Introduction to Machine Learning

SCP8084699 - LT Informatica

Support Vector Machines: loss function

Prof. Lamberto Ballan

DIPARTIMENTO MATEMATICA
Department of Mathematics "Tullio Levi-Civita"

What we learned until now

(a brief summary)

- By now you have seen a range of different machine learning algorithms

  ‣ Linear regression / classification

  ‣ Logistic regression

  ‣ Artificial Neural Networks

  Parametric models

  Goal: $h_\theta(\mathbf{x}),\ \boldsymbol{\theta}$

- Regularization, bias-variance tradeoff, evaluation and diagnosing machine learning systems

# Support Vector Machines (SVM)

- SVM is a very powerful and popular algorithm

    ‣ It is a <u>supervised learning</u> algorithm usually used for classification (but can be used also for regression)

    ‣ It is a <u>non-probabilistic</u> (binary) classifier, although there are methods such as Platt scaling to give a probabilistic interpretation of the SVM output

    ‣ It is a <u>linear classification</u> model but SVM can efficiently perform non-linear classification using the *kernel trick*

Another view on Logistic Regression

- Hypothesis representation:

$$h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

where $g(z) = \dfrac{1}{1 + e^{-z}}$ (*Sigmoid* or *Logistic* function)
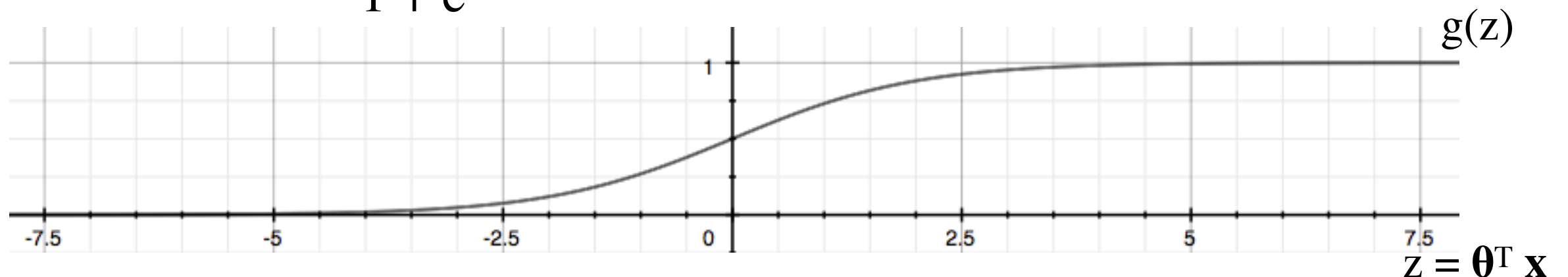
- Interpretation (i.e. what we'd like logistic regression to do):

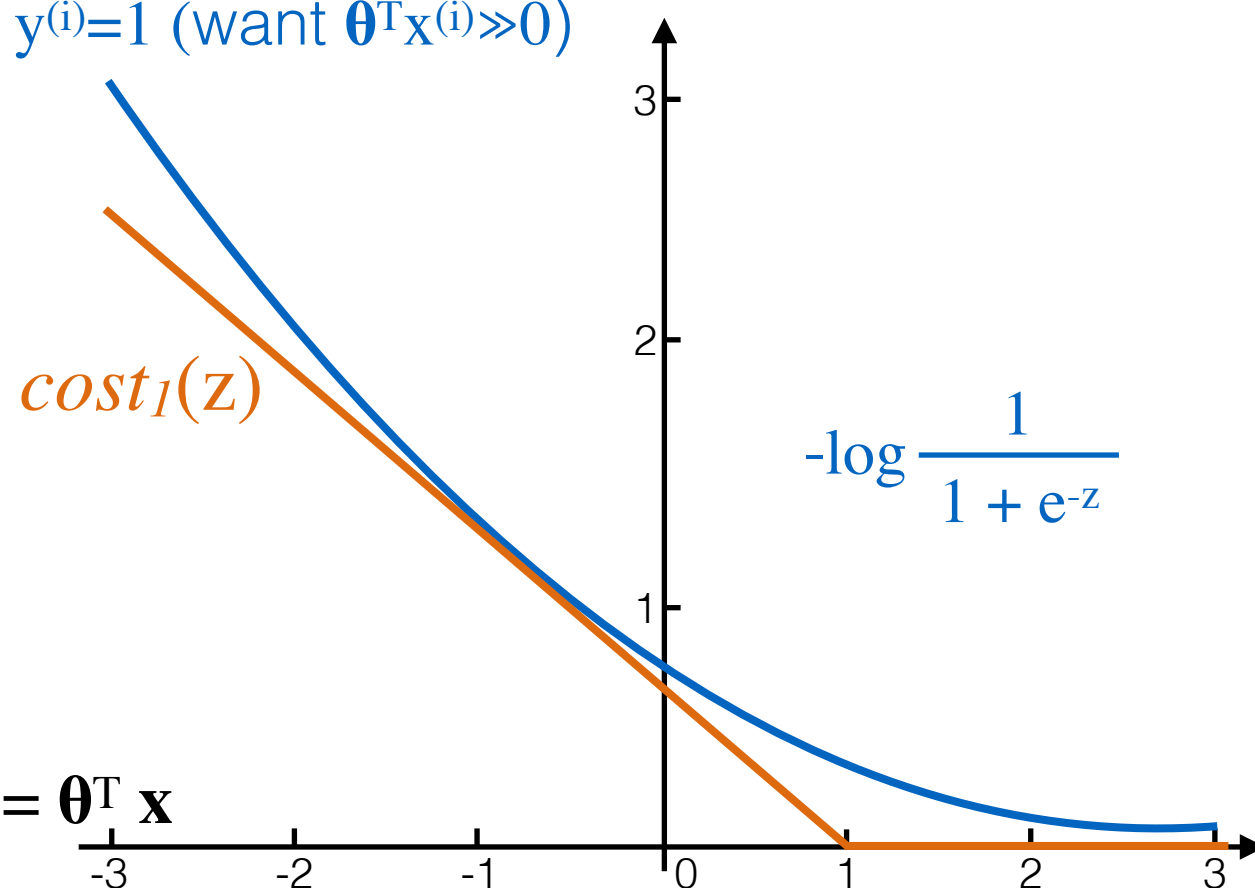 If $y = 1$, we want $h_\theta(\mathbf{x}) \approx 1$, $\boldsymbol{\theta}^T \mathbf{x} \gg 0$

  If $y = 0$, we want $h_\theta(\mathbf{x}) \approx 0$, $\boldsymbol{\theta}^T \mathbf{x} \ll 0$

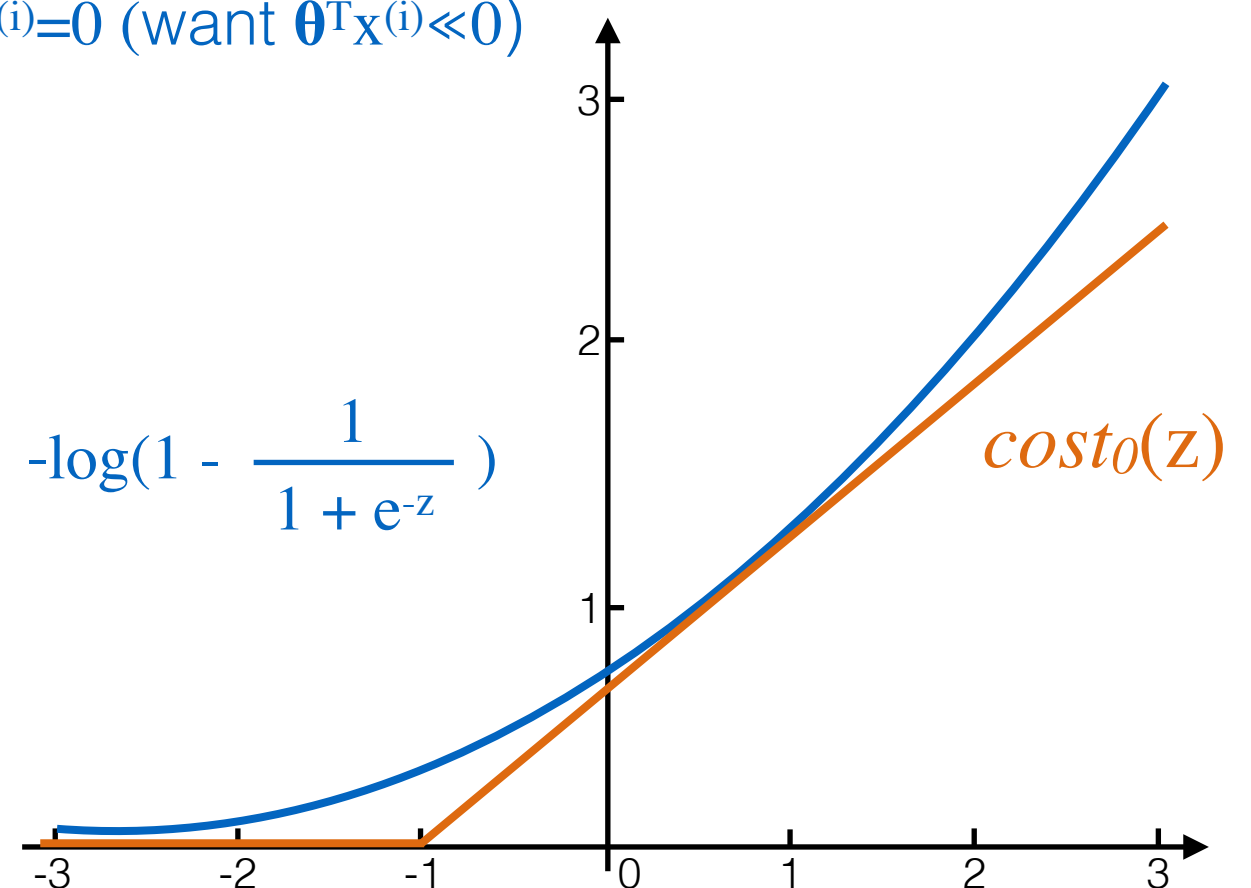# Another view on Logistic Regression

- Loss function: $J(\boldsymbol{\theta}) = \dfrac{1}{m} \displaystyle\sum_{i=1}^{m} cost(h_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)})$ *where:*

  ▸ If we take the definition for $h_\theta(\mathbf{x})$ and plug it in, we get:

$$cost(\cdot) = \boxed{-\mathbf{y}^{(i)} \cdot \log\left(\dfrac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}^{(i)}}} \right)} \boxed{- (1 - \mathbf{y}^{(i)}) \cdot \log\left(1 - \dfrac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}^{(i)}}} \right)}$$

if $\mathbf{y}^{(i)}=1$ (want $\boldsymbol{\theta}^T\mathbf{x}^{(i)} \gg 0$)

$cost_1(z)$

$-\log \dfrac{1}{1 + e^{-z}}$

$z = \boldsymbol{\theta}^T \mathbf{x}$

if $\mathbf{y}^{(i)}=0$ (want $\boldsymbol{\theta}^T\mathbf{x}^{(i)} \ll 0$)

$-\log\left(1 - \dfrac{1}{1 + e^{-z}}\right)$

$cost_0(z)$

SVM: optimization objective

- (Regularized) Logistic Regression:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^{m} - y^{(i)} \cdot \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \cdot \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

- Support Vector Machine:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \cdot cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) \cdot cost_0(\theta^T x^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

$$C\,A\ +\ B\quad,\ \ C = \frac{1}{\lambda}$$

$$\Rightarrow \quad \min_{\theta}\ C \sum_{i=1}^{m} \left[y^{(i)} \cdot (cost_1(\theta^T x^{(i)})) + (1-y^{(i)}) \cdot (cost_0(\theta^T x^{(i)})) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$
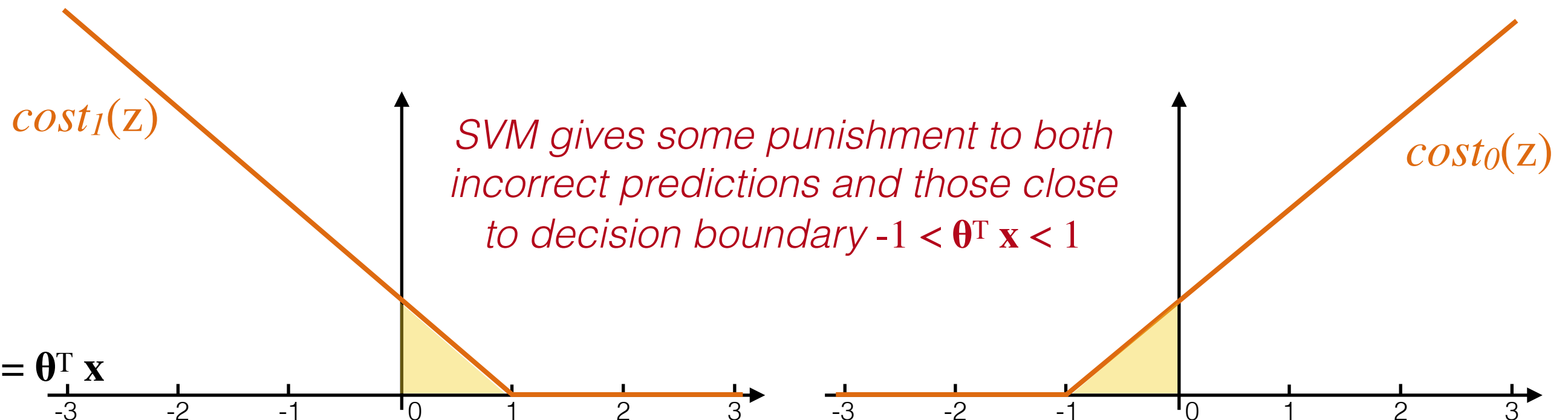
SVM: large margin intuition

- Optimization objective:

$$\min_{\boldsymbol{\theta}} \; C \sum_{i=1}^{m} \left[y^{(i)} \cdot (cost_1(\boldsymbol{\theta}^T x^{(i)})) + (1-y^{(i)}) \cdot (cost_0(\boldsymbol{\theta}^T x^{(i)})) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

‣ Interpretation (i.e. what we'd like SVM to do):

If $y^{(i)} = 1$, we want $\boldsymbol{\theta}^T x^{(i)} \geq 1$ (not just ≥ 0)

If $y^{(i)} = 0$, we want $\boldsymbol{\theta}^T x^{(i)} \leq -1$ (not just < 0)

$cost_1(z)$

SVM gives some punishment to both incorrect predictions and those close to decision boundary $-1 < \boldsymbol{\theta}^T x < 1$

$cost_0(z)$

$z = \boldsymbol{\theta}^T x$
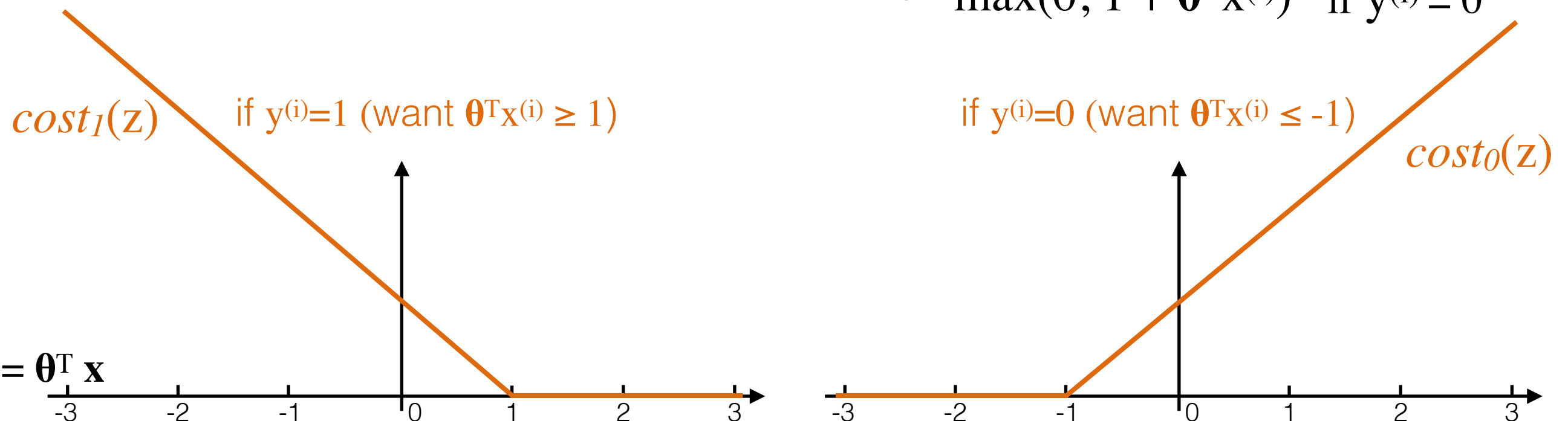
-3 -2 -1 0 1 2 3

-3 -2 -1 0 1 2 3

SVM: hypothesis and loss function

- Optimization objective:

$$\min_{\boldsymbol{\theta}} \; C \sum_{i=1}^{m} \left[y^{(i)} \cdot (cost_1(\boldsymbol{\theta}^T x^{(i)})) + (1-y^{(i)}) \cdot (cost_0(\boldsymbol{\theta}^T x^{(i)})) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

- Let's write the SVM's loss: $J(\boldsymbol{\theta}) = \sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y^{(i)})$

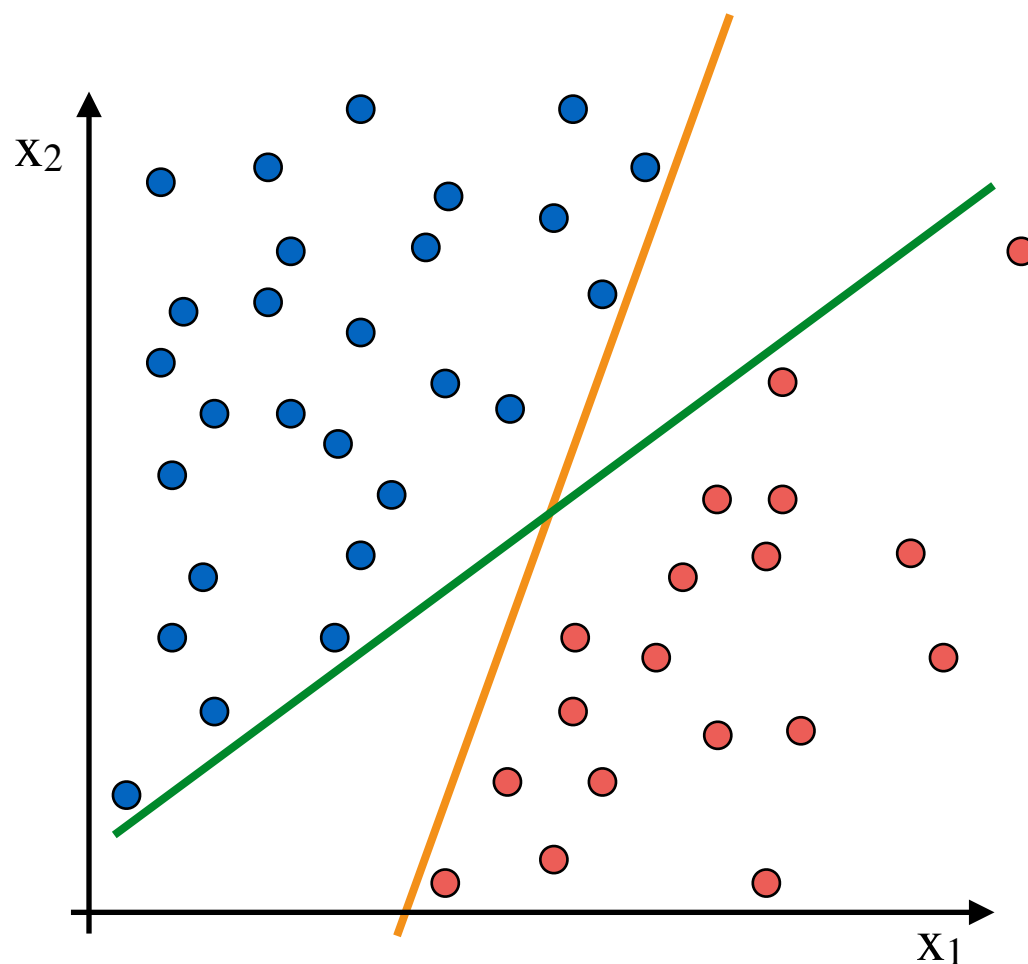 ‣ Hinge loss: $cost(h_\theta(x^{(i)}), y^{(i)}) = \begin{cases} \max(0, 1 - \boldsymbol{\theta}^T x^{(i)}) & \text{if } y^{(i)} = 1 \\ \max(0, 1 + \boldsymbol{\theta}^T x^{(i)}) & \text{if } y^{(i)} = 0 \end{cases}$

$cost_1(z)$ if $y^{(i)}=1$ (want $\boldsymbol{\theta}^T x^{(i)} \geq 1$)   if $y^{(i)}=0$ (want $\boldsymbol{\theta}^T x^{(i)} \leq -1$) $cost_0(z)$

$z = \boldsymbol{\theta}^T x$

-3 -2 -1 0 1 2 3   -3 -2 -1 0 1 2 3

SVM: decision boundary

• Large margin intuition:

(Linearly separable case)

Let's start with two possible lines…

SVM: decision boundary

- Large margin intuition:

$h_\theta(\mathbf{x})$

x_2

x_1

$\theta^T x = -1$

$\theta^T x = 0$

$\theta^T x = 1$
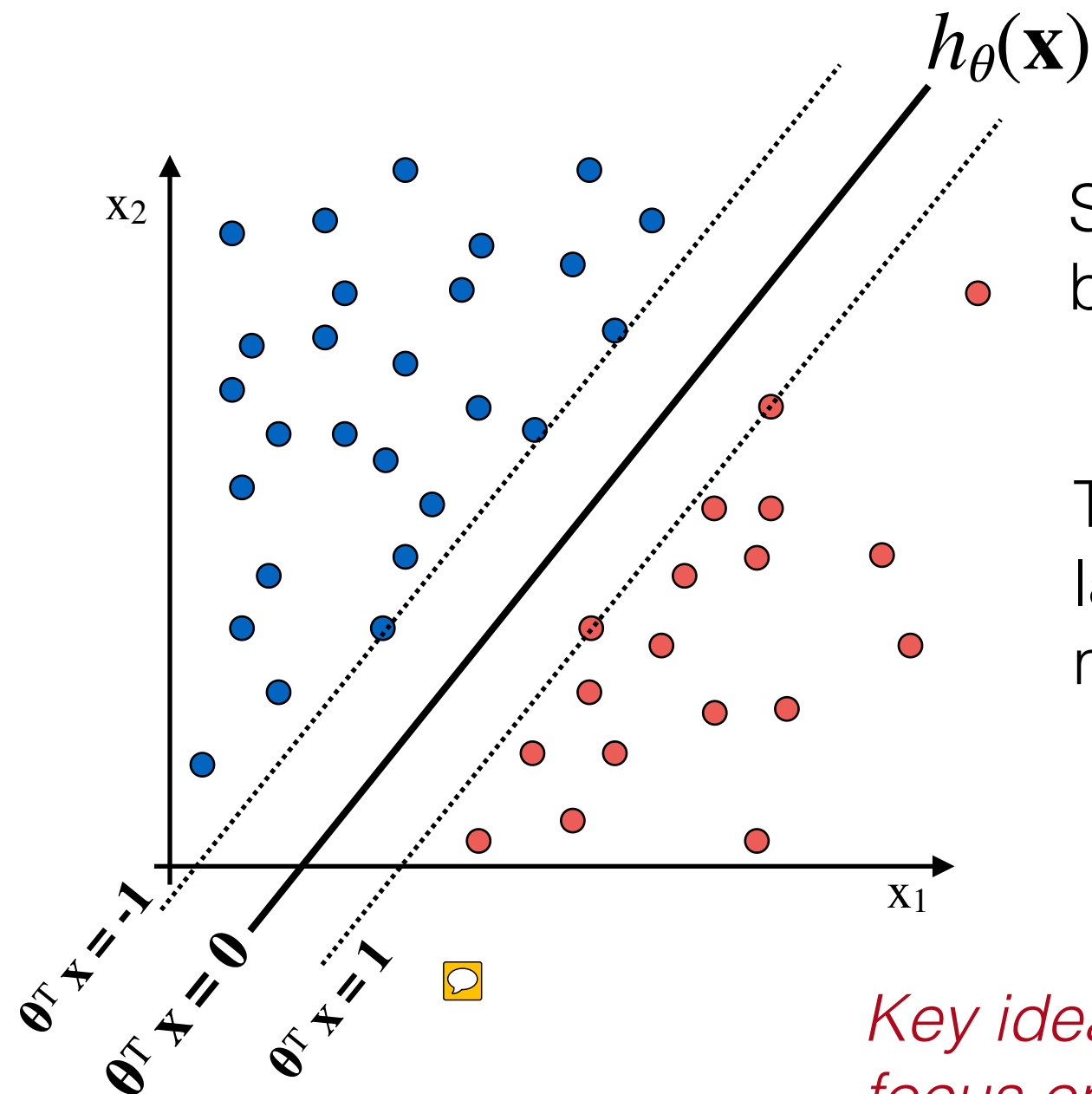
SVM will instead pick a decision boundary such as this one

This decision boundary has a larger min distance from any of my training samples

Key idea: instead of fitting all the points focus on boundary points
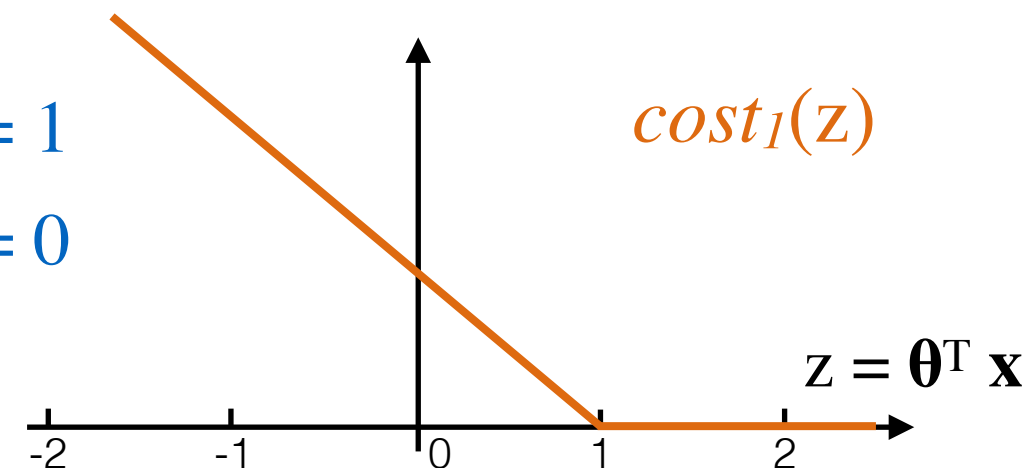
SVM: decision boundary

- Optimization objective:

$$\min_{\boldsymbol{\theta}} \; C \left[\sum_{i=1}^{m} \left[y^{(i)} \cdot (cost_1(\boldsymbol{\theta}^T x^{(i)})) + (1-y^{(i)}) \cdot (cost_0(\boldsymbol{\theta}^T x^{(i)})) \right] \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

e.g. $C \approx 10^5$ $\qquad\qquad\qquad\qquad\qquad\qquad \approx 0$
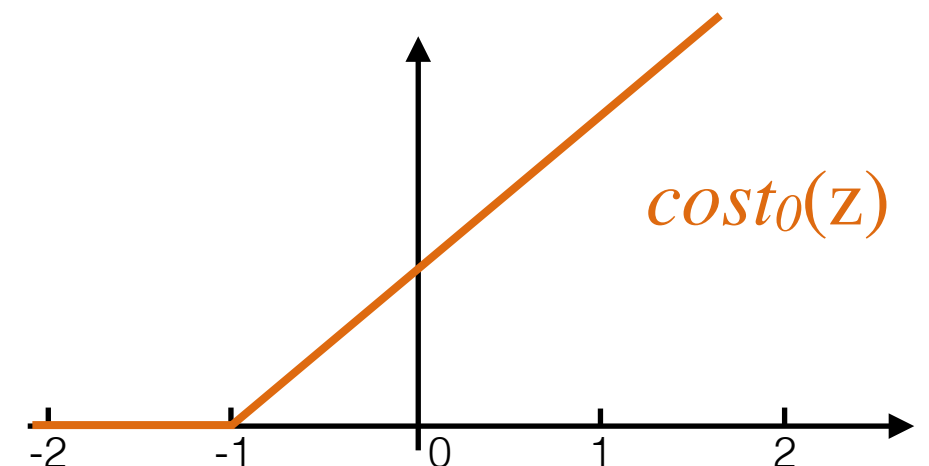
$$\min_{\boldsymbol{\theta}} \; \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \quad \text{s.t. } \boldsymbol{\theta}^T \mathbf{x}^{(i)} \geq 1 \text{ if } y^{(i)} = 1$$

$$\boldsymbol{\theta}^T \mathbf{x}^{(i)} \leq -1 \text{ if } y^{(i)} = 0$$

Whenever $y^{(i)} = 1$: $\boldsymbol{\theta}^T \mathbf{x}^{(i)} \geq 1$

$cost_1(z)$

$z = \boldsymbol{\theta}^T \mathbf{x}$

Whenever $y^{(i)} = 0$: $\boldsymbol{\theta}^T \mathbf{x}^{(i)} \leq -1$
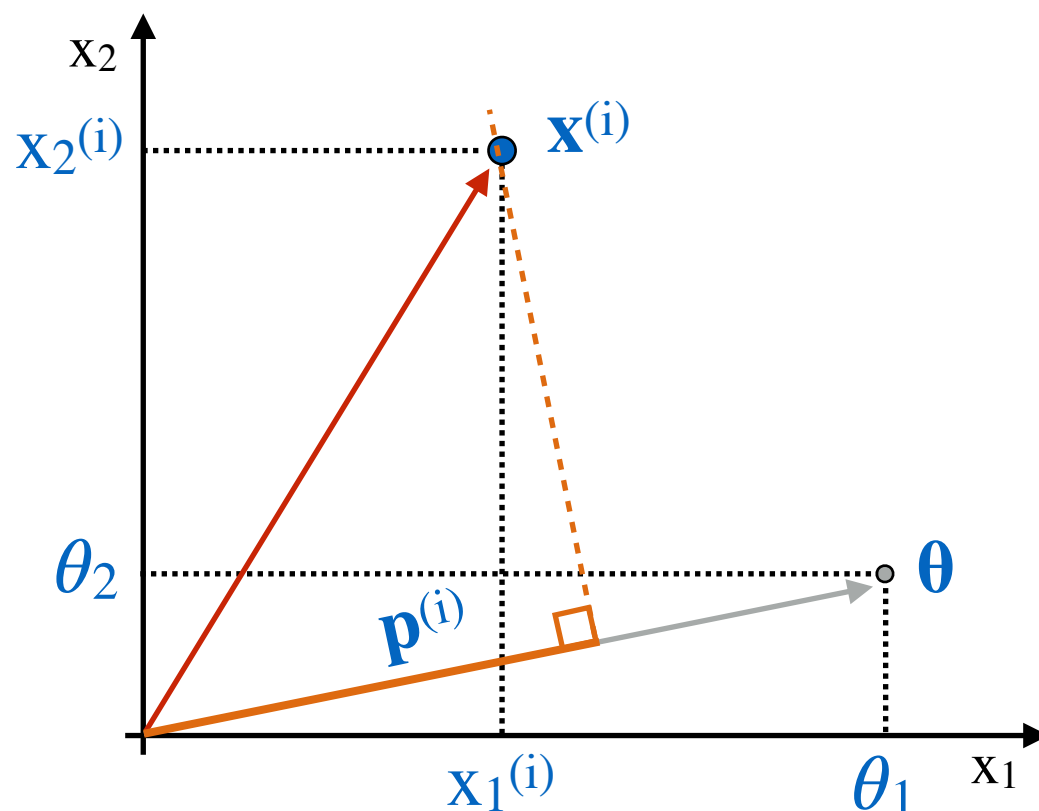
$cost_0(z)$

SVM: decision boundary

- Optimization objective:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\sqrt{(\theta_1^2 + \theta_2^2)} = \frac{1}{2}\|\boldsymbol{\theta}\|^2$$

s.t. $\boxed{\begin{array}{l} \boldsymbol{\theta}^T \mathbf{x}^{(i)} \geq 1 \\ \boldsymbol{\theta}^T \mathbf{x}^{(i)} \leq -1 \end{array}}$ $\begin{array}{l} \text{if } y^{(i)} = 1 \\ \text{if } y^{(i)} = 0 \end{array}$ $\Rightarrow$ s.t. $\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \geq 1 \quad \text{if } y^{(i)} = 1$

$\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \leq -1 \quad \text{if } y^{(i)} = 0$

$$\boldsymbol{\theta}^T \mathbf{x}^{(i)} = \boxed{\mathbf{p}^{(i)} \|\boldsymbol{\theta}\|}$$

Thus we can write our optimization objective w.r.t. $\mathbf{p}^{(i)}$

SVM: decision boundary
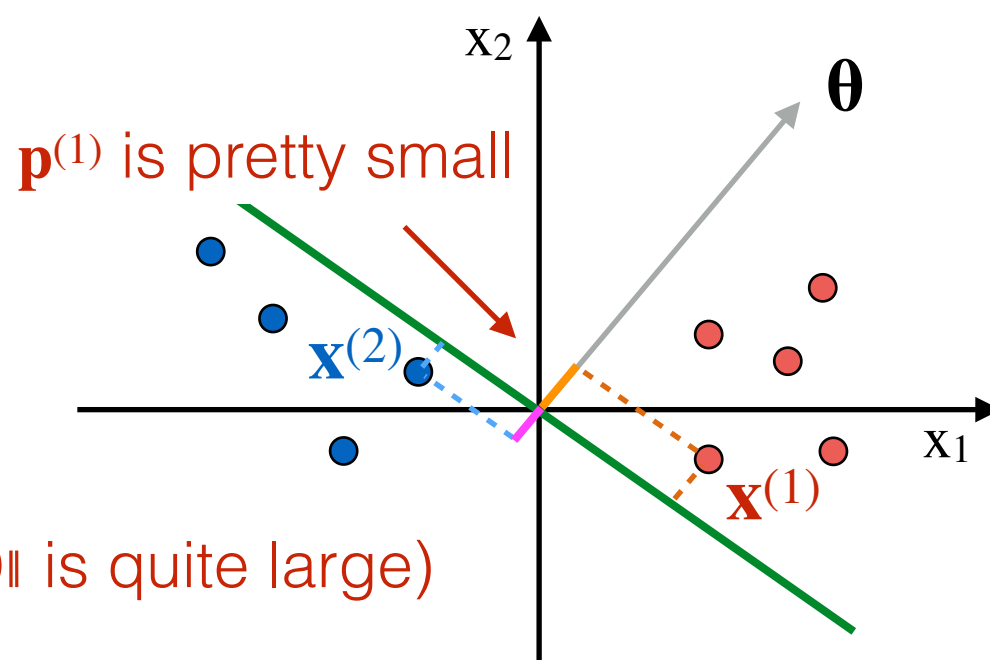
- Optimization objective:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\boldsymbol{\theta}\|^2$$

s.t. $\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \geq 1$ if $y^{(i)} = 1$

$\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \leq -1$ if $y^{(i)} = 0$

Where $\mathbf{p}^{(i)}$ is the projection of $\mathbf{x}^{(i)}$ onto the vector $\boldsymbol{\theta}$

‣ An example:

$\mathbf{p}^{(1)}$ is pretty small

$\mathbf{x}^{(2)}$

$\boldsymbol{\theta}$

x_2

x_1

$\mathbf{x}^{(1)}$

(therefore $\|\boldsymbol{\theta}\|$ is quite large)

SVM: decision boundary

• Optimization objective:

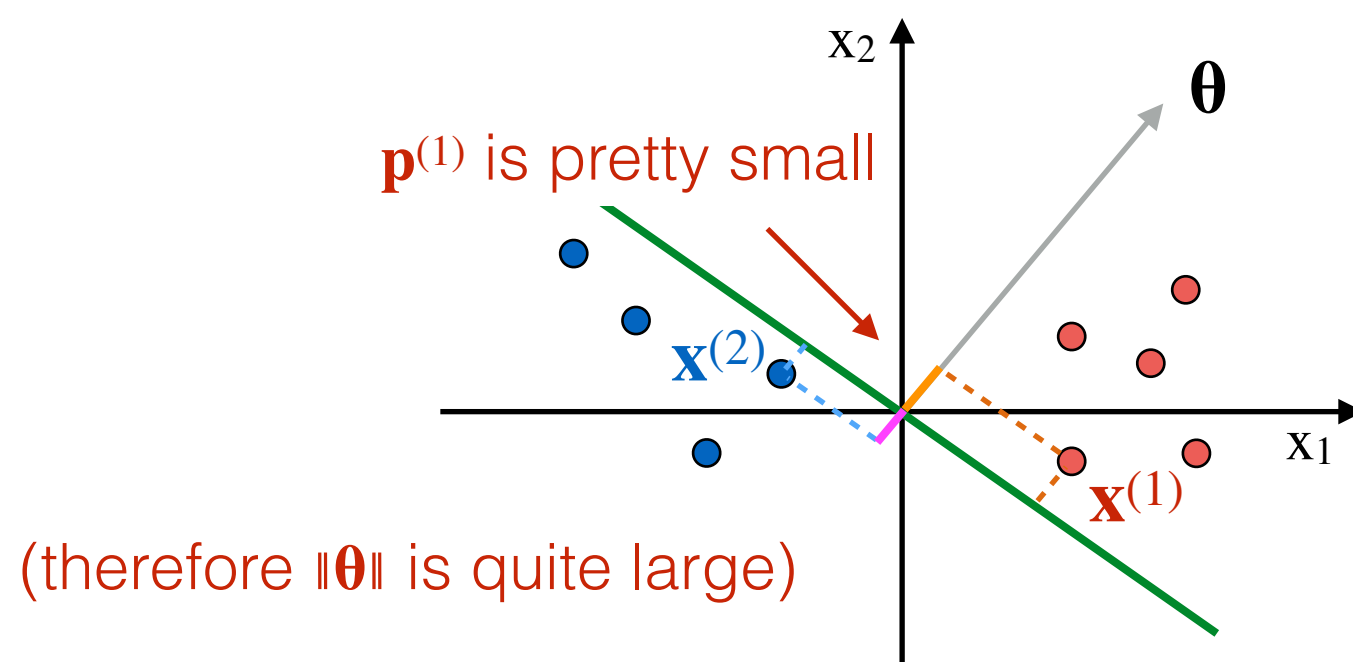$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\theta\|^2$$

s.t. $\mathbf{p}^{(i)} \|\theta\| \geq 1$ if $y^{(i)} = 1$
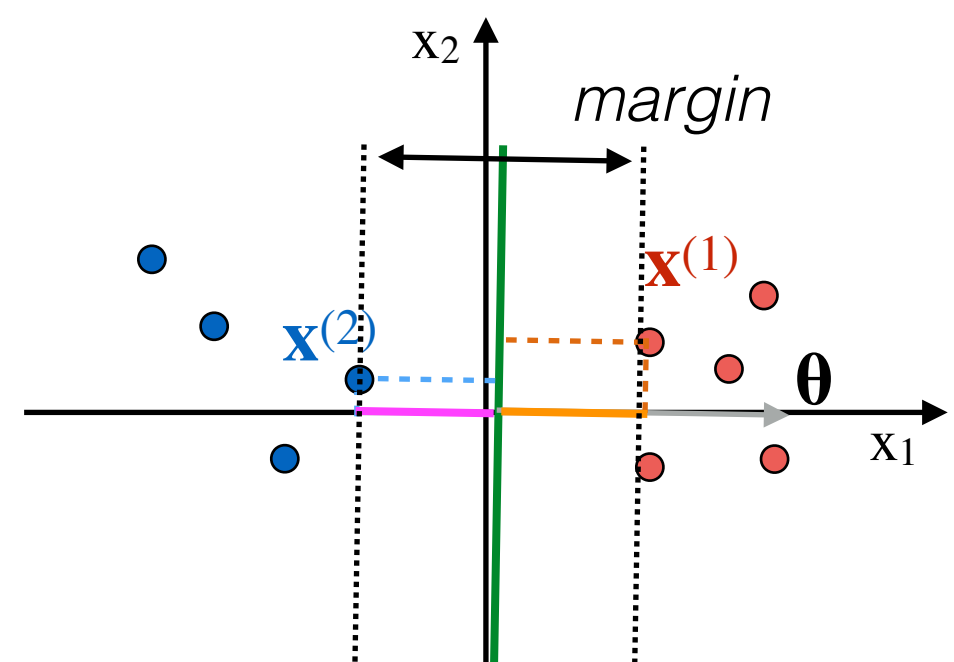
$\mathbf{p}^{(i)} \|\theta\| \leq -1$ if $y^{(i)} = 0$

Where $\mathbf{p}^{(i)}$ is the projection of $\mathbf{x}^{(i)}$ onto the vector θ

‣ An example:

SVM hypothesis

$\mathbf{p}^{(1)}$ is pretty small

$\mathbf{x}^{(2)}$

θ

x_2

x_1

$\mathbf{x}^{(1)}$

(therefore $\|\theta\|$ is quite large)

margin

x_2

$\mathbf{x}^{(2)}$

$\mathbf{x}^{(1)}$

θ

x_1

Adapted from slide by A. Ng

SVM: decision boundary

- Optimization objective:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\boldsymbol{\theta}\|^2$$

s.t. $\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \geq 1$ if $y^{(i)} = 1$

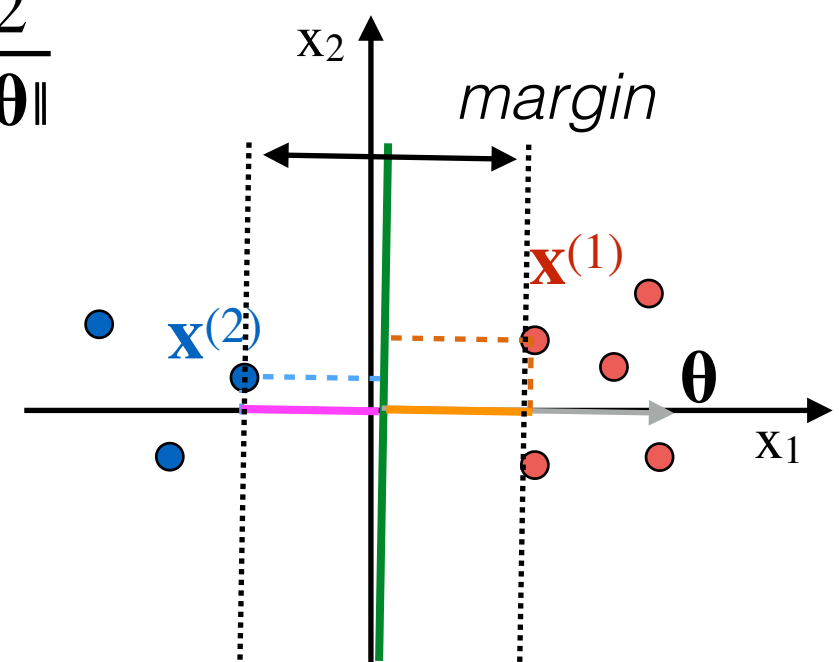$\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \leq -1$ if $y^{(i)} = 0$

Where $\mathbf{p}^{(i)}$ is the projection of $\mathbf{x}^{(i)}$ onto the vector $\boldsymbol{\theta}$

‣ Large-margin classification:

  ‣ Geometrically, the margin is: $M = \frac{2}{\|\boldsymbol{\theta}\|}$

  ‣ A key consequence is that the max-margin hyperplane is defined by those $\mathbf{x}^{(i)}$ that lie nearest to it

  (support vectors)
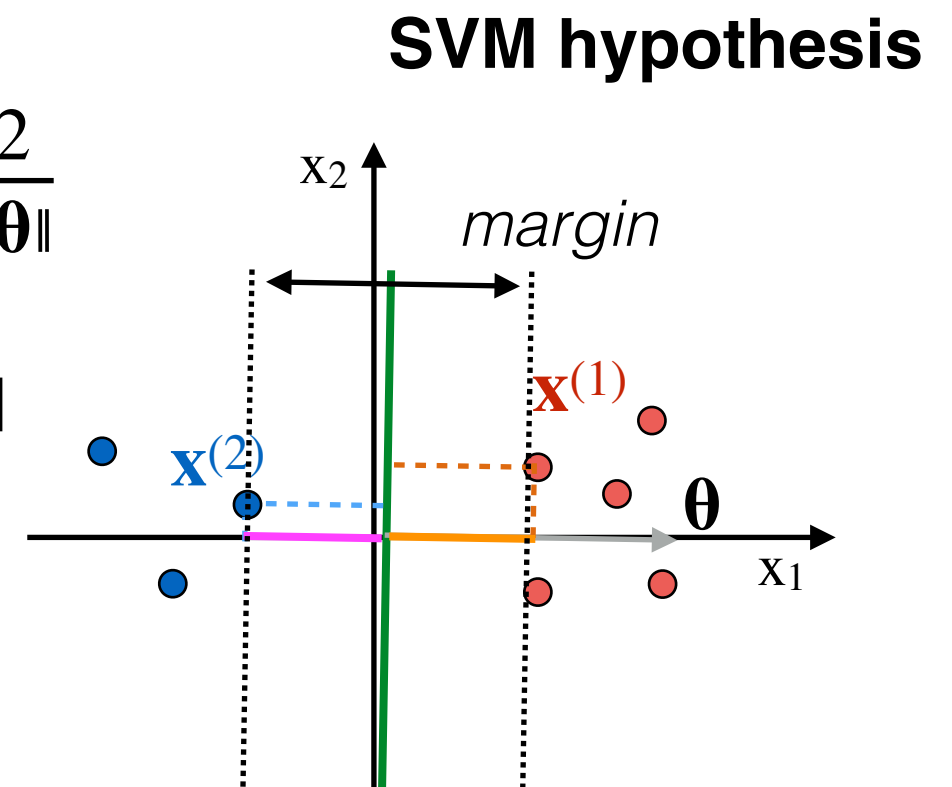
SVM hypothesis

SVM: decision boundary

- Optimization objective:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\boldsymbol{\theta}\|^2$$

s.t. $\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \geq 1$ if $y^{(i)} = 1$ *In SVM usually we use a different notation*

$\mathbf{p}^{(i)} \|\boldsymbol{\theta}\| \leq -1$ if $y^{(i)} = -1$

‣ Large-margin classification:

SVM hypothesis

    ‣ Geometrically, the margin is: $M = \dfrac{2}{\|\boldsymbol{\theta}\|}$

    ‣ A key consequence is that the max-margin hyperplane is defined by those $\mathbf{x}^{(i)}$ that lie nearest to it

(support vectors)

SVM: decision boundary

• Optimization objective:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\boldsymbol{\theta}\|^2$$

This is called the *primal formulation* of Support Vector Machines

s.t. $\mathbf{y}^{(i)} (\mathbf{p}^{(i)} \|\boldsymbol{\theta}\|) \geq 1$

In SVM usually we use a different notation (this simplify a little bit our formulation)
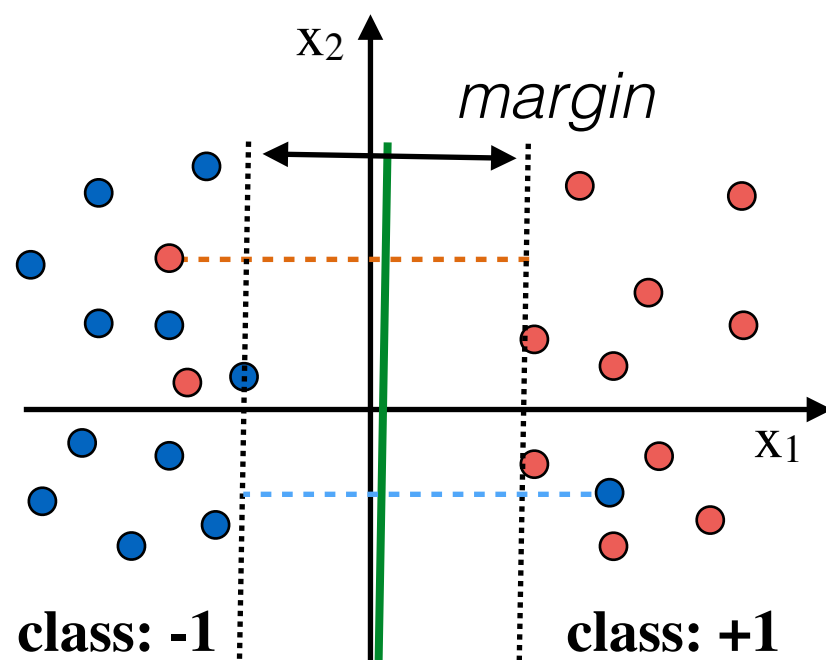
‣ Large-margin classification:

   ‣ Geometrically, the margin is: $M = \dfrac{2}{\|\boldsymbol{\theta}\|}$

   ‣ A key consequence is that the max-margin hyperplane is defined by those $\mathbf{x}^{(i)}$ that lie nearest to it

(support vectors)

SVM hypothesis
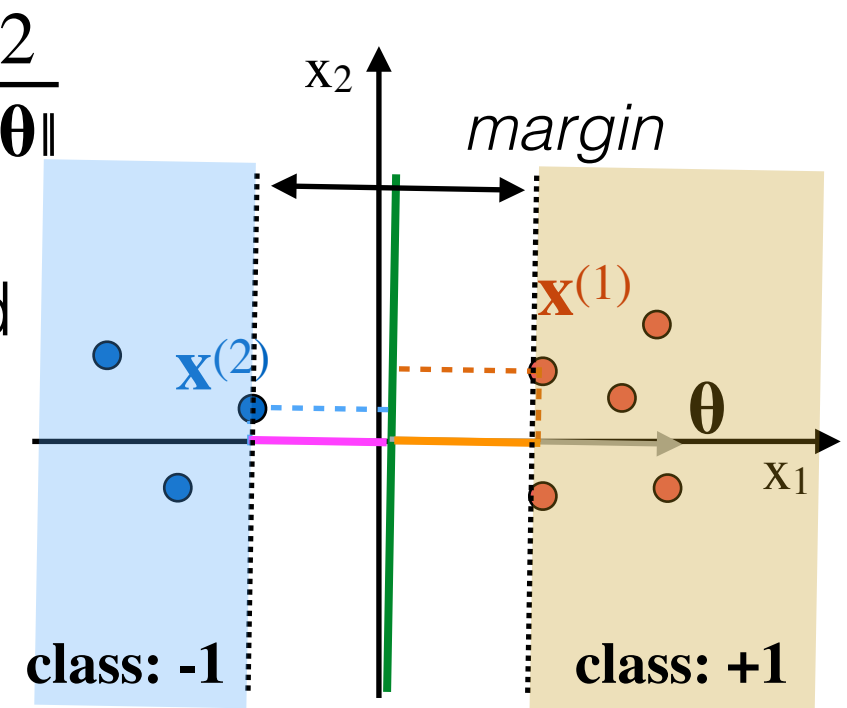
What if data is not linearly separable?

- What we have seen until know is called *hard-margin*

- What about examples that lies on the wrong side?

class: -1   class: +1

α trades off training error

vs model complexity

‣ Introduce the slack variable $\xi^{(i)}$

‣ New optimization objective:

$$\min_{\theta} \ \frac{1}{2} \ \|\theta\|^2 + \alpha \sum_{i=1}^{m} \xi^{(i)}$$

$$\text{s.t.} \ \ \xi^{(i)} \geq 0 \ , \ \forall i: \ y^{(i)}(p^{(i)} \|\theta\|) \geq 1 - \xi^{(i)}$$

‣ $\sum_{i=1}^{m} \xi^{(i)}$ bounds num. of training errors

‣ This is called *soft-margin* extension

Hinge Loss and soft-margins

- The soft-margin extension leads to the definition of the hinge loss (that we have previously seen)

  ‣ Hinge loss: $\ell(h_\theta(\mathbf{x}^{(i)}), y^{(i)}) = \max(0, 1 - y^{(i)}(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$

    where $\boldsymbol{\theta}^T \mathbf{x}^{(i)} = \mathbf{p}^{(i)} \|\boldsymbol{\theta}\|$, i.e. $\mathbf{p}^{(i)}$ is the projection of $\mathbf{x}^{(i)}$ onto the vector $\boldsymbol{\theta}$

  ‣ The hinge loss $\ell(\cdot)$, is a convex function and can be optimized using gradient descent

  ‣ Interpretation:

    ‣ When $y^{(i)}$ and $\mathbf{p}^{(i)}$ have the same sign and $|y^{(i)}| \geq 1$, $\ell(\cdot)=0$

    ‣ When they have opposite sign, ℓ increases linearly with $y^{(i)}$

    ‣ Similarly, if $|y^{(i)}| < 1$ (it has the same sign: correct prediction but not by enough margin) ℓ increases linearly with $y^{(i)}$

Non-linear Classification

- The *soft-margin* extension is great, but SVM is still a linear classification model

- However, SVM can efficiently perform non-linear classification using the so-called "kernel trick"

  ‣ SVM is the most popular example of a class of methods / algorithms called **kernel machines** (or kernel methods)

  ‣ Kernel methods use kernel functions (i.e. similarity functions over pairs of data samples) which enable them to operate in high-dimensional "implicit" feature spaces

Non-linear Classification

- From feature combination to kernels:

  ‣ If you remember, we already introduced a trick to "extend" linear models by introducing feature transformations
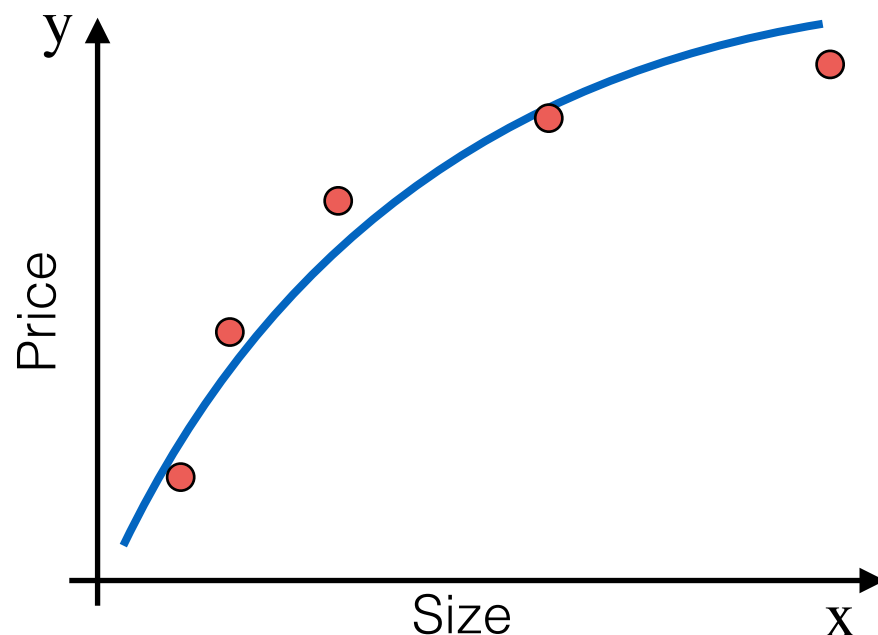
Features and Polynomial Regression

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

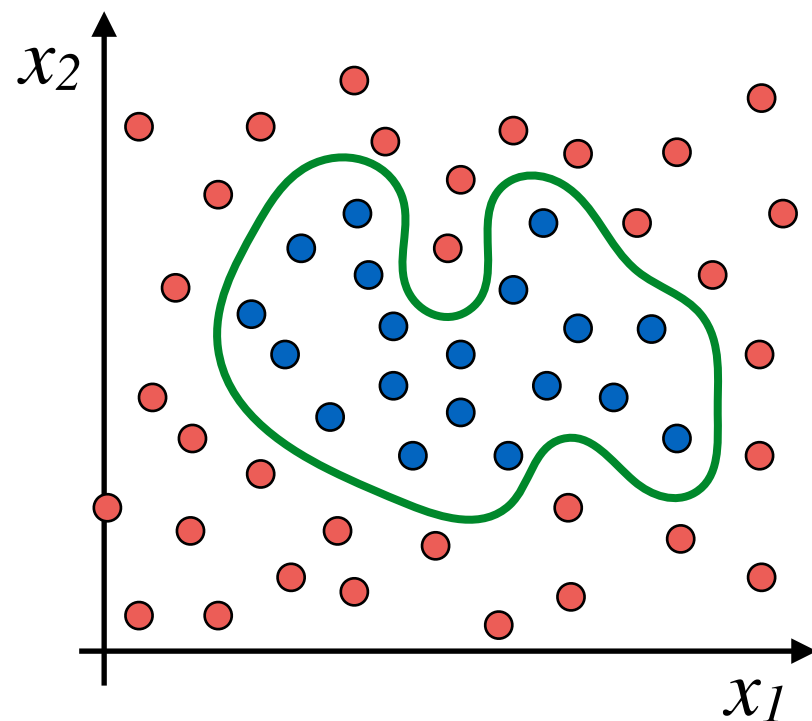How to get this $h_\theta(x)$ from our linear regression model $\boldsymbol{\theta}^T \mathbf{x}$?

$$h_\theta(x) = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}^T \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$

x_1: size of house

x_2: (size of house)^2

Non-linear Classification

- We introduce a trick to adapt linear regression (or classification) models to non-linear data:

Negative class

Positive class

Predict $y = 1$ if:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \ldots \geq 0$$

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
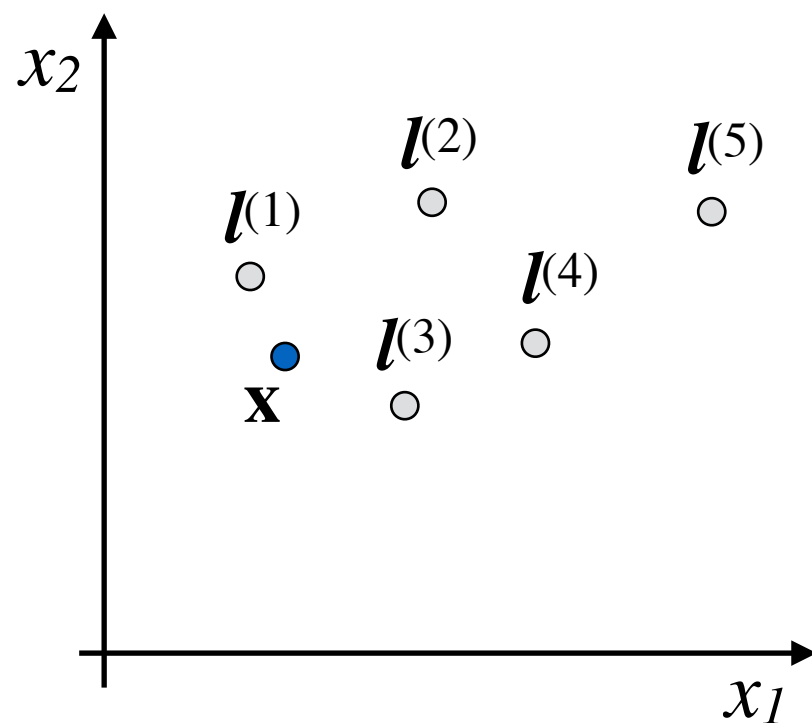
$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \theta_5 f_5 + \ldots$$

where $f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, $f_4 = x_1^2$, $f_5 = x_2^2$, ...

How to choose features f_1, f_2, ... ?

Kernels - Intuition

- We introduce a trick to adapt linear regression (or classification) models to non-linear data:

Negative class

Positive class

Given $\mathbf{x}$, compute features $\mathbf{f}$ depending on proximity to landmarks $l^{(1)}, l^{(2)}, \ldots$

$$f_1 = \text{similarity}(\mathbf{x}, l^{(1)})$$

$$f_2 = \text{similarity}(\mathbf{x}, l^{(2)})$$
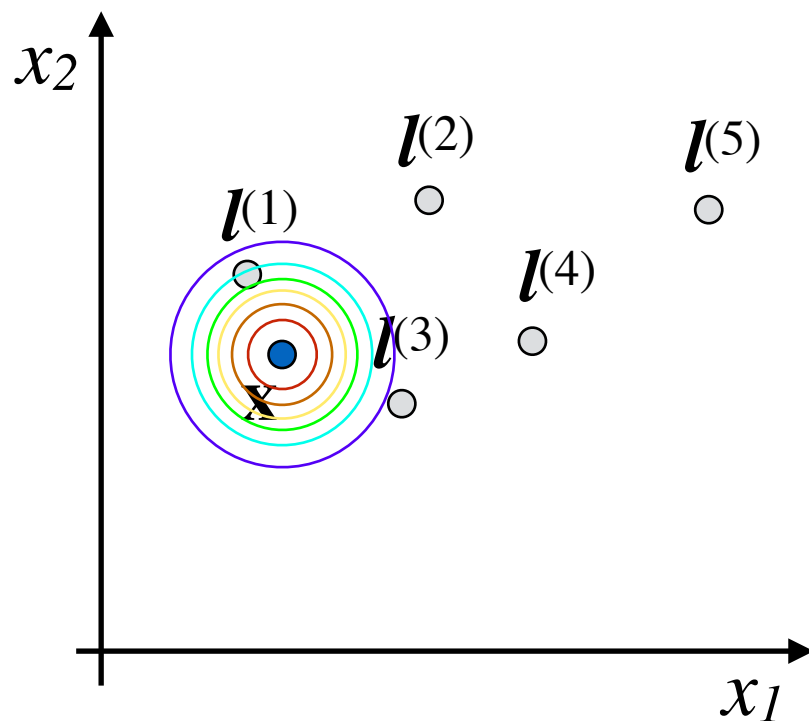
$$\vdots$$

Predict $y = 1$ if:

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \theta_5 f_5 + \ldots \geq 0$$

where $f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, $f_4 = x_1^2$, $f_5 = x_2^2$, $\ldots$

Kernels - Intuition

- We introduce a trick to adapt linear regression (or classification) models to non-linear data:

Given $\mathbf{x}$, compute features $\mathbf{f}$ depending on proximity to landmarks $\mathit{l}^{(1)}, \mathit{l}^{(2)}, \dots$

x_2

$\mathit{l}^{(2)}$ $\mathit{l}^{(5)}$

$\mathit{l}^{(1)}$

$\mathit{l}^{(4)}$

$\mathit{l}^{(3)}$

$\mathbf{x}$

x_1
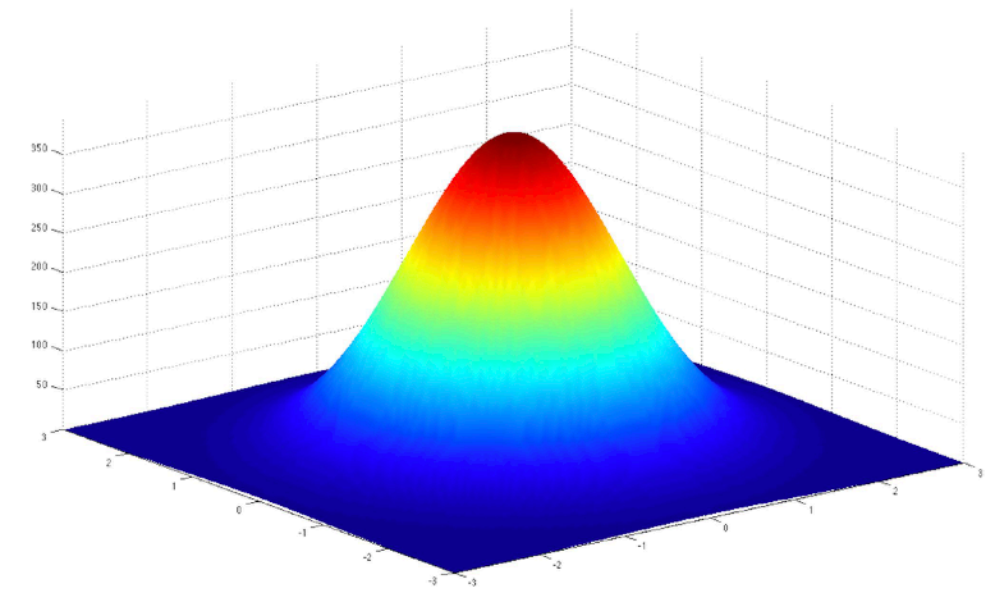
$f_1 = k(\mathbf{x}, \mathit{l}^{(1)})$

$f_2 = k(\mathbf{x}, \mathit{l}^{(2)})$

$f_3 = k(\mathbf{x}, \mathit{l}^{(3)})$

$f_4 = k(\mathbf{x}, \mathit{l}^{(4)})$

$f_5 = k(\mathbf{x}, \mathit{l}^{(5)})$

if $\mathbf{x} \approx \mathit{l}^{(i)}$: $f_1 = \exp(- 0^2/(2\sigma^2)) \approx 1$

if $\mathbf{x}$ is far from $\mathit{l}^{(i)}$: $f_1 \approx 0$
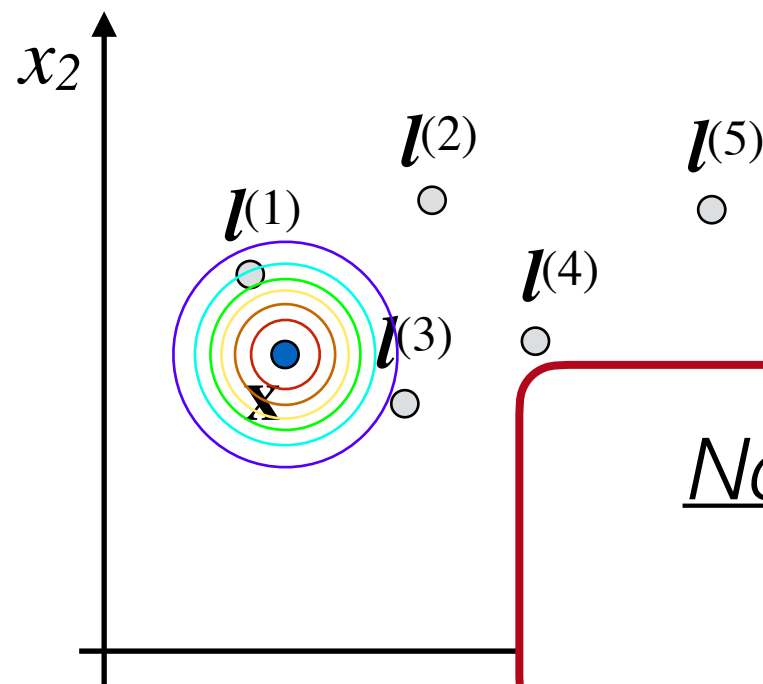
e.g. $k(\mathbf{x}, \mathit{l}^{(i)}) = \exp(- \dfrac{\|\mathbf{x} - \mathit{l}^{(i)}\|^2}{2\sigma^2})$

Gaussian Kernel

Kernels - Intuition

- We introduce a trick to adapt linear regression (or classification) models to non-linear data:

Given $\mathbf{x}$, compute features $\mathbf{f}$ depending on proximity to landmarks $\boldsymbol{l}^{(1)}, \boldsymbol{l}^{(2)}, \dots$

$$f_1 = k(\mathbf{x}, \boldsymbol{l}^{(1)})$$

$$f_2 = k(\mathbf{x}, \boldsymbol{l}^{(2)})$$

Note: this is the popular RBF kernel, i.e.

$$k(\mathbf{x}, \boldsymbol{l}^{(i)}) = \exp(-\gamma \, \|\mathbf{x} - \boldsymbol{l}^{(i)}\|^2)$$

if $\mathbf{x} \approx \boldsymbol{l}^{(i)}$: $f_1 = \exp(-0^2/(2\sigma^2)) \approx 1$

if $\mathbf{x}$ is far from $\boldsymbol{l}^{(i)}$: $f_1 \approx 0$

e.g. $k(\mathbf{x}, \boldsymbol{l}^{(i)}) = \exp(- \dfrac{\|\mathbf{x} - \boldsymbol{l}^{(i)}\|^2}{2\sigma^2})$ *Gaussian Kernel*
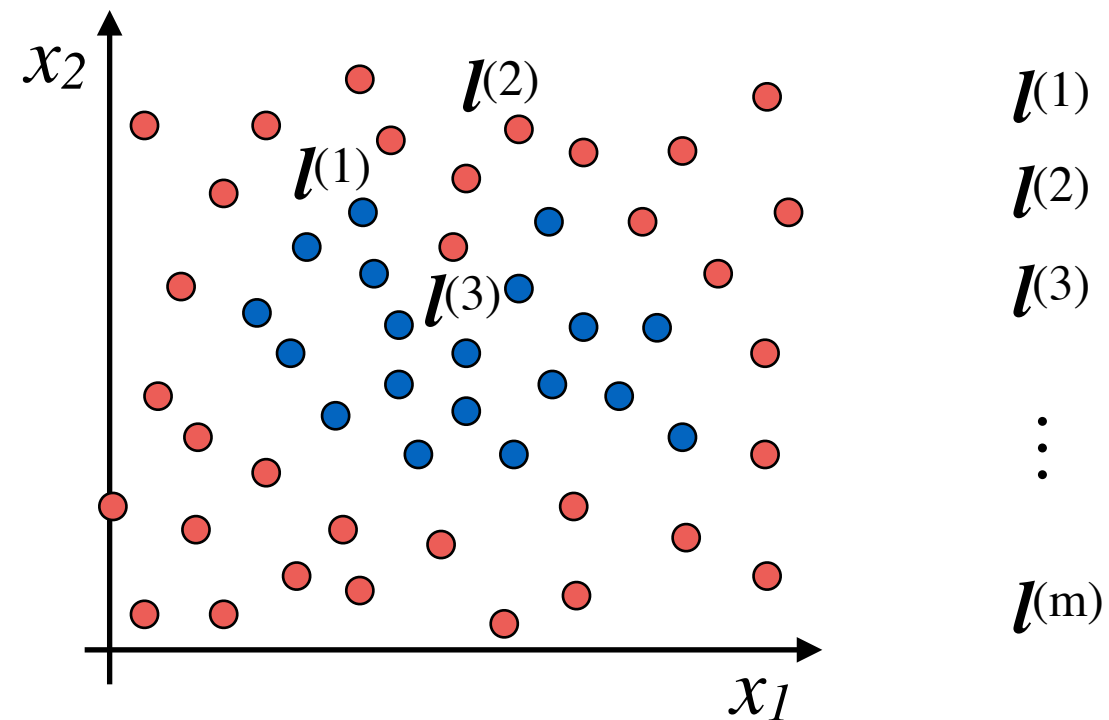
Kernels - Intuition

- We introduce a trick to adapt linear regression (or classification) models to non-linear data:

Where to get the landmarks $l^{(1)}, l^{(2)}, \ldots$?
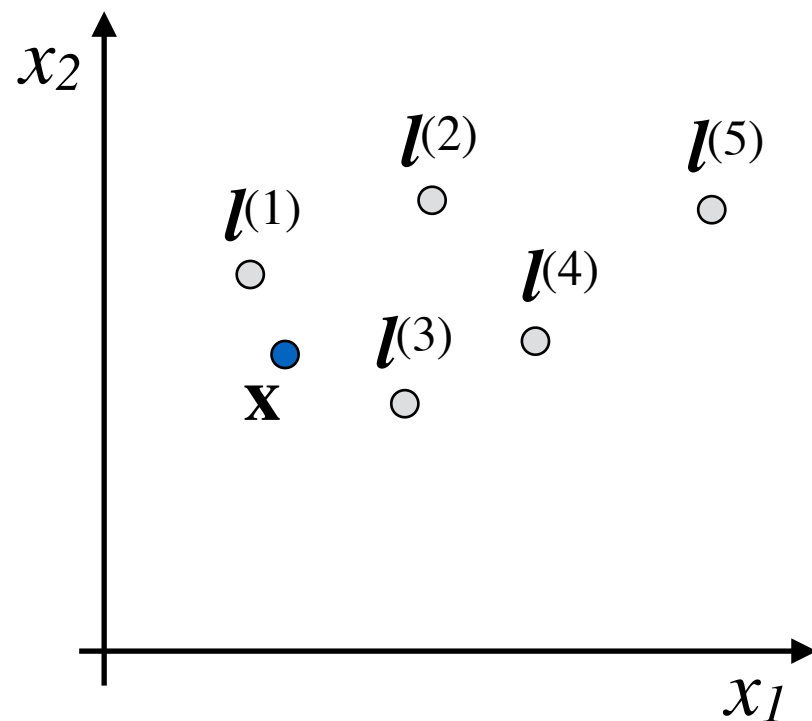
- $\bullet$ Negative class
- $\bullet$ Positive class

Kernel Methods

- Goal: transform/map raw data into feature vectors

  ‣ Instead of using explicit user-specified feature maps, rely on a *similarity function* (kernel) over pairs of data points

  ‣ A kernel is a function $\varphi\colon \mathbb{R}^n \to \mathbb{R}^m$ which maps $x \to \varphi(x)$

  ‣ Then, given this mapping, we try to find a linear decision boundary in the feature space

Input Space

$$x \to \varphi(x)$$

Feature Space
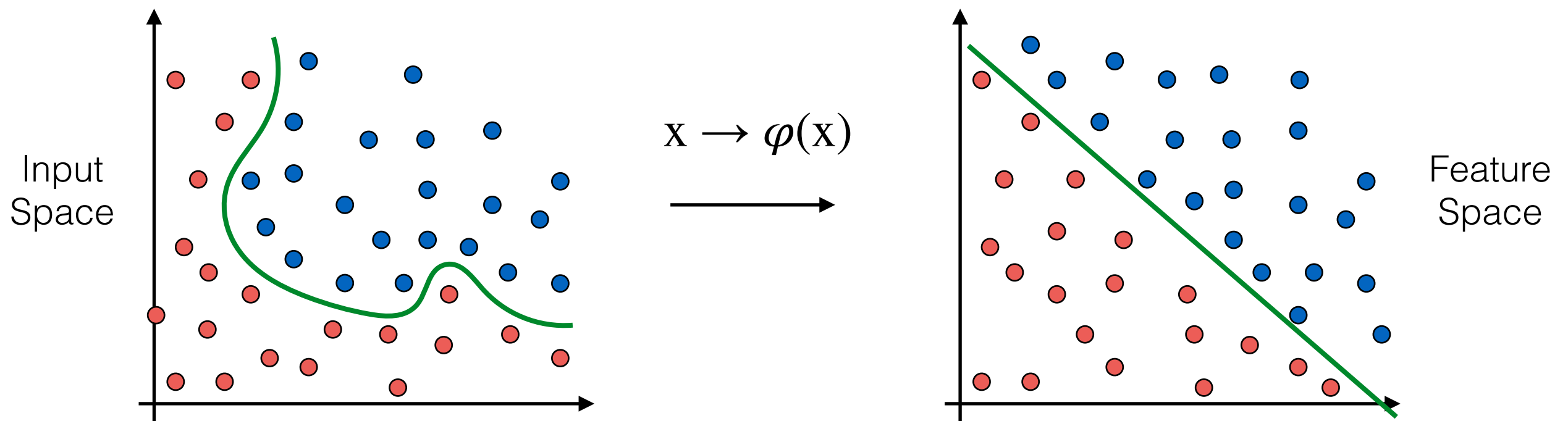
Kernel Methods

- Summarizing what we have seen till now:

  ‣ A kernel is a function $\varphi \colon \mathbb{R}^n \to \mathbb{R}^m$ which maps our vectors in $\mathbb{R}^n$ to some (possibly very high dimensional) space $\mathbb{R}^m$

  ‣ Kernels are sometimes called "generalized dot product":

  $$\mathbf{x}, \mathbf{y} \in \mathbb{R}^n \qquad k(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y}) \quad \text{corresponds to a dot product in } \mathbb{R}^n$$

  Kernels give a way to compute dot products in some feature space without even knowing what this space is and what is φ

Kernel Methods

- Let's see an example (polynomial kernel):

$\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ $k(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^{\mathrm{T}}\varphi(\mathbf{y})$ corresponds to a dot product in $\mathbb{R}^n$

$k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^{\mathrm{T}}\mathbf{y})^2$ assuming $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$, i.e. $\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2)$

$k(\mathbf{x}, \mathbf{y}) = (1 + x_1 y_1 + x_2 y_2)^2 = 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2$

Note that this is nothing else but a dot product between vectors:

$\varphi(\mathbf{x}) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$, $\varphi(\mathbf{y}) = (1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2)$

Contact

- **Office:** Torre Archimede 6CD, room 622

- **Office hours** (ricevimento)**:** Friday 11:00-13:00

✉ lamberto.ballan@unipd.it

🏠 http://www.lambertoballan.net

🏠 http://vimp.math.unipd.it

@ twitter.com/lambertoballan