Module 1 :

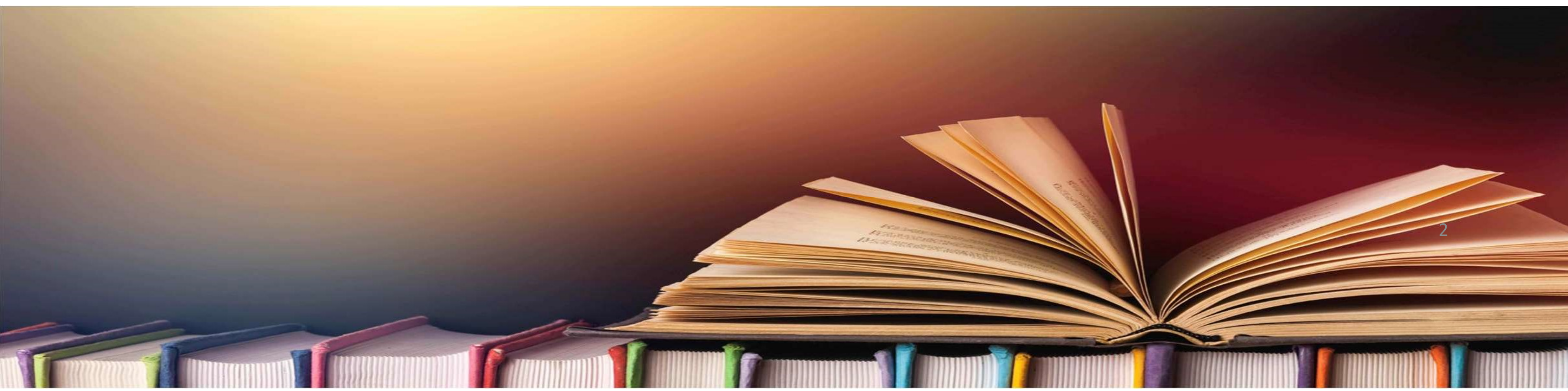
Machine Learning Review

Machine Learning Introduction

Géraldine Conti, August 2020

Bibliography

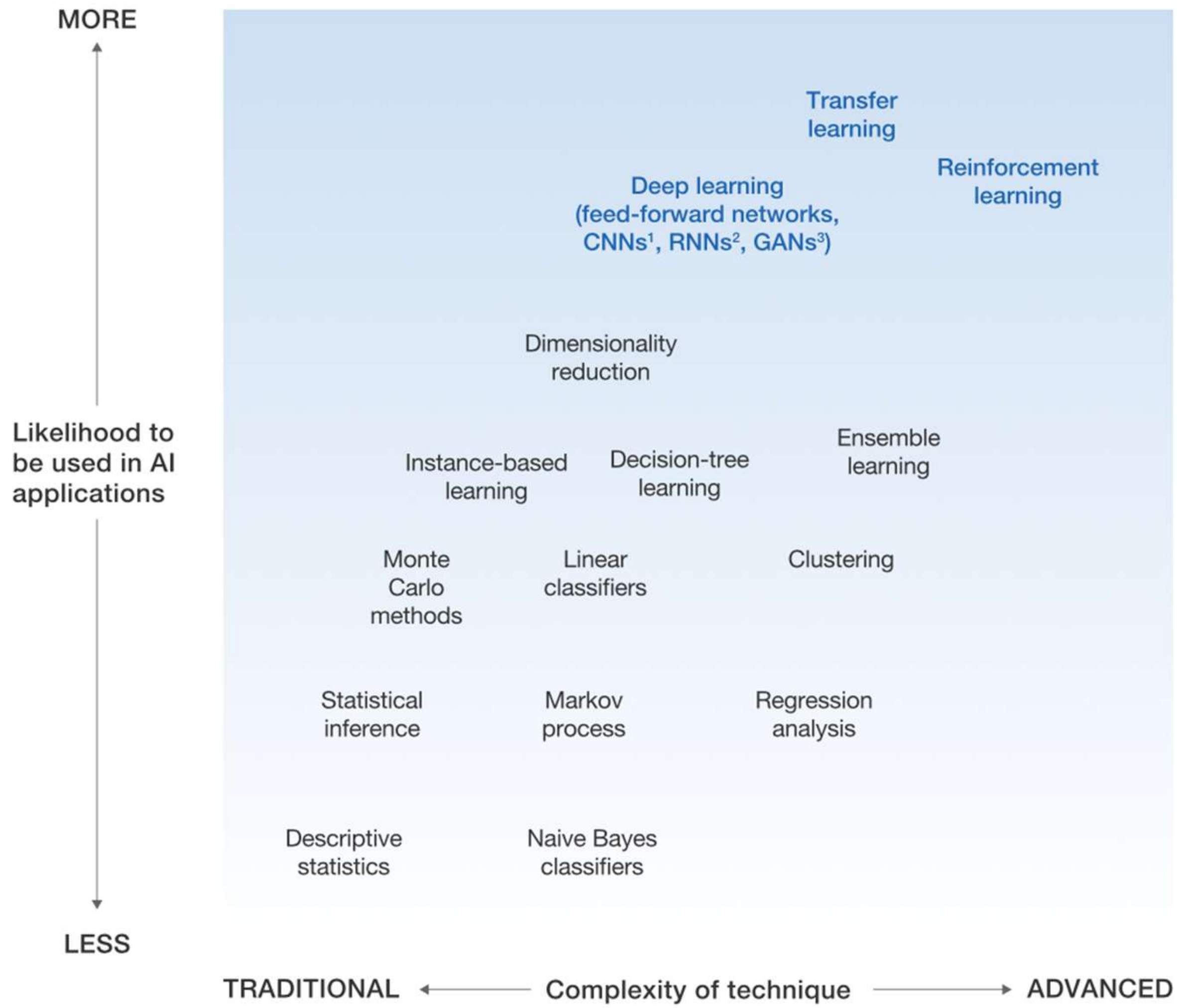
- Deep Learning book (Goodfellow, Bengio, Courville)
- Machine Learning @ Stanford (Prof Andrew Ng)
- Hands-On Machine Learning with Scikit-Learn & Tensorflow (Aurélien Géron)

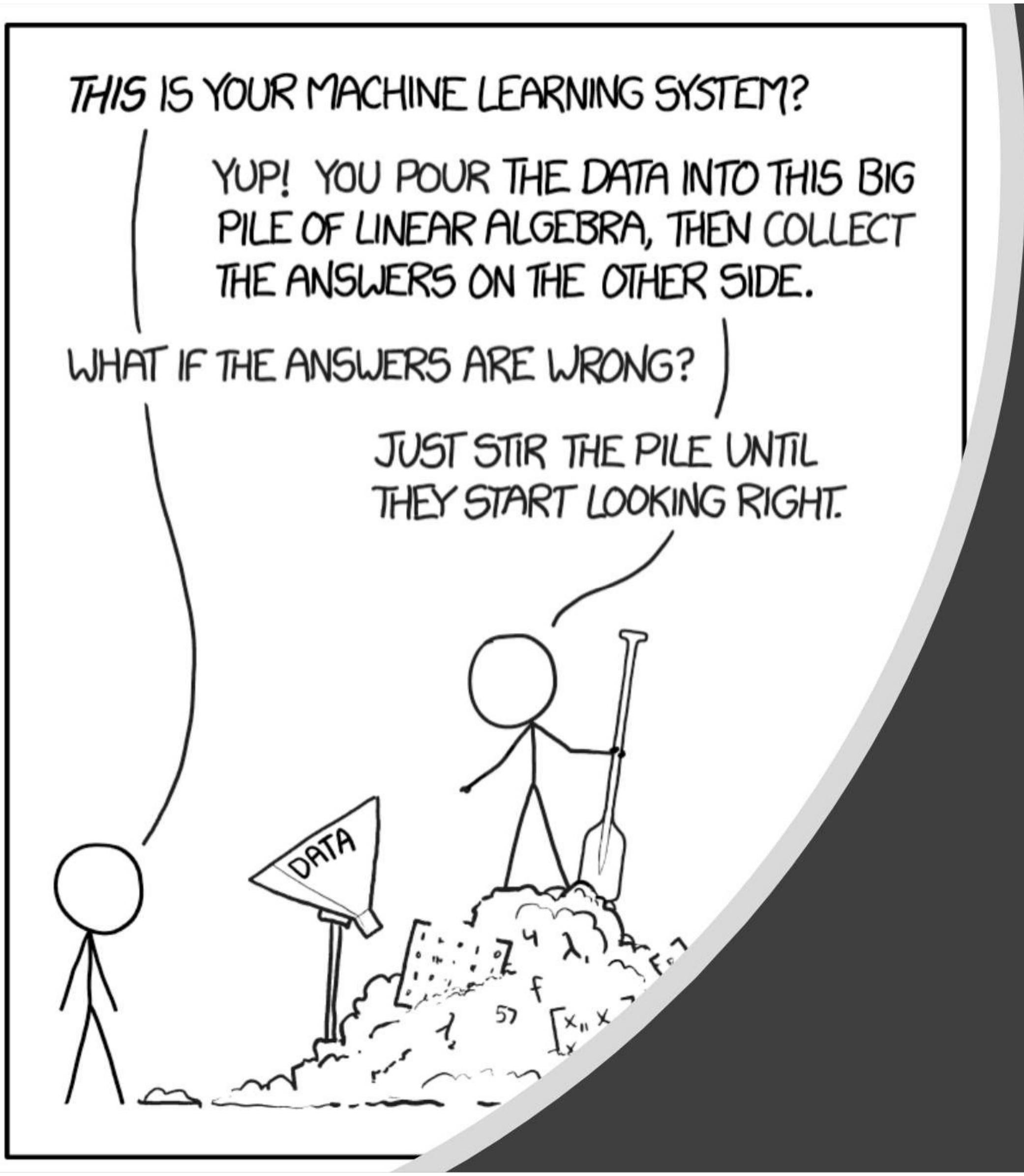


Learning Objectives

- What is Machine Learning ?
- Types of ML systems
- Main Challenges
- Data Preparation
- Optimization
- How to choose an algorithm ?
- Technical Details

Differences between ML and Statistical Modeling

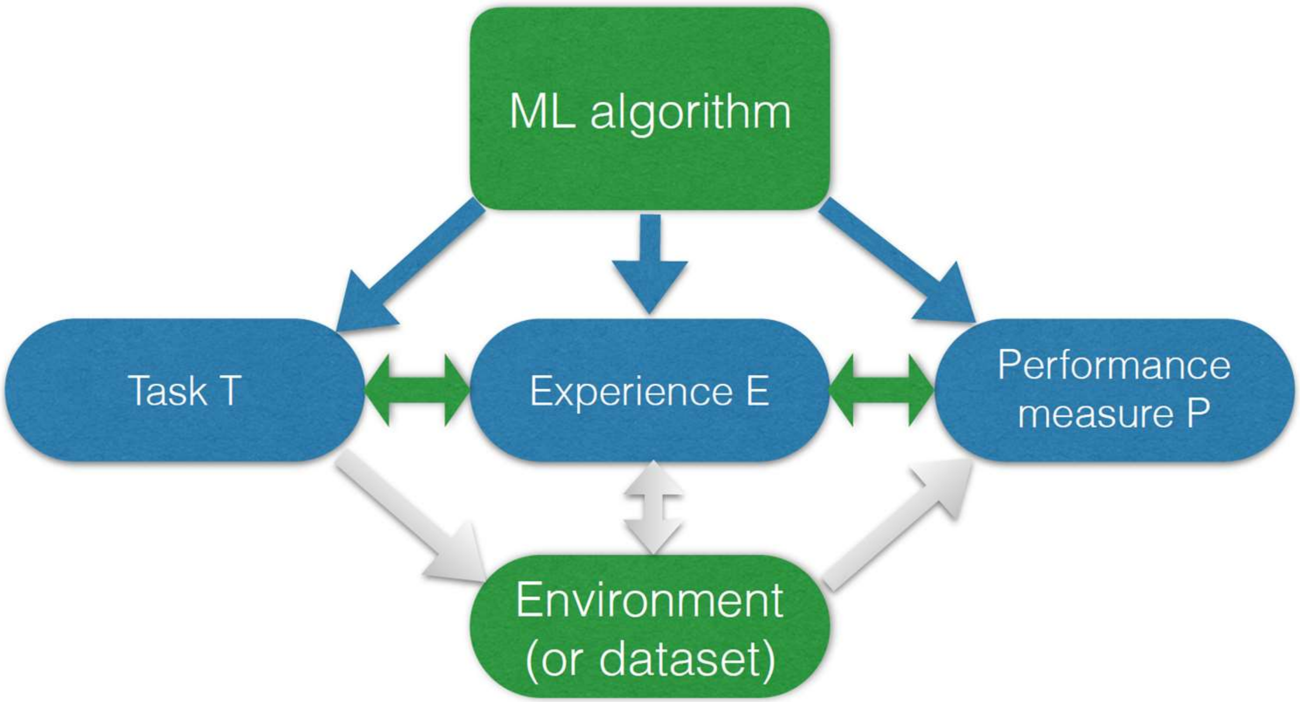




What is Machine Learning ?

"Can machines do what we (as thinking entities) can do?" A. Turing

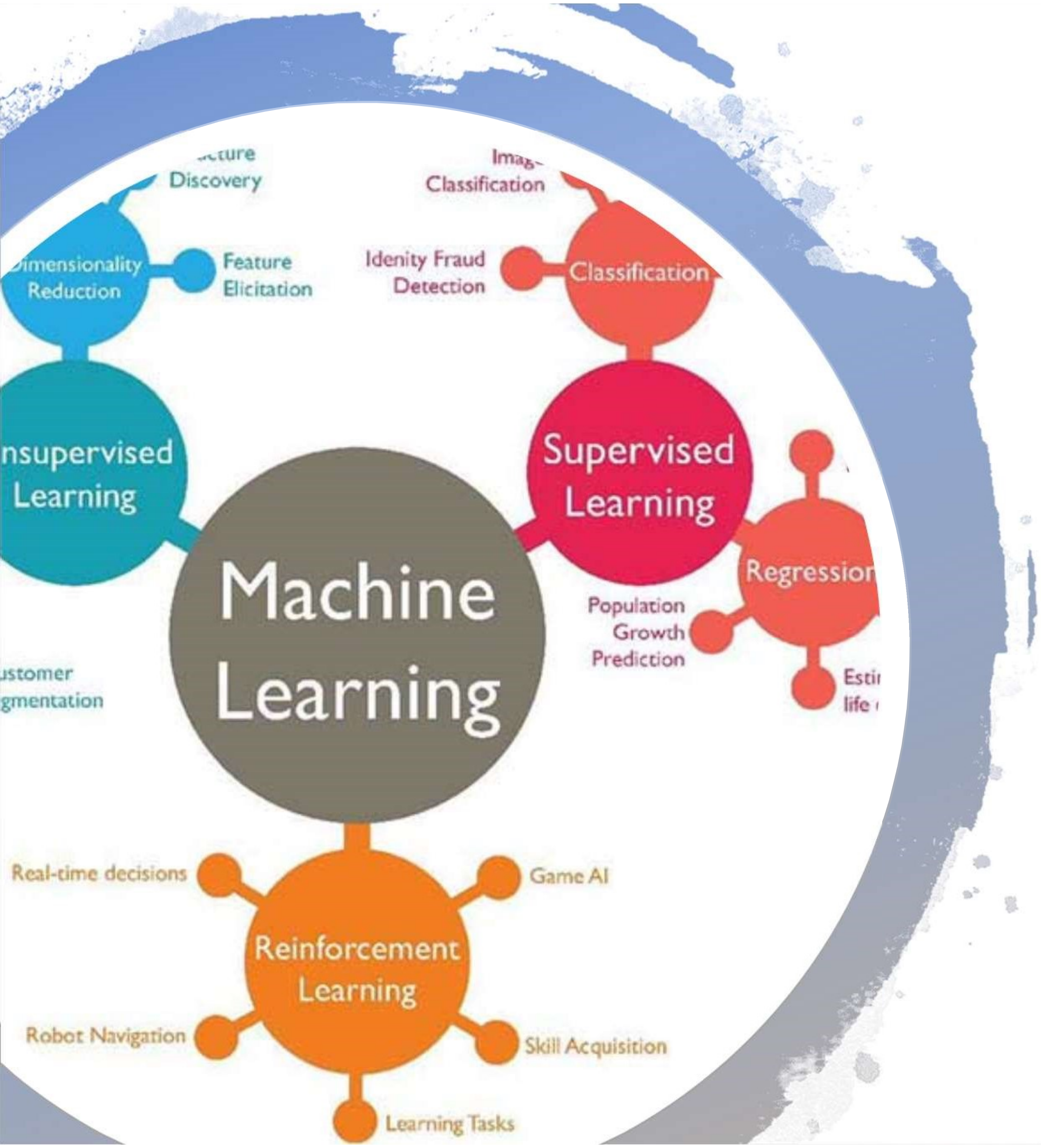
Definition



« A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. » Tom M. Mitchell (1997)

Machine Learning Use Cases

- Problems for which existing solutions require a lot of hand-tuning of long lists of rules
  - ML *simplifies* code and *performs better*
- Complex problems for which there is no good solution at all using a traditional approach
  - ML can *find a solution*
- Fluctuating environments
  - ML system can *adapt* to new data
- Getting insights about complex problems and large amounts of data



Types of Machine Learning Systems

Online learning

Supervised learning

Machine Learning Systems

Batch learning

Unsupervised learning

instance-based learning

Model-based learning

Trained with human supervision (learning pillars) ?

NO **YES**

Semi-supervised learning

Can learn incrementally on the fly ?

NO **YES**

Compare new data points to known data points (or instead detect patterns) ?

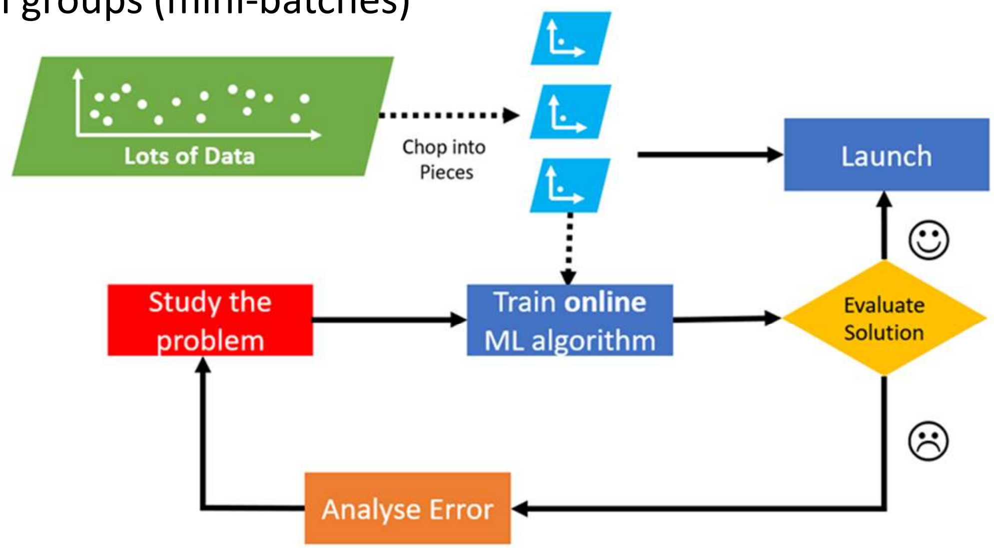
NO **YES**

Batch Learning

- Not capable of learning incrementally
- Must be trained using all the available data
- With new data (new types), need to train a new version of the system from scratch on the full dataset
- Advantage :
- Drawbacks:

Online
Learning

- Train the system incrementally by feeding data instances sequentially, system can learn about new data on the fly
  - Individually or in small groups (mini-batches)



- Important parameter : learning rate
  - *How fast the system should adapt to changing data*
  - *High learning rate* = adapt quickly, but forgets quickly
  - *Low learning rate* = system with more inertia

- Challenge : bad data will gradually kill the performance
  - Monitor your data (anomaly detection algorithms) and performance

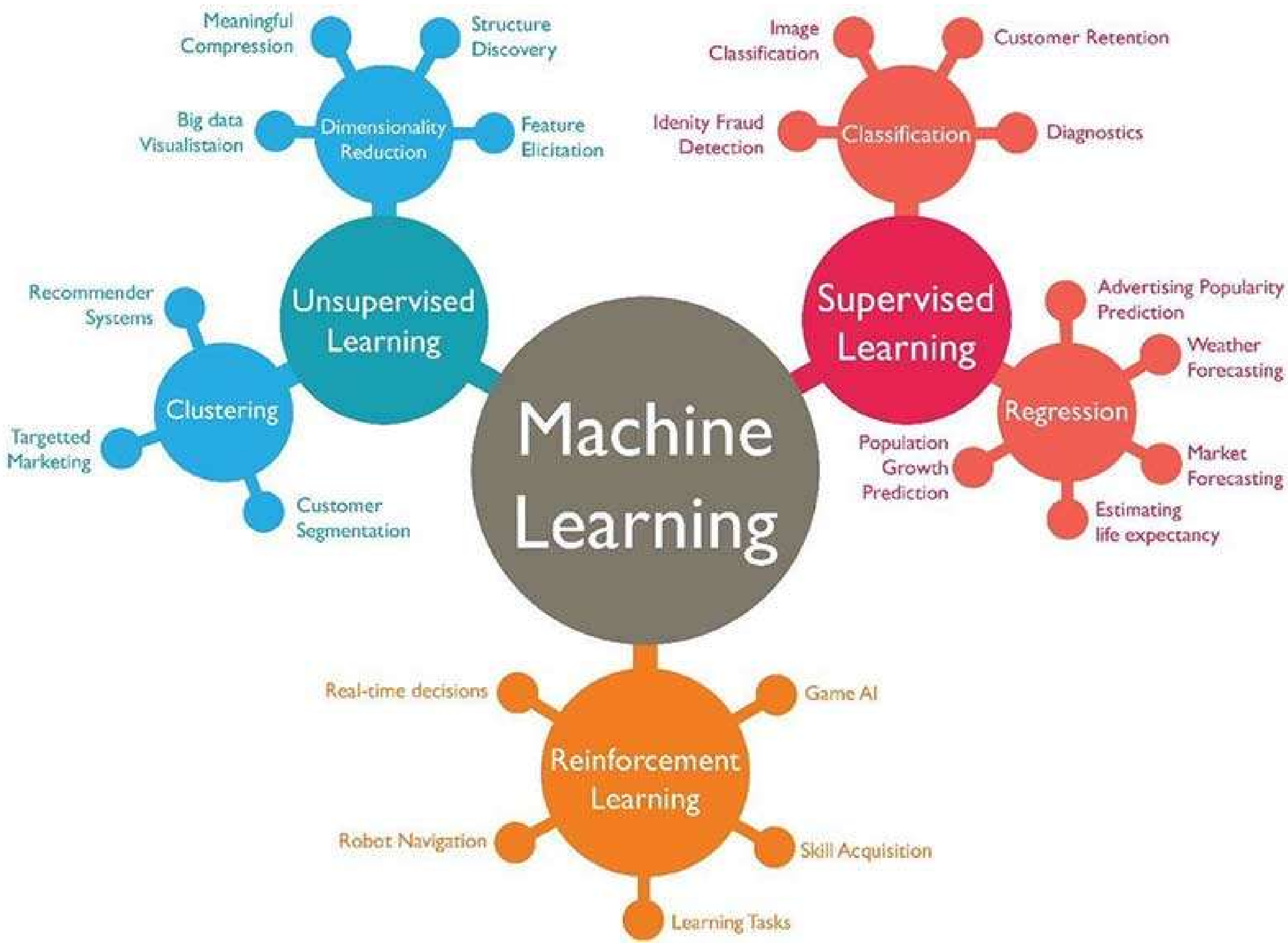
Generalization

- Instance-based Learning
  - System learns examples by heart
  - Generalizes to new cases using a similarity measure
  - Examples : k-nearest neighbors, decision trees

- Model-based learning
  - Build a model of the examples
  - Use the model to make predictions
  - Examples : Neural Networks,..

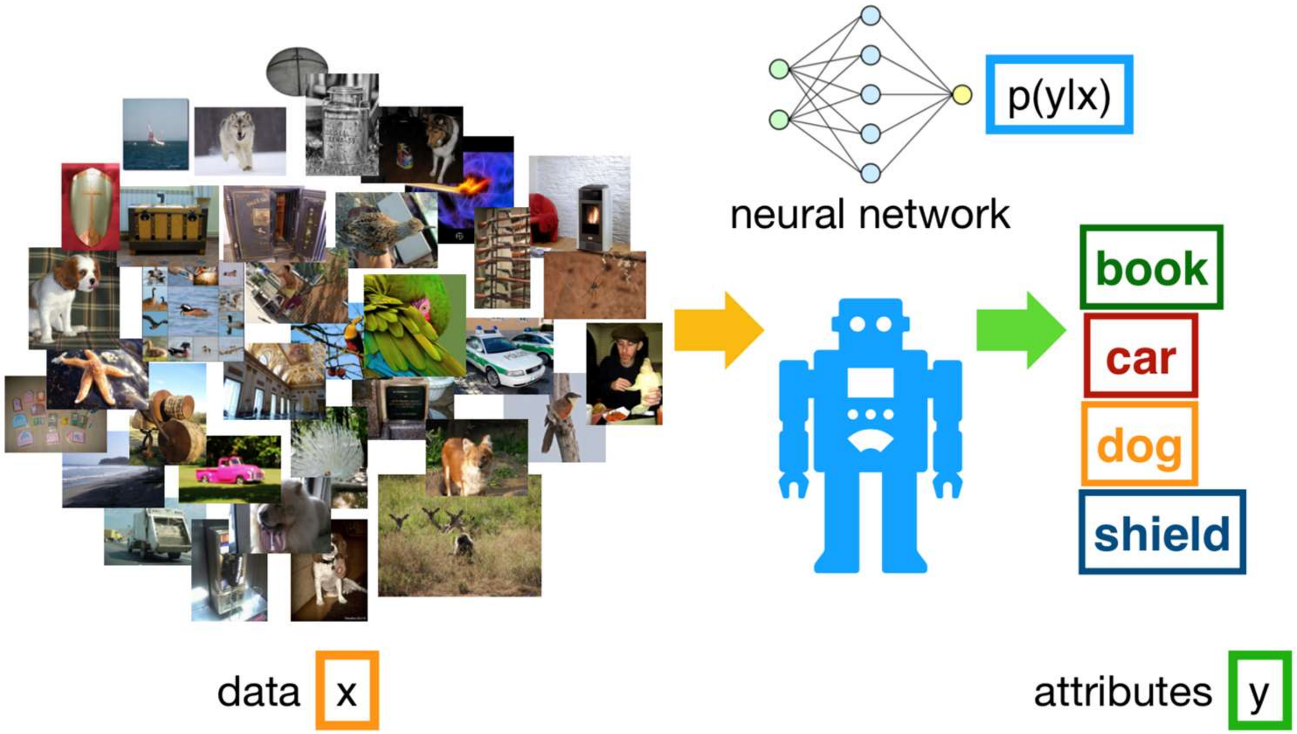
How does the number of parameters vary with the size of the training data ?

Learning Pillars



Supervised Learning

- Prediction of an output **y** given an input **x**



Regression

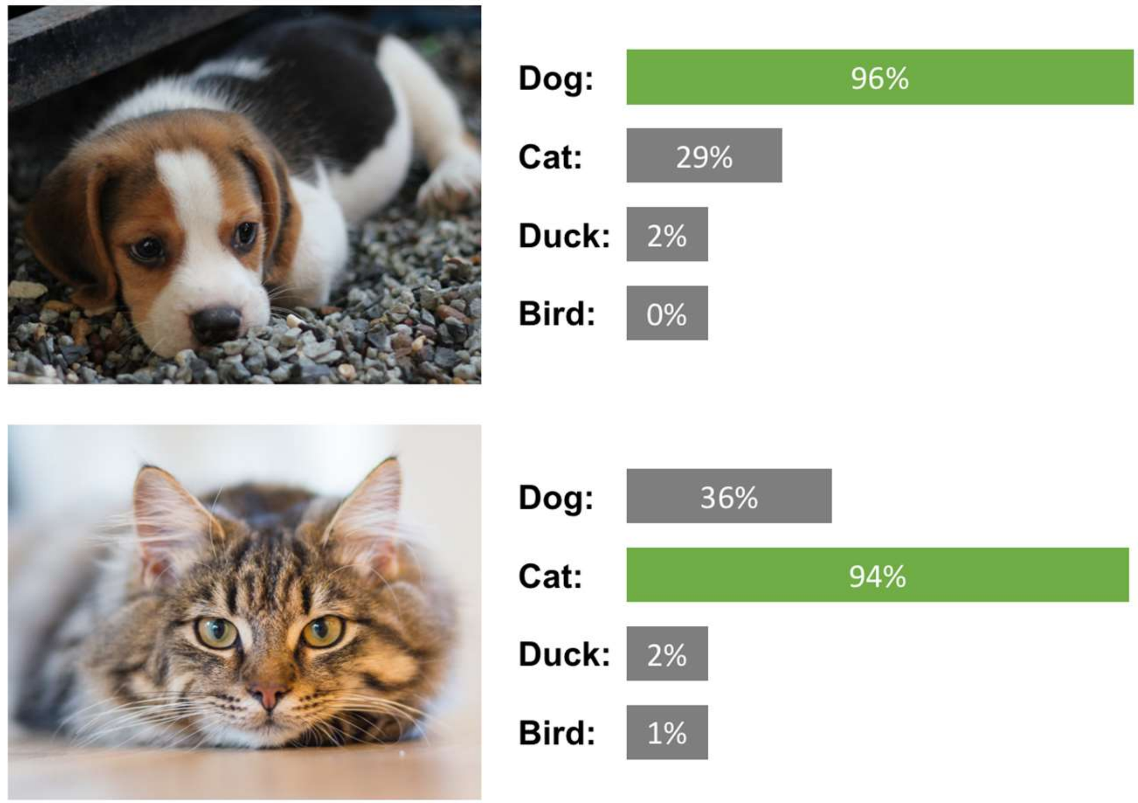
$82000

$55500

???

Predict results within *a continuous output*

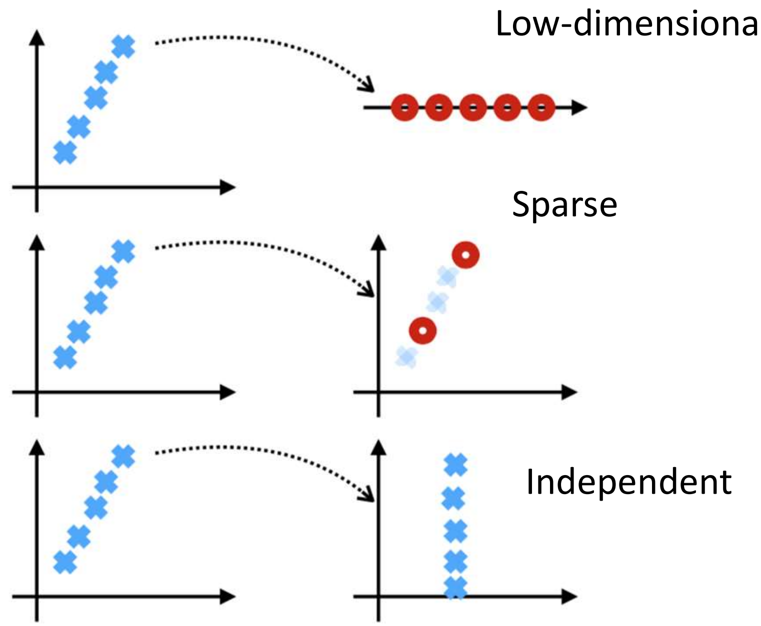
Classification

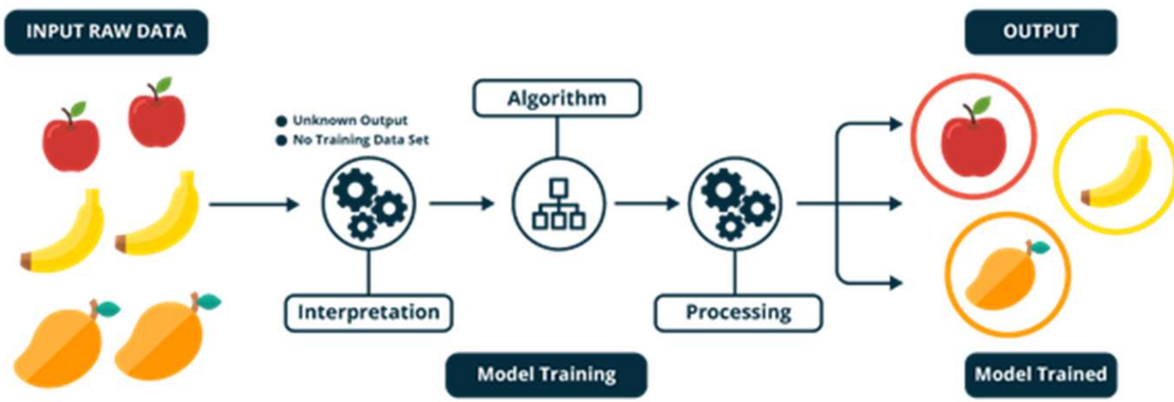


categorize new inputs as belonging to one of a set of categories
→ Predict results within *a* *discrete output (categories)*

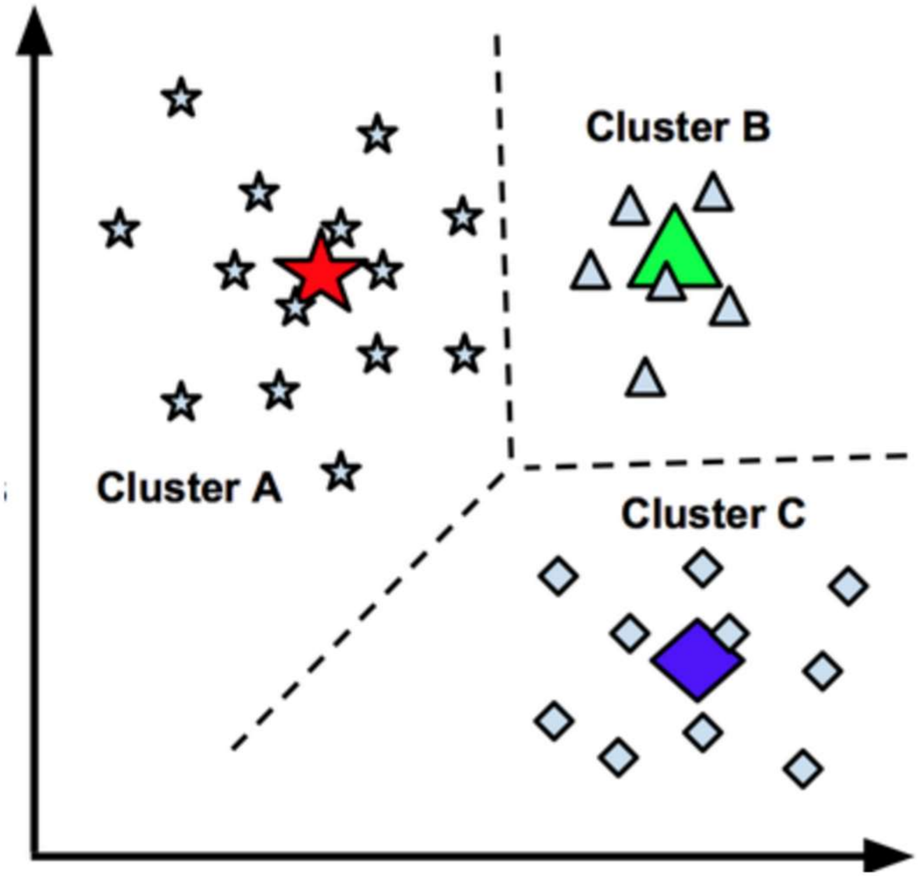
Unsupervised Learning

- Find a *suitable data representation*
  - Preserving all task-relevant information
  - Simpler than the original data and easier to use





Clustering



Create a set of categories, for which individual data instances have a set of common or similar characteristics.

Representation Learning

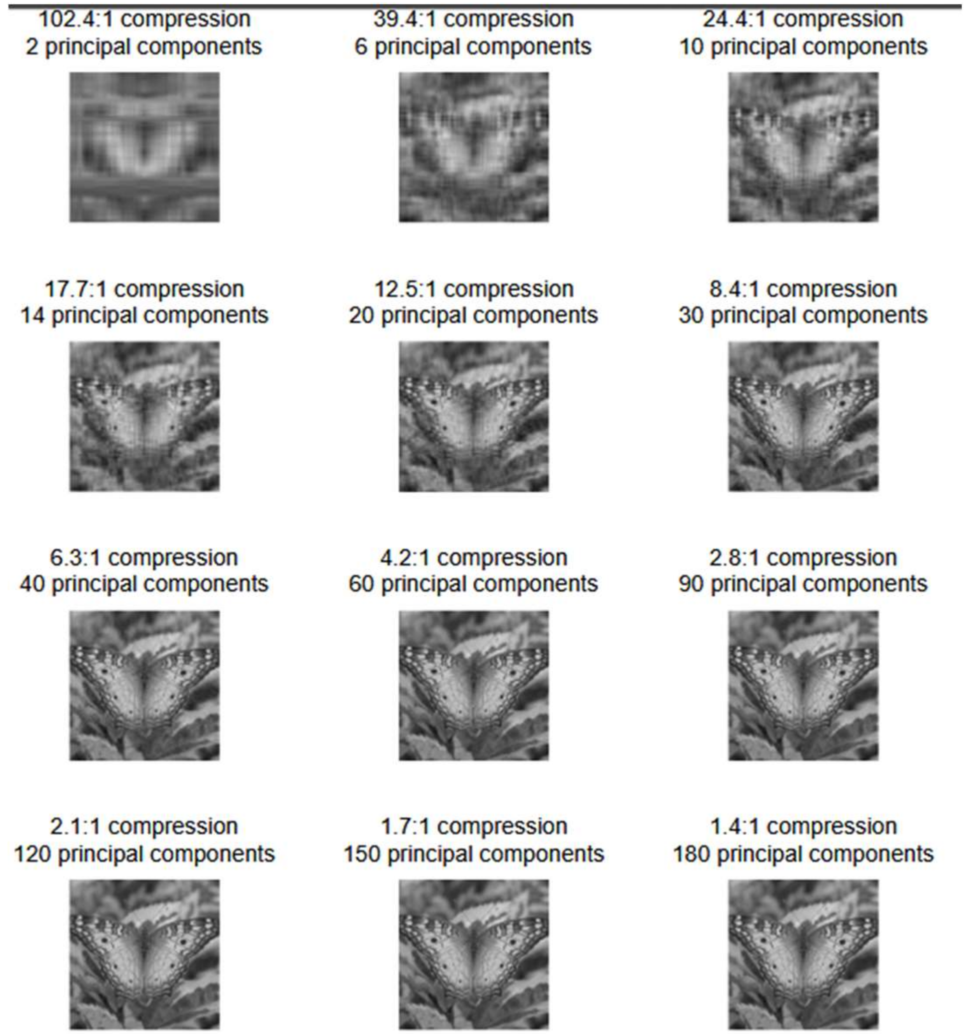
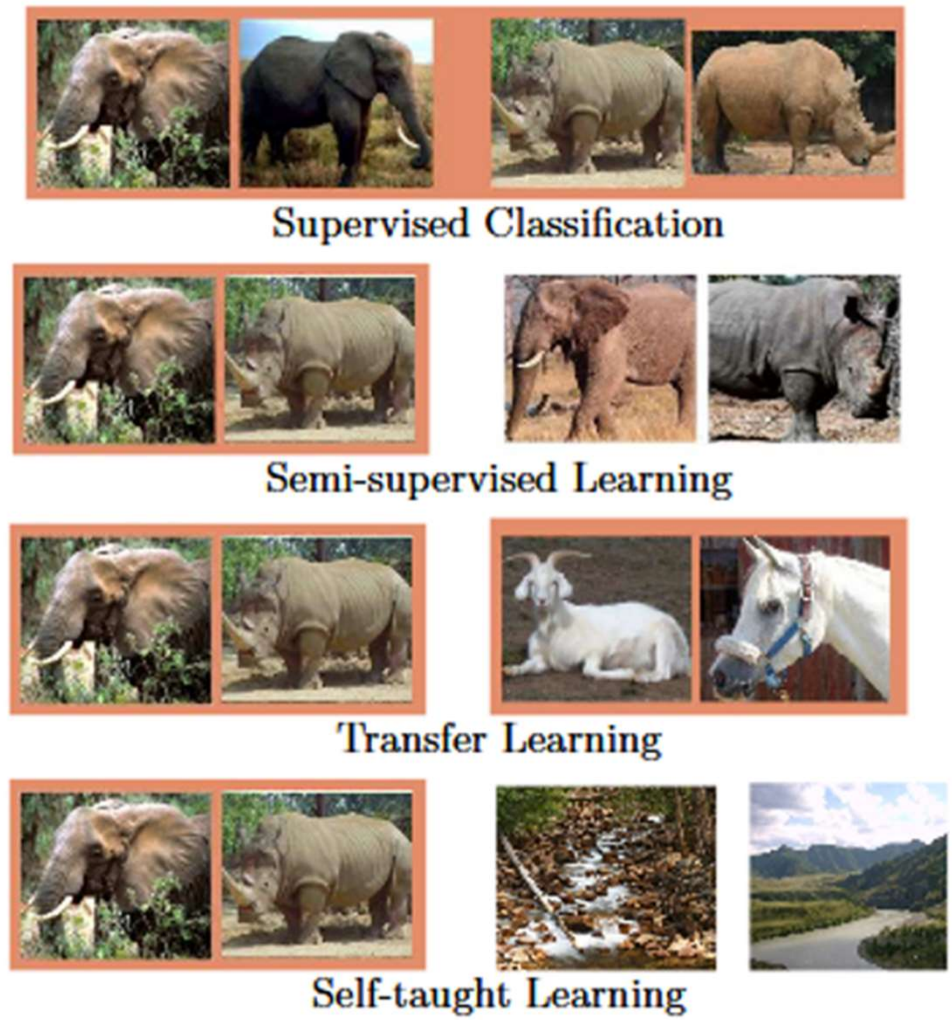


Figure 10: The visual effect of retaining principal components

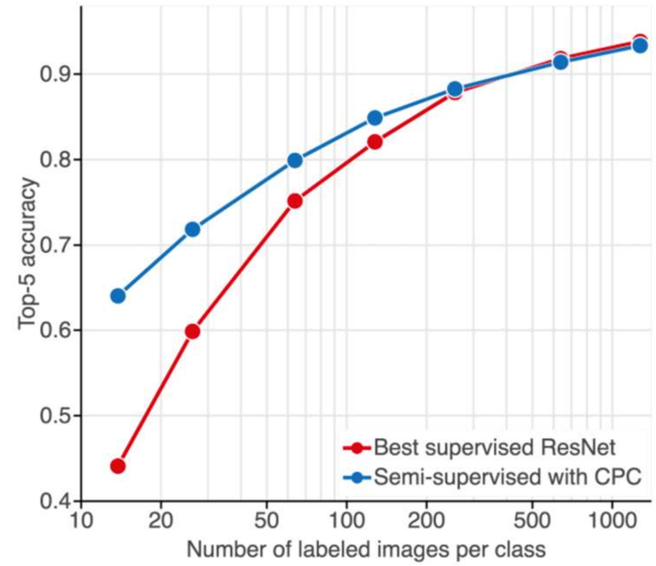
Mark Richardson (2009)

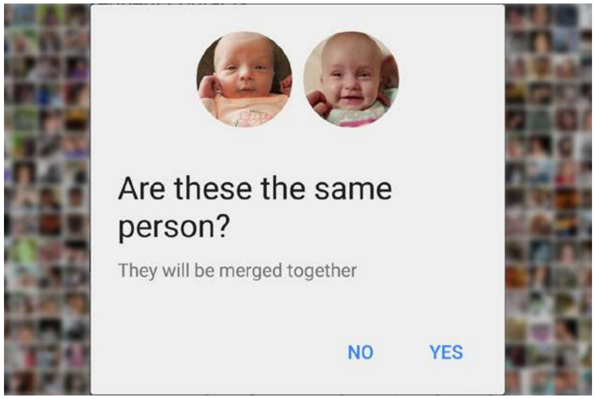
More nuances…



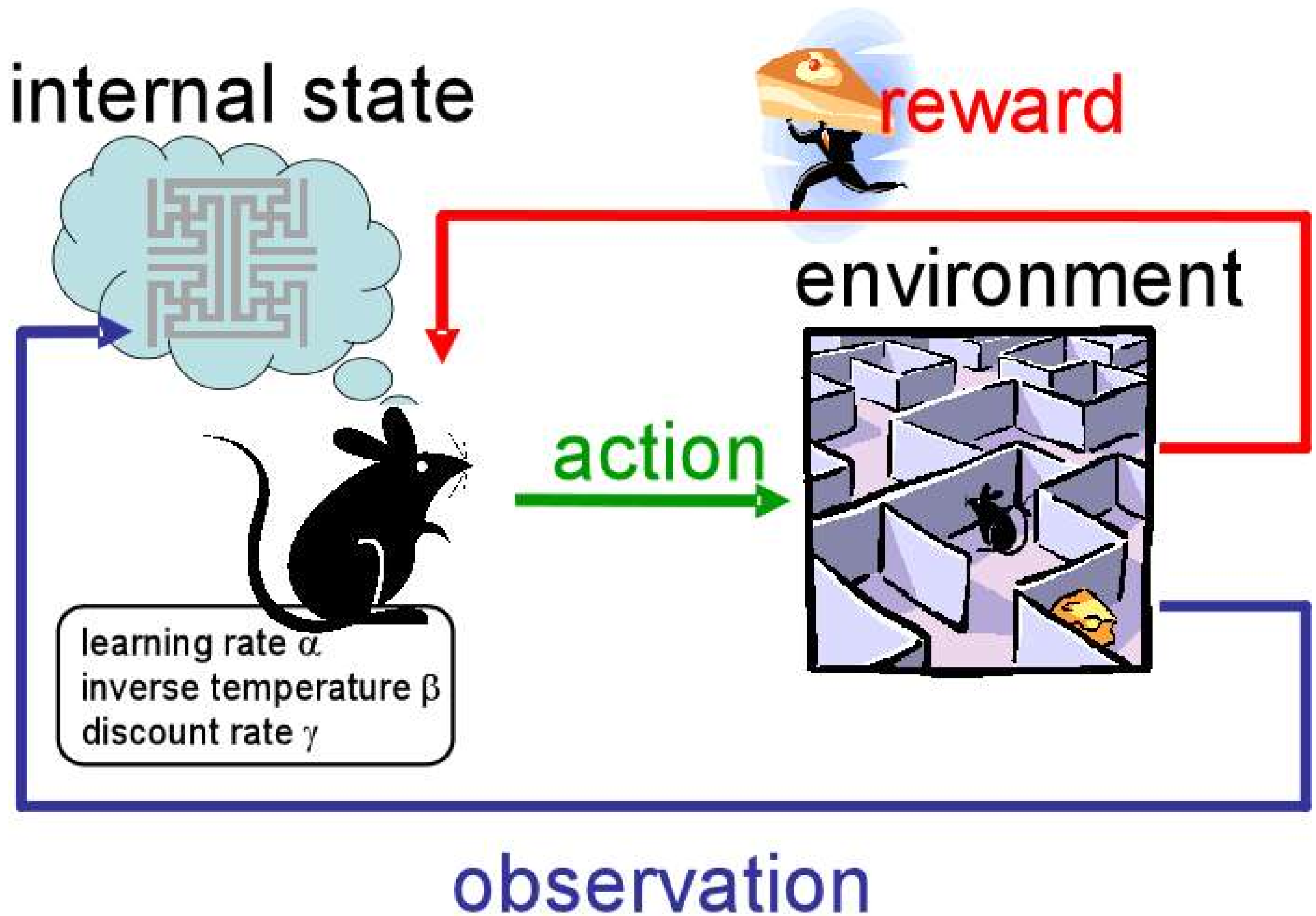
Semi-supervised Learning

- Partially labelled training data
  - Lots of unlabeled data and a little bit of labelled data
- Why bother ?
- Most semi-supervised learning algorithms are combinations of unsupervised and supervised algorithms
  - Example : Google Photos





Reinforcement Learning



Two-Minute Papers

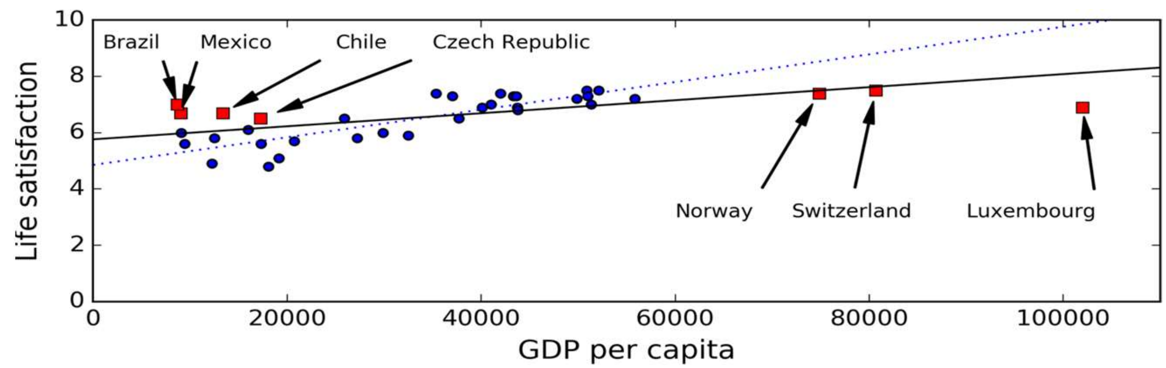
Main Challenges

- Bad data
- Bad model

Bad Data

- Insufficient training data
  - Try to get more data, data augmentation,…

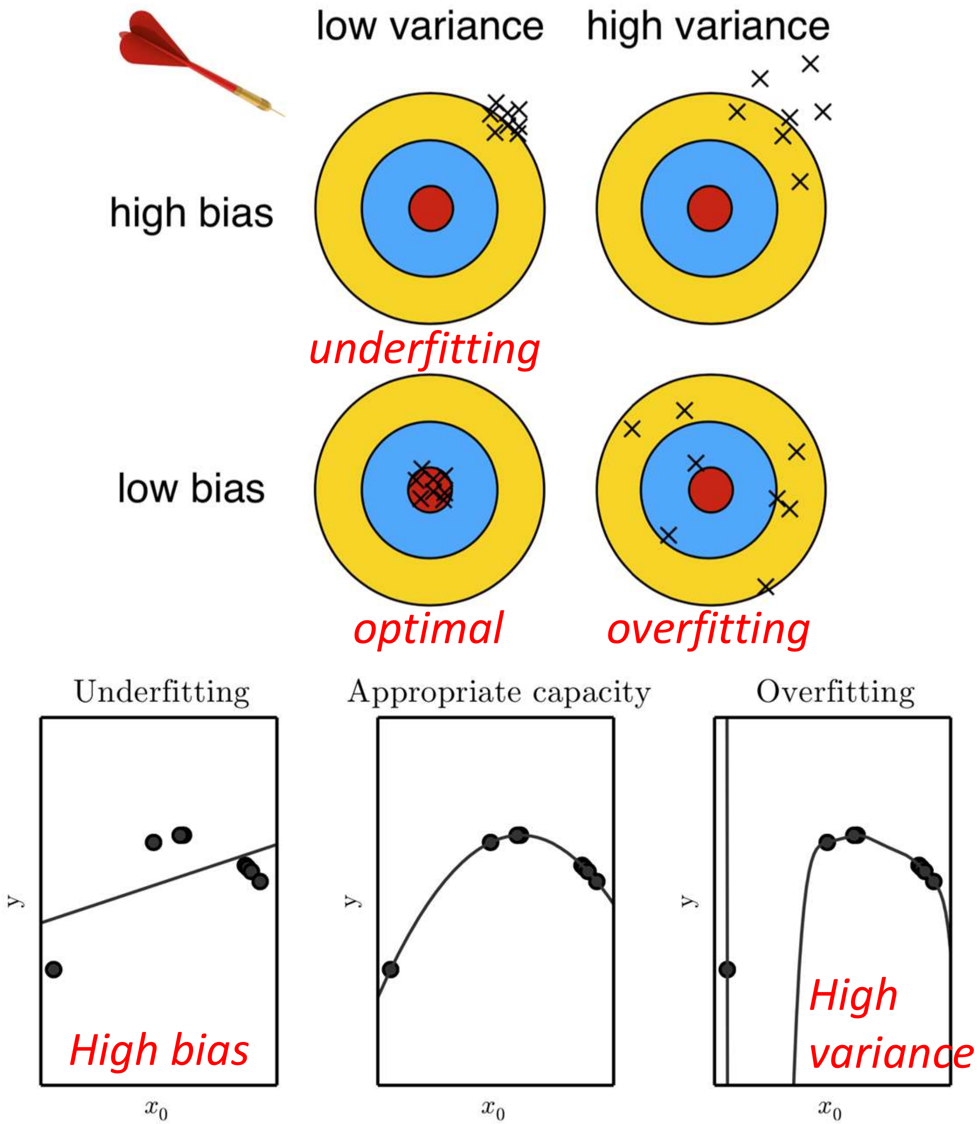
- Non-representative training data



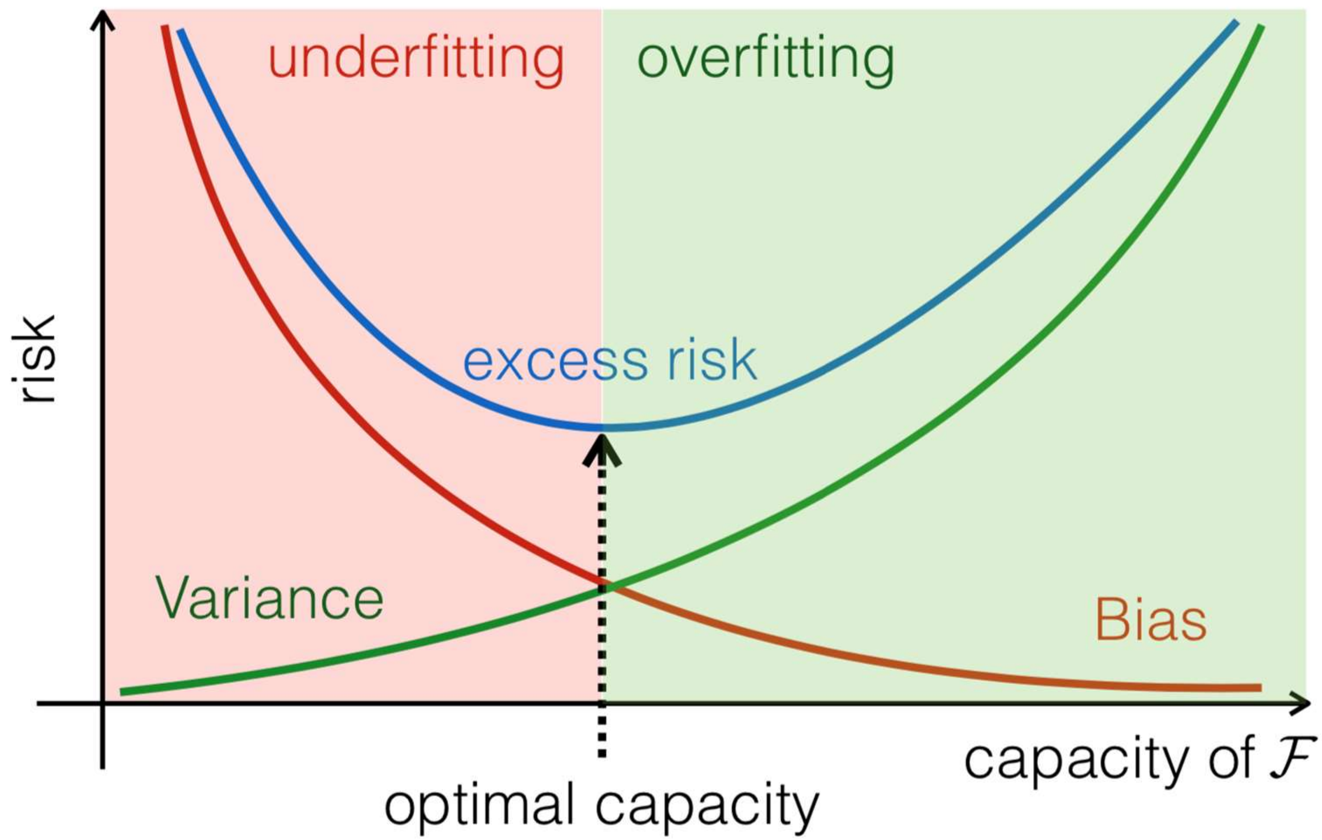
- Poor quality data
  - Data cleaning

- Irrelevant Features
  - Feature selection (select most useful features)
  - Feature extraction (combine features to produce a more useful one)

Model : Overfitting and underfitting



Model : Bias, variance and capacity



Cross-
validation

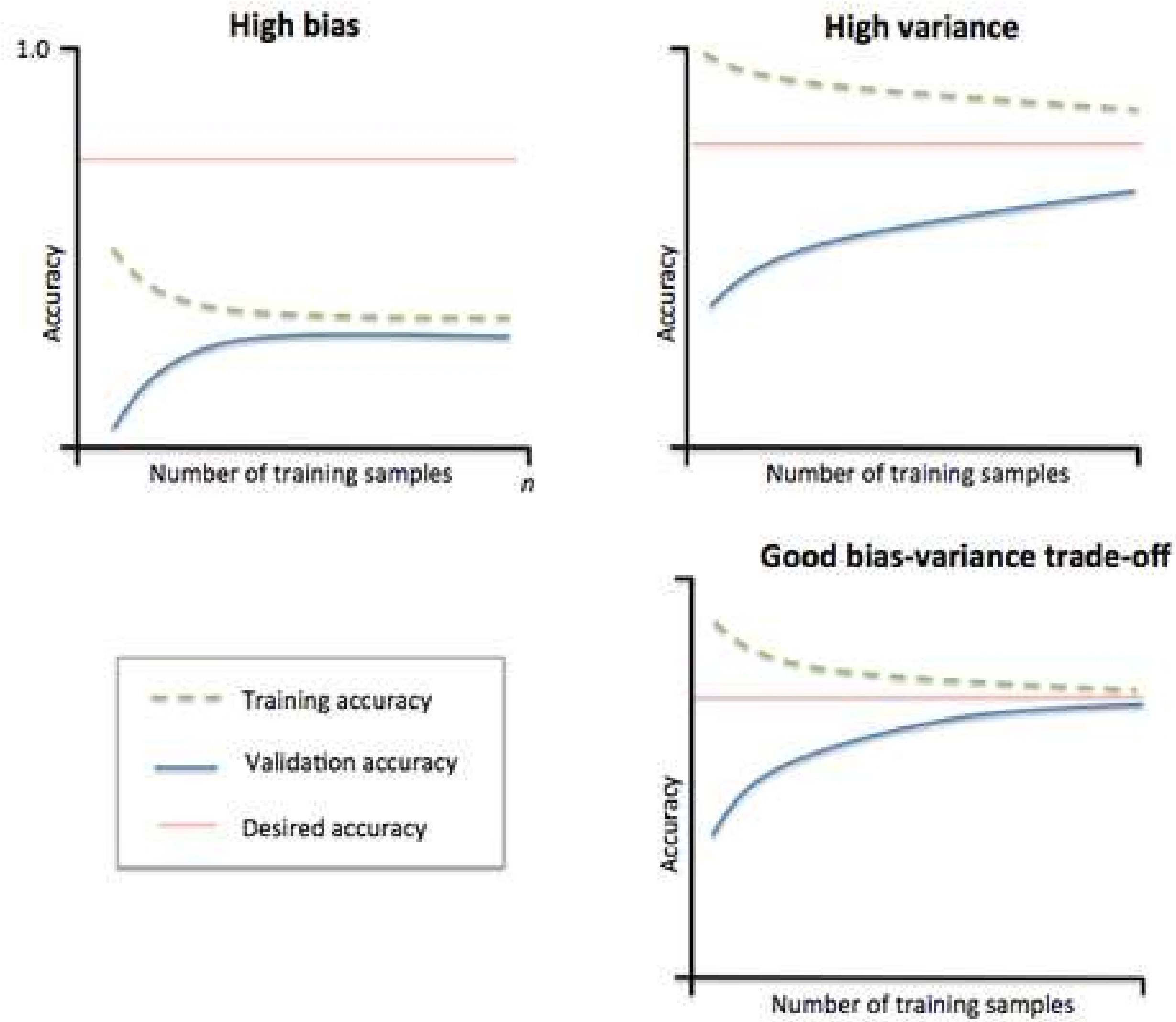
- Validation technique that assesses how the forecasts of a model generalize on unseen data

- A score can be computed on the forecasts across multiple test sets

Cross-validation : Holdout

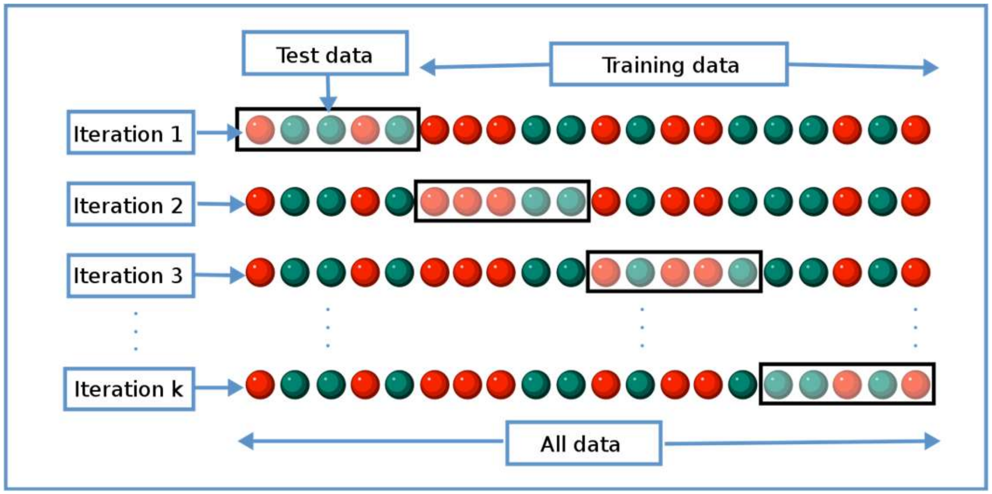
- Separate the data into 2 or 3 sets
  - Training set for training
  - Validation set to find the best parameters
  - (Test set to estimate the performance in case of competing models)
- Separation depends on size of the dataset

Bias-Variance Trade-off



Cross-validation : k-fold

- To avoid “wasting” too much training data
- test on multiple splits so we can get a better idea on how the model will perform on unseen data



- In principle better than using the holdout method because the holdout method score is dependent on how the data is split into train and test sets

With which type of input data shouldn’t it be used ?

Summary of cross-validation techniques

Name	Descrption
Leave-one-out	Use one observation as the test set, and the remaining observations as the train set. Out of N observations, there are N train-test splits.
Leave-p-out	Use p observations as the test set, and the remaining observations as the train set. Out of N observations, there are $\binom{N}{p}$ train-test splits.
Holdout	Randomly assign observations to the train and test sets. Caveat: There is a single run, and the result may not be representative.
K-fold	Split the Nobservations into Kregular folds and apply a leave-one-out CVon the folds. The data may or may not be shuffled prior to forming the folds.
Combinatorial K-fold	Split the N observations into K regular folds and apply a leave-p-out CVon the folds. The data may or may not be shuffled prior to forming the folds.
Monte Carlo	Train sets are generated from random subsampling of the rows of (X,y) (random sampling without replacement)

Data Preparation

Naming conventions

- Data Cleaning : identify and fix errors in the data prior to modeling (outliers, missing values, …)
- Feature Engineering : create new input variables from raw data
- Data Wrangling : more general or colloquial term for data preparation that might include some data cleaning and feature engineering.

Data Assumption

Independent and Identically Distributed (IID)

1) Come from the same distribution

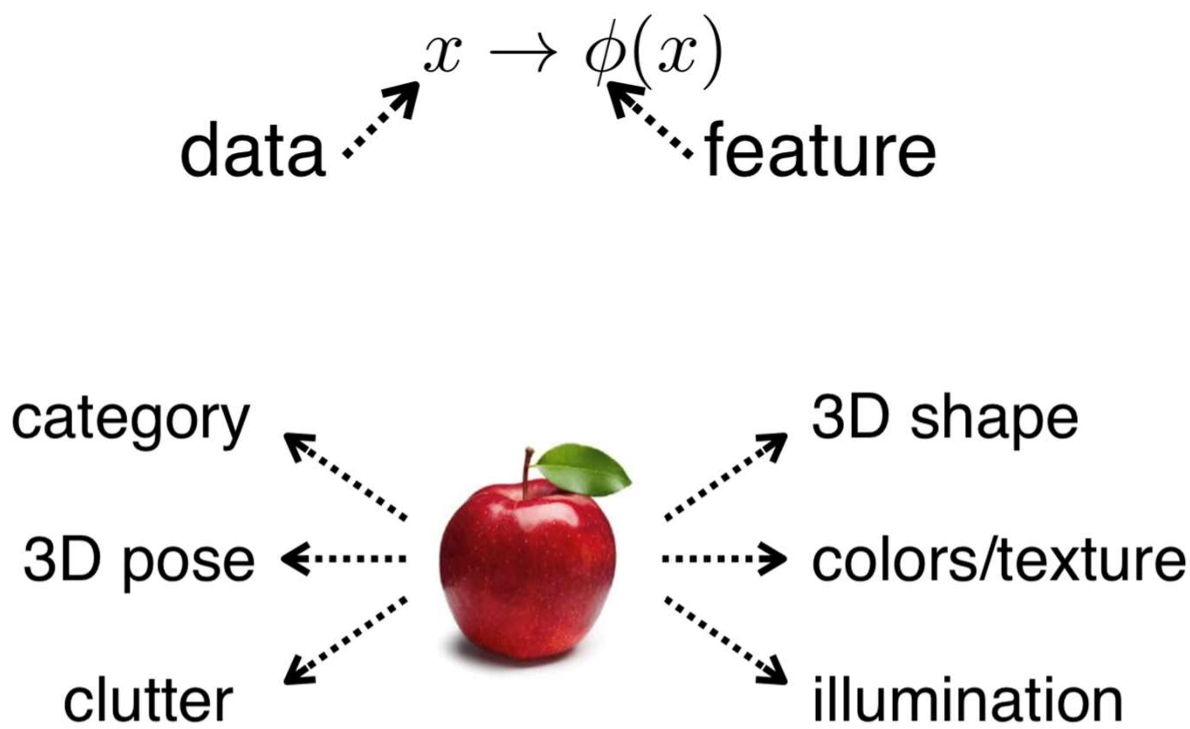
$$p_{x^{(i)}}(x) = p_{x^{(j)}}(x)$$

2) Are *independent*

$$p\left(x^{(1)}, \ldots, x^{(m)}\right) = \prod_{i=1}^{m} p\left(x^{(i)}\right)$$

Data Preparation

Data often encoded into more focused relevant information (features or internal representation) to simplify the decision



Data Pre-Processing

- Convert labels into one-hot encoded vectors to avoid biases

[1 2 3 0 2] → [0100 0010 0001 1000 0010]

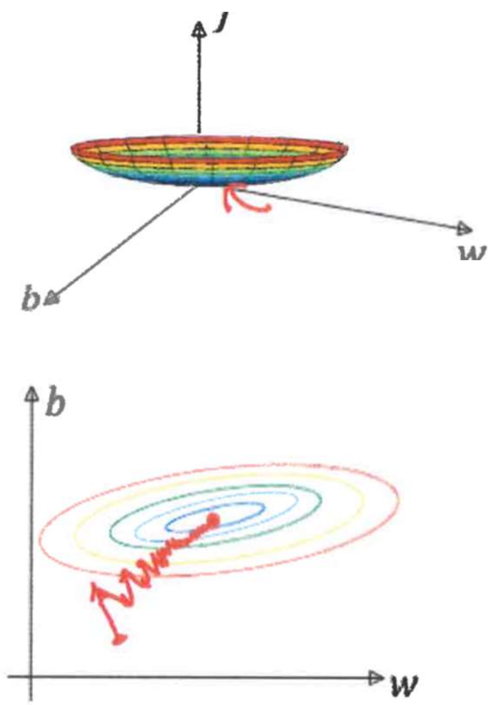
- Shuffling to ensure independent samples and an unbiased estimate

- Remove under-represented categories

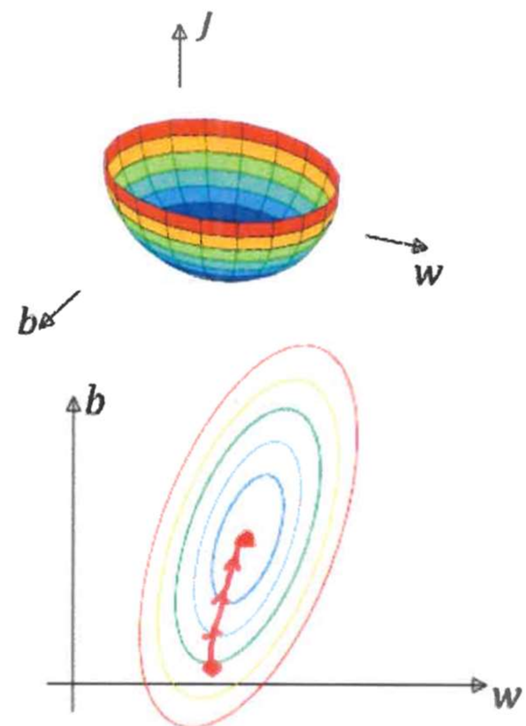
Standardization

- for faster convergence and more reliable predictions

Unnormalized :



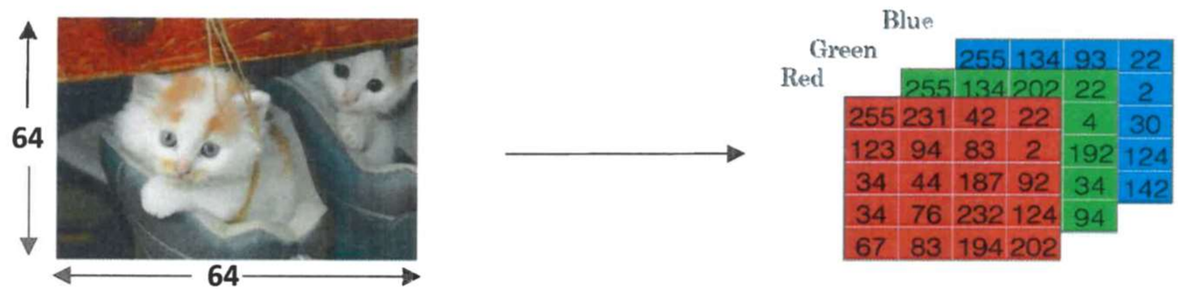
Normalized :



Reshaping into feature vector

Example with binary classification :

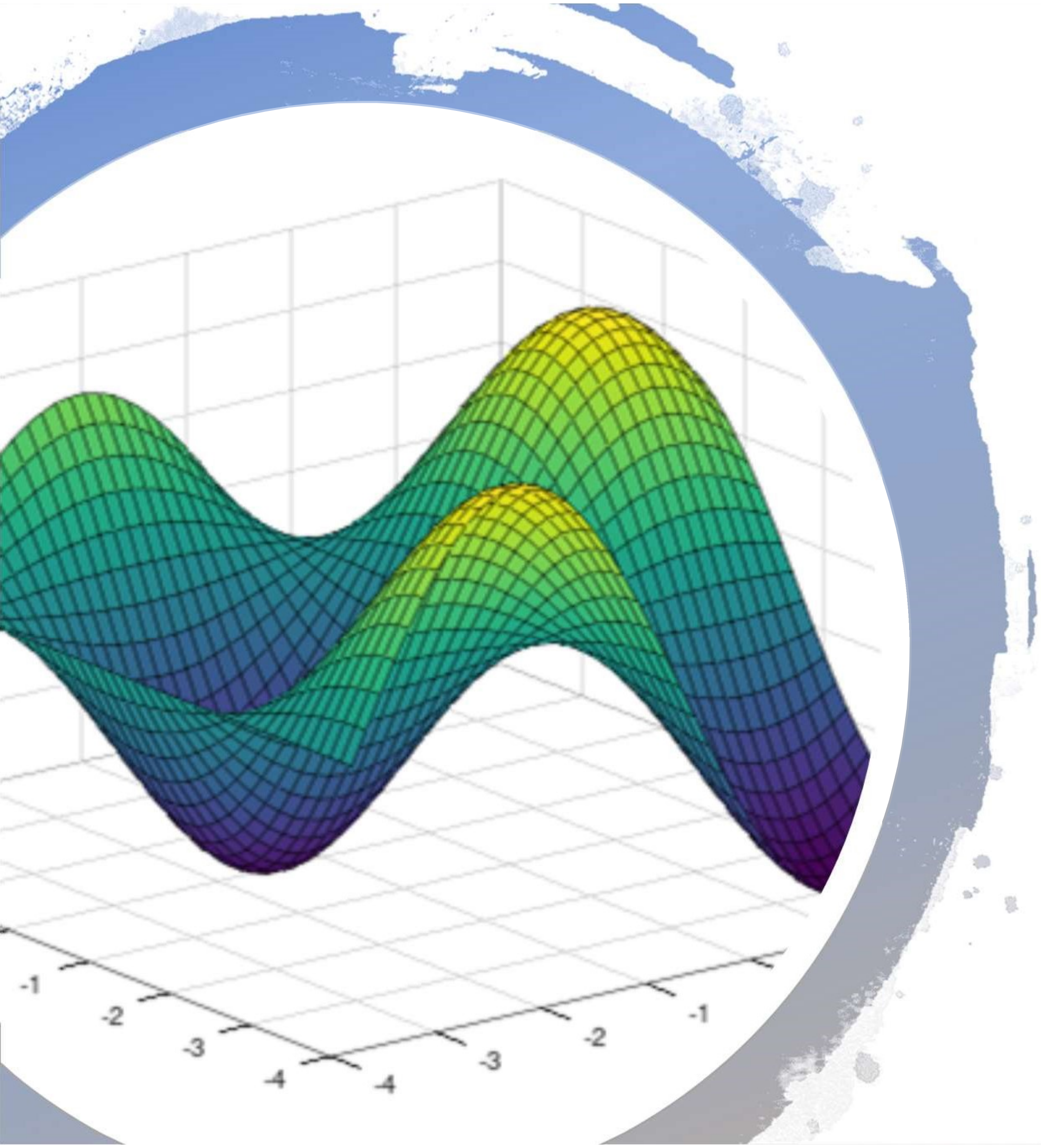
- Image stored in 3 matrices (red, green, blue color channels)
  - Similar size as the image



- Reshape the values into a feature vector x

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{red} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{green} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{blue} \end{matrix}$$

- Normalization : divide by 255 (more convenient for data)



Optimization

Methods to learn the model

1) Simple linear regression
   - use statistics (means, std, correlations and covariance)
   - Need for all the data

2) Ordinary Least Squares
   - Minimize the sum of the squared residuals
   - Treats the data as a matrix and uses linear algebra to estimate the optimal values for the coefficients
   - Need for all the data
   - Need for enough memory

Loss and Cost Functions

- Loss function $L\left(\hat{y}^{(i)}, y^{(i)}\right)$, also called error function, measures how different the prediction $\hat{y} = f(x)$ and the desired output y are

- Cost function $J(w, b)$ is the average of the loss function on the entire training set

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

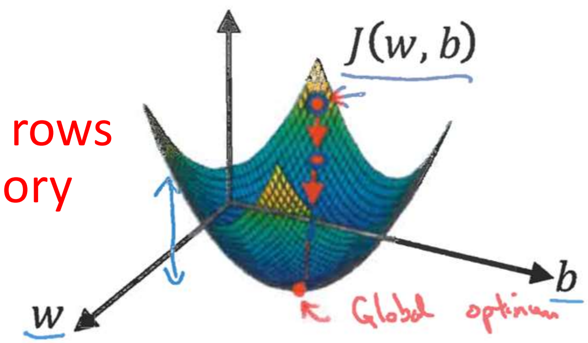
- Goal of the optimization is to find the parameters $\theta = (\mathrm{w}, \mathrm{b})$ that minimize the cost function

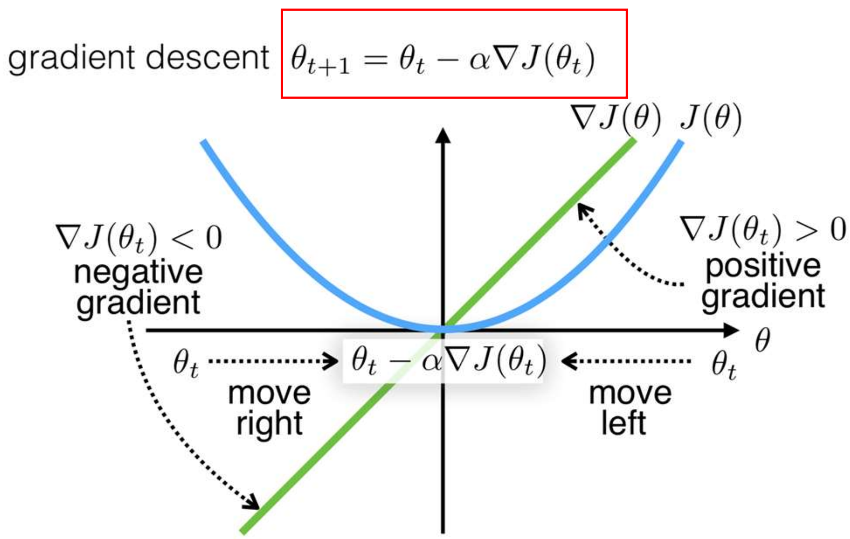
Gradient Descent

- iterative method to find the parameters $\theta = (w, b)$ that minimize $J(\theta)$

- Useful with very large datasets (either rows or columns) that may not fit into memory

- Learning rate α



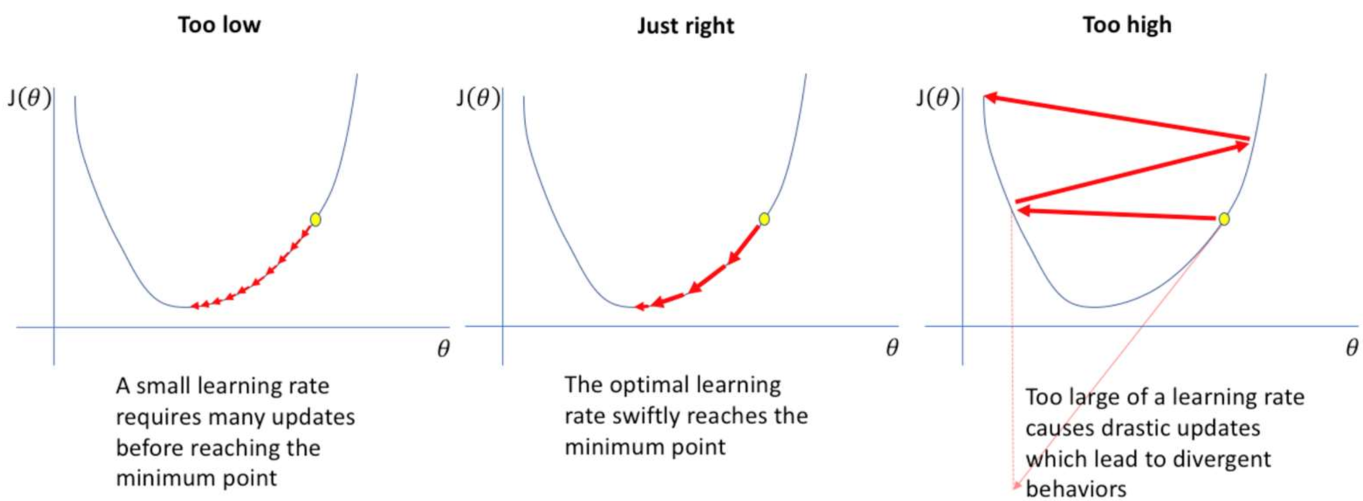


$$\nabla J(w) = \frac{dJ(w,b)}{dw}$$

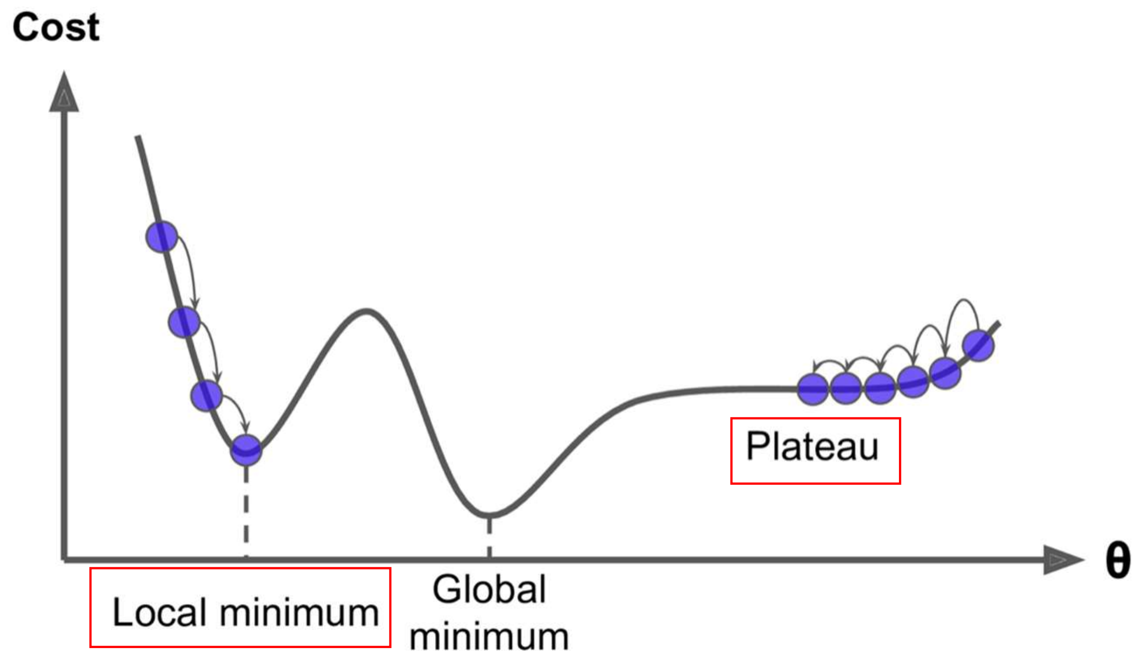
$$\nabla J(b) = \frac{dJ(w,b)}{db}$$

learning rate
α

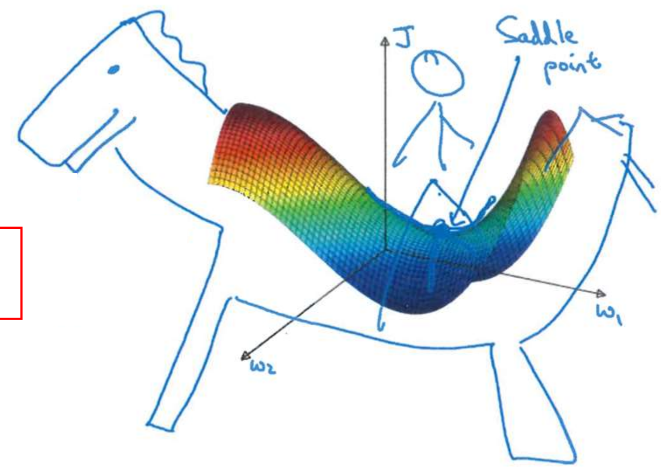
- Has a significant impact on the model performance, while being one of the most difficult parameters to set



Optimization Pitfalls



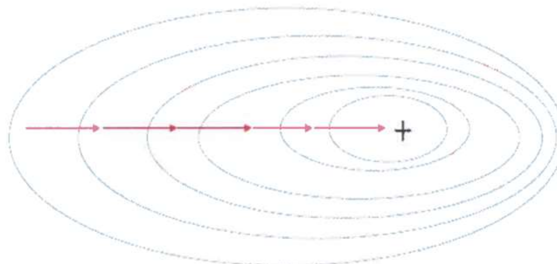
Saddle point



Mini-batch
Gradient Descent
and Stochastic
Gradient Descent

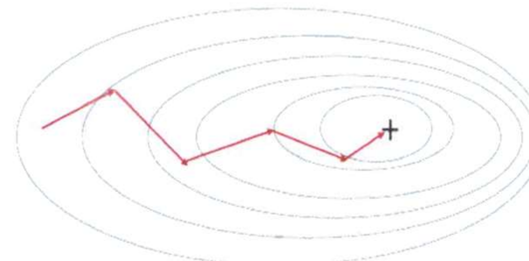
- At each iteration :

Gradient descent (GD)



the *whole* training set

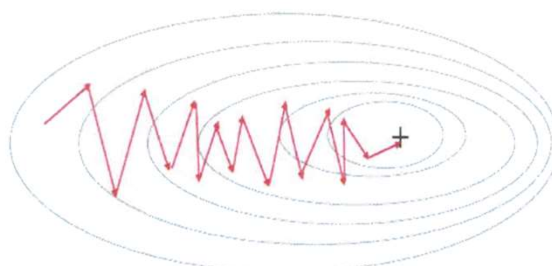
Mini-batch gradient
descent



a *batch* of samples

Batch size choice *typically*
32,64,128,256,512

Stochastic gradient
descent (SGD)

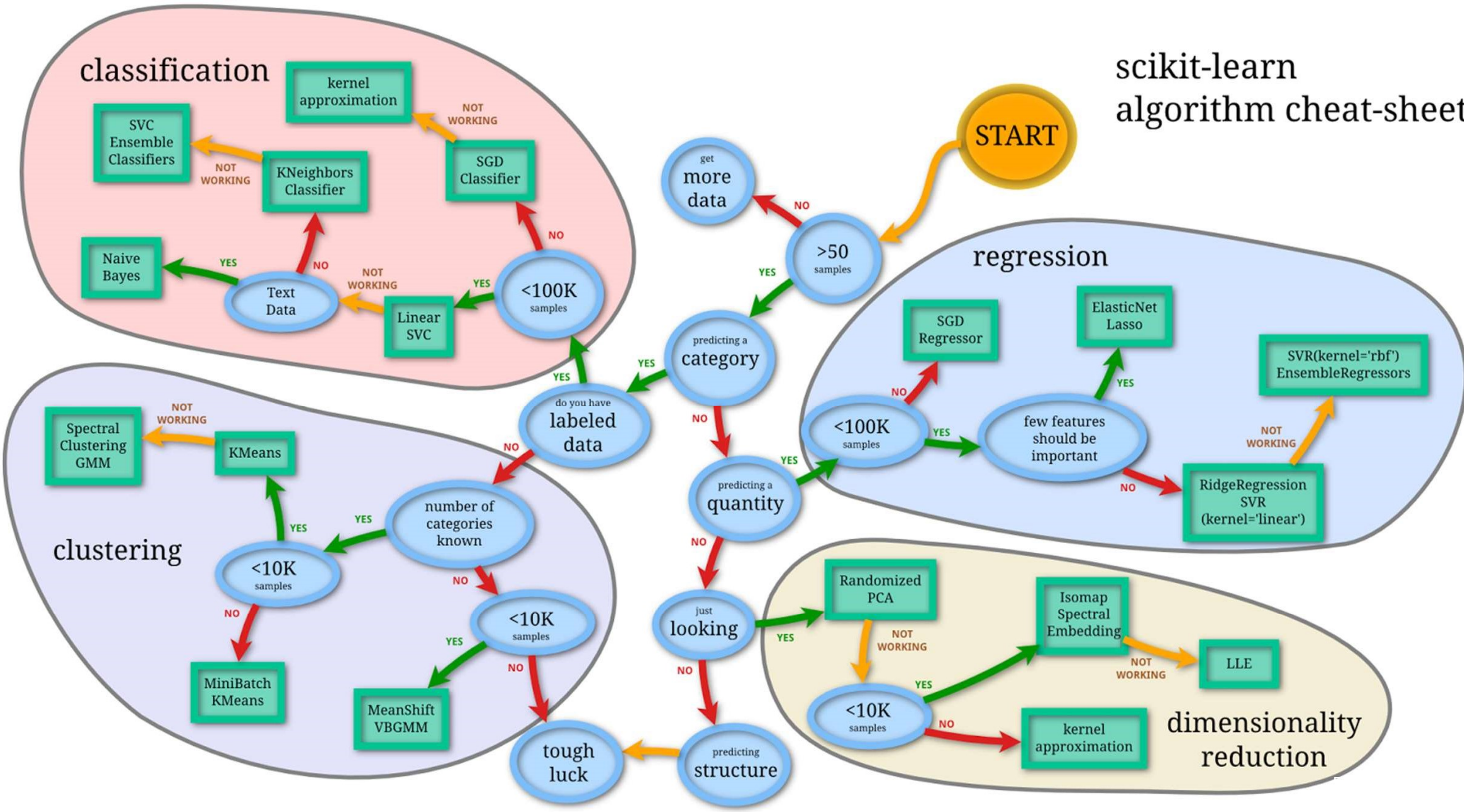


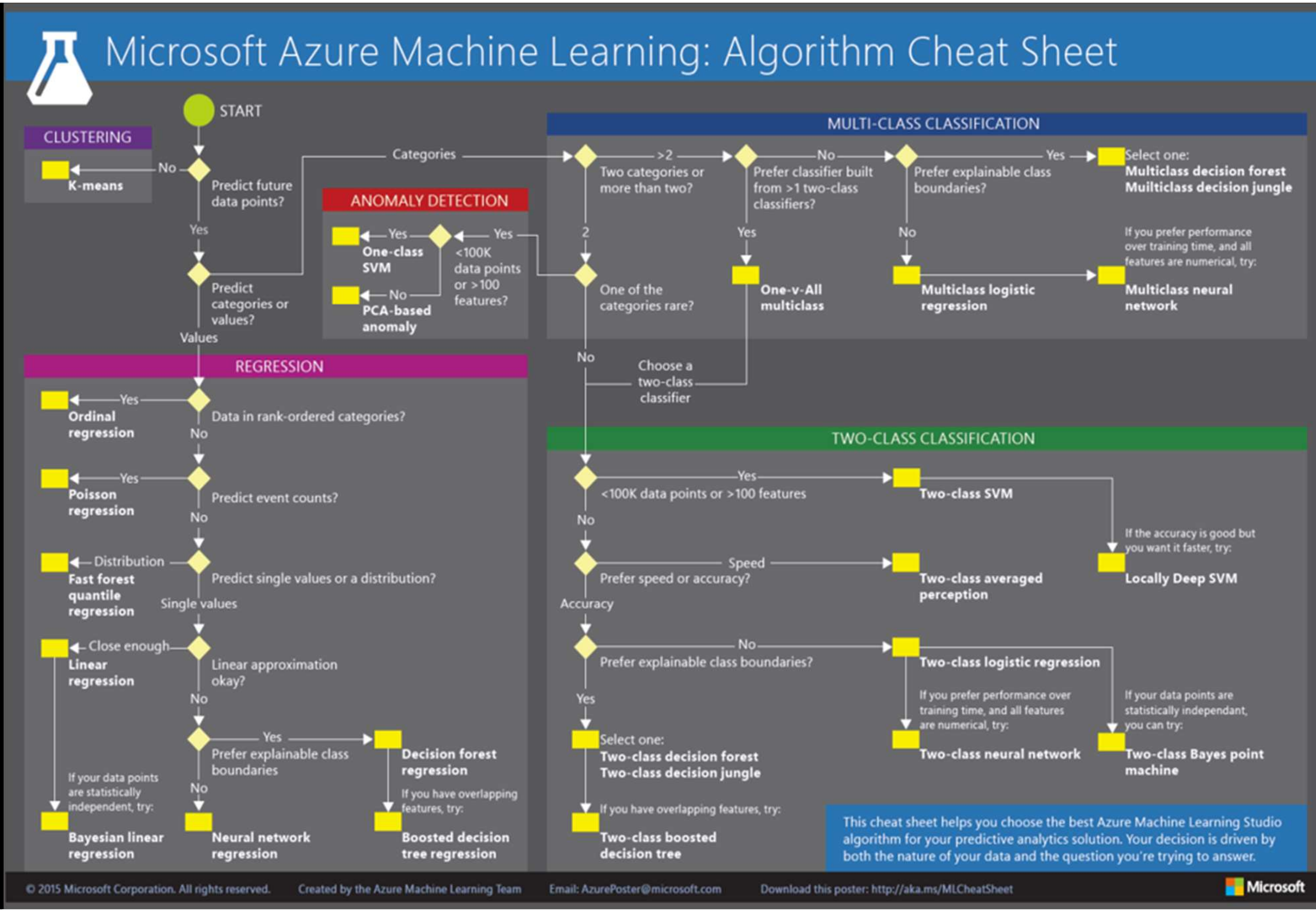
1 sample

How to find its way in the algorithm jungle ?

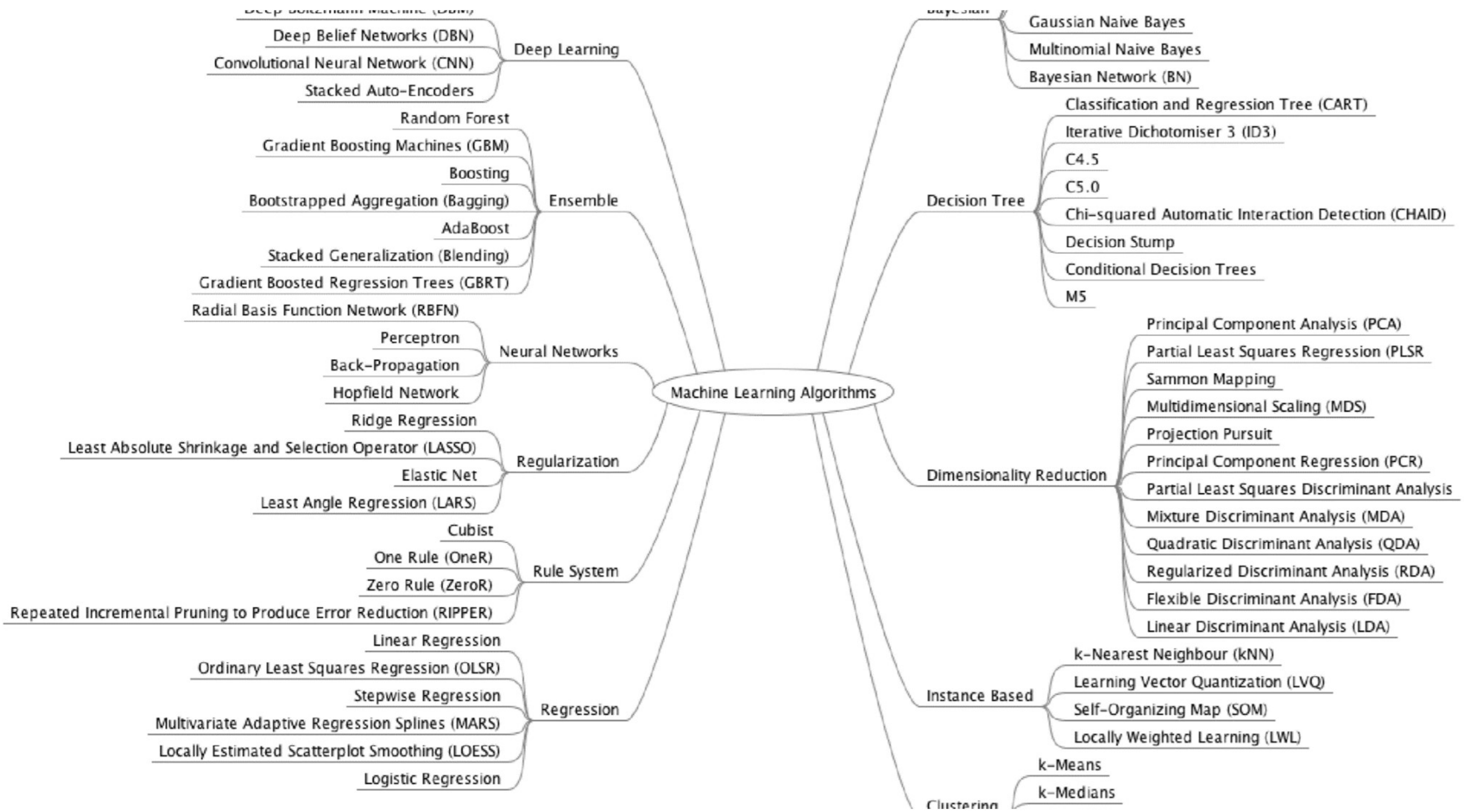
No Free Lunch Theorem

- If you make no assumption about the data, there is no reason to prefer one model over any other
- No model that is a priori guaranteed to work better
- In practice, impossible to test all the models, so make some reasonable assumptions about the data and evaluate only a few models


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC Ensemble Classifiers
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering GMM
MiniBatch KMeans
MeanShift VBGMM
regression
SGD Regressor
few features should be important
ElasticNet Lasso
RidgeRegression SVR (kernel='linear')
SVR(kernel='rbf') EnsembleRegressors
dimensionality reduction
Randomized PCA
Isomap Spectral Embedding
LLE
YES
NO
NOT WORKING



Microsoft Azure Machine Learning: Algorithm Cheat Sheet
START
CLUSTERING
K-means
No
Predict future data points?
Yes
Categories
Predict categories or values?
Values
ANOMALY DETECTION
Yes
One-class SVM
<100K data points or >100 features?
No
PCA-based anomaly
MULTI-CLASS CLASSIFICATION
>2
Two categories or more than two?
2
One of the categories rare?
No
Prefer classifier built from >1 two-class classifiers?
Yes
One-v-All multiclass
Prefer explainable class boundaries?
Select one:
Multiclass decision forest
Muilticlass decision jungle
Multiclass logistic regression
If you prefer performance over training time, and all features are numerical, try:
Multiclass neural network
Choose a two-class classifier
REGRESSION
Data in rank-ordered categories?
Ordinal regression
Predict event counts?
Poisson regression
Predict single values or a distribution?
Distribution
Fast forest quantile regression
Single values
Linear approximation okay?
Close enough
Linear regression
If your data points are statistically independent, try:
Bayesian linear regression
Prefer explainable class boundaries
Decision forest regression
If you have overlapping features, try:
Boosted decision tree regression
Neural network regression
TWO-CLASS CLASSIFICATION
<100K data points or >100 features
Two-class SVM
If the accuracy is good but you want it faster, try:
Locally Deep SVM
Prefer speed or accuracy?
Speed
Two-class averaged perception
Accuracy
Prefer explainable class boundaries?
Two-class logistic regression
If you prefer performance over training time, and all features are numerical, try:
Two-class neural network
If your data points are statistically independant, you can try:
Two-class Bayes point machine
Select one:
Two-class decision forest
Two-class decision jungle
If you have overlapping features, try:
Two-class boosted decision tree
This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.
© 2015 Microsoft Corporation. All rights reserved.
Created by the Azure Machine Learning Team
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLCheatSheet
Microsoft


Machine Learning Algorithms
Deep Learning
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
k-Means
k-Medians

The Lesson

The lesson is always to fit a simple model first, and then only adopt a more complicated ML model if the extra predictive accuracy (value) it provides is worth it.

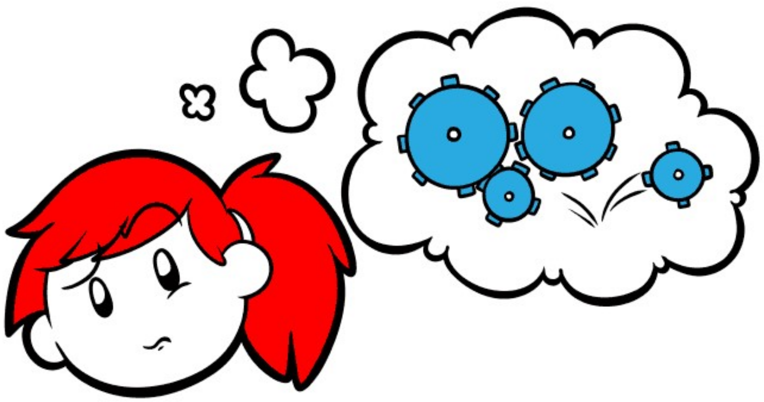
Exercise

- Create your own summary (graph) for :
  - Regression
  - Classification

Quiz

https://b.socrative.com/login/student/

Room : CONTI6128



Technical Details

Tools

data structure & analysis

main package for scientific computing in Python

Python-based ecosystem of open-source software for mathematics, science, and engineering.

famous library to plot graphs in Python

provides simple and efficient tools for data mining and data analysis

interactive coding environments embedded in a webpage

h5py : common package to interact with a dataset that is stored on an H5 file

Real Data

- http://archive.ics.uci.edu/lm/
- https://www.Kaggle.com/datasets
- http://aws.amazon.com/fr/dataset/

Models

- https://github.com/tensorflow/models

Notebooks (1)

Use Anaconda Navigator

New environment with tensorflow v.2.1.0:

conda create -n tf tensorflow

conda activate tf

Add pip v.20.1.1 :

conda install pip

Notebooks (2)

Add libraries :

conda install matplotlib

pip install sklearn

pip install seaborn

pip install ipywidgets

jupyter nbextension enable --py widgetsnbextension

Notebooks (3)

- **Define git command :**
- install git : https://git-scm.com/download/win (64-bit Git for Windows setup)
- Modifying PATH on Windows 10:
- In the Start Menu or taskbar search, search for "environment variable".
- Select "Edit the system environment variables".
- Click the "Environment Variables" button at the bottom.
- Double-click the "Path" entry under "System variables".
- With the "New" button in the PATH editor, add C:\Program Files\Git\bin\ and C:\Program Files\Git\cmd\ to the end of the list.
- Close and re-open your console.