CGnal

business innovation through algorithms

Introduction to Machine Learning

CGnal S.r.l. – Corso Venezia 43 - Milano

Novembre 2022| Milano

WHO ARE WE?

Giovanni Prestipino, MSc
Senior Data Scientist, CGnal Srl

Giovanni has a background in Mathematics and he obtained the M.Sc. degree at Università degli studi di Milano. During the master he moved to Stuttgart to write his thesis as a part of the erasmus program. Giovanni has been working as Data Scientist for over 6 years across several industries and businesses. From 2020 he is working in Cgnal where he has been developing machine learning solutions.

Gioa Boschi, PhD
Data Scientist, CGnal Srl

Gioia is a data scientist with a background in physics and mathematics. She obtained a PhD in applied math from King's College London in 2021, focused on theoretical modelling and simulations of complex systems. As a researcher she presented her work to several international conferences and brought her research to the UK Parliament as a finalist of STEM for Britain. After the PhD she switched from academia to industry to work in data science where she applied machine learning to solve different business problems including calibration of highly realistic simulations, forecasting and user-behaviour modelling.

Siddhant Tandon, MSc
Data Scientist, CGnal Srl

Siddhant has a background in Electronics and Communication Engineering. After that, he obtained an M.Sc. degree in Data Science at Sapienza Università di Roma. He obtained his master thesis at Nokia Bell Labs in Paris here he developed deep reinforcement learning models with applications in the field of cloud computing. Since 2019 he is working in Cgnal where he has been developing machine learning solutions in the marketing and finance sector.

DAY 1

Introduction

- Brief overview of Machine Learning (Supervised, Unsupervised)
- Introduction to Graph, Graph Theory and main metrics for characterizing graphs

DAY 2

Graph Machine Learning

- Community detection on graphs
- Supervised Machine Learning on Graphs

DAY 3

Explainability & Interpretability

- Introduction to explainability problem
- LIME & SHAP

DAY 2

Simple Neural Networks

- Introduction to TensorFlow and Computational Graph
- Implementation and training of simple Neural Networks

DAY 3

Advanced Neural Networks

- Convolutional Neural Networks and Recurrent Neural Networks
- Advanced Topics

DAY 1

Introduction

- Brief overview of Machine Learning (Supervised, Unsupervised)
- Introduction to Graph, Graph Theory and main metrics for characterizing graphs

DAY 2

Graph Machine Learning

- Community detection on graphs
- Supervised Machine Learning on Graphs

DAY 3

Explainability & Interpretability

- Introduction to explainability problem
- LIME & SHAP

DAY 2

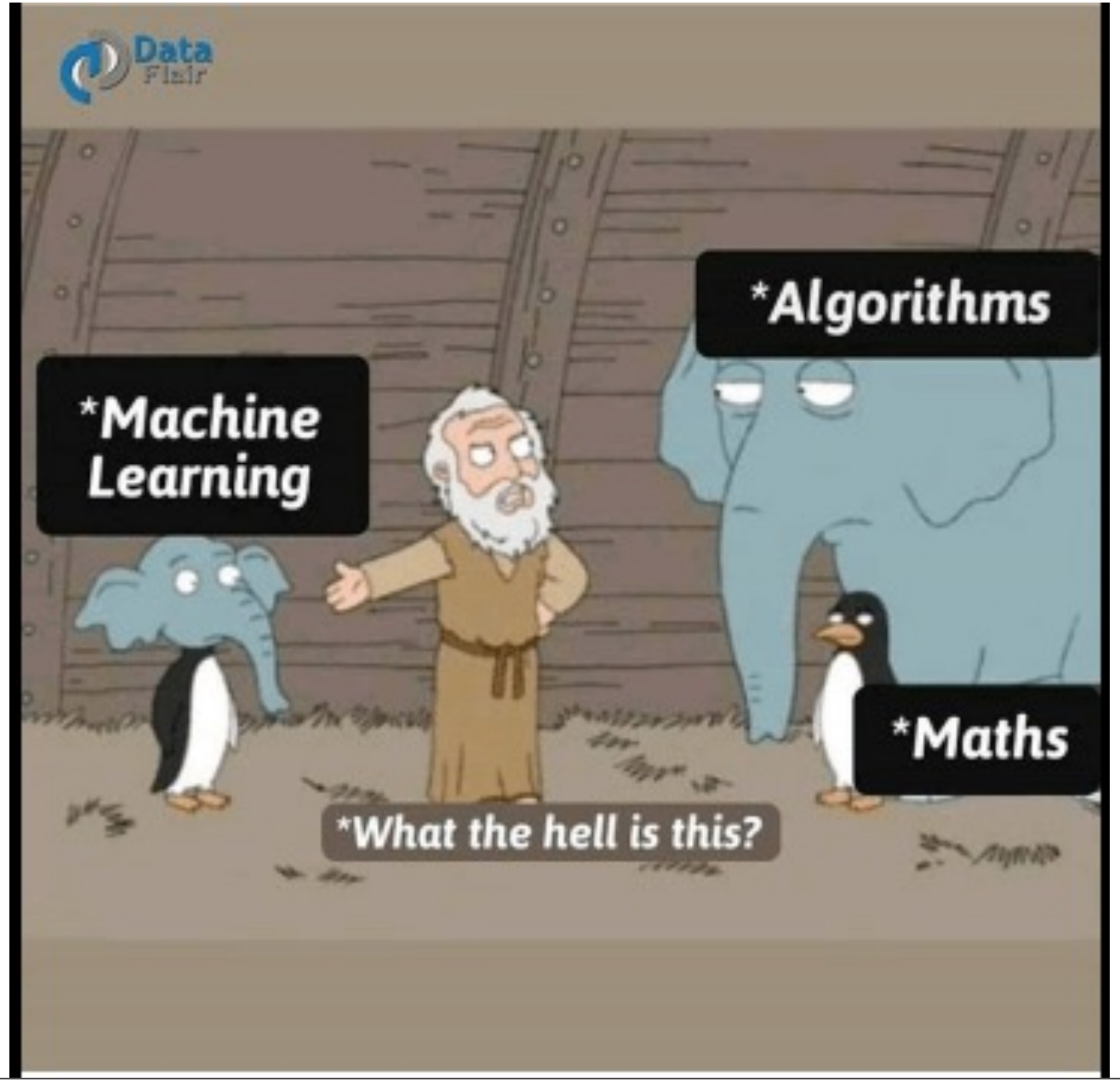
Simple Neural Networks

- Introduction to TensorFlow and Computational Graph
- Implementation and training of simple Neural Networks

DAY 3

Advanced Neural Networks

- Convolutional Neural Networks and Recurrent Neural Networks
- Advanced Topics


Data Flair
*Machine Learning
*Algorithms
*Maths
*What the hell is this?

What is Artificial Intelligence?

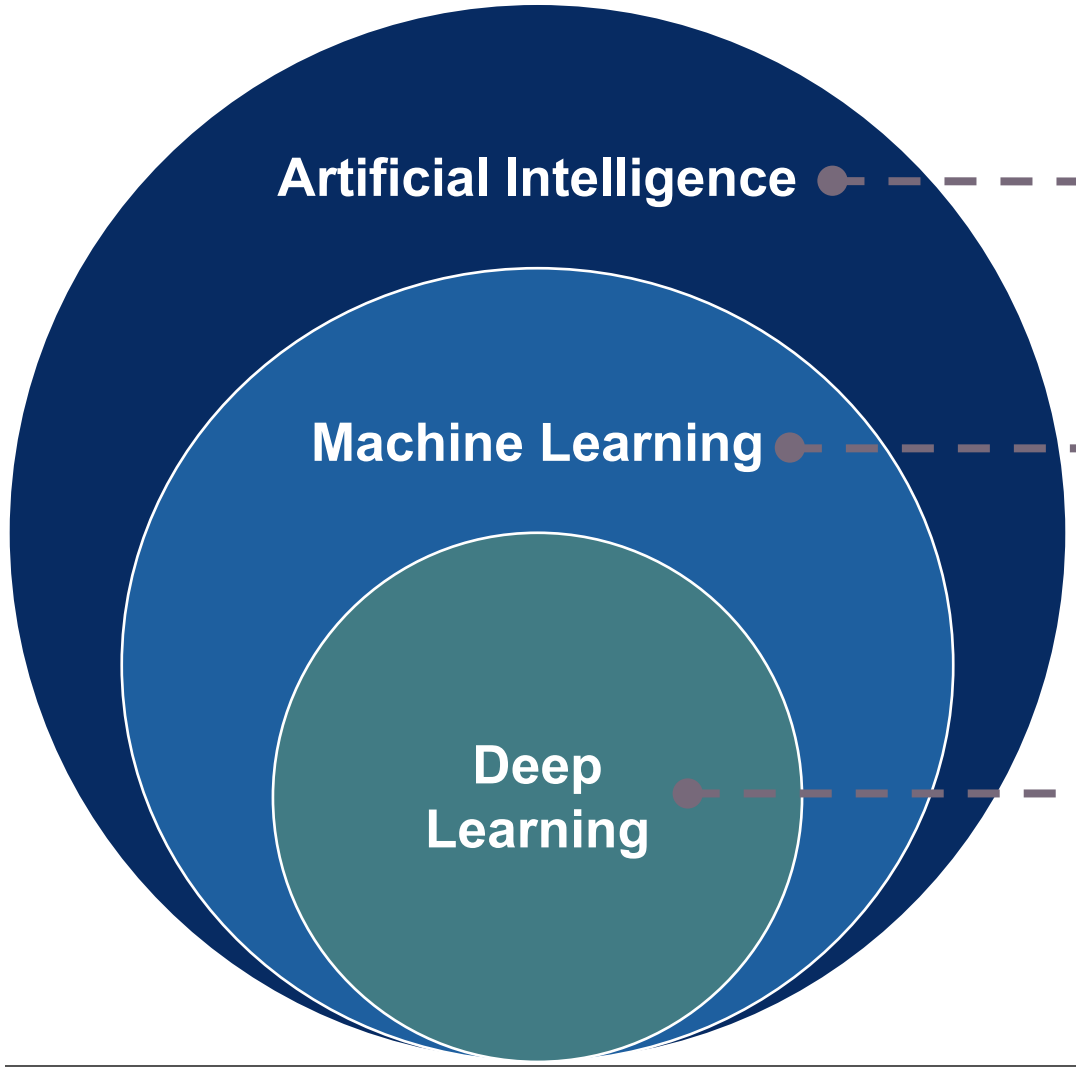
Artificial intelligence is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

Encyclopedia Britannica

Artificial intelligence (AI) is intelligence demonstrated by machines. Leading AI textbooks define the field as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.

Wikipedia

Types of Artificial Intelligence



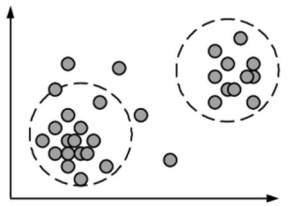
Artificial Intelligence

Softwares that are mimicking human reasoning and can perform “intelligent” tasks

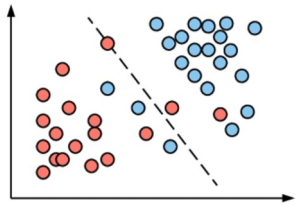
Machine Learning

Algorithms that are NOT explicitely programmed but are able to “learn” a specific task from observing appropriate data samples

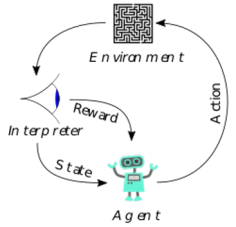
Usupervised



Supervised

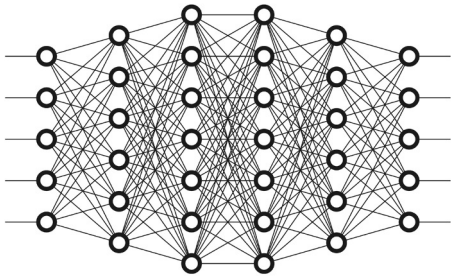


Reinforcement



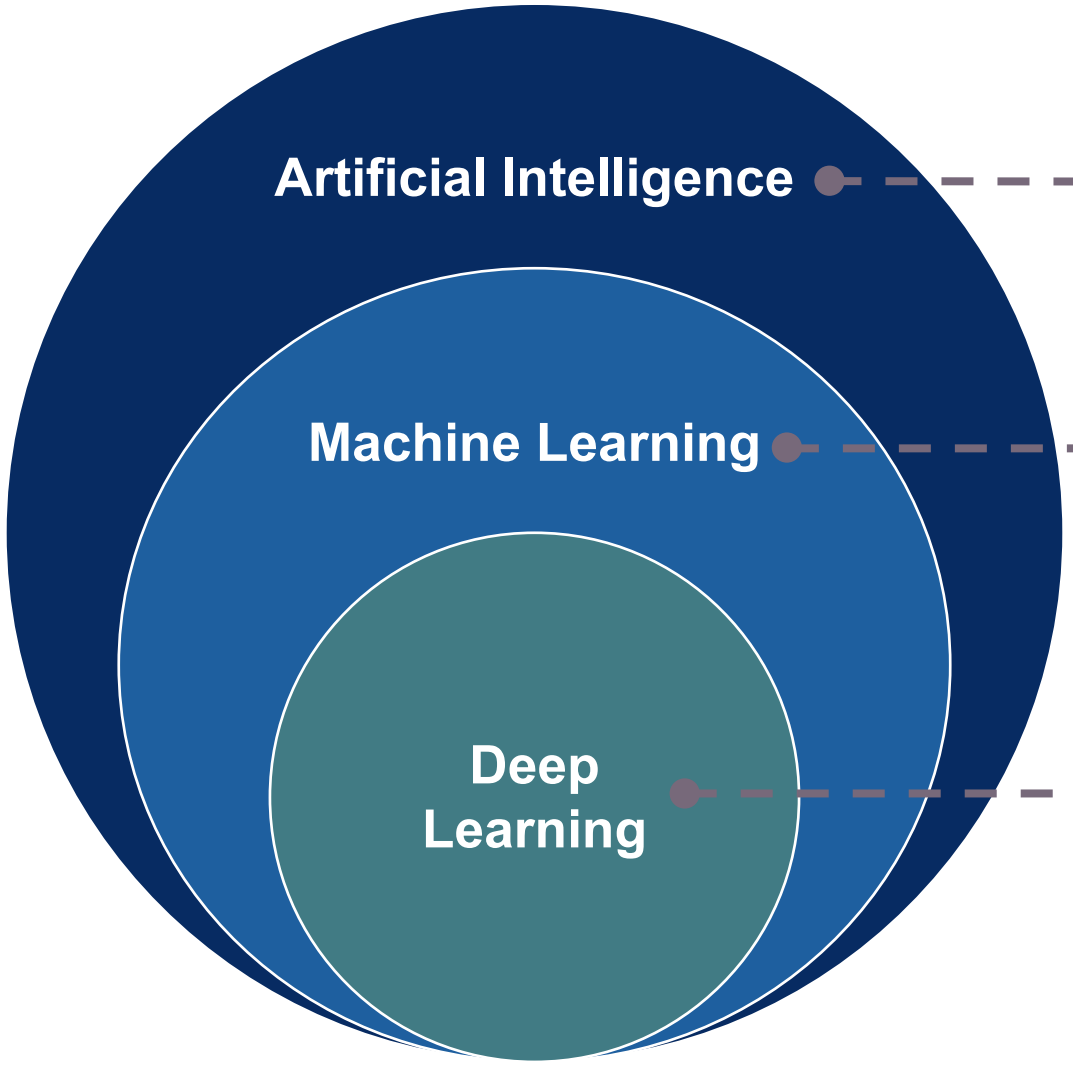
Deep Learning

ML Algorithms that are based on multilayered Artificial Neural Networks



CGnal

Types of Artificial Intelligence



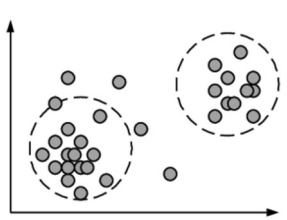
Artificial Intelligence

Softwares that are mimicking human reasoning and can perform “intelligent” tasks

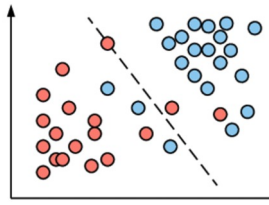
Machine Learning

Algorithms that are NOT explicitely programmed but are able to “learn” a specific task from observing appropriate data samples

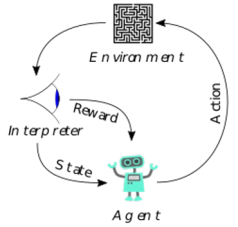
Usupervised



Supervised

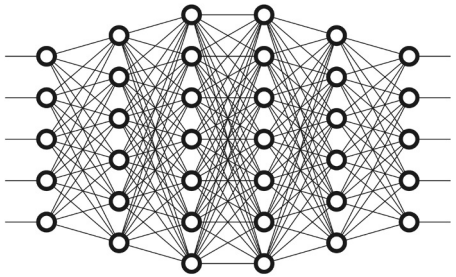


Reinforcement



Deep Learning

ML Algorithms that are based on multilayered Artificial Neural Networks

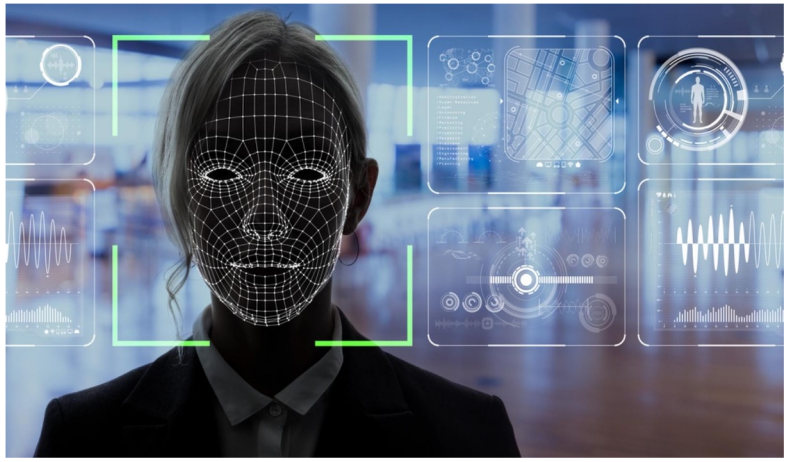


CGnal

Some examples

Speech Recognition

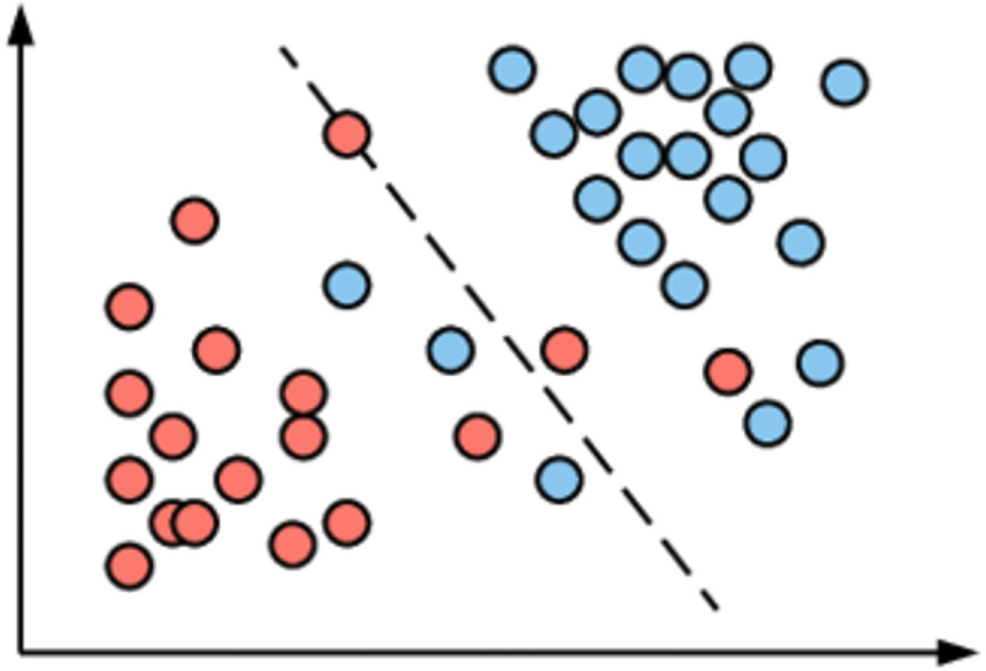
Image Recognition



Sentiment Analysis

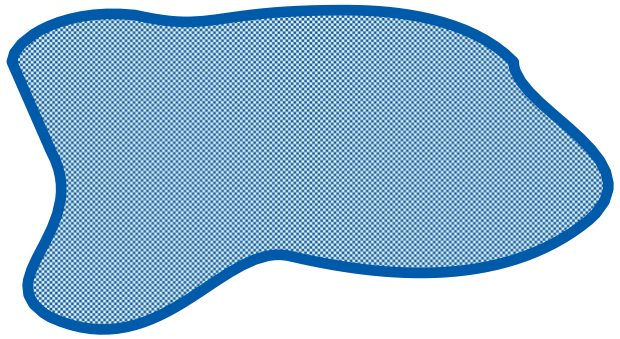
Recommendation Systems

Supervised



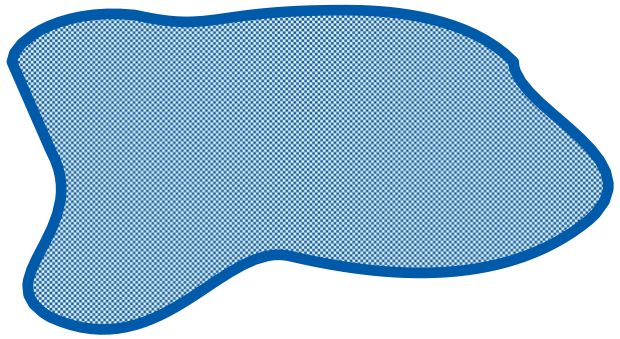
Problem setup

X – a set of objects

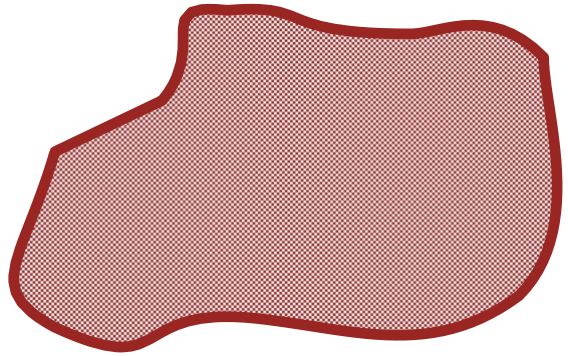


Problem setup

X – a set of objects



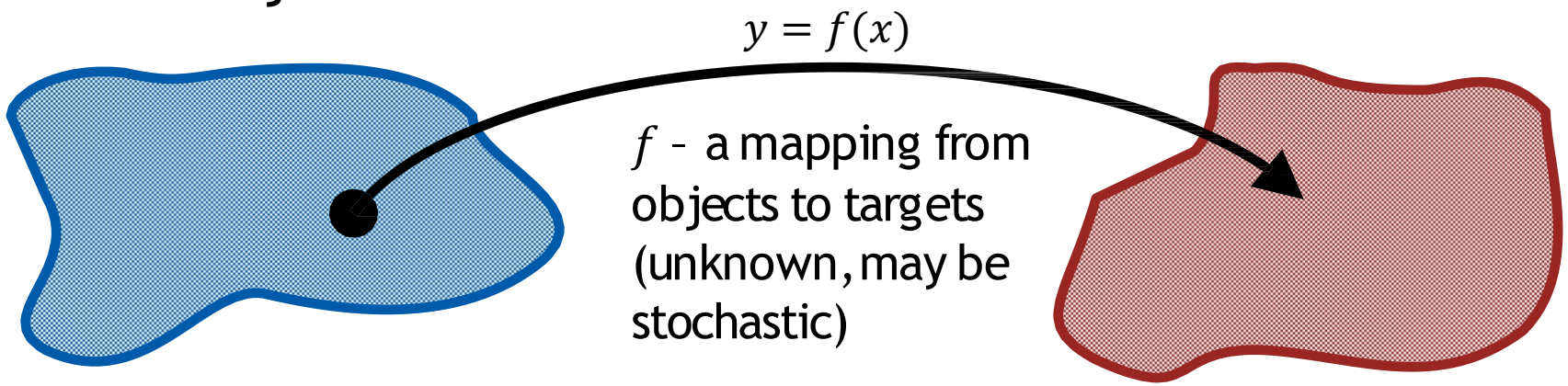
Y – a set of targets



Problem setup

X – a set of objects

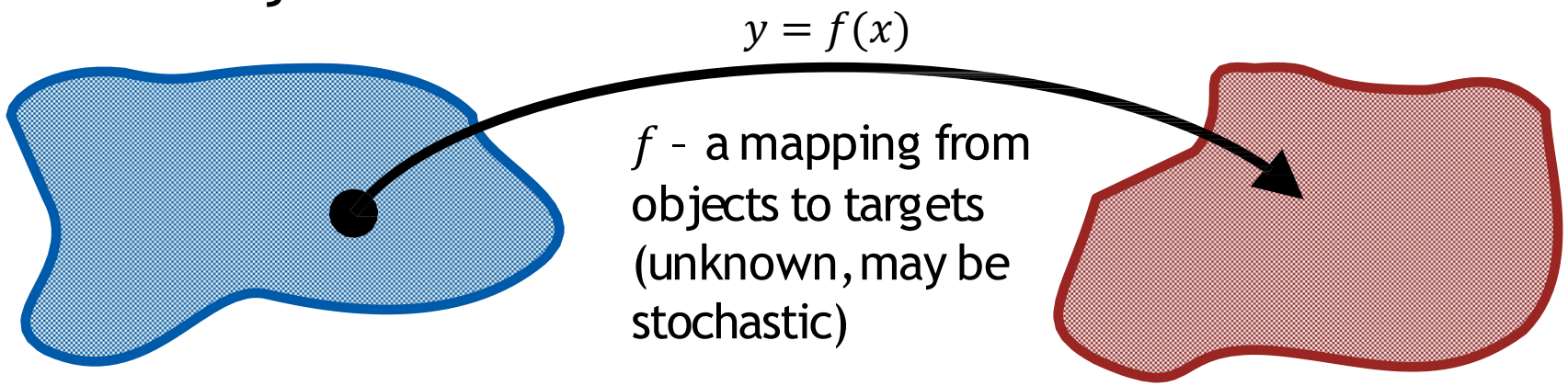
Y – a set of targets



Problem setup

X – a set of objects

Y – a set of targets



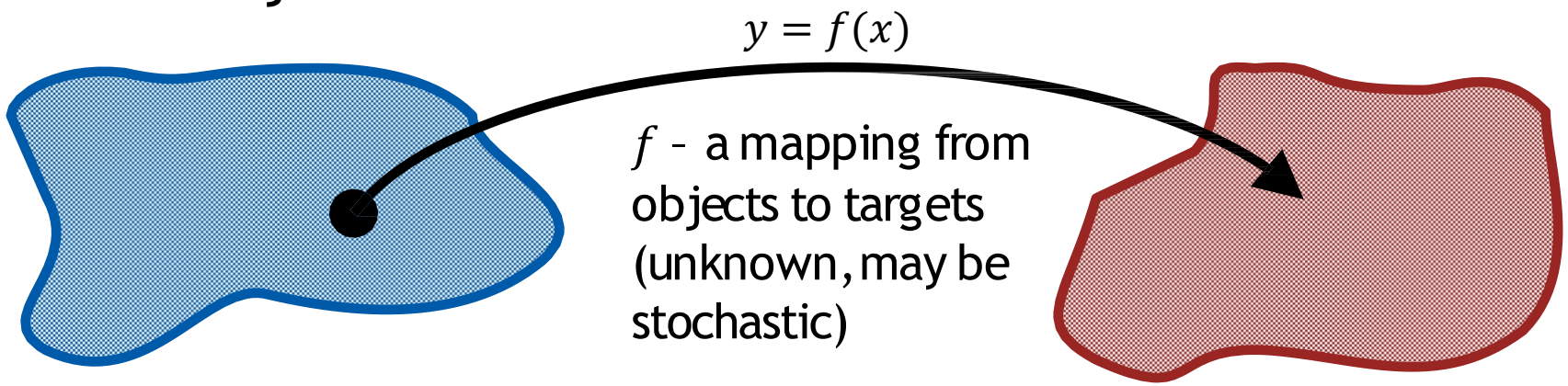
A dataset: $D = \{(x_i,\ y_i) : i = 1,\ 2, \dots,\ N\}$

$x_i \in Y, \qquad y_i = f(x_i) \in Y$

Problem setup

X – a set of objects

Y – a set of targets



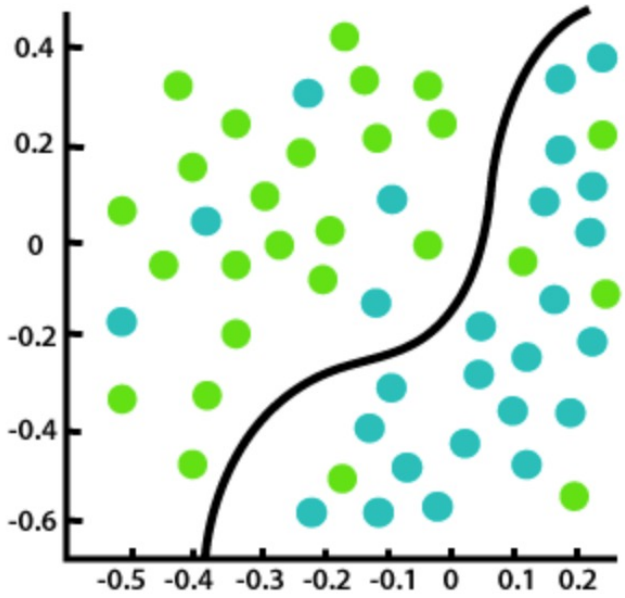
A dataset: $D = \{(x_i,\ y_i) : i = 1,\ 2, \dots,\ N\}$

$x_i \in Y, \qquad y_i = f(x_i) \in Y$

Goal: **approximate f given D**
i.e. learn to **recover targets from objects**

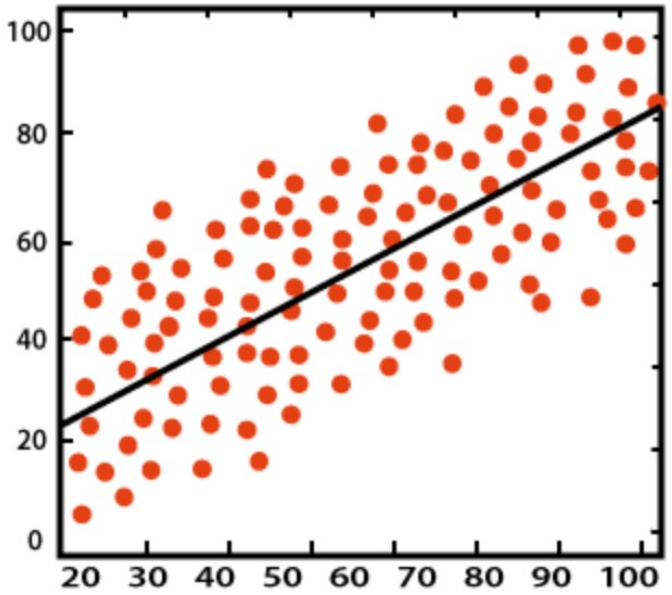
Classification or Regression?

Classification



Classification is a process of finding a function which helps in dividing the dataset into **unordered classes**. The model, trained on a labelled dataset, aims at categorizing the unobserved sample into the different classes, usually providing **confidence/probability** for each class.

Regression



Regression algorithms aim to find the function that maps the input variables (features) to a continuous output target. During training, the function that **best fits** the observed samples is sought and such a function is then used to produce the prediction. Sometimes confidence interval around the prediction value can be produced.

Examples - Classification

- Iris flower species classification

Objects

Individual flowers, described by the length and width of their sepals and petals

Targets

Species to which this particular flower belongs

Mapping

Different shapes of sepals and petals correspond to different species

(non-deterministic)



images source: wikipedia.org

Examples - Classification

- Spam filtering

Objects

E-mails (sequences of characters)

Targets

“spam” / ”not spam”

Mapping

Message content defines whether it’s spam or not

(non-deterministic, varies from person to person)

Examples - Regression

- Boston Housing Dataset

Objects

Properties of house, building, neighborhood, location

Targets

Selling price or renting price

Mapping

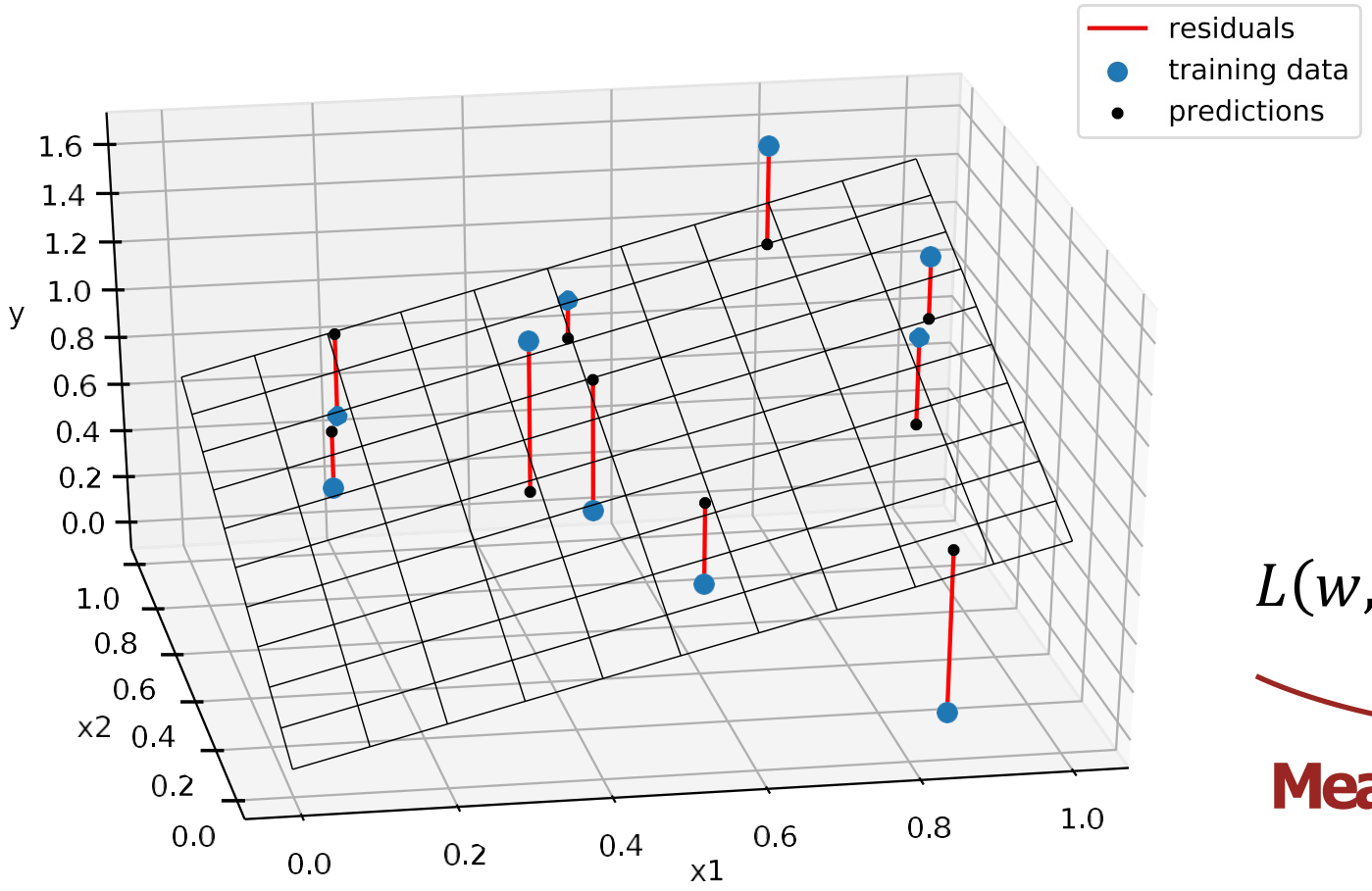
The features of the property define its value

(non deterministic, depending on human needs and willingness to buy)

Algorithms and Metrics – Classification and Regression

	Algorithms	Quality Metrics
Classification	• Logistic Regression • K-Nearest Neighbours • Support Vector Machines / Kernel SVM • Naïve Bayes • Decision Tree / Random Forest Classification	• Accuracy • Confusion Matrix and related metrics (Precision, Recall, Sensitivity, Specificity, F1-score) • ROC, AUC e AUC-PR • Log-likelihood Loss / Cross-entropy Loss
Regression	• Polynomial Regression • Support Vector Regression • Decision Tree / Random Forest Regression • Regressions • Prophet / BSTM / ARIMA	• Root Mean Square Error • Mean Absolute Error • Mean Absolute Percentage Error • Explained Variance - R2

Example: Linear regression



$$\hat{f}_{w,b}(x) = w^T \cdot x + b$$

$$w \in \mathbb{R}^d$$

$$b \in \mathbb{R}$$

$$x \in Y \subset \mathbb{R}^d$$

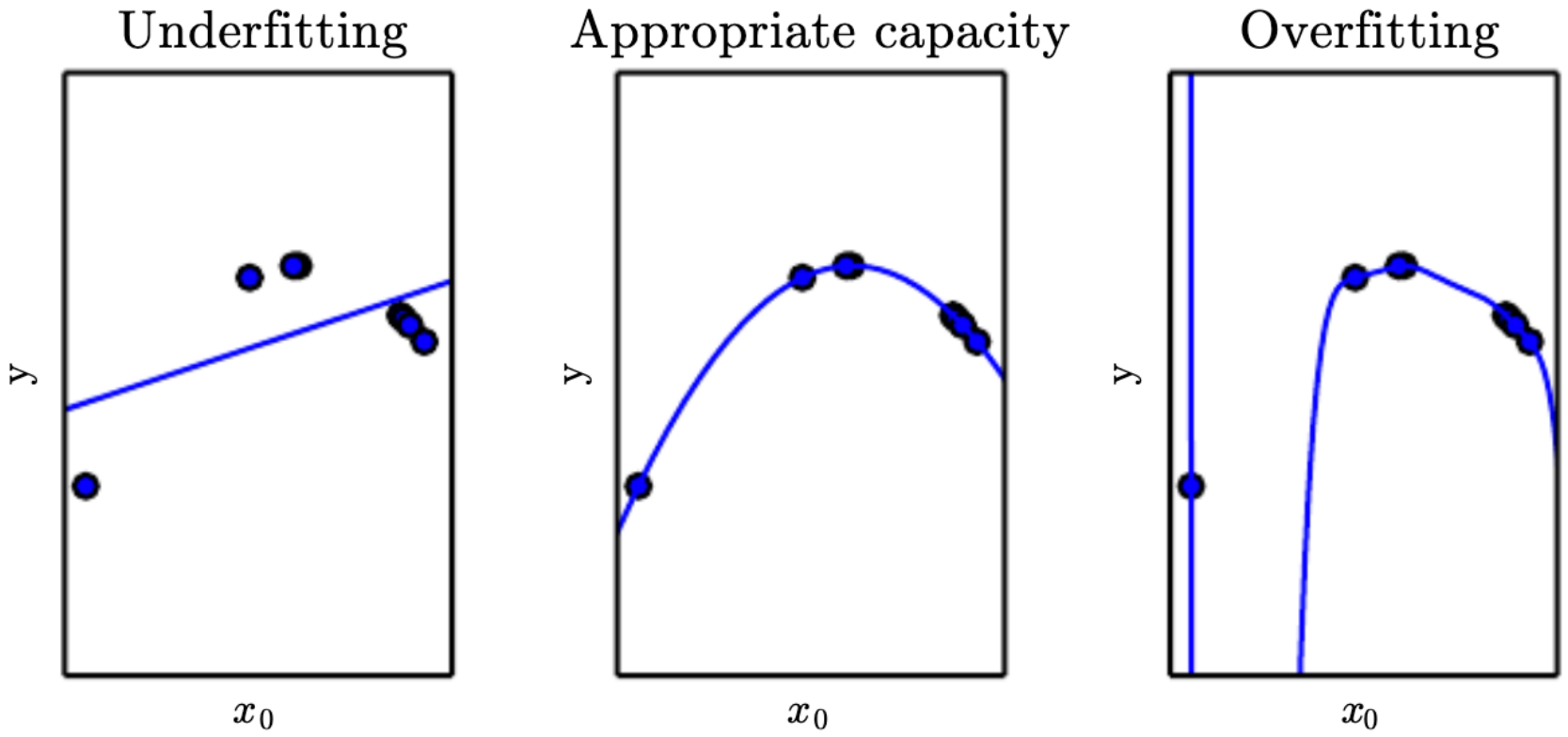
$$L(w,b) = \frac{1}{N}\sum_i \left(y_i - \hat{f}_{w,b}(x_i)\right)^2 \xrightarrow[w,b]{} \min$$

Mean Squared Error (Loss function)

Underfitting & Overfitting

Example: Polynomial regression

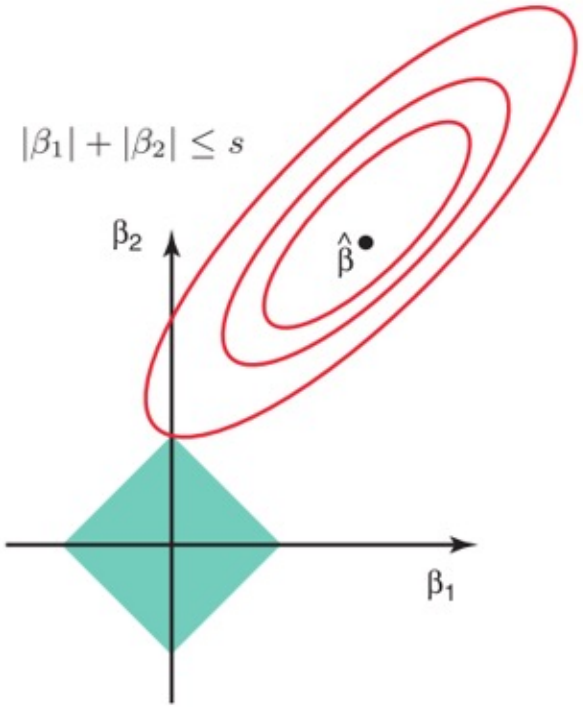
$$\hat{f}_{w,b}(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n = w^T \cdot \boldsymbol{x} + b$$



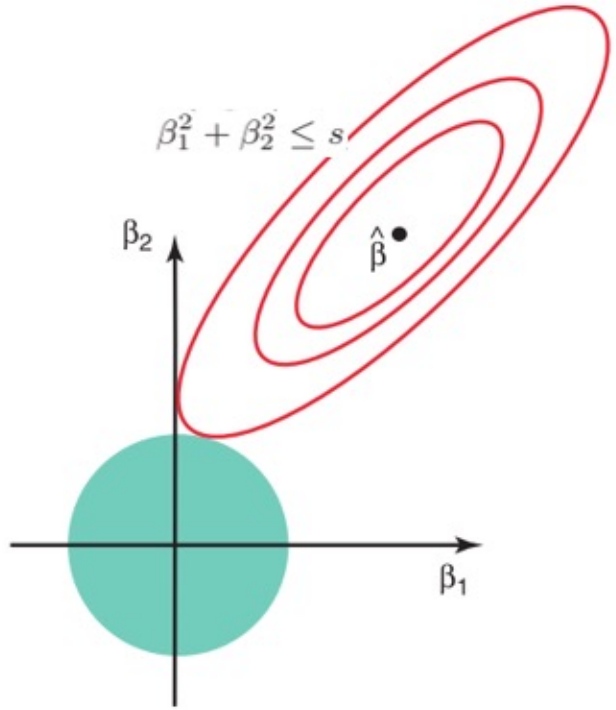
Example: Polynomial regression

$$L(w, b, \alpha) = \frac{1}{N}\sum_i \left(y_i - \hat{f}_{w,b}(x_i)\right)^2 + \alpha\|w\|$$

$\alpha\|w\|^2$



Lasso Regression



Ridge Regression

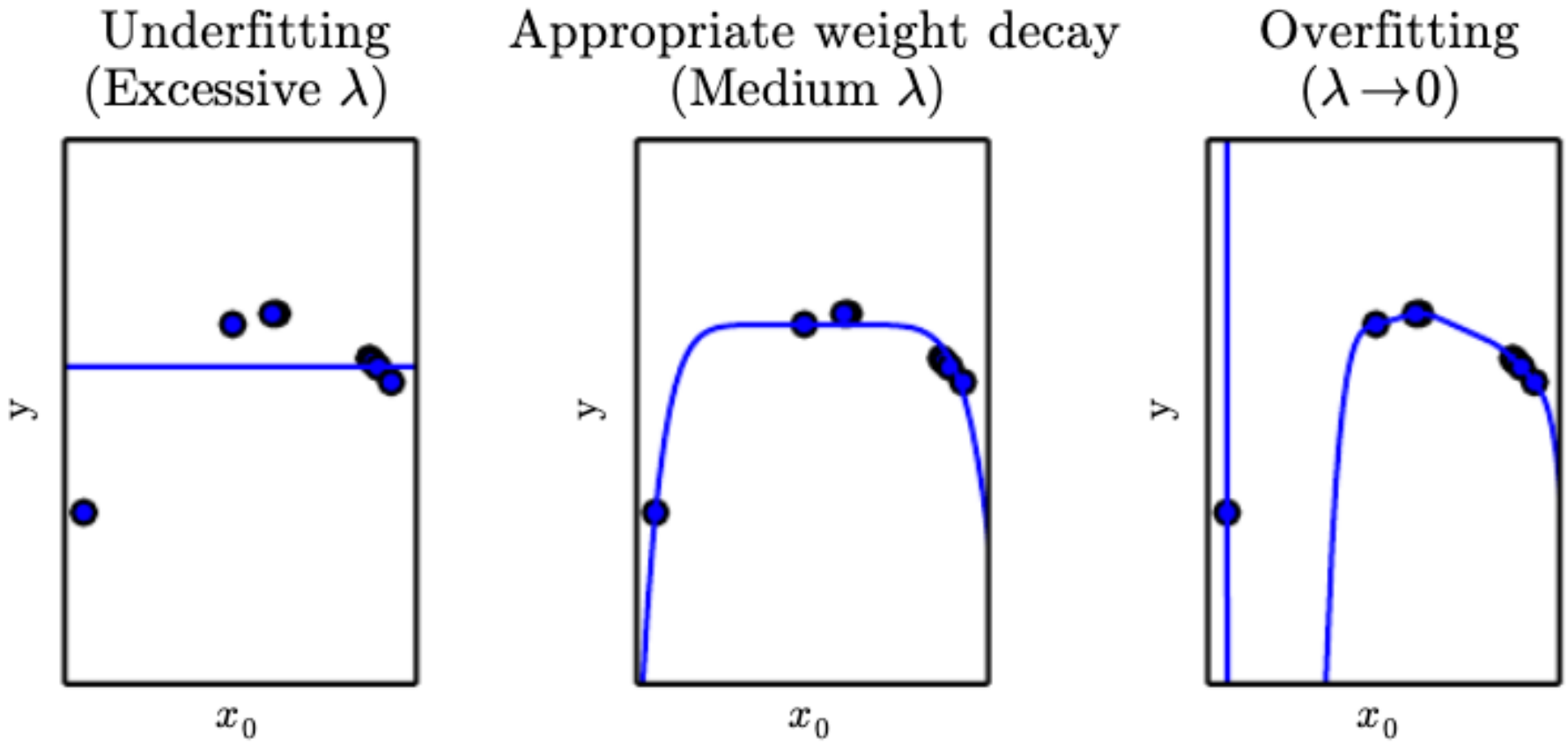
If β ----> $\hat{\beta}$

If β ----> $\hat{\beta}$

At fixed n

Example: Polynomial regression

$$L(w, b, \alpha) = \frac{1}{N}\sum_i \left(y_i - \hat{f}_{w,b}(x_i)\right)^2 + \alpha\|w\|$$

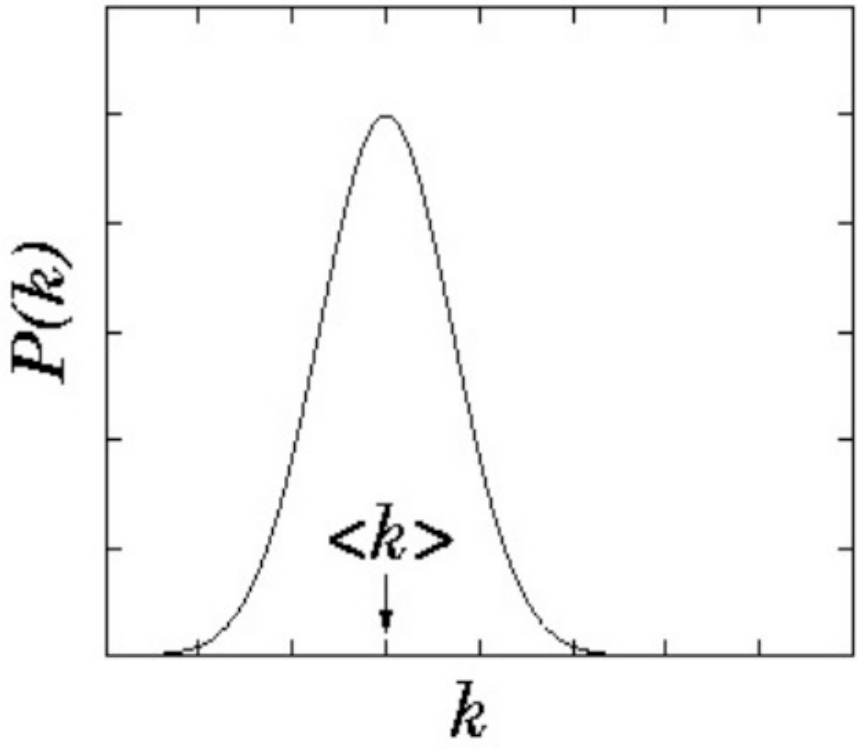


At fixed n

Bias and Variance Error

$$\hat{\theta}_m = f(\{(x_i, y_i) \mid i \in D_m\})$$

Estimator for a generic model parameter or quantity, e.g. a parameter of a distribution



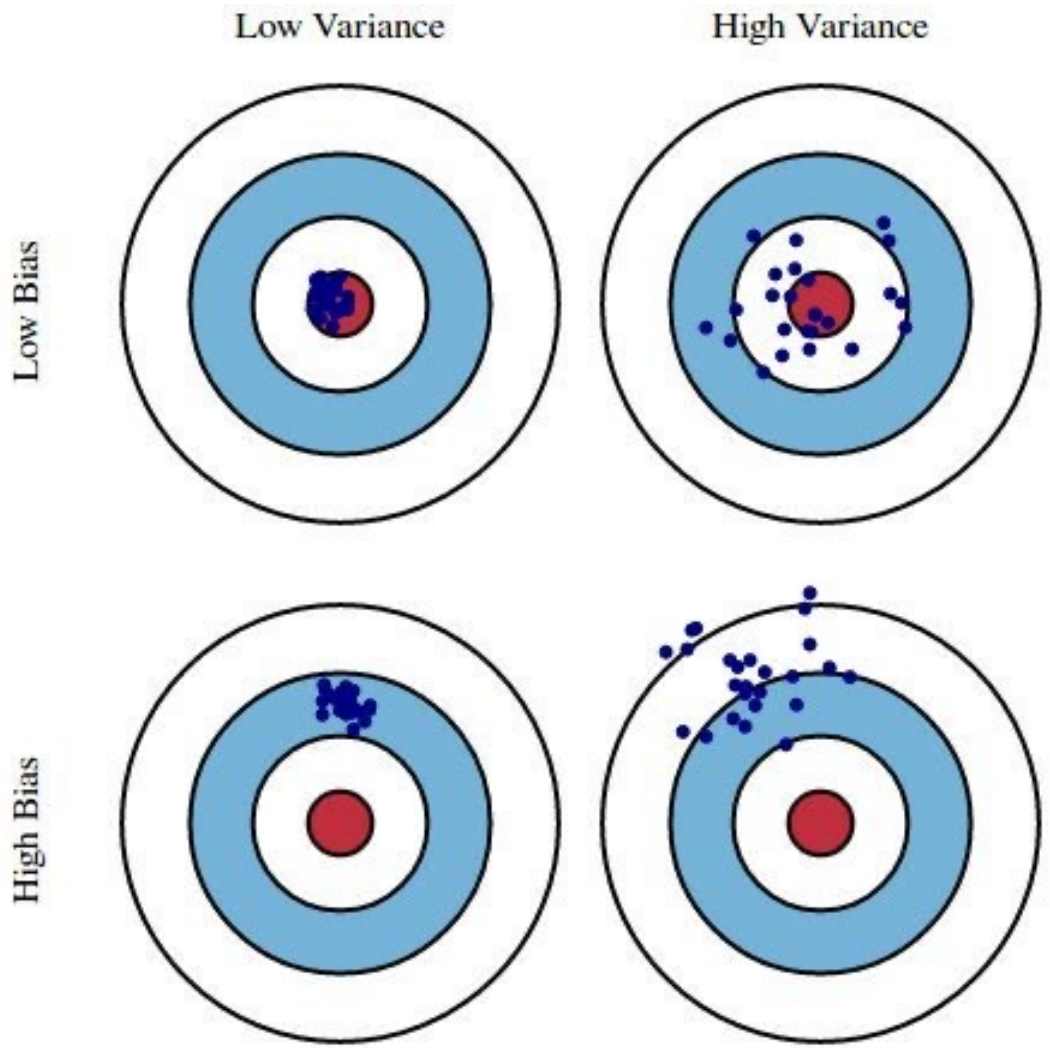
μ

Bias $$Bias(\hat{\theta}_m) = E[\hat{\theta}_m] - \theta$$

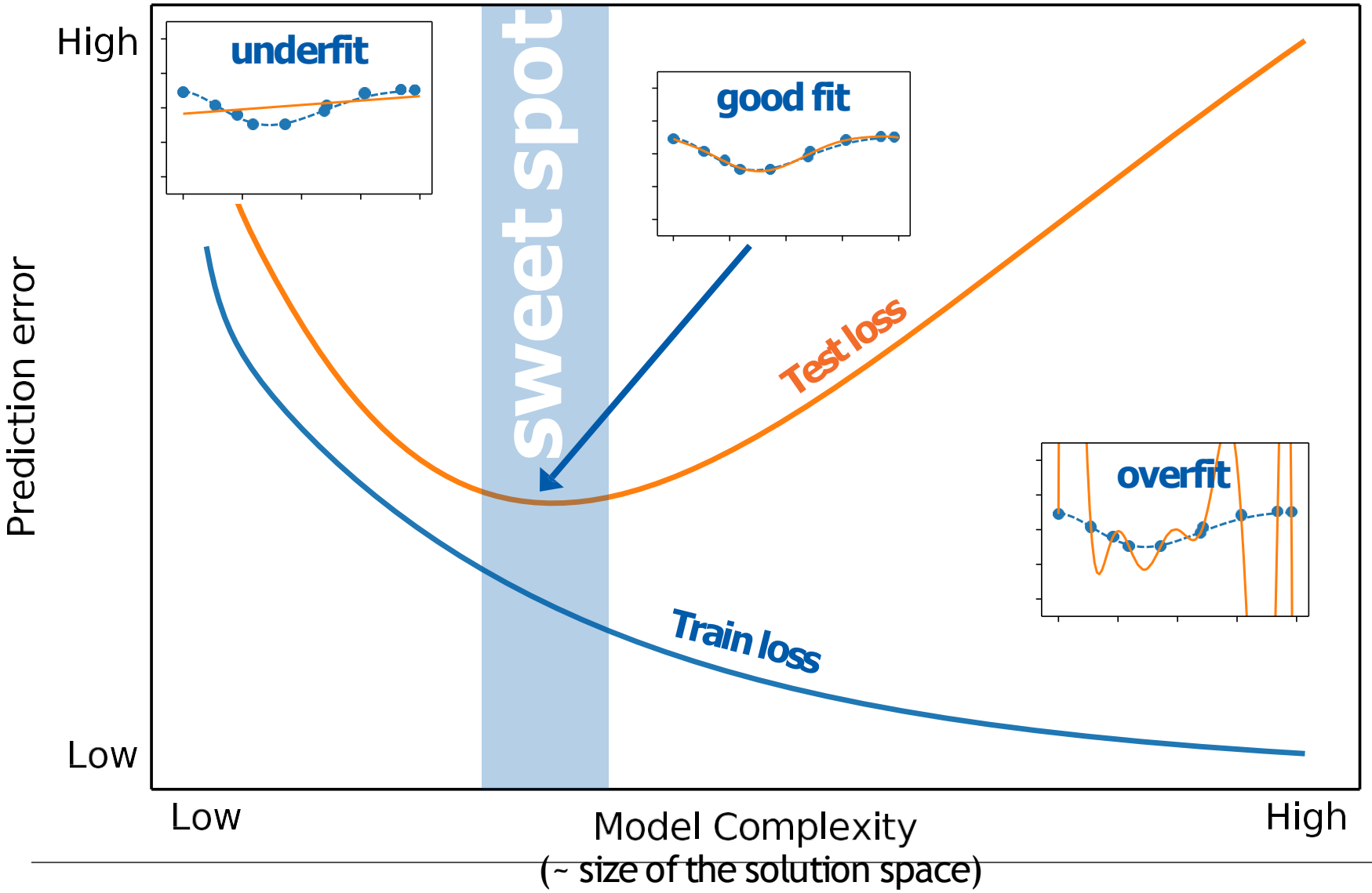
Variance $$Var(\hat{\theta}_m) = E\left[(\hat{\theta}_m - \mu)^2\right]$$

$$E\left[(\hat{\theta}_m - \theta)^2\right] = Bias(\hat{\theta}_m)^2 + Var(\hat{\theta}_m)$$

Bias and Variance Error



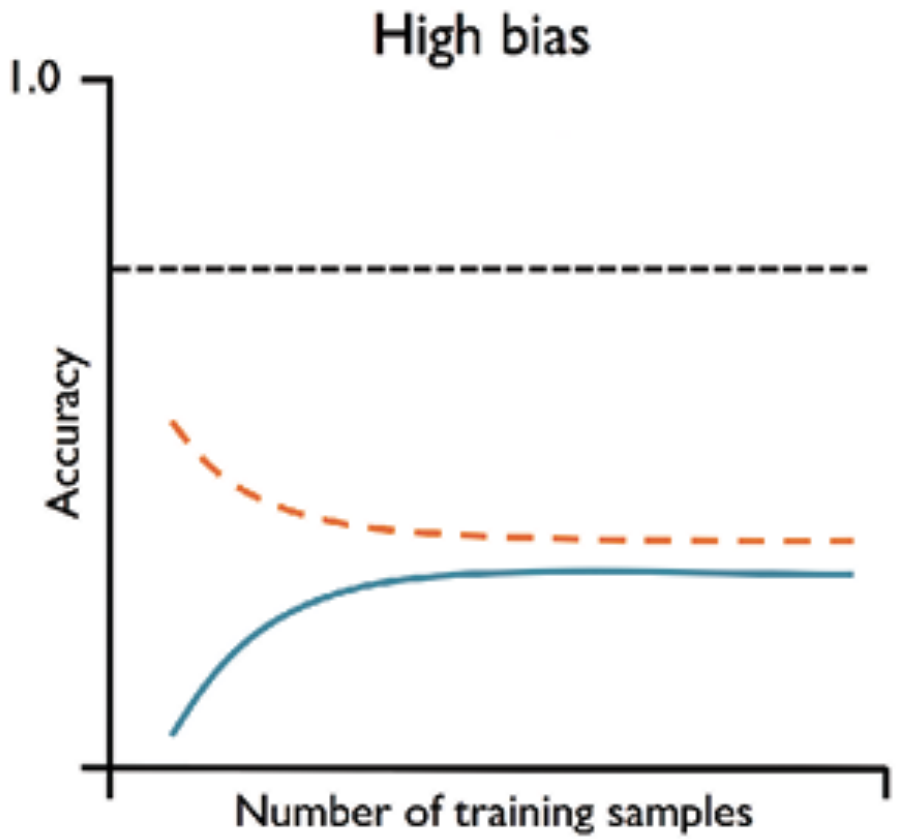
How to check whether a model is good?

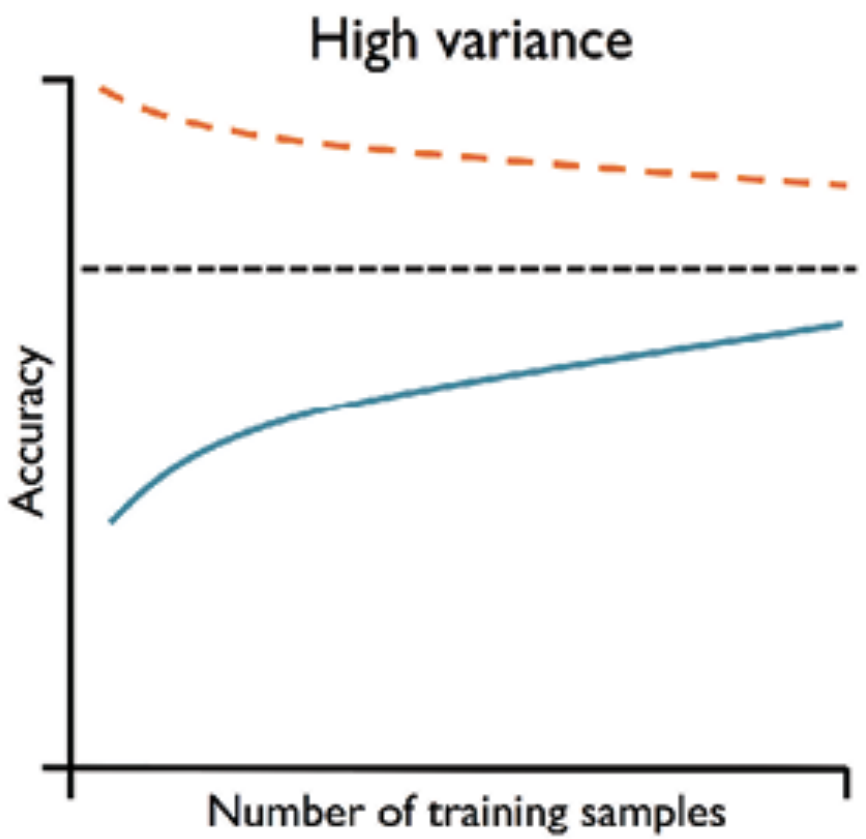


Check the loss on the **test data** - i.e. data that the learning algorithm "hasn't seen"

The goal is to find the **right level of limitations** - not too strict, not too loose

Learning curves



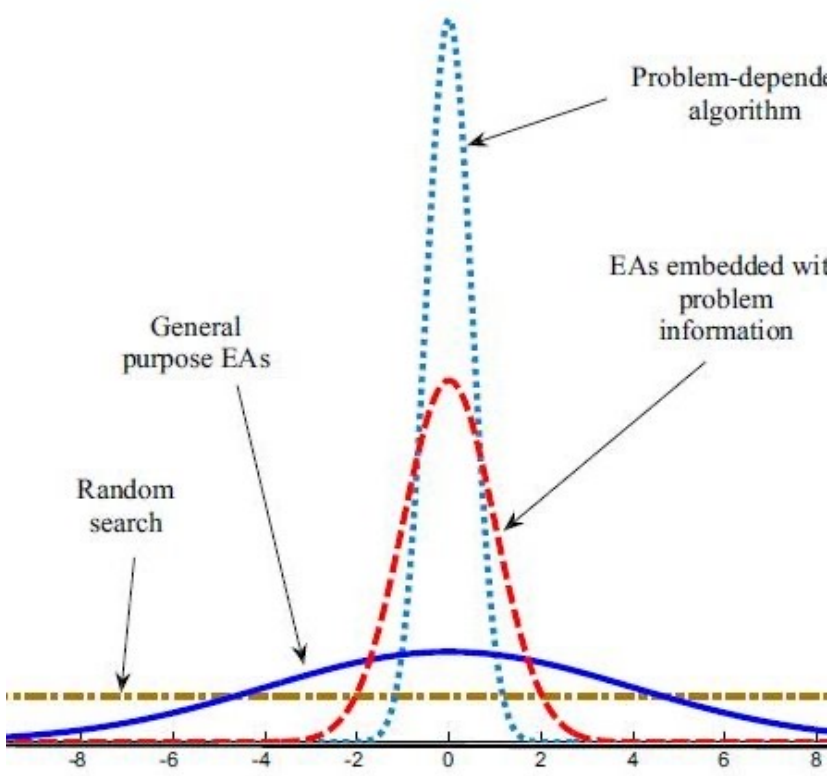


No Free Lunch Theorem

Theorem. For any pair of optimization algorithms, a_1 and a_2

$$\sum_f P\left(d_m^y \middle| f, m, a_1\right) = \sum_f P\left(d_m^y \middle| f, m, a_2\right)$$

- f a arbitrary function to be minimized or approximated
- m iterations
- d_m^y is the cost associated to iteration m



This theorem explicitly demonstrates that what an algorithm gains in performance on one class of problems it necessarily pays for on the remaining problems that is the only way that all algorithms can have the same f-averaged performances

Hyper-Parameter Tuning and Cross-Validation

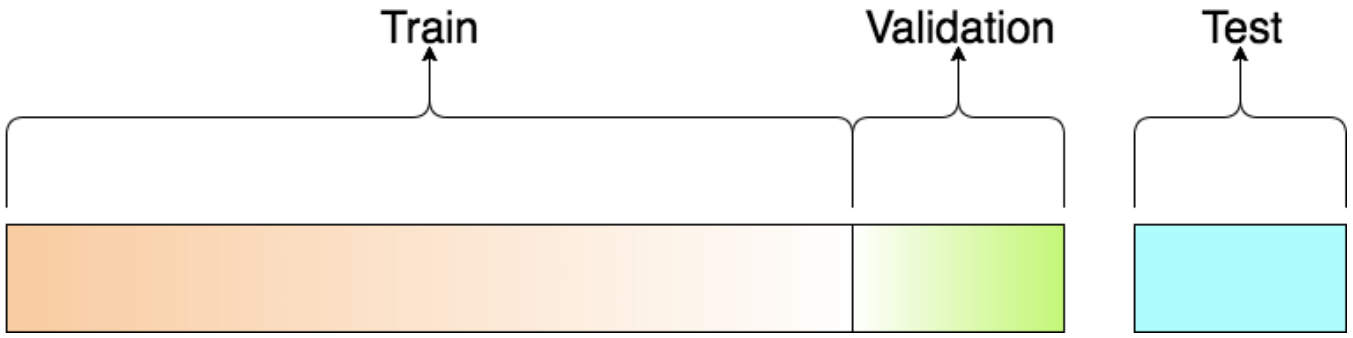
$$L(w, b, n, \alpha) = \frac{1}{N}\sum_{i}\left(y_i - \hat{f}^{n}_{w,b}(x_i)\right)^2 + \alpha\|w\|$$

Parameters

Internal model parameters to be **fitted** using training data. They are generally what optimization algorithms tune in order to **minimize** the loss function

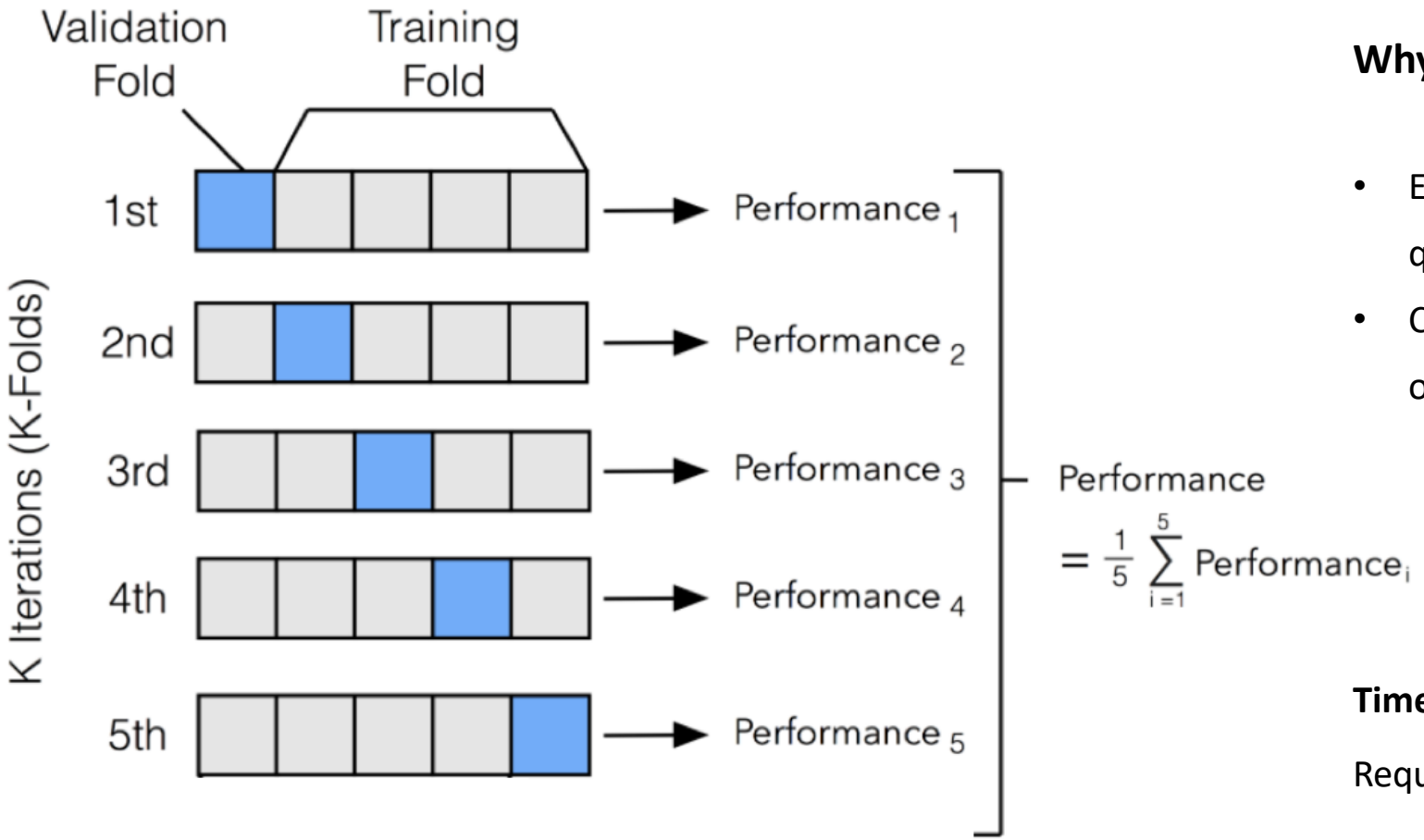
Hyper-parameters

Parameters that control the algorithmic behaviour and usually controls **model capacity.** These parameters are **not** learned on the training dataset but needs to be chosen by evaluating the generalization error of a model



- **Train:** use to fit and tune a model
- **Validation:** use to measure quality of the model during its tuning (trainig set for the hyperparameters)
- **Test:** use for final quality measurement (after tuning)

Hyper-Parameter Tuning and Cross-Validation



Why?

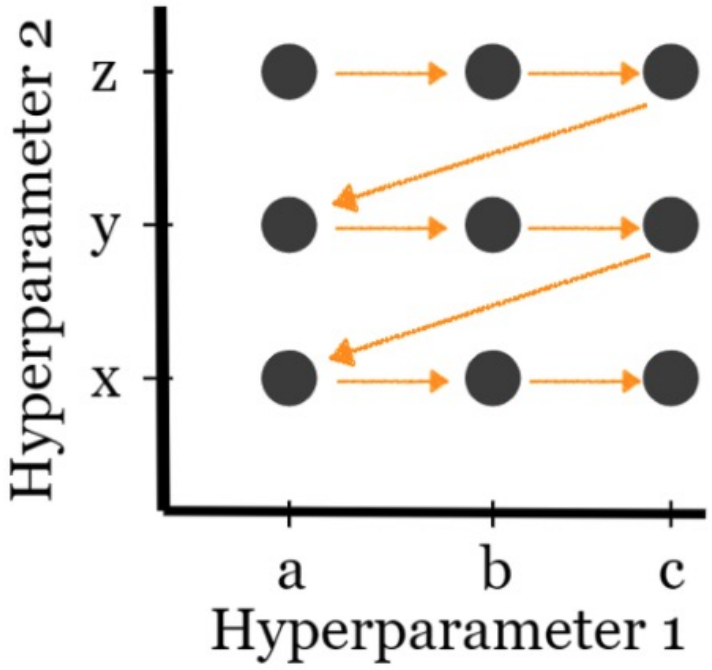
- Estimate **mean** and **standard deviation** of the quality metrics
- Collect out-of-bag predictions to compute overall quality metrics

Time consuming

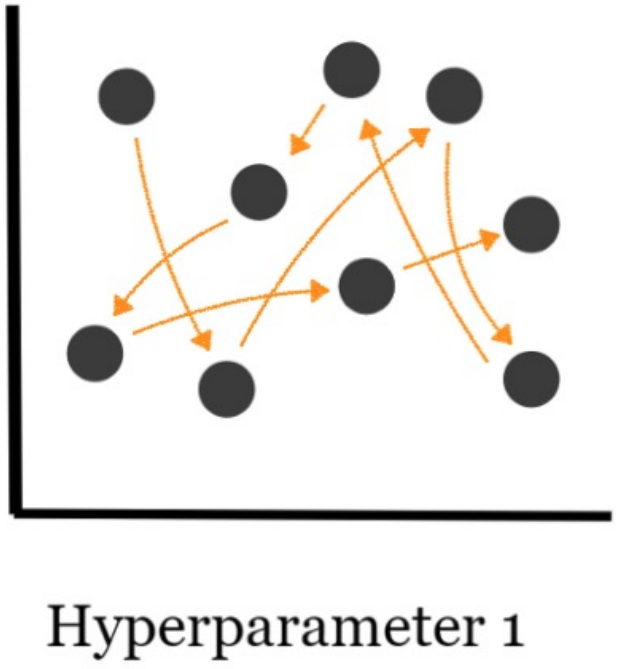
Requires to fit the model K times

Hyper-Parameter Tuning and Cross-Validation

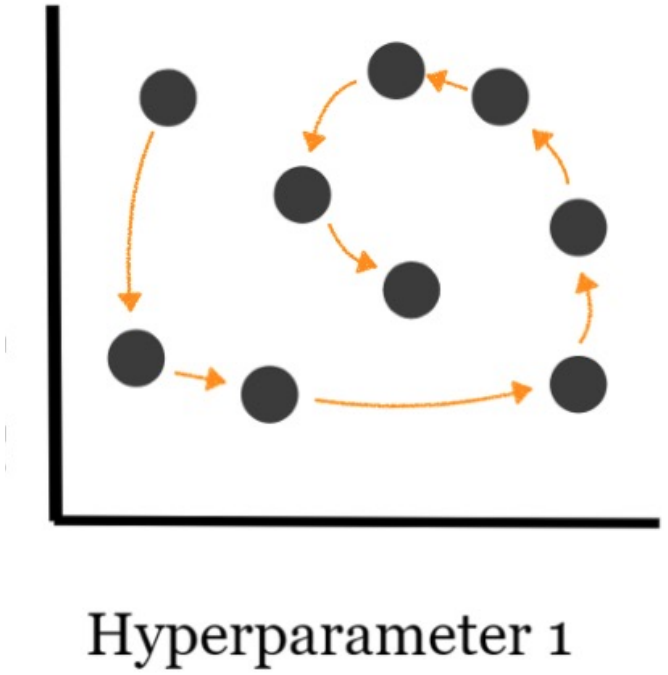
Hyper-parameters optimization algorithms



GridSearch
(*simple*)



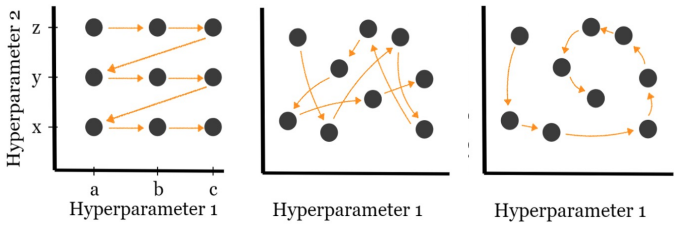
RandomSearch
(*scalable*)



BayesianSearch
(*efficient*)

Hyper-Parameter Tuning and Cross-Validation

	Computational Demanding	Model Performance	Benefits from opt parameters?
Grid Search			
Random Search			
Bayesian Search			



Which is the best method?

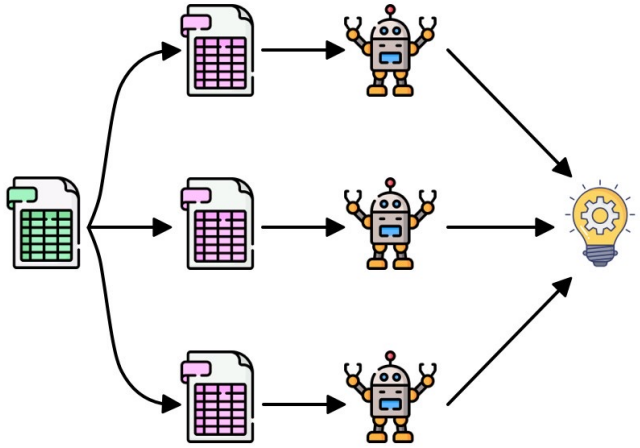
We cannot come to a consensus. The ideal hyperparameter tuning method depends on the use case

*P.S. I promise I will try to use **Bayesian Search** more frequently !!!!!!*

Ensembling

Ensemble learning is a machine learning paradigm where multiple models (even “weak learners”) are trained to solve the same problem and combined to get better results. The main hypothesis is that when weak models are correctly combined we can obtain more accurate and/or robust models.

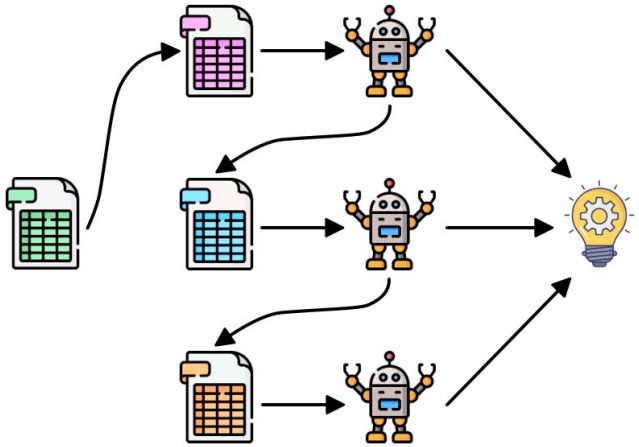
Bagging



Parallel and Scalable

Es. Random Forest

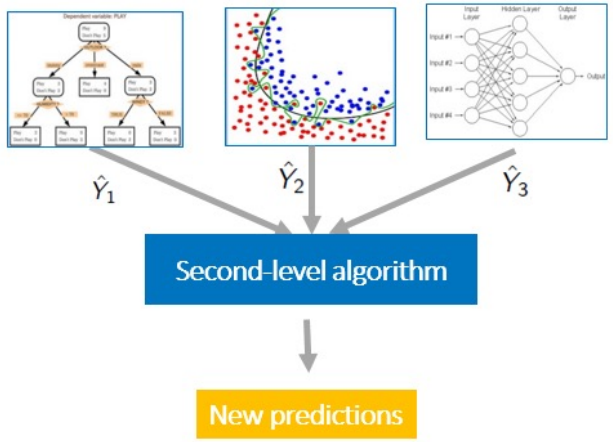
Boosting



Sequential

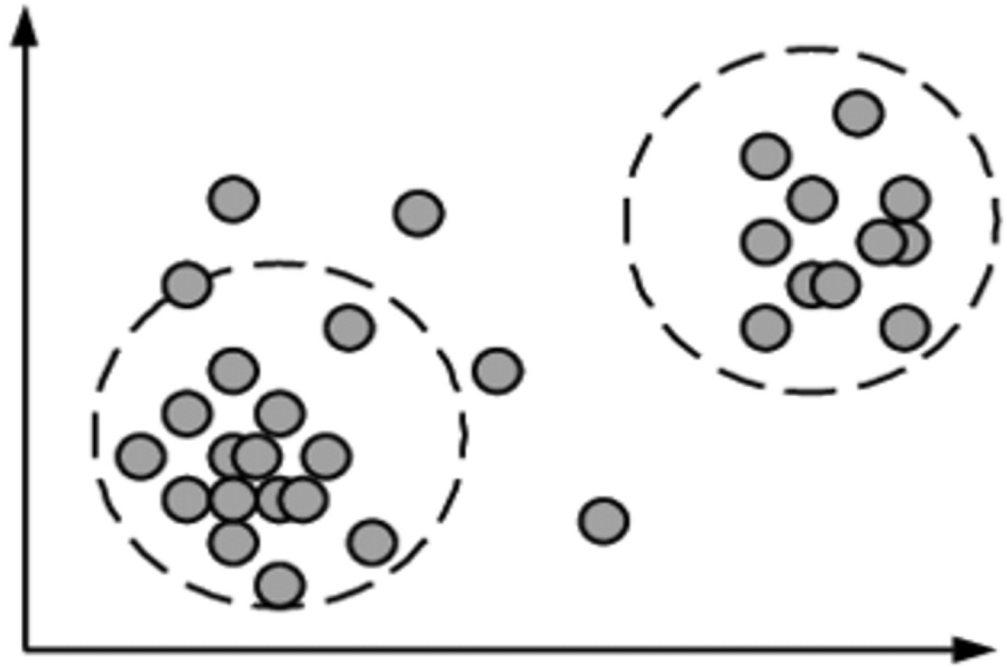
Es. Adaboost

Stacking



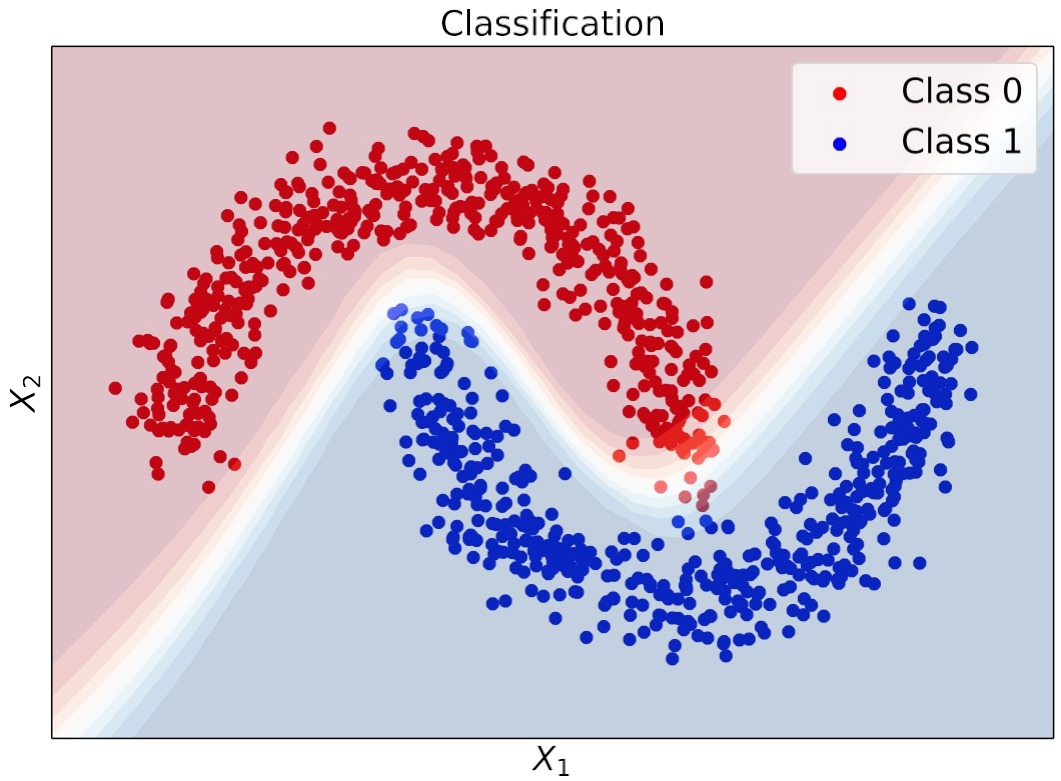
Multiple approaches can be integrated, but care needs to be taken when training the “Meta Learner”

Unsupervised



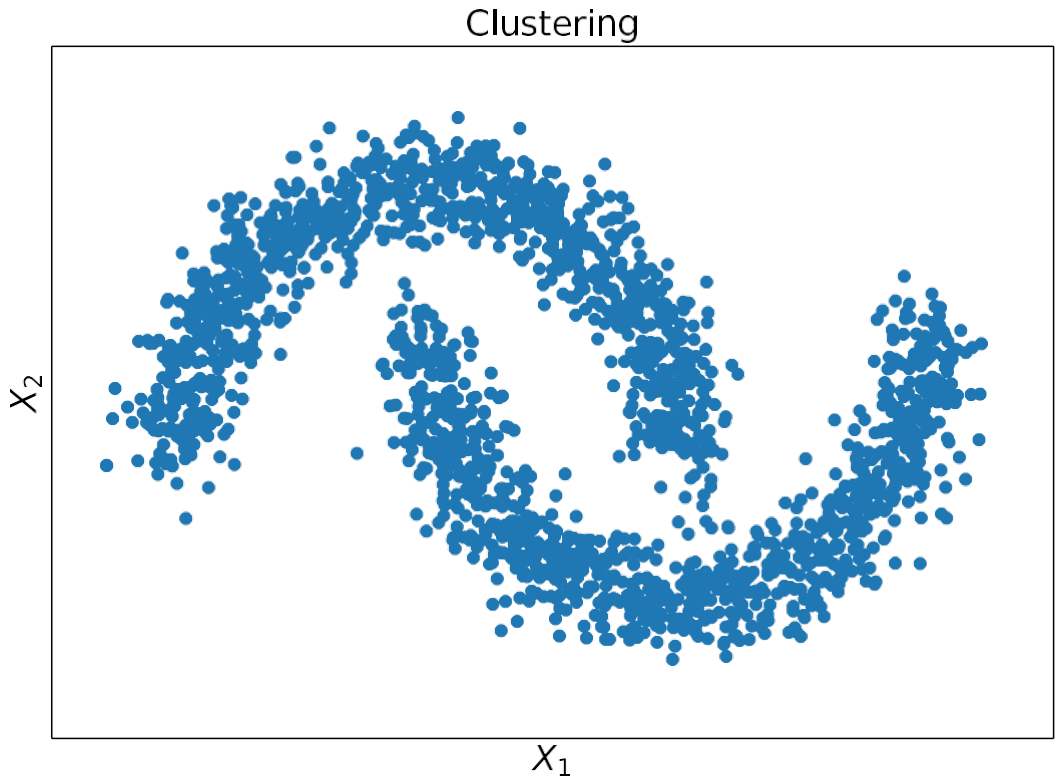
Clustering vs classification

- In classification, we have object features X and class labels $y \in \{0, 1\}$
- A classifier learns decision rule f, so that $f(X) \approx y$
- The trained classifier predicts class labels for new objects



Clustering vs classification

- In clustering, we don't have class labels y
- The goal is to divide all objects into separate groups using only object features X
- Objects inside groups are similar
- Objects from different groups are dissimilar



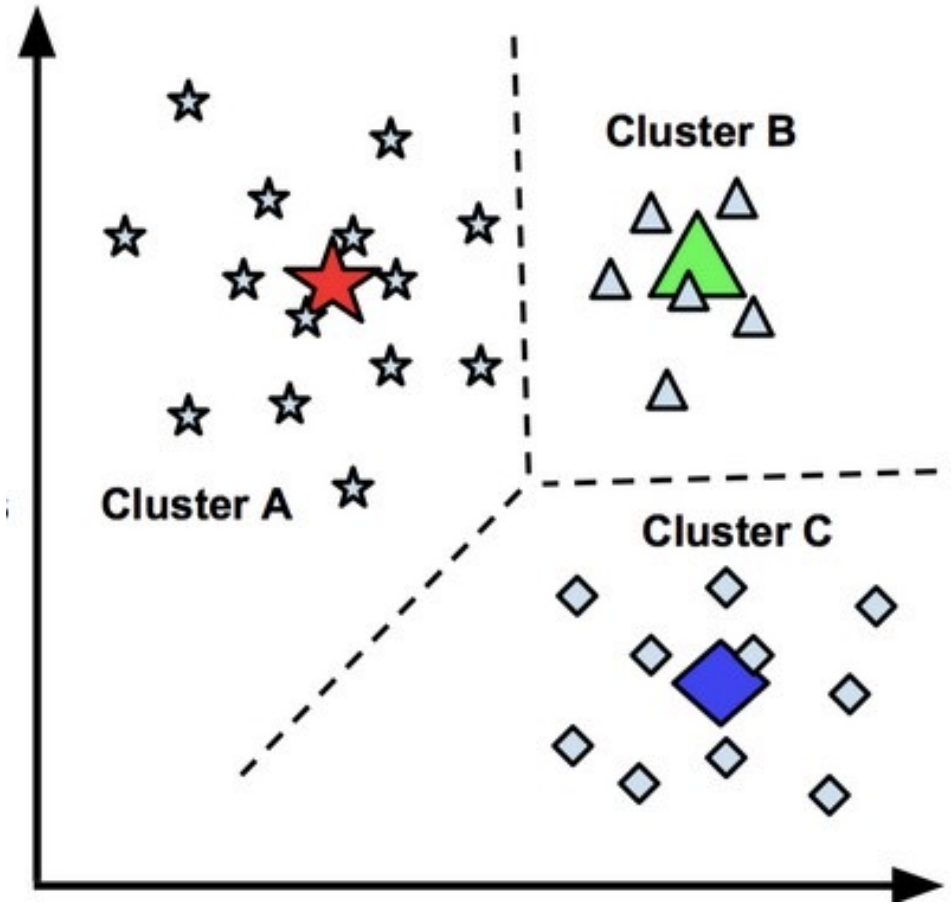
Clustering assumptions

Most of clustering algorithms are based on the following assumptions:

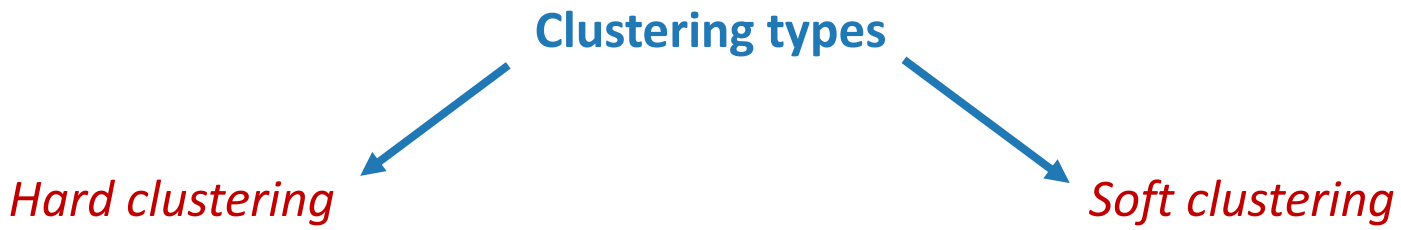
- Objects form dense clusters
- Objects from one cluster are similar
- Objects from different clusters are dissimilar
- Objects similarity is often based on distance between them
- Distances between neighbors within one cluster are smaller than between objects from different clusters

Clustering intuition

- Each cluster is represented by its center
- All objects are assigned to the closest center
- The goal is to find such centers that form the most compact clusters



Clustering algorithms



- **Connectivity models**: for example, hierarchical clustering builds models based on distance connectivity.
- **Centroid models**: for example, the k-means algorithm represents each cluster by a single mean vector.
- **Distribution models**: clusters are modelled using statistical distributions, such as multivariate normal distribution and parameters are fitted using expectation-maximization algorithm.
- **Density models**: for example, DBSCAN and OPTICS defines clusters as connected dense regions in the data space.
- **Graph-based models**: based on representing data as a graph and then applying clustering techniques on the generated graph
- **Neural models**: clustering obtained using neural networks, via combination of non-linear and linear transformations

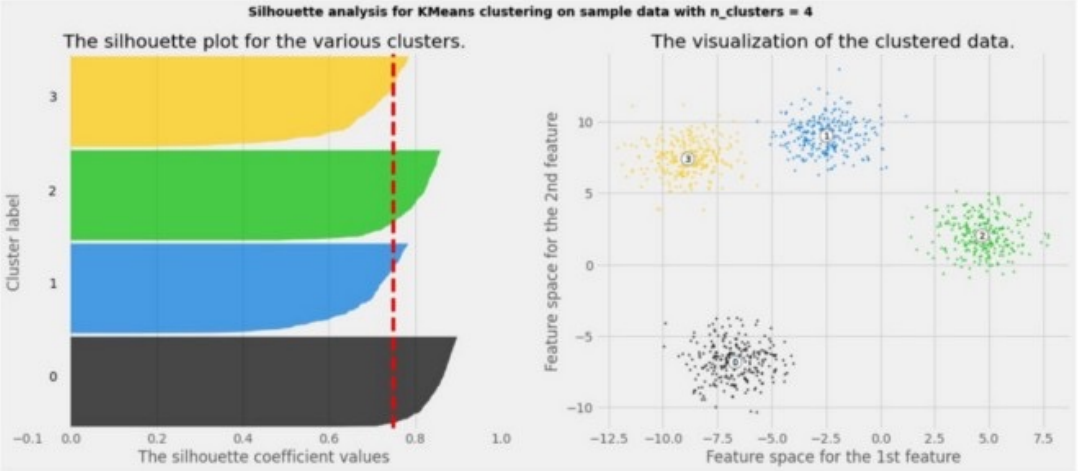
Metrics and optimal number of clusters

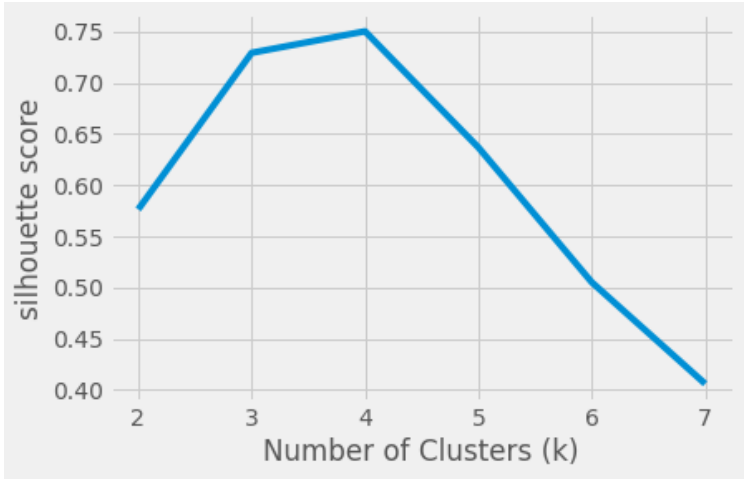
Silhouette is unsupervised quality metric defined as:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{d_i - s_i}{\max(d_i, s_i)}$$

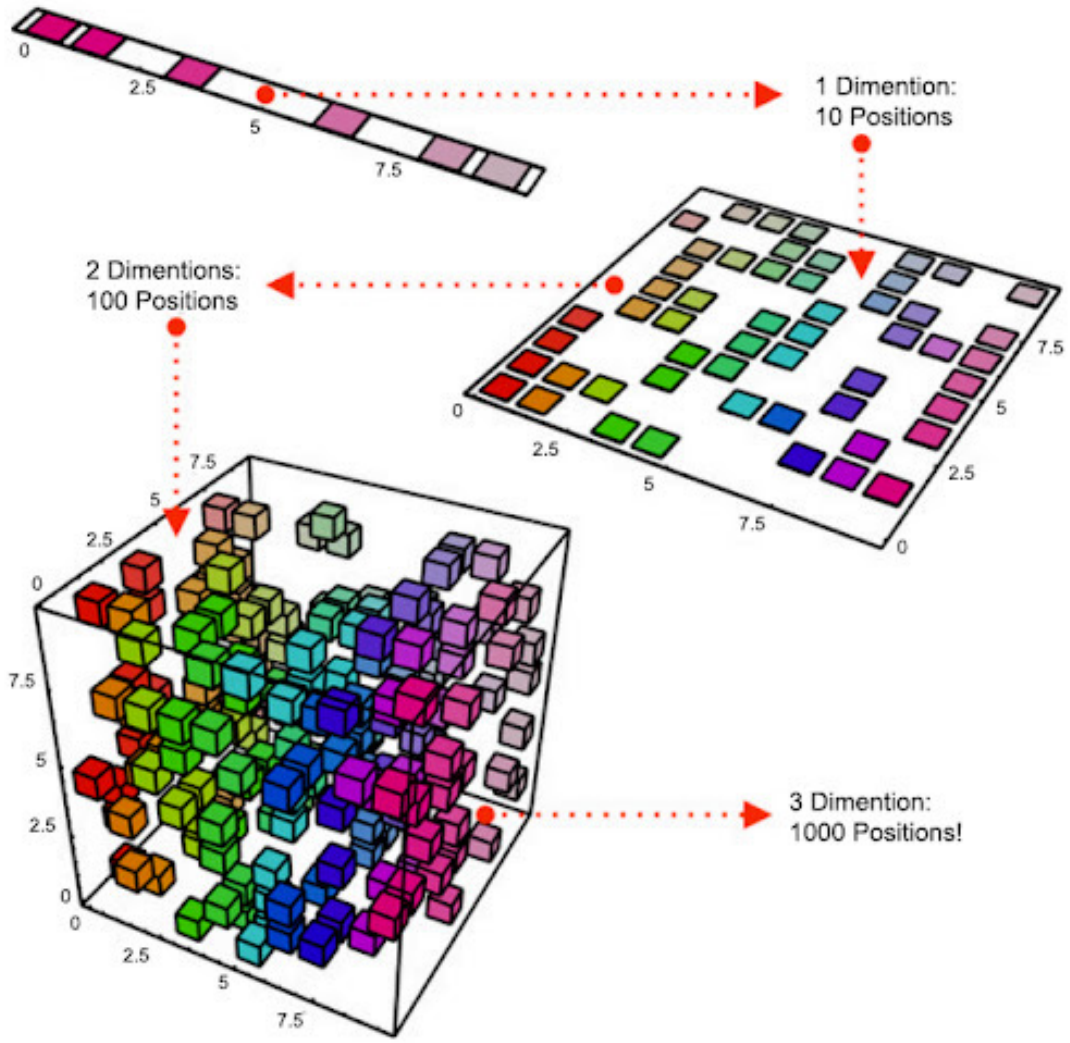
s_i - mean distance between the i-th object and all objects in the same cluster,

d_i - mean distance between the i-th object and all objects in the nearest cluster.





Curse of dimensionality



With ML model and data analysis, we might be tempted to use large number of independent variables, to allow the model to have full information but there are potential issues arising from considering huge number of dimensions:

1. **Large Capacity and Overfitting**
   If we have more features than observations than we run the risk of massively overfitting our model — this would generally result in terrible out of sample performance.

2. **Data Sparsity**
   When we have too many features, observations become harder to cluster — believe it or not, too many dimensions causes every observation in your dataset to be very sparse and appear (almost) equidistant from all the others in a Euclidean sense.

Dimensionality reduction

Data Sparsity
Especially when dealing with one-hot-encoded categorical features (many feature or many classes), feature space may be a very large sparse matrix.

Correlated data
There are a large number of correlated variables, which makes training the model less efficient and somewhat undetermined

AIM: Compress relevant information (with linear and non linear combination of variables) into a low-dimensional space

Methods:

- **Clustering**
- **Principal Component Analysis**
- **Matrix Factorization**
- **T-SNE**
- **Autoencoders**
- **Embeddings**

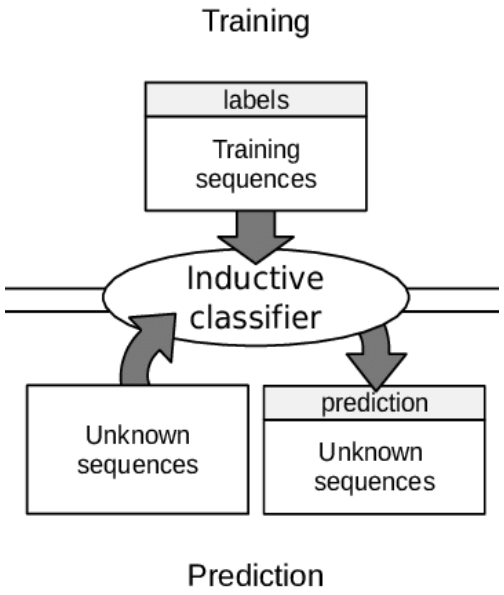
Inductive or Transductive?

Inductive

Induction, in the context of learning, is the attempted discovery of rules/generalizations based on analysis of collected data.

The main characteristic of inductive learning is the **building of a model** (inductive classifier), that is then used to classify / make prediction on unobserved samples

We can learn and progressively update our inductive classifier (see Neural Networks)

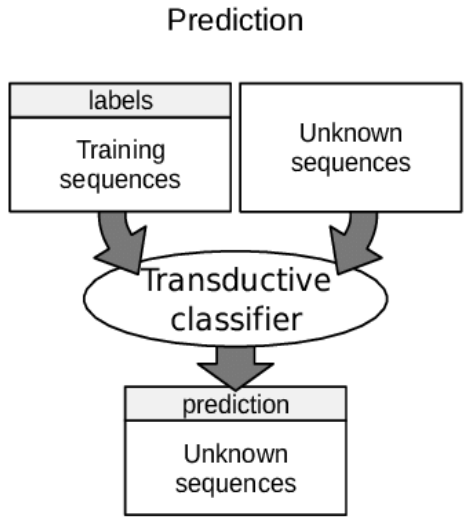


Transductive

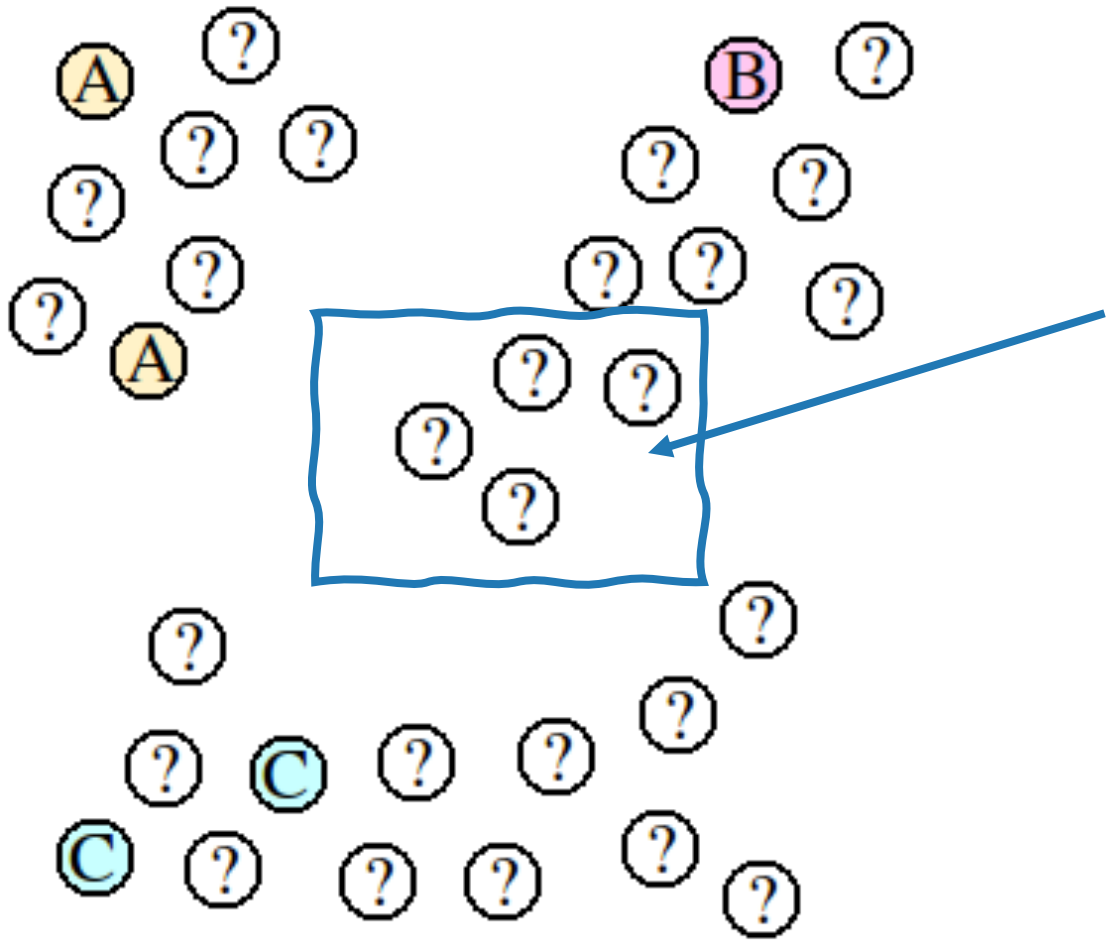
Transduction or **transductive inference** is reasoning from observed, specific (training) cases to specific (test) cases. No general model is built, but the inference happens on the **very specific** dataset of labelled and unlabelled data.

Every time one wants to classify a new set of instances, the whole training must be **re-done** again.

It should be used when there are only few labelled datapoint (**semi-supervised** problem)



Inductive or Transductive?



What are the labels of the nodes at the center?