CS286: AI for Science and Engineering

Lecture 2: Machine Learning Landscape

Jie Zheng (郑杰)

PhD, Associate Professor

School of Information Science and Technology (SIST), ShanghaiTech University

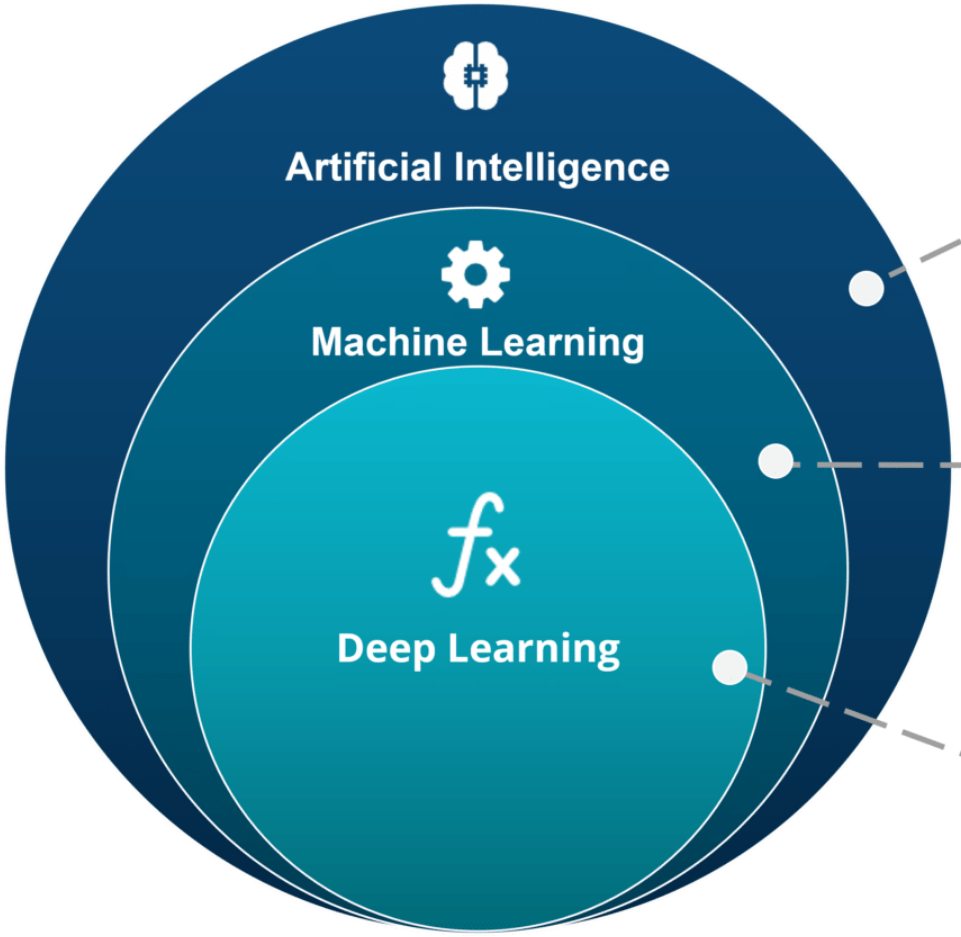
Fall, 2023

Learning objectives

- After taking this lecture, you should be able to:
  - Define what machine learning is
  - Describe paradigms of machine learning
  - Explain when and why use machine learning
  - Explain how to measure the performance of machine learning methods
  - Tell what are the main challenges of machine learning
  - Describe how to train a machine learning model using regression as an example

Introduction to Machine Learning

Machine learning as a subfield of AI



ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

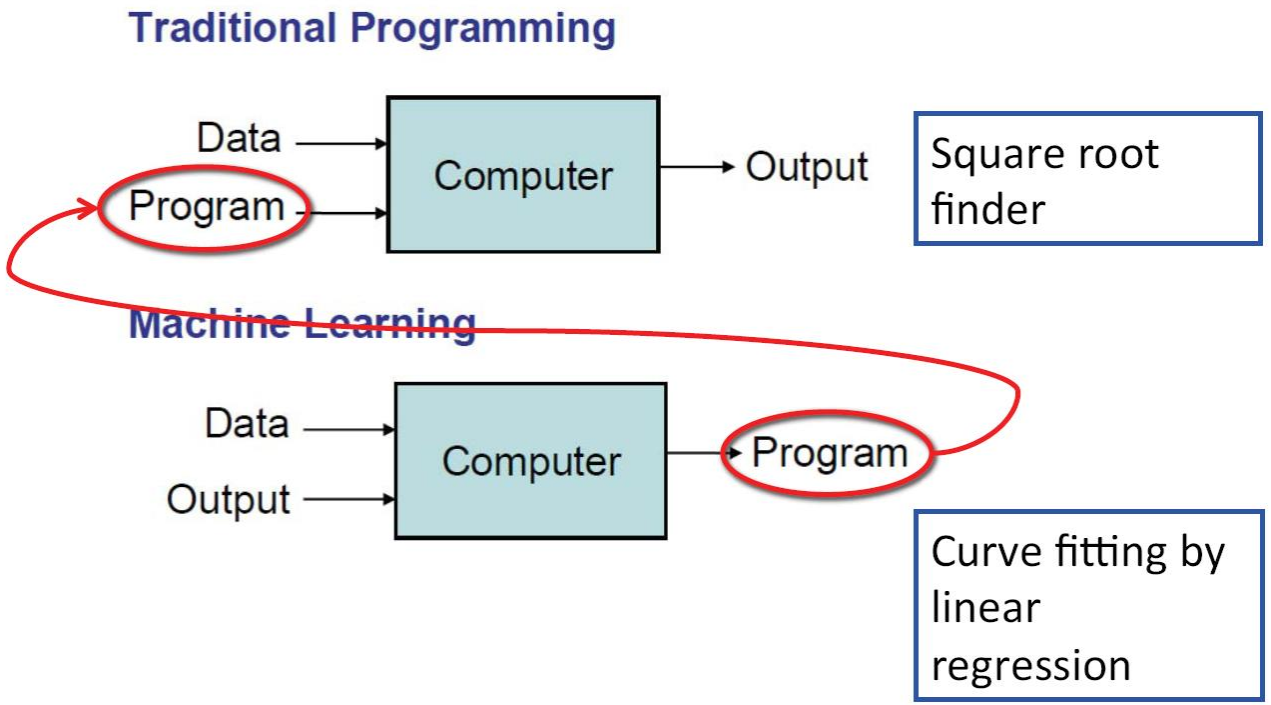
MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Figure credit: https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/

What is machine learning?

- **Machine learning** is a "field of study that gives computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)
- Program by example: Rather than program the computer to solve a task directly, with machine learning the computer will come up with its own program based on examples given to it

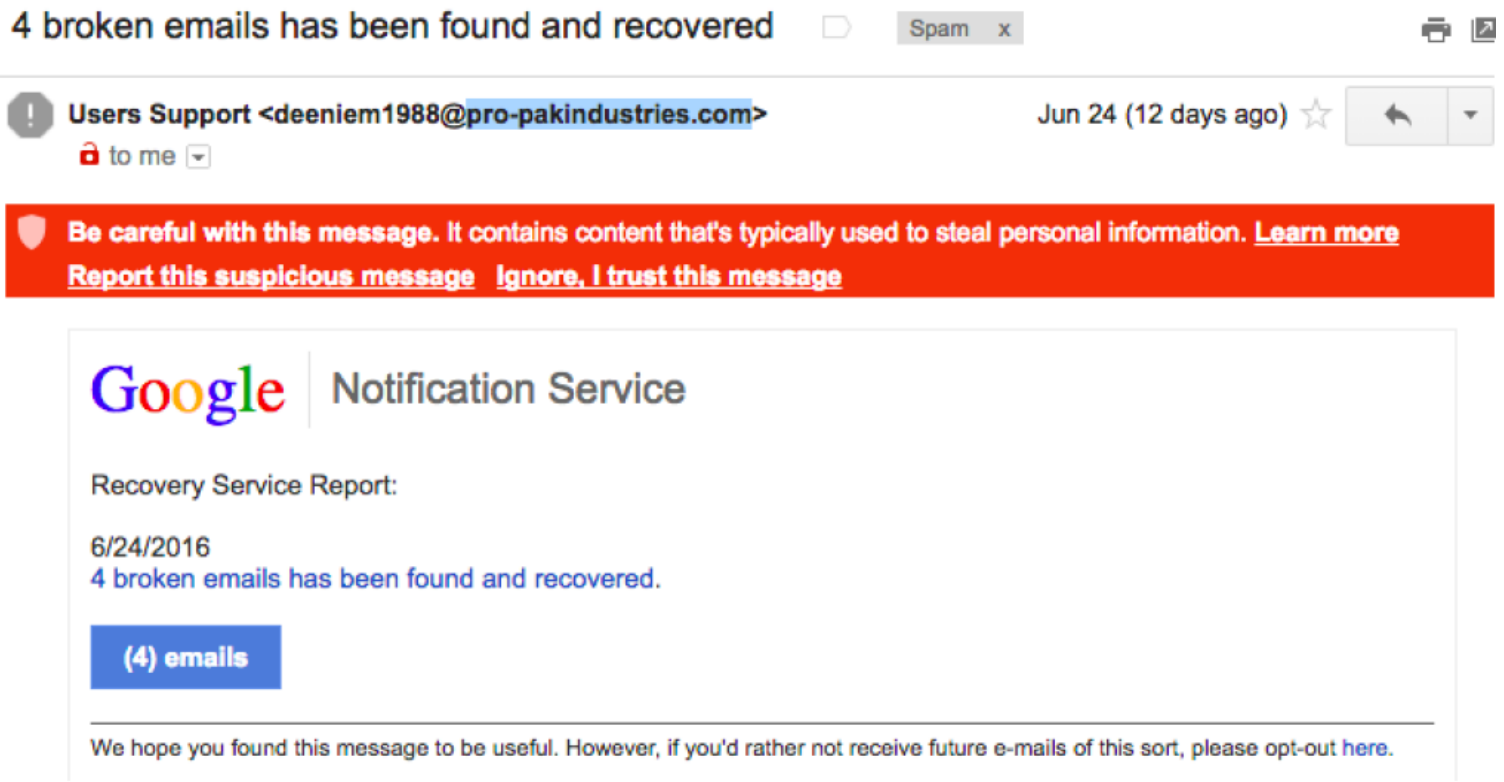


What is machine learning? (continued)

- Engineering-oriented view: A computer program is said to **learn from experience E with respect to some task T and some performance measure P**, if its performance on T, as measured by P, improves with experience E (Tom Mitchell, 1997)
- A common theme is to solve a prediction problem:
  - Given an **input** x
  - Predict an "appropriate" **output** y
- Let us start with a canonical example

Example: Spam filtering

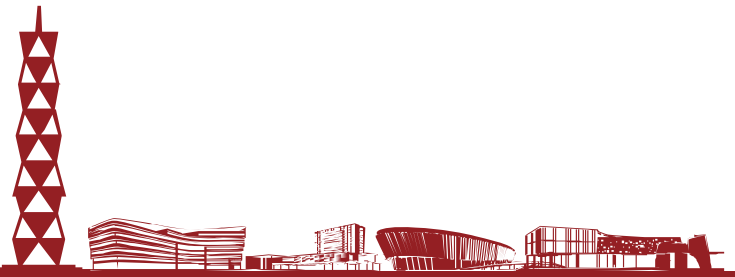
- **Input**: Incoming emails



- **Output**: "SPAM" or "NOT SPAM"
- A **binary classification** problem, i.e. having only 2 possible outputs

Example: Spam filtering (continued)

- **Task T**: To flag spam for new emails
- **Experience E**: The training data
  - Examples of spam emails (e.g. flagged by users)
  - Examples of regular (nonspam) emails
- **Performance measure P**: To be defined
  - e.g. **accuracy** (ratio of correctly classified emails)

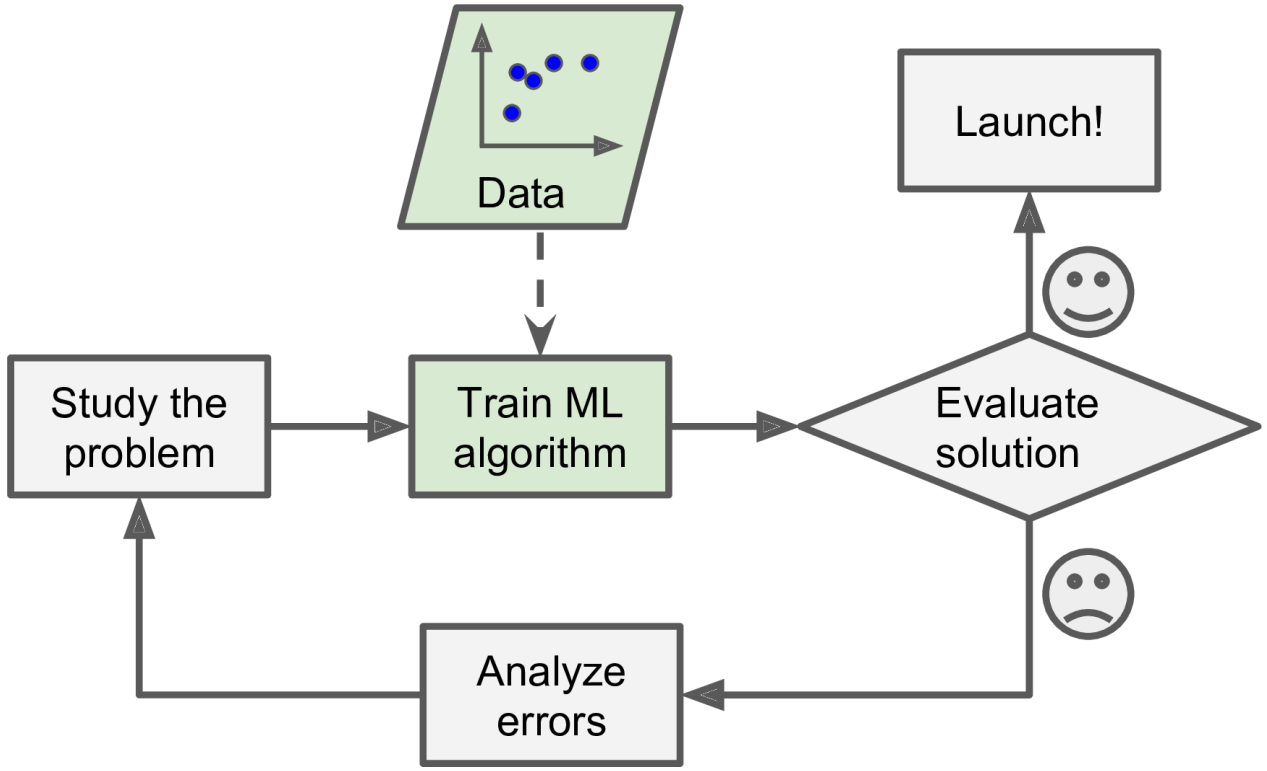


The prediction function

- A **prediction function** takes input x and produces an output y
- We look for prediction functions that solve particular problems
- Machine learning techniques aim to find the **best** prediction function

Machine learning algorithms

- A machine learning algorithm:
  - **Input**: Training data
  - "Learns" from the training data
  - **Output**: A prediction function that produces output y given x



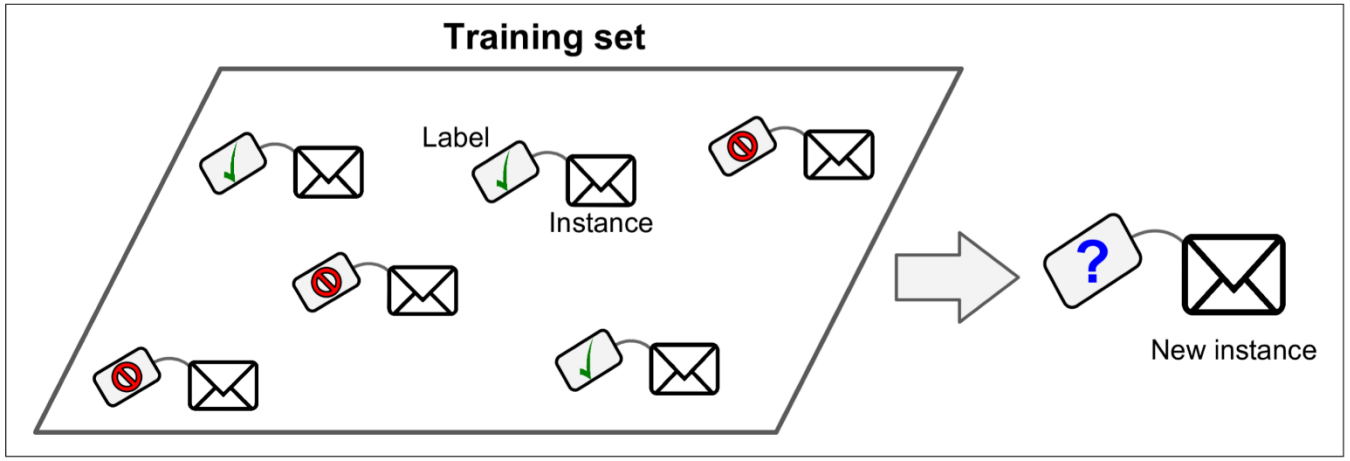
Paradigms of machine learning

- **Supervised learning**: Given a set of feature/label pairs, find a rule (or function) that predicts the label associated with a previously unseen input
  - Examples: classification, regression
  - Semisupervised learning: Training dataset is partially labeled
- **Unsupervised learning**: Given a set of feature vectors (without labels), group them into "natural clusters" or create labels for the groups
  - Example: clustering
- **Reinforcement learning**: An area of machine learning concerned with how an agent ought to take actions in an environment so as to maximize some notion of cumulative reward. The algorithm (or agent) learns by interacting with its environment.

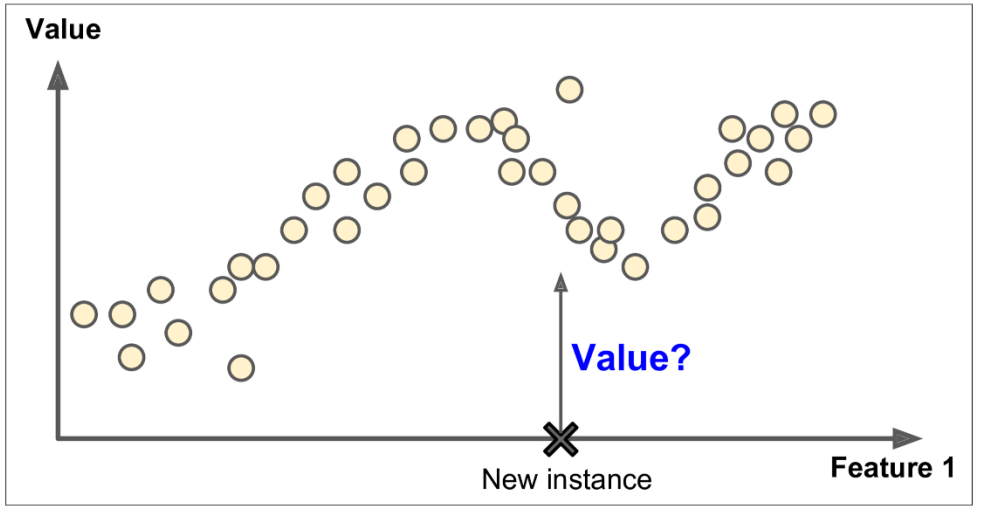
Performance Measures

Classification vs. regression

- **Classification** is a typical supervised learning task, e.g. the spam filter is trained with many example emails along with their class (spam or not spam) and it must learn how to classify new emails
- **Regression** is a task to predict a target numeric value, given a set of features called predictor, e.g. to predict the price of a car given its mileage, age, brand, etc.



Spam classification



Regression

How to measure the performance of a classification model (i.e. classifier)?

Training/test splitting

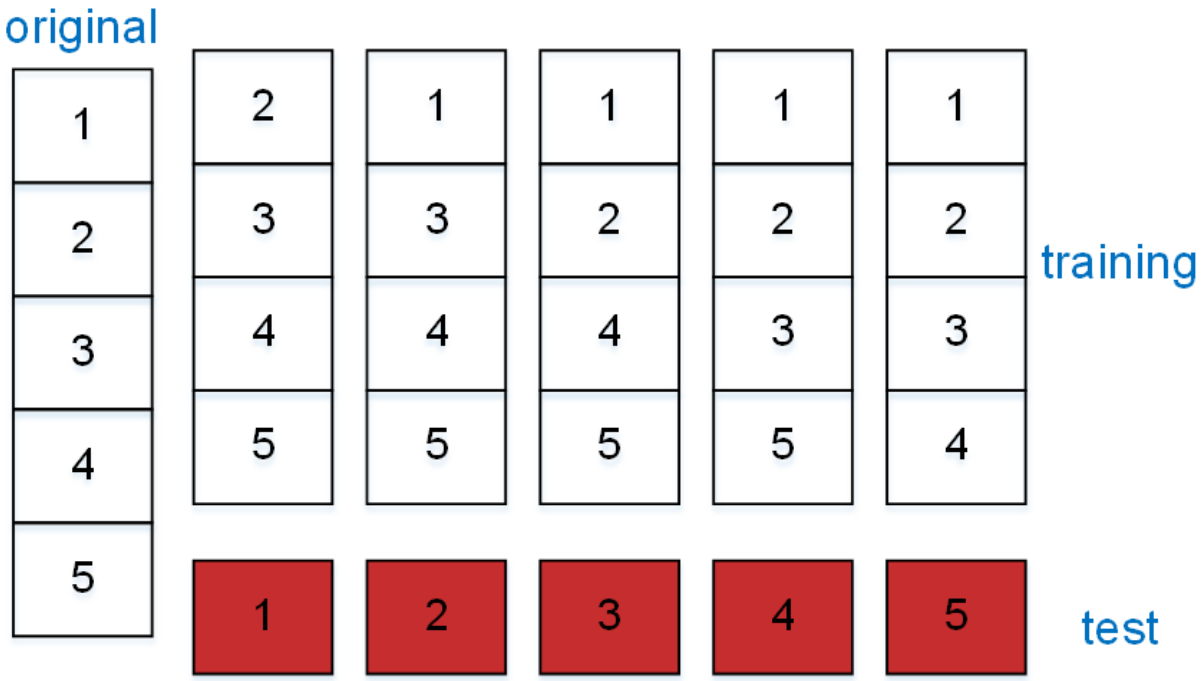
- Training set: only for **training prediction functions**
- Test set: only for **evaluating model performance**, and must be independent of training data
- Training/test set is usually **randomly** split
- But if the data contain **time series**, two sets should be created by splitting in time
  - Training set is everything before time T
  - Test set is everything after time T

Cross-validation

- An intuitive method to test the model when training:
  - Randomly make *training* / *validation* split on the original training data
  - Perform the machine learning algorithm using the *training* set
  - Estimate the "future" result with the *validation* set
- A very simple evaluation method ☺
- We might waste too much training data, and both too large or too small datasets will cause the model evaluation imprecise ☹

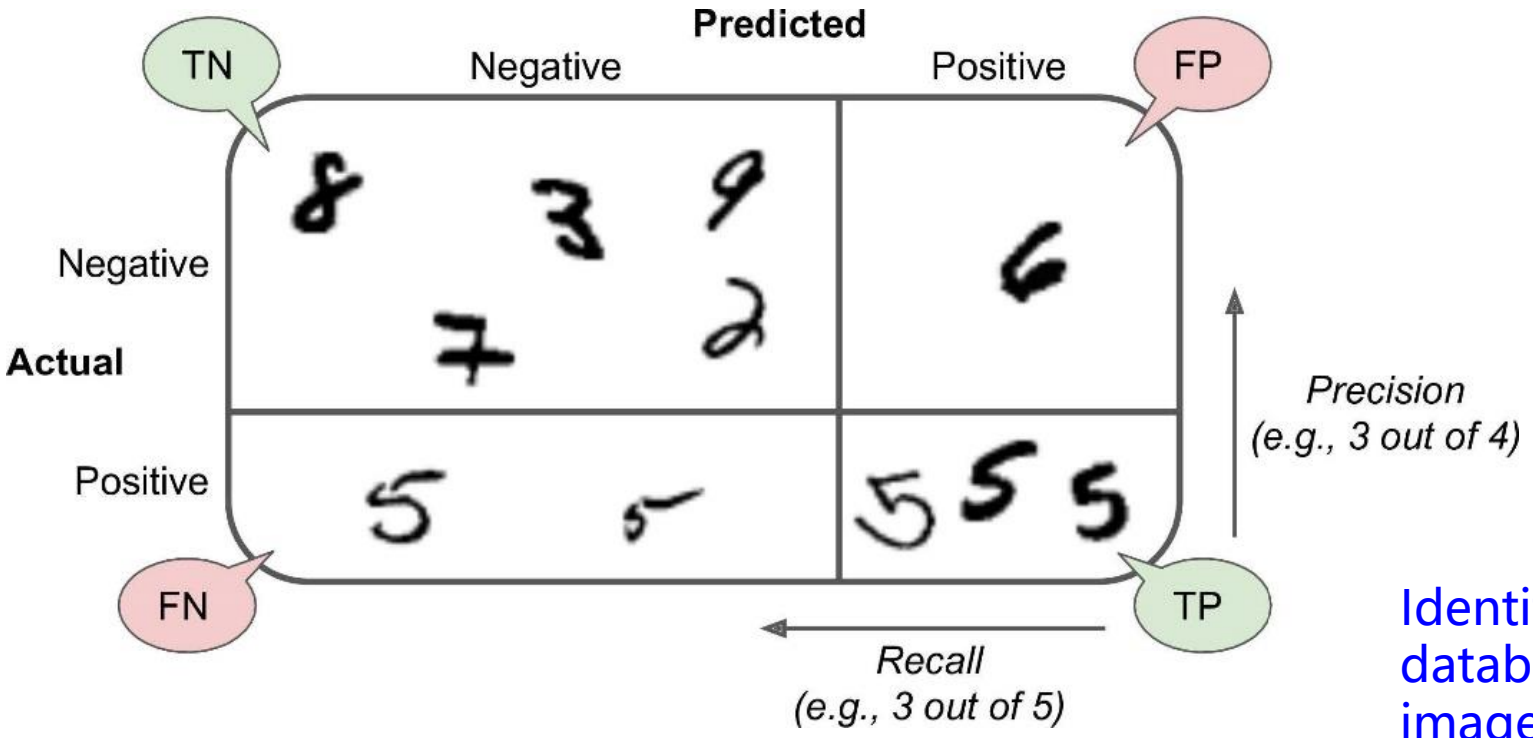
Cross-validation

- Make full use of a dataset with a k-**fold cross-validation**:
  1. uniformly divide data into k blocks
  2. for $j = 1$ to k
  3. train on blocks except for the jth block
  4. validate on the jth block
  5. evaluate the result
  6. average the results



Confusion matrix

- To evaluate a classifier, **accuracy** (i.e. ratio of correct predictions) may be biased for **skewed** datasets (e.g. when some classes are much more frequent than others)
- A better metric for classifier performance is to count the number of instances of class A classified as class B, e.g. for a **binary** classifier:



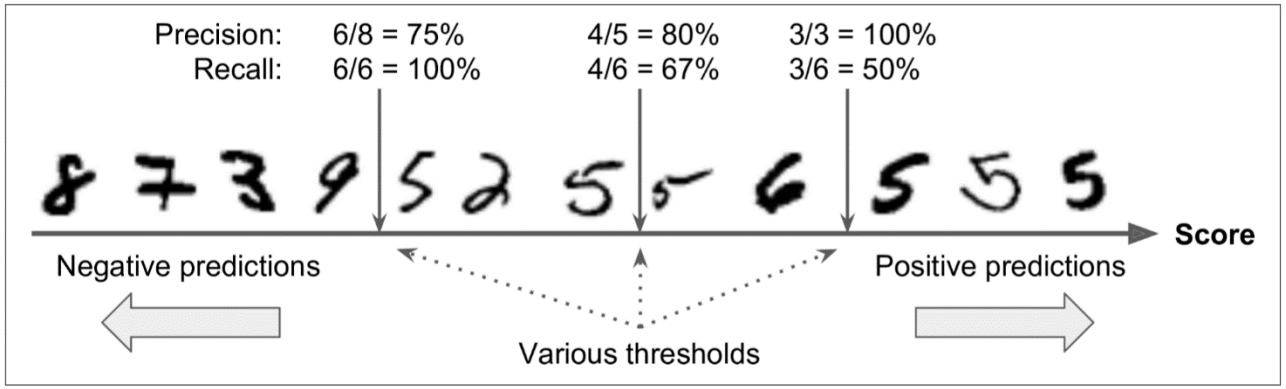
TP: true positive
TN: true negative
FP: false positive
FN: false negative

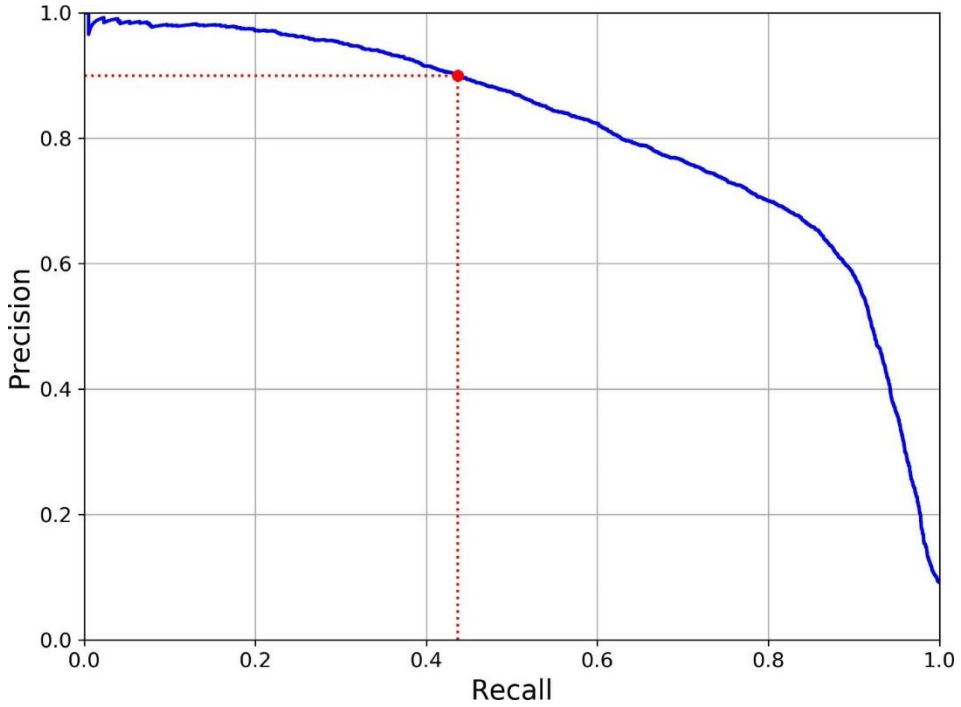
Identify digit 5 from the MNIST database of handwritten digit images (Fig. 3-2 of A. Géron book)

Precision/Recall tradeoff

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

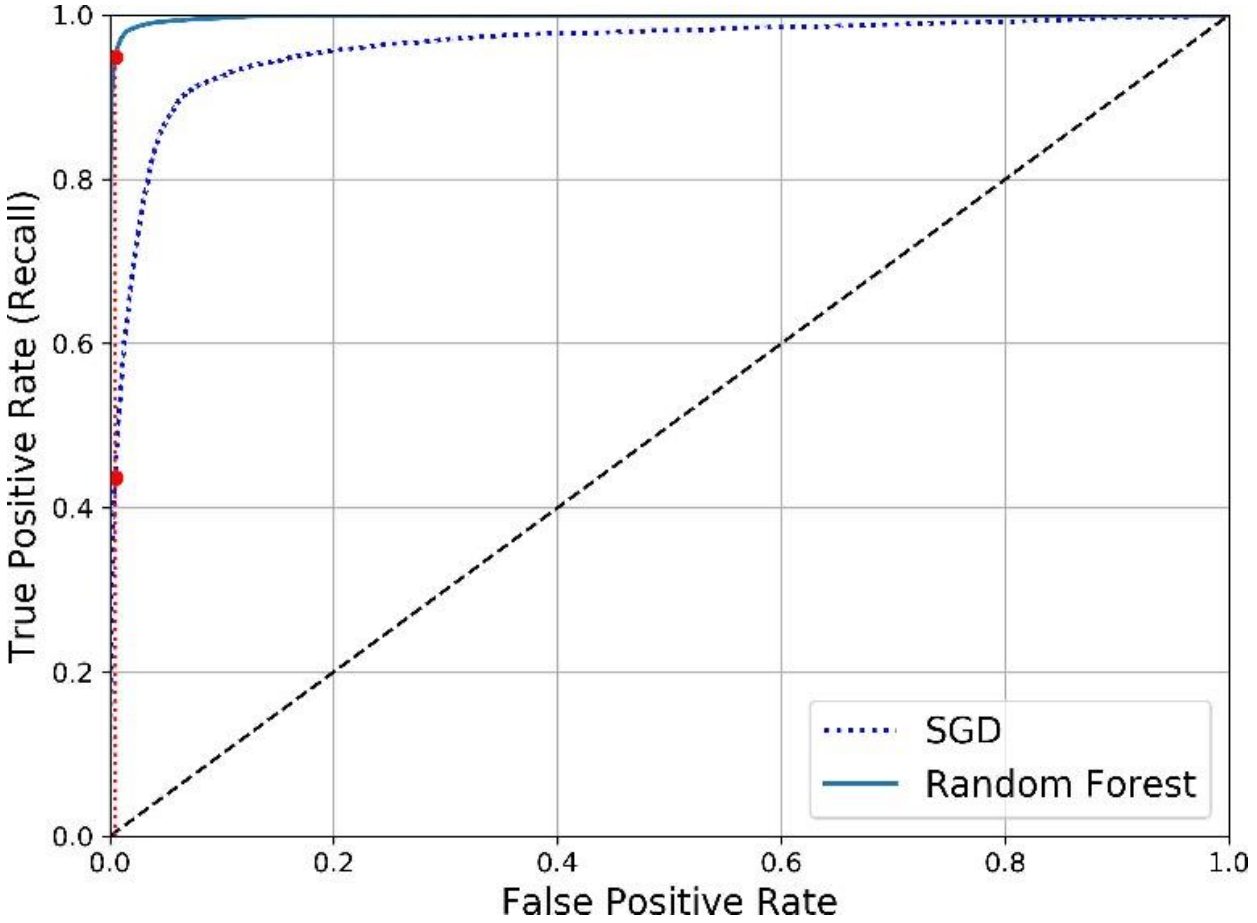
- Adjusting the **decision threshold** (i.e. if score is above such a threshold it is assigned positive, or else to negative) would change relative values on the **precision/recall (PR) curve**





ROC (receiver operating characteristic) curve

- Common used for binary classifiers: TP rate (recall) vs FP rate
- Ideal classifiers have ROC AUC (area under the curve) equal to 1



$$TPR = TP / (TP + FN)$$

$$FPR = FP / (FP + TN)$$

Hyperparameters (or “Tuning Parameters”)

- Almost every learning algorithm has
  - At least one “hyperparameter” or “tuning parameter”
- You must tune these values
- Hyperparameters control various things
  - Model complexity (e.g. polynomial order)
  - Type of model complexity control (e.g. L1 vs L2 regularization)
  - Optimization algorithm (e.g. learning rate)
  - Model type (e.g. loss function, kernel type, …)

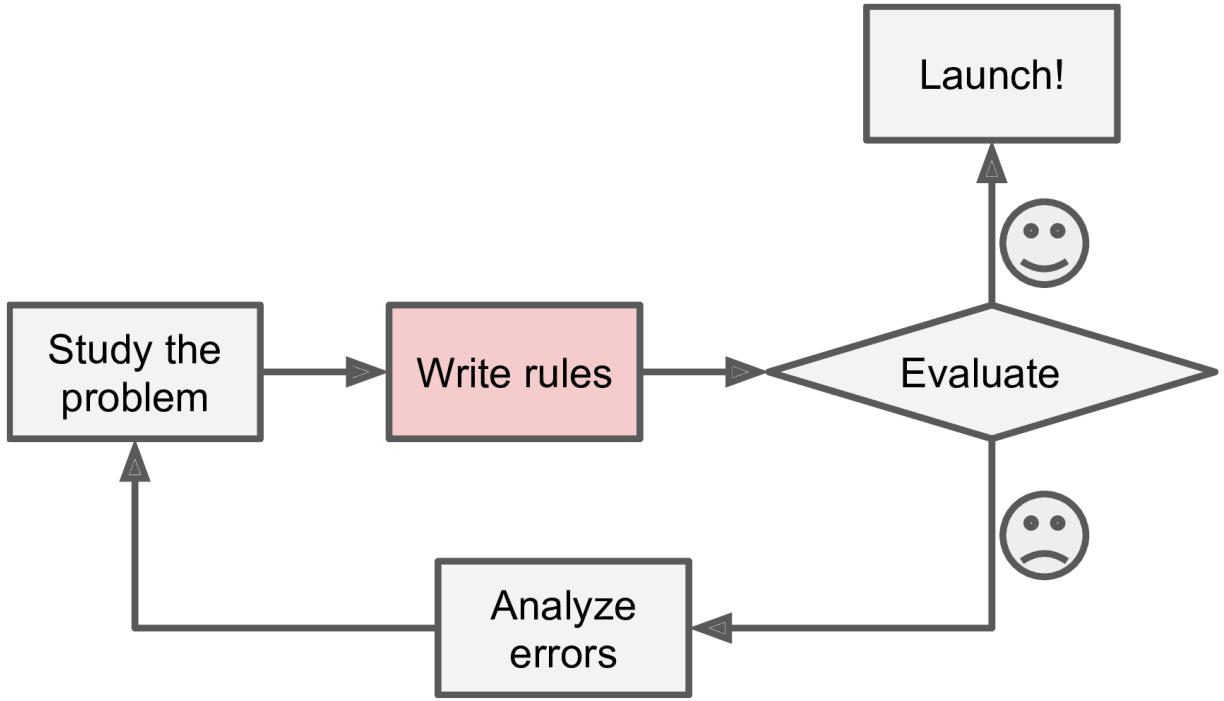
Strengths of Machine Learning

Why machine learning?

- Some problems are too difficult to program by hand:
  - Fuzzy and “noisy” patterns
  - Large amounts of data
  - Mechanisms are still unclear
  - Lack general theories to guide programming
- Natural sciences (such as cancer biology) are often faced with the above challenges
- Other applications:
  - Face recognition
  - Spam filtering
  - Medical diagnosis
  - Natural language processing
  - Product recommendation
  - Fraud detection
  - Weather forecasting

Traditional rule-based approaches

- A traditional method for spam filtering:
  - Study what spam typically looks like
  - A spam email usually contains some special key words in the subject, e.g. “4U”, “credit card”, … (encoded as rules)
  - Write and test a spam detection algorithm



Issues with rule-based approaches

- Very labor intensive to build a system
- End up with a long list of rules, which is hard to maintain
- Rules may work very well for specific areas they cover, but cannot naturally handle uncertainty

Machine learning approach

- A machine "learns" on its own without manually designed rules
- We provide "training data", i.e. many examples of (input x, output y), e.g.
  - A set of emails, and whether or not each is SPAM
  - A set of images, and whether or not each has a bird

Strengths of machine learning

- Usually simplify code and perform better for complex problems
- Machine learning systems can adapt to new data
- Can help extract patterns and insights from large amounts of data, i.e. automating knowledge discovery

Challenges of Machine Learning

Main challenges

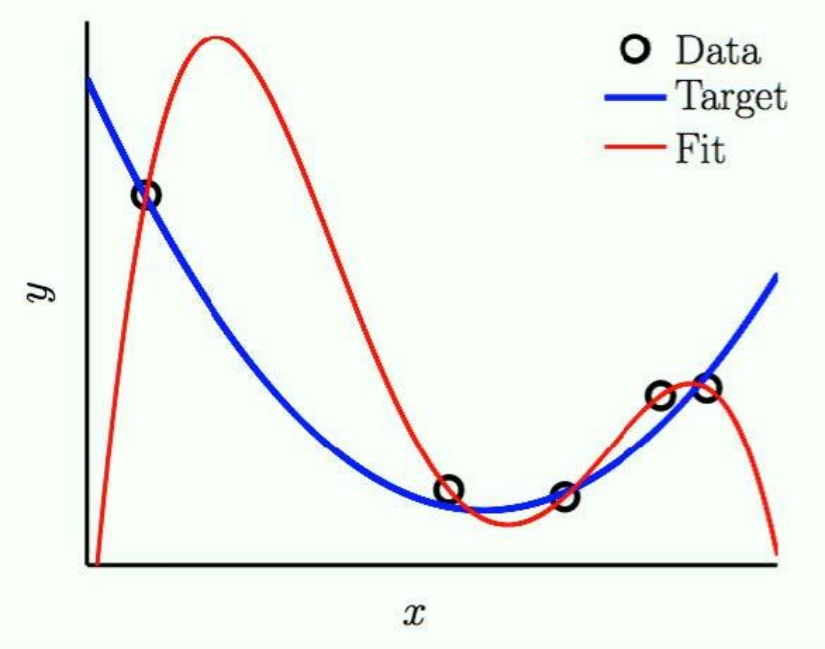
- **Insufficient quantity of training data**: More data would provide more complete features learned by the model, and thus generate more accurate results
- **Nonrepresentative training data**: Representative training data could generalize the learning algorithms to the expected new cases
  - e.g. sampling bias due to “unfair” sampling methods
- **Poor-quality data**: Errors, outliers, missing values or noise in the data will make it harder for the system to detect underlying patterns

Main challenges (continued)

- Dependency on data:
  - Data might be insufficient, noisy or biased
  - Garbage in, garbage out
- A model is often a "black box": More interpretability is needed

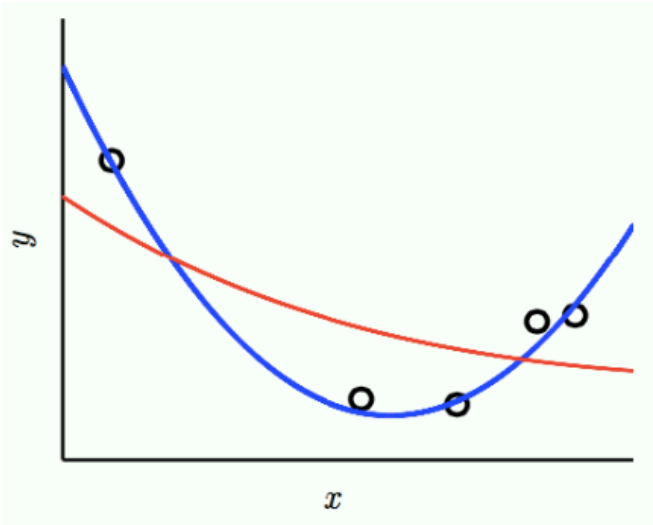
Overfitting

- **Overfitting the training data**:
  - Performance is good on the training data
  - But it does not generalize well
- Potential reasons:
  - **High dimensionality** (too many features)
  - **Model is too complex** (too many parameters)
  - Noise in the training data



Underfitting

- A model **underfits the training data** when it is too simple to learn the underlying structure of the data
- The model
  - cannot fit much of the data, and has poor training performance
  - should be made more powerful to catch richer features

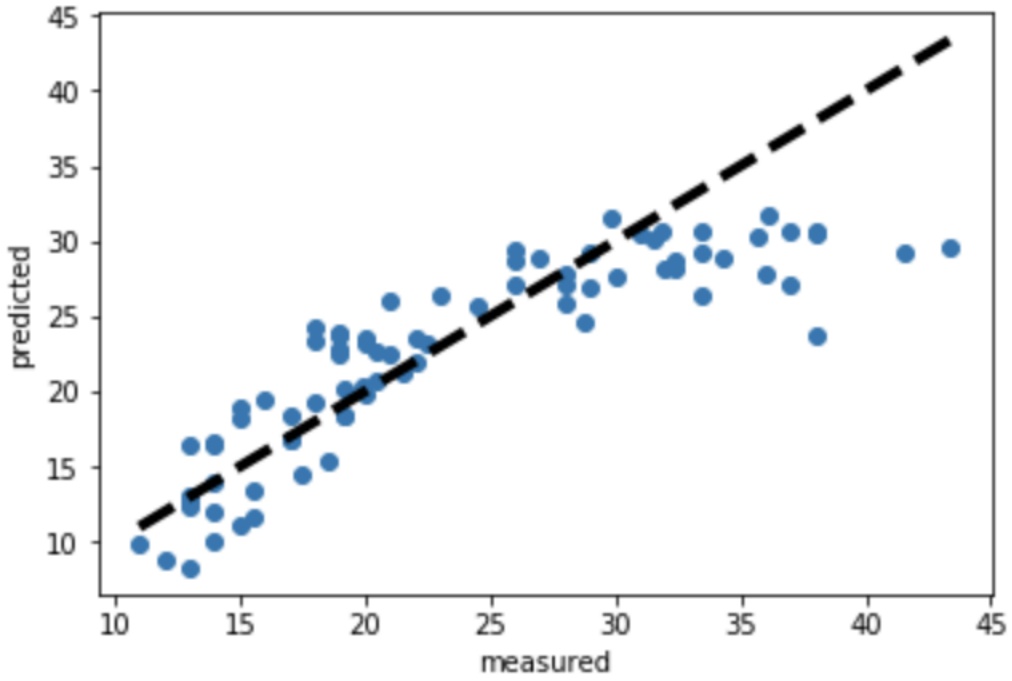


A model with only one parameter (hair length): Sun Y.S. is a boy.

Training Regression Models

Definition of regression

- Given a set of attributes $\boldsymbol{x}: (x_1, x_2, x_3, \ldots, x_n)$ of an object, estimate the mapping function from input $\boldsymbol{x}$ to a continuous output variable y based on training examples.



What is linear regression?

- **Linear Regression model** : A linear model makes a prediction by simply computing a weighted sum of the input features $\boldsymbol{x}$:

$$\hat{y} = h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \ldots \theta_n x_n$$

  - $\hat{y}$: predicted value
  - n : number of the features
  - x_i : the i^{th} feature value
  - θ_j : the j^{th} model parameter (include bias term θ_0)

Linear regression model

- A linear regression model (in **vectorized** form):

$$\hat{y} = \boldsymbol{\theta} \cdot \boldsymbol{x} = \boldsymbol{\theta}^T \boldsymbol{x}$$

- $\boldsymbol{\theta}$: model parameter vector, containing a bias term and feature weights from θ_0 to θ_n
- $\boldsymbol{x}$: feature vector of an instance (with $x_0 = 1$)
- $\boldsymbol{\theta} \cdot \boldsymbol{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and $\boldsymbol{x}$ which is equal to : $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \theta_n x_n$

Linear regression's objective

- How to obtain a proper linear regression model from training examples?
  - Training a model means setting its parameters so that the model best fits the training dataset
- What does "best fits" mean?
  - The most common performance measure of a regression model is the **Root Mean Square Error (RMSE)**:

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

  - In practice, it is simpler to minimize the **Mean Square Error (MSE)**:

$$\mathrm{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

  - Lower RMSE or MSE scores represent better model fitting

Linear regression's objective (continued)

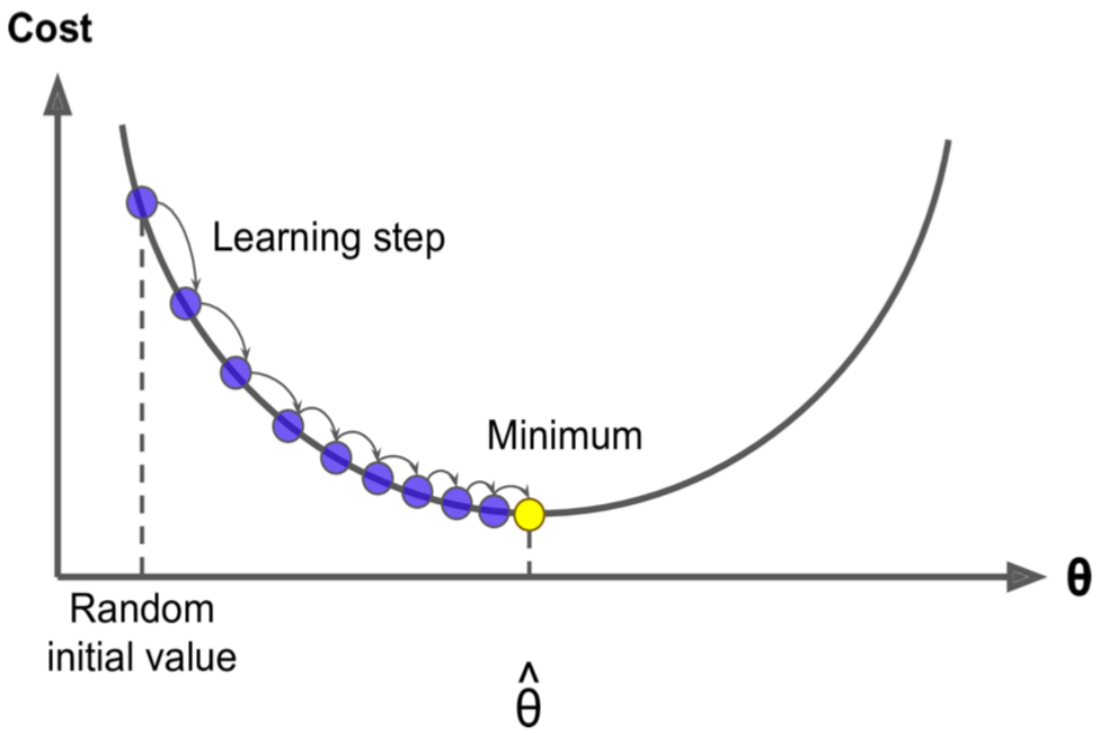
- After combining the MSE and linear regression model, we can build a cost function :

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

  - Our objective is to **minimize the MSE cost** by tuning the parameter θ, i.e. to make the model fit to the training samples.

Gradient Descent

- Goal: To decrease the MSE cost via updating parameter θ
- Approach: **Gradient Descent**
  - Filling θ with random values
  - Keep changing θ to reduce the objective function $J(\theta)$



Gradient Descent

- How to change the parameter $\boldsymbol{\theta}$?

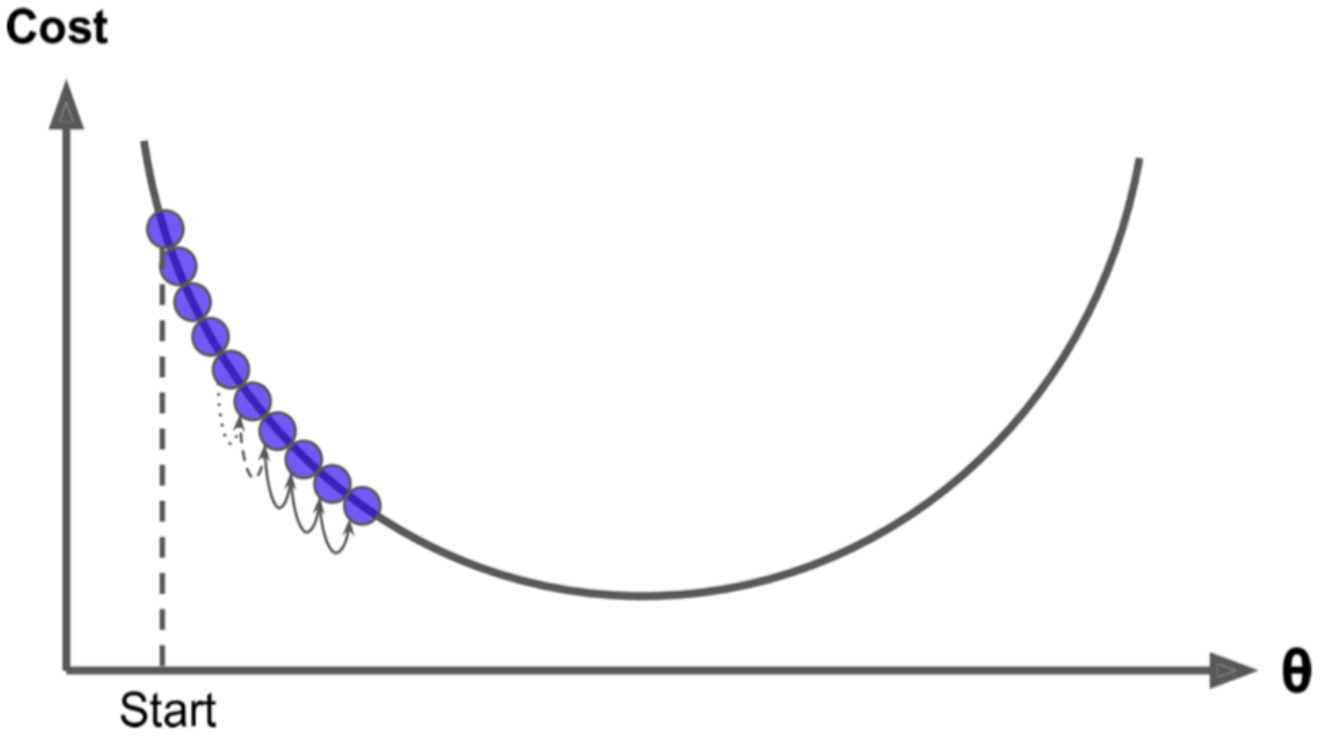
$$\theta_j^{k+1} = \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}^k)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}^k) = \frac{2}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

- k : iteration count
- θ_j^k : the j^{th} parameter in the k^{th} iteration
- α : step size (also known as **learning rate**)

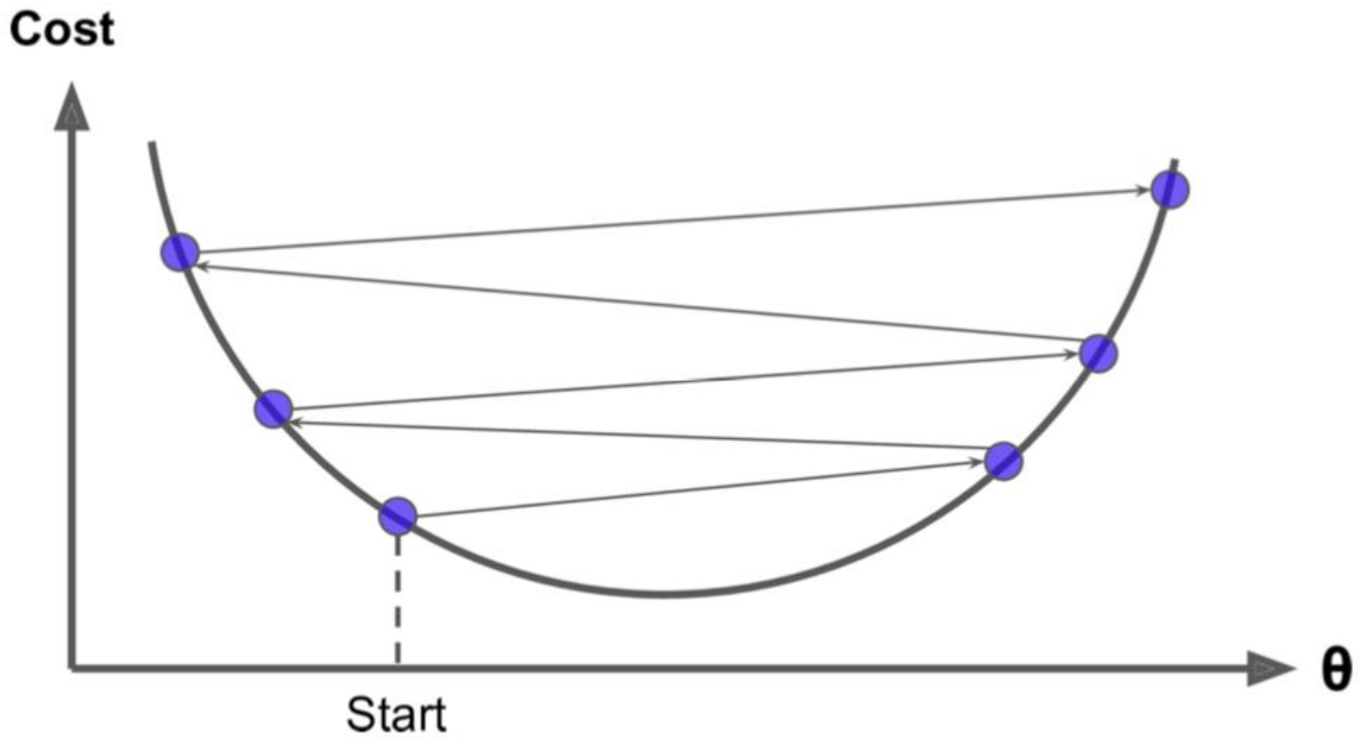
Learning rate in Gradient Descent

- If the learning rate is too small, it is hard to converge



Learning rate in Gradient Descent

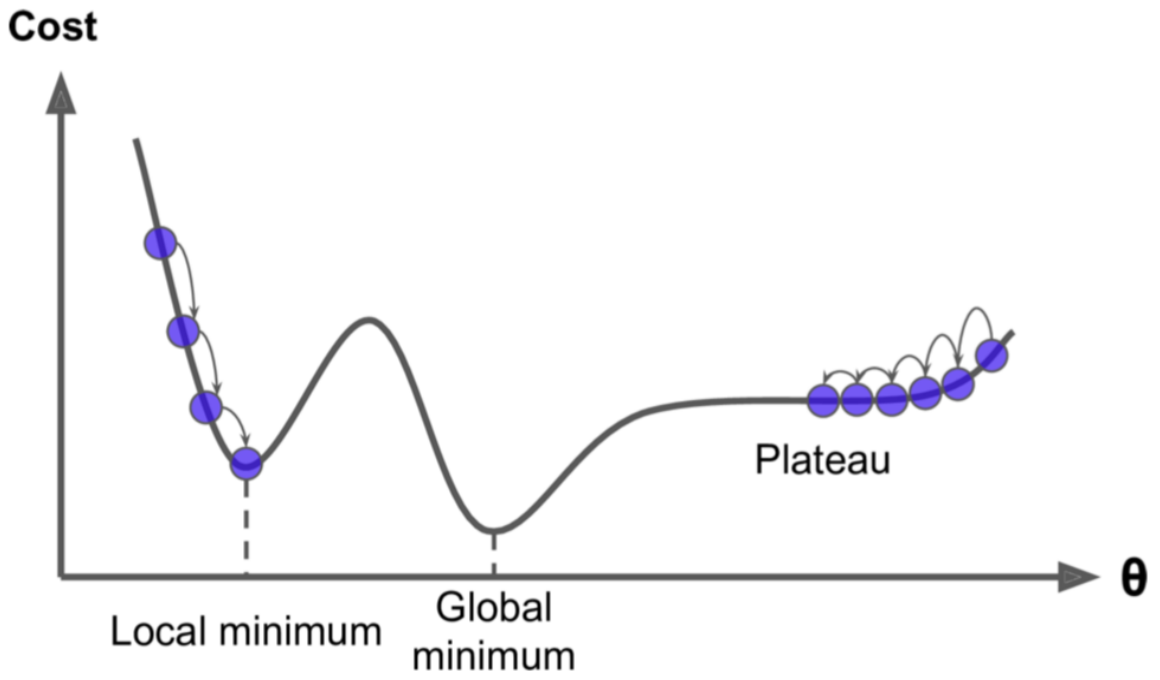
- If the learning rate is too large, results are unstable



Pitfalls of Gradient Descent

Some cost functions may cause problems:

- Local Minimum : It cannot reach the optimal solution by being stuck in local minimum
- Plateau : It takes a long time to cross the plateau

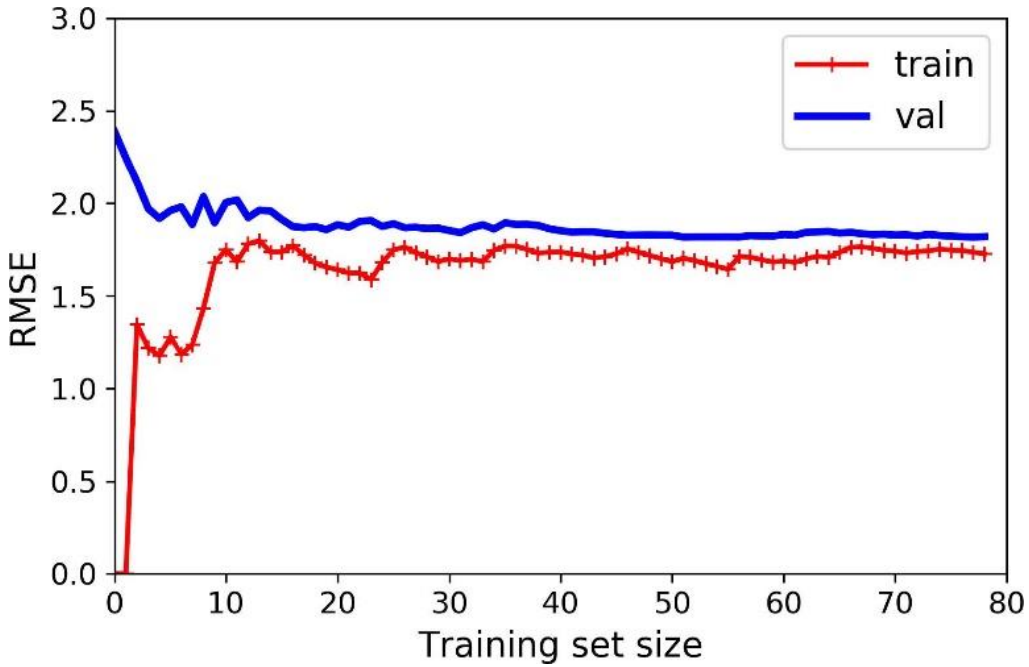


Other Gradient Descent methods

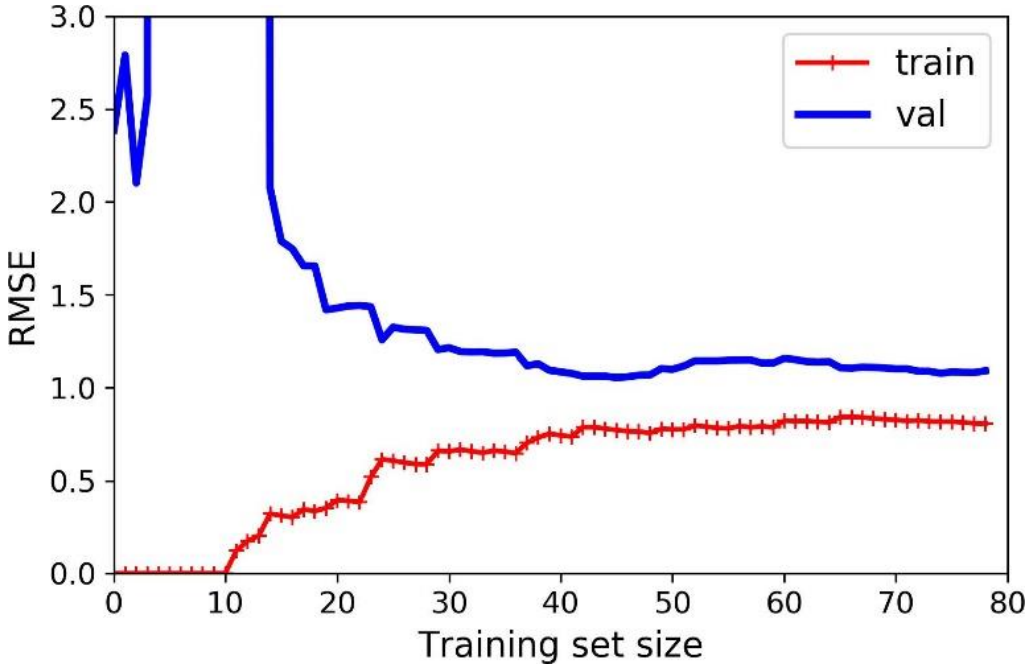
- **Batch Gradient Descent**: Instead of individually computing the partial derivatives, solve them all in one in vectorized form
- **Stochastic Gradient Descent (SGD)**: Pick one random instance from training set at each iteration to compute gradient based on that instance
- **Mini-batch Gradient Descent**: Compute the gradients on small random subsets of instances

Learning curves

- Learning curves are plots of the model performance (e.g. cost function RMSE) on the training and validation sets as the training set size (or training iteration) changes
  - e.g. training two models on the same data:



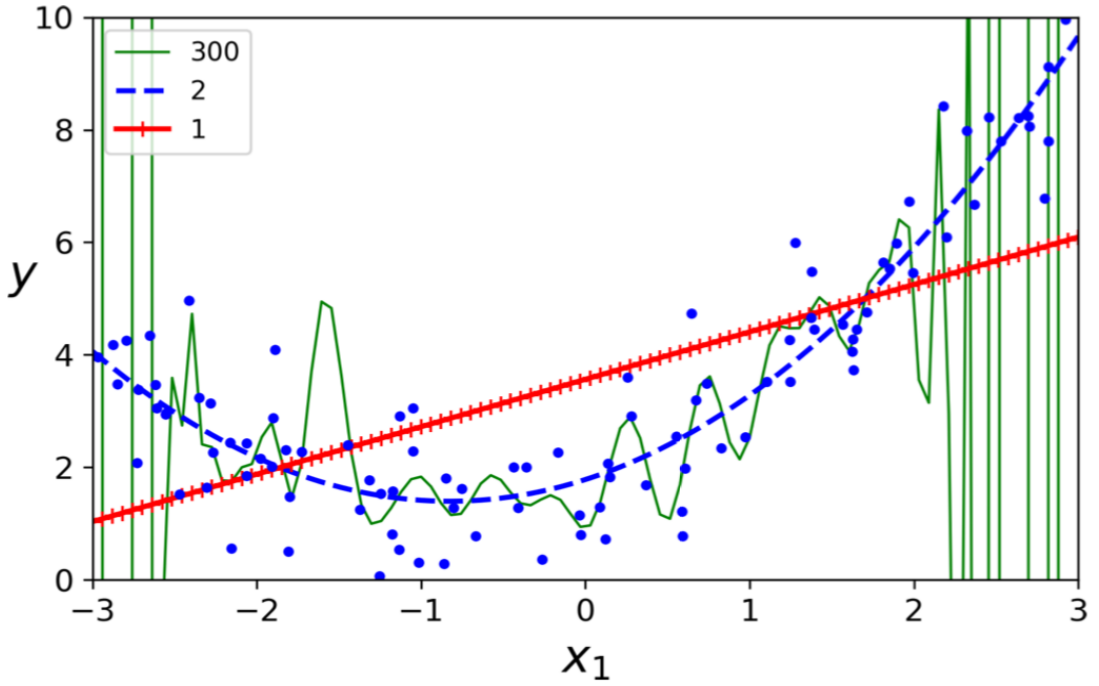
Linear regression: **Underfitting**



Polynomial regression: **Overfitting**

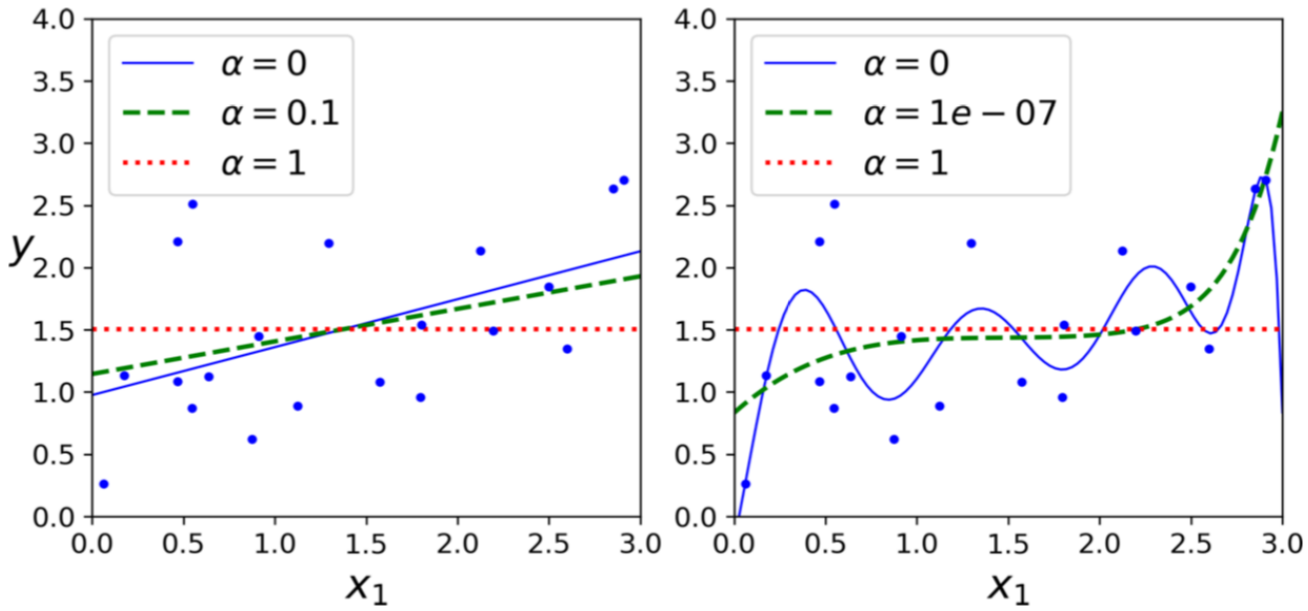
Regularized linear model

- Can we do better for point fitting?
  - Increase the degree of model
- High-degree Polynomial Regression:
  - Low error
  - **Overfitting!**
- **Regularization**
  - Constraining the model to make it simpler and harder for it to overfit the data
  - Example: to regularize a polynomial model by reducing the number of polynomial degrees



Regularized linear model

- Add a **regularized term** in Linear Regression Cost
- **Ridge Regression** : $J(\theta) = MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$
  - Enforce the parameter $\boldsymbol{\theta}$ to be small
- **Lasso Regression** : $J(\theta) = MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} |\theta_i|$
  - Enforce the parameter $\boldsymbol{\theta}$ to be sparse
- α controls the extent to which you want to regularize a model



Lasso Regression with different α values

Classical machine learning methods

- Support vector machine (SVM)
- Decision tree
- Ensemble learning (e.g. Random Forest)
- Dimensionality reduction (e.g. PCA, t-SNE)
- Clustering algorithms (e.g. K-means, DBSCAN)
- Bayesian statistics
- Neural network

Summary

- In this lecture we have learned:
  - What machine learning is
  - Types of machine learning systems, e.g. supervised and unsupervised learning, etc.
  - Methods to measure the performance of classifiers, e.g. cross-validation, confusion matrix, ROC curve, etc.
  - Why use machine learning: when traditional methods fail
  - Some challenges of machine learning
  - How to train a machine learning model, using linear regression as an example

References

- Aurélien Géron. “Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow”, O’ Reilly, 2019 (Ed. 2), Chapters 1, 3 and 4.
- Ian Goodfellow, et al. “Deep Learning”, The MIT Press, 2016, Chapter 5.