UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

Lesson 2.

Linear regression

DATA SCIENCE AND
AUTOMATION COURSE

MASTER DEGREE SMART
TECHNOLOGY ENGINEERING

TEACHER
Mirko Mazzoleni

PLACE
University of Bergamo

Outline

1. Components of learning
2. Linear regression
3. Least squares
4. Gradient descent

Outline

1. **Components of learning**

2. Linear regression

3. Least squares

4. Gradient descent

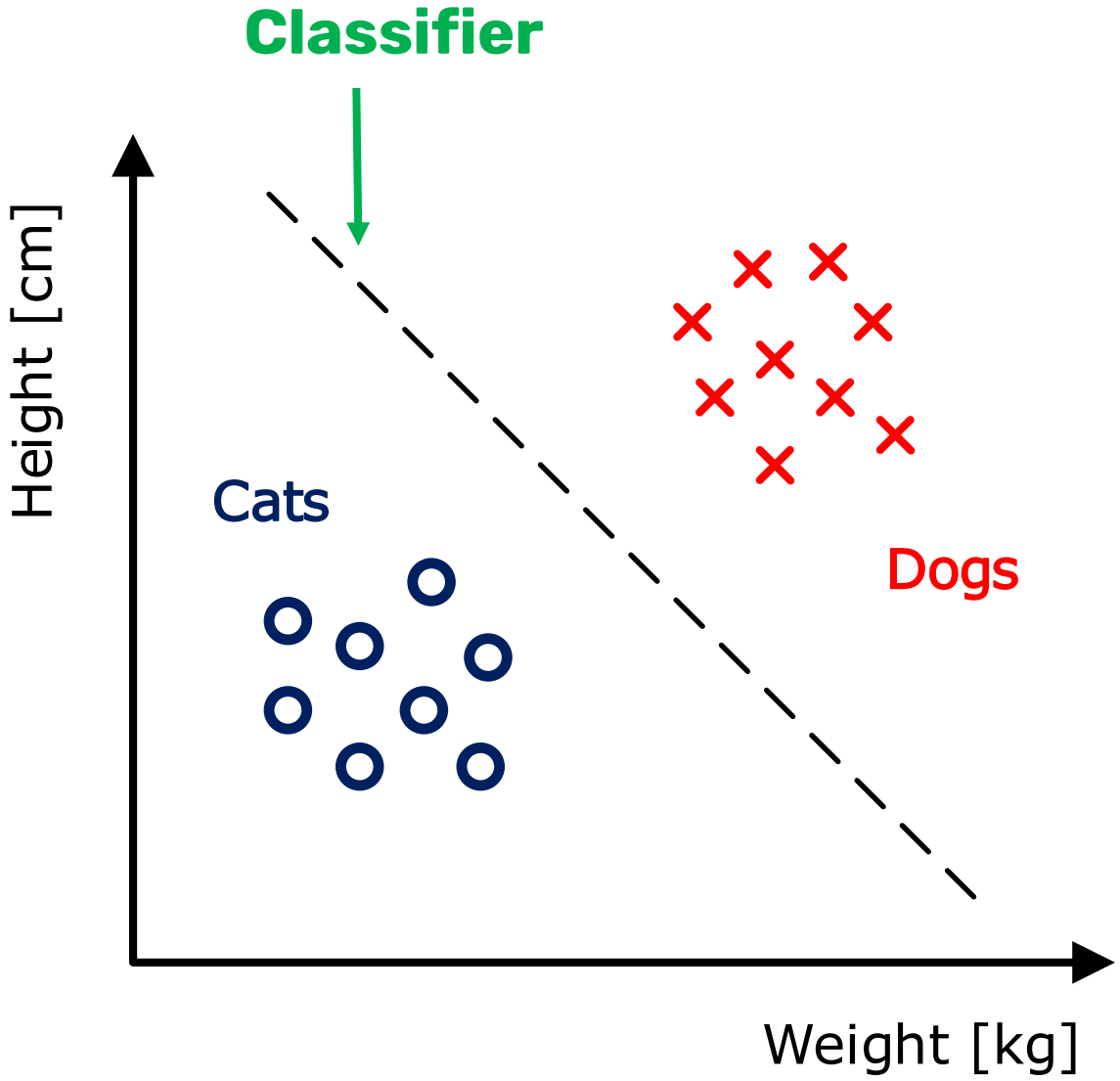
Components of learning (in general)

- Input: $\boldsymbol{x}$ (e-mail textual content) → each dimension is some e-mail attribute
- Output: y (spam / not spam?) → the decision that we have to take in the end
- Target function: $f: \mathcal{X} \to \mathcal{Y}$ (Ideal spam filter formula) → unknown, we have to learn it
- Data: $\mathcal{D} = \{(\boldsymbol{x}(1), y(1)), \ldots, (\boldsymbol{x}(N), y(N))\}$ (historical records of e-mail examples)
  - ✓ Each **feature vector** $\boldsymbol{x}$ consists of different **regressors** or **features**, i.e. information used to predict the output variable
- Hypothesis: $g: \mathcal{X} \to \mathcal{Y}, g \in \mathcal{H}$ (formula to be used) → g is an **approximation** of f

$\mathcal{H}$ is called the **Hypothesis space**. This, together with the **Learning algorithm**, form the **learning model**

Supervised learning

- The “correct answer” (output label) y is given
- Predict y from a set of inputs $\boldsymbol{x} \in \mathbb{R}^{d \times 1}$
- **Regression:** predict a continuous output $y \in \mathbb{R}$ **(real value)**
- **Classification:** predict a discrete categorical output $y \in \{1,2,\dots,C\}$ **(class)**



Example: house prices regression

- Suppose we want to find a linear function which relates the **measured** regressors x_1, x_2, x_3, x_4 with the **observed** output y

Single feature x_3

Output variable y

Number of observations N

Size $[\text{feet}^2]$	Number of bedrooms	Number of floors	Age of home [year]	Price [$]
2104	5	1	45	$4.60 \cdot 10^5$
1416	3	2	40	$2.32 \cdot 10^5$
1534	2	1	30	$3.15 \cdot 10^5$
⋮	⋮	⋮	⋮	⋮
↓	↓	↓	↓	↓
x_1	x_2	x_3	x_4	y

Single observation (feature vector) x

- The number of rows is the number of data points (also known as number of observations) N
- The i-th observation is the vector $\boldsymbol{x}(i) = [x_1(i) \;\; x_2(i) \;\; x_3(i) \;\; x_4(i)]^T \in \mathbb{R}^{4x1}$
- Each feature vector $\boldsymbol{x}$ has associated a response $y \in \mathbb{R}$ that we want to predict for new observations $\boldsymbol{x}^*$

Example: house prices classification

- The components of the features vector are the same. The difference lies in the response variable, which now is a **class** (categorical data type) and not a real value

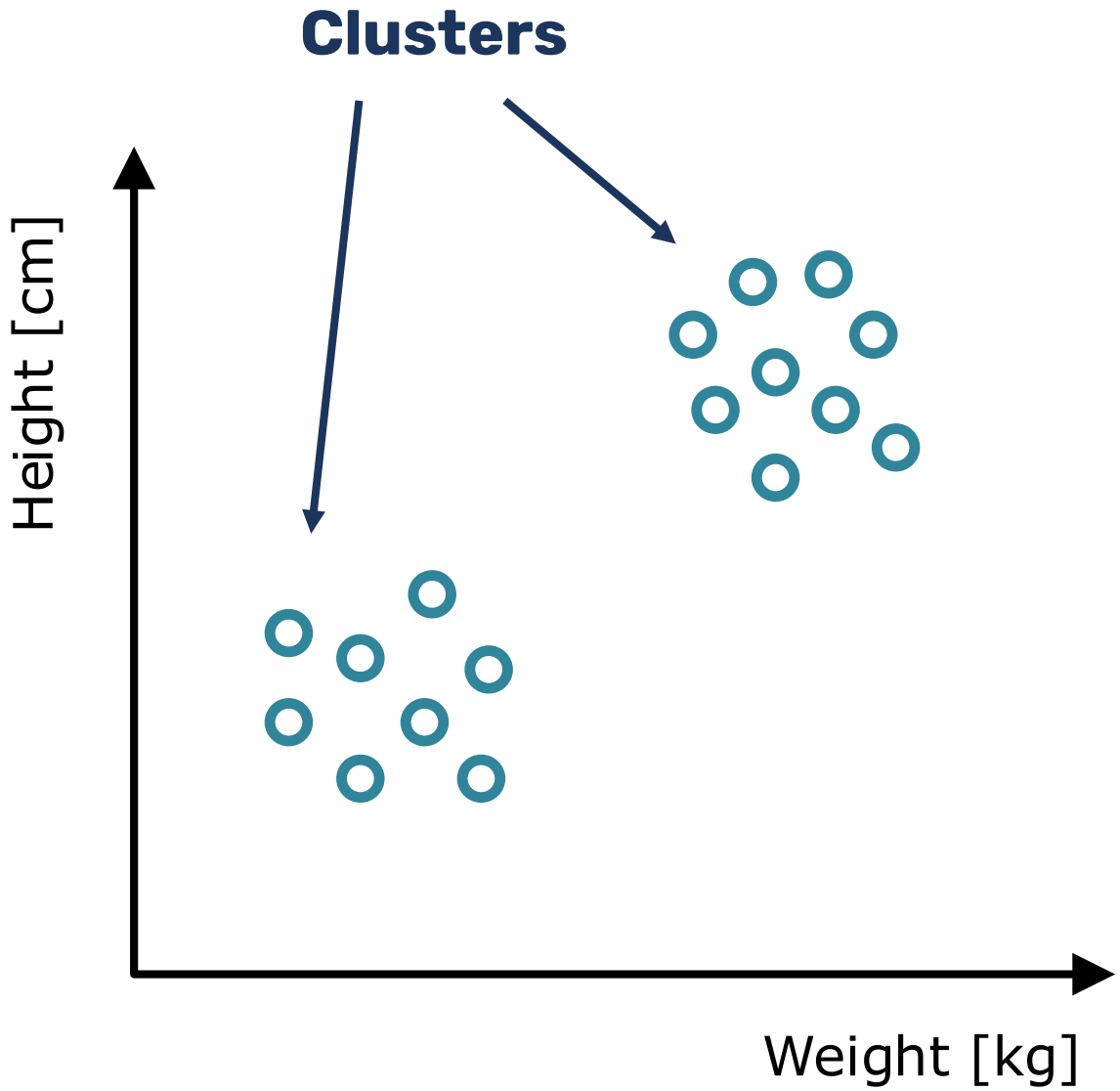
- Suppose that instead of the price value in dollars, we want to classify houses as **expensive** *(class* $y = 1$*)* or **cheap** *(class* $y = 0$*)*

Size $[\text{feet}^2]$	Number of bedrooms	Number of floors	Age of home [year]	Price [class]
2104	5	1	45	1
1416	3	2	40	0
1534	2	1	30	1
⋮	⋮	⋮	⋮	⋮
↓	↓	↓	↓	↓
x_1	x_2	x_3	x_4	y

- The point $\boldsymbol{x}$ could be classified to class $y = 1$ if *the probability of* $\boldsymbol{x}$ *to belong to class 1 is* ≥ 0.5

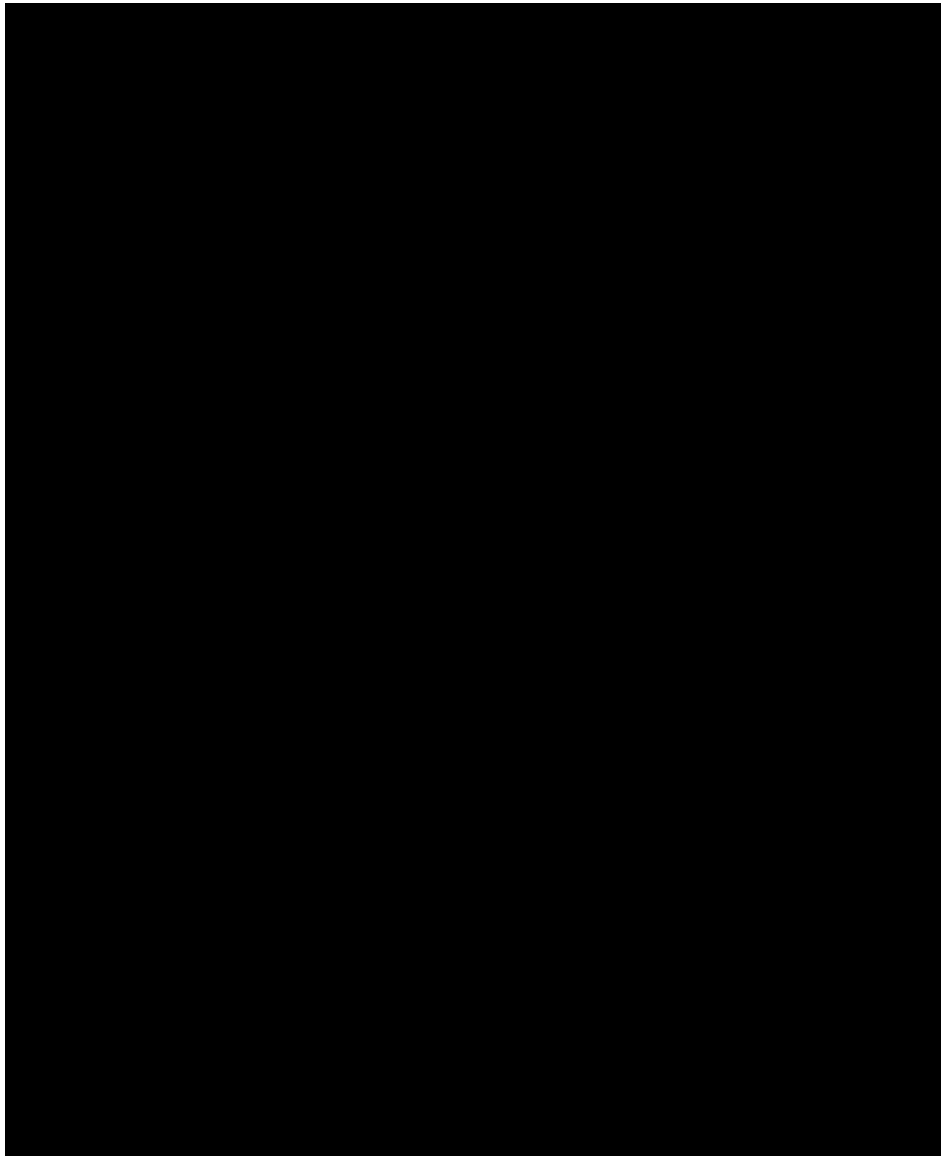
Unsupervised learning

- Instead of **(input, output)** we get **(input, ?)**
- Here there is no a function f to learn
- Find properties of the inputs $\boldsymbol{x} \in \mathbb{R}^{d \times 1}$
- High-level representation of the input
- Elements into the same cluster have similar properties



Reinforcement learning

- Instead of **(input, output)** we get **(input, output, reward)**
- The algorithm tries to learn what action to take, in order to maximize the reward
- This is called a policy
- Applications in control, robotics, A/B testing



Business problems as data science examples - revisited

Supervised

- Spam e-mail detection system **Classification**
- Credit approval **Classification**
- Recognize objects in images **Classification**
- Find the relation between house prices and house sizes **Regression**
- Predict the stock market **Regression**

Unsupervised

- Market segmentation **Clustering**
- Market basket analysis **Co-occurence grouping**
- Language models (word2vec) **Similarity matching**
- Social network analysis **Link prediction**
- Low-order data representations **Data reduction**

- Movies recommendation **Similaritiy matching**

Supervised or unsupervised

Supervised learning: problem statement

The aim is to **learn an unknown function** f given a dataset $\mathcal{D}$

- The function is searched in the hypothesis space $\mathcal{H}$, where $h \in \mathcal{H}$ is a specific function
- We want to find a function h that approximates f well, on the **whole domain** $\mathcal{X}$

What does $h \approx f$ mean?

- We need to define an **error measure**
- Almost always this is a **pointwise** definition: $e\big(f(\boldsymbol{x}), h(\boldsymbol{x})\big)$

Cost functions

Pointwise error examples

- *Squared error:* $e\big(f(\boldsymbol{x}), h(\boldsymbol{x})\big) = \big(f(\boldsymbol{x}) - h(\boldsymbol{x})\big)^2$ → used for regression
- *Binary error:* $e\big(f(\boldsymbol{x}), h(\boldsymbol{x})\big) = \mathbb{I}[f(\boldsymbol{x}) \neq h(\boldsymbol{x})]$ → used for classification

It is interesting to look at the **overall error**, which considers all N examples:

Overall error examples

- *In-sample error:* $E_{in} = \frac{1}{N}\sum_{i=1}^{N} e\big(f(\boldsymbol{x}), h(\boldsymbol{x})\big)$ → error on data that I actually have
- *Out-of-sample error:* $E_{out} = \mathbb{E}_{\boldsymbol{x}}\big[e\big(f(\boldsymbol{x}), h(\boldsymbol{x})\big)\big]$ → error on data I could possibly observe

Outline

1. Components of learning

2. **Linear regression**

3. Least squares

4. Gradient descent

Linear regression

Aim: Suppose to have at disposal a dataset $\mathcal{D} = \{(\boldsymbol{x}(1), y(1)), \dots, (\boldsymbol{x}(N), y(N))\}$. Find the relation between a set of input variables $\boldsymbol{x} \in \mathbb{R}^{(d-1)\times 1}$ and an output variable $y \in \mathbb{R}$, using a **linear model:**

$$y(i) = \theta_0 + \theta_1 x_1(i) + \cdots + \theta_{d-1} x_{d-1}(i) + \epsilon(i) = \sum_{j=0}^{d-1} \theta_j \, x_j(i) + \epsilon(i)$$

$$= \underset{1\times d}{\boldsymbol{x}(i)^T} \underset{d\times 1}{\boldsymbol{\theta}} + \underset{1\times 1}{\epsilon(i)}$$

$$\begin{bmatrix} \cdots \end{bmatrix} \begin{bmatrix} \vdots \end{bmatrix}$$

i −th observation

- $x_0 = 1$
- $\boldsymbol{x} = [x_0, x_1, \dots, x_{d-1}]^T \in \mathbb{R}^{d\times 1}$
- $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_{d-1}]^T \in \mathbb{R}^{d\times 1}$

Hypothesis $h(\boldsymbol{\theta})$: linear model with parameters $\boldsymbol{\theta}$

Hypothesis space $\mathcal{H}$: set of all linear models

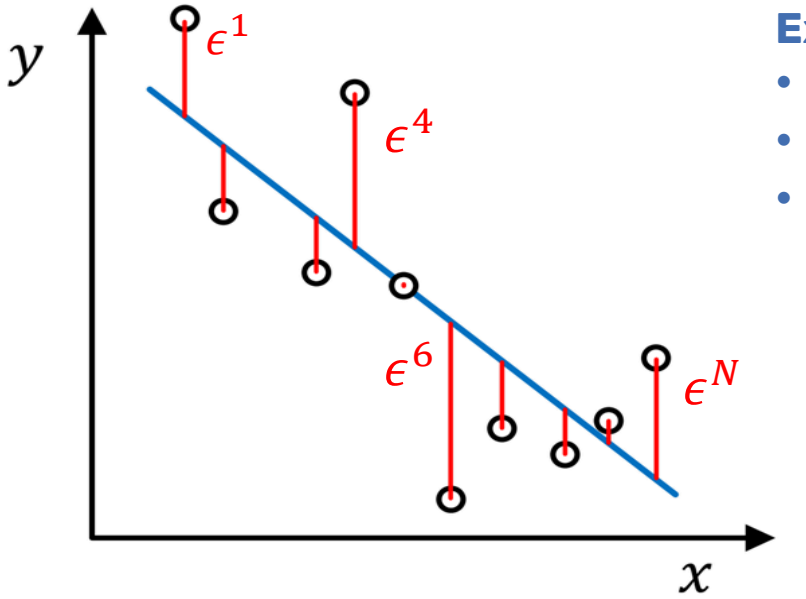
- The vector $\boldsymbol{\theta}$ is called **parameters vector** → to be found by minimizing a cost function
- The vector $\boldsymbol{x}(i)$ is called **features vector** for i-th observation → attributes of individuals
- The quantity $\epsilon(i)$ is the error due not perfect explanation of the $y(i)$ using $\boldsymbol{x}(i)$

Geometrical interpretation

One-variable case

In this case, there is only **one feature** x_1 and **two parameters** θ_0, θ_1

$$y(i) = \theta_0 + \theta_1 x_1(i) + \epsilon(i)$$



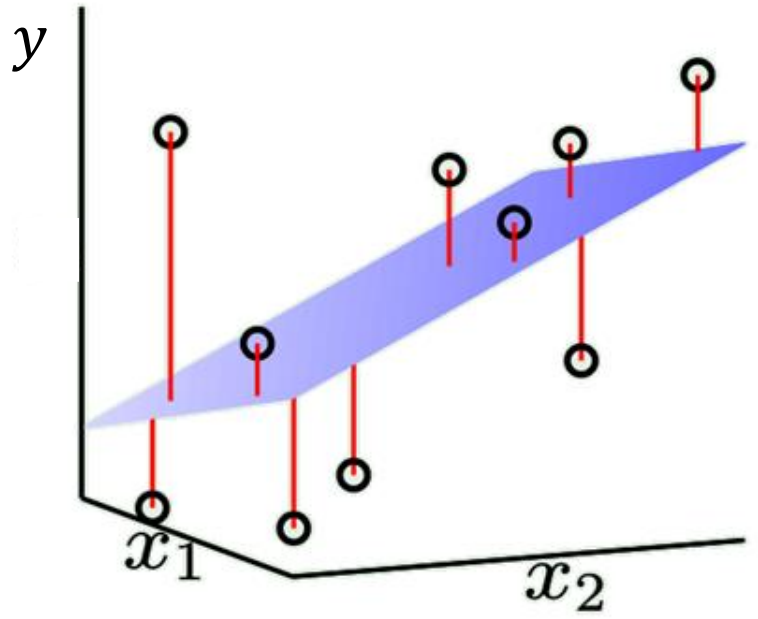
Example

- y: weight [kg]
- x_1: height [m]
- x_2: sex [M/F]

Two-variables case

In this case, there are **two features** x_1, x_2 and **three parameters** $\theta_0, \theta_1, \theta_2$

$$y(i) = \theta_0 + \theta_1 x_1(i) + \theta_2 x_2 + \epsilon(i)$$



Outline

1. Components of learning

2. Linear regression

3. **Least squares**

4. Gradient descent

Least squares cost function

How well does $h(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\theta}$ approximates $f(\boldsymbol{x})$?

- Linear regression estimates the parameters $\boldsymbol{\theta}$ using the **least squares method**, i.e. by minimizing the **squared error** between **observed** and **predicted** output

Cost function

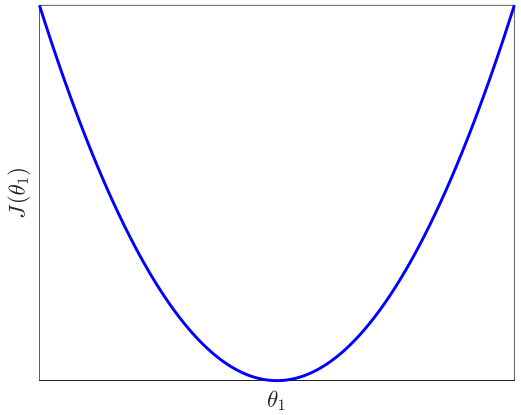
$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} J(\boldsymbol{\theta}) \qquad J(\boldsymbol{\theta}) = E_{in}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y(i) - \boldsymbol{x}(i)^T\boldsymbol{\theta})^2 = \frac{1}{N}\sum_{i=1}^{N}\big(\epsilon(i)\big)^2$$

$\boldsymbol{x}(i)^T$: $\begin{bmatrix} \cdots \end{bmatrix}$ $\boldsymbol{\theta}$: $\begin{bmatrix} \vdots \end{bmatrix}$

One-variable case

with $\theta_0 = 0$

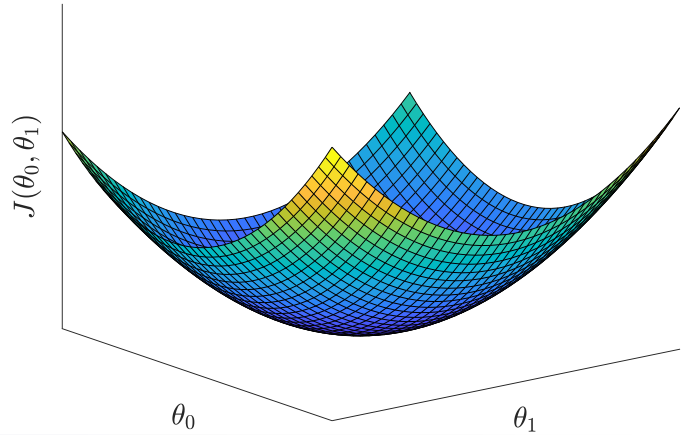
$$y(i) = \theta_1 x_1(i) + \epsilon(i)$$



One-variable case

with $\theta_0 \neq 0$

$$y(i) = \theta_0 + \theta_1 x_1(i) + \epsilon(i)$$



Cost function - minimization

$$J(\boldsymbol{\theta}) = E_{in}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y(i) - \boldsymbol{x}(i)^T\boldsymbol{\theta})^2$$

$$\nabla J(\boldsymbol{\theta}) = \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0} \Rightarrow \frac{2}{N}\sum_{i=1}^{N} \boldsymbol{x}(i) \cdot (y(i) - \boldsymbol{x}(i)^T\boldsymbol{\theta}) = \mathbf{0} \Rightarrow \sum_{i=1}^{N} \boldsymbol{x}(i)y(i) - \sum_{i=1}^{N} \boldsymbol{x}(i)\boldsymbol{x}(i)^T\boldsymbol{\theta} = \mathbf{0}$$

$$\Rightarrow \left[\sum_{i=1}^{N} \boldsymbol{x}(i)\boldsymbol{x}(i)^T\right]\boldsymbol{\theta} = \sum_{i=1}^{N} \boldsymbol{x}(i)y(i)$$

$$\Rightarrow \widehat{\boldsymbol{\theta}} = \left[\sum_{i=1}^{N} \boldsymbol{x}(i)\boldsymbol{x}(i)^T\right]^{-1} \cdot \left[\sum_{i=1}^{N} \boldsymbol{x}(i)y(i)\right]$$

- Since the model is **linear in the parameters** and the error measure is **quadratic**, the cost function is **convex** → it admits a **unique** (global) minimum
- In this case the minimum can be found in **closed-form**

Least squares cost function – matrix form

We can express the linear regression problem using matrices

Features vector $\boldsymbol{x}^T(i)$

$$\underset{N\times d}{X} = \begin{bmatrix} 1 & x_1(1) & x_2(1) & \cdots & x_{d-1}(1) \\ 1 & x_1(2) & x_2(2) & & x_{d-1}(2) \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & x_1(N) & x_2(N) & \cdots & x_{d-1}(N) \end{bmatrix} \quad \underset{d\times 1}{\boldsymbol{\theta}} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{d-1} \end{bmatrix} \quad \underset{N\times 1}{Y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \quad \underset{N\times 1}{E} = \begin{bmatrix} \epsilon(1) \\ \epsilon(2) \\ \vdots \\ \epsilon(N) \end{bmatrix}$$

$$\underset{N\times d}{X} = \begin{bmatrix} \boldsymbol{x}^T(1) \\ \boldsymbol{x}^T(2) \\ \vdots \\ \boldsymbol{x}^T(N) \end{bmatrix}$$

$$Y = X\boldsymbol{\theta} + E \Rightarrow \boxed{J(\boldsymbol{\theta}) = \frac{1}{N}\|Y - X\boldsymbol{\theta}\|_2^2 = \frac{1}{N}\underset{1\times N}{(Y - X\boldsymbol{\theta})^T}\underset{N\times 1}{(Y - X\boldsymbol{\theta})}}$$

IN-DEPTH ANALYSIS

Least squares cost function – matrix form

It is useful to remember these matrix derivation properties:

$$\underset{1\times d}{\nabla_{\boldsymbol{x}}}\left(\underset{1\times d}{\boldsymbol{x}^T}\cdot\underset{d\times d}{A}\cdot\underset{d\times 1}{\boldsymbol{x}}\right)=\underset{d\times d}{(A+A^T)}\cdot\underset{d\times 1}{\boldsymbol{x}} \qquad \underset{1\times d}{\nabla_{\boldsymbol{x}}}\left(\underset{1\times d}{\boldsymbol{x}^T}\cdot\underset{d\times 1}{\boldsymbol{b}}\right)=\underset{d\times 1}{\boldsymbol{b}}$$

$$J(\boldsymbol{\theta})=\frac{1}{N}(Y-X\boldsymbol{\theta})^T(Y-X\boldsymbol{\theta})=\frac{1}{N}(Y^TY-\underset{1\times N\ N\times d\ d\times 1}{Y^TX\cdot\boldsymbol{\theta}}-\underset{1\times d\ d\times N\ N\times 1}{\boldsymbol{\theta}^T\cdot X^T\,Y}+\boldsymbol{\theta}^T\cdot X^T\,X\cdot\boldsymbol{\theta})$$

$$=\frac{1}{N}(Y^TY-\ 2\cdot\boldsymbol{\theta}^T\cdot X^T\,Y\ +\ \boldsymbol{\theta}^T\cdot X^T\,X\cdot\boldsymbol{\theta})$$

$$\nabla J(\boldsymbol{\theta})=\mathbf{0}\Rightarrow\frac{1}{N}(-2X^TY+2X^TX\boldsymbol{\theta})=\mathbf{0}\Rightarrow \qquad \underset{d\times 1}{\widehat{\boldsymbol{\theta}}}=\underset{d\times d}{(X^TX)^{-1}}\underset{d\times N\ N\times 1}{X^TY}$$

Normal equations

$$\widehat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T Y$$

Normal equations

What if the matrix $X^T X$ is **not invertible**? → Use **pseudo-inverse**. In MatLab:

```
theta_hat = pinv(X'*X)*X*y
```

- Redundant features (linearly dependent)
  - ✓ $x_1 = \text{height in m}^2$
  - ✓ $x_2 = \text{height in feet}^2$
- Too many features (ex. $N \leq d$)
  - ✓ Delete some feature
  - ✓ Use regularization (later in the course)

- The method of normal equations is **slow** if d is very large
  - ✓ To solve this, iterative methods using **gradient descent** can be used

Outline

1. Components of learning
2. Linear regression
3. Least squares
4. **Gradient descent**

Gradient descent

- The **gradient descent** is a general iterative method for minimizing differentiable functions

- The value of the parameters at iteration $t + 1$ is (given a random initial point $\widehat{\boldsymbol{\theta}}(0)$)

$$\underset{d \times 1}{\widehat{\boldsymbol{\theta}}(t+1)} = \underset{d \times 1}{\widehat{\boldsymbol{\theta}}(t)} - \underset{1 \times 1}{\alpha} \cdot \underset{d \times 1}{\nabla J(\boldsymbol{\theta})} \Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}(t)}$$

$\alpha \in \mathbb{R}_{>0}$: learning rate

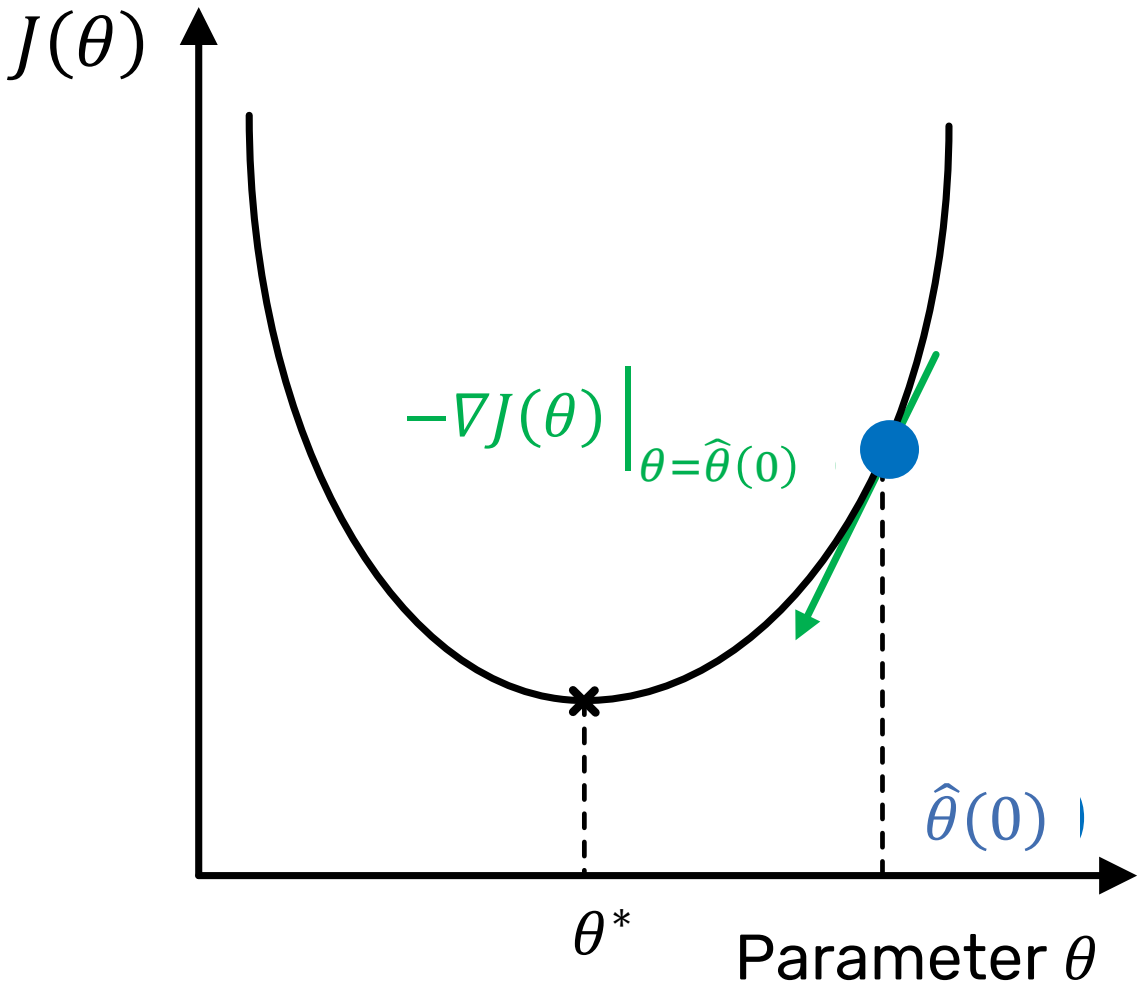
Gradient descent

Consider the **scalar** case where $\theta \in \mathbb{R}$

$$\hat{\theta}(t+1) = \hat{\theta}(t) - \alpha \nabla J(\theta)\Big|_{\theta=\hat{\theta}(t)}$$

- $\nabla J(\theta)|_{\theta=\hat{\theta}(t)} > 0 \Rightarrow \hat{\theta}(t+1) < \hat{\theta}(t)$

The new estimate is closer to the optimal value θ^*



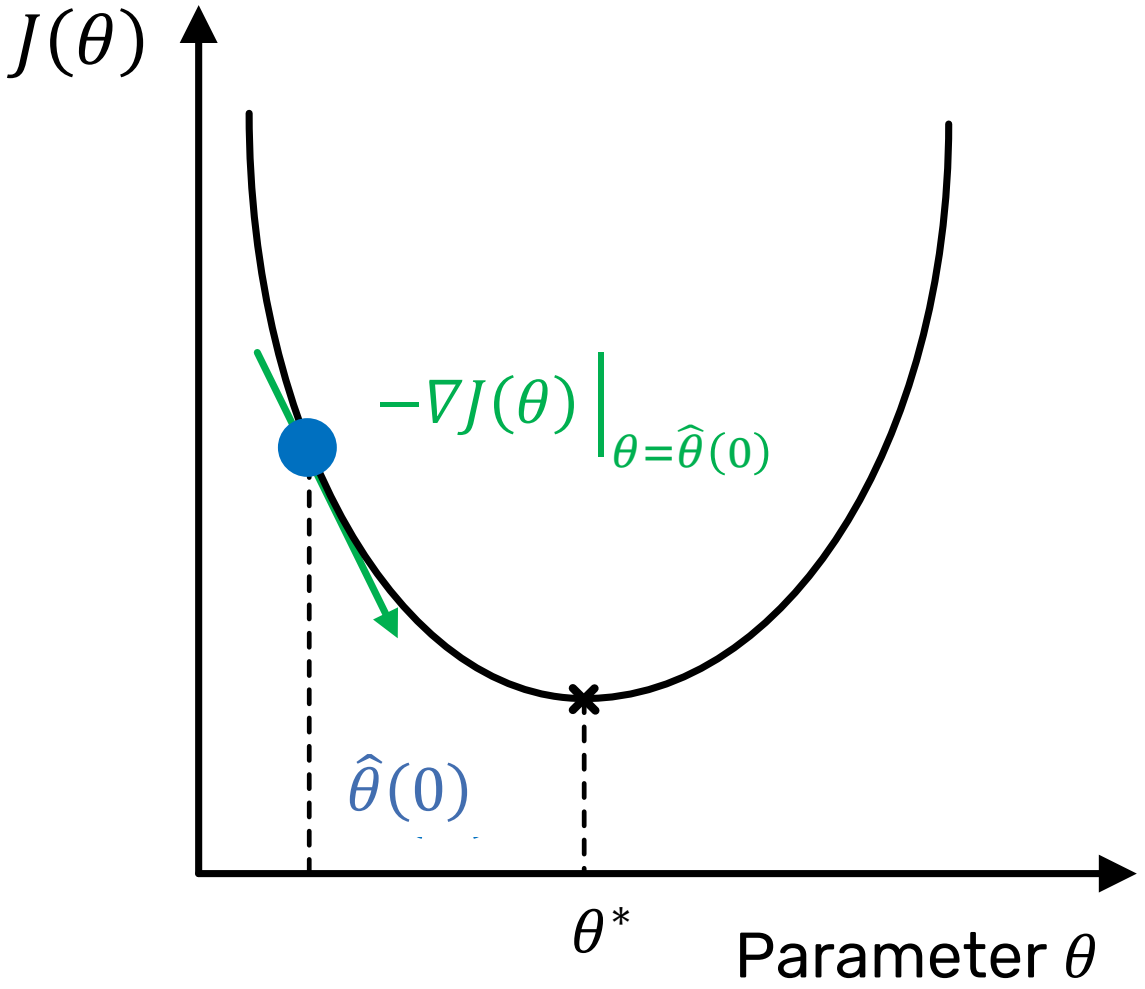
Gradient descent

Consider the **scalar** case where $\theta \in \mathbb{R}$

$$\hat{\theta}(t+1) = \hat{\theta}(t) - \alpha \nabla J(\theta)\Big|_{\theta=\hat{\theta}(t)}$$

- $\nabla J(\theta)|_{\theta=\hat{\theta}(t)} < 0 \Rightarrow \hat{\theta}(t+1) > \hat{\theta}(t)$

The new estimate is closer to the optimal value θ^*



Linear regression laboratory: predict house prices

We want to **predict the price** of the houses in Portland, Oregon

- Each house is described by the following features
  - ✓ x_1: Number of **bedrooms**
  - ✓ x_2: **Size** [feet^2]
- The **training set** consists of $N = 47$ houses with $x_1(i), x_2(i)$ and $y(i)$, for $i = 1, \dots, N$

$$y(i) = \boldsymbol{x}(i)^T \boldsymbol{\theta} + \epsilon(i) \qquad \underset{3 \times 1}{\boldsymbol{x}(i)} = [1 \quad x_1(i) \quad x_2(i)]^T$$

$$\underset{47 \times 3}{X} = \begin{bmatrix} \boldsymbol{x}^T(1) \\ \boldsymbol{x}^T(2) \\ \vdots \\ \boldsymbol{x}^T(N) \end{bmatrix} \qquad \underset{3 \times 1}{\boldsymbol{\theta}} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad \underset{47 \times 1}{Y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(47) \end{bmatrix}$$

```
% Read data from file
data = csvread('housedata.txt');
X = data(:, 1:2); % Features
y = data(:, 3); % Price
N = length(y); % Number of data

% Add intercept term to X
X = [ones(N, 1) X];

% Calculate the parameters from
the normal equation
theta_hat = pinv(X'*X)*X'*y;
```

```
% Estimate the price of a 1650
sq-ft, 3 br house
price_hat = [1 3 1650]*theta_hat;
```

Point not seen during training (i.e. estimation of θ)