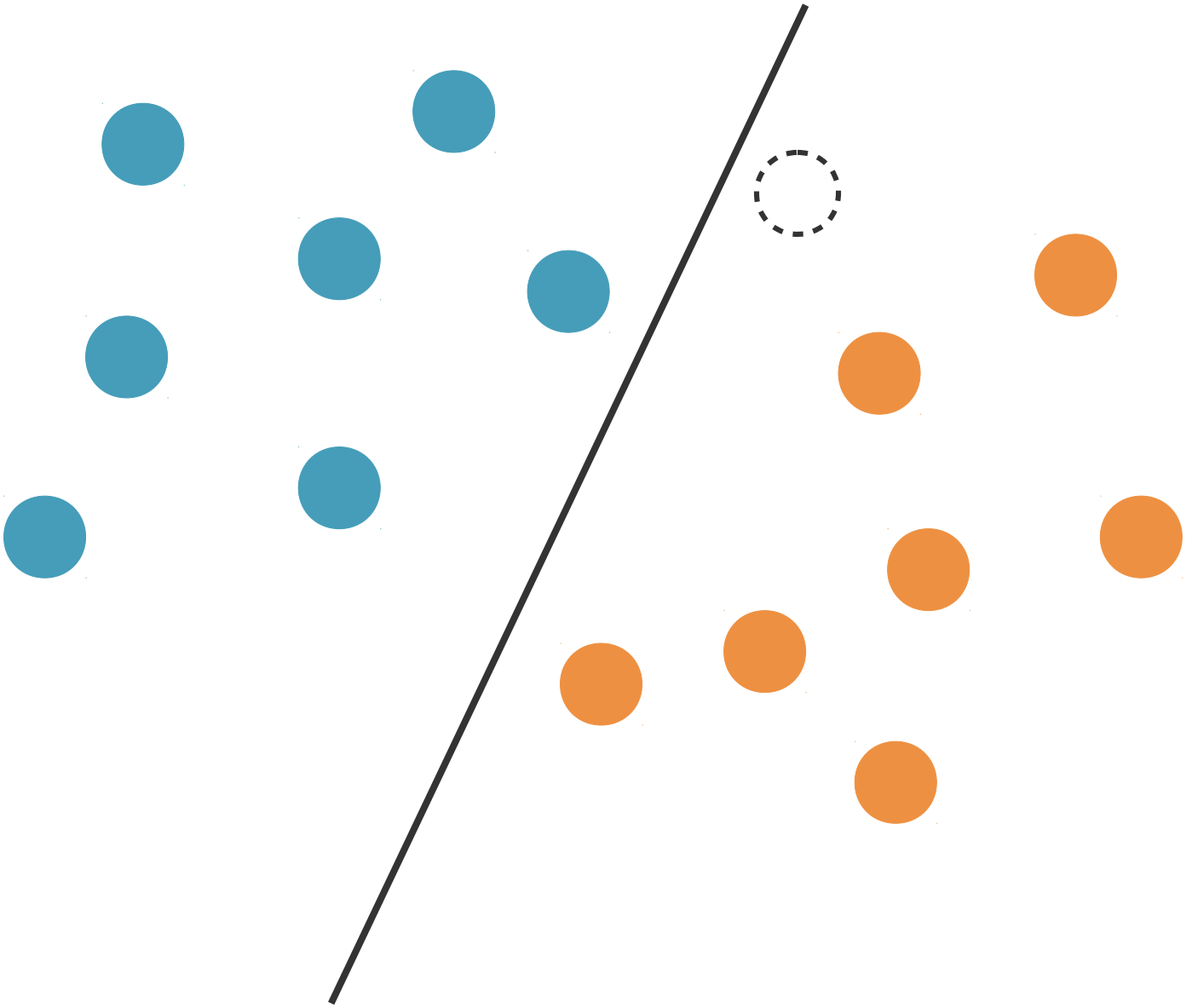
Linear classifier



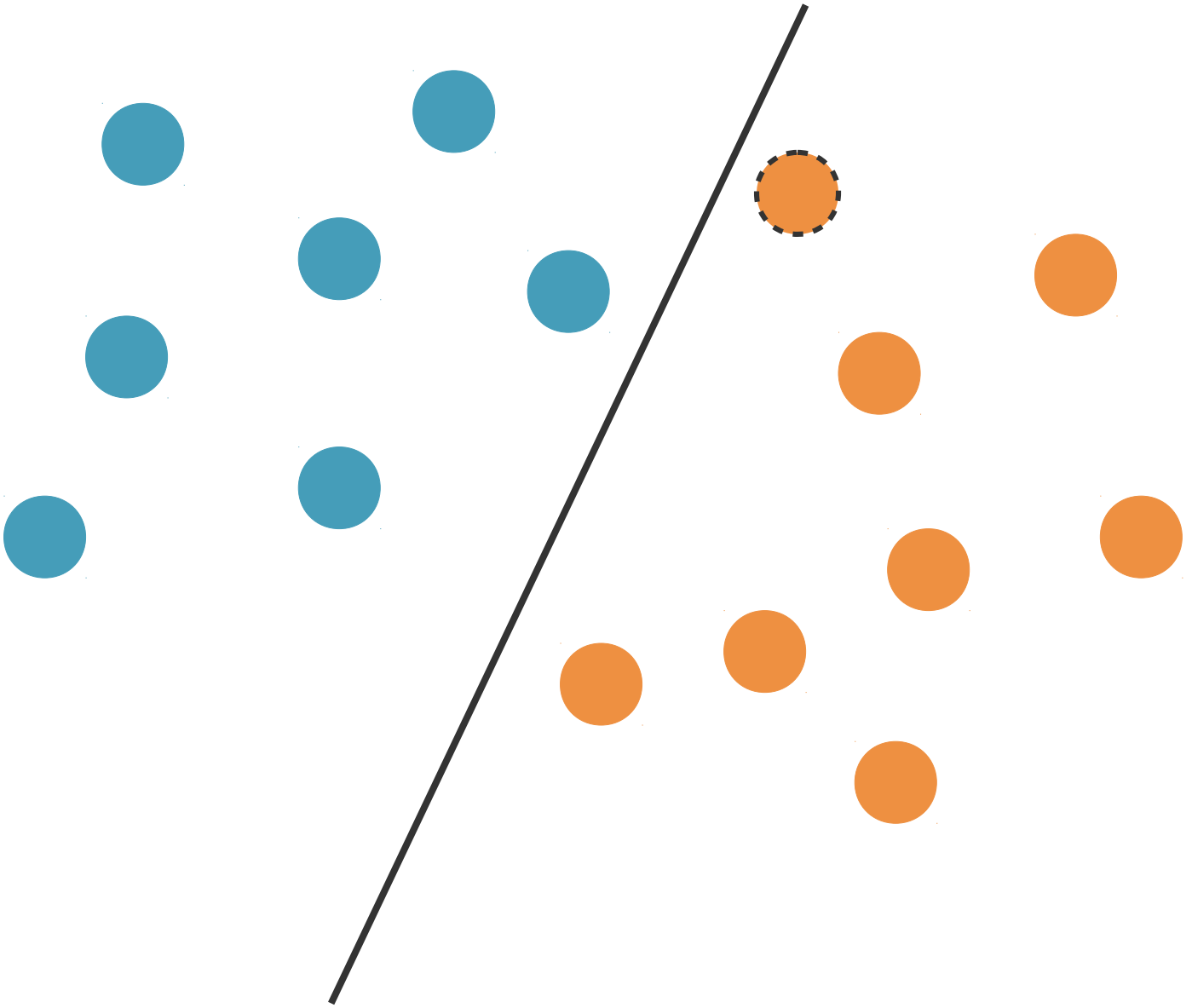
Linear classifier



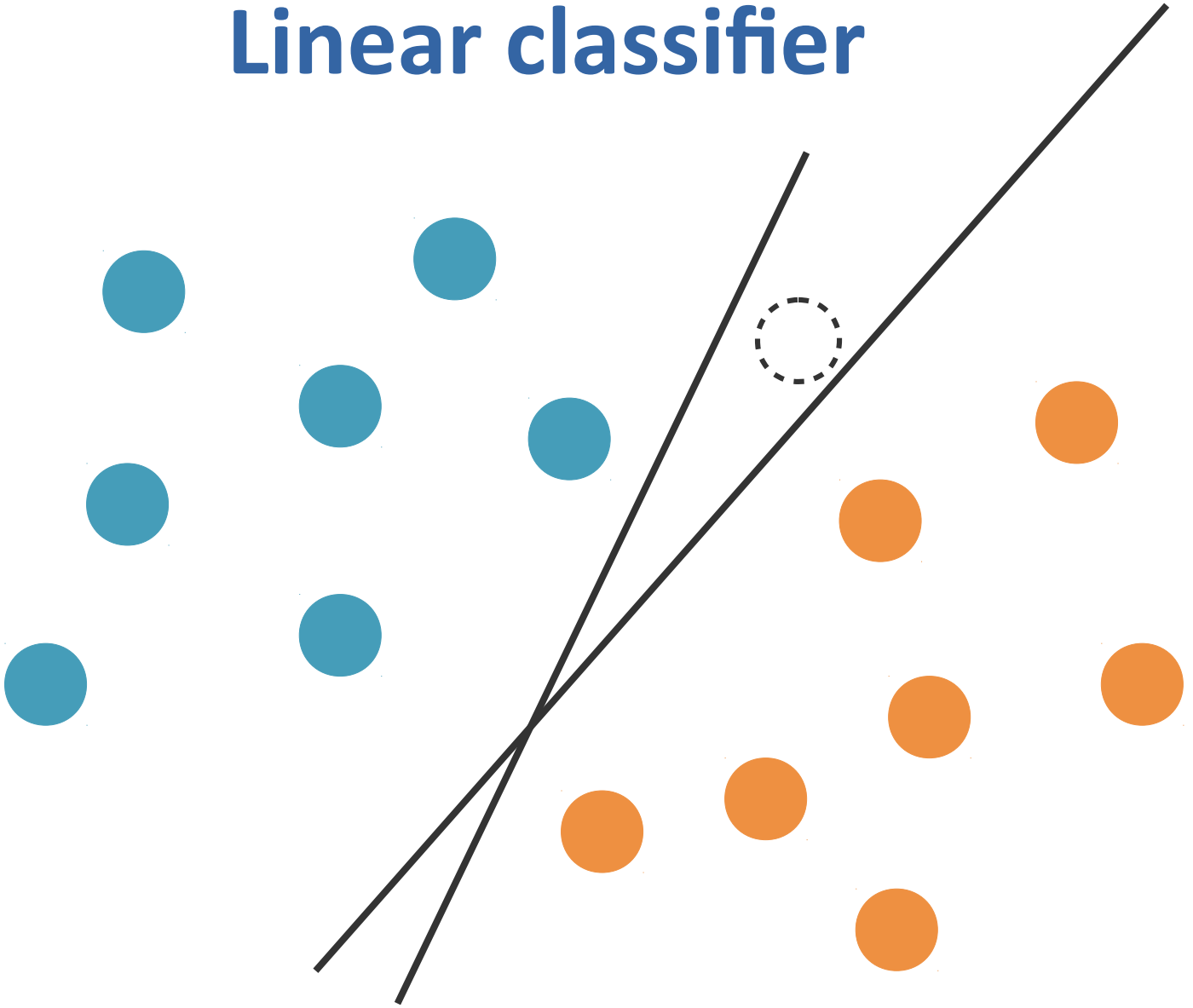
Linear classifier



Linear classifier

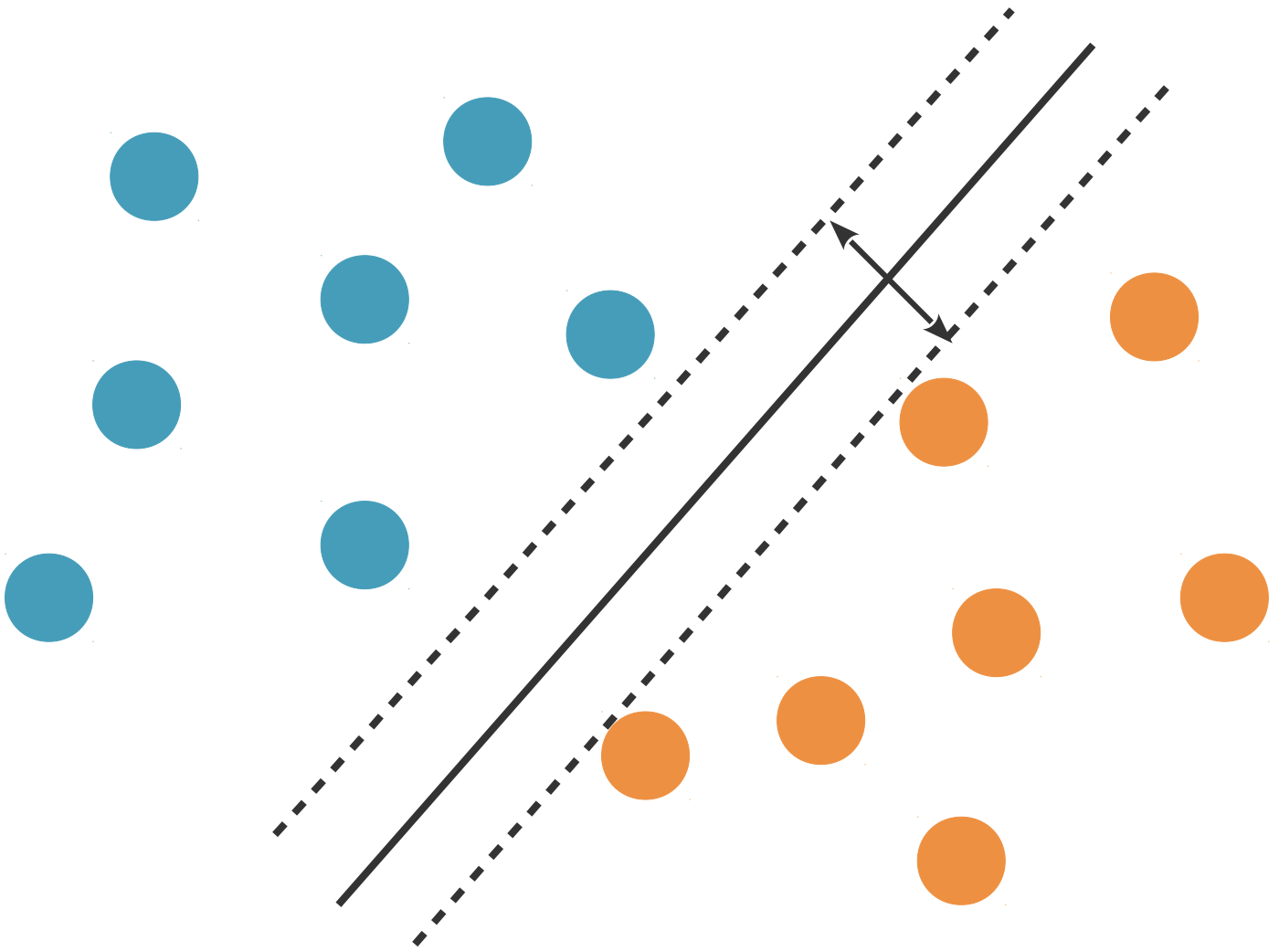


Linear classifier



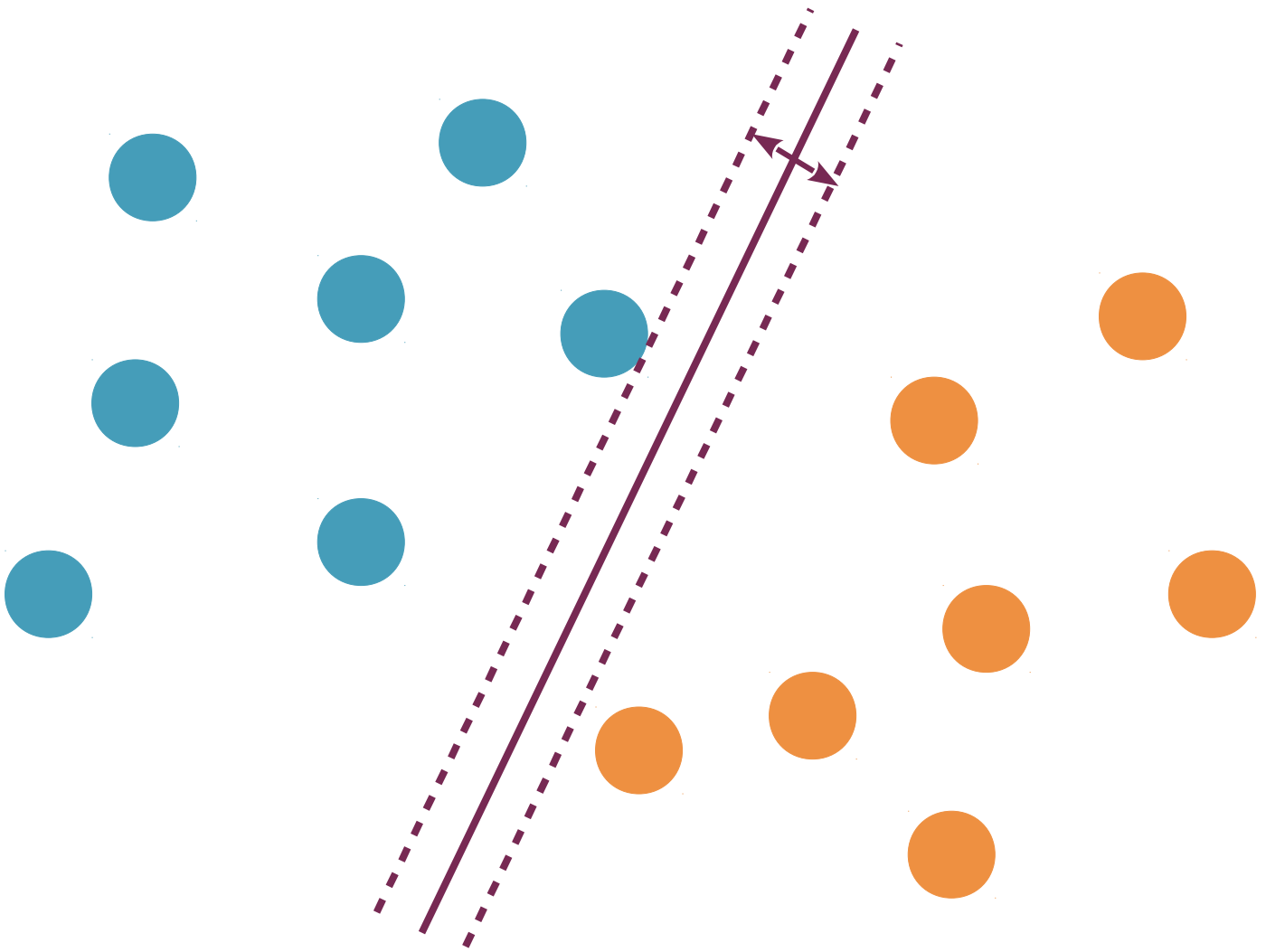
Which one is better?

Margin of a linear classifier

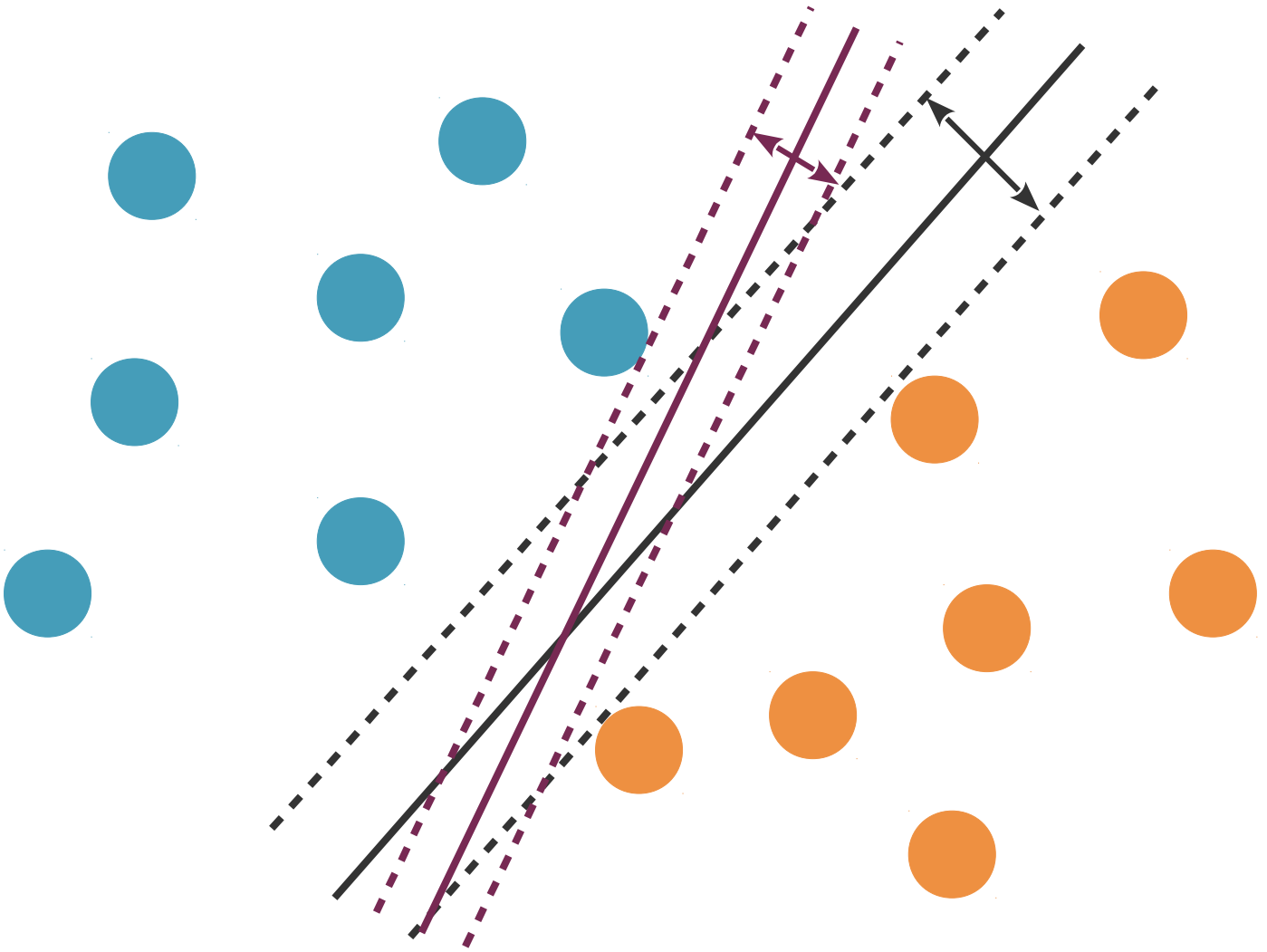


Margin: Twice the distance from the separating hyperplane to the closest training point.

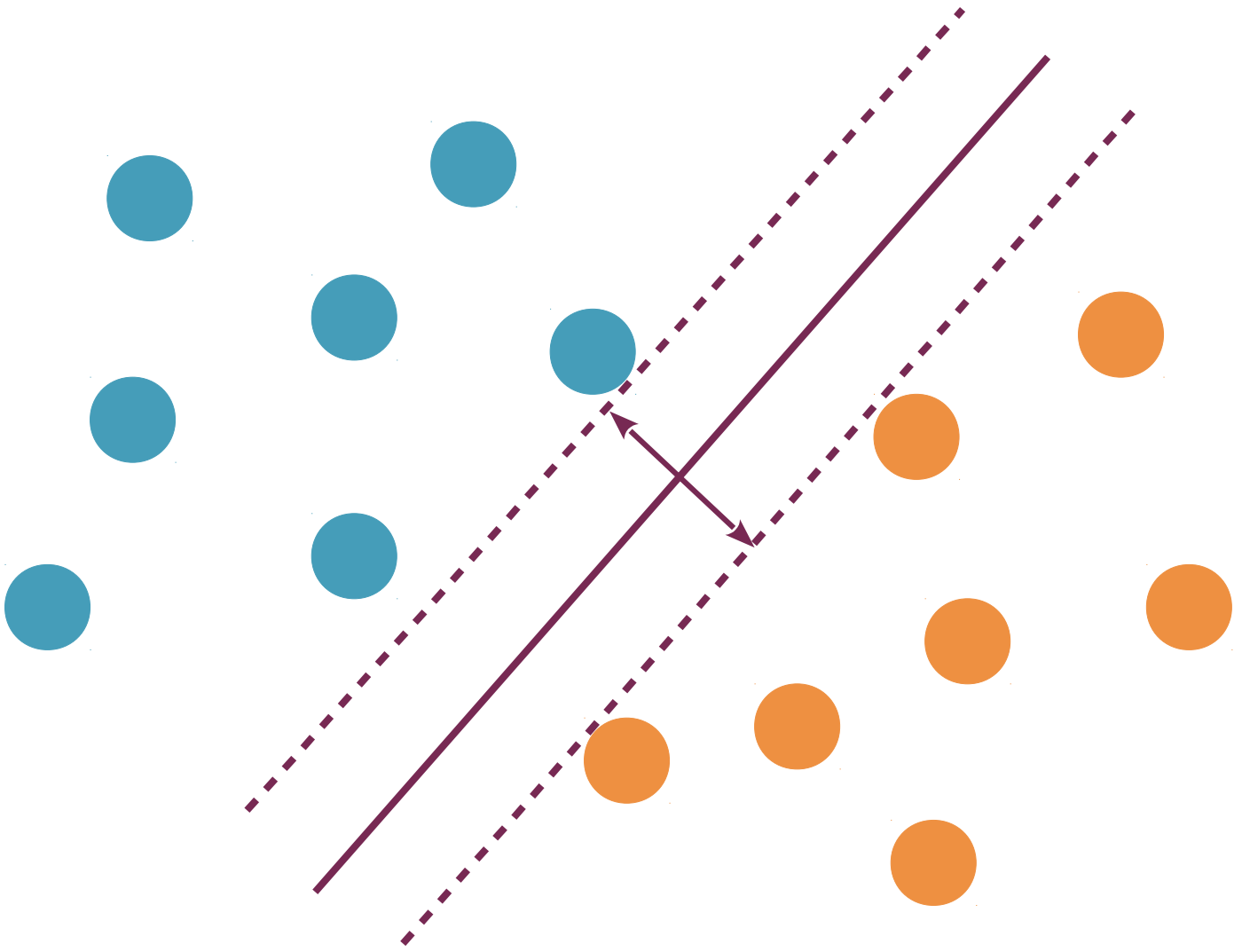
Margin of a linear classifier



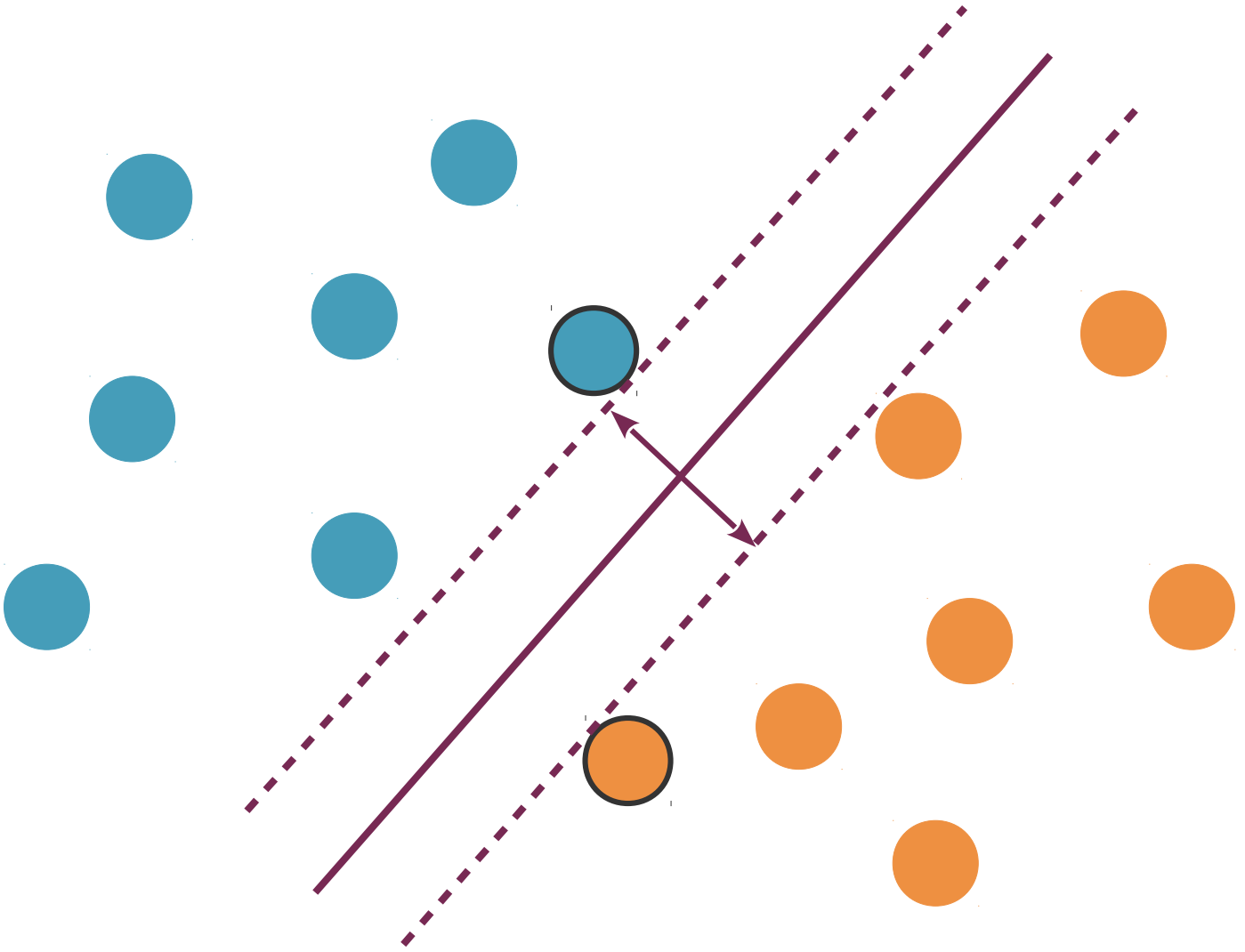
Margin of a linear classifier



Largest margin classifier: Support vector machines



Support vectors

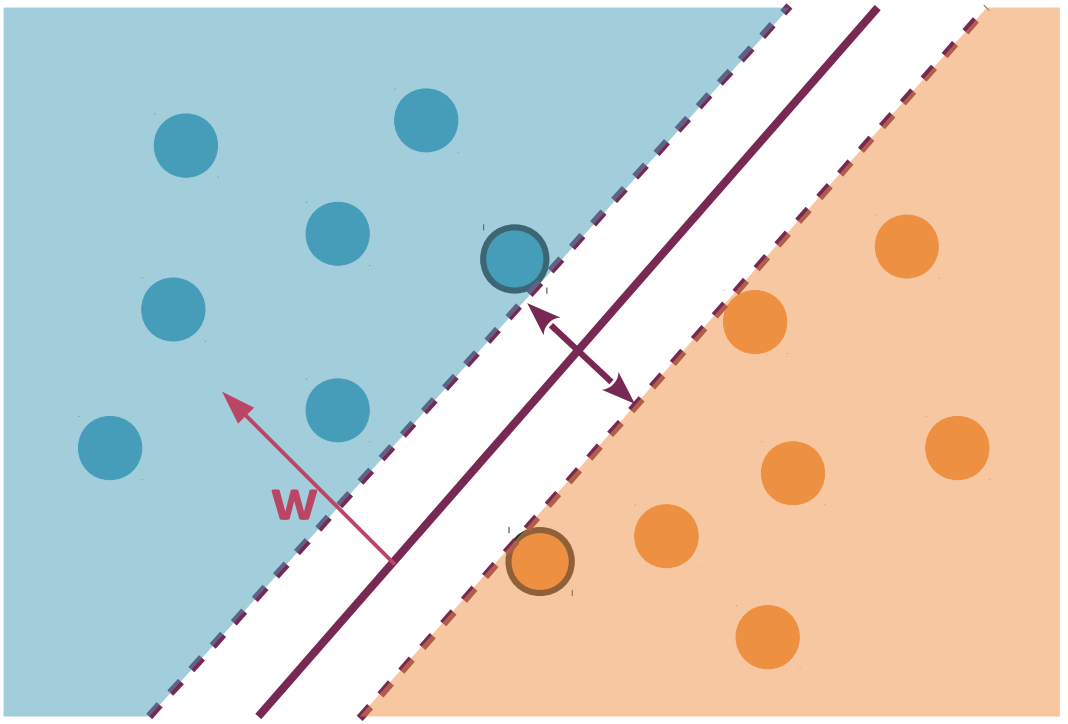


Formalization

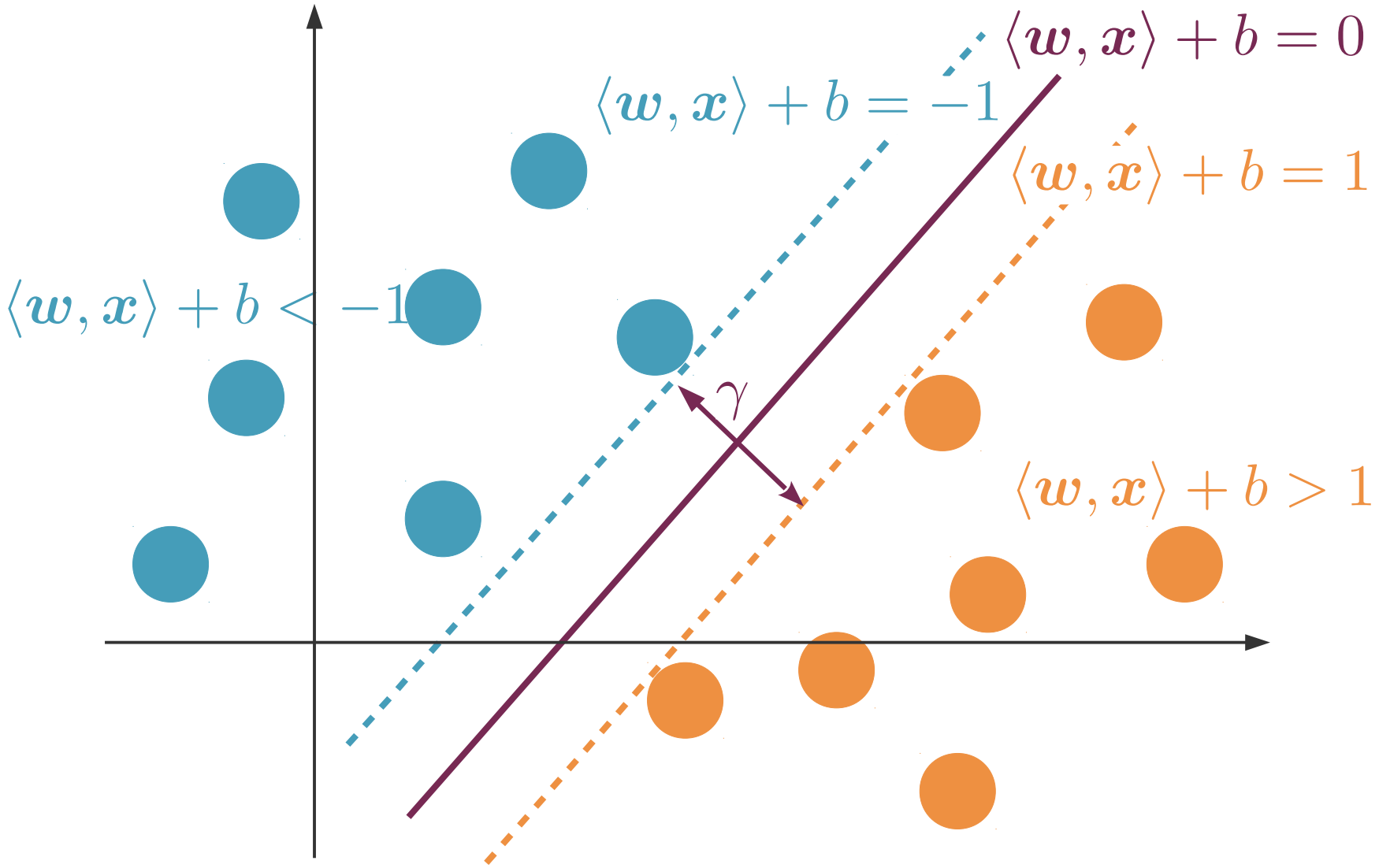
- **Training set**

$$\mathcal{D} = \{(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^n, y^n)\} \qquad \boldsymbol{x}^i \in \mathbb{R}^p \qquad y^i \in \{-1, +1\}$$

- **What are the equations of the 3 parallel hyperplanes?**
- **How is the "blue" region defined? The "orange" one?**

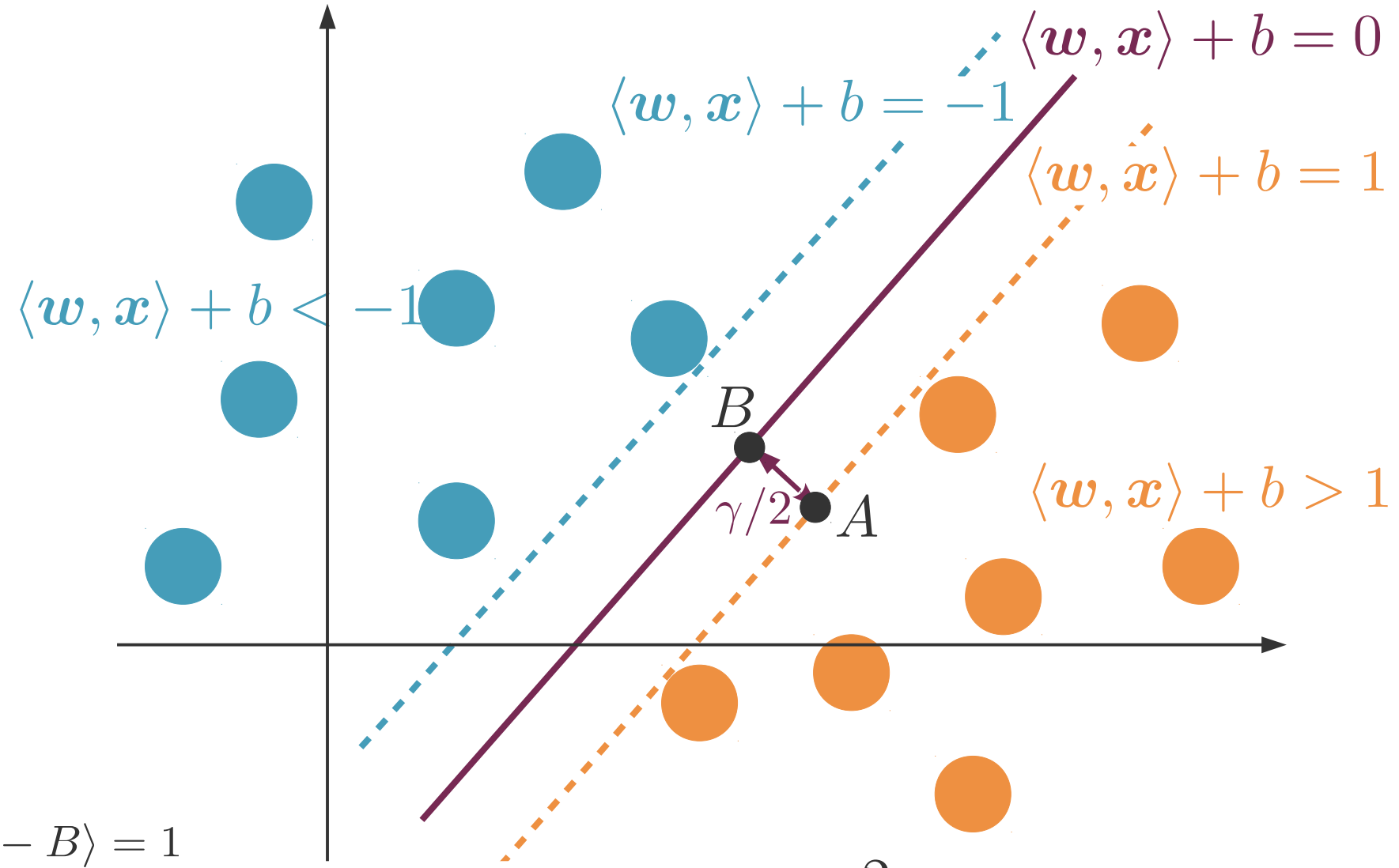


Largest margin hyperplane



What is the size of the margin γ?

Largest margin hyperplane



$\langle \boldsymbol{w}, A - B \rangle = 1$

$$\gamma = 2||A - B|| = \frac{2}{||\boldsymbol{w}||}$$

Optimization problem

- **Training set**

$$\mathcal{D} = \{(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^n, y^n)\} \qquad \boldsymbol{x}^i \in \mathbb{R}^p \qquad y^i \in \{-1, +1\}$$

- Assume the data to be **linearly separable**

$$\exists (\boldsymbol{w}, b) \in \mathbb{R}^p \times \mathbb{R} \text{ s.t. } \begin{cases} \langle \boldsymbol{w}, x^i \rangle + b > 0 & \text{if } y^i = +1 \\ \langle \boldsymbol{w}, x^i \rangle + b < 0 & \text{if } y^i = -1 \end{cases}$$

- Goal: Find (w^*, b^*) that define the hyperplane with largest margin.

Optimization problem

- **Margin maximization:**

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

- **Correct classification of the training points:**
  - For positive examples:

    $$y^i = 1 \text{ and } \langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b \geq 1$$

  - For negative examples:

    $$y^i = -1 \text{ and } \langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b \leq -1$$

  - Summarized as 

Optimization problem

- **Margin maximization:**

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

- **Correct classification of the training points:**
  - For positive examples:

    $$y^i = 1 \text{ and } \langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b \geq 1$$

  - For negative examples:

    $$y^i = -1 \text{ and } \langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b \leq -1$$

  - Summarized as: $y^i.(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) \geq 1$

- **Optimization problem:**

$$\arg \min_{\boldsymbol{w} \in \mathbb{R}^p, b \in \mathbb{R}} \frac{1}{2}||\boldsymbol{w}||^2 \text{ s.t. } y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1 \geq 0 \qquad \forall i \in \{1, \ldots, n\}$$

Optimization problem

- Find $(\boldsymbol{w}^*, b^*)$ that minimize $\frac{1}{2}||\boldsymbol{w}||^2$
  under the n constraints $y^i(\langle \boldsymbol{w}, x^i \rangle + b) - 1 \geq 0$

Optimization problem

- Find $(\boldsymbol{w}^*, b^*)$ that minimize $\frac{1}{2}||\boldsymbol{w}||^2$
  under the n constraints $y^i(\langle \boldsymbol{w}, x^i \rangle + b) - 1 \geq 0$

- We introduce one **dual variable** α_i for each constraint (i.e. each training point)
- **Lagrangian:**

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1 \right).$$

$$\boldsymbol{w} \in \mathbb{R}^p \quad \boldsymbol{\alpha} \in \mathbb{R}_+^n \quad b \in \mathbb{R}$$

Lagrange dual of the SVM

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right)$$

- **Lagrange dual function:** $q : \mathbb{R}^n \to \mathbb{R}$

$$q(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w}, b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})$$

- **Lagrange dual problem:**

$$\max_{\boldsymbol{\alpha}} \inf_{\boldsymbol{w}, b} \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right)$$

- **Strong duality:** Under **Slater's conditions,** the optimum of the primal is the optimum of the dual.

The function to optimize is convex and the equality constraints are affine.

Minimizing the Lagrangian of the SVM

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right)$$

- L(**w**, b, **α**) is **convex quadratic in w** and minimized for 

Minimizing the Lagrangian of the SVM

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right)$$

- L(**w**, b, **α**) is **convex quadratic in w** and minimized for:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, b^*, \boldsymbol{\alpha}) = 0 \Leftrightarrow \boldsymbol{w}^* - \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i = 0 \Leftrightarrow \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i.$$

Minimizing the Lagrangian of the SVM

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right)$$

- L(**w**, b, **α**) is **convex quadratic in w** and minimized for:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, b^*, \boldsymbol{\alpha}) = 0 \Leftrightarrow \boldsymbol{w}^* - \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i = 0 \Leftrightarrow \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i.$$

- L(**w**, b, **α**) is **affine in b**. Its minimum is $-\infty$ except if:

Minimizing the Lagrangian of the SVM

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right)$$

- L(**w**, b, **α**) is **convex quadratic in w** and minimized for:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, b^*, \boldsymbol{\alpha}) = 0 \Leftrightarrow \boldsymbol{w}^* - \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i = 0 \Leftrightarrow \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i.$$

- L(**w**, b, **α**) is **affine in b**. Its minimum is $-\infty$ except if:

$$\nabla_b L(\boldsymbol{w}^*, b^*, \alpha) = 0 \Leftrightarrow \sum_{i=1}^{n} \alpha_i y^i = 0.$$

SVM dual problem

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1\right) \qquad \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y^i \boldsymbol{x}^i$$

$$L(\boldsymbol{w}^*, b, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - b \sum_{i=1}^{n} \alpha_i y^i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle \qquad \sum_{i=1}^{n} \alpha_i y^i = 0$$

$$L(\boldsymbol{w}^*, b, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle$$

- **Lagrange dual function:**

$$\begin{aligned} q(\boldsymbol{\alpha}) &= \inf_{\boldsymbol{w} \in \mathbb{R}^p, b \in \mathbb{R}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) \\ &= \begin{cases} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle & \text{if } \sum_{i=1}^{n} \alpha_i y^i = 0 \\ -\infty & \text{otherwise.} \end{cases} \end{aligned}$$

- **Dual problem:** maximize q(**α**) subject to **α** ≥ 0.

  Maximizing a quadratic function under box constraints can be solved efficiently using dedicated software.

Optimal hyperplane

- Once the optimal **α*** is found, we recover (**w***, b*)

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y^i \boldsymbol{x}^i$$

$$b^* = \frac{\max_{i:y^i=-1} \langle \boldsymbol{w}^*, \boldsymbol{x}^i \rangle + \min_{i:y^i=+1} \langle \boldsymbol{w}^*, \boldsymbol{x}^i \rangle}{2}$$

- Determining b*:
  - Closest positive point to the separating hyperplane: $\arg \min_{i:y^i=+1} \langle \boldsymbol{w}^*, \boldsymbol{x}^i \rangle$ verifies $\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^* = 1$
  - Closest negative point to the separating hyperplane: $\arg \max_{i:y^i=-1} \langle \boldsymbol{w}^*, \boldsymbol{x}^i \rangle$ verifies $\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^* = -1$
- The **decision function** is hence:

$$\begin{aligned} f^*(\boldsymbol{x}) &= \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^* \\ &= \sum_{i=1}^{n} \alpha_i^* y^i \langle \boldsymbol{x}^i, \boldsymbol{x} \rangle + b^* \end{aligned}$$

Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

abusive notation: g(w, b)

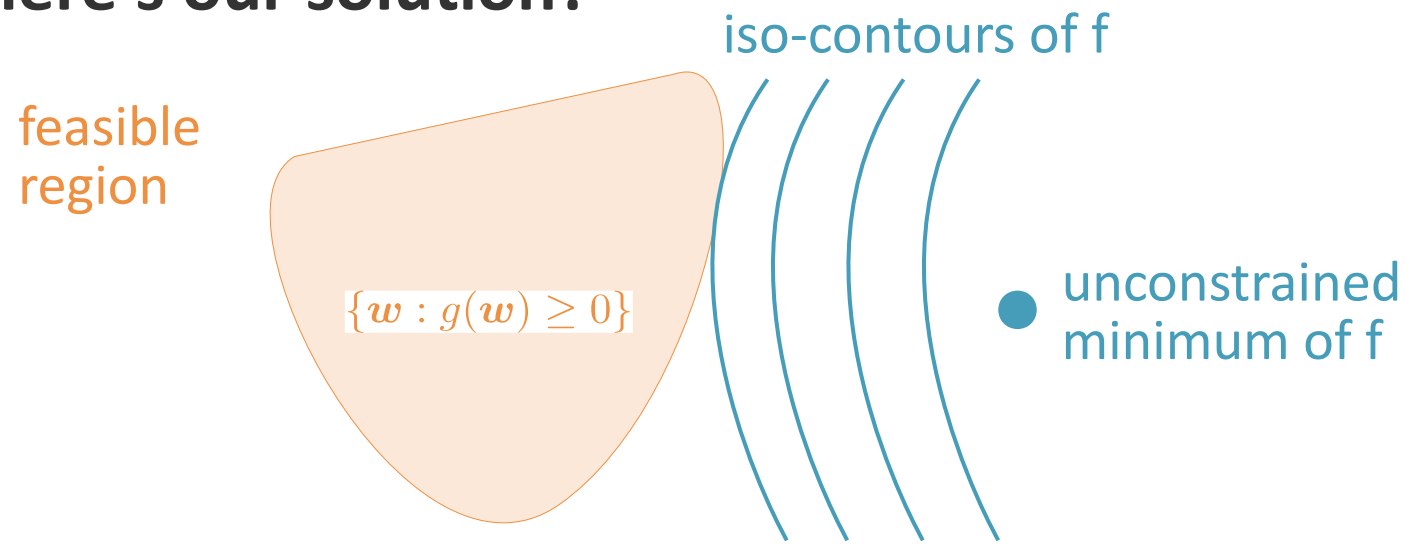
How do we write this in terms of the gradients of f and g?

Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

If the minimum of f(w) doesn't lie in the feasible region, where's our solution?

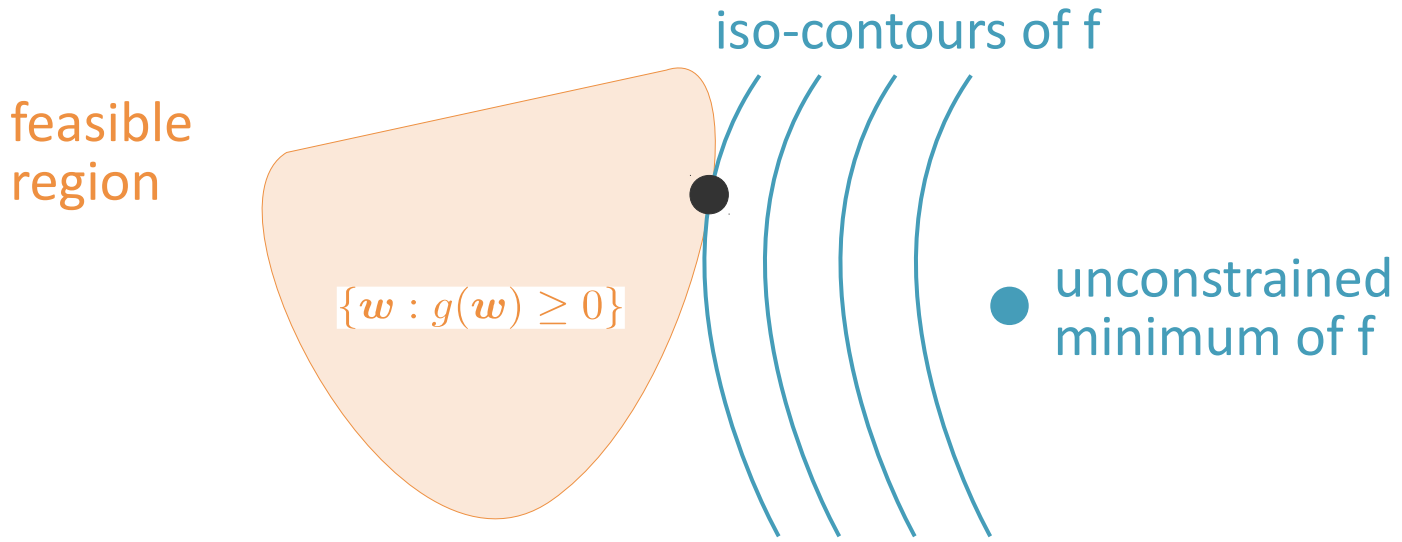


Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

How do we write this in terms of the gradients of f and g?

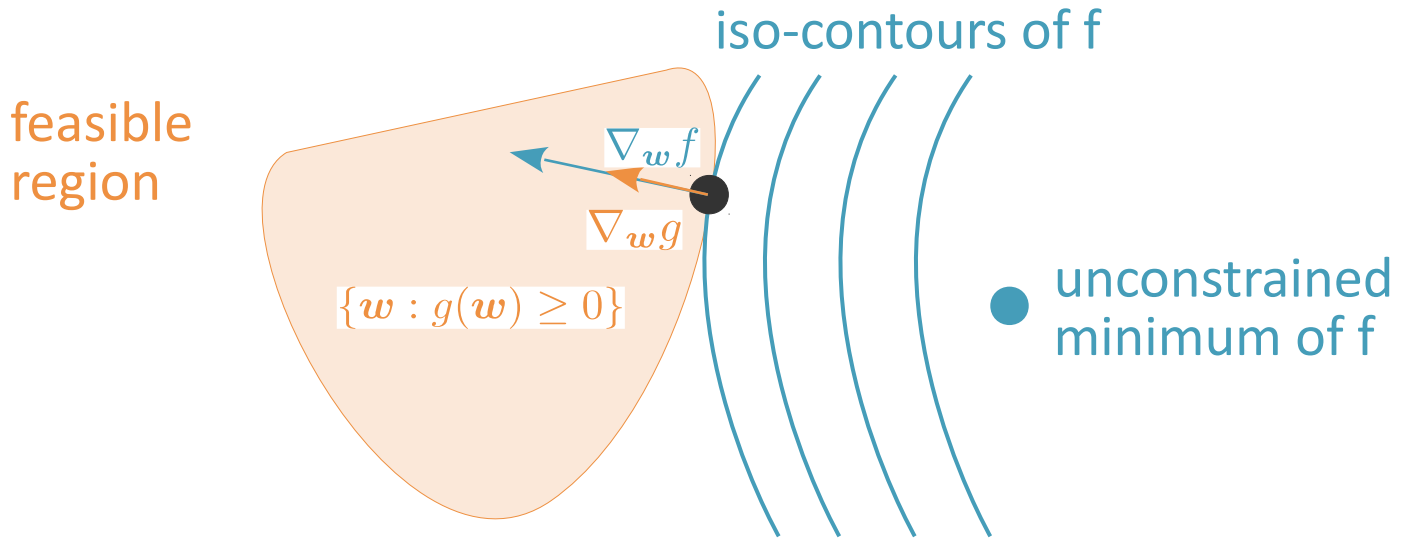


Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

How do we write this in terms of the gradients of f and g?

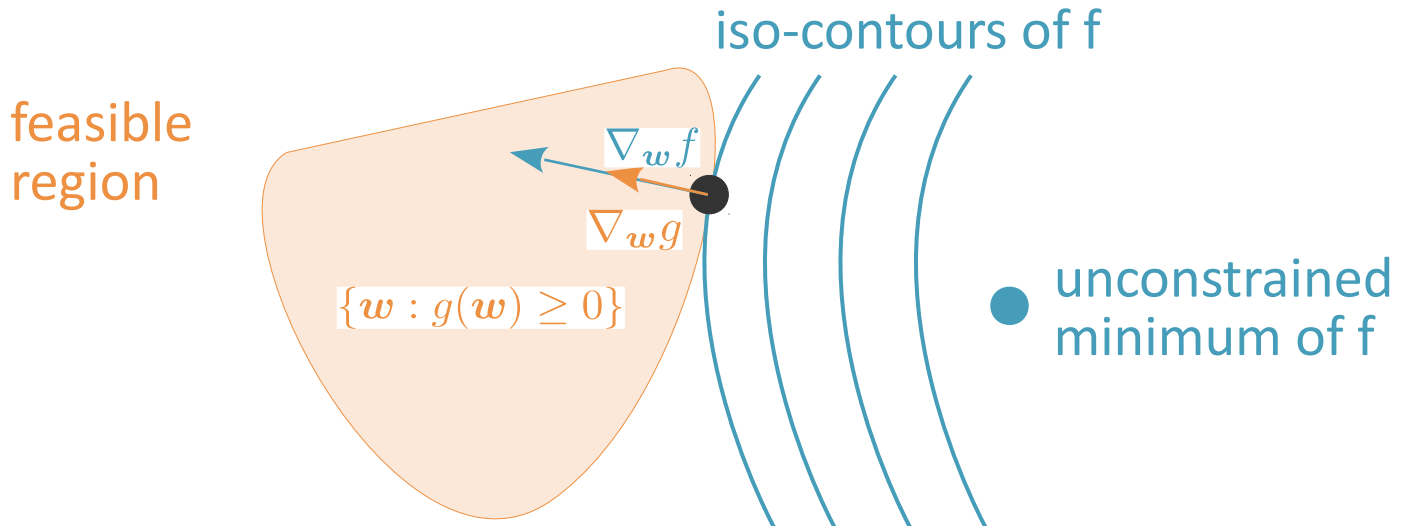


Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

How do we write this in terms of the gradients of f and g?



$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \alpha \nabla g(\boldsymbol{w}) \text{ and } g(\boldsymbol{w}) = 0, \alpha > 0$$

Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

Case 1: the unconstraind minimum lies in the feasible region.

Case 2: it does not.

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \alpha \nabla g(\boldsymbol{w}) \text{ and } g(\boldsymbol{w}) = 0, \alpha > 0$$

How do we summarize both cases?

Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

Case 1: the unconstraind minimum lies in the feasible region.

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = 0 \text{ and } g(\boldsymbol{w}) \geq 0$$

Case 2: it does not.

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \alpha \nabla g(\boldsymbol{w}) \text{ and } g(\boldsymbol{w}) = 0, \alpha > 0$$

– **Summarized as:**

$$\begin{cases} \nabla_{\boldsymbol{w}}(f(\boldsymbol{w}) - \alpha g(\boldsymbol{w})) = 0 \\ \alpha g(\boldsymbol{w}) = 0 \end{cases} \quad \text{and } \alpha \geq 0.$$

Lagrangian

- **minimize f(w) under the constraint g(w) ≥ 0**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) - 1$$

$$\begin{cases} \nabla_{\boldsymbol{w}}(f(\boldsymbol{w}) - \alpha g(\boldsymbol{w})) = 0 \\ \alpha g(\boldsymbol{w}) = 0 \end{cases} \quad \text{and } \alpha \geq 0.$$

Lagrangian: $L(\boldsymbol{w}, \alpha) = f(\boldsymbol{w}) - \alpha g(\boldsymbol{w})$

α is called the **Lagrange multiplier.**

Lagrangian

- **minimize f(w) under the constraints $g_i(w) \geq 0$**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g_i(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1$$

$$\begin{cases} \nabla_{\boldsymbol{w}}(f(\boldsymbol{w}) - \alpha_i g_i(\boldsymbol{w})) = 0 \\ \alpha_i g_i(\boldsymbol{w}) = 0 \end{cases} \quad \text{and } \alpha_i \geq 0.$$

How do we deal with n constraints?

Lagrangian

- **minimize f(w) under the constraints $g_i(w) \geq 0$**

$$f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 \qquad g_i(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1$$

$$\begin{cases} \nabla_{\boldsymbol{w}}(f(\boldsymbol{w}) - \alpha_i g_i(\boldsymbol{w})) = 0 \\ \alpha_i g_i(\boldsymbol{w}) = 0 \end{cases} \quad \text{and } \alpha_i \geq 0.$$

Use n Lagrange multipliers

- **Lagrangian:**

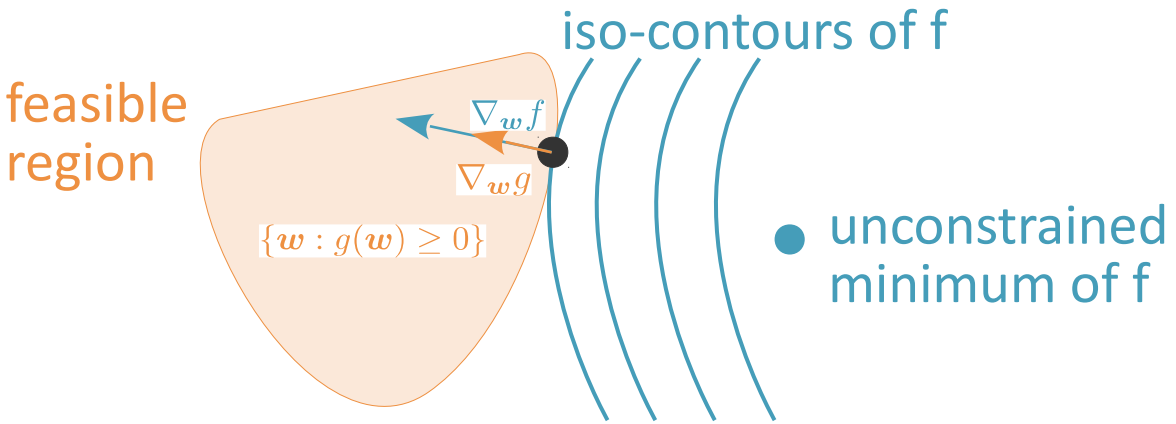
$$L(\boldsymbol{w}, \boldsymbol{\alpha}) = f(\boldsymbol{w}) - \sum_{i=1}^{n} \alpha_i g_i(\boldsymbol{w})$$

Support vectors

- **Karun-Kush-Tucker conditions:**

  Either $\alpha_i = 0$ (case 1) or $g_i = 0$ (case 2)

$$g_i(\boldsymbol{w}) = y(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) - 1 \quad \begin{cases} \nabla_{\boldsymbol{w}}(f(\boldsymbol{w}) - \alpha_i g_i(\boldsymbol{w})) = 0 \\ \alpha_i g_i(\boldsymbol{w}) = 0 \end{cases} \quad \text{and } \alpha_i \geq 0.$$

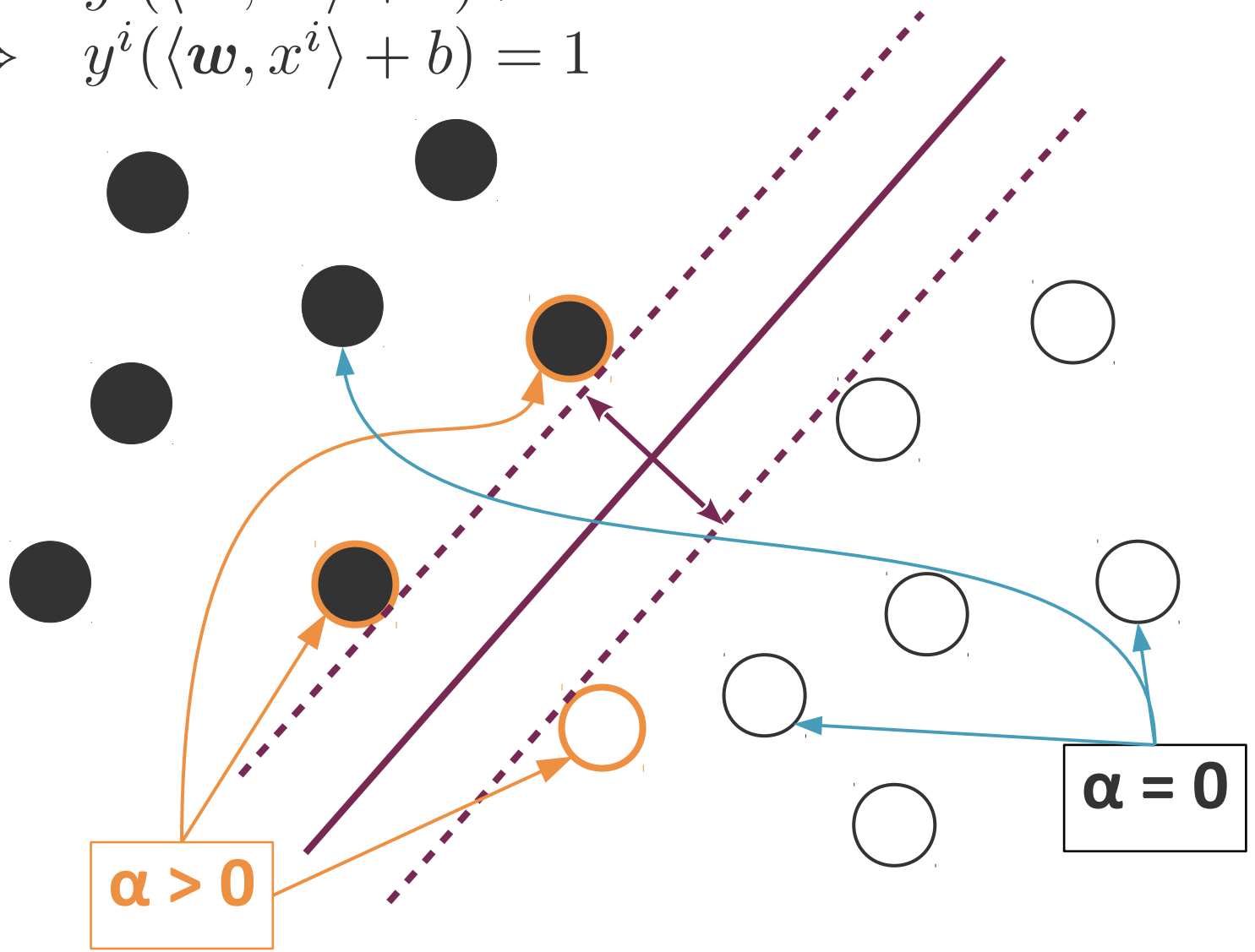


Case 1: $\alpha_i = 0 \quad \Rightarrow \quad y^i(\langle \boldsymbol{w}, x^i \rangle + b) > 1$

Case 2: $\alpha_i > 0 \quad \Rightarrow \quad y^i(\langle \boldsymbol{w}, x^i \rangle + b) = 1$

Support vectors

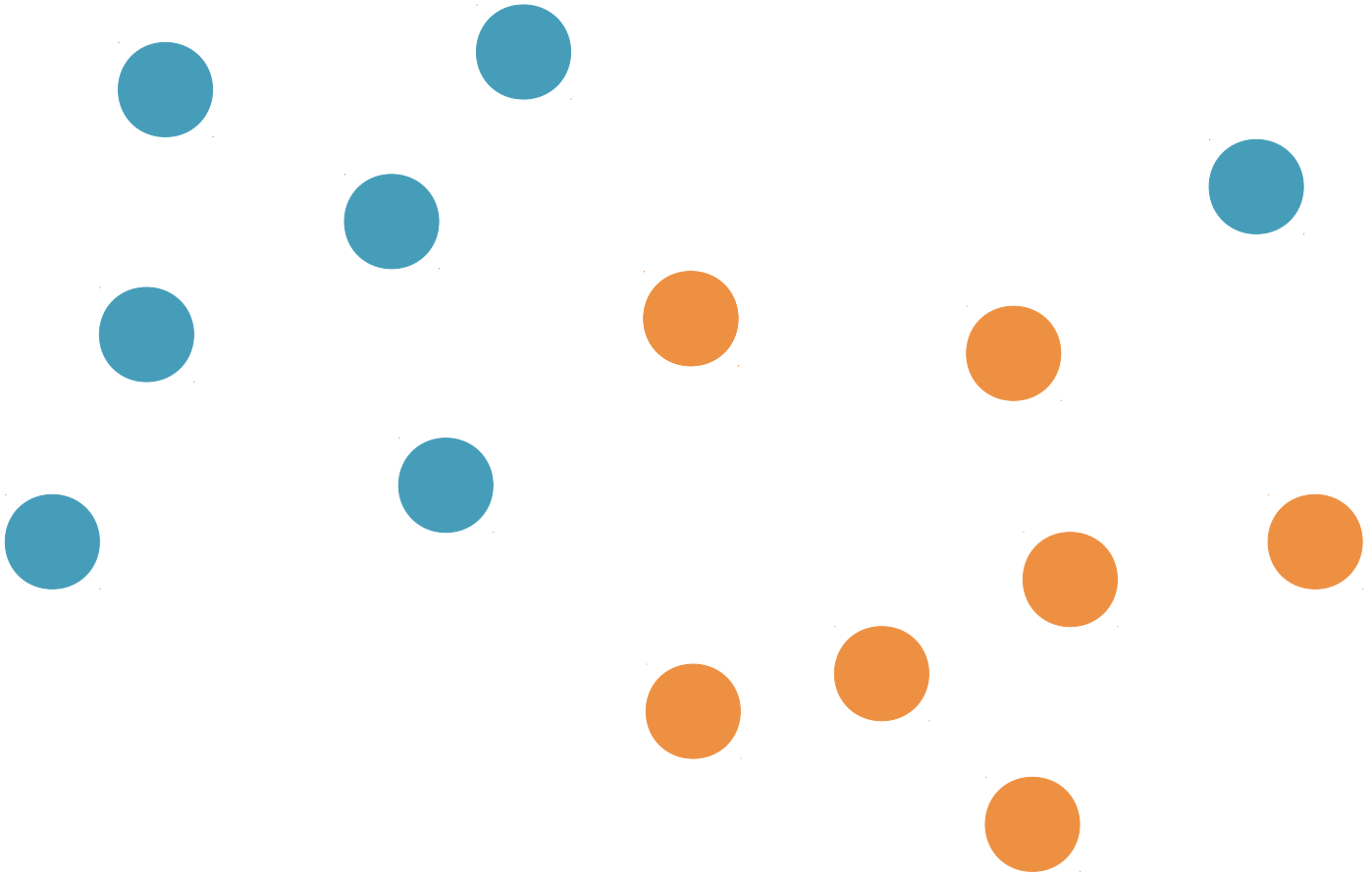
$$\alpha_i = 0 \quad \Rightarrow \quad y^i(\langle \boldsymbol{w}, x^i \rangle + b) > 1$$
$$\alpha_i > 0 \quad \Rightarrow \quad y^i(\langle \boldsymbol{w}, x^i \rangle + b) = 1$$



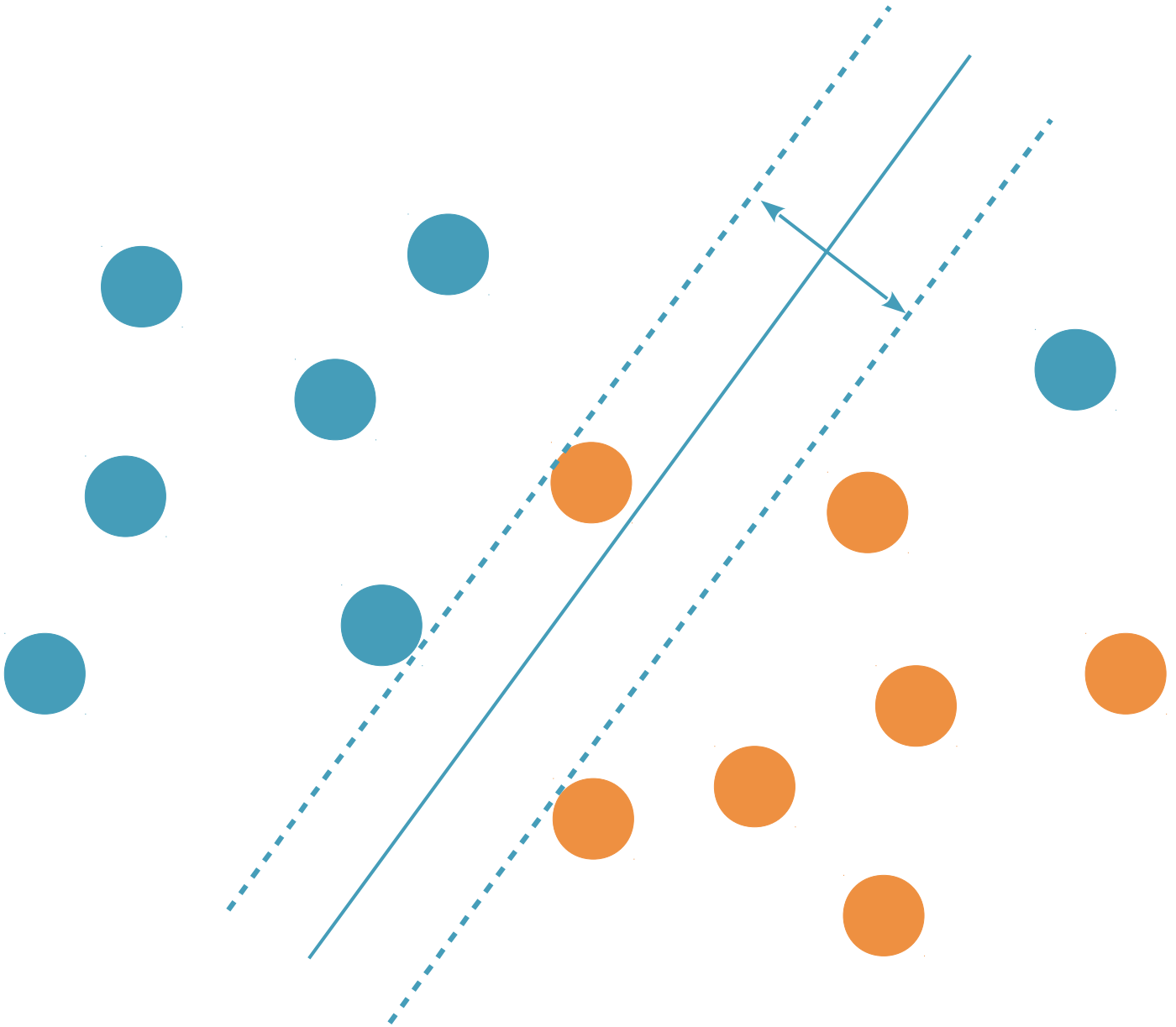
The non-linearly separable case: soft-margin SVMs.

Soft-margin SVMs

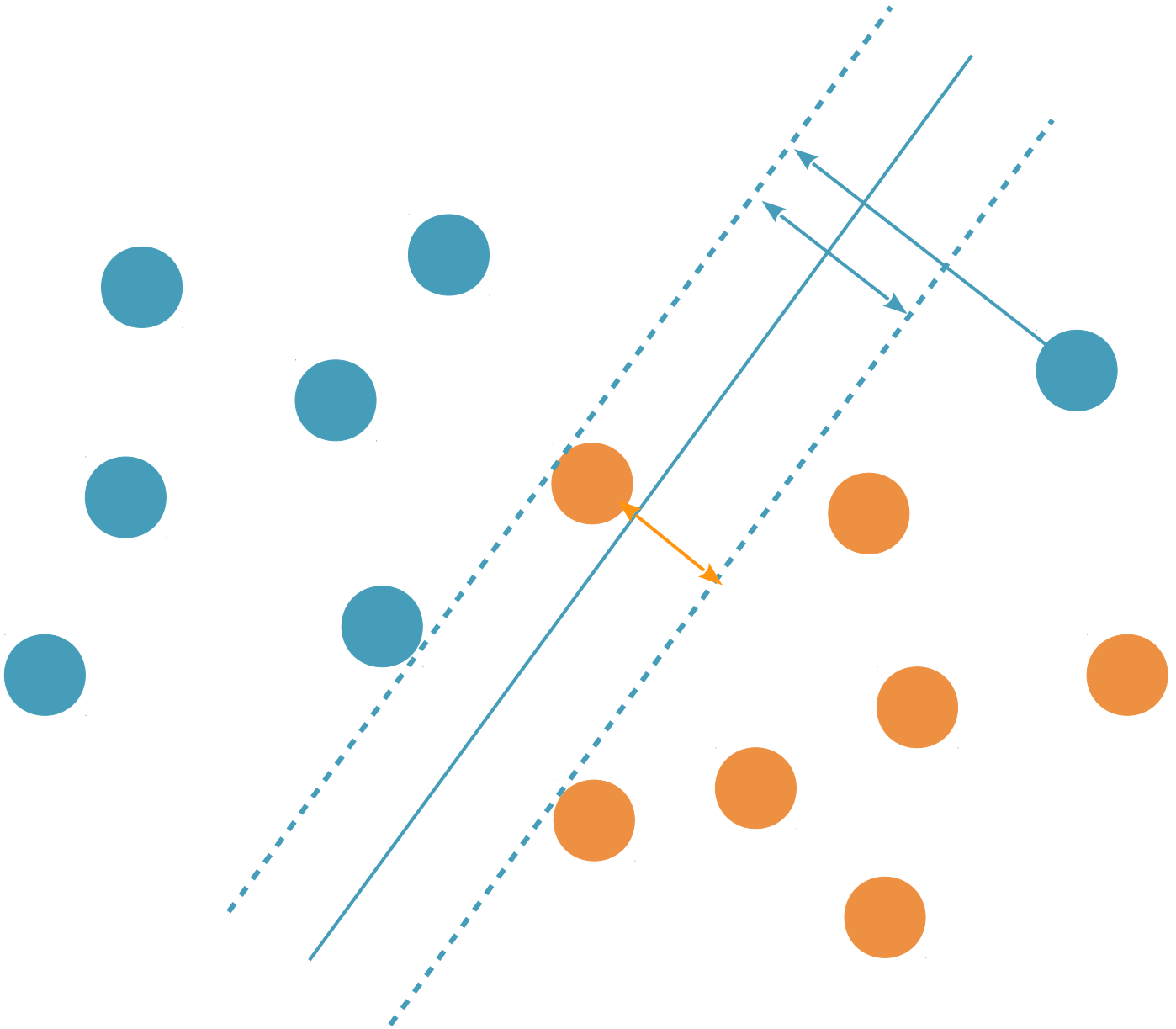
What if the data are not linearly separable?



Soft-margin SVMs



Soft-margin SVMs



Soft-margin SVMs

- Find a trade-off between **large margin** and **few errors.**

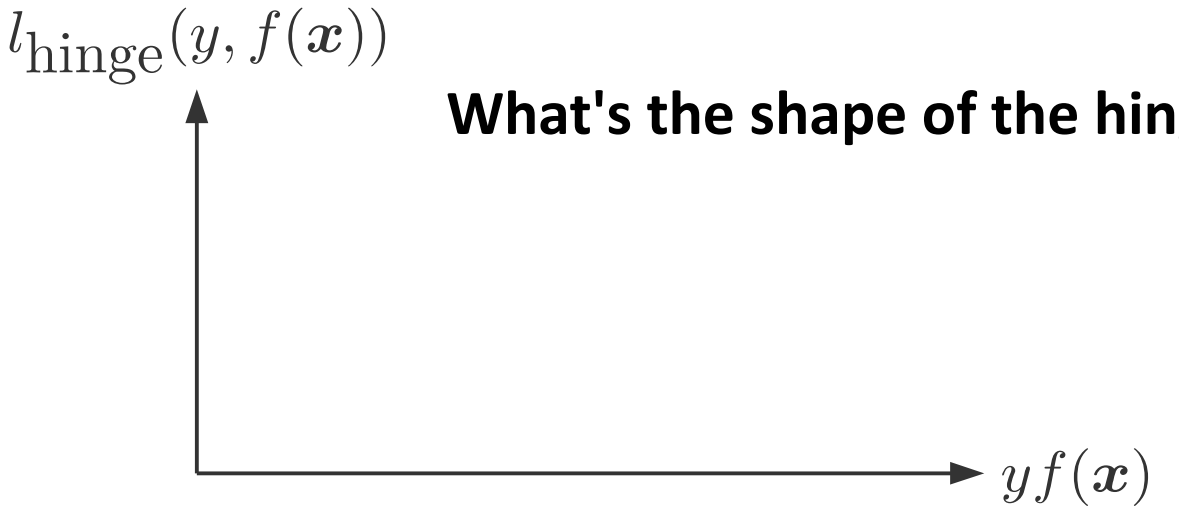
$$\min_f \left(\frac{1}{\text{margin}(f)} + C \times \text{error}(f) \right)$$

What does this remind you of?

SVM error: hinge loss

- We want for all i: $y^i f(\boldsymbol{x}^i) \geq 1$
- **Hinge loss** function:

$$
\begin{aligned}
l_{\text{hinge}}(u, y) &= \max(1 - yu, 0) \\
&= \begin{cases} 0 & \text{if } yu \geq 1 \\ 1 - yu & \text{otherwise} \end{cases}
\end{aligned}
$$

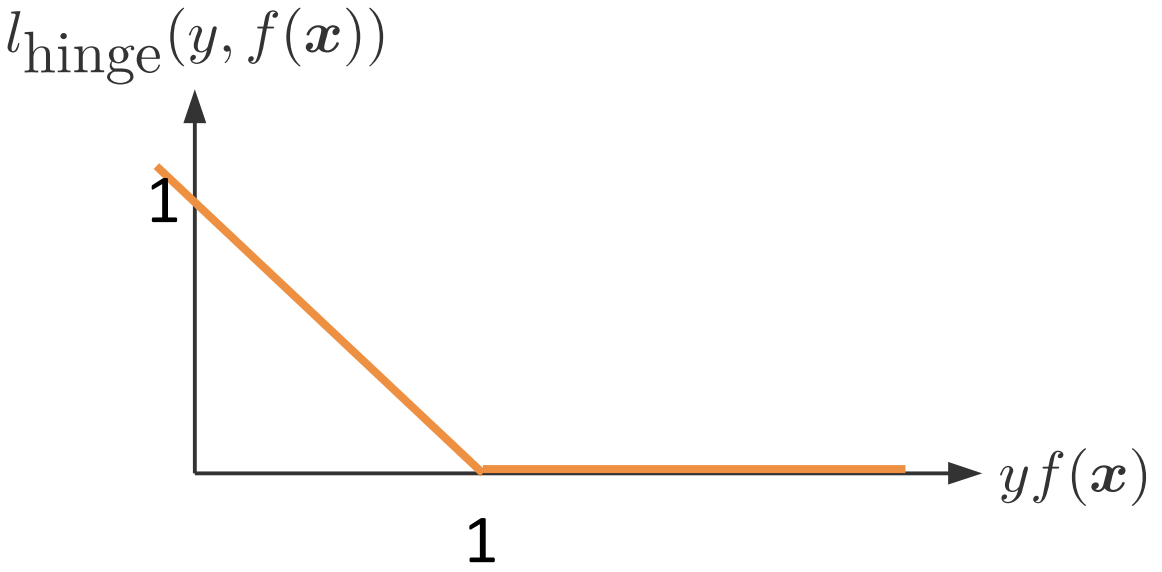


What's the shape of the hinge loss?

SVM error: hinge loss

- We want for all i: $y^i f(\boldsymbol{x}^i) \geq 1$
- **Hinge loss** function:

$$
\begin{aligned}
l_{\text{hinge}}(u, y) &= \max(1 - yu, 0) \\
&= \begin{cases} 0 & \text{if } yu \geq 1 \\ 1 - yu & \text{otherwise} \end{cases}
\end{aligned}
$$



Soft-margin SVMs

- Find a trade-off between **large margin** and **few errors.**

$$\min_f \left(\frac{1}{\text{margin}(f)} + C \times \text{ error}(f) \right)$$

- **Error:**

$$\begin{cases} 0 & \text{if } y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \geq 1 \\ 1 - y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) & \text{otherwise.} \end{cases}$$

- The **soft-margin SVM** solves:

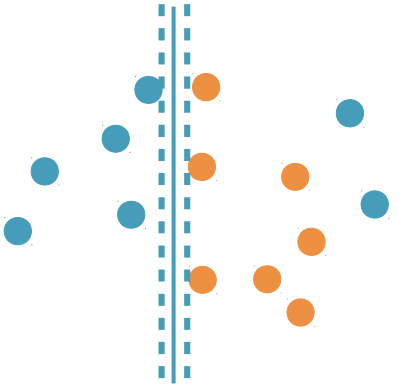
$$\arg\min_{\boldsymbol{w},b} \left(\frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{n} \max(0, 1 - y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b)) \right)$$

The C parameter

$$\min_f \left(\frac{1}{\text{margin}(f)} + C \times \text{error}(f) \right)$$

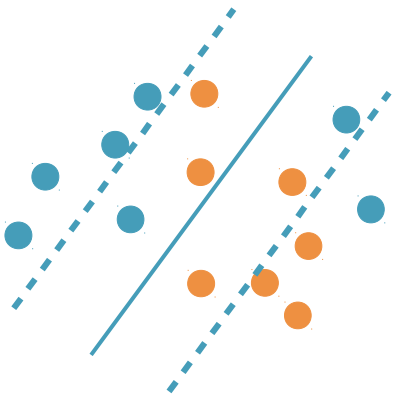
- **Large C**

  makes few errors



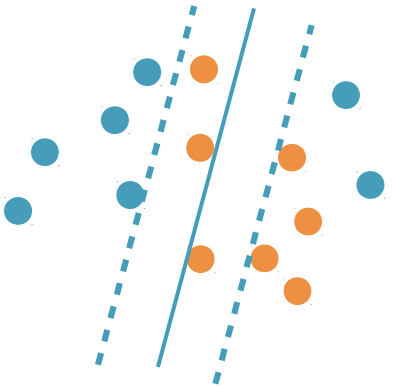
- **Small C**

  ensures a large margin

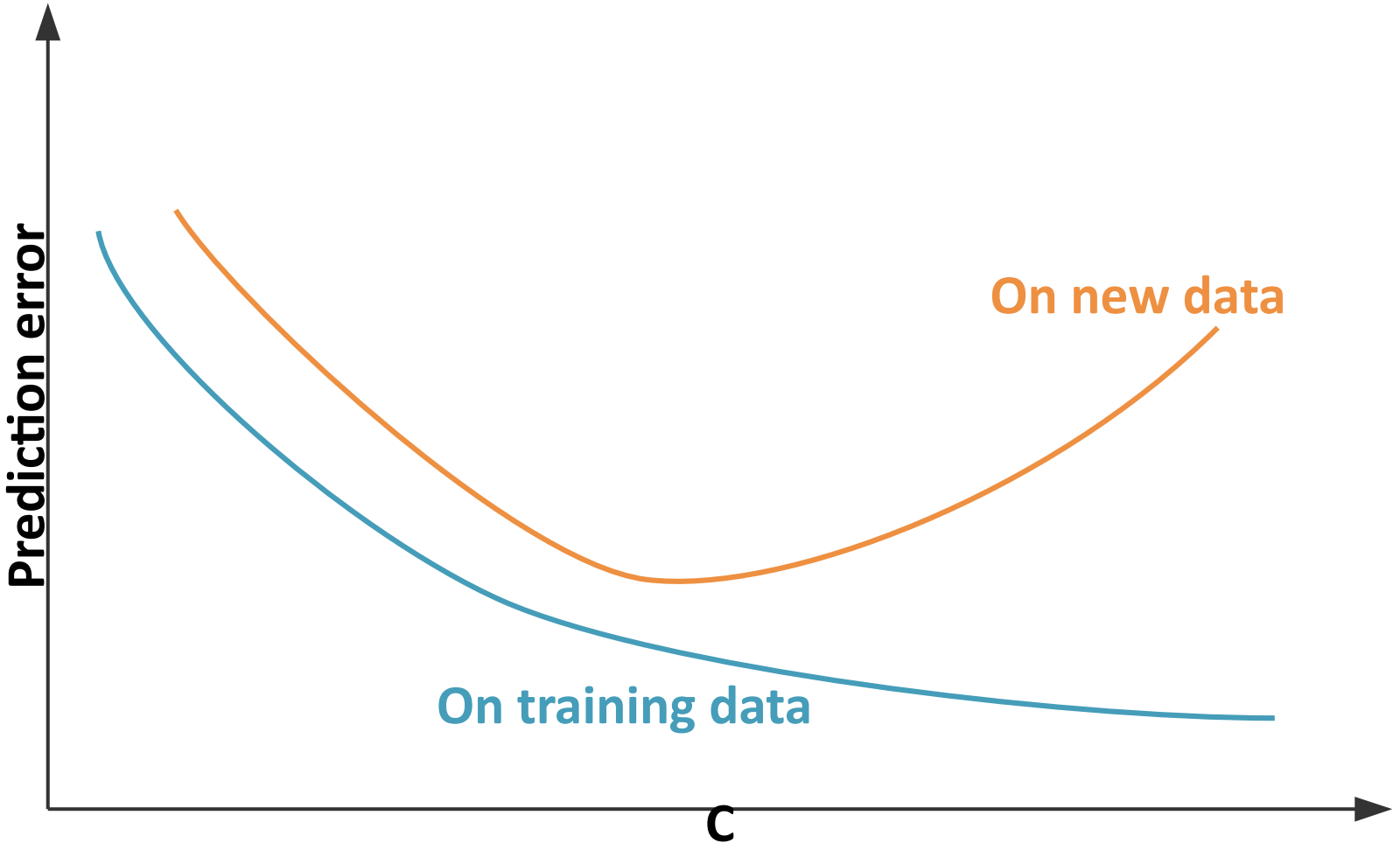


- **Intermediate C**

  finds a tradeoff



It is important to control C



Slack variables

$$\arg\min_{\boldsymbol{w},b}\left(\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\max(0, 1 - y^i(\langle \boldsymbol{w}, x^i\rangle + b))\right)$$

is equivalent to:

$$\begin{aligned} & \arg\min & & \tfrac{1}{2}||\boldsymbol{w}||^2 + C\textstyle\sum_{i=1}^{n}\xi_i \\ \text{s. t. } & y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i\rangle + b) & \geq & \ 1 - \xi_i \\ & \xi_i & \geq & \ 0 \ \ \forall i \end{aligned}$$

slack variable:
distance btw y.f(x) and 1

Lagrangian of the soft-margin SVM

- **Primal**

$$\begin{array}{rcl} \arg\min & & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s. t. } y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i\rangle + b) & \geq & 1-\xi_i \\ \xi_i & \geq & 0 \ \ \forall i \end{array}$$

- **Lagrangian**

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i\rangle + b) - 1 + \xi_i\right) - \sum_i r_i\xi_i$$

- Min the Lagrangian (partial derivatives in **w**, b, **ξ**)

$$\begin{array}{l} \boldsymbol{w} = \sum_i \alpha_i y^i \boldsymbol{x}^i \\ \sum_i \alpha_i y^i = 0 \\ \alpha_i = C - r_i \end{array}$$

- **KKT conditions**

$$\begin{array}{l} \alpha_i\left(y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i\rangle + b) - 1 + \xi_i\right) = 0 \\ r_i\xi_i = 0 \end{array}$$

Dual formulation of the soft-margin SVM

- **Dual:** Maximize

$$q(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j \left\langle \boldsymbol{x}^i, \boldsymbol{x}^j \right\rangle$$

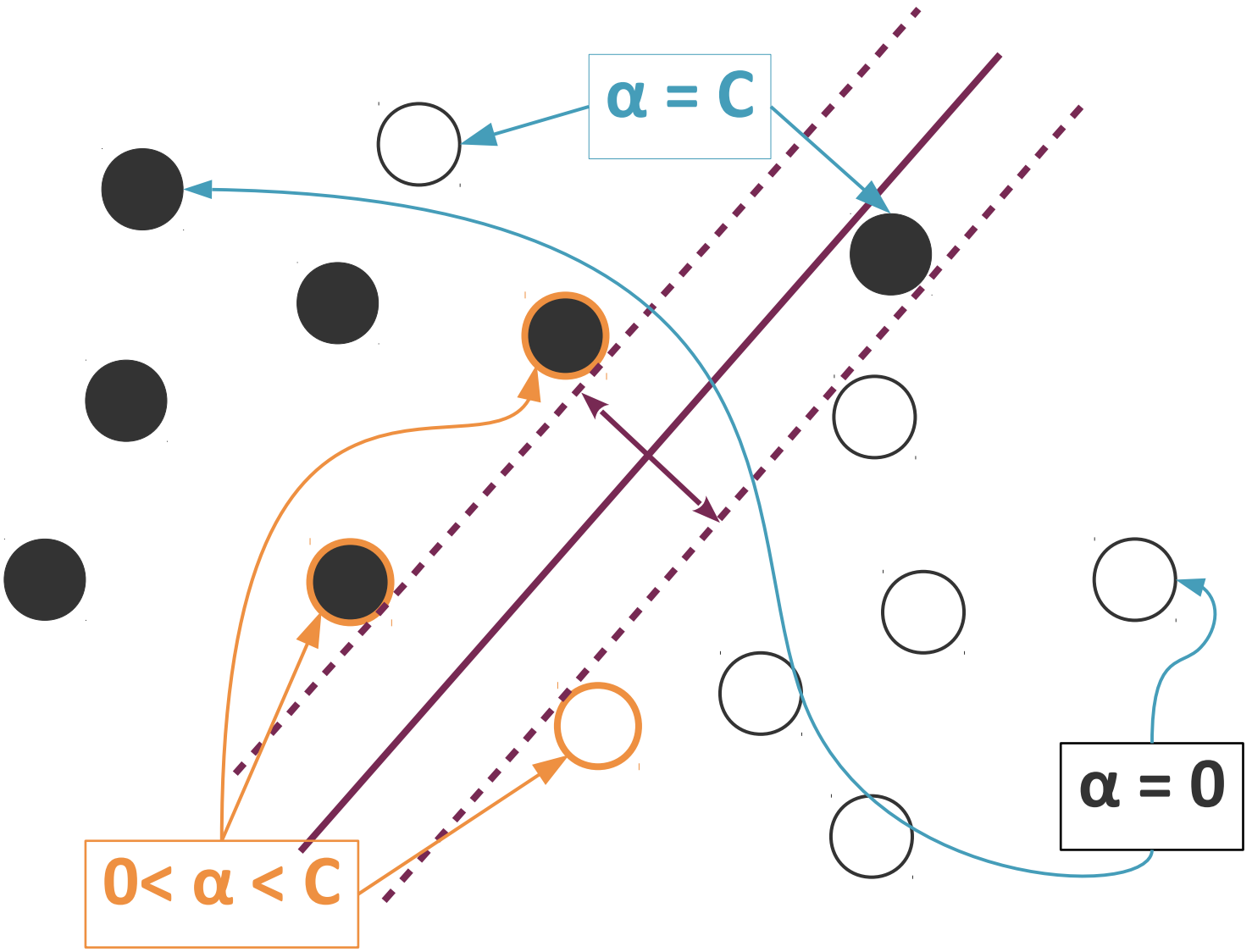
- under the constraints

$$\begin{cases} 0 \leq \alpha_i \leq C & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y^i = 0 \end{cases}$$

- **KKT conditions:**

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) > 1 && \text{“easy”} \\ \alpha_i = C &\Rightarrow y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) < 1 && \text{“hard”} \\ 0 < \alpha_i < C &\Rightarrow y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) = 1 && \text{“somewhat hard”} \end{aligned}$$

Support vectors of the soft-margin SVM



Primal vs. dual

- **What is the dimension of the primal problem?**

$$\arg\min_{\boldsymbol{w},b}\left(\sum_{i=1}^{n} l_{\text{hinge}}(\langle \boldsymbol{w}, x^i\rangle + b, y^i) + \lambda||\boldsymbol{w}||^2\right)$$

- **What is the dimension of the dual problem?**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j\rangle$$

$$0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n}\alpha_i y^i = 0$$

Primal vs. dual

- Primal: (**w**, b) has **dimension (p+1).**

$$\arg\min_{\boldsymbol{w},b}\left(\sum_{i=1}^{n} l_{\text{hinge}}(\langle \boldsymbol{w}, x^i\rangle + b, y^i) + \lambda||\boldsymbol{w}||^2\right)$$

Favored if the data is **low-dimensional.**

- Dual: **α** has **dimension n.**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j\rangle$$

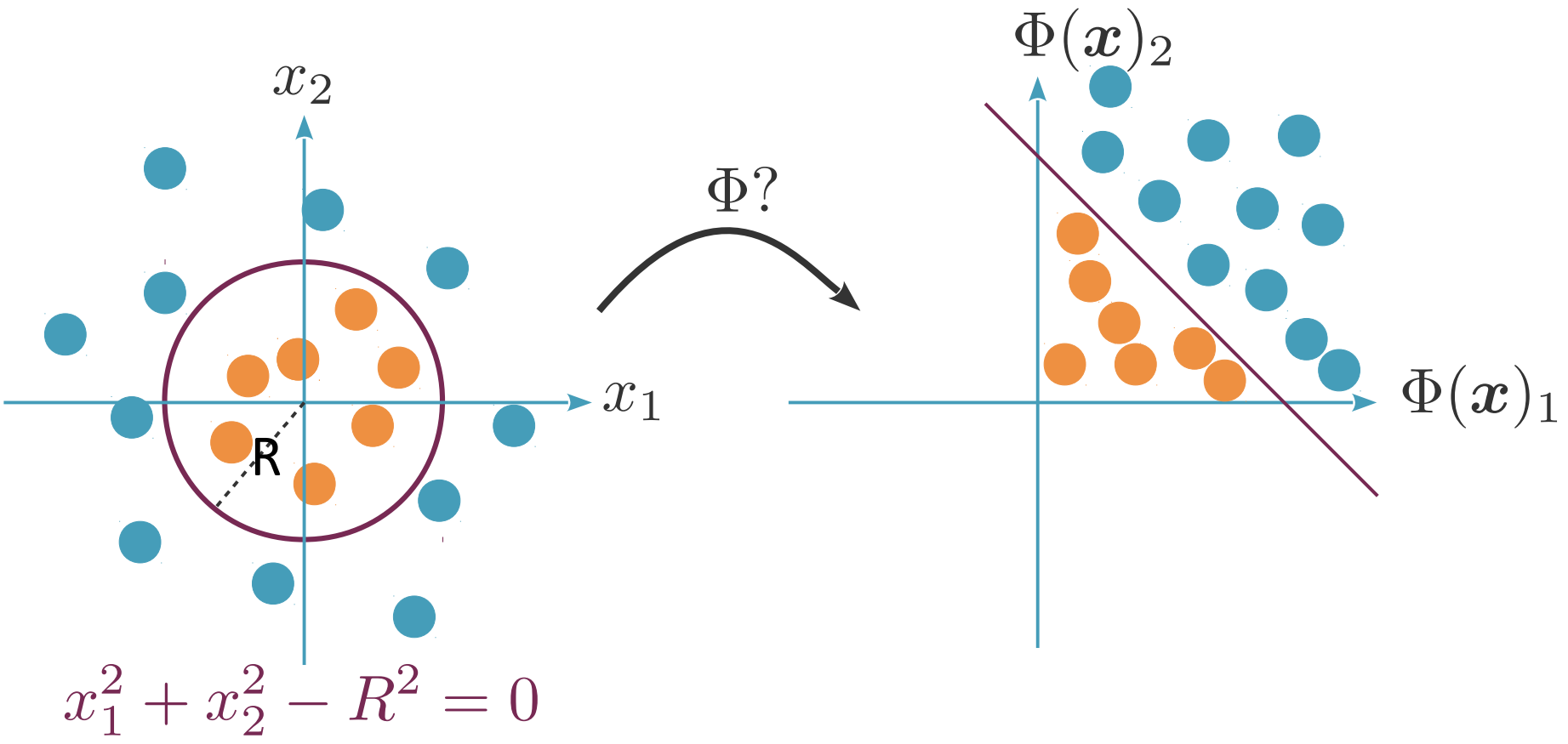
$$0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n}\alpha_i y^i = 0$$

Favored is there is **little data** available.

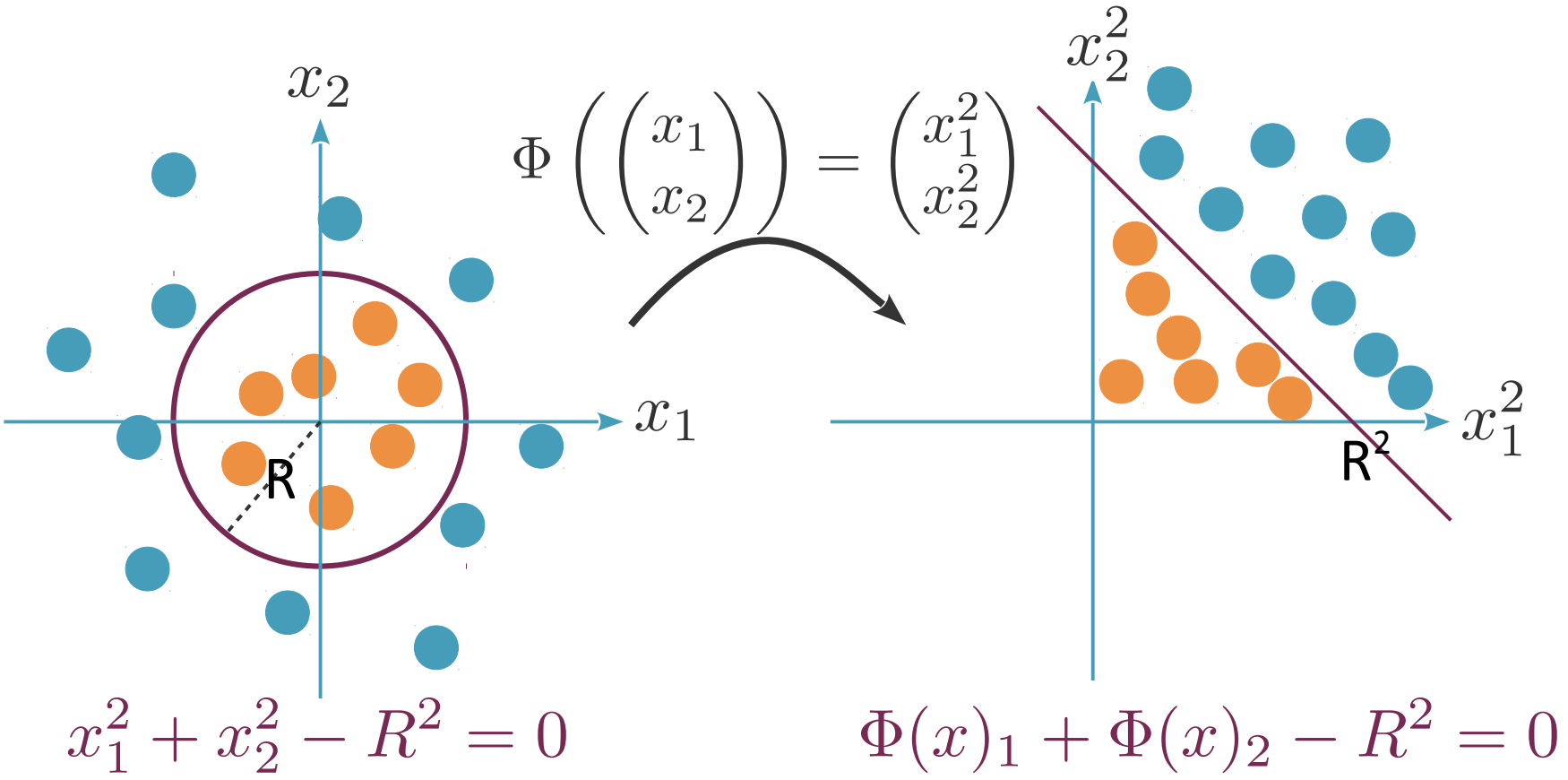
The non-linear case: kernel SVMs.

Non-linear SVMs

Non-linear mapping to a feature space



Non-linear mapping to a feature space



SVM in the feature space

- **Train:**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j \langle \Phi(\boldsymbol{x}^i), \Phi(\boldsymbol{x}^j) \rangle_{\mathcal{H}}$$

under the constraints

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n} \alpha_i y^i = 0$$

- **Predict** with the decision function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y^i \langle \Phi(\boldsymbol{x}^i), \Phi(\boldsymbol{x}) \rangle_{\mathcal{H}} + b^*$$

Kernels

For a given mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H}$$

from the space of objects X to some Hilbert space H, the **kernel** between two objects x and x' is the inner product of their images in the feature spaces.

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

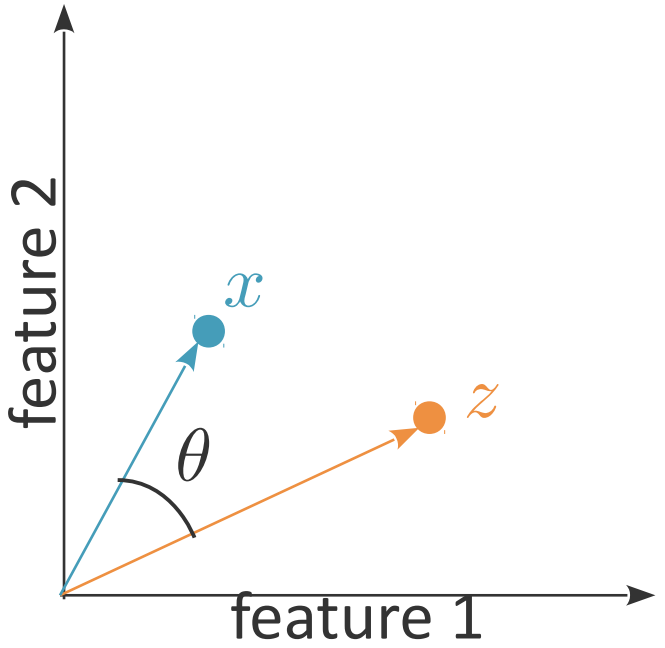
$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

- E.g. $$K(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}, \begin{pmatrix} {x'_1}^2 \\ {x'_2}^2 \end{pmatrix} \right\rangle = x_1^2 {x'_1}^2 + x_2^2 {x'_2}^2$$
- **Kernels allow us to formalize the notion of similarity.**

Dot product and similarity

- **Normalized dot product = cosine**

$$\rho(\boldsymbol{x}, \boldsymbol{z}) = \frac{\sum_{j=1}^{p} x_j z_j}{\sqrt{\sum_{j=1}^{p} x_j{}^2}\sqrt{\sum_{j=1}^{p} z_j{}^2}} = \frac{\langle \boldsymbol{x}, \boldsymbol{z} \rangle}{||\boldsymbol{x}||.||\boldsymbol{z}||} = \cos\theta$$



Kernel trick

- Many linear algorithms (in particular, linear SVMs) can be performed in the feature space H **without explicitly computing the images ϕ(x)**, but instead by computing kernels **K(x, x')**

- It is sometimes easy to compute kernels which correspond to large-dimensional feature spaces: **K(x, x') is often much simpler to compute than ϕ(x).**

SVM in the feature space

- **Train:**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j \langle \Phi(\boldsymbol{x}^i), \Phi(\boldsymbol{x}^j) \rangle_{\mathcal{H}}$$

under the constraints

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n} \alpha_i y^i = 0$$

- **Predict** with the decision function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y^i \langle \Phi(\boldsymbol{x}^i), \Phi(\boldsymbol{x}) \rangle_{\mathcal{H}} + b^*$$

SVM with a kernel

- **Train:**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j K(\boldsymbol{x}^i, \boldsymbol{x}^j)$$

under the constraints

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n} \alpha_i y^i = 0$$

- **Predict** with the decision function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y^i K(\boldsymbol{x}^i, \boldsymbol{x}) + b^*$$

Which functions are kernels?

- A function K(**x**, **x'**) defined on a set X is a **kernel** iff it exists a Hilbert space H and a mapping ϕ: X $\rightarrow$H such that, for any **x**, **x'** in X:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

- A function K(**x**, **x'**) defined on a set X is **positive definite** iff it is **symmetric** and satisfies:

$$\forall N \in \mathbb{N}, \forall (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N) \in \mathcal{X}^N \text{ and } (a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K(\boldsymbol{x}^i, \boldsymbol{x}^j) \geq 0.$$

- Theorem [Aronszajn, 1950]: **K is a kernel iff it is positive definite.**

Positive definite matrices

- Have a **unique Cholesky decomposition** $K = LL^\top$

  L: lower triangular, with positive elements on the diagonal

- **Sesquilinear form is an inner product**

$$\boldsymbol{x}, \boldsymbol{x}' \mapsto \boldsymbol{x}^\top K \boldsymbol{x}'$$

  - **conjugate symmetry** $\boldsymbol{x}^\top K \boldsymbol{x}' = (\boldsymbol{x}'^\top K \boldsymbol{x})^*$
  - **linearity in the first argument**

$$a(\boldsymbol{x}^\top K \boldsymbol{x}') = (a\boldsymbol{x})^\top K \boldsymbol{x}'$$
$$(\boldsymbol{x} + \boldsymbol{z})^\top K \boldsymbol{x}' = \boldsymbol{x}^\top K \boldsymbol{x}' + \boldsymbol{z}^\top K \boldsymbol{x}'$$

  - **positive definiteness**

$$\boldsymbol{x}^\top K \boldsymbol{x}' \geq 0$$
$$\boldsymbol{x}^\top K \boldsymbol{x} = 0 \Rightarrow \boldsymbol{x} = 0$$

Polynomial kernels

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \qquad \Phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \in \mathbb{R}^3$$

Compute $K(\boldsymbol{x}, \boldsymbol{x}')$ 

Polynomial kernels

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \qquad \Phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \in \mathbb{R}^3$$

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{x}') &= x_1^2 {x_1'}^2 + 2x_1x_2x_1'x_2' + x_2^2 {x_2'}^2 \\ &= \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2 \end{aligned}$$

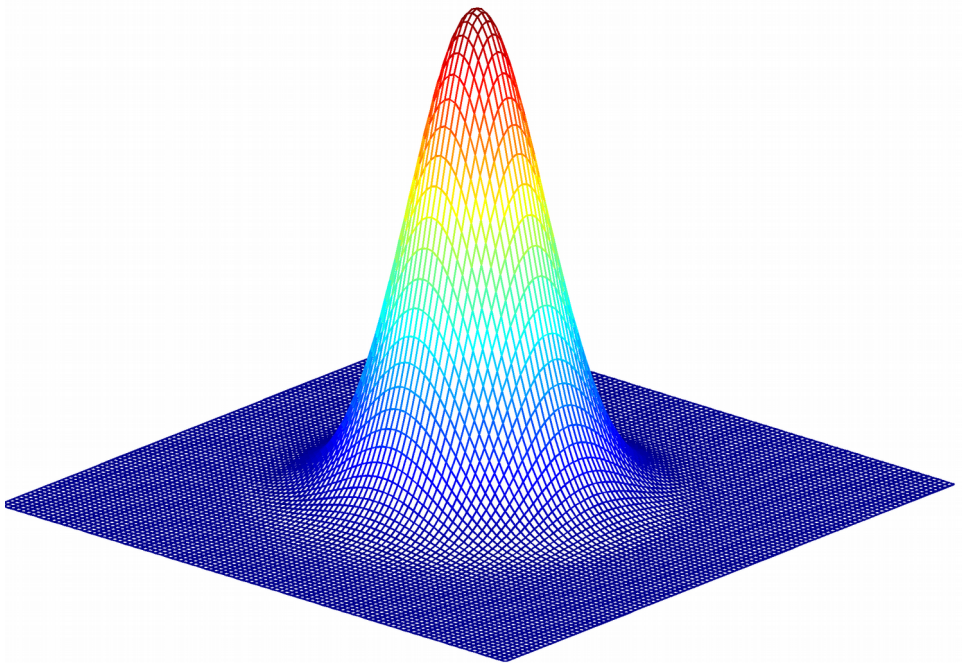
More generally, for $\mathcal{X} = \mathbb{R}^p$

$$K(x, x') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$$

is an inner product in a feature space of all monomials of degree up to d.

Gaussian kernel

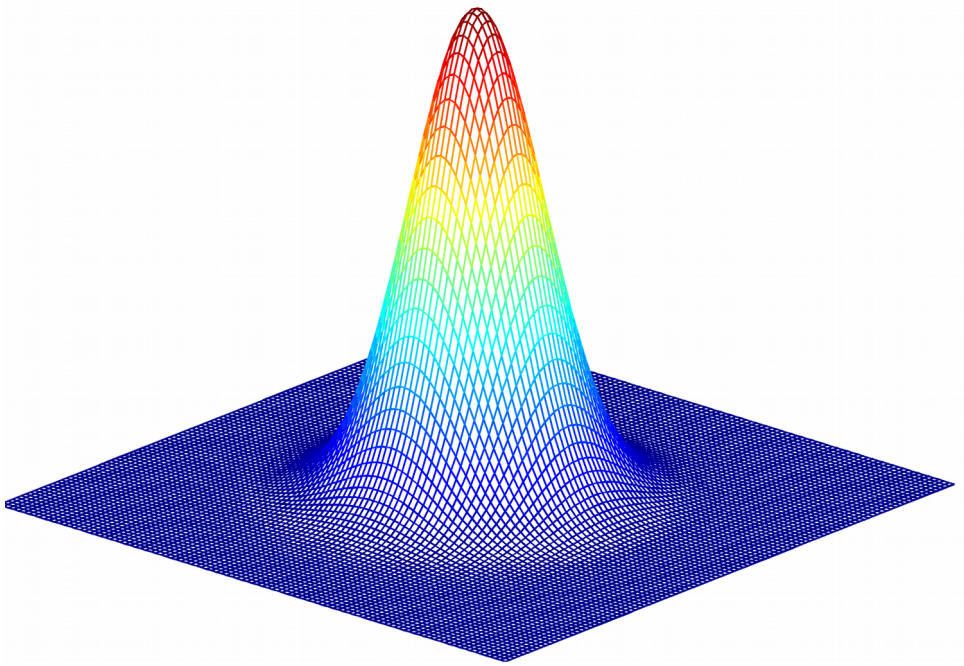
$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}\right)$$



What is the dimension of the feature space?

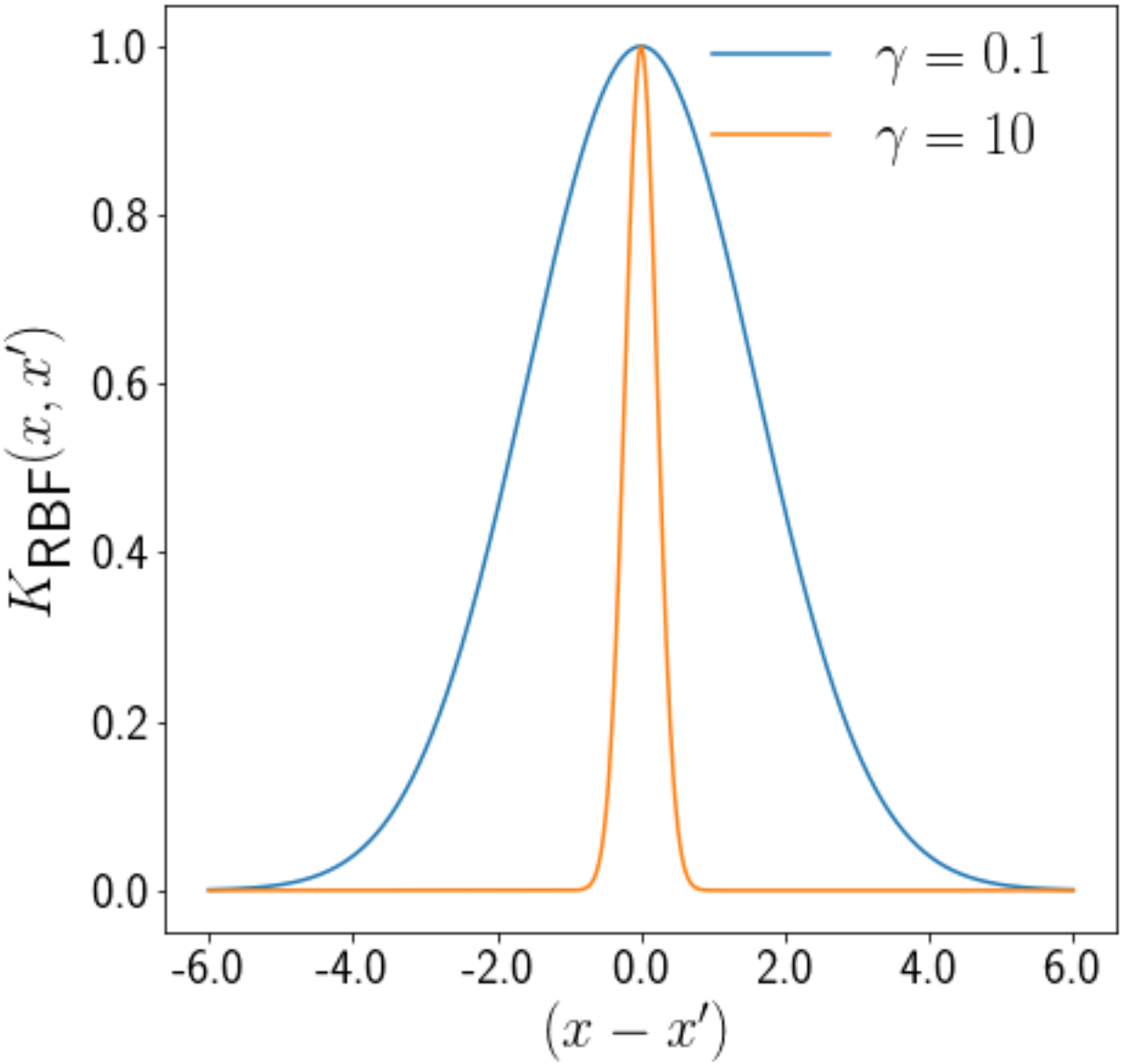
Gaussian kernel

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

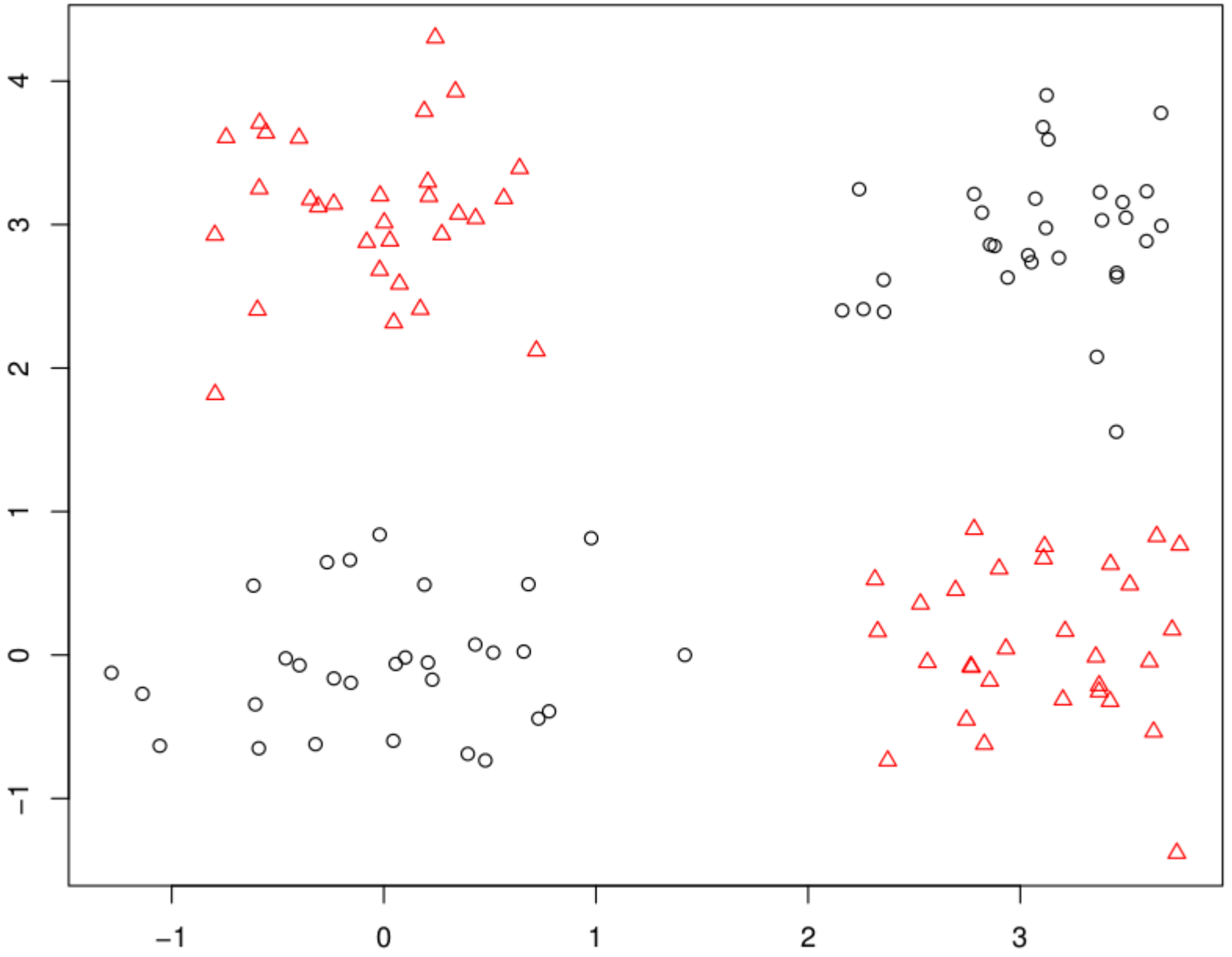


The feature space has **infinite dimension.**

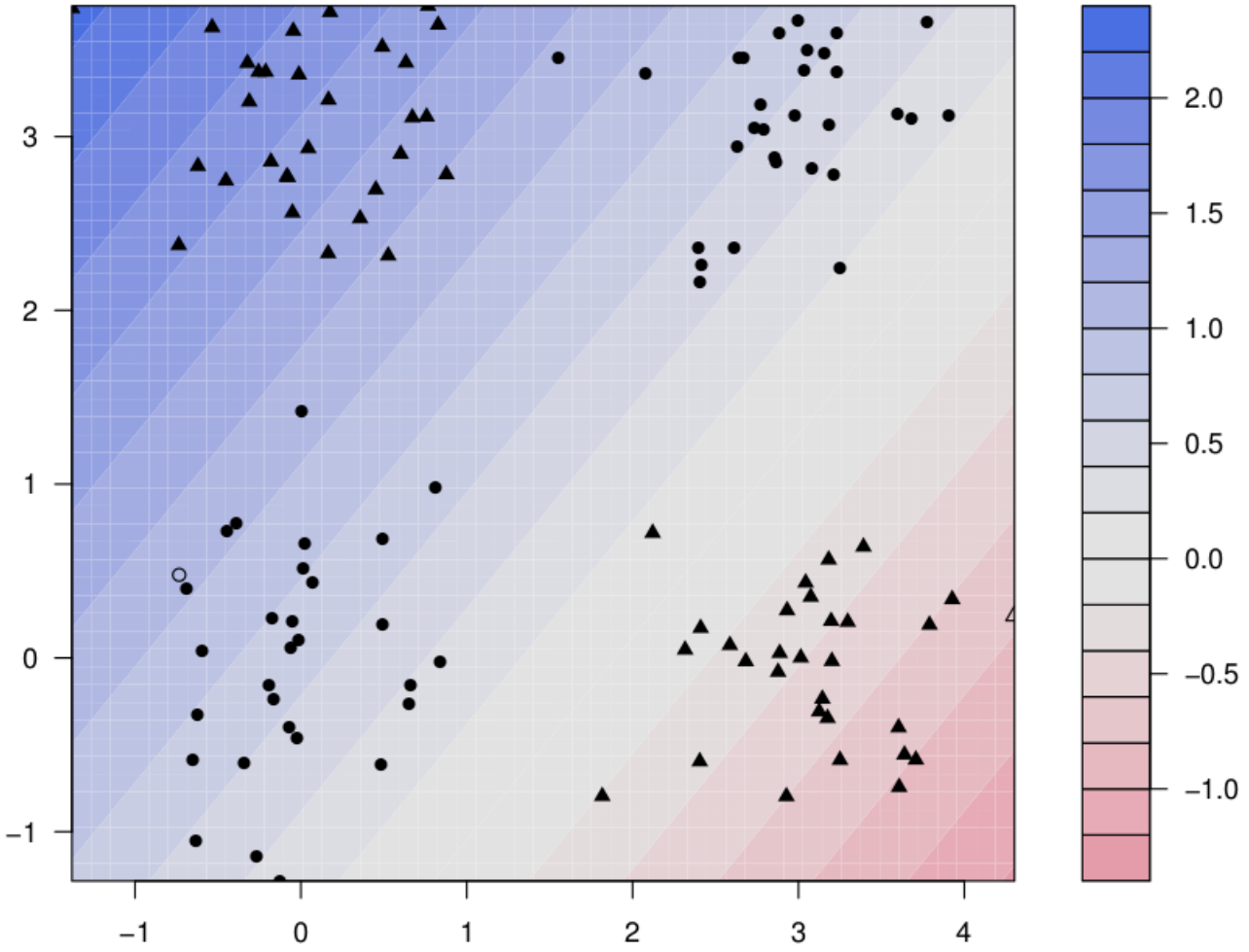
$$\begin{aligned} K(x, x') &= \exp\left(\tfrac{-1}{2\sigma^2}||x||^2\right) \exp\left(\tfrac{1}{\sigma^2}\langle x, x'\rangle\right) \exp\left(\tfrac{-1}{2\sigma^2}||x'||^2\right) \\ &= f(x) \sum_{r=0}^{+\infty} \frac{\langle x, x'\rangle^r}{\sigma^{2r} r!} f(x') \end{aligned}$$


1.0
0.8
0.6
0.4
0.2
0.0
$K_{\text{RBF}}(x, x')$
-6.0
-4.0
-2.0
0.0
2.0
4.0
6.0
$(x - x')$
$\gamma = 0.1$
$\gamma = 10$

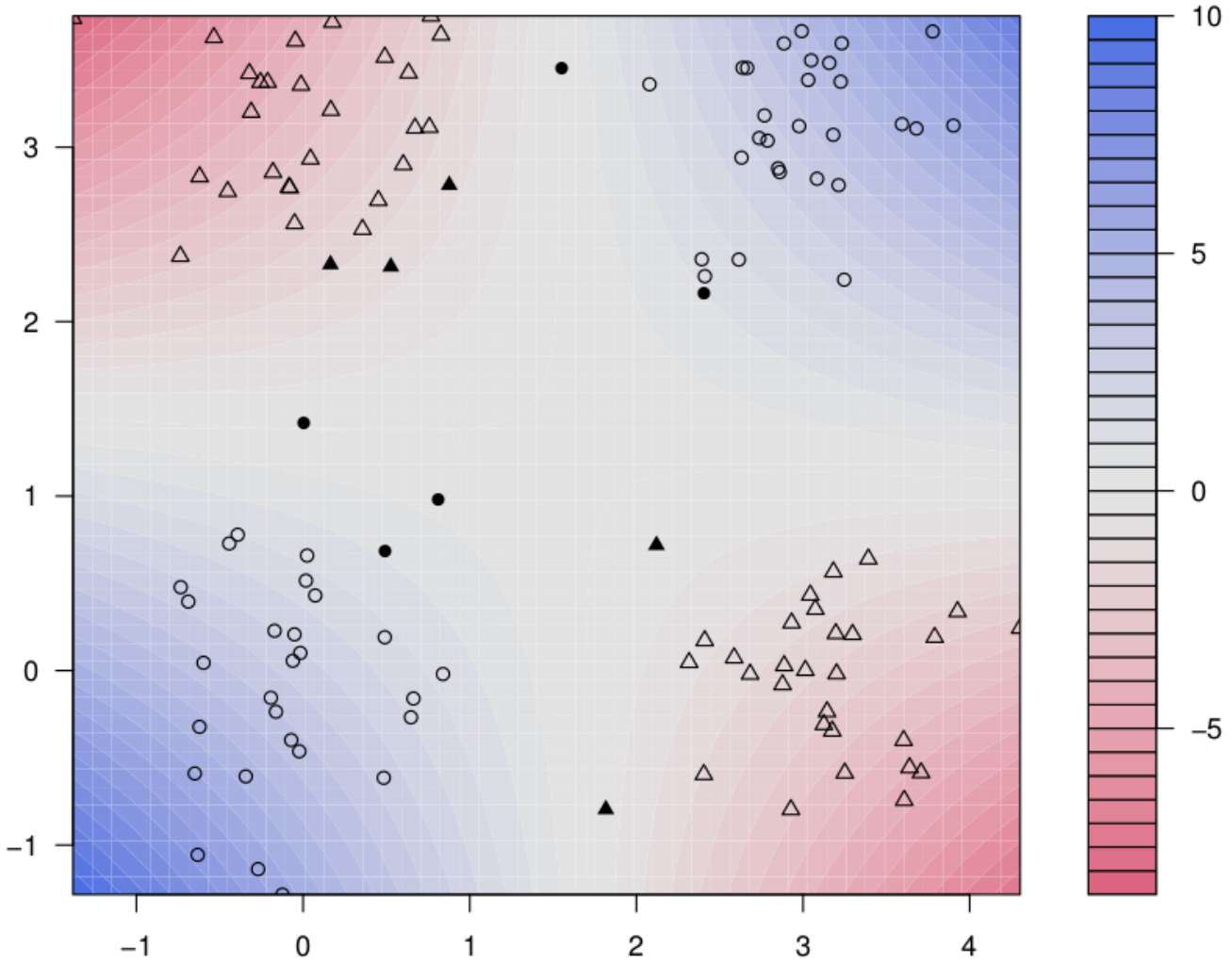
Toy example



Toy example: linear SVM



Toy example: polynomial SVM (d=2)



Kernels for strings

Protein sequence classification

Goal: predict which proteins are secreted or not, based on their sequence.

- Secreted proteins:

  MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...
  MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
  MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...
  ...

- Non-secreted proteins:

  MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...
  MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
  MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..
  ...

Substring-based representations

- Represent strings based on the presence/absence of substrings of fixed length.

$$\Phi(\boldsymbol{x}) = \{\Phi_u(\boldsymbol{x})\}_{u \in \mathcal{A}^k}$$

?

Strings of length k

Substring-based representations

- Represent strings based on the presence/absence of substrings of fixed length.

$$\Phi(\boldsymbol{x}) = \{\Phi_u(\boldsymbol{x})\}_{u \in \mathcal{A}^k}$$

  - Number of occurrences of u in **x**: **spectrum kernel** [Leslie et al., 2002].

Substring-based representations

- Represent strings based on the presence/absence of substrings of fixed length.

$$\Phi(\boldsymbol{x}) = \{\Phi_u(\boldsymbol{x})\}_{u \in \mathcal{A}^k}$$

  - Number of occurrences of u in **x**: **spectrum kernel** [Leslie et al., 2002].
  - Number of occurrences of u in x, up to m mismatches: **mismatch kernel** [Leslie et al., 2004].

Substring-based representations

- Represent strings based on the presence/absence of substrings of fixed length.

$$\Phi(\boldsymbol{x}) = \{\Phi_u(\boldsymbol{x})\}_{u \in \mathcal{A}^k}$$

  - Number of occurrences of u in **x**: **spectrum kernel** [Leslie et al., 2002].
  - Number of occurrences of u in x, up to m mismatches: **mismatch kernel** [Leslie et al., 2004].
  - Number of occcurrences of u in x, allowing gaps, with a weight decaying exponentially with the number of gaps: **substring kernel** [Lohdi et al., 2002].

Spectrum kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{u \in \mathcal{A}^k} \Phi_u(\boldsymbol{x}) \Phi_u(\boldsymbol{x}')$$

- **Implementation:**
  - Formally, a sum over $|A^k|$ terms
  - **How many non-zero terms in $\Phi(x)$?**

Spectrum kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{u \in \mathcal{A}^k} \Phi_u(\boldsymbol{x}) \Phi_u(\boldsymbol{x}')$$

- **Implementation:**
  - Formally, a sum over $|A^k|$ terms
  - At most $|\mathbf{x}|$ - k + 1 non-zero terms in $\Phi(\boldsymbol{x})$
  - Hence: Computation in $O(|\mathbf{x}|+|\mathbf{x}'|)$
- **Prediction** for a new sequence **x:**

$$f(\boldsymbol{x}) \quad = \quad \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b$$

Write f(x) as a function of only |x|-k+1 weights.

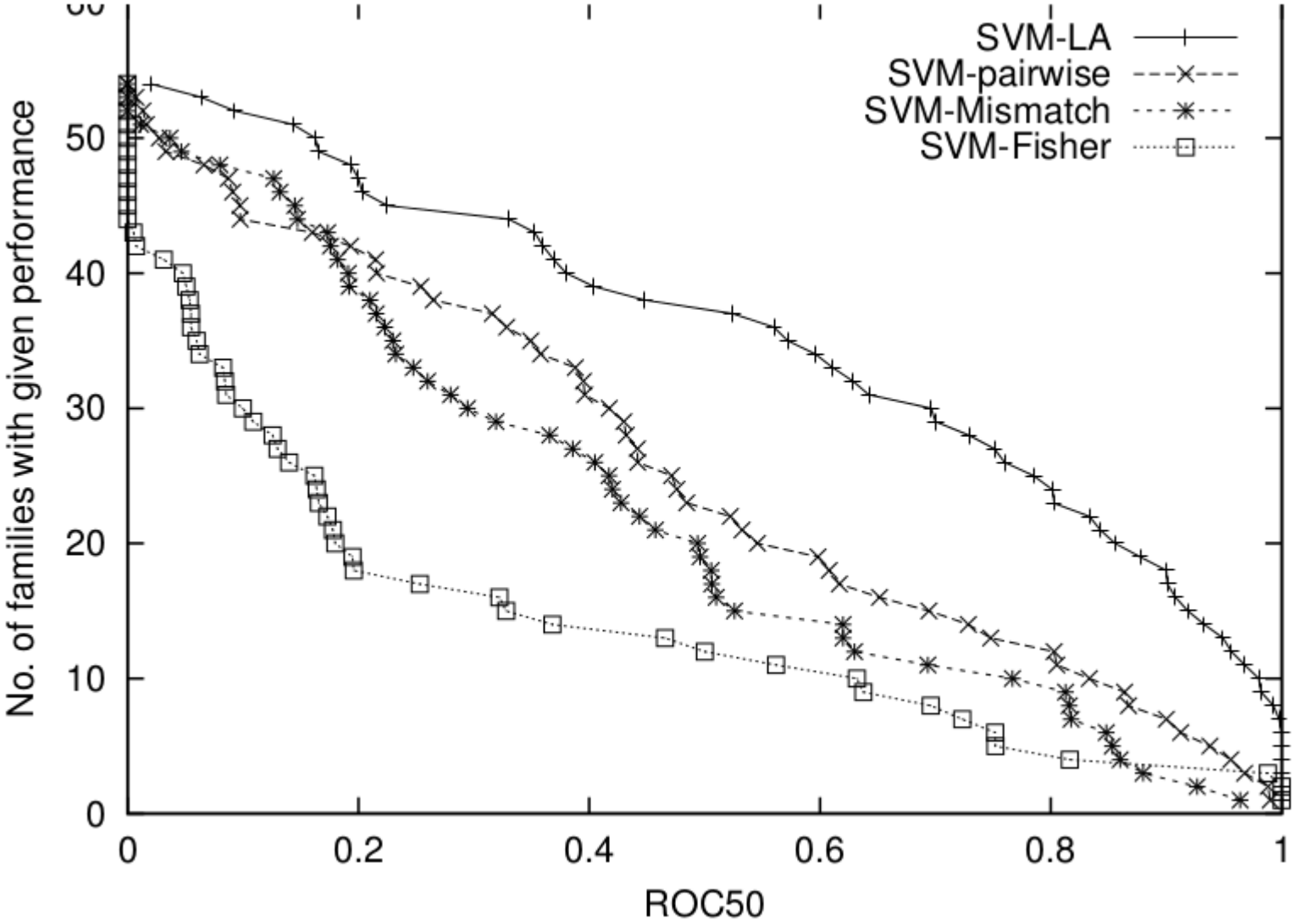
Spectrum kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{u \in \mathcal{A}^k} \Phi_u(\boldsymbol{x}) \Phi_u(\boldsymbol{x}')$$

- **Implementation:**
  - Formally, a sum over $|A^k|$ terms
  - At most $|\mathbf{x}|$ - k + 1 non-zero terms in $\Phi(\boldsymbol{x})$
  - Hence: Computation in $O(|\mathbf{x}|+|\mathbf{x}'|)$
- **Fast prediction** for a new sequence **x:**

$$\begin{aligned} f(\boldsymbol{x}) &= \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b \\ &= \sum_{u \in \mathcal{A}^k} w_u \Phi_u(\boldsymbol{x}) + b \\ &= \sum_{j=1}^{|\boldsymbol{x}|-k+1} w_{x_j x_{j+1} \ldots x_{j+k-1}} + b \end{aligned}$$

The choice of kernel matters

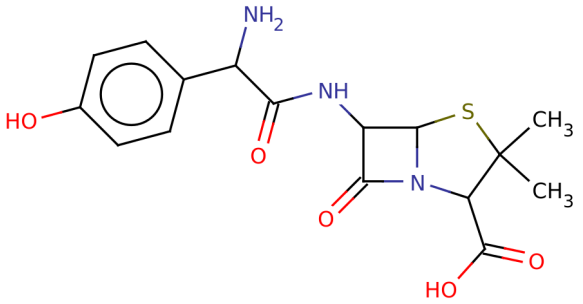


Performance of several kernels on the SCOP superfamily recognition kernel [Saigo et al., 2004]

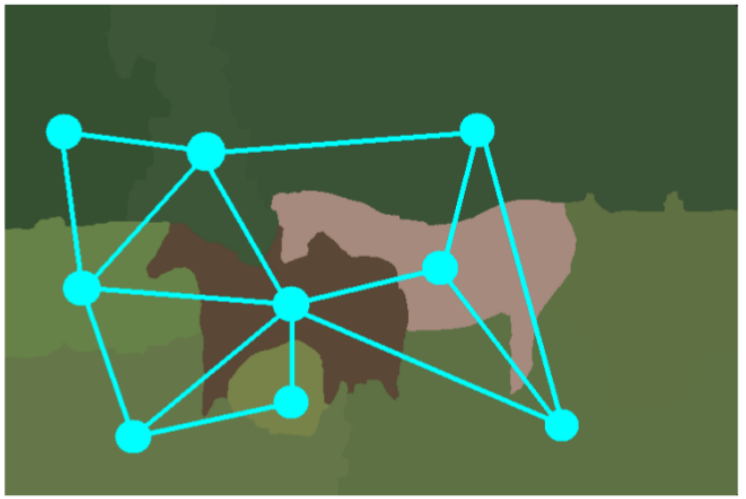
Kernels for graphs

Graph data

- **Molecules**



- **Images**



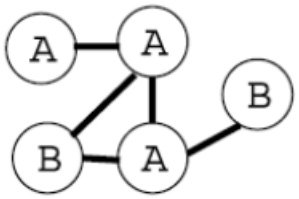
[Harchaoui & Bach, 2007]

Subgraph-based representations

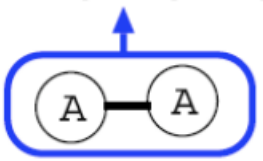
0 1 1 0 0 1 0 0 0 1 0 1 0 0 1

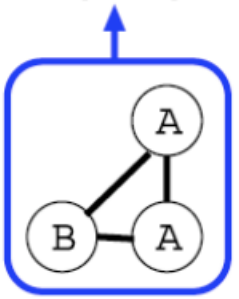
no occurrence of the 1st feature

1+ occurrences of the 10th feature



(0,...,0,1,0,...,0,1,0,...)





Tanimoto & MinMax

- The Tanimoto and MinMax similarities are kernels

$$s(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{\sum_{j=1}^{p}(x_j^1 \text{ AND } x_j^2)}{\sum_{j=1}^{p}(x_j^1 \text{ OR } x_j^2)}$$

$$s(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{\sum_{j=1}^{p} \min(x_j^1, x_j^2)}{\sum_{j=1}^{p} \max(x_j^1, x_j^2)}$$

Which subgraphs to use?

- **Indexing by all subgraphs...**
  - Computing all subgraph occurences is NP-hard.
  - Actually, finding whether a given subgraph occurs in a graph is NP-hard in general.

A Quasipolynomial Time Algorithm for Graph Isomorphism: The Details

Posted on November 12, 2015 by j2kun

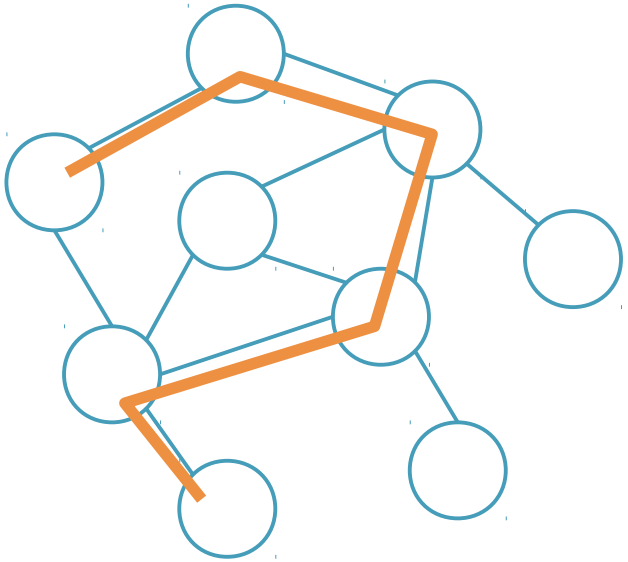
Update 2015-11-16: Laci has posted the talk on his website. It's an hour and a half long, and I encourage you to watch it if you have the time

http://jeremykun.com/2015/11/12/a-quasipolynomial-time-algorithm-for-graph-isomorphism-the-details/

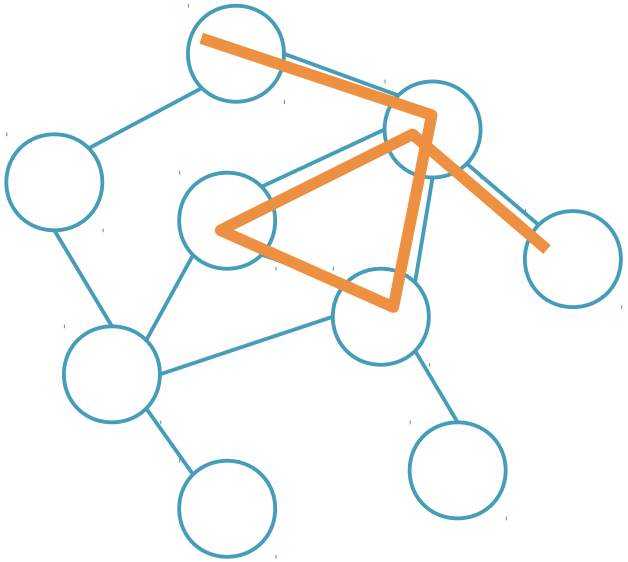
Which subgraphs to use?

- **Specific subgraphs** that lead to computationally efficient indexing:
  - Subgraphs selected based on **domain knowledge**
    E.g. chemical fingerprints
  - All **frequent subgraphs** [Helma et al., 2004]
  - All **paths** up to length k [Nicholls 2005]
  - All **walks** up to length k [Mahé et al., 2005]
  - All **trees** up to depth k [Rogers, 2004]
  - All **shortest paths** [Borgwardt & Kriegel, 2005]
  - All **subgraphs up to k vertices (graphlets)** [Shervashidze et al., 2009]

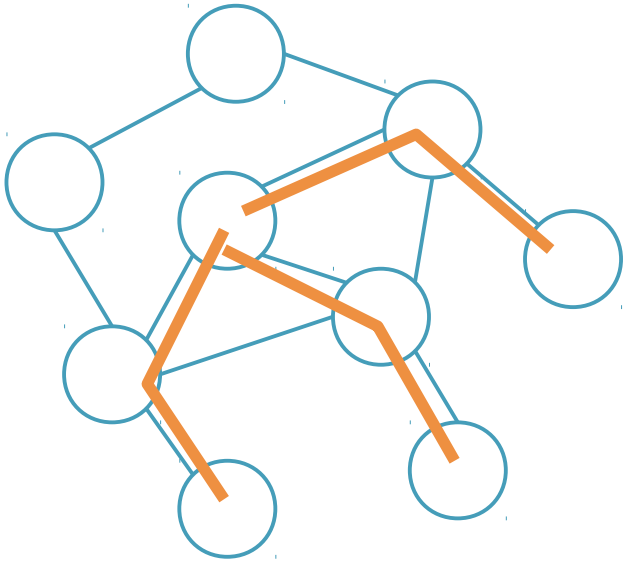
Which subgraphs to use?



Path of length 5

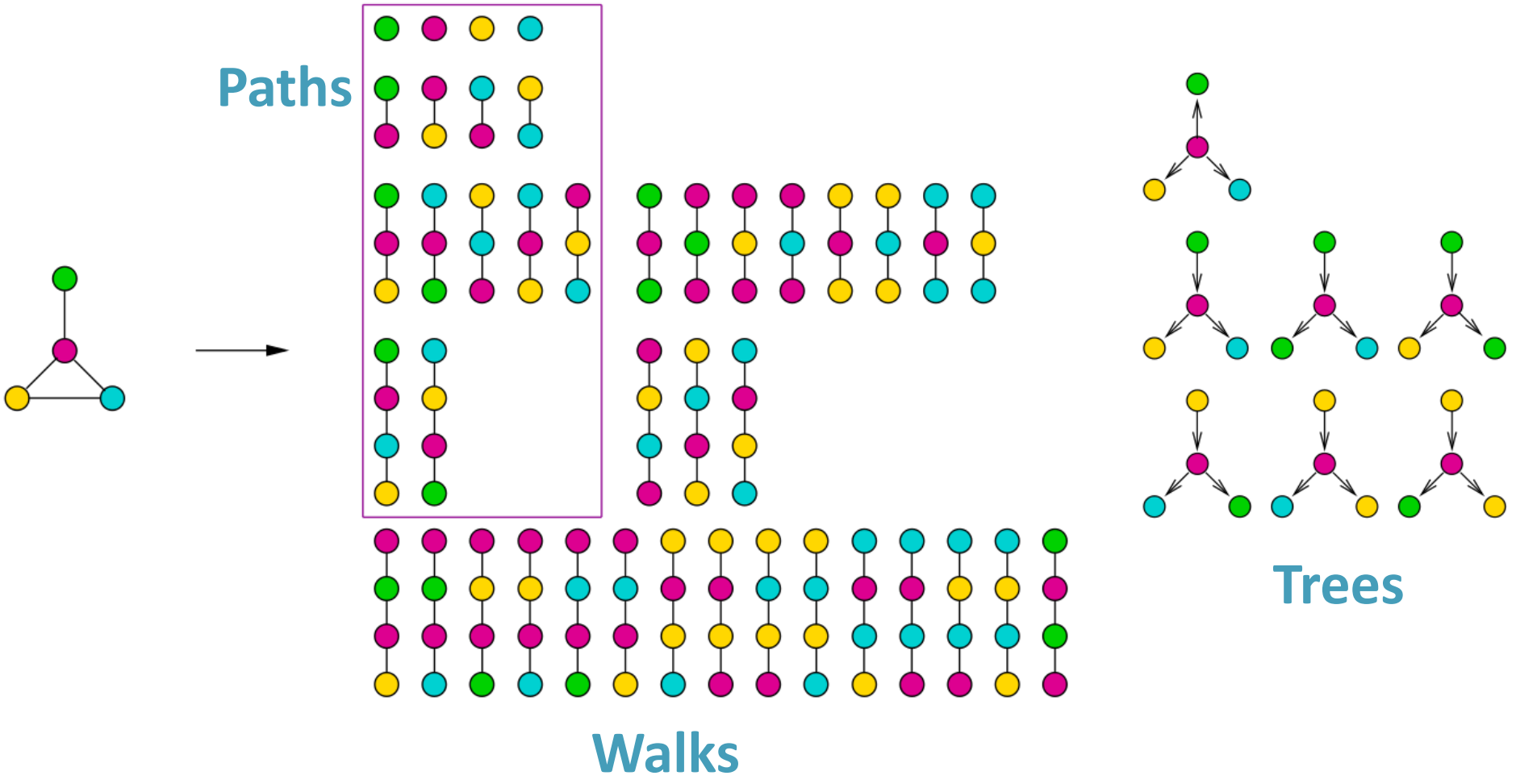


Walk of length 5



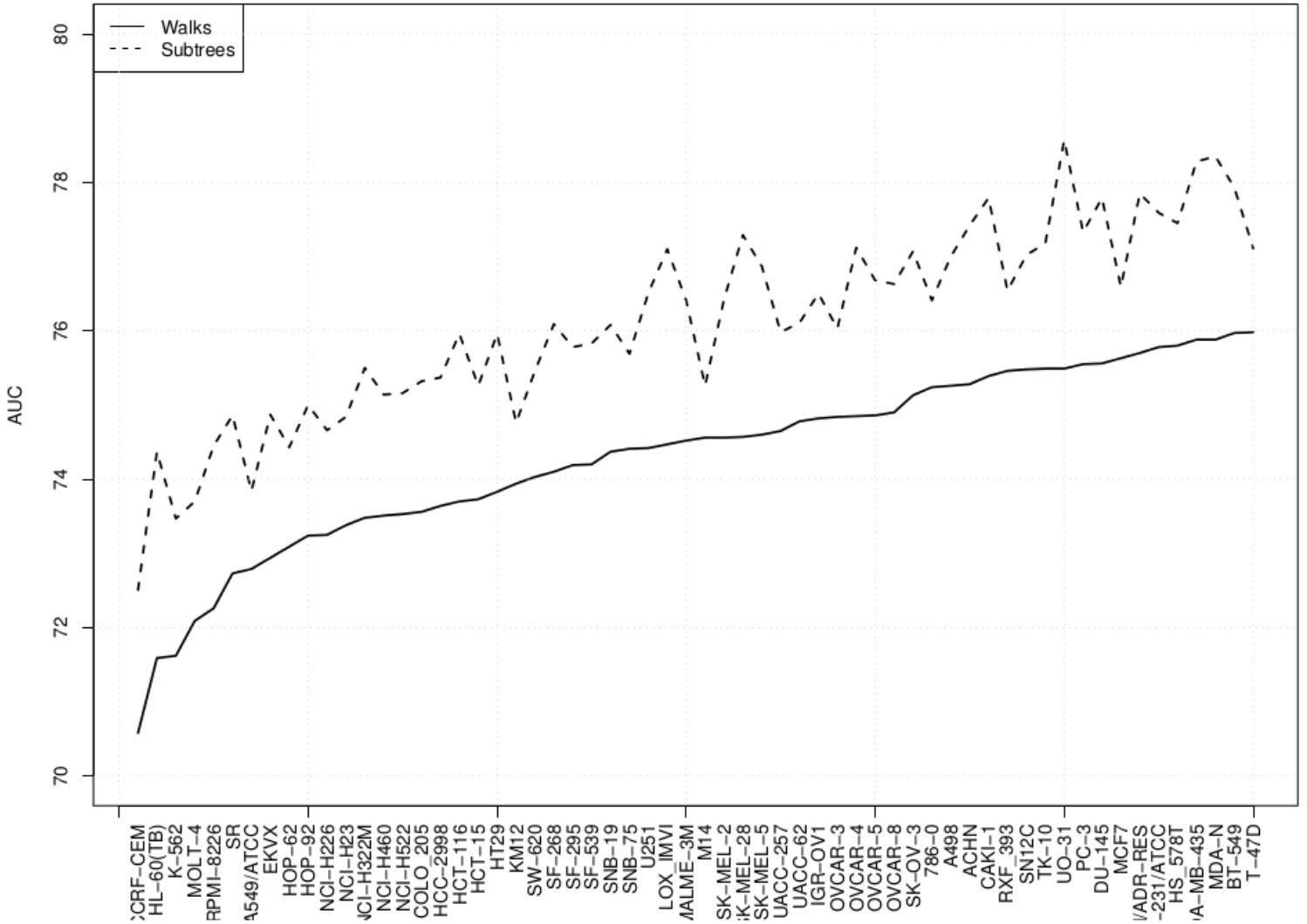
Tree of depth 2

Which subgraphs to use?



[Harchaoui & Bach, 2007]

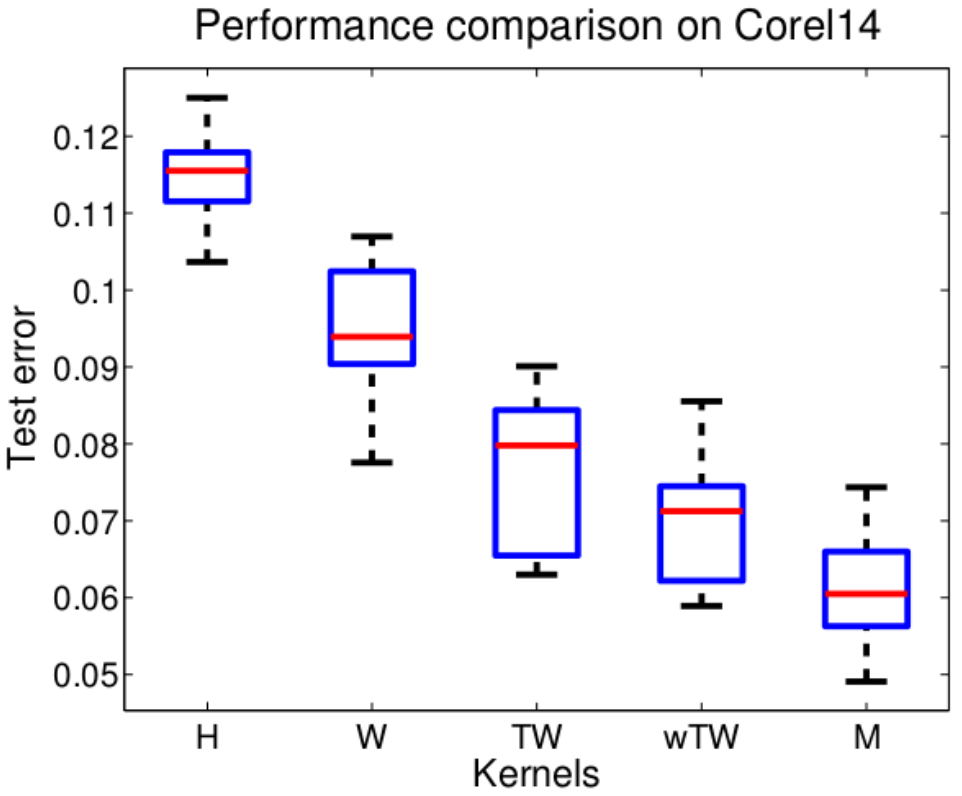
The choice of kernel matters



Predicting inhibitors for 60 cancer cell lines [Mahé & Vert, 2009]

The choice of kernel matters

- COREL14: 1400 natural images, 14 classes
- **Kernels:** histogram (H), walk kernel (W), subtree kernel (TW), weighted subtree kernel (wTW), combination (M).



[Harchaoui & Bach, 2007]

Summary

- Linearly separable case: **hard-margin SVM**
- Non-separable, but still linear: **soft-margin SVM**
- Non-linear: **kernel SVM**
- Kernels for
  - real-valued data
  - strings
  - graphs.

- *A Course in Machine Learning.*
  `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`
  - **Soft-margin SVM :** Chap 7.7
  - **Kernel SVM:** Chap 11.1 – 11.6
- *The Elements of Statistical Learning.*
  `http://web.stanford.edu/~hastie/ElemStatLearn/`
  - **Separating hyperplane:** Chap 4.5.2
  - **Soft-margin SVM:** Chap 12.1 – 12.2
  - **Kernel SVM:** Chap 12.3
  - String kernels: Chap 18.5.1
- *Learning with Kernels*
  `http://agbs.kyb.tuebingen.mpg.de/lwk/`
  - **Soft-margin SVM:** Chap 1.4
  - **Kernel SVM:** Chap 1.5
  - **SVR**: Chap 1.6
  - **Kernels:** Chap 2.1
- *Convex Optimization*
  `https://web.stanford.edu/~boyd/cvxbook/`
  - **SVM optimization :** Chap 8.6.1

Practical matters

- **Preparing for the exam**
  - Previous exams with solutions on the course website
- **Next week: special session! 2 x 1.5 hrs**
  - Introduction to **artificial neural networks**
  - Introduction to **deep learning** and **Tensorflow** (J. Boyd)

    Jupyter notebook will be available for download
  - **Deep learning for bioimaging** (P. Naylor)

Lab

- Redefining cross_validate

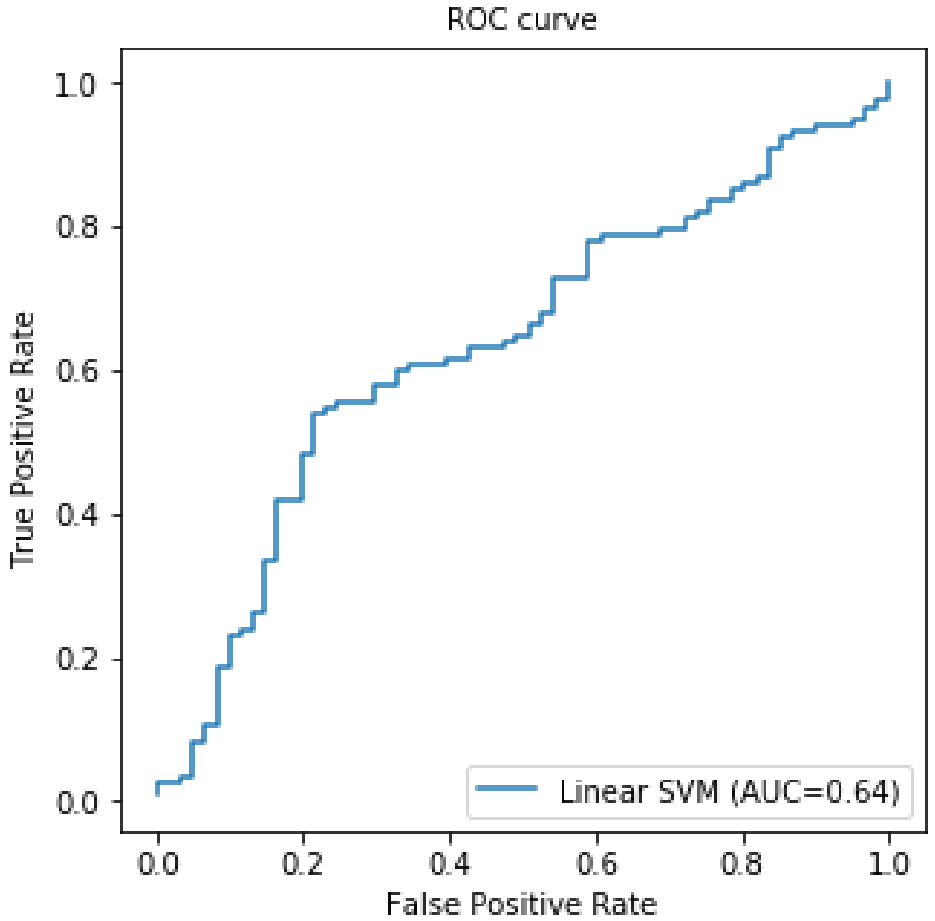
```
def cross_validate(design_matrix, labels, classifier, cv_folds):
```

```
pred = np.zeros(labels.shape) # Hold all predictions, in correct order.
for tr, te in cv_folds:
    # Restrict data to train/test folds
    Xtr = design_matrix[tr, :]
    ytr = labels[tr]
    Xte = design_matrix[te, :]

    # Fit classifier
    classifier.fit(Xtr, ytr)

    # Compute decision function on test data
    # TODO
    yte = clf.decision_function(Xte)

    # Update pred
    pred[te] = yte

return pred
```

Linear SVM

The data is not easily separated by a hyperplane

Support vectors are either correctly classified points that support the margin or errors.



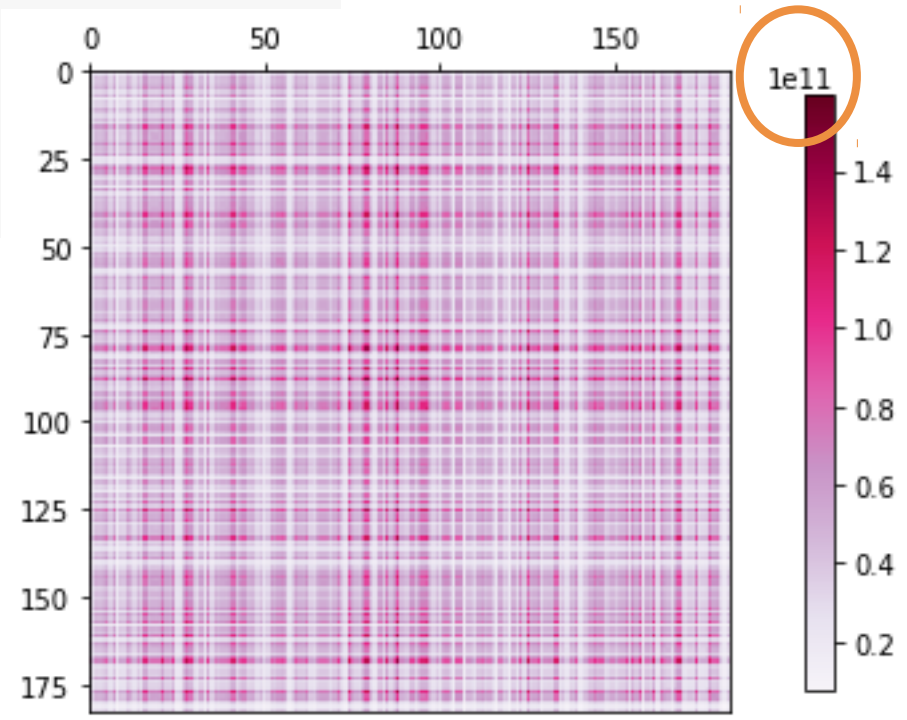

```
# TODO
print('No. support vectors class 0: %d' % clf.n_support_[0])
print('No. support vectors class 1: %d' % clf.n_support_[1])
print('No. training samples: %d' % X.shape[0])
```

```
No. support vectors class 0: 43
No. support vectors class 1: 62
No. training samples: 183
```

Many support vectors suggest the data is not easy to separate and there are many erros.

Linear kernel matrix

```
kmatrix = X.dot(X.T) # TODO

# heatmap + color map
fig, ax = plt.subplots(figsize=(5, 5))
plot = ax.imshow(kmatrix, cmap=plt.cm.PuRd)

# set axes boundaries
ax.set_xlim([0, X.shape[0]]) ; ax.set_ylim([0, X.shape[0]])

# flip the y-axis
ax.invert_yaxis() ; ax.xaxis.tick_top()

# plot colorbar to the right
plt.colorbar(plot, pad=0.1, fraction=0.04)
```



No visible pattern.

Dark lines correspond to vectors with highest magnitude.

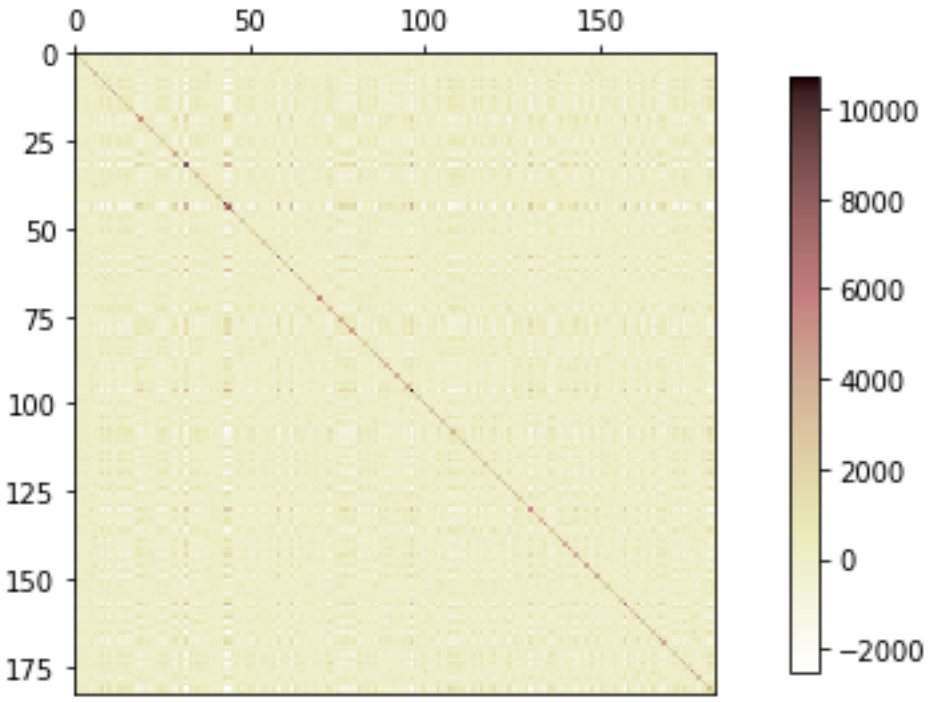
Linear kernel matrix (after feature scaling)

```
# TODO
X_scaled = preprocessing.scale(X)
kmatrix = X_scaled.dot(X_scaled.T)

# heatmap + color map
fig, ax = plt.subplots(figsize=(5, 5))
plot = ax.imshow(kmatrix, cmap=plt.cm.pink_r)
```

The kernel values are on a smaller scale than previously.

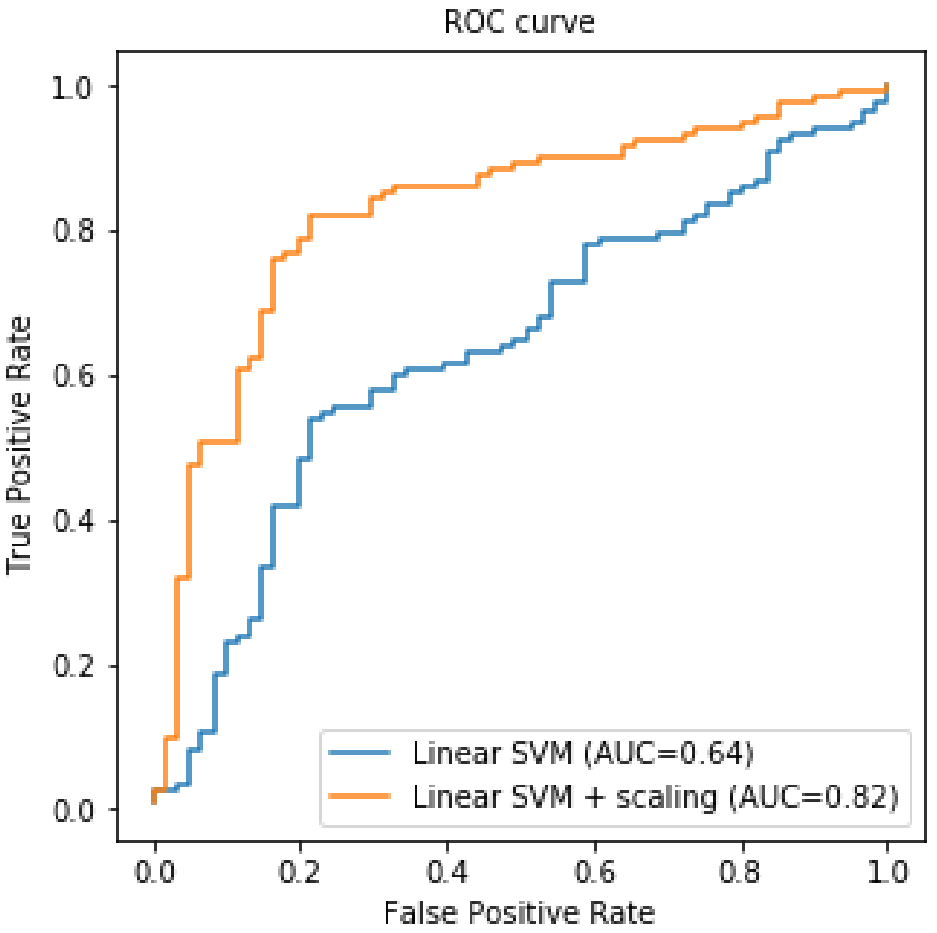
The diagonal emerges (the most similar sample to an observation is itself).

Many small values.



```
def cross_validate_with_scaling(design_matrix, labels, classifier, cv_folds):
```

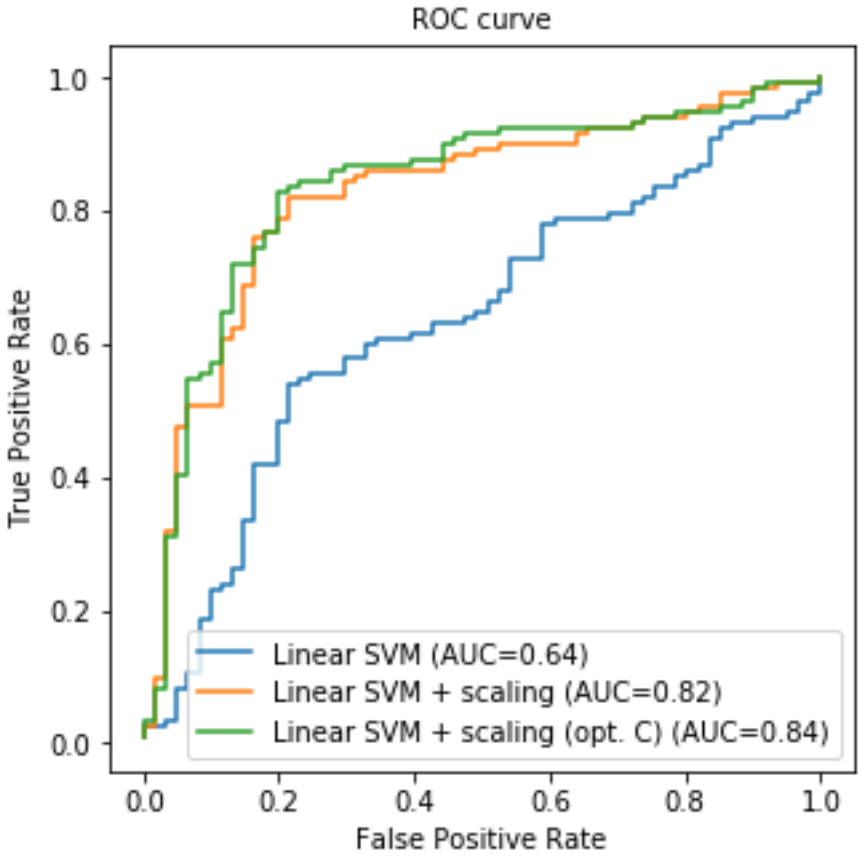
```
pred = np.zeros(labels.shape)
for tr, te in cv_folds:
    # Restrict data to train/test folds
    Xtr = design_matrix[tr, :]
    ytr = labels[tr]
    Xte = design_matrix[te, :]

    # Create scaler object
    # TODO
    sc = preprocessing.StandardScaler()

    # Fit the scaler and transform training data
    # TODO
    Xtr = sc.fit_transform(Xtr)

    # Transform test data
    # TODO
    Xte = sc.transform(Xte)

    # Fit classifier
    classifier.fit(Xtr, ytr)

    # Compute decision function on test data
    # TODO
    yte = classifier.decision_function(Xte)

    # Update pred
    # TODO
    pred[te] = yte

return pred
```



Linear SVM with optimal C

An SVM classifier with optimized C

```
from sklearn.model_selection import GridSearchCV
parameters_dict = {'C': np.logspace(-4, 2, 7)}

clf_C = GridSearchCV(svm.SVC(kernel='linear'),
                     parameters_dict, scoring='roc_auc')

ypred_linear_scaled = cross_validate_with_scaling(X, y, clf_C, folds)
```



On each pair (tr, te):
- scaling factors are computing on Xtr
- Xtr, Xte are scaled accordingly
- for each value of C:
  - an SVM is cross-validated on Xtr_scaled
- the best of these SVM is trained on the full Xtr_scaled and applied to Xte_scaled (this produces one prediction per data point of X)

Polynomial kernel SVM

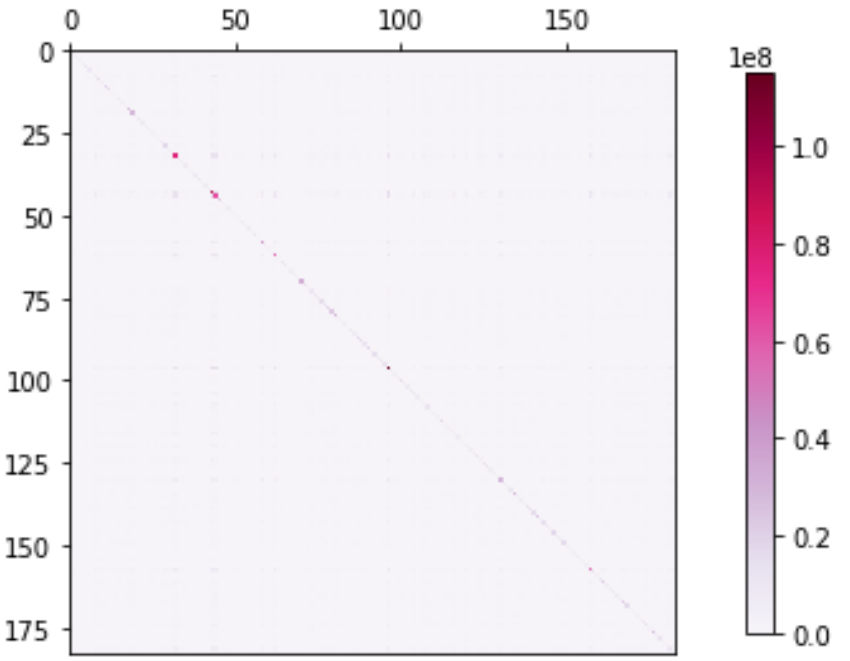
Polynomial kernel with
r=0
d=2

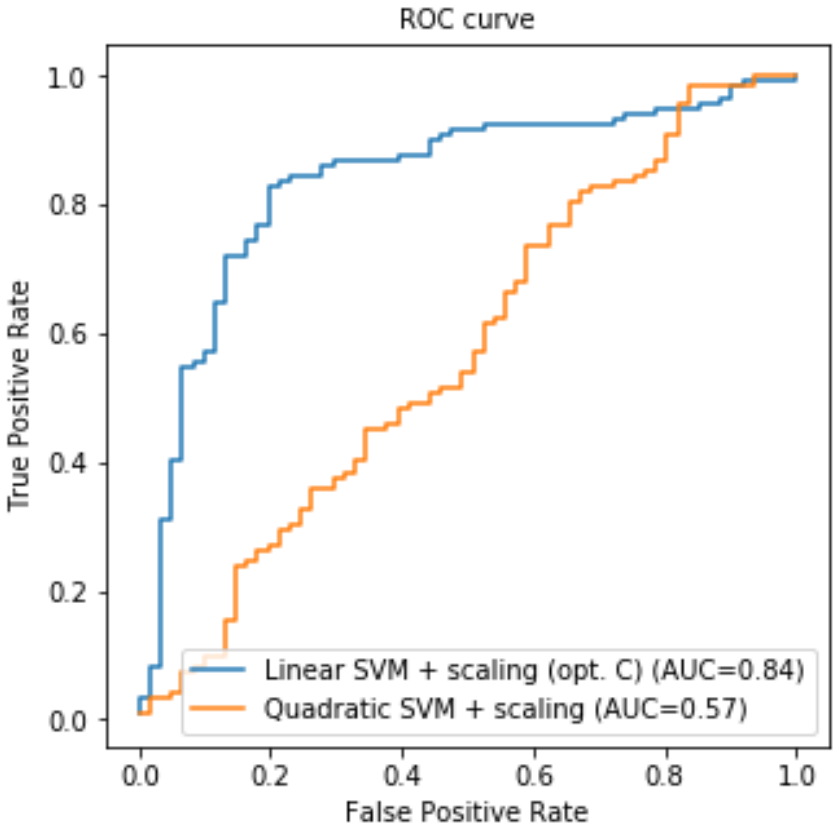
Computed on X_scaled

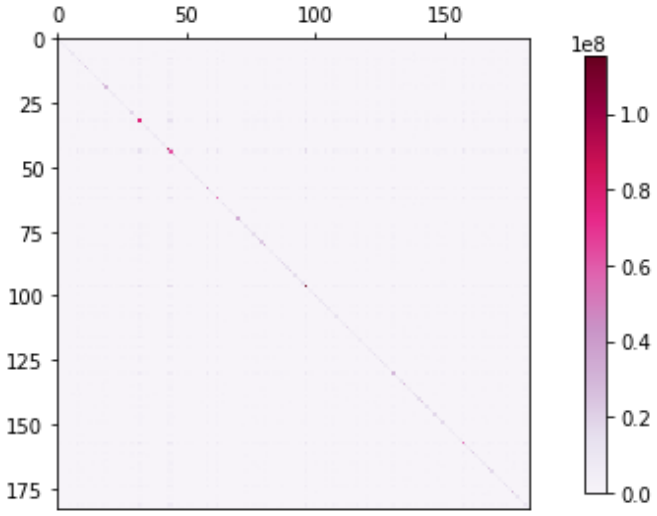
The matrix is really close to identity, nothing can be learned.

This gets worse if you increase d.

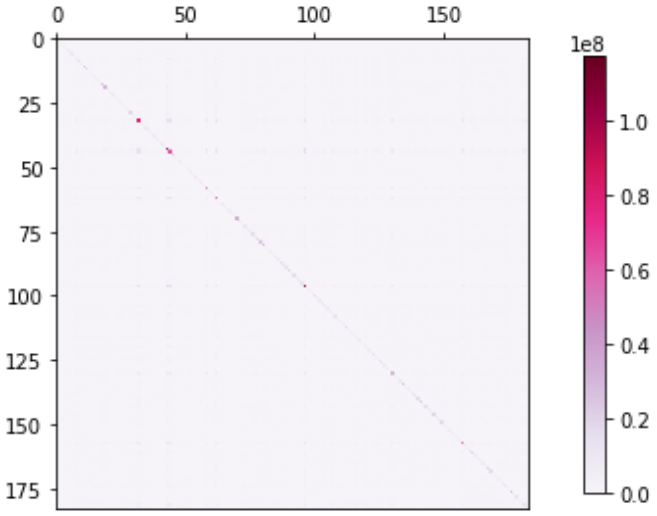
Changing r can give us a more reasonable matrix.



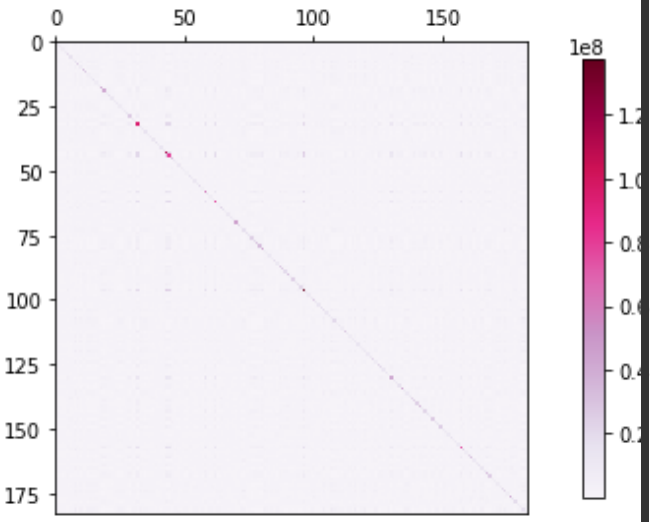




r=10

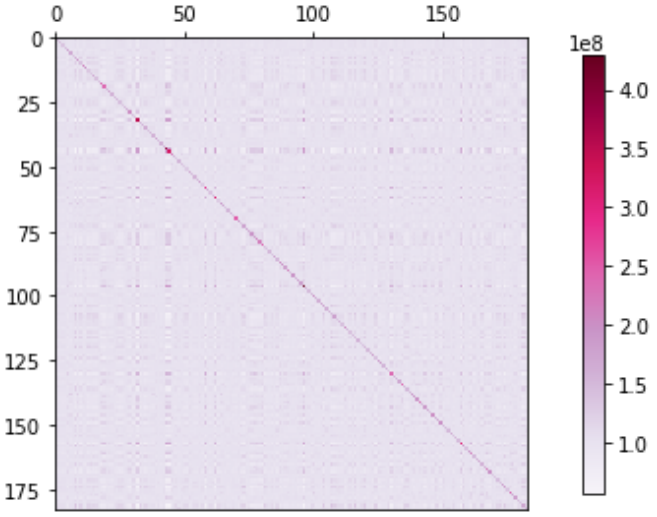


r=100

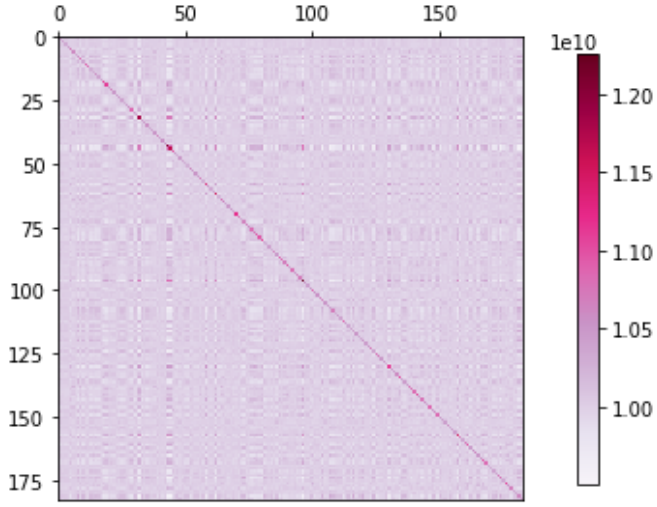


r=1000

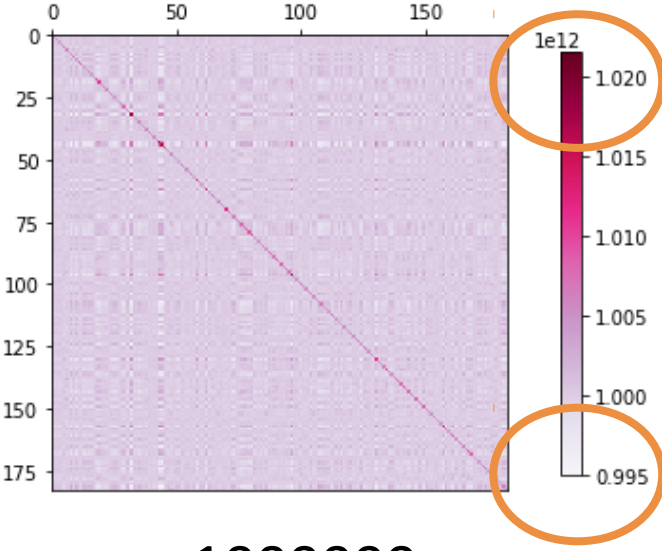
The kernel matrix is almost the identity matrix



r=10000

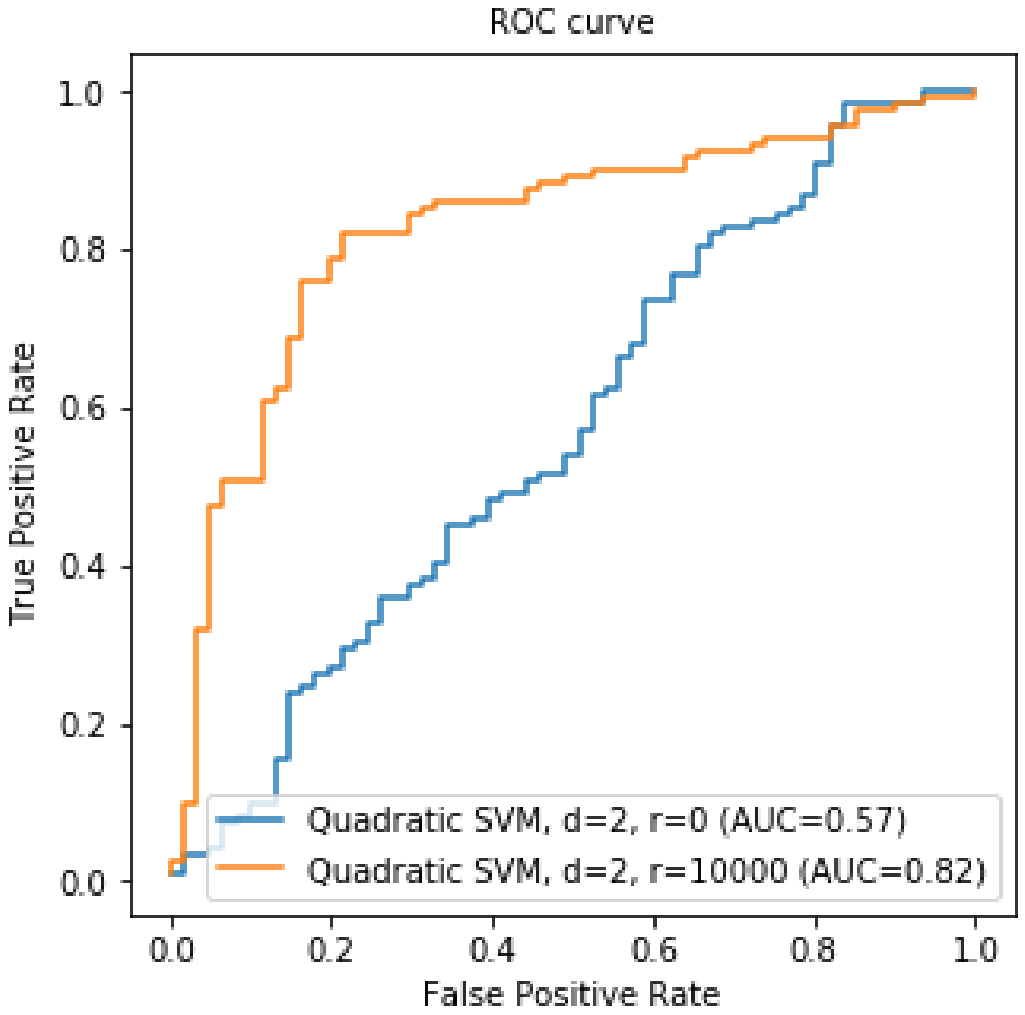


r=100000



r=1000000
Almost all 1s

Reasonable range of values for r



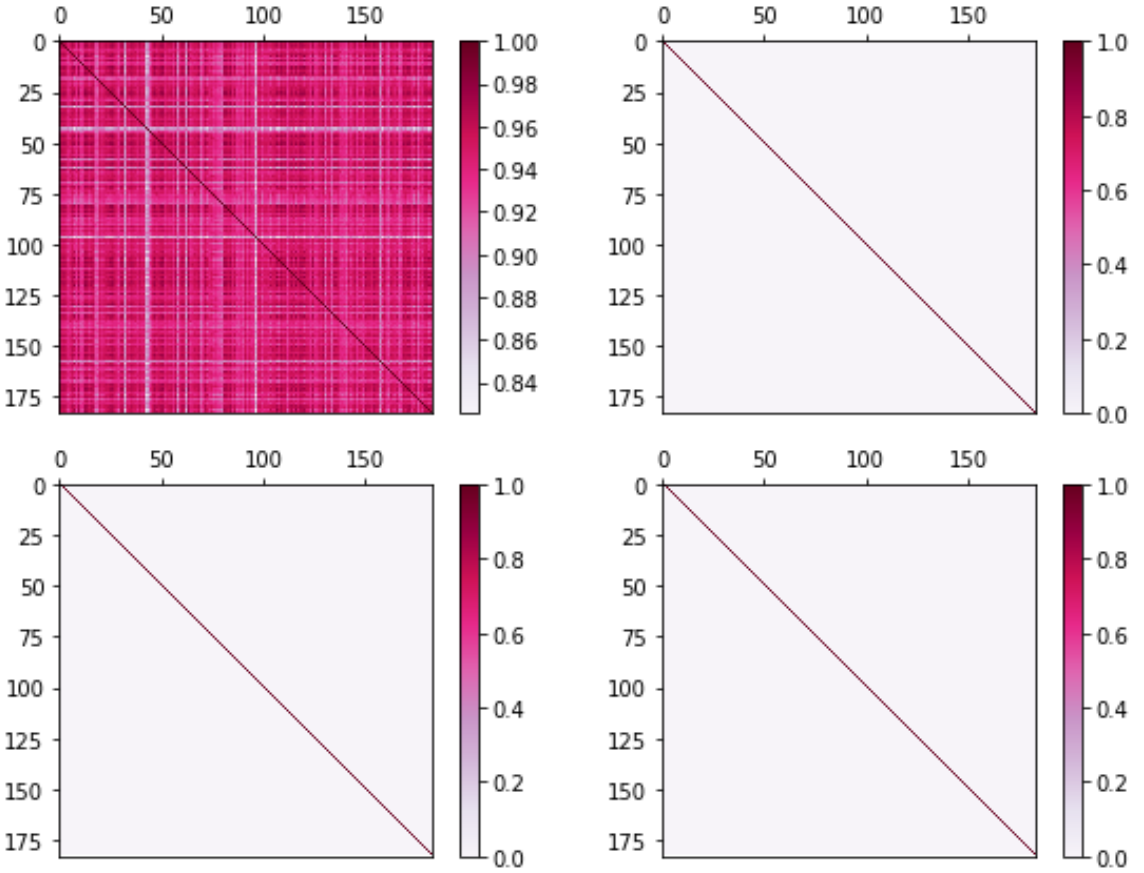
- For a fair comparison with the linear kernel, cross-validate C and r.
- For r, use a logspace between 10000 and 100000 based on your observation of the kernel matrix.

Gaussian kernel SVM

- What values of gamma should we use? Start by spreading out values.

```
for i, gamma in enumerate([1e-5, 1e-2, 1e2, 1e5]):
    ax = fig.add_subplot(2, 2, i + 1)
    pairwise_sq_dists = rbf_kernel(X_scaled, gamma=gamma)
```

- When gamma > 1e-2, the kernel matrix is close to the identity.
- When gamma = 1e-5, the kernel matrix is getting close to a matrix of all 1s.
- If we choose gamma much smaller, the kernel matrix is going to be so close to a matrix of all 1s the SVM won't learn well.

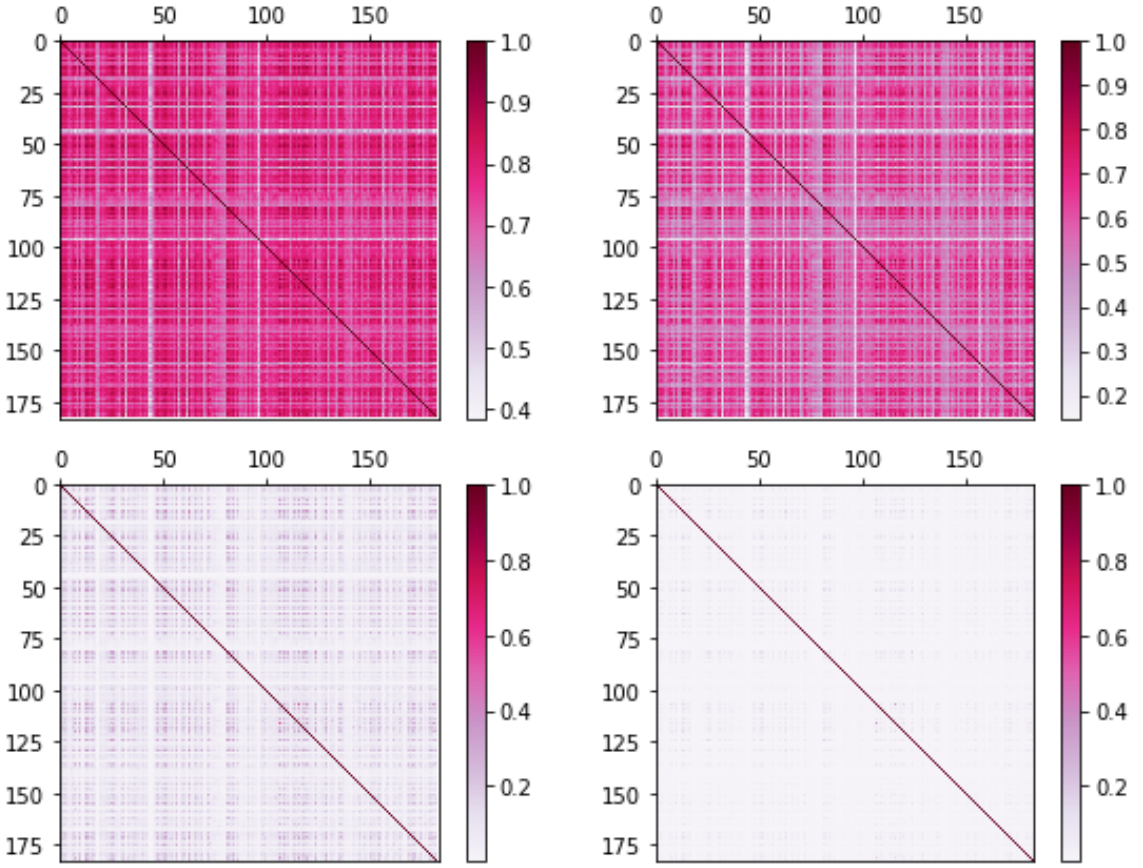


Gaussian kernel SVM

- What values of gamma should we use? Start by spreading out values.

```
for i, gamma in enumerate([5e-5, 1e-4, 5e-4, 1e-3]):
    ax = fig.add_subplot(2, 2, i + 1)
    pairwise_sq_dists = rbf_kernel(X_scaled, gamma=gamma)
```

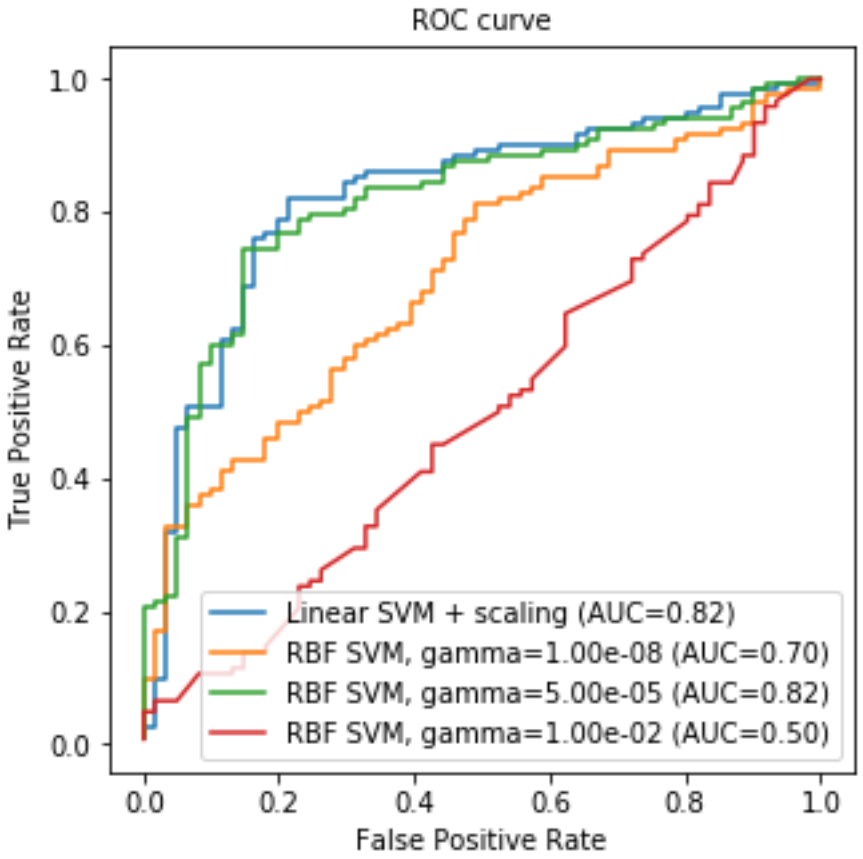
- The kernel matrix is more reasonable when gamma is between 5e-5 and 5e-4.



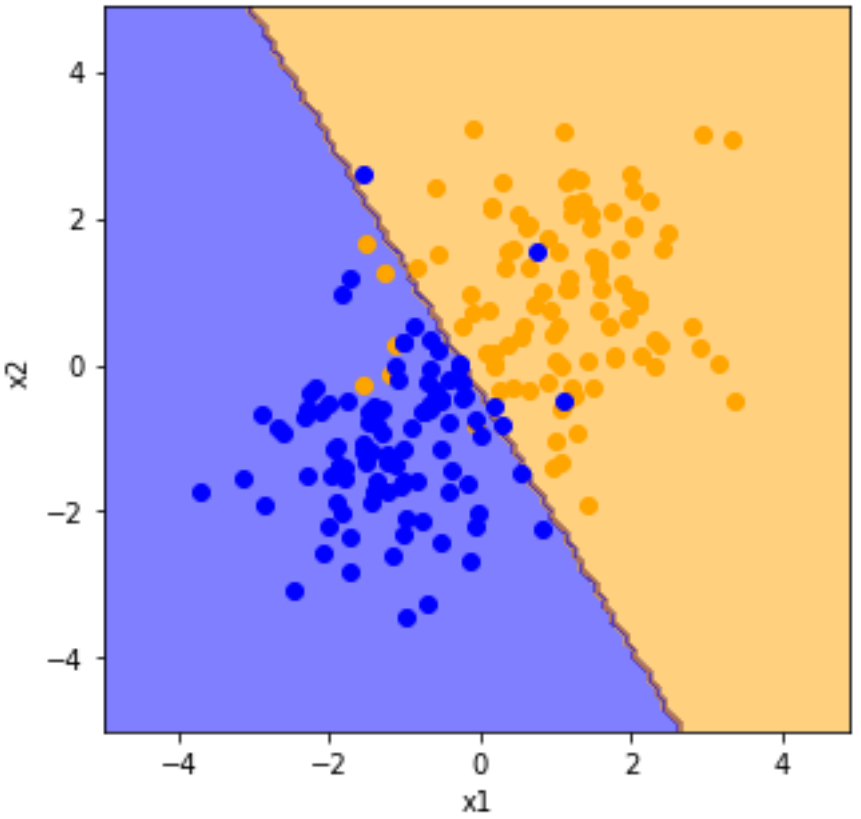
Gaussian kernel SVM

```
fig, ax = plt.subplots(figsize=(5, 5))
ax.plot(fpr2, tpr2,
         label='Linear SVM + scaling (AUC=%.2f)' % auc2)

for g in [1e-8, 5e-5, 1e-2]:
    clf = svm.SVC(kernel='rbf', gamma=g)
    ypred_rbf = cross_validate_with_scaling(X, y, clf, folds)

    fpr5, tpr5, thresholds5 = metrics.roc_curve(y, ypred_rbf)
    auc5 = metrics.auc(fpr5, tpr5)

    plt.plot(fpr5, tpr5,
             label='RBF SVM, gamma=%.2e (AUC=%.2f)' %
```

- The best performance we obtain is indeed for a gamma of 5e-5.
- To fairly compare to the linear SVM, one should cross-validate C.

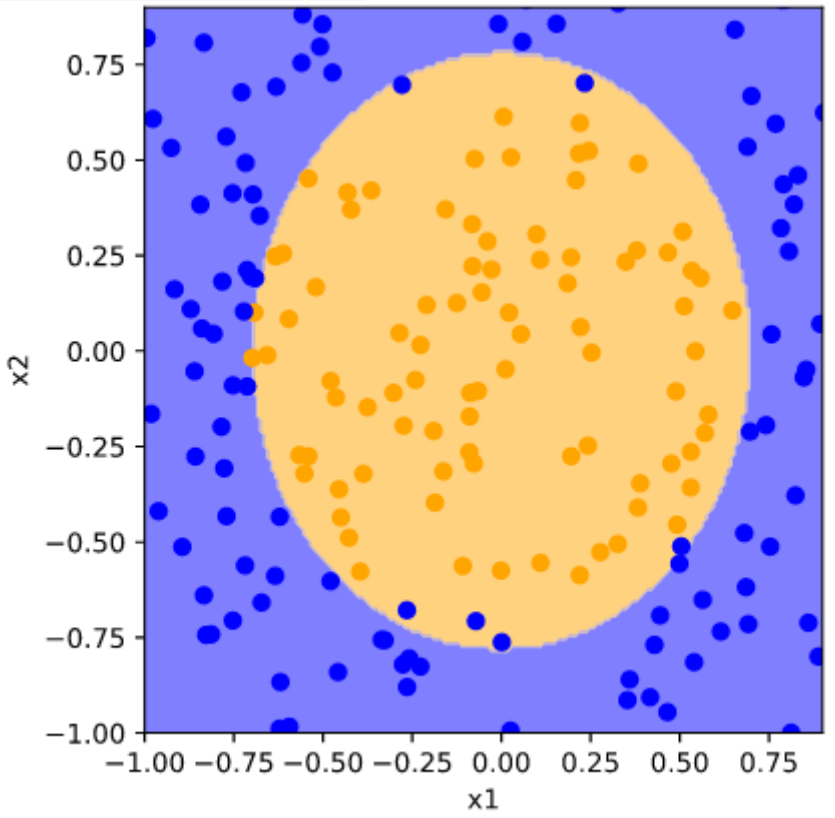


Linear SVM decision boundary



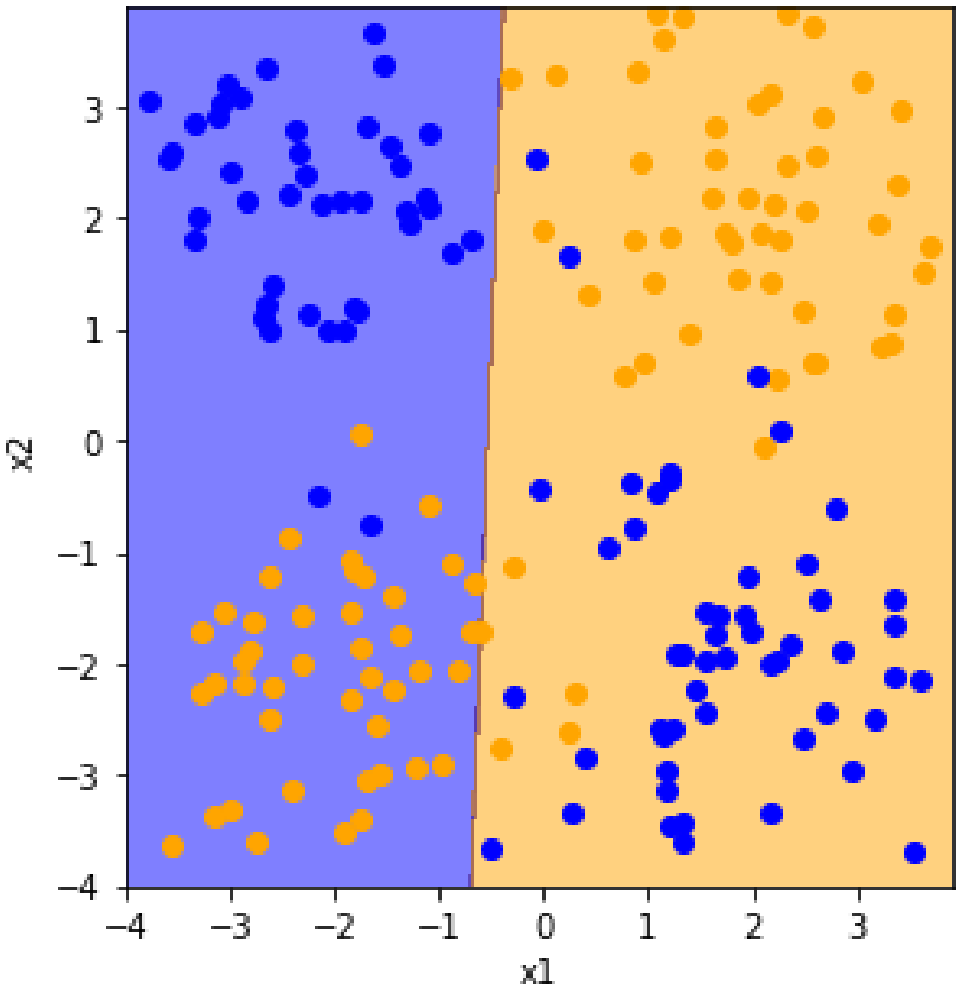
Quadratic SVM decision boundary

```
# create mesh coordinates
lim = 1
xx, yy = np.meshgrid(np.arange(-lim, +lim, 0.01), np.arange(-lim, +lim, 0.01))
zz_poly = np.zeros(xx.shape)

# create mesh of predictions
for i in range(xx.shape[0]):
    for j in range(xx.shape[1]):
        zz_poly[i, j] = clf_poly.predict(np.array([xx[i, j], yy[i, j]]))

# visualise mesh as contour plot
fig = plt.figure(figsize=(10, 5))

ax = fig.add_subplot(1, 2, 2)
ax.set_xlabel('x1') ; ax.set_ylabel('x2')
ax.set_xlim([-lim, +lim-0.1]) ; ax.set_ylim([-lim, +lim-0.1]
ax.contourf(xx, yy, zz_poly, alpha=0.5, colors=('blue', 'ora

# plot training data
ax.scatter(X[class_pos, 0], X[class_pos, 1], color='orange')
ax.scatter(X[class_neg, 0], X[class_neg, 1], color='blue')
```



Separating XOR

linear kernel, accuracy=0.48



RBF kernel, accuracy=0.98