Module 1 :

Machine Learning Review

Machine Learning Introduction

Géraldine Schaller-Conti, August 2024

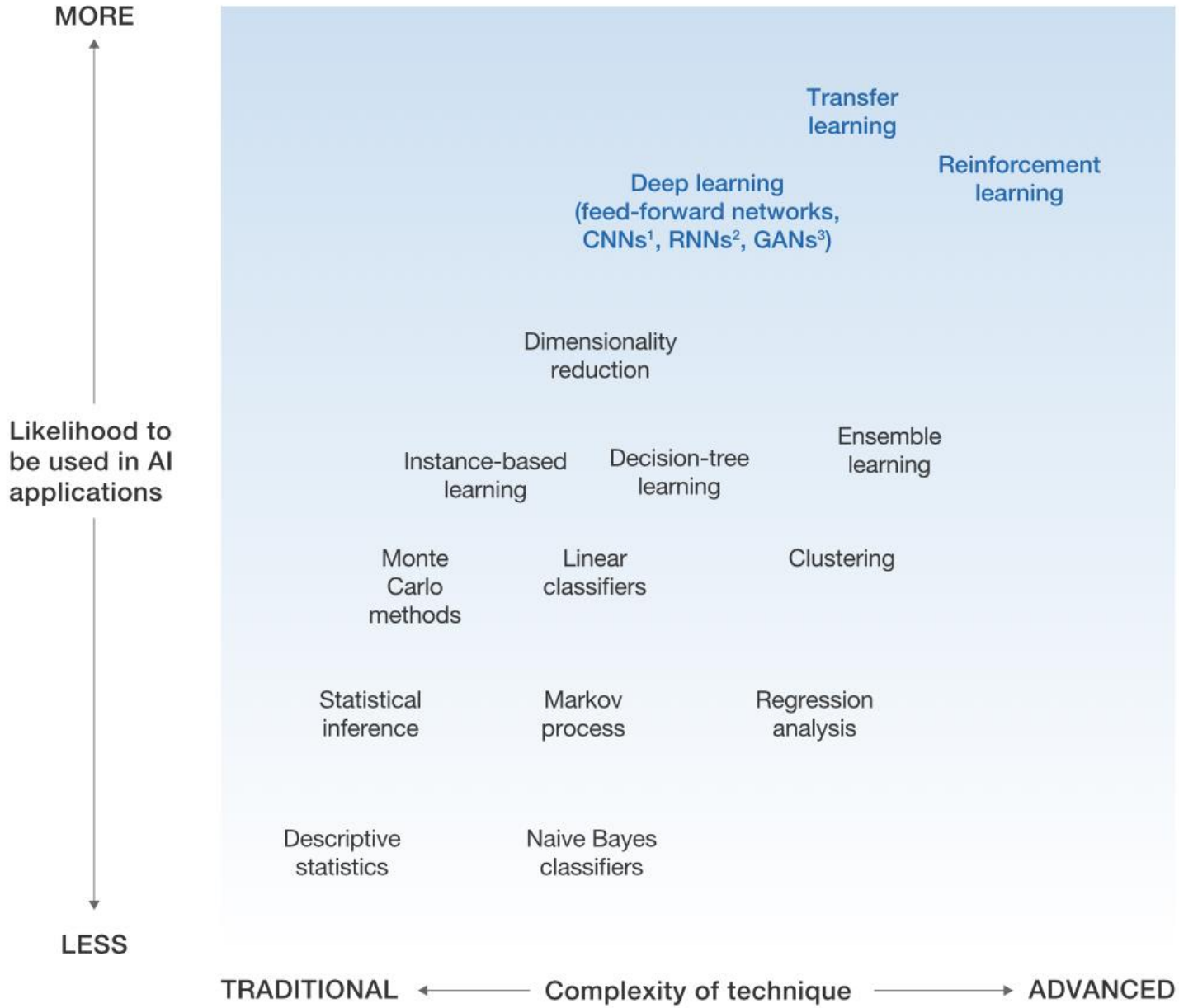
Bibliography

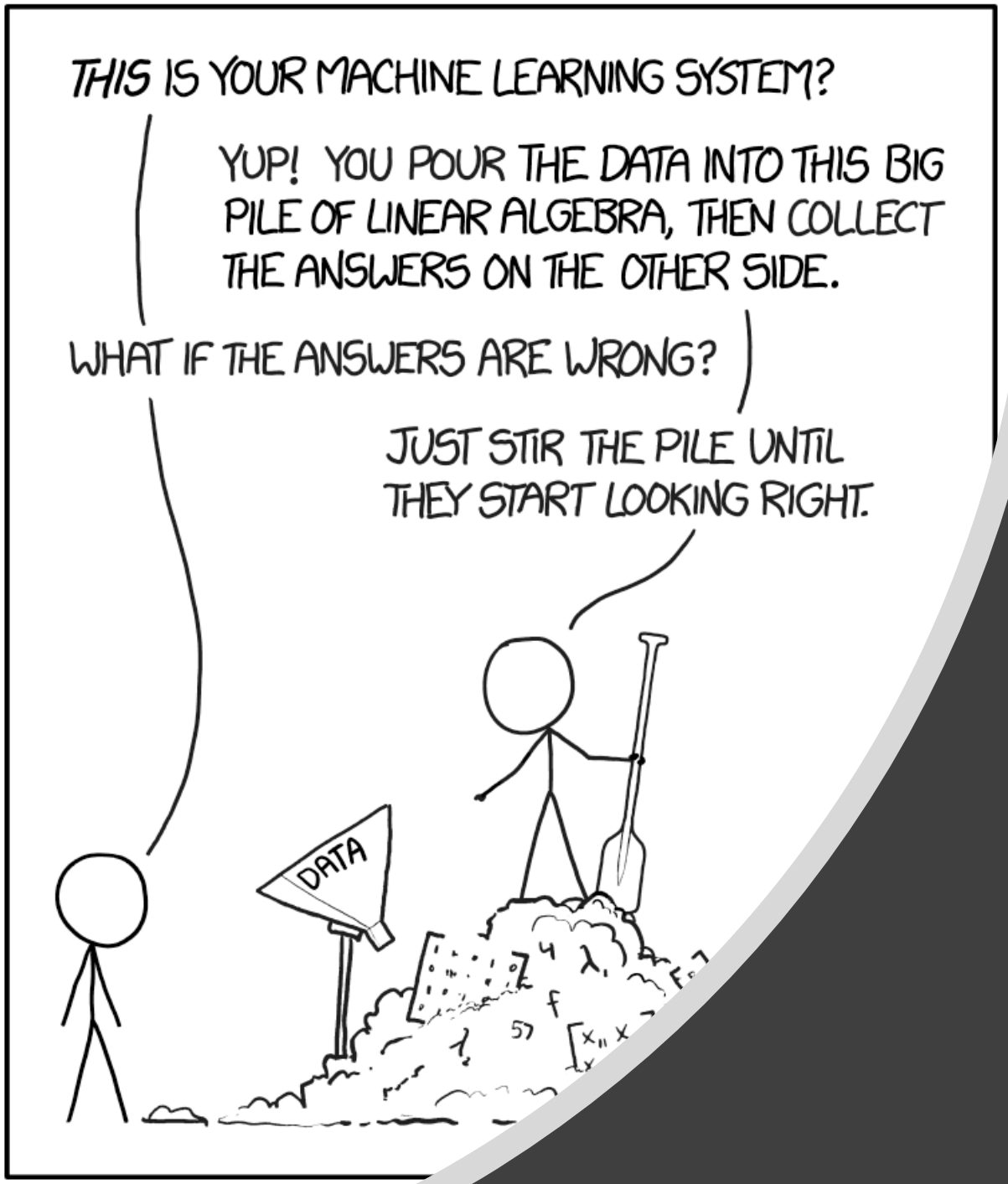
- Deep Learning book (Goodfellow, Bengio, Courville)
- Machine Learning @ Stanford (Prof Andrew Ng)
- Hands-On Machine Learning with Scikit-Learn & Tensorflow (Aurélien Géron)

Learning Objectives

- What is Machine Learning ?
- Learning pillars
- Main Challenges
- Data Preparation
- Optimization
- Technical Details
- How to choose an algorithm ?

Differences between ML and Statistical Modeling

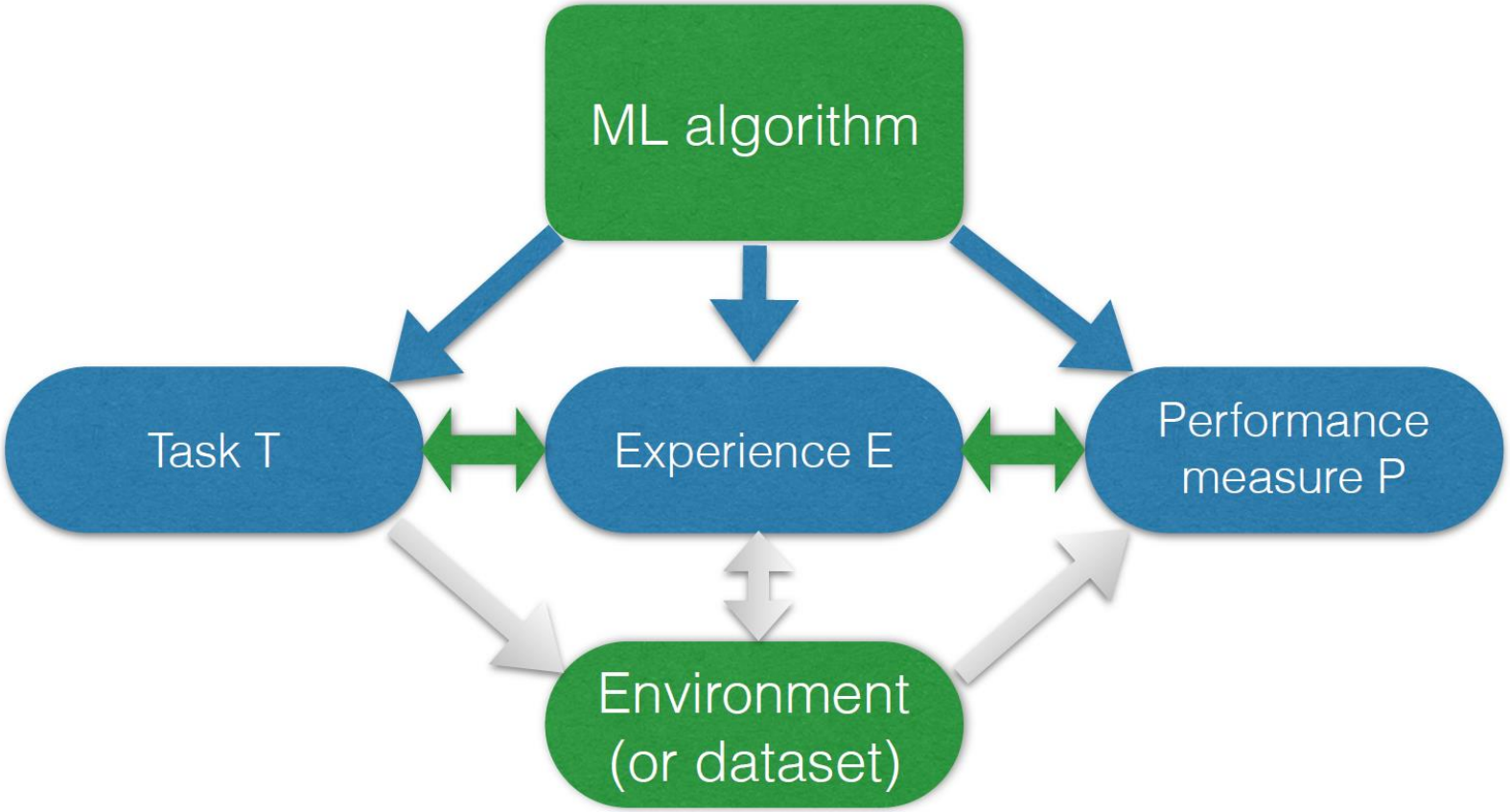




What is Machine Learning ?

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

Definition



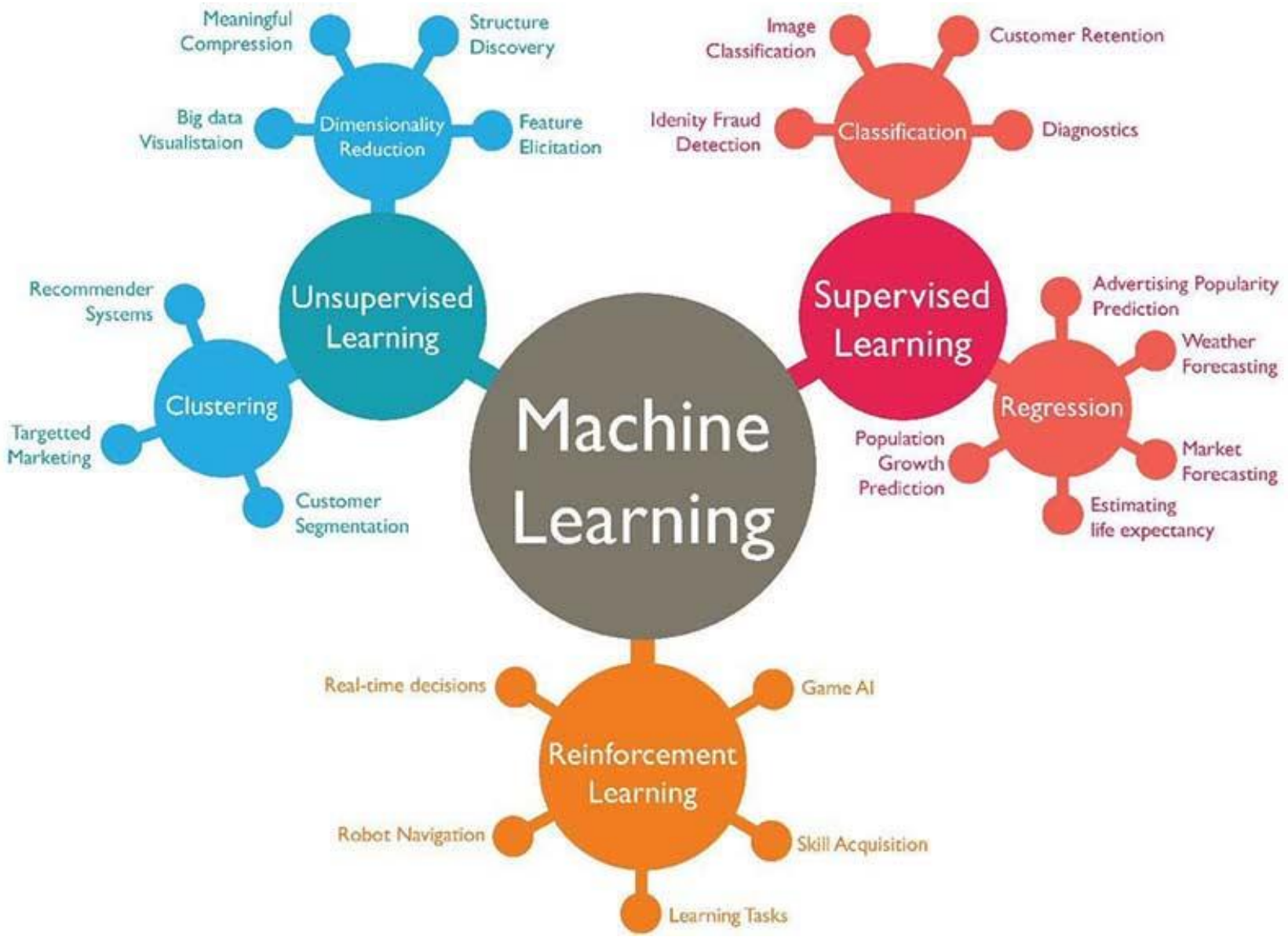
« A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. »

Tom M. Mitchell (1997)

Machine Learning Use Cases

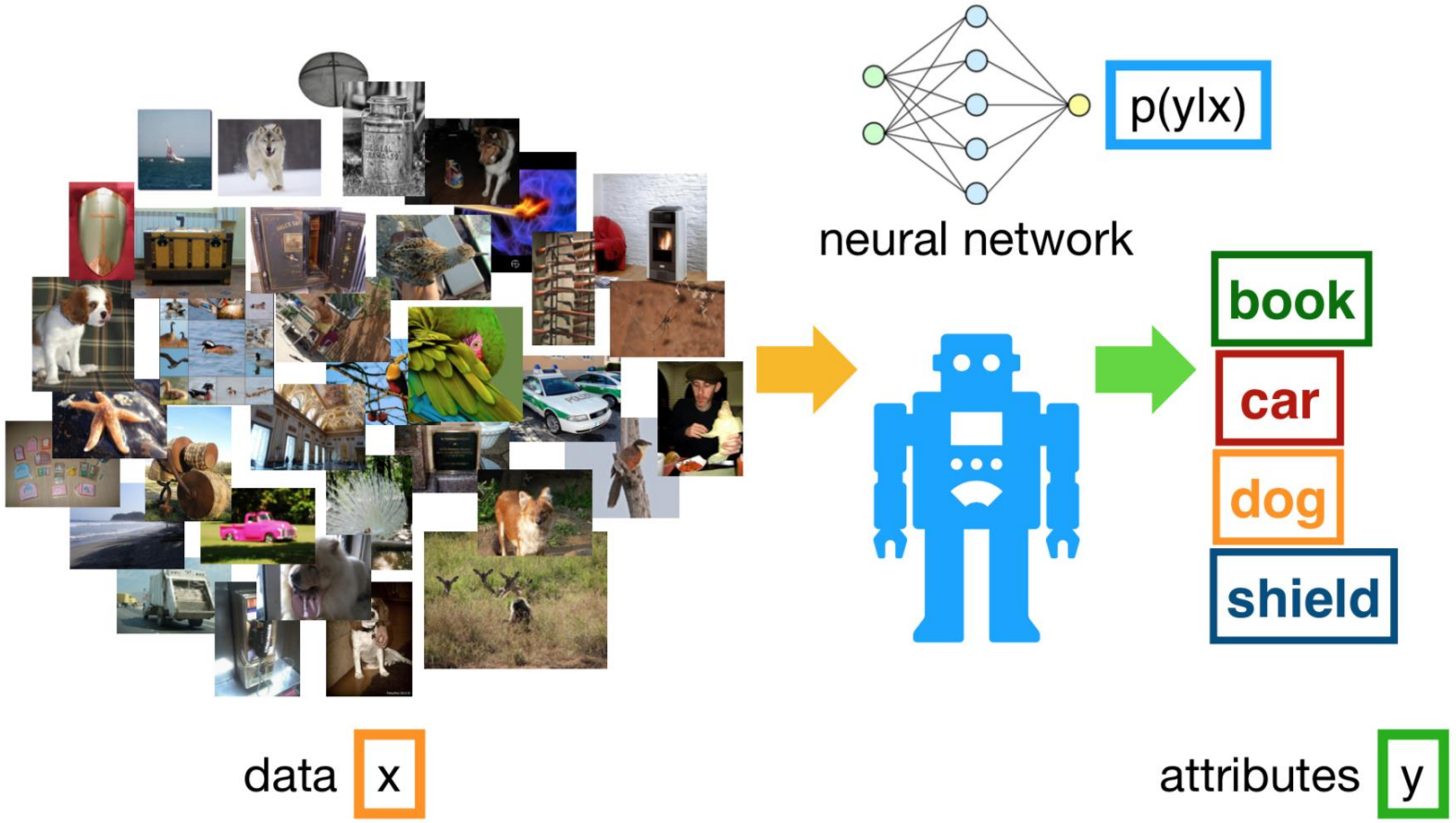
- Problems for which existing solutions require a lot of hand-tuning of long lists of rules
  - ML *simplifies* code and *performs better*
- Complex problems for which there is no good solution at all using a traditional approach
  - ML can *find a solution*
- Fluctuating environments
  - ML system can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

Learning Pillars



Supervised Learning

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



Regression

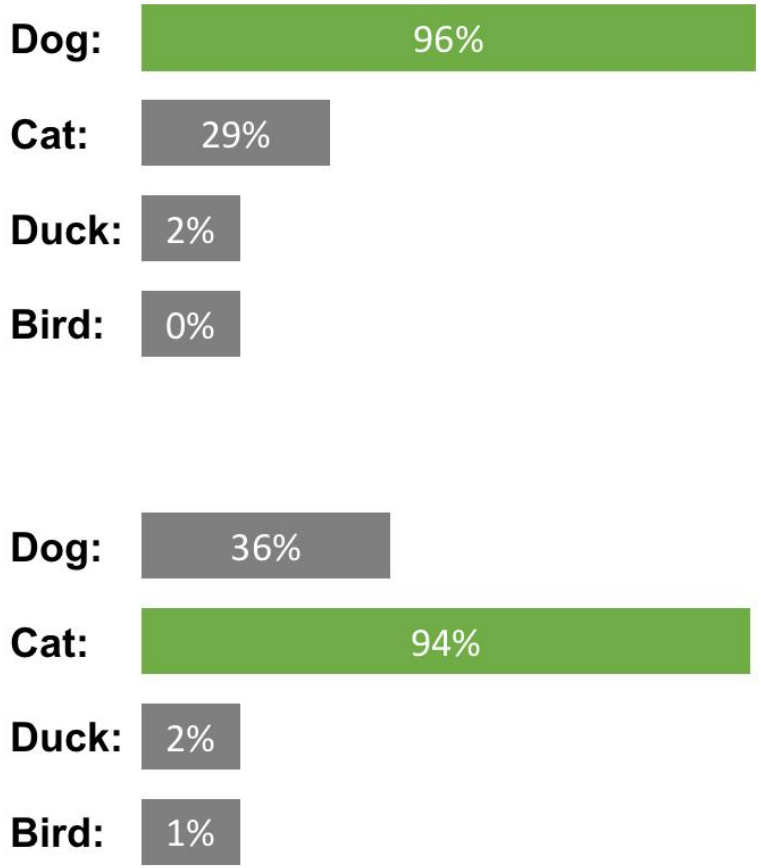
$82000

$55500

???

Predict results within *a continuous output*

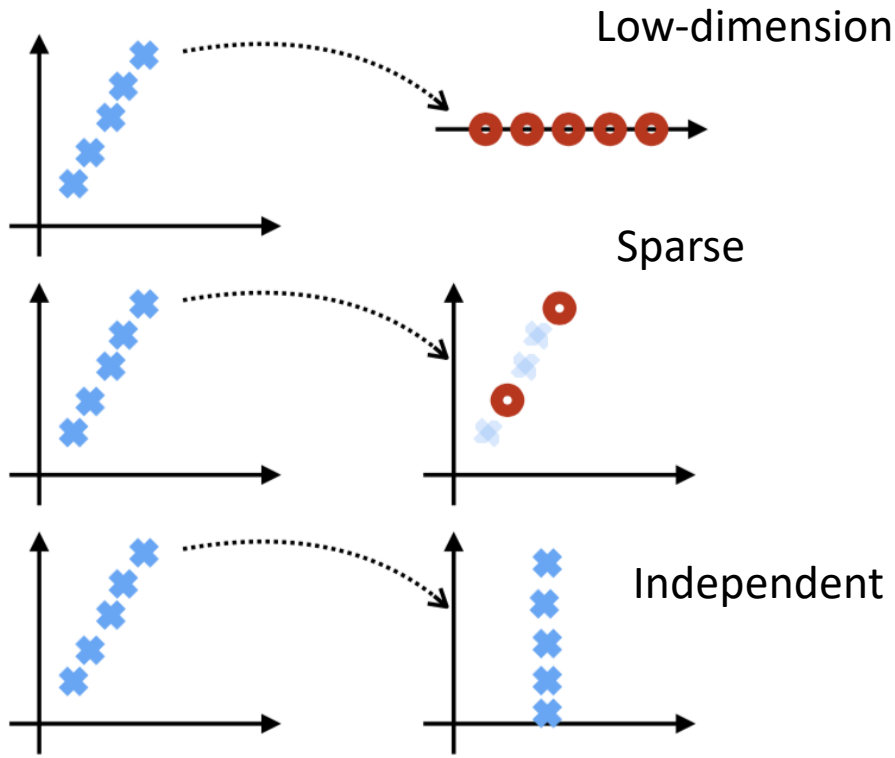
Classification

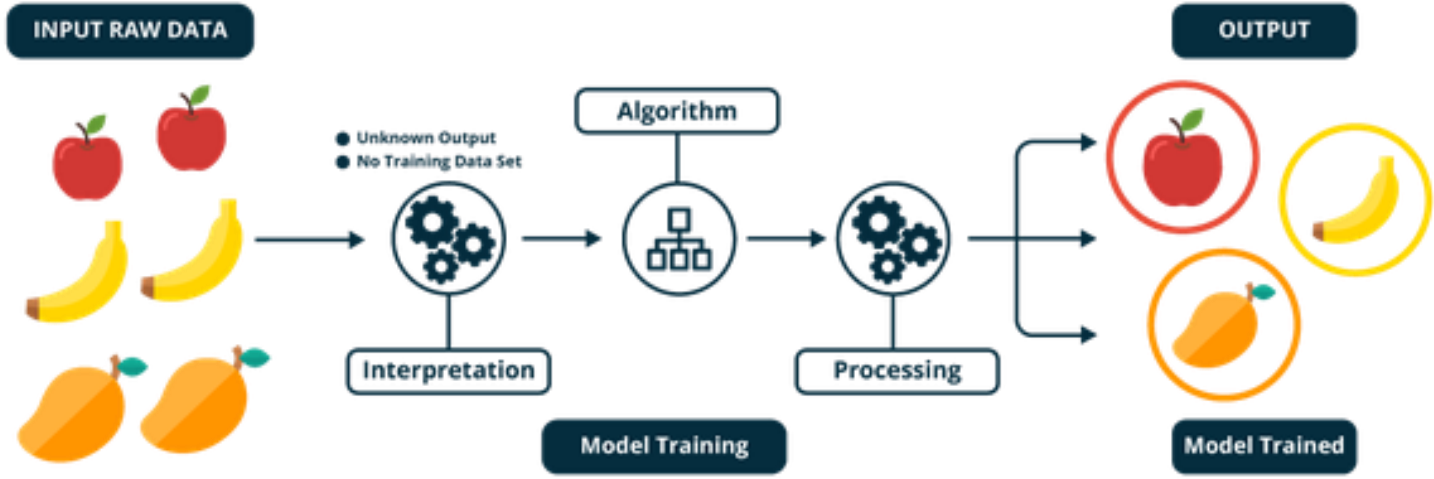


categorize new inputs as belonging to one of a set of categories
→ Predict results within *a discrete output (categories)*

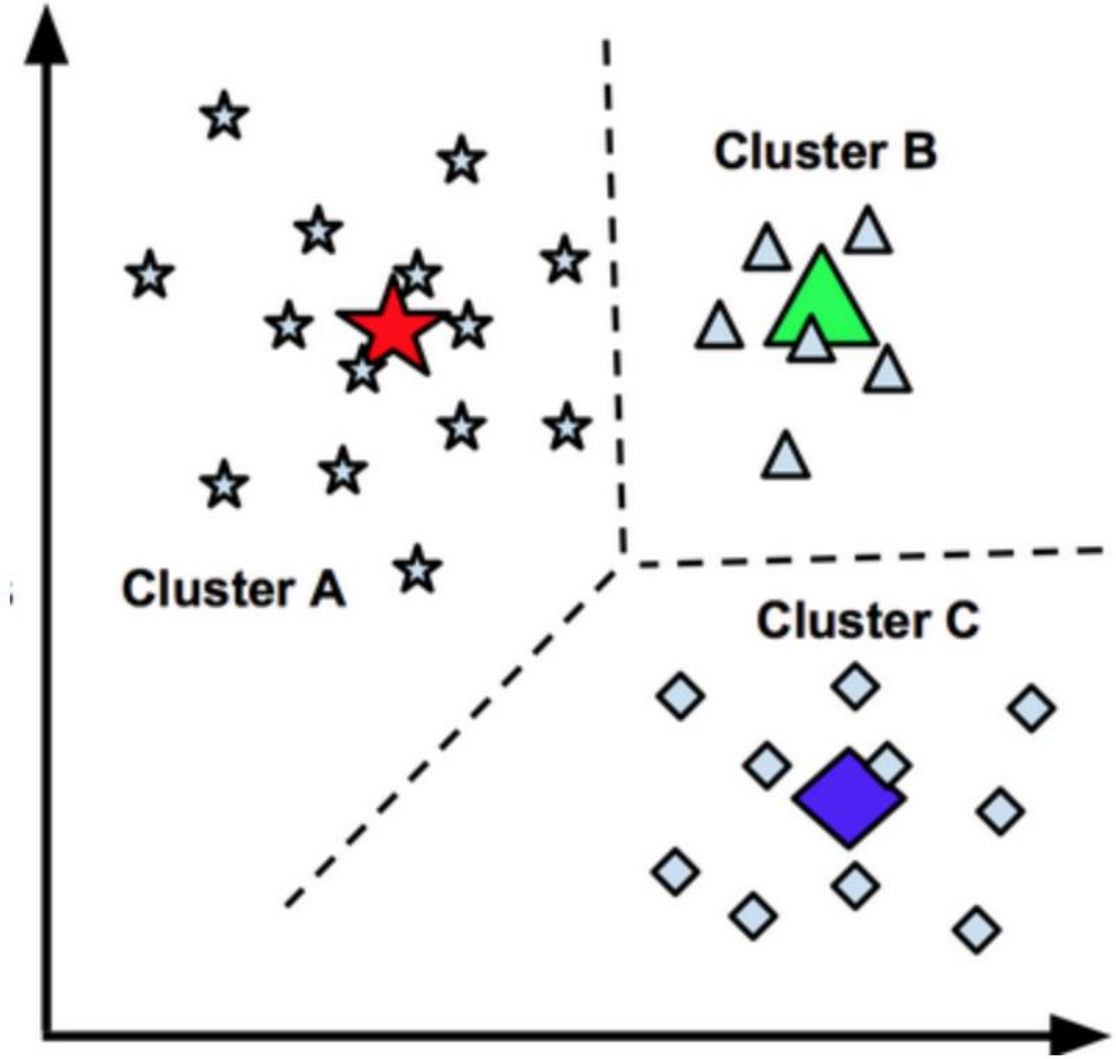
Unsupervised Learning

- 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

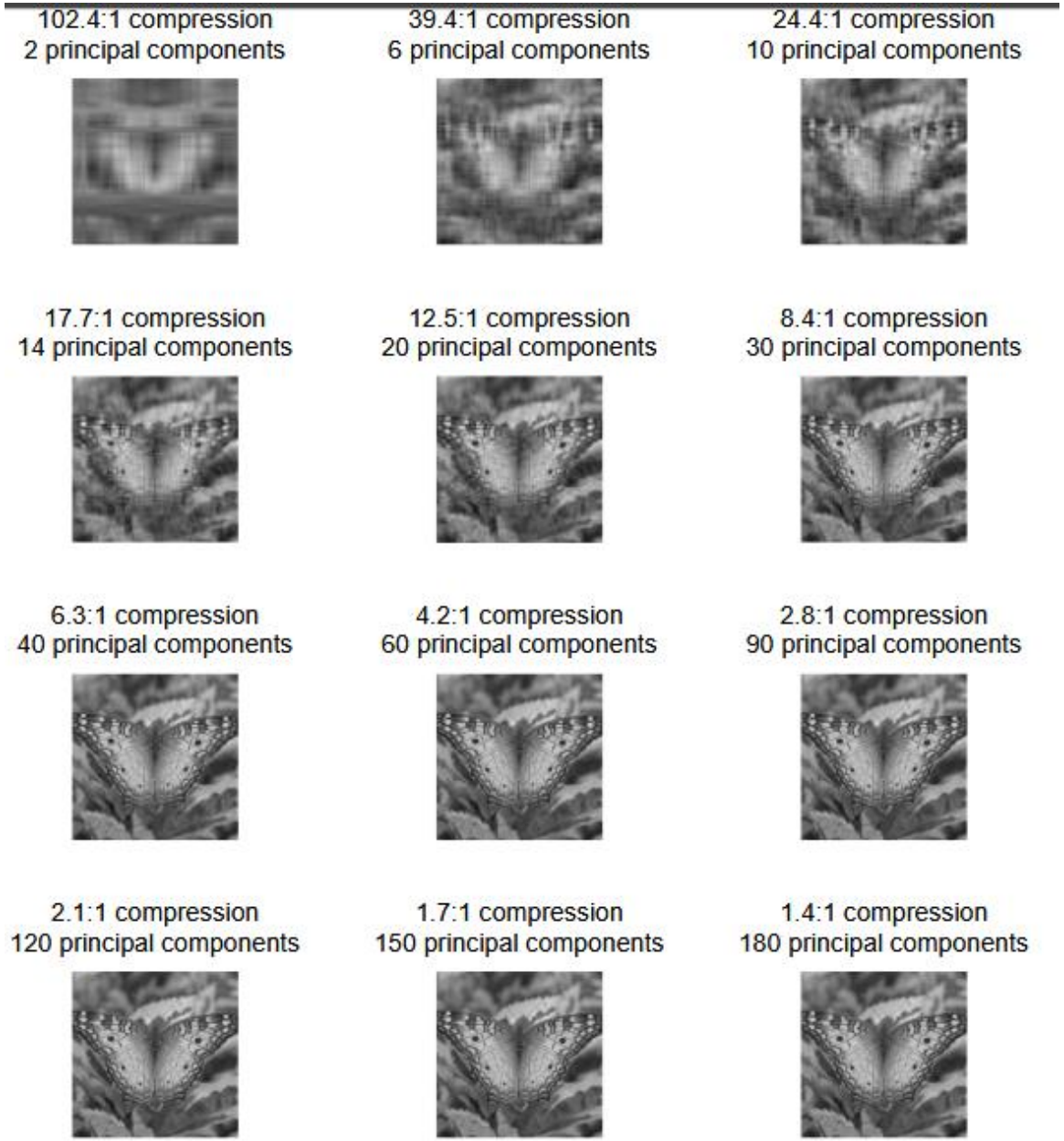


Figure 10: The visual effect of retaining principal components

Mark Richardson (2009)

More nuances…

Supervised Classification

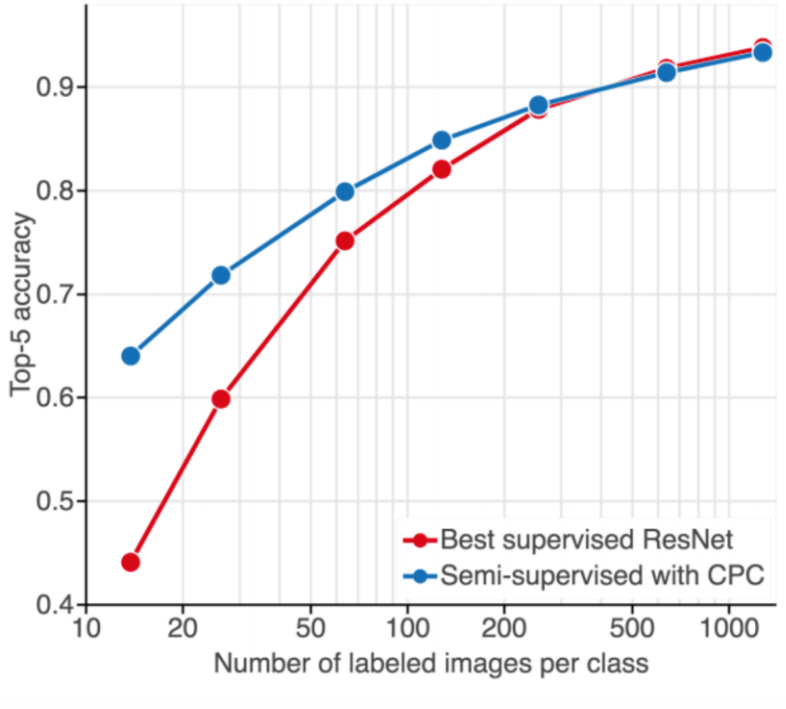
Semi-supervised Learning

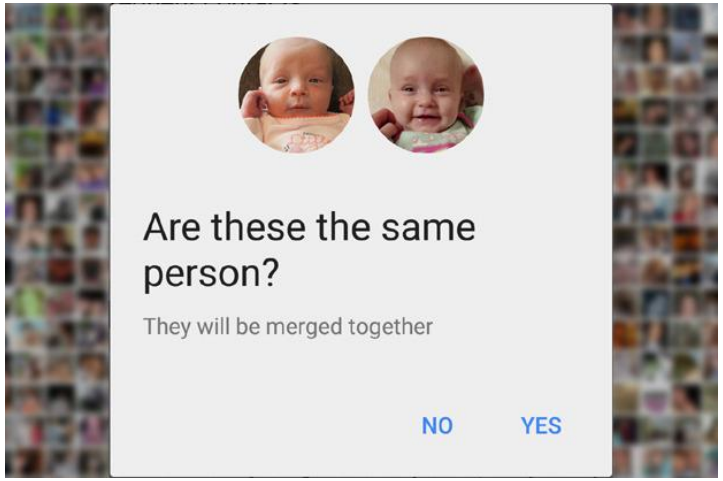
Transfer Learning

Self-taught Learning

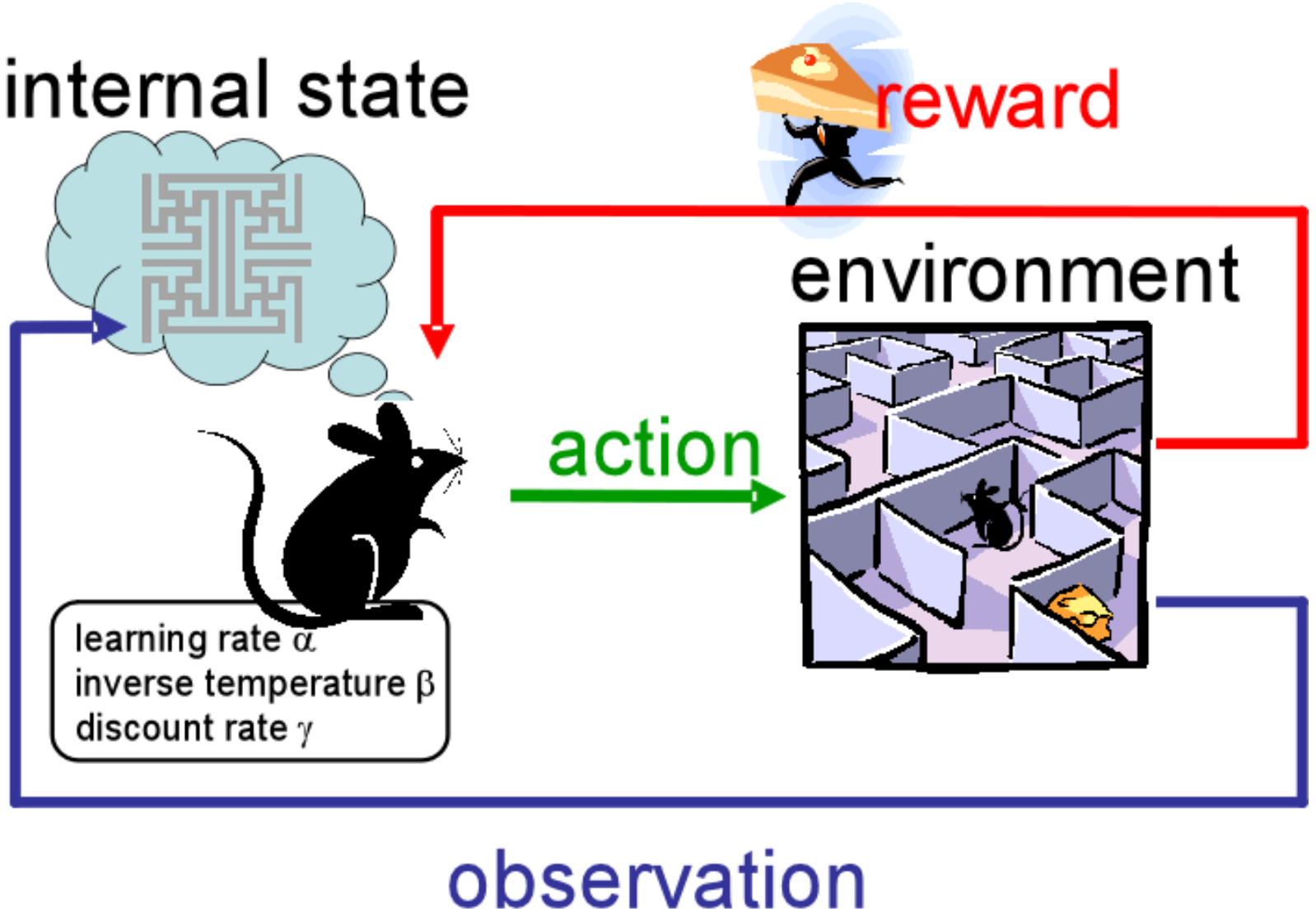
Semi-supervised Learning

- 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



https://www.youtube.com/watch?v=Bui3DWs02h4

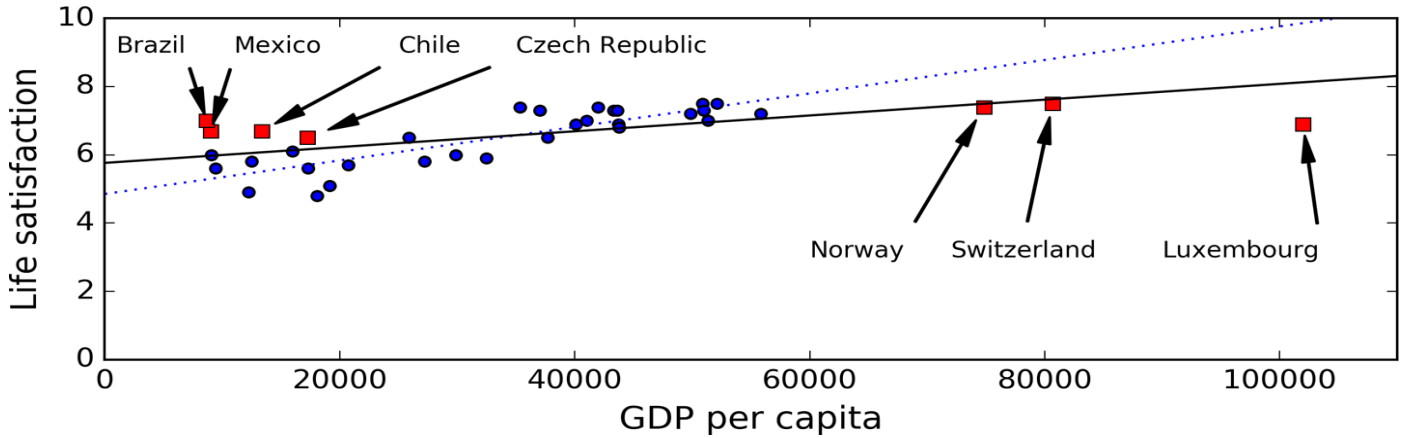
Two-Minute Papers

Main Challenges

- Bad data
- Bad model

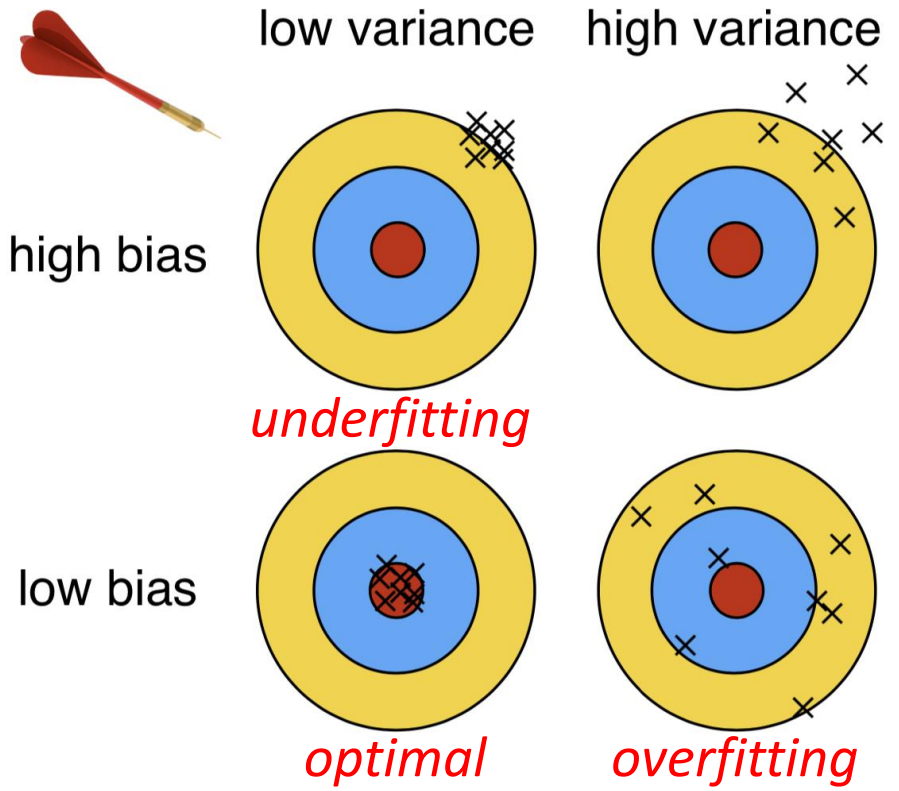
Bad Data

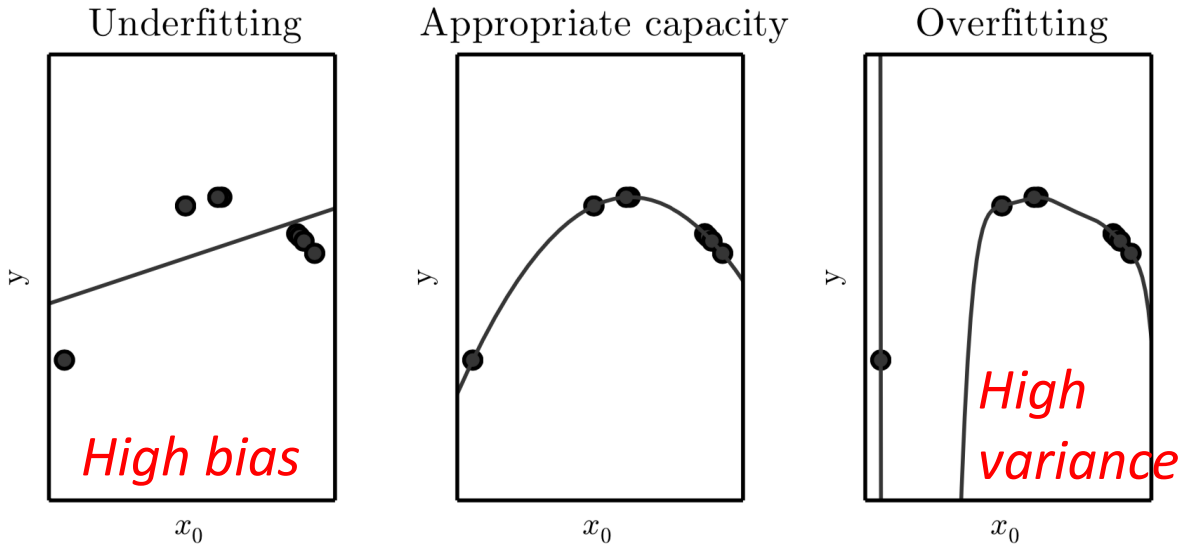
- Insufficient training data
  - Try to get more data, data augmentation,…
- Non-representative training data



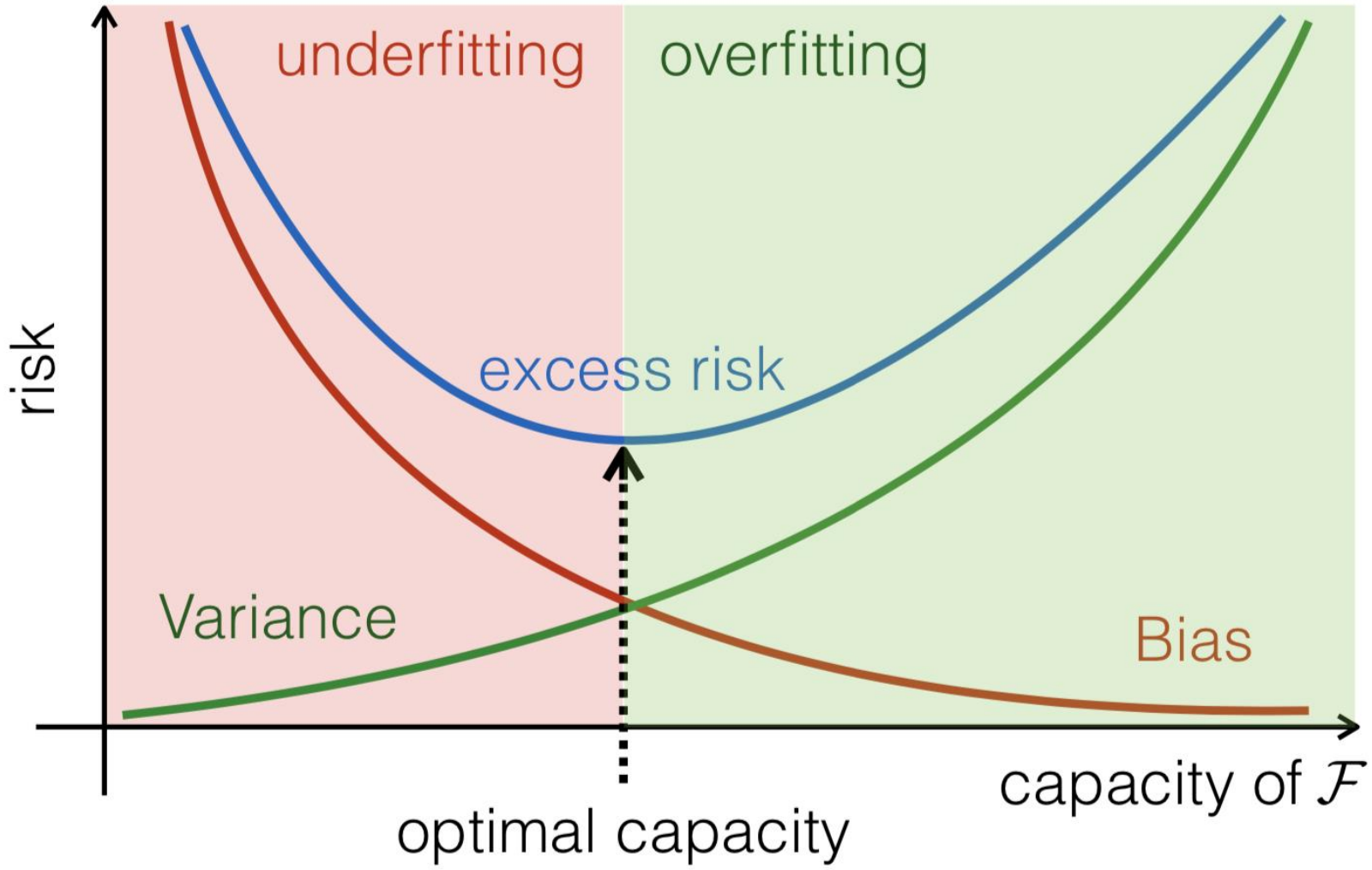
- Poor quality data
  - Data cleaning
- Irrelevant Features
  - Feature selection (select most useful features)
  - Feature extraction (combine features to produce a more useful one)

Bad 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

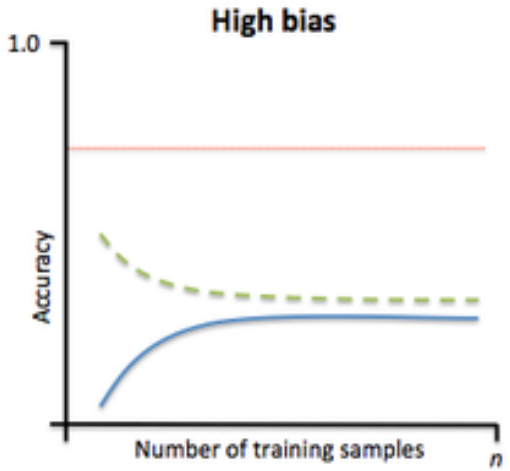
- Separate the data into 3 sets
  - Training set for training
  - Validation/dev set to fine-tune the model hyperparameters
  - Test set to estimate the FINAL model performance
- Separation depends on size of the dataset

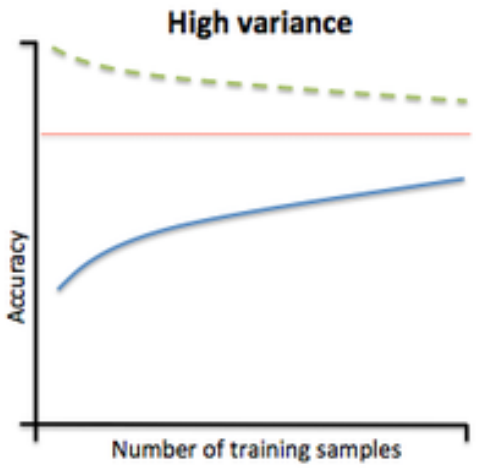
Validation set: use cases

- **Hyperparameter tuning :** train the model on the training set and evaluate its performance on the validation set

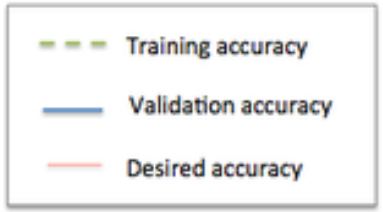
  → adjust the hyperparameters to find the best configuration, ie guide decisions and monitor training process

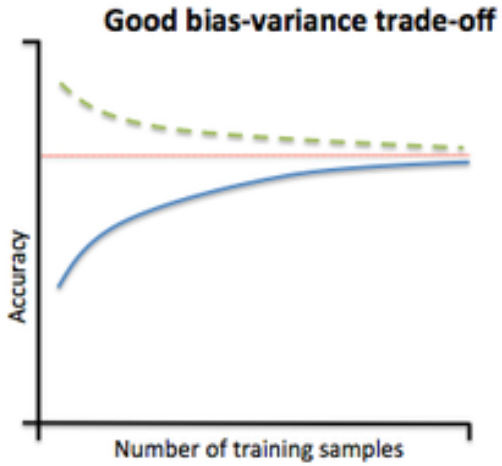
- **Early Stopping**: stop training early to prevent overfitting





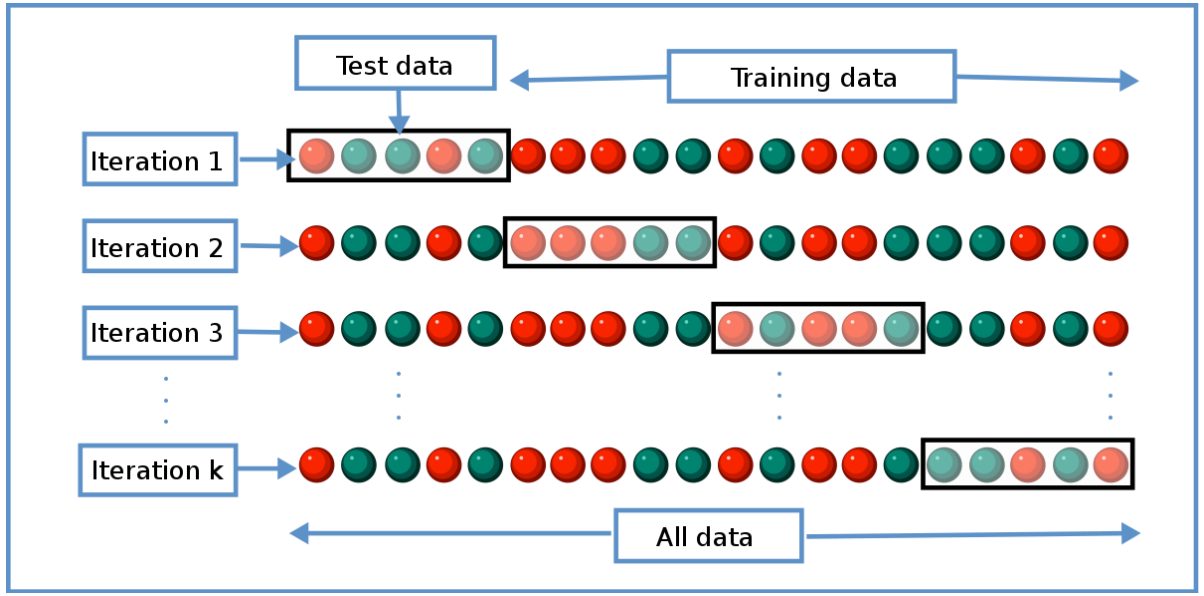
- **Model Selection**: If you're experimenting with different algorithms or architectures, compare their performance on the validation set to choose the best one BEFORE proceeding to the final evaluation on the test set.





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

It should not be used with time series data

Data Preparation

Raw Data

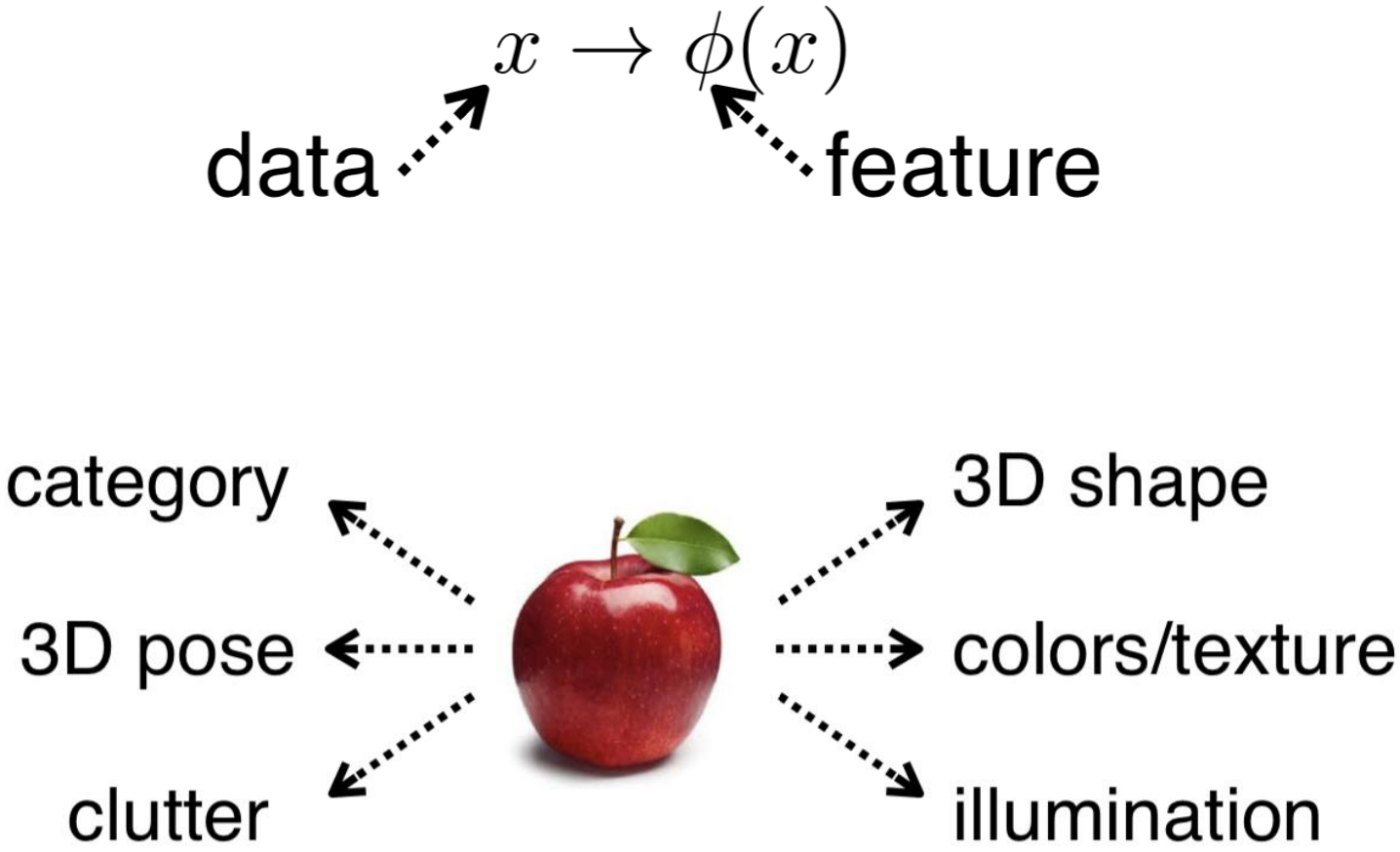
Type	Examples
Nominal	Name, Phone, Gender,…
Ordinal	1.Satisfied, 2. Indifferent, 3. Dissatisfied
Interval	Temperature, time, IQ test values,…
Ratio	<2 hours, 3-4 hours, 4-5 hours,…
Dichotomous	Sex (F/M), completed (Y/N),…

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 (*next slide*)
- Data Wrangling : more general or colloquial term for data preparation that might include some data cleaning and feature engineering.

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

Feature Engineering



1. Data Pre-Processing

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

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

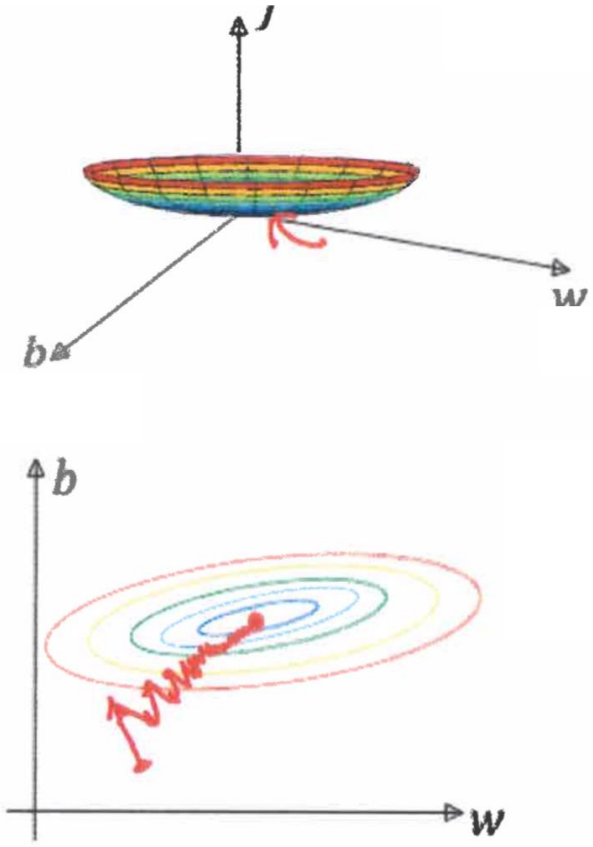
- Shuffling to ensure independent samples and an unbiased estimate

- Remove under-represented categories

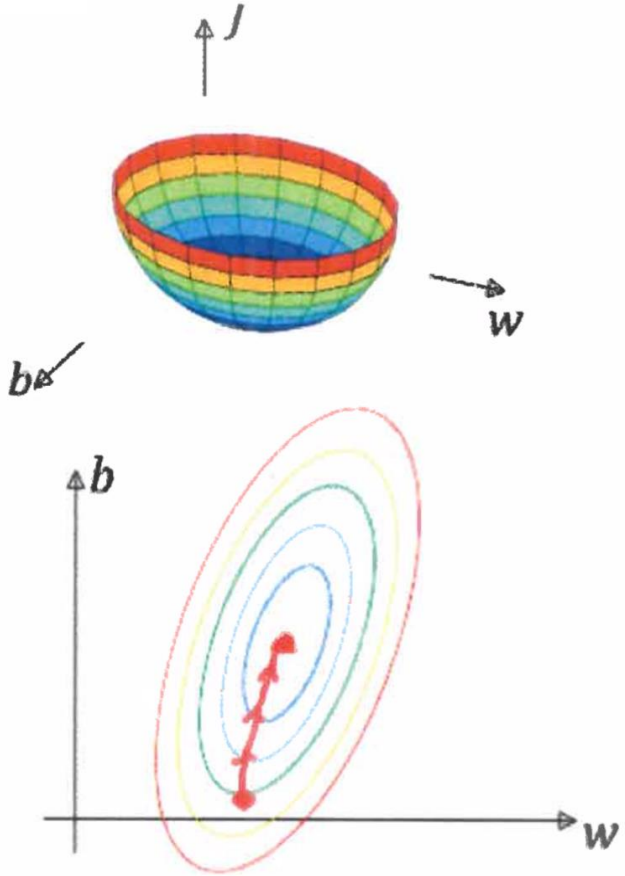
2. Standardization (Normalization)

- for faster convergence and more reliable predictions
- Applied to training data (scaler.fit_transform)
  - Learn mean and standard deviation, which are then applied to the validation and test samples (scaler.transform)

Unnormalized :



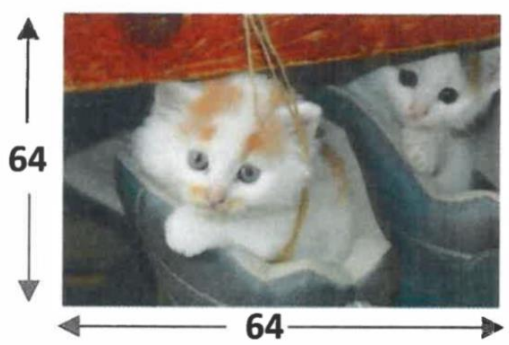
Normalized :



Data Preparation Example : Binary Classification

1. Data Preprocessing :

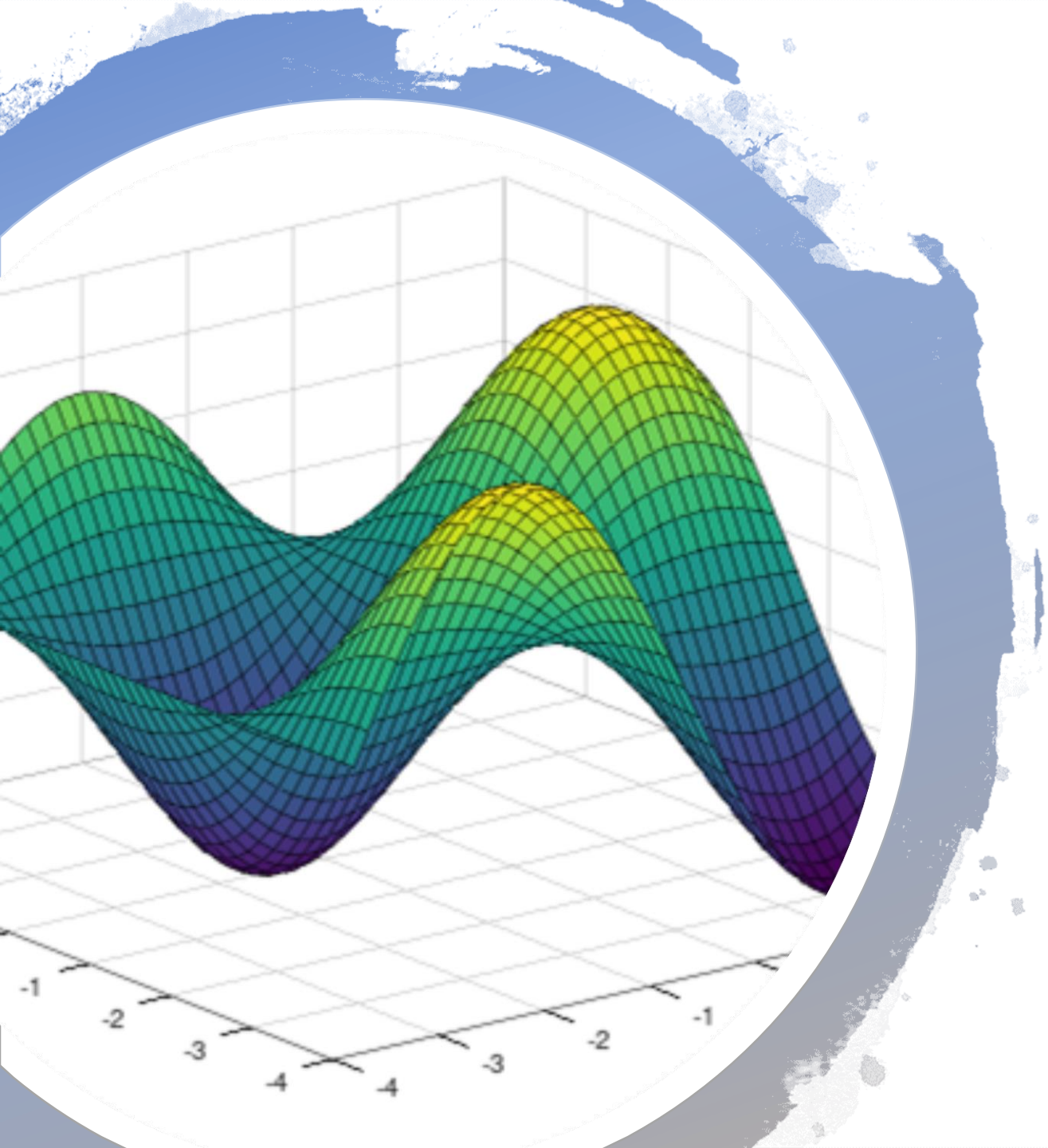
- Image stored in 3 matrices (red, green, blue color channels)
  - Similar size as the image



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

- Reshape the values into a feature vector x

2. Normalization : divide by 255 (more convenient for data)



Optimization

Loss and Cost Functions

- Loss function $L(\hat{y}^{(i)}, y^{(i)})$, 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

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

weights, biases

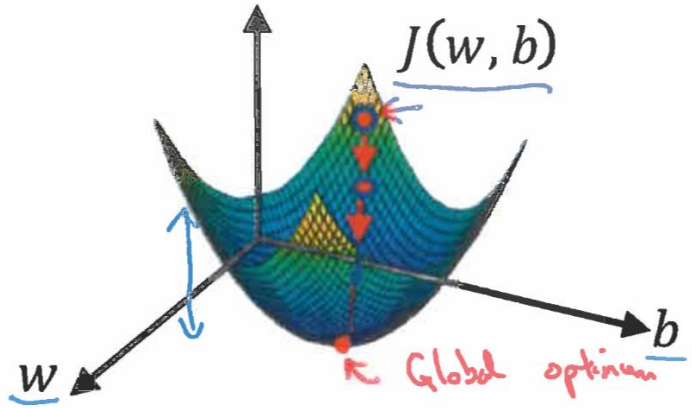
- Goal of the optimization is to find the parameters $\theta = (w, b)$ that minimize the cost function

More complex Loss Function

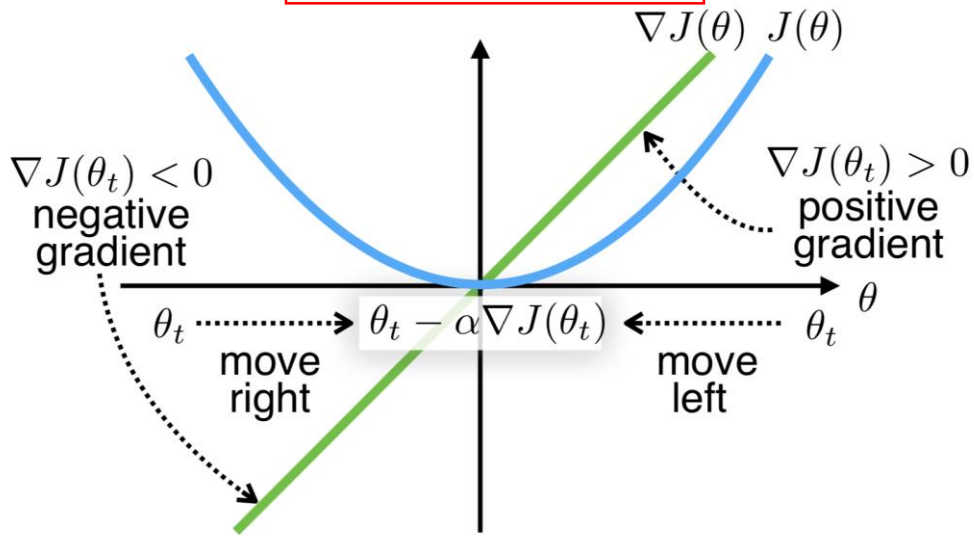
- Can include more features :
  - **Regularization terms**: incorporate additional information to prevent overfitting, like L1 (lasso) and L2 (ridge)
  - **Feature Importance Weights**: assign different degrees of importance to different features (more of less relevant features)
  - **Attention Mechanisms**: give different levels of importance to different parts of the input sequence (transformers).
  - **Multi-Task Learning**: the loss function combines the losses from each task, possibly with different weights to reflect the importance of each task.

Gradient Descent

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



gradient descent $\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$

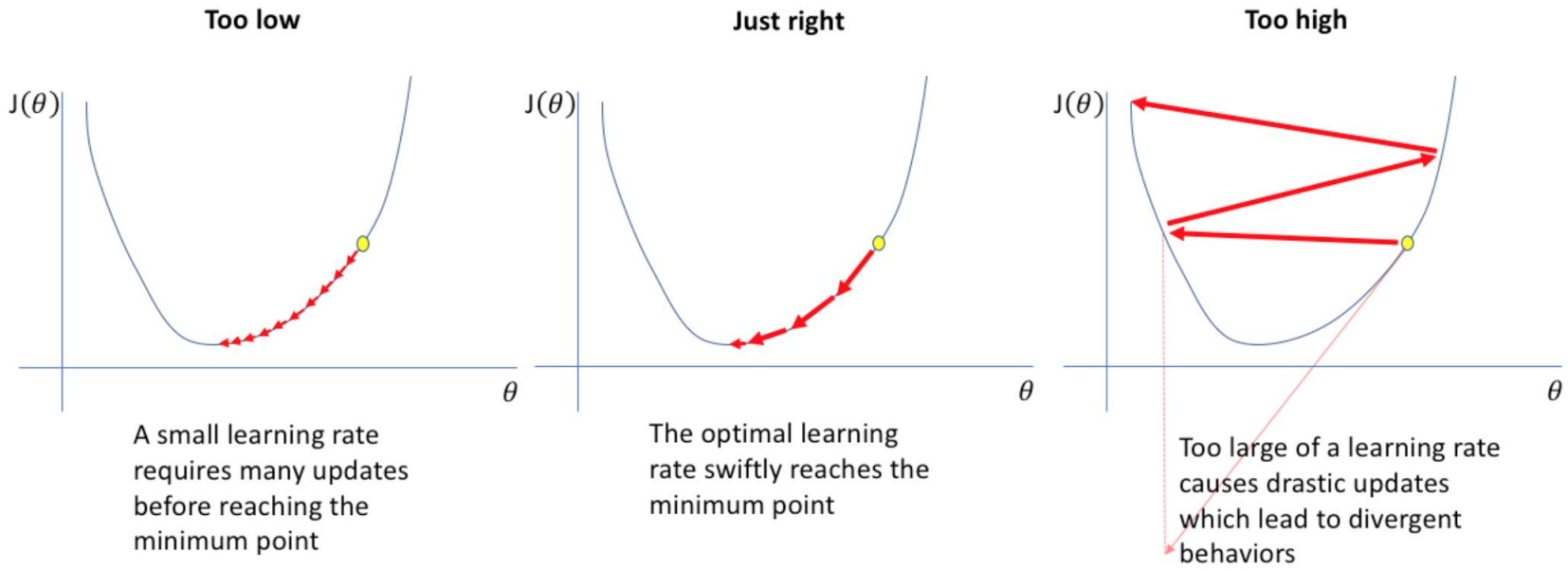


$$\nabla J(w) = \frac{dJ(w,b)}{dw}$$

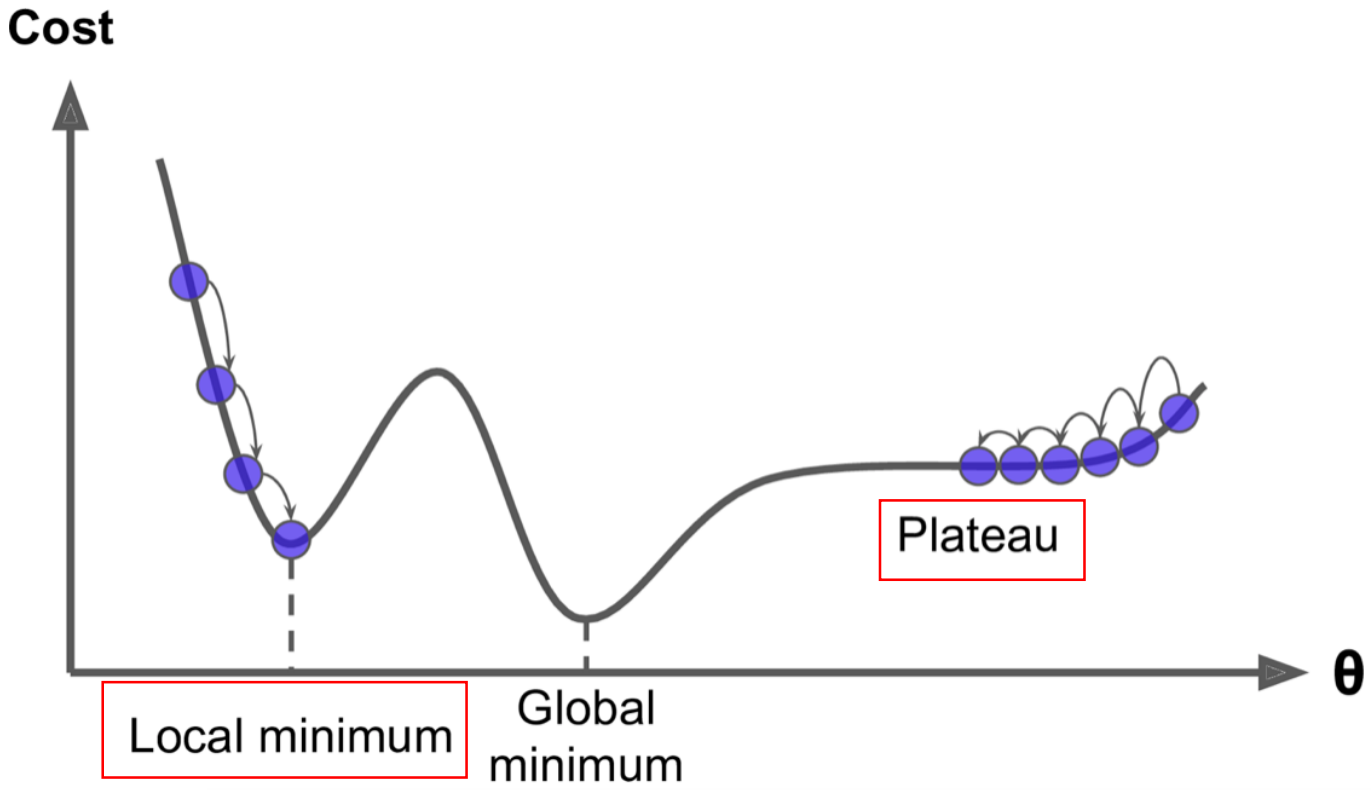
$$\nabla J(b) = \frac{dJ(w,b)}{db}$$

learning rate α

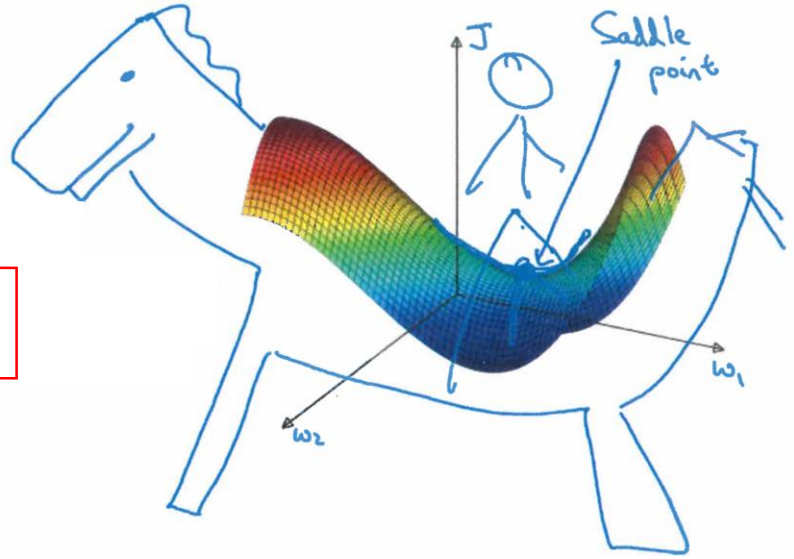
- Has a significant impact on the model performance, while being one of the most difficult parameters to set



Optimization Pitfalls



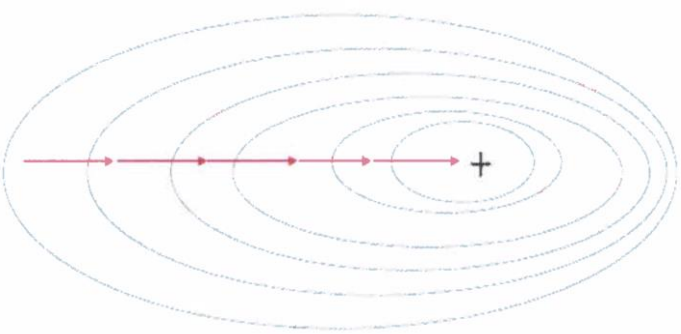
Saddle point



Mini-batch Gradient Descent and Stochastic Gradient Descent

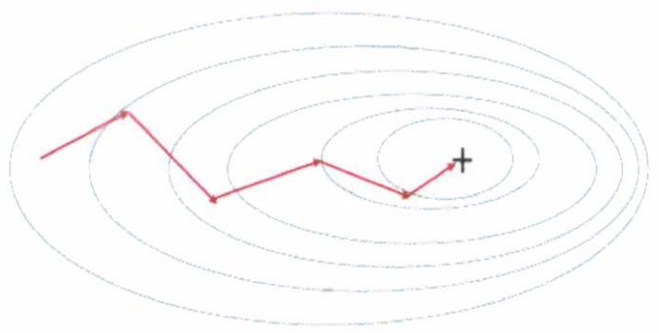
- At each iteration :

Gradient descent (GD)



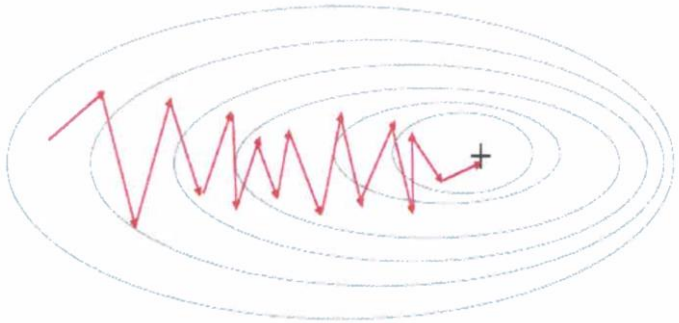
the *whole* training set

Mini-batch gradient descent



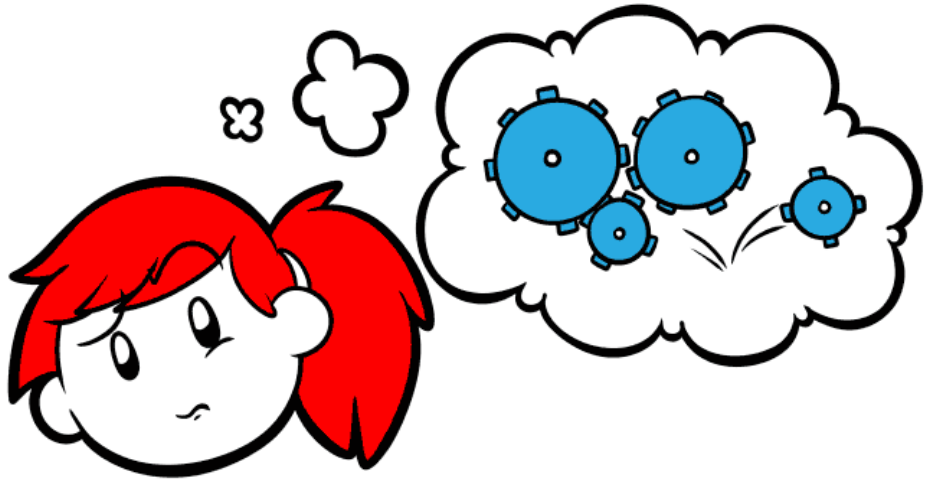
a *batch* of samples

Stochastic gradient descent (SGD)



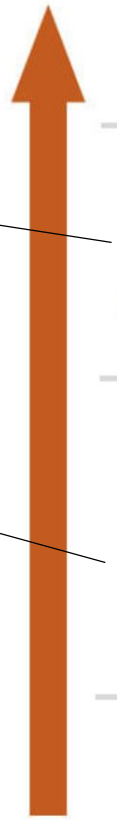
1 sample

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



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

Notebooks (1)

Use Anaconda Navigator

New environment with tensorflow :

conda create -n tf tensorflow

conda activate tf

Notebooks (2)

Add libraries :

conda install matplotlib
conda install scikit-learn
conda install pandas
conda install seaborn
conda install tensorflow
conda install git
conda install pydot
conda install graphviz
conda install anaconda-client

Anaconda cloud : file sharing

- You can run on the Anaconda cloud to get faster results
  - Log into https://anaconda.org/
  - conda install anaconda-client

- Command : anaconda upload notebook_M1D4_BON.ipynb

- Results : https://anaconda.org/username/notebook_m1d4_bon

Cloud computing

	Format	Pros	Cons
Google Colab	.ipynb	No setup required Access to GPU/TPU Integration with Google Drive	Limited runtime and resources for FREE version
Kaggle Kernels	.ipynb	Access to Kaggle datasets Competitive challenges	Limited resource availability Kernel runtime limits
Microsoft Azure	.ipynb	Integration with Microsoft services Access to Azure services	Limites resources and usage
IBM Watson studio	.ipynb	Integration with IBM services Easy data access	Limited resources for FREE version
AWS Educate	.ipynb .py	Real-world cloud experience Access to various AWS services	Limited credits, requires setup
Free Tier Cloud VM	.py	More flexibility Full control over environment	Setup required, limited resources, potential charges if usage exceeds FREE tier limits

Google Colab

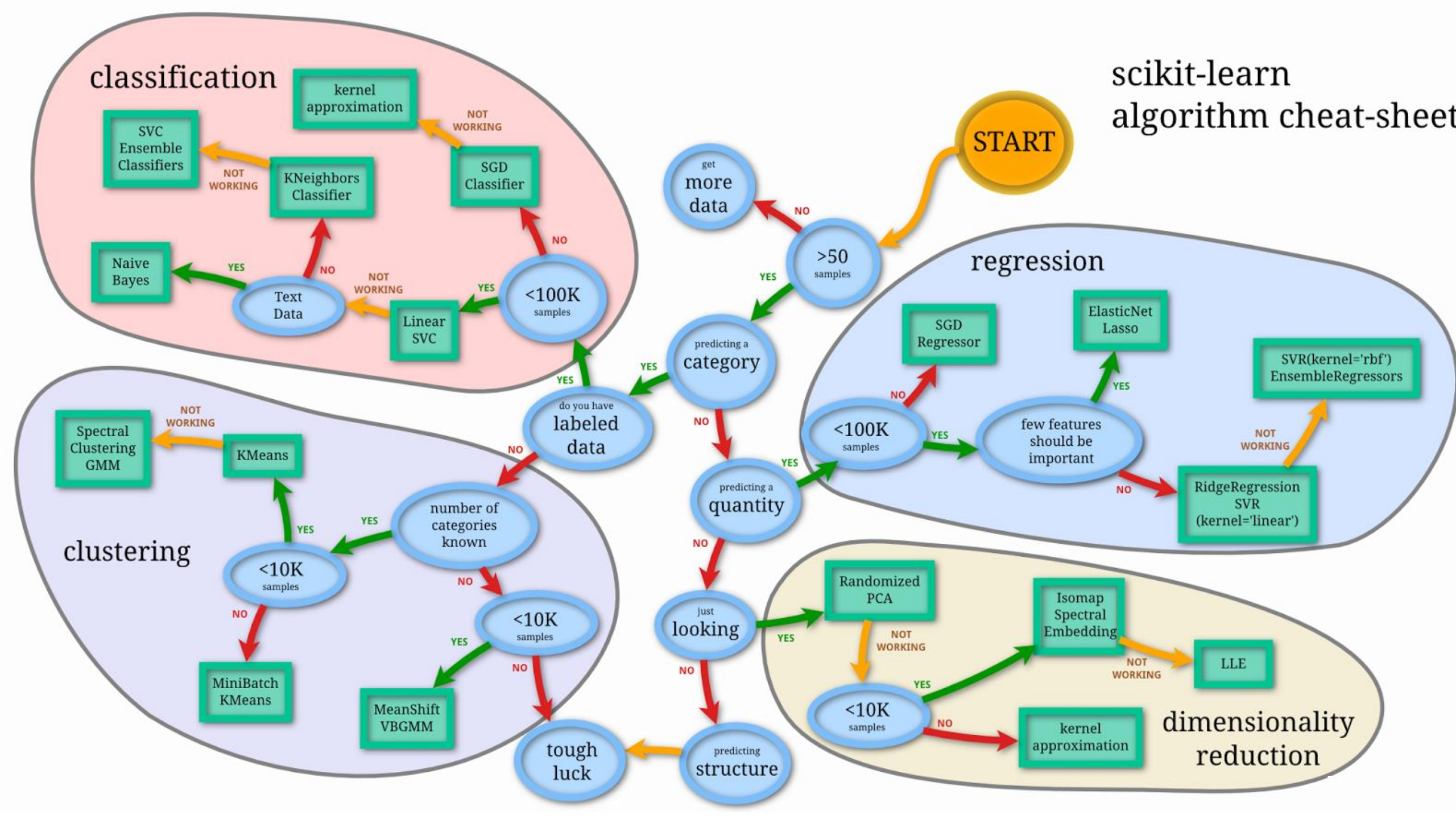
- Here : https://colab.research.google.com/

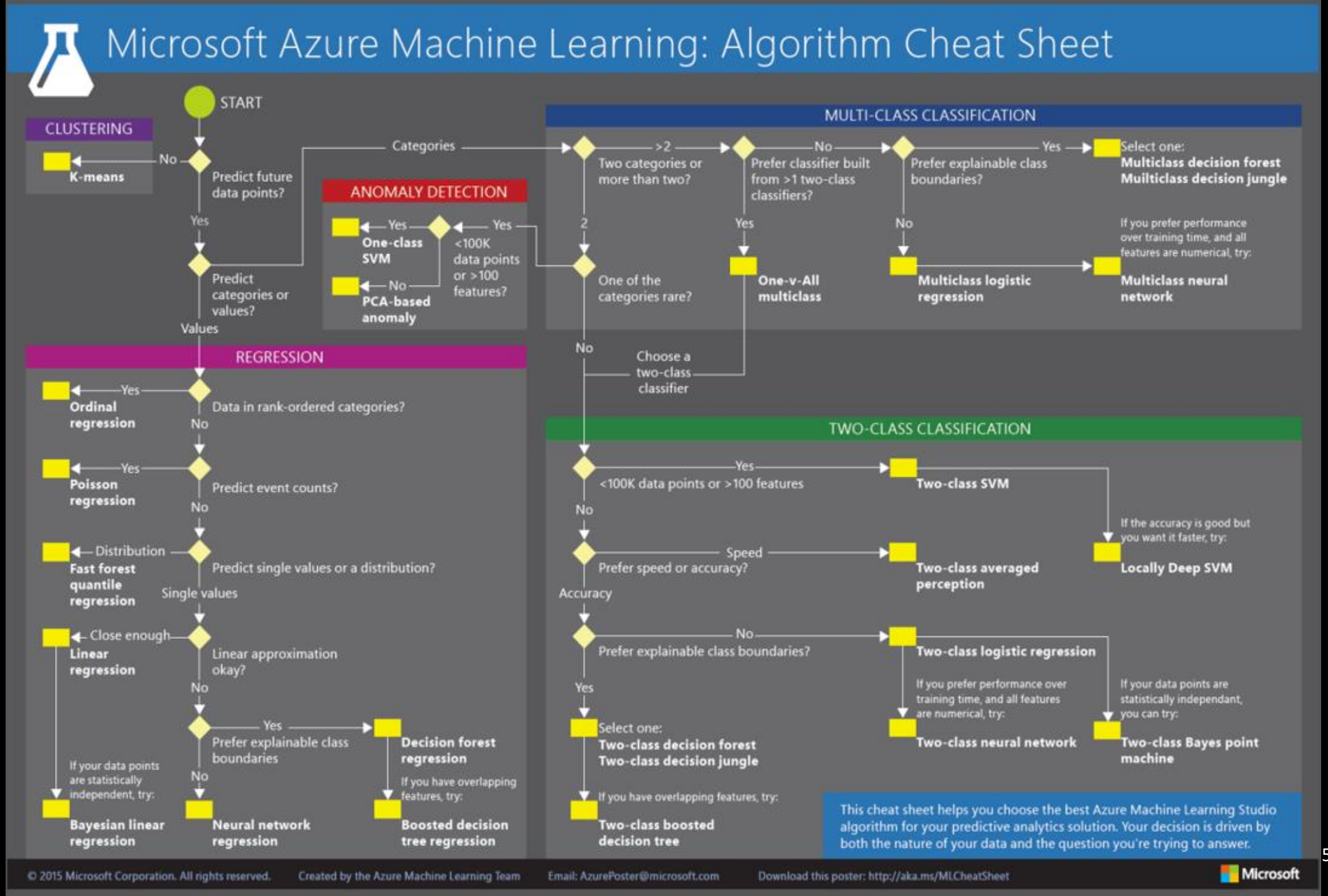
- Import the Jupyter notebook and execute

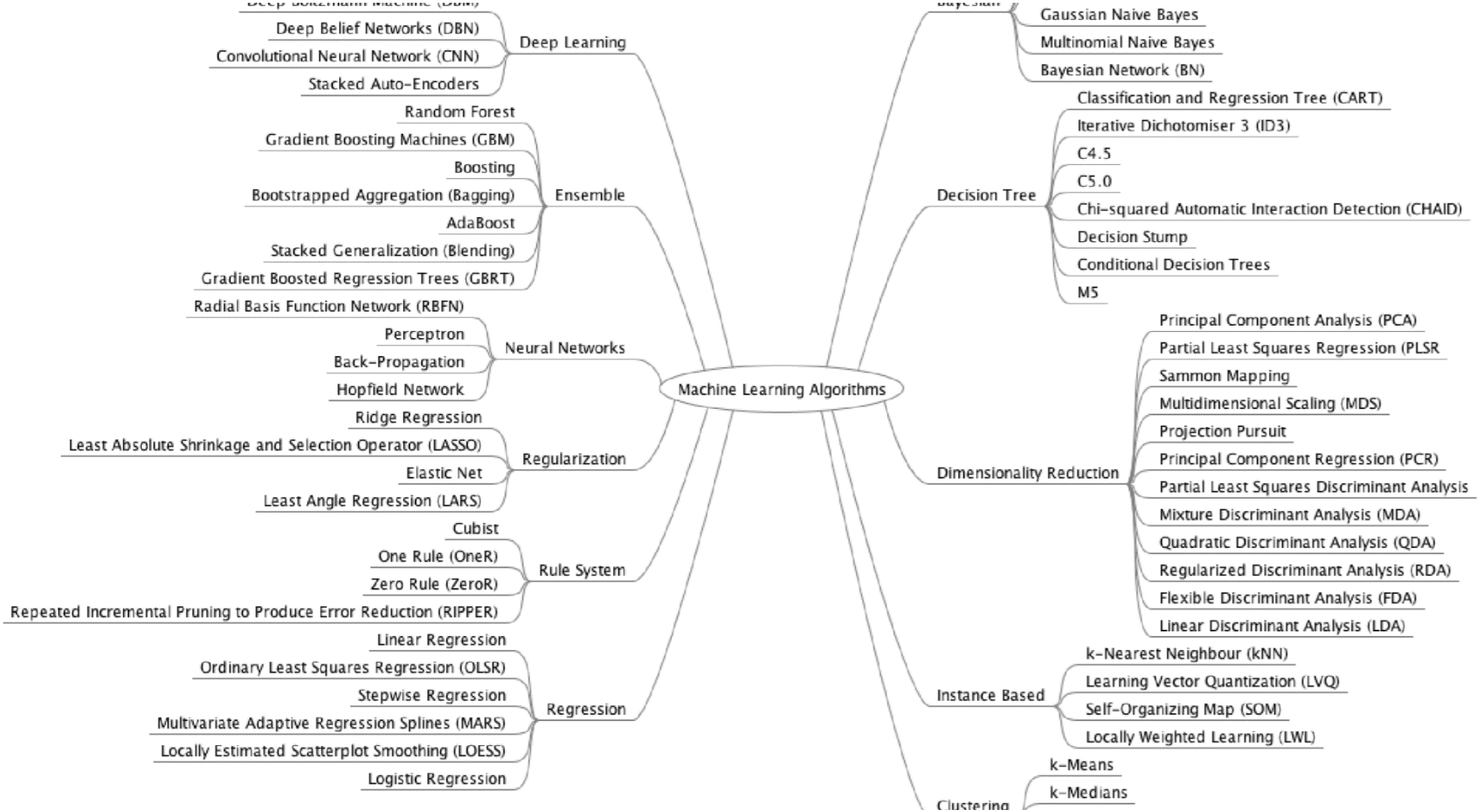
How to find its way in the algorithm jungle ?

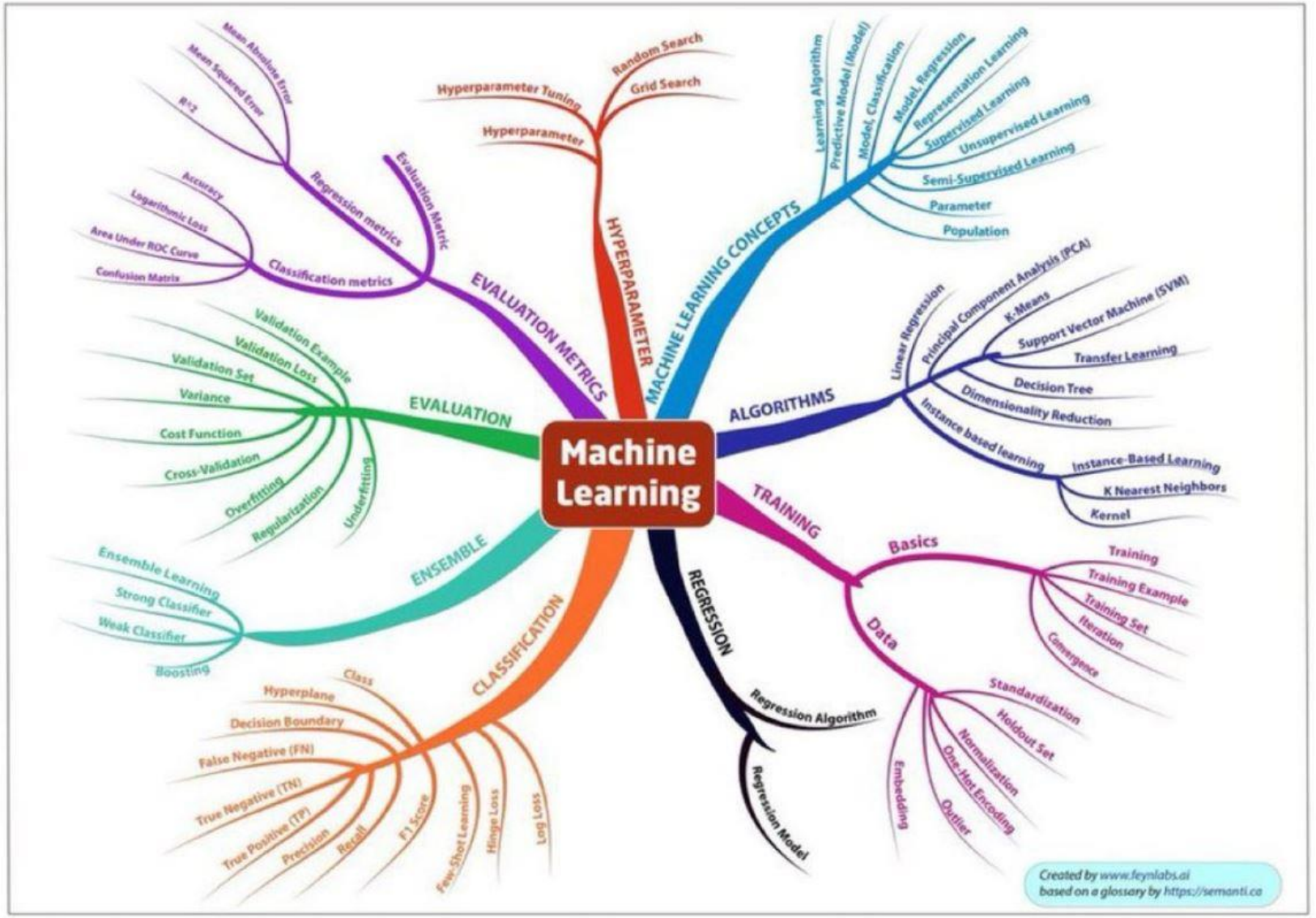
No Free Lunch Theorem

- 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
<100K samples
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC Ensemble Classifiers
SGD Classifier
kernel approximation
classification
<100K samples
SGD Regressor
few features should be important
ElasticNet Lasso
SVR(kernel='rbf') EnsembleRegressors
RidgeRegression SVR (kernel='linear')
regression
number of categories known
<10K samples
KMeans
Spectral Clustering GMM
MiniBatch KMeans
<10K samples
MeanShift VBGMM
clustering
Randomized PCA
Isomap Spectral Embedding
LLE
<10K samples
kernel approximation
dimensionality reduction
YES
NO
NOT WORKING


Microsoft Azure Machine Learning: Algorithm Cheat Sheet
START
CLUSTERING
No
K-means
Predict future data points?
Yes
Predict categories or values?
Categories
Values
ANOMALY DETECTION
Yes
One-class SVM
No
PCA-based anomaly
Yes
<100K data points or >100 features?
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
No
Choose a two-class classifier
Prefer explainable class boundaries?
Yes
Select one:
Multiclass decision forest
Muilticlass decision jungle
No
Multiclass logistic regression
If you prefer performance over training time, and all features are numerical, try:
Multiclass neural network
REGRESSION
Yes
Ordinal regression
Data in rank-ordered categories?
No
Yes
Poisson regression
Predict event counts?
No
Distribution
Fast forest quantile regression
Predict single values or a distribution?
Single values
Close enough
Linear regression
Linear approximation okay?
No
Yes
Prefer explainable class boundaries
Decision forest regression
If you have overlapping features, try:
Boosted decision tree regression
No
Neural network regression
If your data points are statistically independent, try:
Bayesian linear regression
TWO-CLASS CLASSIFICATION
Yes
<100K data points or >100 features
Two-class SVM
No
Speed
Prefer speed or accuracy?
Two-class averaged perception
If the accuracy is good but you want it faster, try:
Locally Deep SVM
Accuracy
No
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
Yes
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)
Clustering
k-Means
k-Medians


Machine Learning
EVALUATION METRICS
Evaluation Metric
Regression metrics
Mean Absolute Error
Mean Squared Error
R^2
Classification metrics
Accuracy
Logarithmic Loss
Area Under ROC Curve
Confusion Matrix
HYPERPARAMETER
Hyperparameter Tuning
Random Search
Grid Search
Hyperparameter
MACHINE LEARNING CONCEPTS
Learning Algorithm
Predictive Model (Model)
Model, Classification
Model, Regression
Representation Learning
Supervised Learning
Unsupervised Learning
Semi-Supervised Learning
Parameter
Population
ALGORITHMS
Linear Regression
Principal Component Analysis (PCA)
K-Means
Support Vector Machine (SVM)
Transfer Learning
Decision Tree
Dimensionality Reduction
Instance based learning
Instance-Based Learning
K Nearest Neighbors
Kernel
TRAINING
Basics
Training
Training Example
Training Set
Iteration
Convergence
Data
Standardization
Holdout Set
Normalization
One-Hot Encoding
Outlier
Embedding
REGRESSION
Regression Algorithm
Regression Model
CLASSIFICATION
Class
Hyperplane
Decision Boundary
False Negative (FN)
True Negative (TN)
True Positive (TP)
Precision
Recall
F1 Score
Few-Shot Learning
Hinge Loss
Log Loss
ENSEMBLE
Ensemble Learning
Strong Classifier
Weak Classifier
Boosting
EVALUATION
Validation Example
Validation Loss
Validation Set
Variance
Cost Function
Cross-Validation
Overfitting
Regularization
Underfitting
Created by www.feynlabs.ai
based on a glossary by https://semanti.ca

General Guidelines

- **Data Amount (number of instances):**
  - More data helps
  - Complex models require more data
  - **Rule of Thumb**: you should have at least 10-20 times as many data points as you have model parameters. This is not a strict rule, but it can help guide you in selecting an appropriate model complexity

- **Feature amount (Number of Dimensions/Features):**
  - **Curse of dimensionality**: As the number of features increases, the amount of data required to generalize well increases exponentially
  - **Feature Selection and Dimensionality Reduction :** reduce the number of dimensions and focus on the most informative ones
  - **Regularization**: help constrain the model's parameters and prevent it from overfitting
  - **Complex Models and Features**: start with simpler models. Complex models with many features can easily overfit the training data and struggle to generalize

The Lesson

The message 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