Linear models are great

- I *lied* to you in the (linear) regression example :

  For small enough data sets, **you can one-shot the solution !**
  (compute it explicitly in a single iteration, no need for iterative methods like GD)

- Use some algebra and find:

Feature maps

Feature maps

Poor model (not very expressive/complex) ?
→ **increase complexity** by **adding** input **features** !
→ But **how** ? Can I make up additional data features ?

- A simple spatial transformation (a *map*)

$$\phi : \mathbb{R}^D \to \mathbb{R}^{D'}, \quad \vec{x}_i \mapsto \vec{\phi}(\vec{x}_i), \quad D < D'$$

e.g. polynomial feature maps

- Data ***standardization*** is the simplest example:

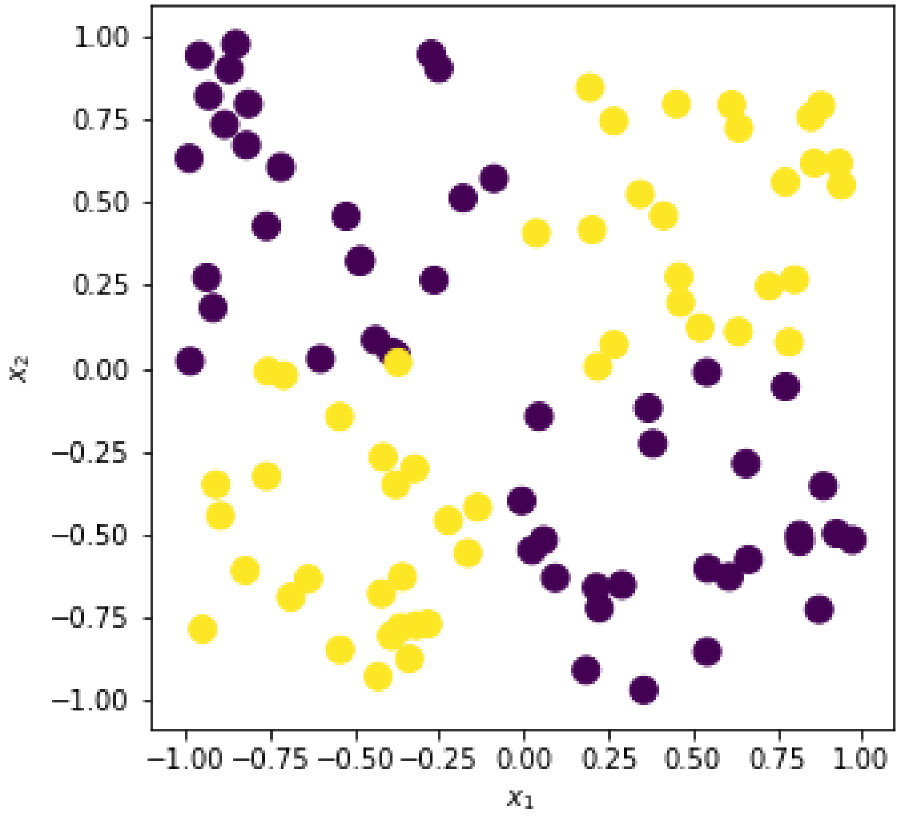
$$\text{for a fixed } \vec{x} = \vec{x}_n, \quad \vec{\phi}(\vec{x}_{(D)}) = (\phi_1(x_1), \phi_2(x_2), \ldots, \phi_D(x_D))^T$$

$$\phi_d(x_{nd}) = \frac{x_{nd} - m_d}{s_d} \qquad m_d = \langle x_{nd} \rangle_n = \frac{1}{N}\sum_n x_{nd}$$

$$s_d = \sigma(x_{nd})_n = \sqrt{\frac{1}{N}\sum_n (x_{nd} - m_d)^2}$$

(subtract by the mean, divide by the standard deviation, each component independently)

Can you ***linearly*** separate *this* ?



The famous XOR data set

Answer: yes you can !! (in a sense)
→ want to see a magic trick ?

Your first ***feature map***

→ we can **make up new features** !
(By combining the original ones)

For instance: *D=3, D'=10*

$$\vec{x}^{(n)} = (x_1, x_2, x_3)^{(n)} \longrightarrow \phi(\vec{x}^{(n)})$$

$$\vec{\phi}(\vec{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3, x_1^2, x_2^2, x_3^2)$$

Why these weird coefficients? Because: (check it)

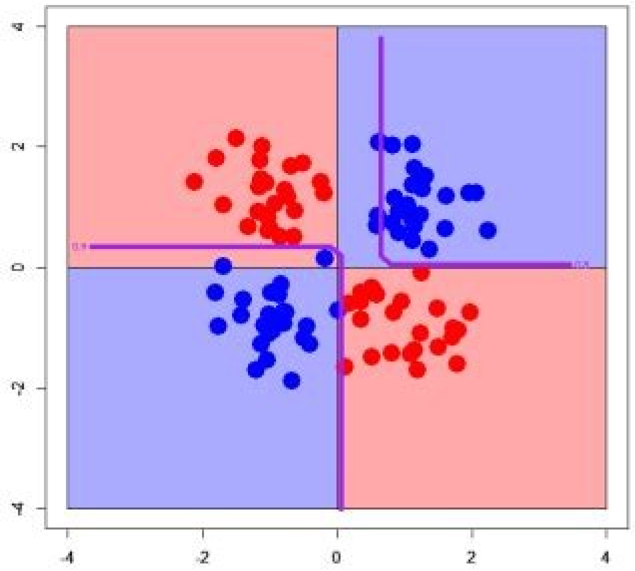
$$\vec{\phi}(x) \cdot \vec{\phi}(x') = (1 + \vec{x} \cdot \vec{x'})^2$$

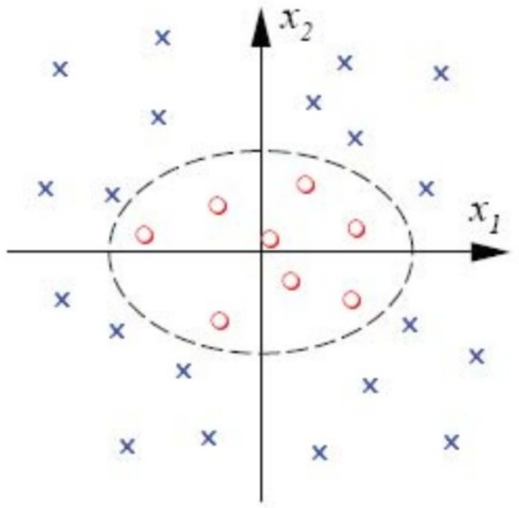
→ equivalent to using a new definition of the scalar product:

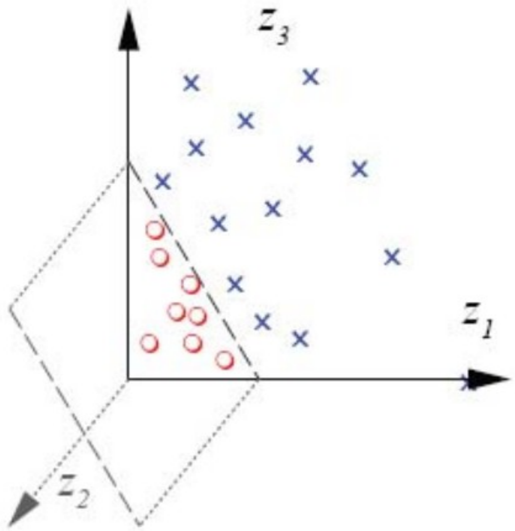
$$\vec{x} \cdot \vec{x}' \mapsto K(x, x') = \phi(x)\phi(x')$$

Other *feature maps*

What about these data sets:





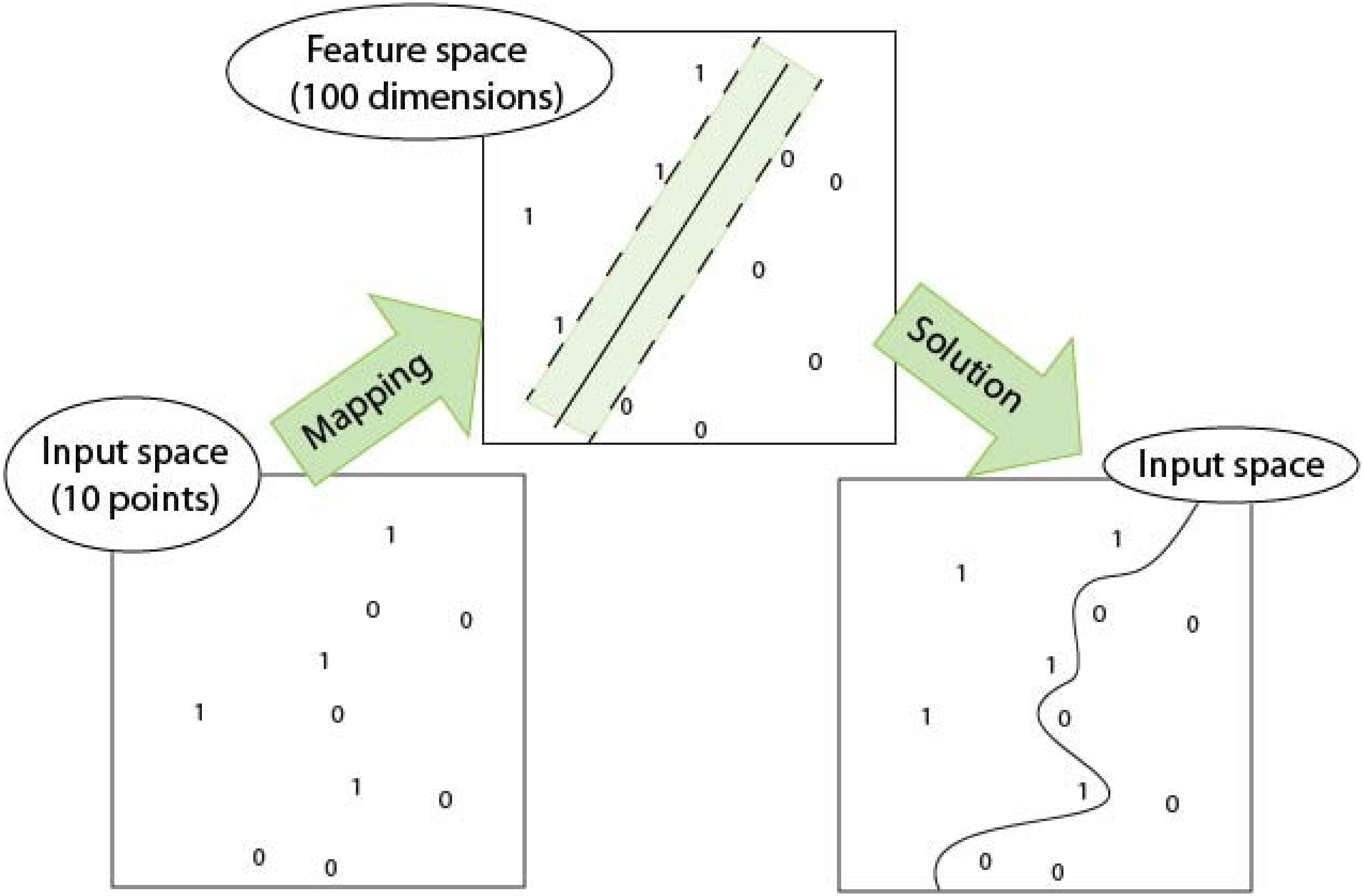


(separable, but not *linearly* separable a priori)

→ But in **sufficiently higher dimensional spaces**, data sets are ***always*** **linearly separable**

→ Be careful: the **number of parameters** (complexity C) can quickly **explode** with D, P.

Intuitive Sketch of *feature maps*



Curse of dimensionality

- Beyond the $(N\text{-}1)^{\text{th}}$ order polynomials for regression, there is a general result: *(Cover theorem)*

  "*Any binary classification problem for N samples can be fitted exactly using N features*" (actually, we need less than that, but it's not important).
- This is ***bad*** news: if we add to many features on not enough data, we will perfectly … **over**-fit the (*training*) data.
- This will typically generalize very poorly
- One needs to keep this in mind at all times

References:

- a very clear explanation for ***feature maps***:
https://scikit-learn.org/stable/modules/linear_model.html#polynomial-regression

- *Bishop book*, page 291-294 (section 6.1)

- our next classes, when we'll talk about Kernels

Before the tutorial
a word on the *sklearn* logic

Scikit-learn Logic

- A word on sklearn:
  https://sklearn.org/tutorial/basic/tutorial.html
- Model = a python class
- Most models have some key methods:
  - fit(X,y)   (or fit(X) when unsupervised)
  - predict(x)
  - score (often)
  - set_params (actually for changing hyper-params)
  - etc
- The logic is sound : understanding it makes you understand ML in general !