

session 1:  
**Basic Predictive Models**

M. Kundegorski

13<sup>th</sup> December 2019

**IAFIG-RMS - Bioimage Analysis With Python**  
Cambridge Bioinformatics Training Centre



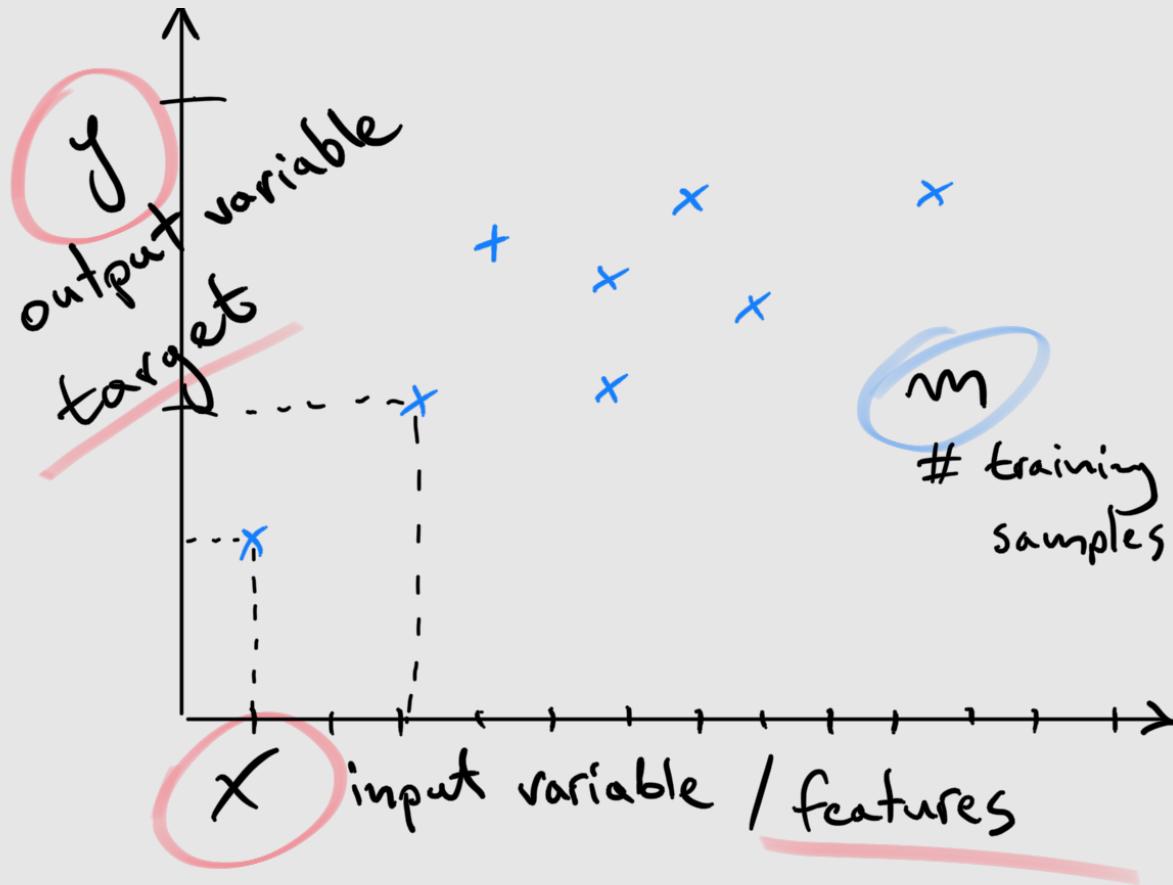
# Basic Models

*Statistics from a different point of view*

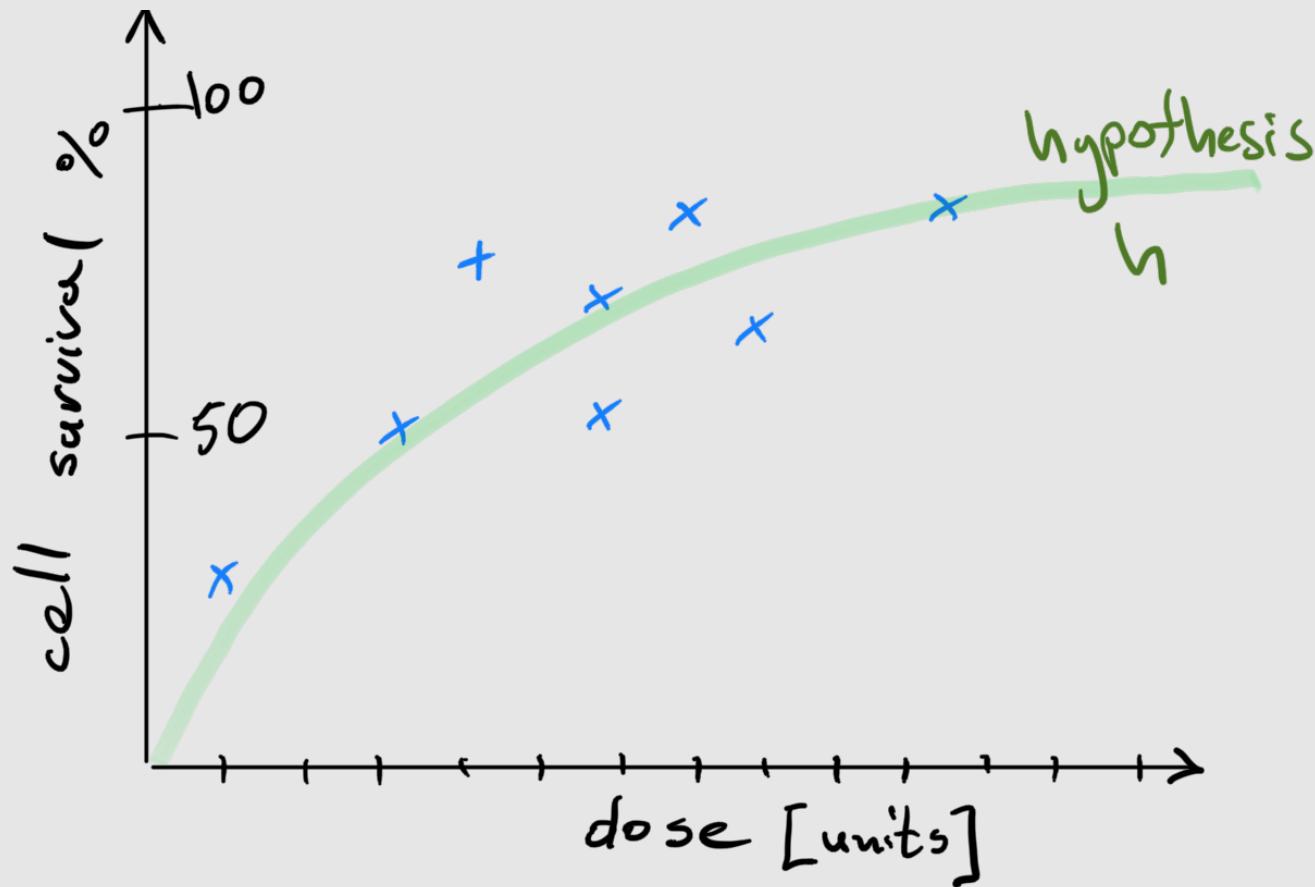
# In this lecture

- Linear regression
- Multivariate regression
- Generalised linear regression
- Model fitting

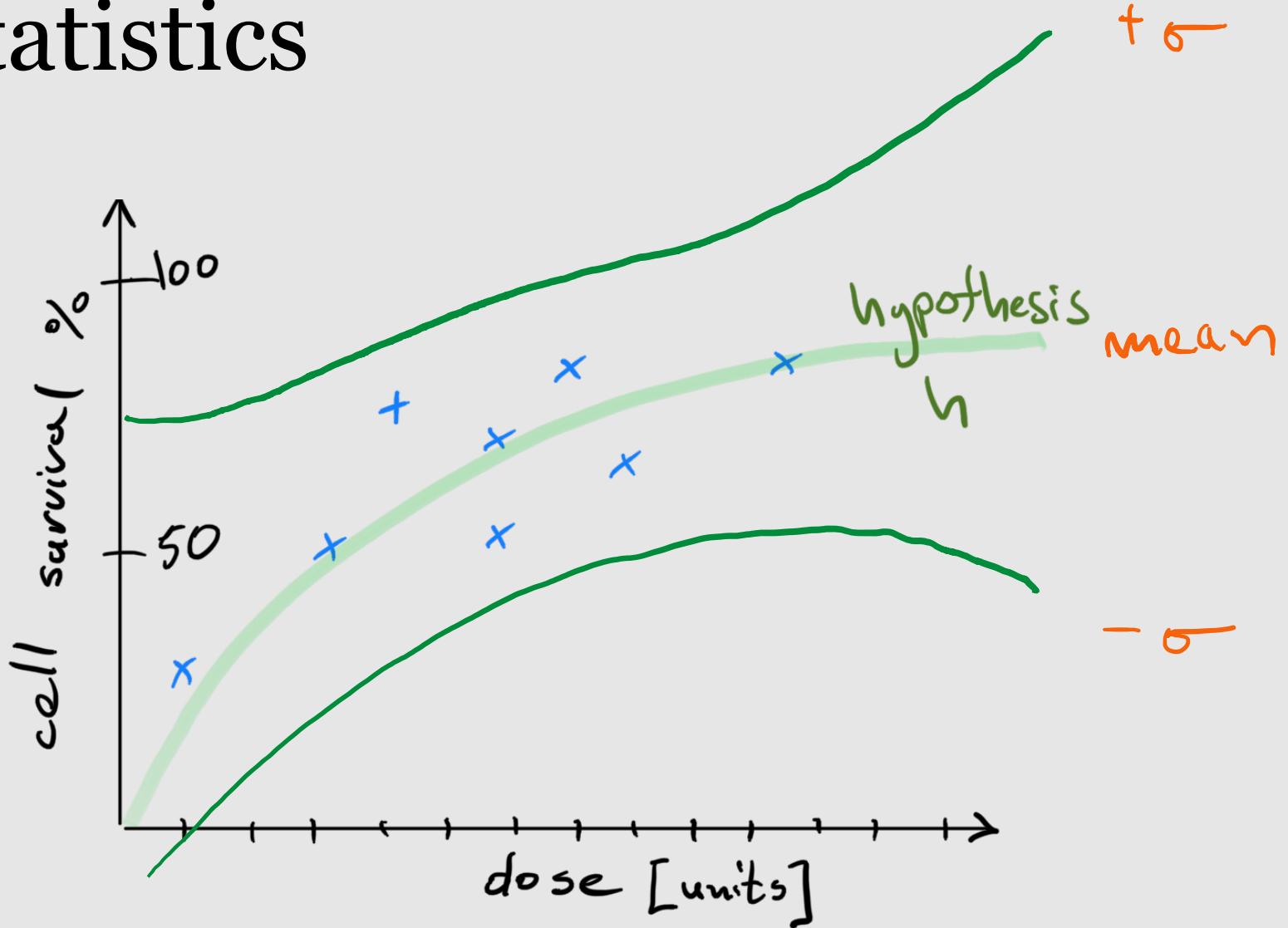
# A problem



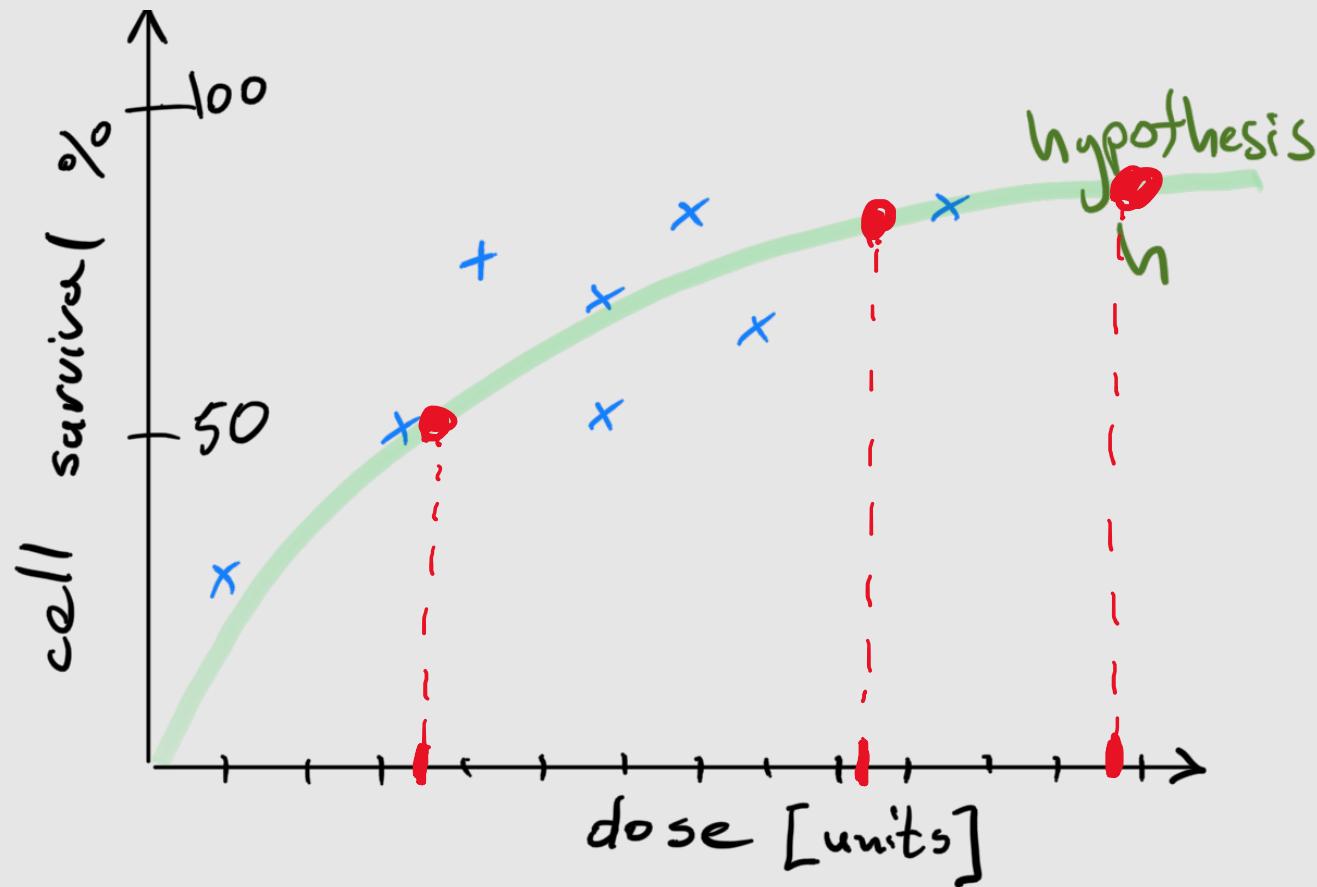
# Model Representation



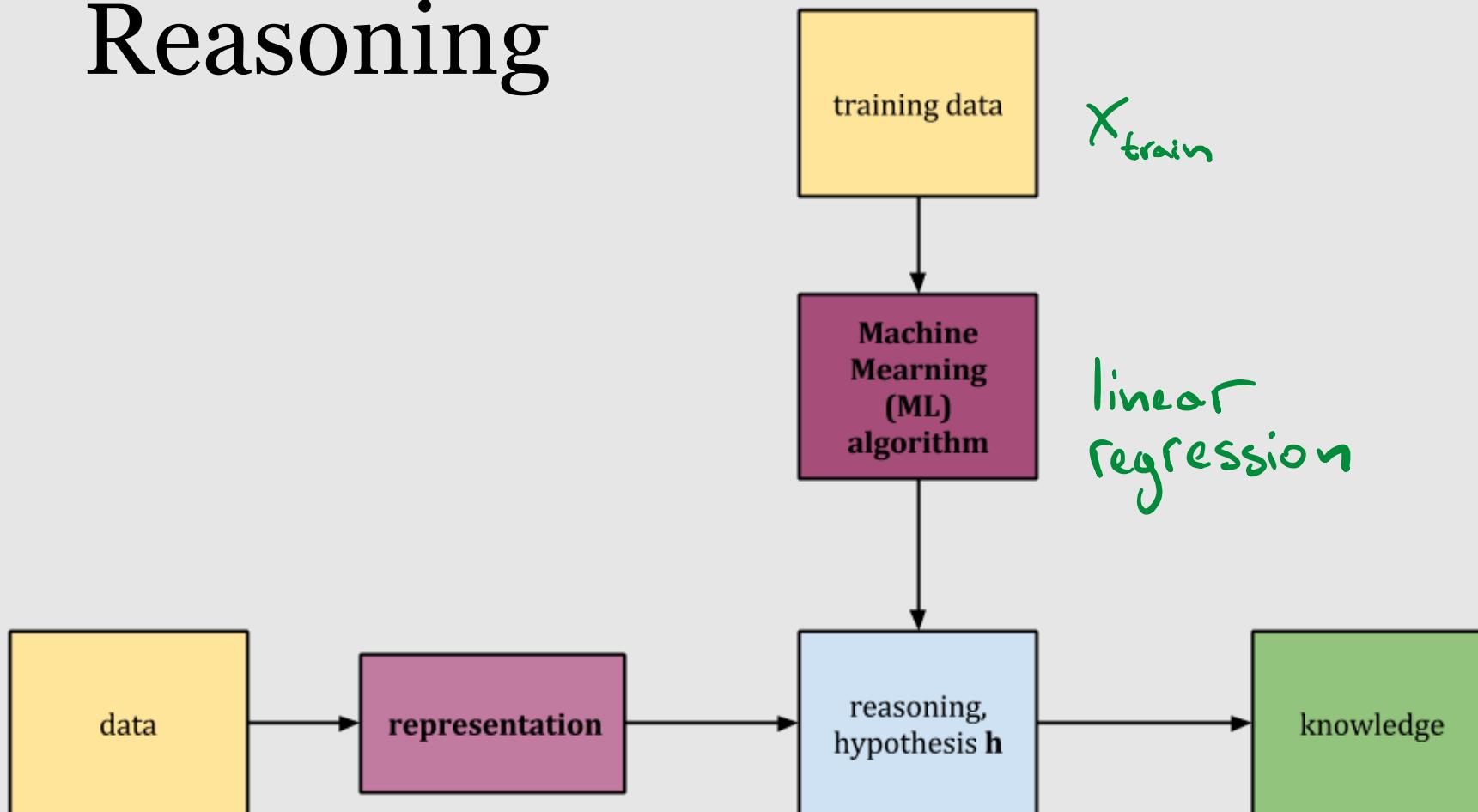
# Statistics



# Machine learning

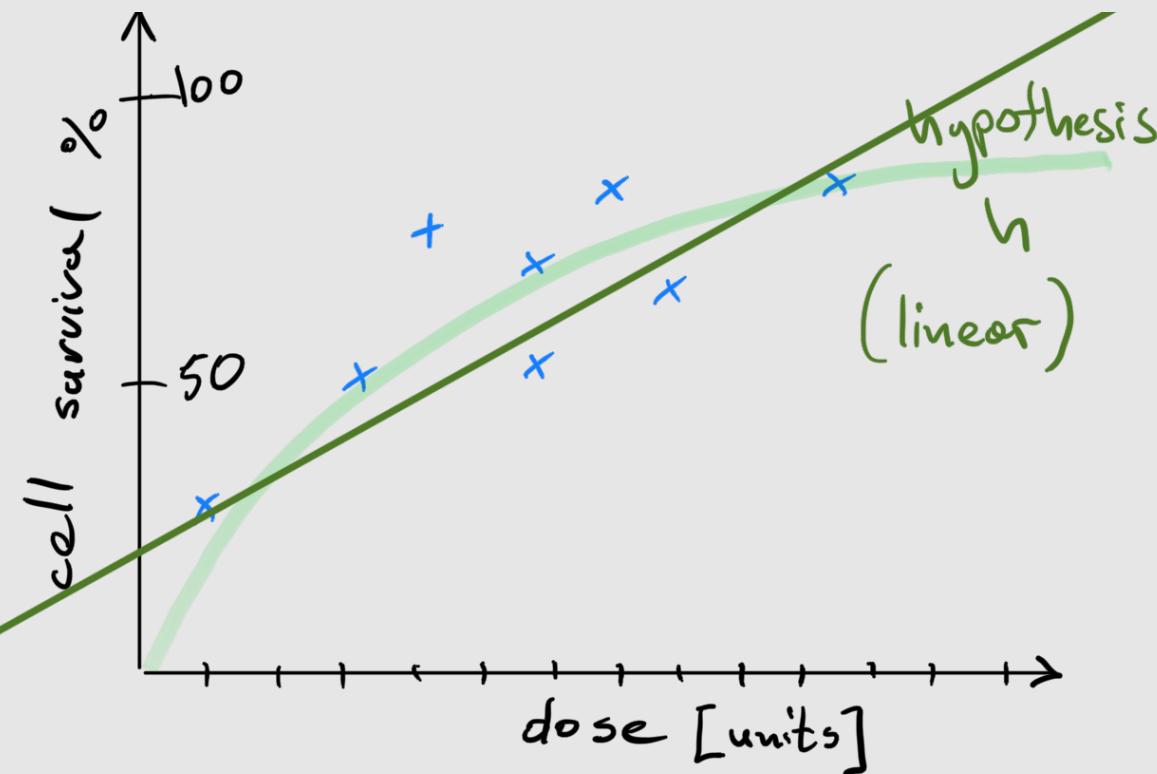


# Reasoning



$$X_? \rightarrow h_{\theta}(x) = \theta_0 + \theta_1 x \rightarrow y_{predicted}$$

# Linear regression

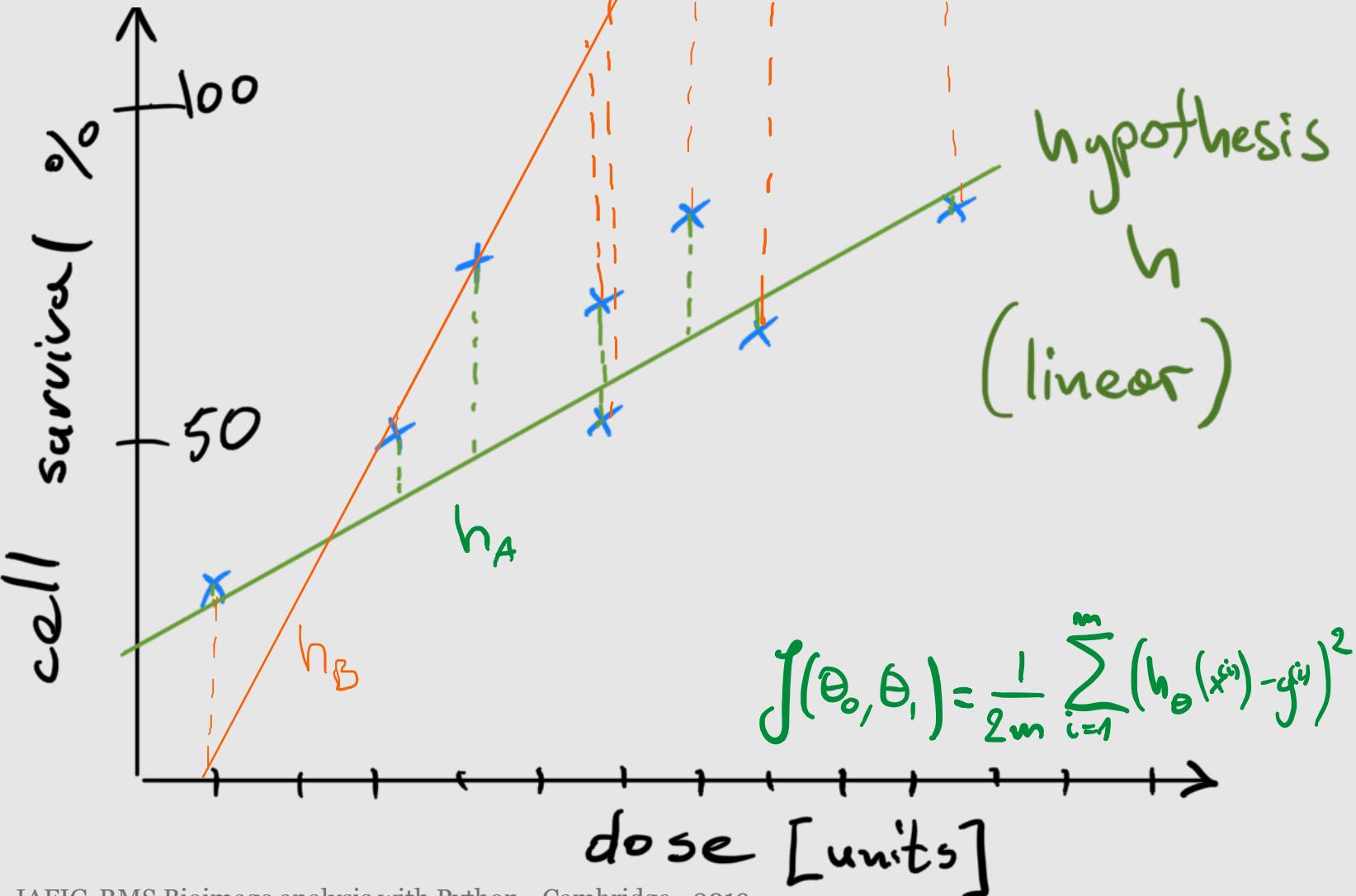


$$\frac{1}{2m} \sum_{i=0}^m (h_0(x_i) - y_i)^2$$

Squared error cost function.

We have  $m$  training examples  
 $(x_i, y_i)$

# Fitting error



# Cost Function

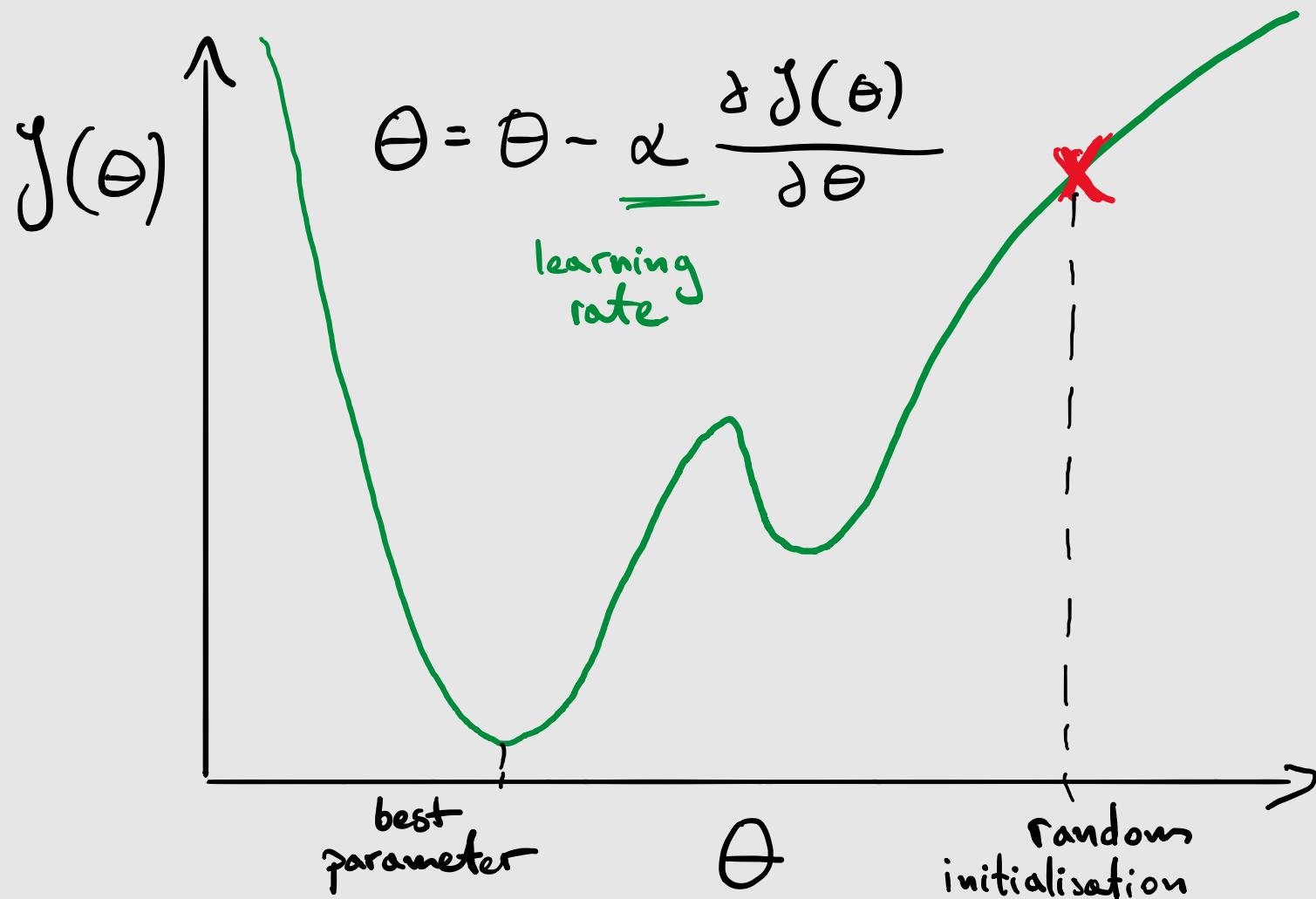
Hypothesis:  $h_{\theta}(x) = \theta_0 + \theta_1 x$

Parameters:  $\theta_0, \theta_1$

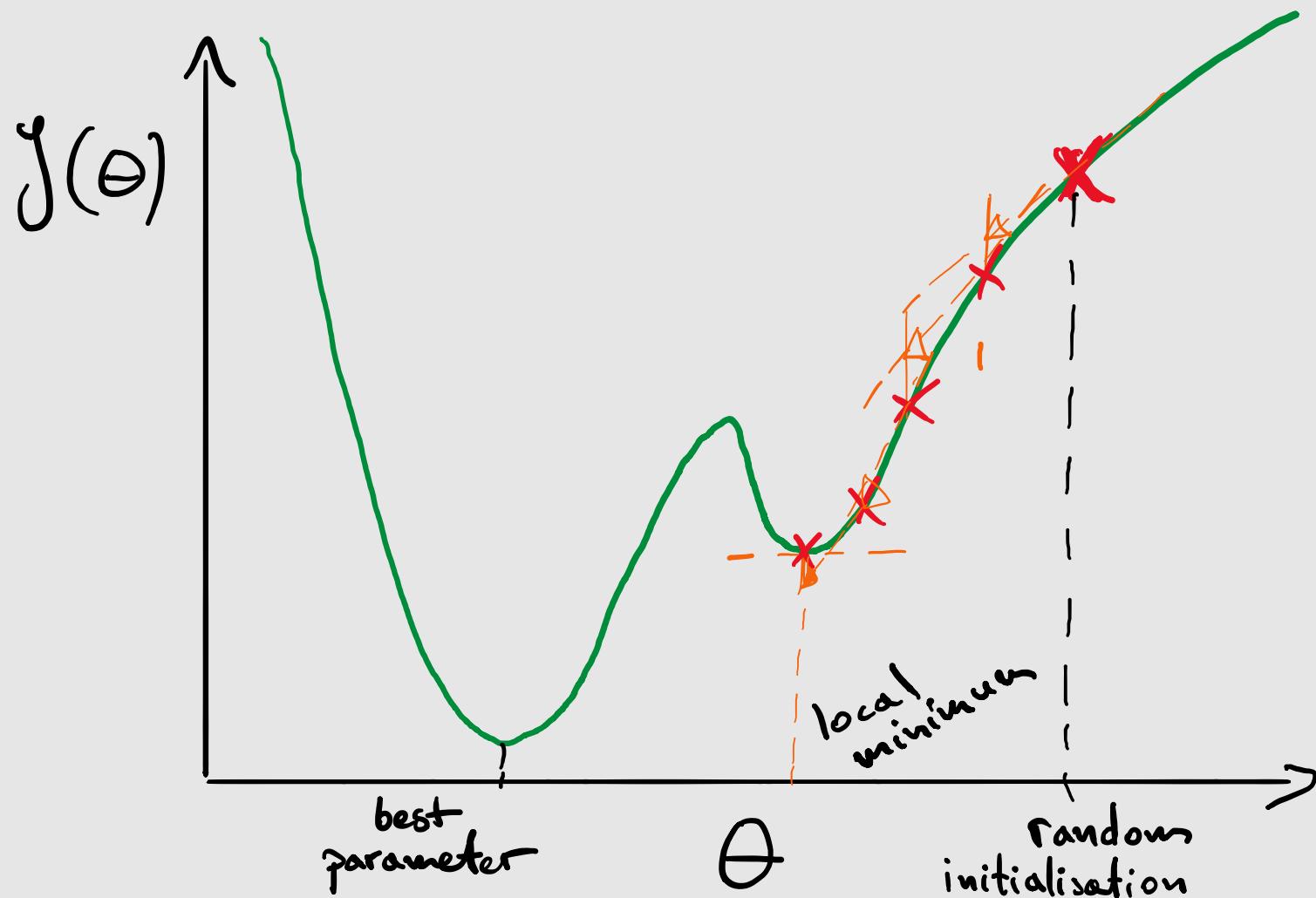
Cost function:  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

Goal:  $\underset{\theta_0, \theta_1}{\operatorname{arg\,min}} J(\theta_0, \theta_1)$

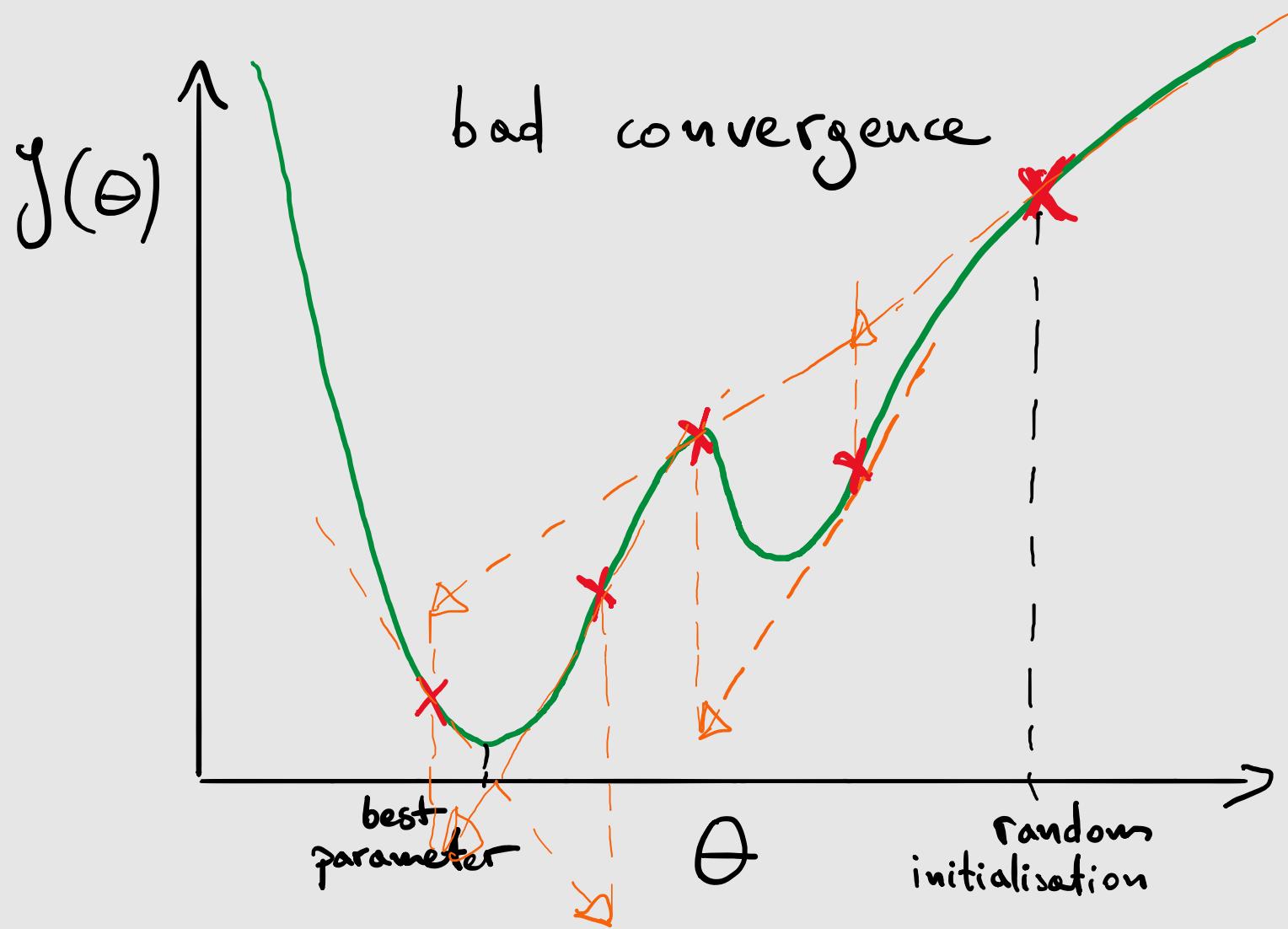
# Gradient descent



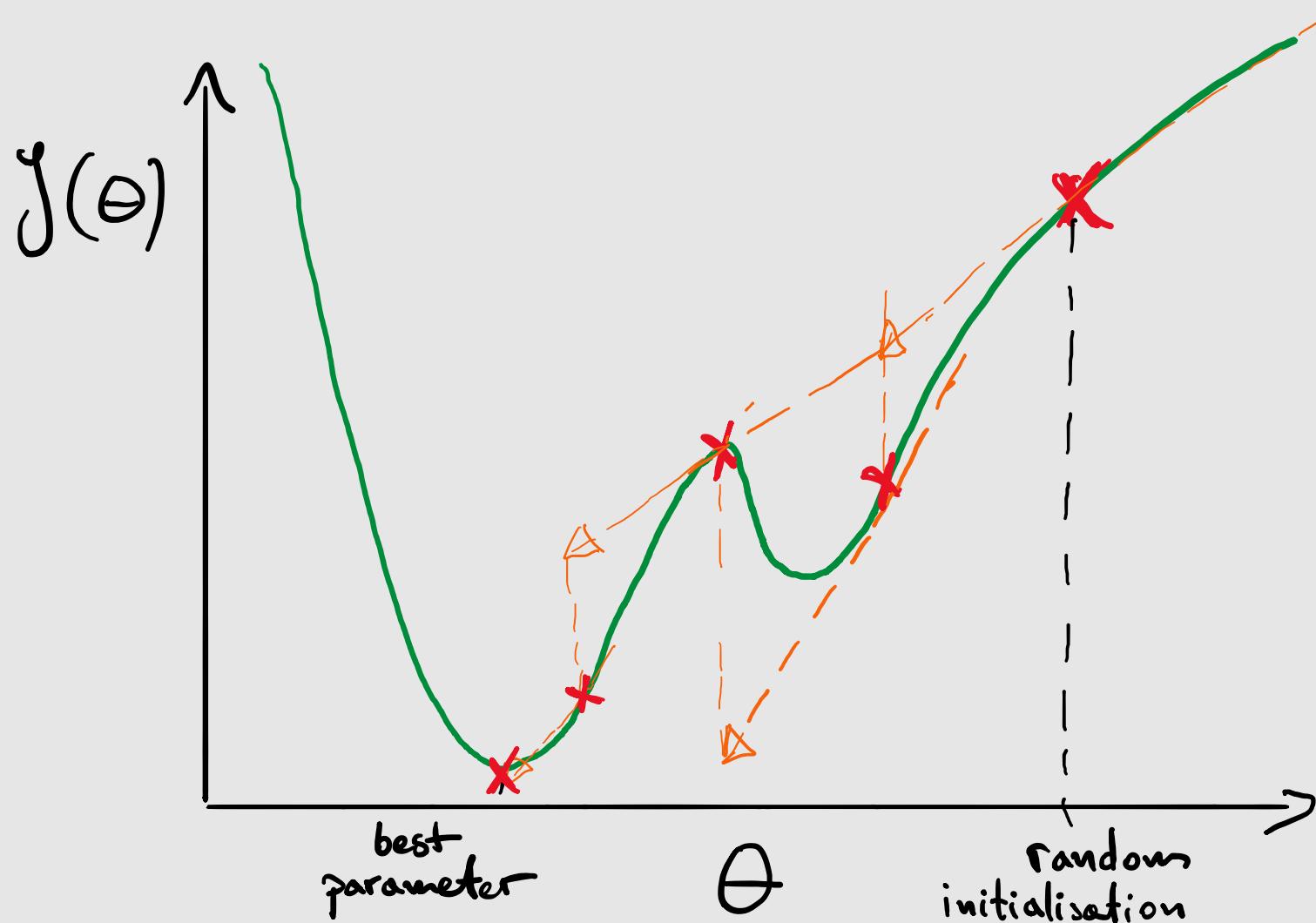
# Gradient descent (small $\alpha$ )



# Gradient descent (big $\alpha$ )



# Gradient descent (adaptive $\alpha$ )



# Gradient descent

- all parameters simultaneous
- batch: uses a "batch" of training examples

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x_i) - y_i)^2$$

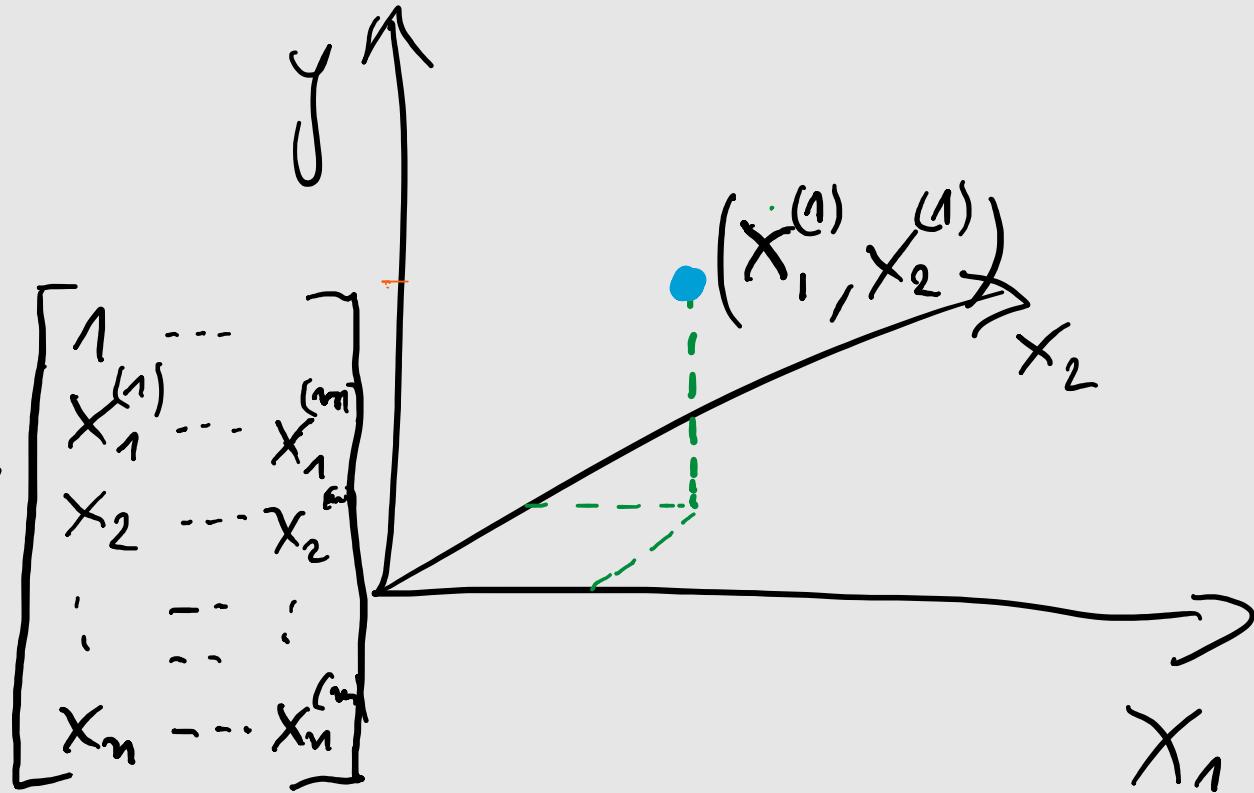
$$\frac{\partial}{\partial \theta_j} J(\theta) = (h_\theta(x) - y) x_j$$

# Multiple features

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

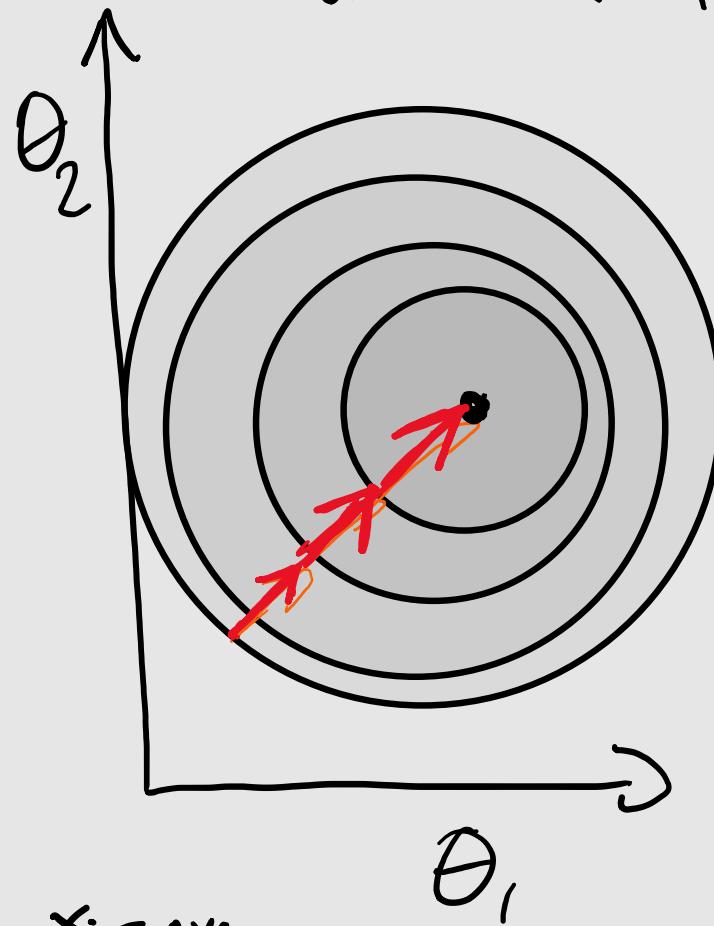
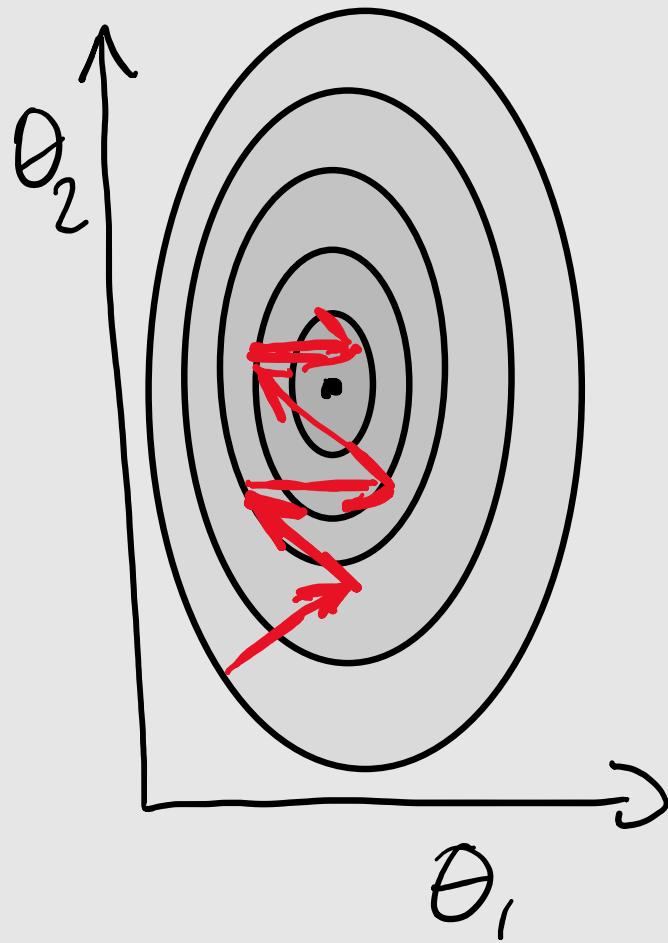
$$h_{\theta}(x) = \theta^T x$$

$$h_{\theta}(x) = [\theta_0, \theta_1, \dots, \theta_n]$$



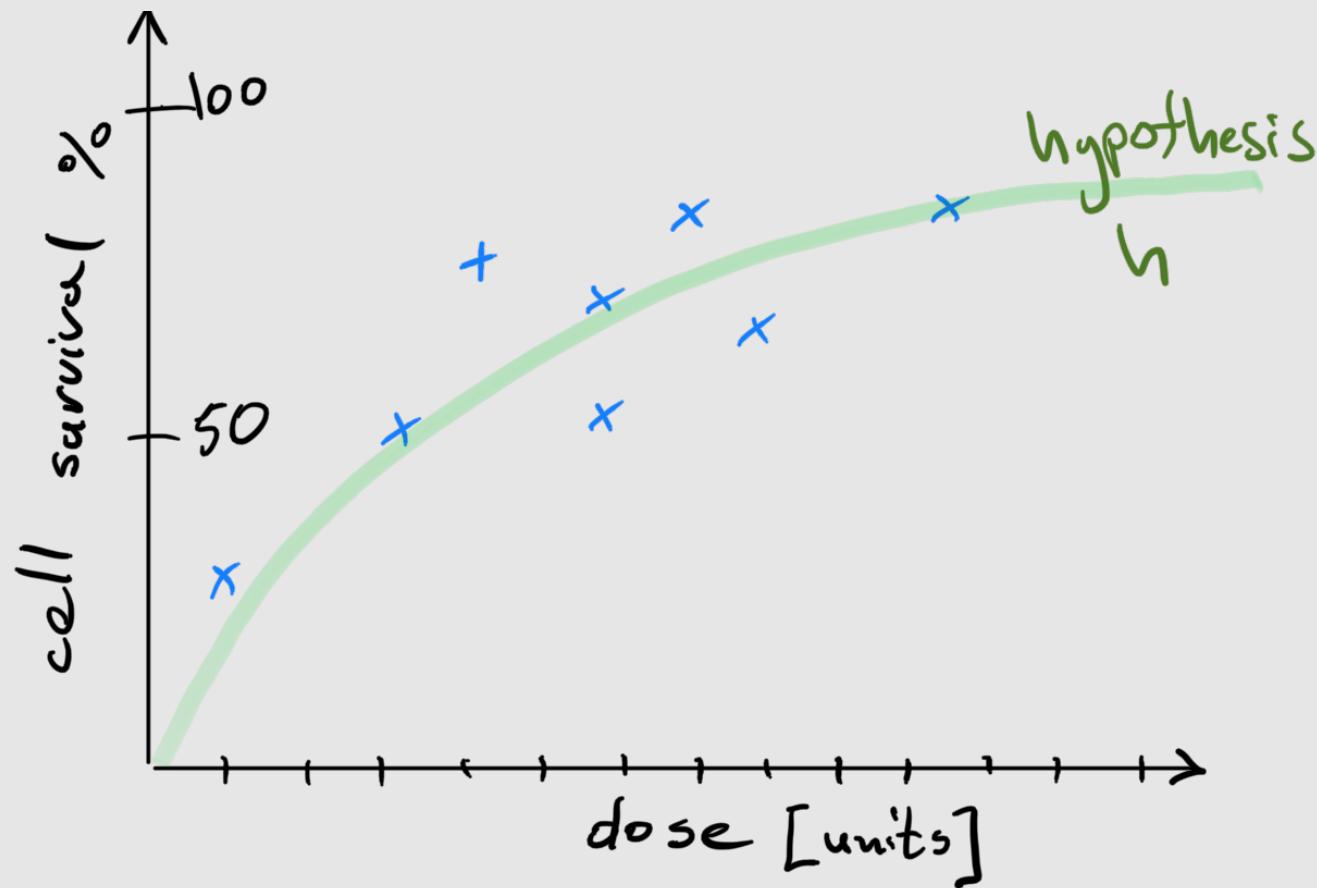
# Multiple features

$\alpha$  - learning rate doesn't like un-scaled features



$$x_{\text{scaled}} = \frac{x_i - \text{avg}}{\text{max} - \text{min}}$$

# Non-linear regression

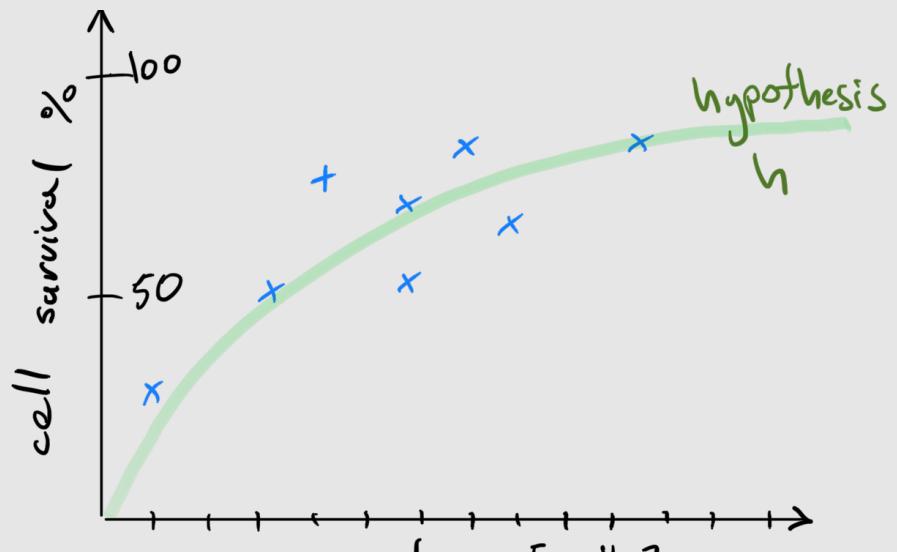


# Multiple non-linear features

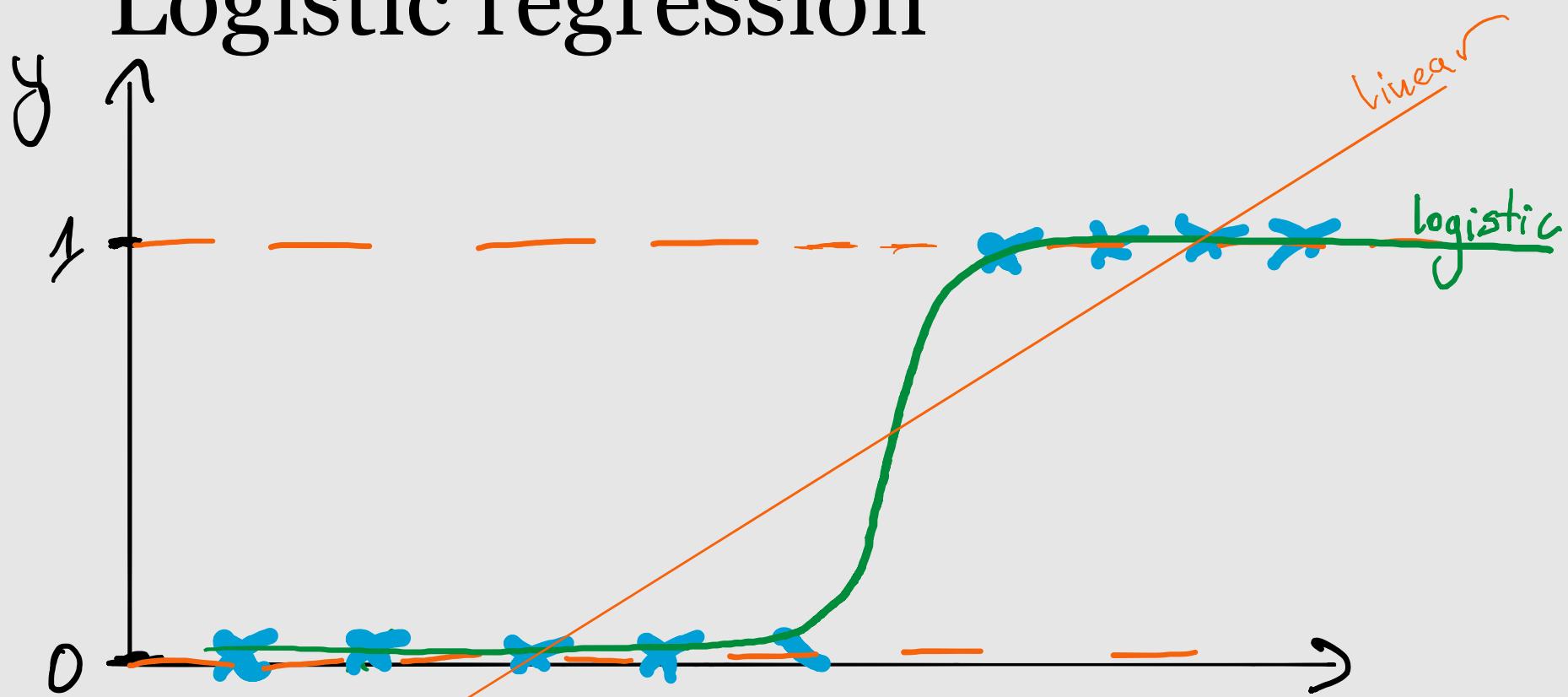
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_n x^n \quad \left. \right\} \text{polynomial}$$

$$h_{\theta}(x) = \theta_0 + \theta_1 \sqrt{x} + \theta_2 \log(x) + \dots \quad \left. \right\} \text{non-linear (generalised)}$$

$J(\theta)$  → stays the same.



# Logistic regression



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

$$J(\theta) = ?$$

# Logistic regression

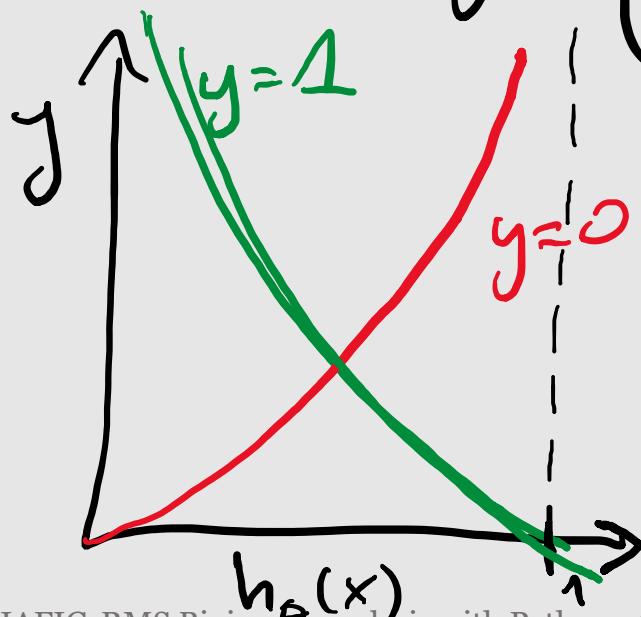
$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

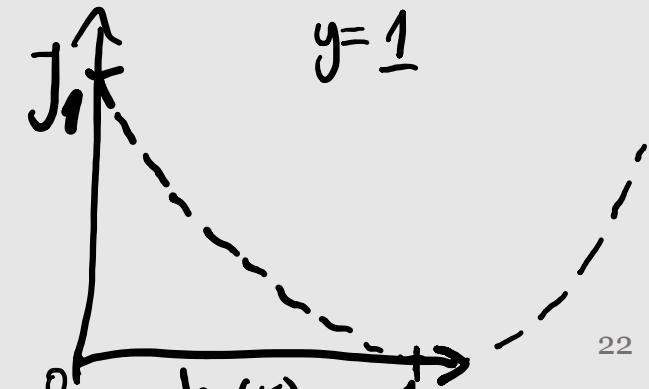
normal sum of squares makes  $J$

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & : y=1 \\ -\log(1-h_{\theta}(x)) & : y=0 \end{cases}$$

$\therefore J$  non-convex



Squared error



# Scikit-learn library

## scikit-learn Machine Learning in Python

Getting Started

What's New in 0.22

GitHub

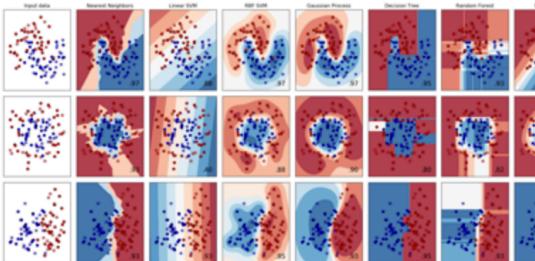
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, and more...



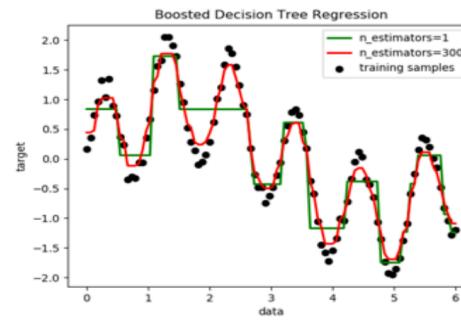
Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, nearest neighbors, random forest, and more...



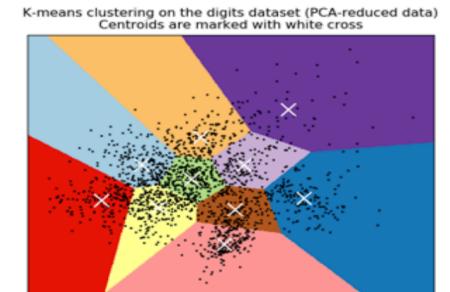
Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, and more...



Examples