Lecture 04

Machine Learning 1:

classical learning - regression

2026-03-12

Sébastien Valade

VNIVERSIDAD NACIONAL
AVTÓNOMA DE
MEXICO

Course update

This *3rd course edition* (2026) places **much more focus on deep learning strategies** for computer vision

$\Rightarrow$ several lectures were removed:

- Morphology & Segmentation
- Homography
- Features & Motion Estimation
- PCA

$\Rightarrow$ refer to the 2nd course edition if interested: https://svalade.github.io/cv4gs/2024/

What is machine learning?

⇒ Term introduced by Arthur Samuel in 1959:
"Machine learning gives computers the ability to learn without being explicitly programmed"

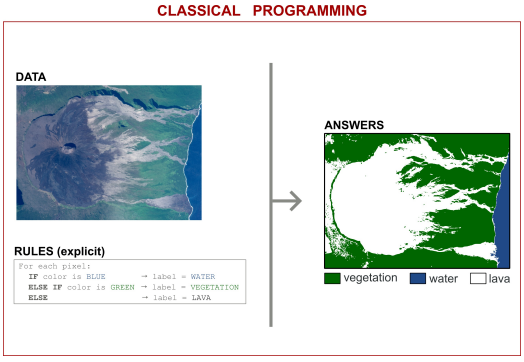
⇒ Radical change of paradigm with respect to classical programming:
rules are learned from data rather than explicitly programmed

What is machine learning?

⇒ Term introduced by Arthur Samuel in 1959:
"Machine learning gives computers the ability to learn without being explicitly programmed"

⇒ Radical change of paradigm with respect to classical programming:
rules are learned from data rather than explicitly programmed

⇒ EX: label pixel land-use in satellite image

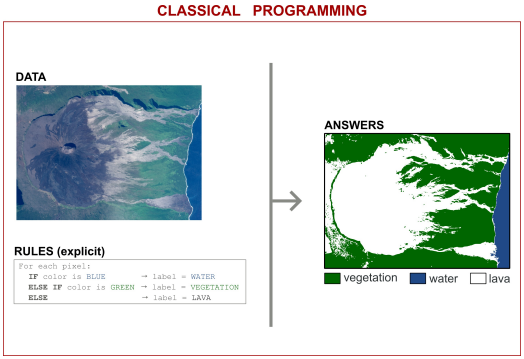


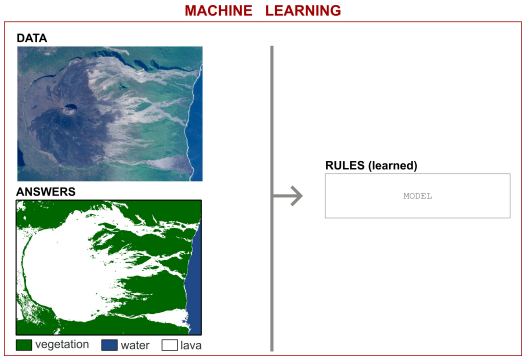
What is machine learning?

⇒ Term introduced by Arthur Samuel in 1959:
"Machine learning gives computers the ability to learn without being explicitly programmed"

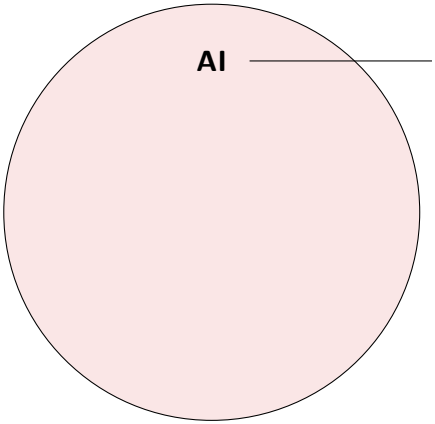
⇒ Radical change of paradigm with respect to classical programming:
rules are learned from data rather than explicitly programmed

⇒ EX: label pixel land-use in satellite image





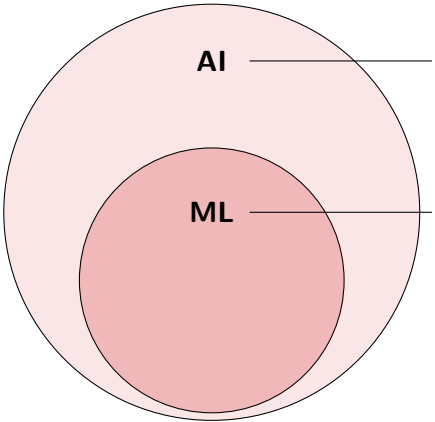
Keywords: AI, ML and DL



Artificial Intelligence

broad concept, whereby machine mimics human behaviour

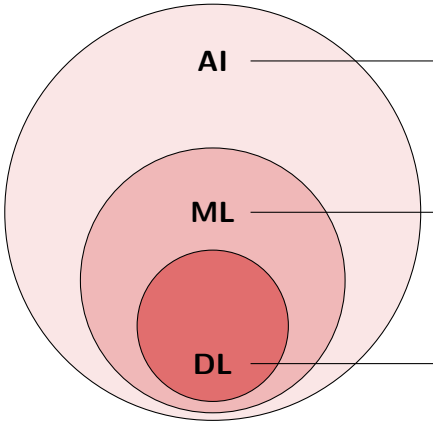
Keywords: AI, ML and DL



Artificial Intelligence
broad concept, whereby machine mimics human behaviour

Machine Learning *(a.k.a. Statistical Learning, Classical Learning)*
subset of AI which uses **statistical** methods
(features are designed by the user)

Keywords: AI, ML and DL

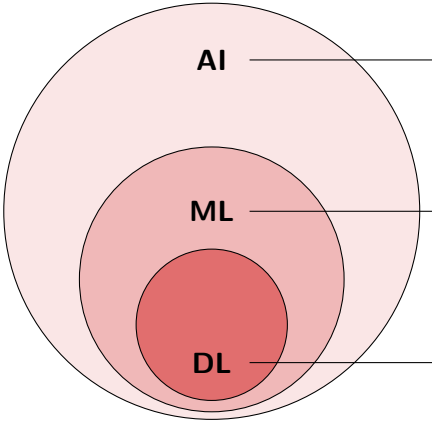


Artificial Intelligence
broad concept, whereby machine mimics human behaviour

Machine Learning *(a.k.a. Statistical Learning, Classical Learning)*
subset of AI which uses **statistical** methods
(features are designed by the user)

Deep Learning *(a.k.a. Modern Machine Learning)*
subset of ML, which uses **multi-layered neural networks**
(features are learned by the network)

Keywords: AI, ML and DL



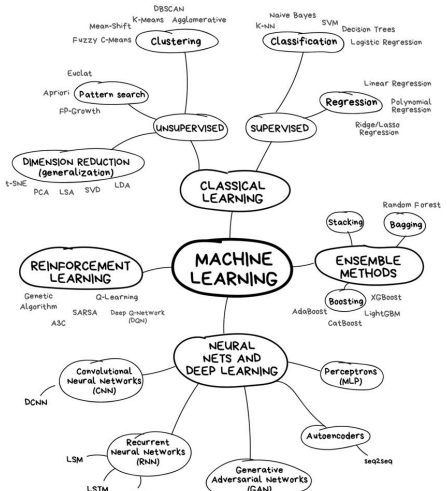
Artificial Intelligence
broad concept, whereby machine mimics human behaviour

Machine Learning *(a.k.a. Statistical Learning, Classical Learning)*
subset of AI which uses **statistical** methods
(features are designed by the user)

Deep Learning *(a.k.a. Modern Machine Learning)*
subset of ML, which uses **multi-layered neural networks**
(features are learned by the network)

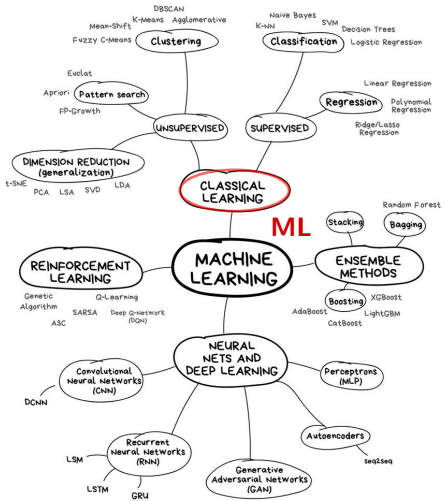
ML: lectures 4 (ML1 today), 5 (ML2 next week)
DL: lectures 6, 7, 8, 9

Mindmap of the machine learning world: ML is a huge (and growing) field!



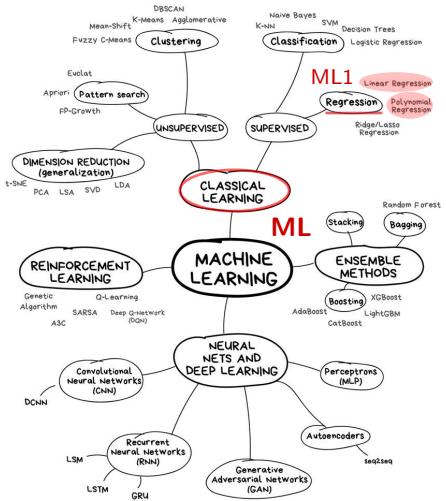
source

Mindmap of the machine learning world: ML is a huge (and growing) field!



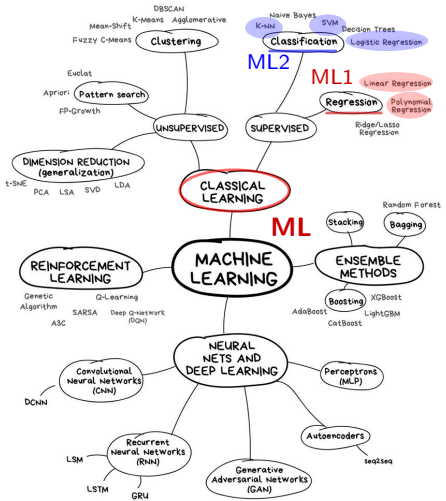
source

Mindmap of the machine learning world: ML is a huge (and growing) field!



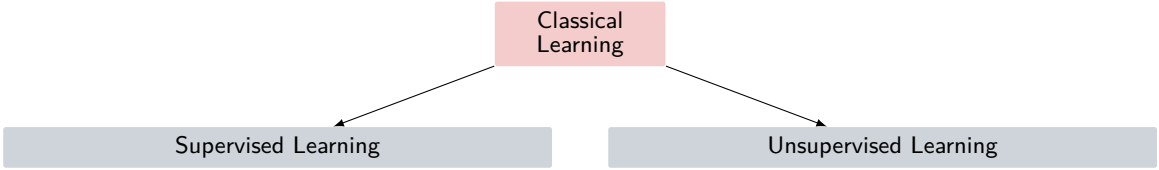
source

Mindmap of the machine learning world: ML is a huge (and growing) field!



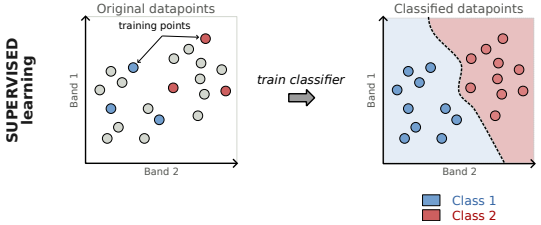
source

Introduction

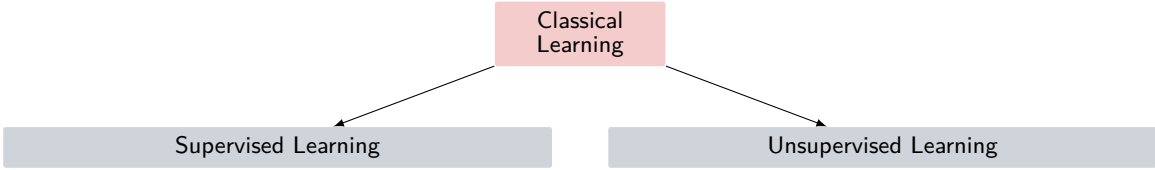


- Learning algorithm is presented inputs and desired outputs:
  training data $D = (in, out)$

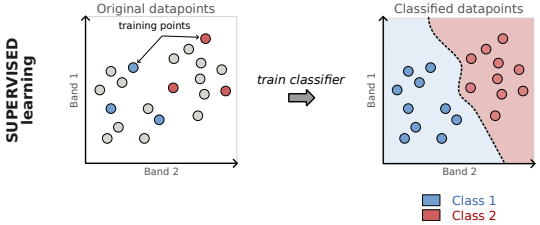
- Goal: learn a general rule f that maps inputs to outputs
  $f(in) = out$



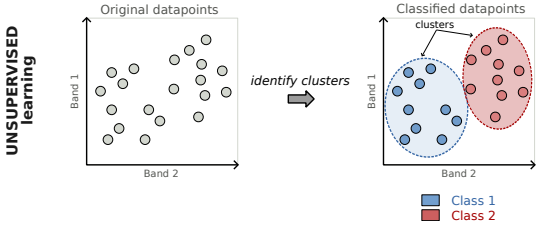
Introduction



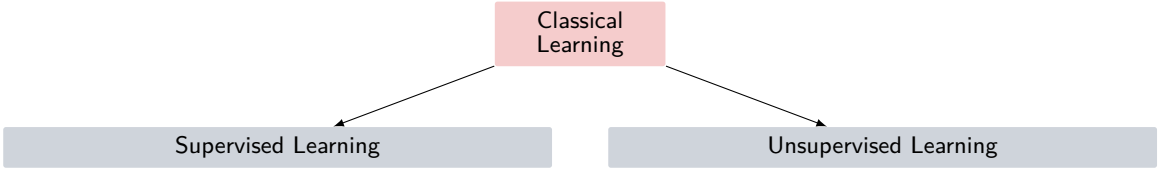
- Learning algorithm is presented inputs and desired outputs:
  training data $D = (in, out)$
- Goal: learn a general rule f that maps inputs to outputs
  $f(in) = out$



- No training data is given to the learning algorithm
- Goal: find structure data, discover hidden patterns, learn features



Introduction



- Learning algorithm is presented inputs and desired outputs: **training data** $D = (in, out)$
- Goal: learn a general rule f that maps inputs to outputs $f(in) = out$

⇒ **Regression task**: *out* is a *continuous* number
  e.g. linear regression, polynomial regression

⇒ **Classification task**: *out* is a *nominal* number (class label)
  e.g. probabilistic (logistic regression)
  e.g. non-probabilistic (kNN, SVM, perceptron)

- No training data is given to the learning algorithm
- Goal: find structure data, discover hidden patterns, learn features

⇒ **Clustering task**
  e.g. K-means

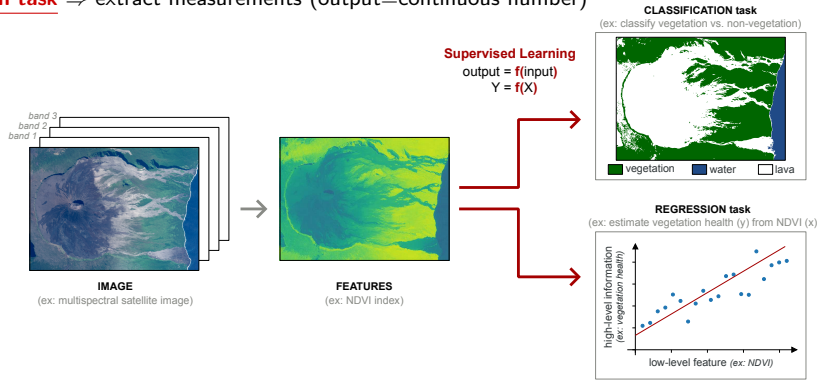
⇒ **Dimension reduction**
  e.g. PCA (→ also used to craft features)

Regression vs. classification

⇒ the goal of **supervised learning** is to learn a function f which maps low-level image features (X) to high-level image information (Y), using training data (i.e. known pairs of (X_i, Y_i)):

→ **classification task** ⇒ extract semantic classes (output=nominal number)

→ **regression task** ⇒ extract measurements (output=continuous number)



**the term "feature"* is here used in a broad sense, referring to any information extracted from the image

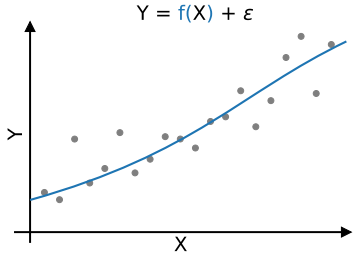
Regression model

$\Rightarrow$ we assume that two variables X & Y are ideally related by a function f:

$$Y = f(X) + \epsilon$$

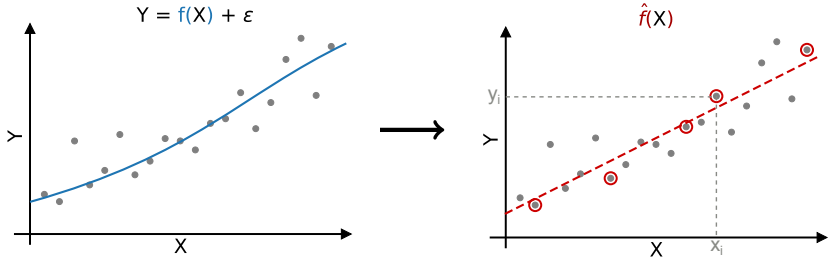
where:

- X = *input variable* (a.k.a. *independent variable*, or *feature*)
- Y = *output variable* (a.k.a. *dependent variable*, or *target variable*)
- ϵ = *random error* (intrinsic dataset error)



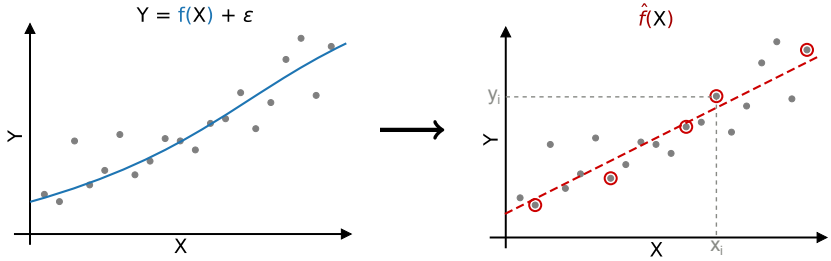
Regression model

⇒ goal: learn the **prediction function** $\hat{f}$ using a set of **training samples** (i.e. pairs of (x_i, y_i))



Regression model

⇒ goal: learn the **prediction function** $\hat{f}$ using a set of **training samples** (i.e. pairs of (x_i, y_i))



⇒ how: minimize a criterion (a.k.a. the **prediction error**, **cost function**, or **loss function**), which measures how well the predicted function f fits our training samples

→ for regression models, this metric is typically the **Mean Squared Error (MSE)**:

$$MSE(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

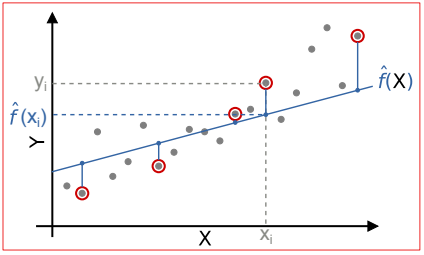
⇒ minimizing the MSE means finding function $\hat{f}$ that best fits the training samples

$$MSE(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

$$\hat{f} = \text{argmin}_{f \in \mathcal{M}} MSE(f)$$

high MSE

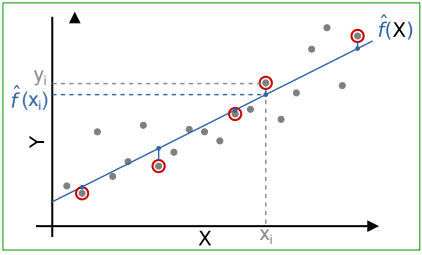
⇒ *bad prediction function* $\hat{f}(X)$



minimize MSE

low MSE

⇒ *good prediction function* $\hat{f}(X)$



* *the expression* $\text{argmin}_{f \in \mathcal{M}}$ *is mathematical notation for "the argument of the minimum", indicating we are trying to find the function* f *from a set of possible functions* $\mathcal{M}$ *that minimizes the MSE*

Parametric method: linear regression

⇒ parametric supervised learning means *we assume that* $\hat{f}$ *takes a specific form*, for example a linear relationship between X and Y:

$$\hat{f}(x) = \alpha x + \beta$$

Parametric method: linear regression

⇒ parametric supervised learning means *we assume that* $\hat{f}$ *takes a specific form*, for example a linear relationship between X and Y:

$$\hat{f}(x) = \alpha x + \beta$$

⇒ the prediction error $MSE(\hat{f})$ therefore depends on 2 parameters (α, β) which need to be determined:

$$\begin{aligned} E(\alpha, \beta) &= \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} (y_i - (\alpha x_i + \beta))^2 \end{aligned}$$

Parametric method: linear regression

$\Rightarrow$ parametric supervised learning means *we assume that* $\hat{f}$ *takes a specific form*, for example a linear relationship between X and Y:

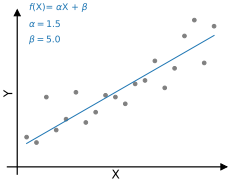
$$\hat{f}(x) = \alpha x + \beta$$

$\Rightarrow$ the prediction error $MSE(\hat{f})$ therefore depends on 2 parameters (α, β) which need to be determined:

$$\begin{aligned} E(\alpha, \beta) &= \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} (y_i - (\alpha x_i + \beta))^2 \end{aligned}$$

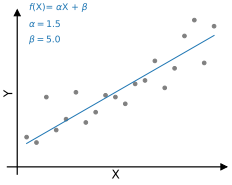
Parametric method: linear regression

$\Rightarrow$ for least-squares with a linear model, the error surface is convex, which means a unique minimum exists

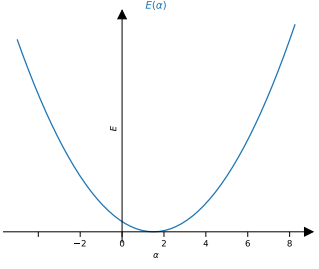


Parametric method: linear regression

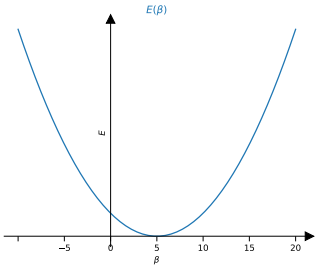
⇒ for least-squares with a linear model, the error surface is convex, which means a unique minimum exists



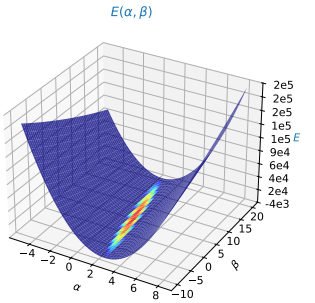
error on α



error on β



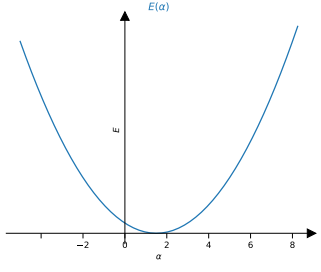
error surface $E(\alpha, \beta)$



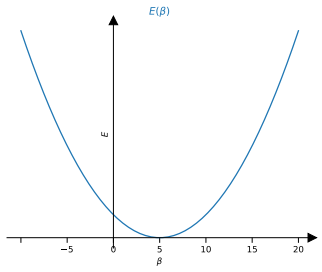
Parametric method: linear regression

⇒ for least-squares with a linear model, the error surface is convex, which means a unique minimum exists

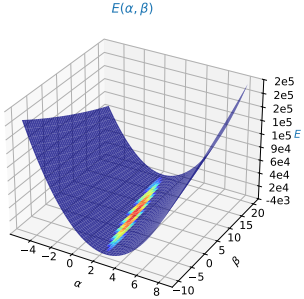
error on α



error on β



error surface $E(\alpha, \beta)$



⇒ solving for $dE/d\alpha = 0$ and $dE/d\beta = 0$ (slopes of error surface) allows for an analytical solution of $(\hat{\alpha}, \hat{\beta})$:

$$\hat{\alpha} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{(X - \bar{X}) \cdot (Y - \bar{Y})}{(X - \bar{X}) \cdot (X - \bar{X})} = \frac{cov(X, Y)}{var(X)}$$

$$\hat{\beta} = \bar{y} - \hat{\alpha}\bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the mean of x and y: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

The analytical solutions for (α, β) can be found using the **normal equation**:

$\Rightarrow$ the linear equation for a dataset with n observations is written as:

$$Y = \alpha X + \beta$$

where:

- $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ = nx1 vector of observed values x_i; $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ = nx1 vector of observed values y_i
- α = slope of the line
- β = intercept

The analytical solutions for (α, β) can be found using the **normal equation**:

$\Rightarrow$ the linear equation for a dataset with n observations is written as:

$$Y = \alpha X + \beta$$

where:

- $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ = nx1 vector of observed values x_i; $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ = nx1 vector of observed values y_i
- α = slope of the line
- β = intercept

$\Rightarrow$ the linear equation can be written in matrix form:

$$Y = X\theta$$

where:

- $X = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$ = nx2 matrix of ones & values of x_i; $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ = nx1 vector of values y_i

  NB: the first column in X are all ones to account for the intercept β
- $\theta = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$ = 2x1 vector of coefficients (β, α)

The analytical solutions for (α, β) can be found using the **normal equation**:

$\Rightarrow$ the linear equation for a dataset with n observations is written as:

$$Y = \alpha X + \beta$$

where:

- $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ = nx1 vector of observed values x_i; $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ = nx1 vector of observed values y_i
- α = slope of the line
- β = intercept

$\Rightarrow$ the linear equation can be written in matrix form:

$$Y = X\theta$$

where:

- $X = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$ = nx2 matrix of ones & values of x_i; $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ = nx1 vector of values y_i

  NB: the first column in X are all ones to account for the intercept β
- $\theta = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$ = 2x1 vector of coefficients (β, α)

$\Rightarrow$ now $\hat{\theta} = (\hat{\beta}, \hat{\alpha})$ is estimated by minimizing the least squares error, which results in the following solution:

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

Reminder: matrix multiplication

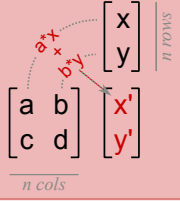
The matrix equation:

$$X' = \theta X$$

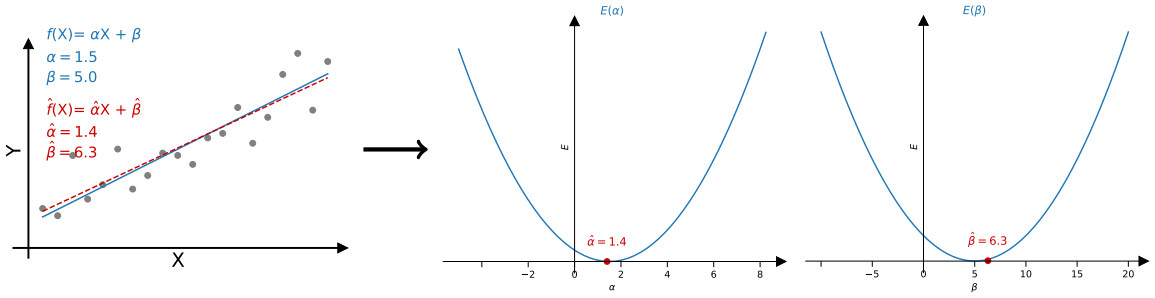
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Can we written as a linear system of equations:

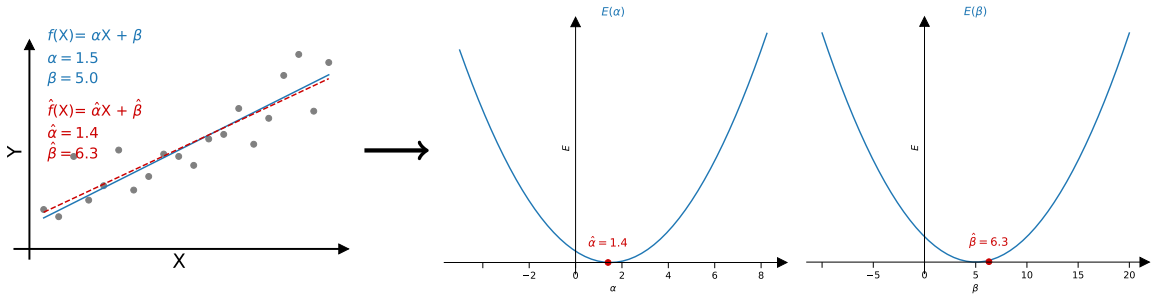
$$\begin{cases} x' = ax + by \\ y' = cx + dy \end{cases}$$



⇒ **Error surface** of the coefficients (α, β) and estimated values $(\hat{\alpha}, \hat{\beta})$:



⇒ **Error surface** of the coefficients (α, β) and estimated values $(\hat{\alpha}, \hat{\beta})$:

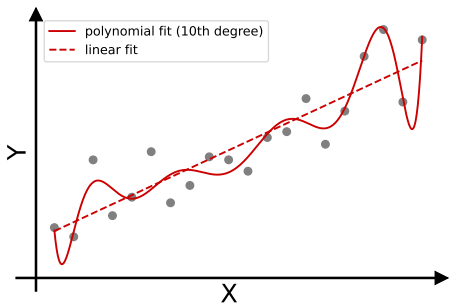


⇒ Note: when no analytical solution exists, the error is minimized using *iterative optimization methods* such as **gradient descent** → *next week*

What if linear regression is not enough?

⇒ **polynomial regression** is a form of linear regression, where the relationship between the independent variable X and the dependent variable Y is modeled as an n^{th} degree polynomial:

$$\hat{f}(x) = \alpha_0 + \alpha_1 x^1 + \alpha_2 x^2 + \ldots + \alpha_n x^n$$



Polynomial regression as a linear model

⇒ **polynomial regression** is a form of linear regression, where the relationship between the independent variable X and the dependent variable Y is modeled as an n^{th} degree polynomial:

$$\hat{f}(x) = \alpha_0 + \alpha_1 x^1 + \alpha_2 x^2 + \ldots + \alpha_n x^n$$

→ we are effectively mapping the feature x in a higher dimensional feature space $x' = (x, ..., x^n)$

→ if we treat x^n as a *new features*, we can think of this as a linear equation in the transformed feature space:

$$y = \alpha_0 + \alpha_1 \cdot (new\_feature\_1) + ... + \alpha_n \cdot (new\_feature\_n)$$

→ this allows us to use linear regression to fit the polynomial model!

Polynomial regression as a linear model

$\Rightarrow$ linearity *in terms of coefficients* allows us to use *linear regression to fit polynomial data of degree d*:

$\rightarrow$ the equation is **non-linear with respect to** x because of the higher-order terms $(x^2, \ldots, x^d)$
$\Rightarrow$ *the model can capture non-linear relationships between X and Y*

$\rightarrow$ however, the equation is **linear with respect to the coefficients** $(\alpha_0, \ldots, \alpha_d)$
$\Rightarrow$ *the model can be solved using linear regression (coefficients found analytically using the normal equation)*:

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

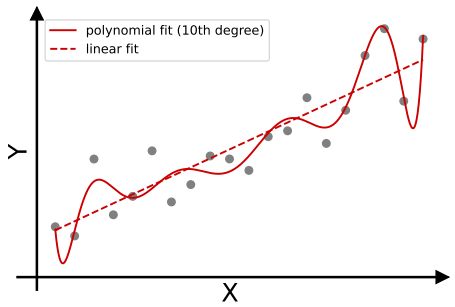
where:

- $X = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^d \end{pmatrix} = n \times (d+1)$ matrix of ones and powers of x_i
- $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = n \times 1$ vector of values y_i
- $\hat{\theta} = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_d \end{pmatrix} = (d+1) \times 1$ vector of coefficients α_i

Polynomial regression as a linear model

⇒ advantages: can model non-linear relationships, more flexible than linear regression

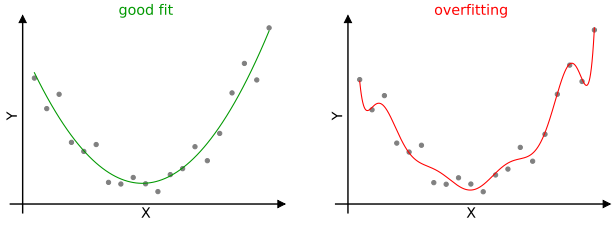
⇒ drawbacks: more complex, more prone to **overfitting**



What is overfitting & underfitting?

overfitting $\Rightarrow$ model is too complex

$\rightarrow$ captures the noise in the training data
$\rightarrow$ does not generalize well to new data



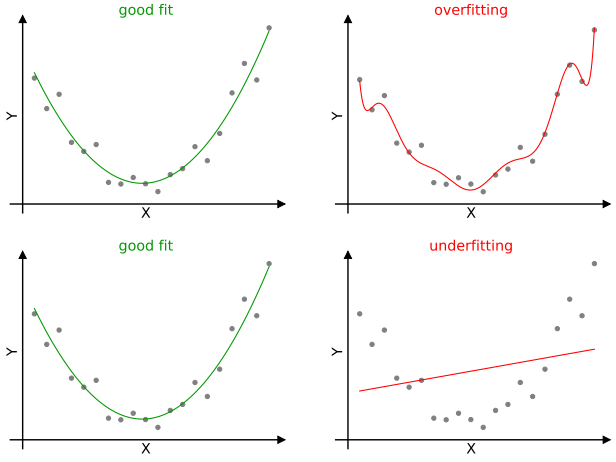
What is overfitting & underfitting?

overfitting ⇒ model is too complex

→ captures the noise in the training data
→ does not generalize well to new data

underfitting ⇒ model is too simple

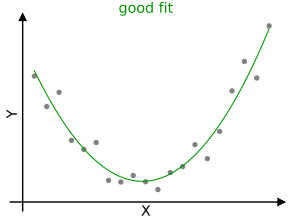
→ does not capture the underlying structure of the data

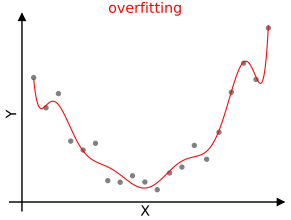


What is overfitting & underfitting?

overfitting ⇒ model is too complex

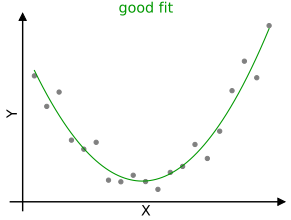
→ captures the noise in the training data
→ does not generalize well to new data

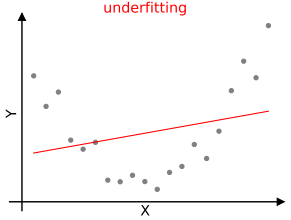




underfitting ⇒ model is too simple

→ does not capture the underlying structure of the data





⇒ optimal model complexity & ability to generalize to unseen data?

Bias-variance trade-off

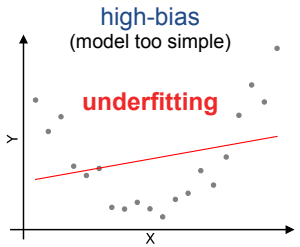
A model's generalization error can be expressed as the sum of three different errors:

- irreducible error: error due to the noisiness of the data itself

Bias-variance trade-off

A model's generalization error can be expressed as the sum of three different errors:

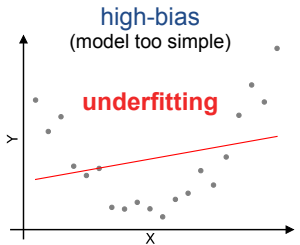
- irreducible error: error due to the noisiness of the data itself
- **bias**: error due to overly simplistic assumptions in the model
  → *high-bias* model ⇒ model is too simple (e.g., linear model for quadratic data), prone to *underfitting*

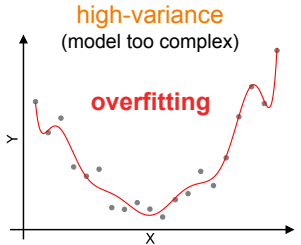


Bias-variance trade-off

A model's generalization error can be expressed as the sum of three different errors:

- irreducible error: error due to the noisiness of the data itself
- **bias**: error due to overly simplistic assumptions in the model
  → *high-bias* model ⇒ model is too simple (e.g., linear model for quadratic data), prone to *underfitting*
- **variance**: error due to excessive sensitivity to small fluctuations in the training data
  → *high-variance* model ⇒ model with many degrees of freedom (e.g. high-deg. polynomial), prone to *overfitting*

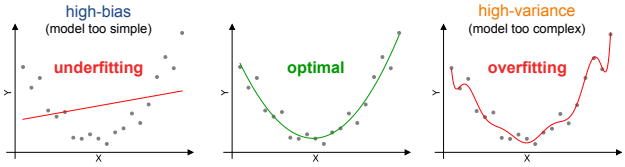




Bias-variance trade-off

⇒ *as the model complexity increases, bias decreases but variance increases*

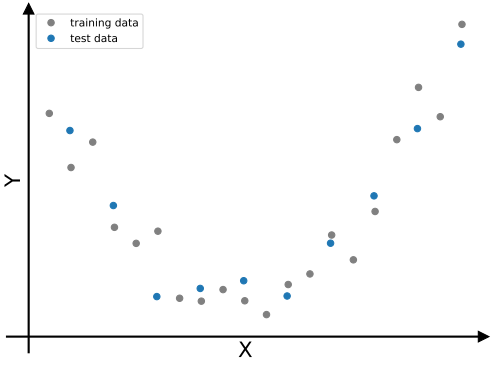
⇒ the optimal model complexity results from a trade-off between bias and variance:

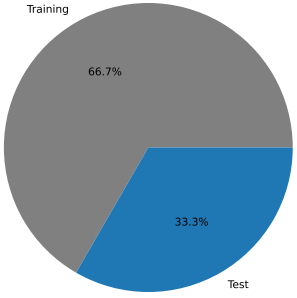


How can we evaluate the model capability to generalize?

⇒ we divide the dataset into 2 subsets *(sometimes more, we'll see that later)*:

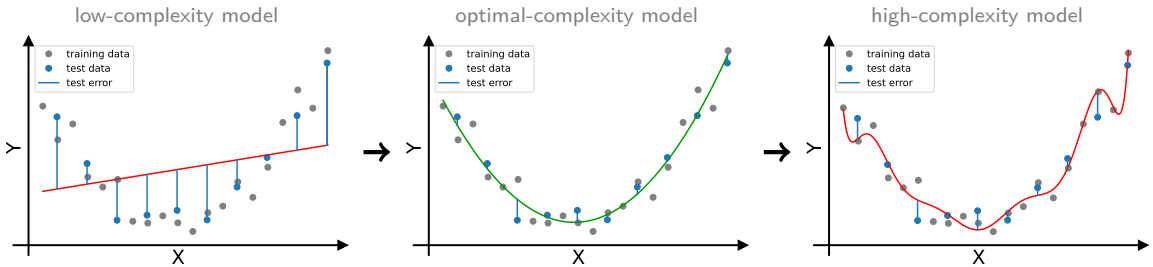
- **training set**: used to train the model (i.e. estimate the coefficients)
- **test set**: used to evaluate the model's performance on unseen data





How can we evaluate the model capability to generalize?

⇒ with increasing model complexity, the *training error decreases*, while the *test error first decreases, then increases*



How can we evaluate the model capability to generalize?

⇒ with increasing model complexity, the *training error decreases*, while the *test error first decreases, then increases*