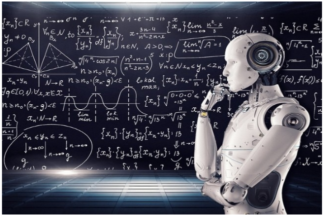


Introduction to Machine Learning

Introduction

Definitions

"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."[a]

"Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed" [b]

[a] Tom Mitchell (1997). Machine Learning. McGraw Hill.

[b] Arthur L Samuel (1959). Some studies in machine learning using the game of checkers. In: IBM Journal of research and development, pp 210-229

Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

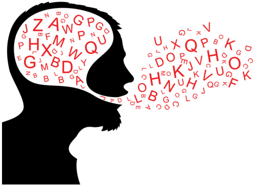
- Human expertise does not exist

Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

- Human expertise does not exist
- Humans are unable to explain their expertise
-
-
-
-
-

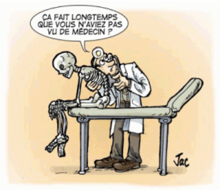


Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

- Human expertise does not exist
- Humans are unable to explain their expertise
- Amount of knowledge is too large for explicit encoding



Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

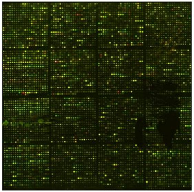
- Human expertise does not exist
- Humans are unable to explain their expertise
- Amount of knowledge is too large for explicit encoding
- Solution changes in time
-
-
-

Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

- Human expertise does not exist
- Humans are unable to explain their expertise
- Amount of knowledge is too large for explicit encoding
- Solution changes in time
- Relationships can be hidden within large amounts of data

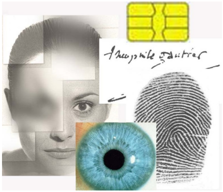


Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

- Human expertise does not exist
- Humans are unable to explain their expertise
- Amount of knowledge is too large for explicit encoding
- Solution changes in time
- Relationships can be hidden within large amounts of data
- Solution needs to be adapted to particular cases
-



Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

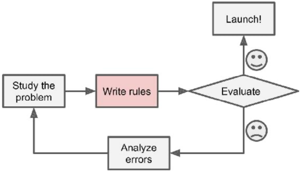
Learning is used when

- Human expertise does not exist
- Humans are unable to explain their expertise
- Amount of knowledge is too large for explicit encoding
- Solution changes in time
- Relationships can be hidden within large amounts of data
- Solution needs to be adapted to particular cases
- New knowledge is constantly being discovered by humans

An illustrative example

Spam detection: naive approach

1. Observe what is a spam and detect recurrent patterns
2. write an algorithm of these patterns
3. If a new email contains these patterns then classify it as a spam
4. iterate until convergence

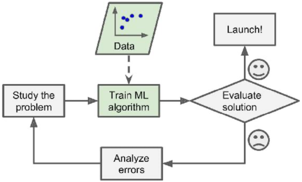


- Complex task
- High number of rules
- Impossible to maintain/update

An illustrative example

Spam detection: Machine Learning approach

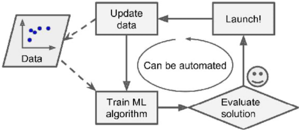
1. A ML spam filter automatically learns relevant patterns
2. Automatic adaptation
3. Can help humans to learn $\longrightarrow$ Data Mining



An illustrative example

Spam detection: Machine Learning approach

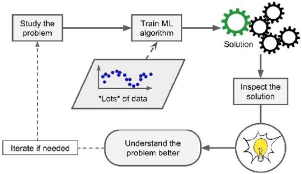
1. A ML spam filter automatically learns relevant patterns
2. Automatic adaptation
3. Can help humans to learn $\longrightarrow$ Data Mining



An illustrative example

Spam detection: Machine Learning approach

1. A ML spam filter automatically learns relevant patterns
2. Automatic adaptation
3. Can help humans to learn $\longrightarrow$ Data Mining



Different types of Machine Learning Algorithms

Several criteria, non exhaustive and combinable

1. Supervised of not
2. Incremental of batch learning
3. Instance-based or model-based algorithms

Taxonomy using supervision

Supervised learning

1. Labeled data
2. Direct feedback
3. Predict outcome

Unupervised learning

1. No Labels/targets
2. No feedback
3. Find hidden structures

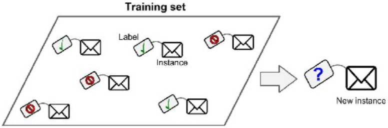
Reinforcement learning

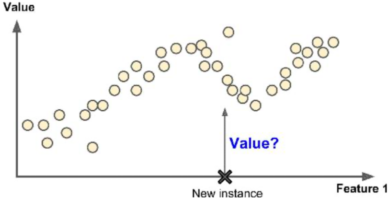
1. Decision process
2. Reward system
3. Learn policy

Supervised Learning

Supervised learning

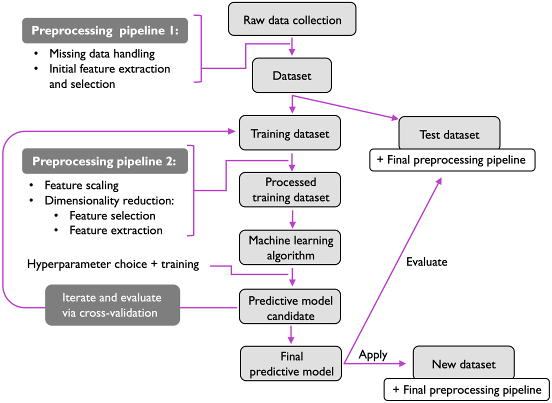
1. Labeled data
2. Direct feedback
3. Predict outcome





Supervised Learning

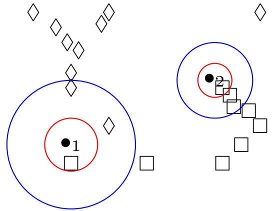
Workflow



Supervised Learning

Main algorithms:

- K-nearest neighbors (K-nn).
- Support Vector Machines (SVM/SVR)
- Linear/logistic regression
- Decision Trees
- Random forests
- Neural networks (shallow and deep)

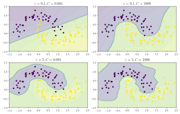


1-NN (red)
and 3-NN (blue) decision rule

Supervised Learning

Main algorithms:

- K-nearest neighbors (K-nn).
- Support Vector Machines (SVM/SVR)
- Linear/logistic regression
- Decision Trees
- Random forests
- Neural networks (shallow and deep)



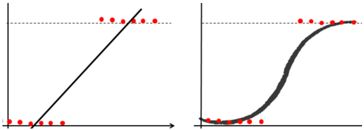
$$\min_{\mathbf{w}\in\mathbb{R}^{\mathbf{d}}} \quad \|\mathbf{w}\|^{\mathbf{2}}$$

$$\text{wrt} \quad y_i(\mathbf{w}^T\mathbf{x_i}) \geq 1, \quad 1 \leq i \leq n$$

Supervised Learning

Main algorithms:

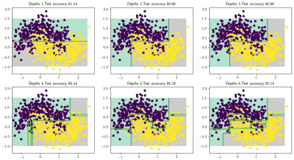
- K-nearest neighbors (K-nn).
- Support Vector Machines (SVM/SVR)
- Linear/logistic regression
- Decision Trees
- Random forests
- Neural networks (shallow and deep)

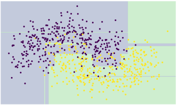


Supervised Learning

Main algorithms:

- K-nearest neighbors (K-nn).
- Support Vector Machines (SVM/SVR)
- Linear/logistic regression
- Decision Trees
- Random forests
- Neural networks (shallow and deep)

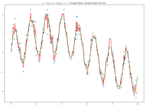




SUPERVISED LEARNING

Main algorithms:

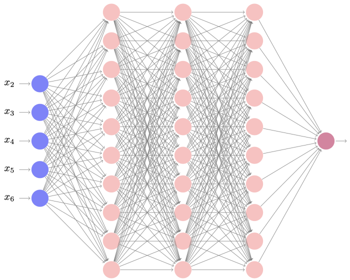
- K-nearest neighbors (K-nn).
- Support Vector Machines (SVM/SVR)
- Linear/logistic regression
- Decision Trees
- Random forests
- Neural networks (shallow and deep)



Supervised Learning

Main algorithms:

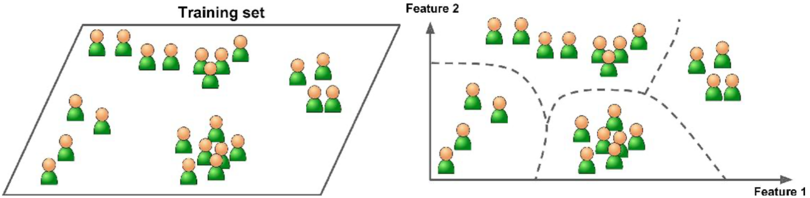
- K-nearest neighbors (K-nn).
- Support Vector Machines (SVM/SVR)
- Linear/logistic regression
- Decision Trees
- Random forests
- Neural networks (shallow and deep)



Unsupervised Learning

Unupervised learning

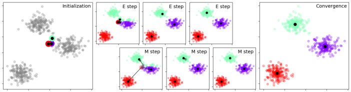
1. No Labels/targets
2. No feedback
3. Find hidden structures



Unsupervised Learning

Main algorithms:

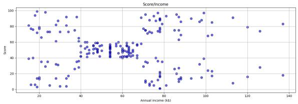
- Clustering
  - K-means and variants
  - Hierarchical cluster analysis
  - EM algorithm
- Visualization
  - Linear methods (PCA, ICA,...)
  - Manifold Learning (ISOMAP, LLE, tSNE,...)
- Association rules

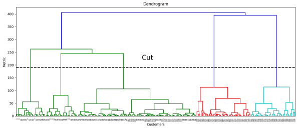


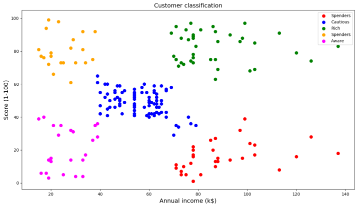
Unsupervised Learning

Main algorithms:

- Clustering
  - K-means and variants
  - Hierarchical cluster analysis
  - EM algorithm
- Visualization
  - Linear methods (PCA, ICA,...)
  - Manifold Learning (ISOMAP, LLE, tSNE,...)
- Association rules



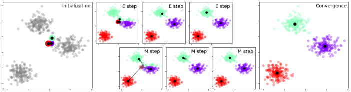




UNSUPERVISED LEARNING

Main algorithms:

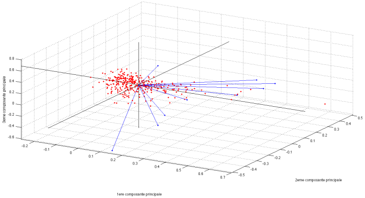
- Clustering
  - K-means and variants
  - Hierarchical cluster analysis
  - EM algorithm
- Visualization
  - Linear methods (PCA, ICA,...)
  - Manifold Learning (ISOMAP, LLE, tSNE,...)
- Association rules



Unsupervised Learning

Main algorithms:

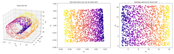
- Clustering
  - K-means and variants
  - Hierarchical cluster analysis
  - EM algorithm
- Visualization
  - Linear methods (PCA, ICA,...)
  - Manifold Learning (ISOMAP, LLE, tSNE,...)
- Association rules



Unsupervised Learning

Main algorithms:

- Clustering
  - K-means and variants
  - Hierarchical cluster analysis
  - EM algorithm
- Visualization
  - Linear methods (PCA, ICA,...)
  - Manifold Learning (ISOMAP, LLE, tSNE,...)
- Association rules



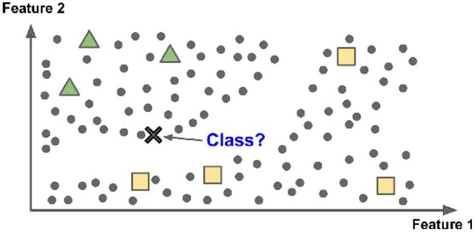
Unsupervised Learning

Main algorithms:

- Clustering
  - K-means and variants
  - Hierarchical cluster analysis
  - EM algorithm
- Visualization
  - Linear methods (PCA, ICA,...)
  - Manifold Learning (ISOMAP, LLE, tSNE,...)
- Association rules

Semi-supervised Learning

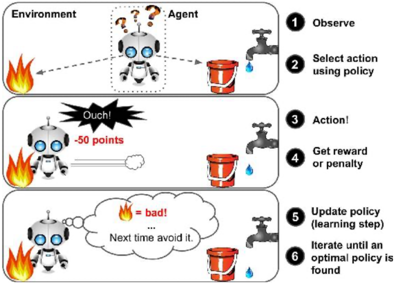
- Mix between supervised and unsupervised learning
- Some training examples contain outputs, but some do not
- Use the labeled training subset to label the unlabeled portion of the training set, which are then also utilized for model training



Reinforcement Learning

Reinforcement learning

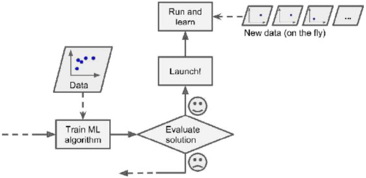
1. Decision process
2. Reward system
3. Learn policy



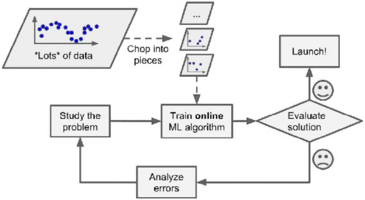
Incremental/batch learning

Availability of the data ?

- Yes: batch Learning
  - Time consuming
  - Must train offline
  - If new data, no update easily possible → Full training
- No: online learning
  - Incremental learning
  - Fast
  - Interests: data flow/ limited ressources / data that cannot be stored.



Online learning



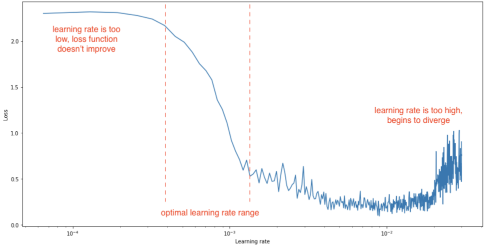
Huge amount of data

Online learning

Learning new data / Forgetting the old one

- too often: instability/ sensitivity to outliers
- too rarely: no adaptation

⇒Learning rate



Generalisation

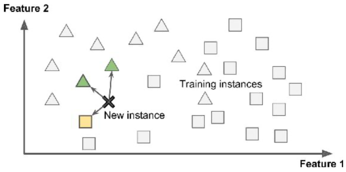
Generalisation

Capacity of an algorithm to predict relevant values/labels on unseen data.

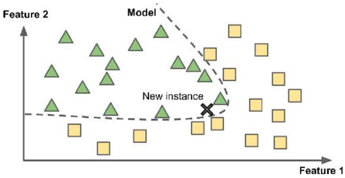
When building an algorithm, 2 strategies are possible:

1. Instance-based algorithm: only relying on the training set
   - Easy to learn by heart
   - How to allow generalization ?
2. Model-based: use of a parametric model
   - What kind of model ?
   - How to tune parameters ?

Generalisation



Instance based



Model based

Introduction

Two things can go wrong

1. Bad data
2. Bad algorithms

Not enough data

Not enough data

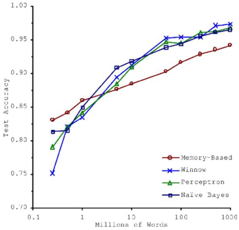
- A child can learn (and generalize) what is an apple with only few examples
- A machine learning algorithm needs thousands of examples

Not enough data

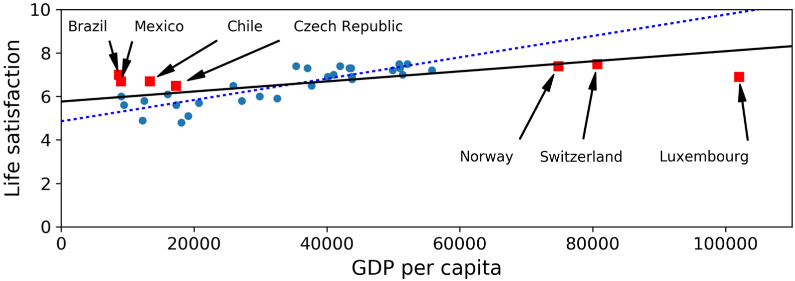
Not enough data

- A child can learn (and generalize) what is an apple with only few examples
- A machine learning algorithm needs thousands of examples

Few data → a simple algorithm.



Unrepresentative data



Poor quality data

Clean the data

- Remove or correct outliers
- Missing values:
  - Ignore the feature
  - Ignore the individual
  - Impute values
  - Combine...

Poor quality data

Clean the data

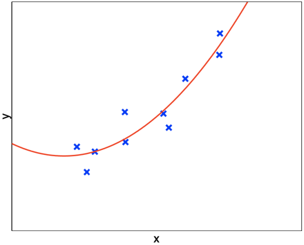
- Remove or correct outliers
- Missing values:
    - Ignore the feature
    - Ignore the individual
    - Impute values
    - Combine...

Unsignificant features

- Feature selection (from the original ones)
- Feature extraction (linear/non linear dimension reduction)
- Collect new features

Under/Overfitting

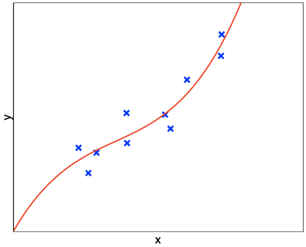
Problem: Linear Least square problem.



Order 2

Under/Overfitting

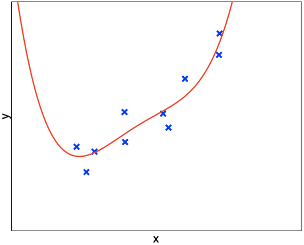
Problem: Linear Least square problem.



Order 3

Under/Overfitting

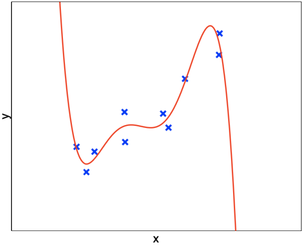
Problem: Linear Least square problem.



Order 4

Under/Overfitting

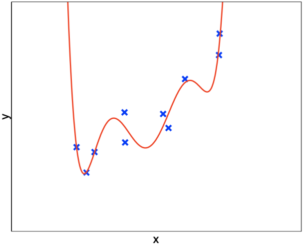
Problem: Linear Least square problem.



Order 5

Under/Overfitting

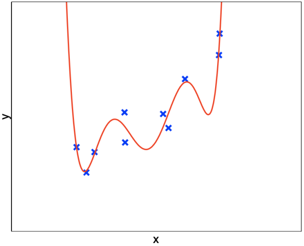
Problem: Linear Least square problem.



Order 6

Under/Overfitting

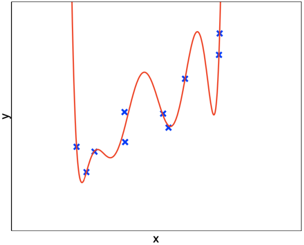
Problem: Linear Least square problem.



Order 7

Under/Overfitting

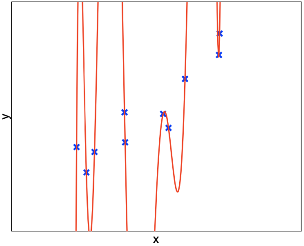
Problem: Linear Least square problem.



Order 8

Under/Overfitting

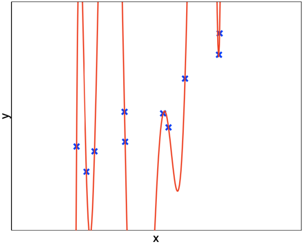
Problem: Linear Least square problem.



Order 9

Under/Overfitting

Problem: Linear Least square problem.



Order 9

The model is...

- Too simple: underfitting
- Too complex: overfitting

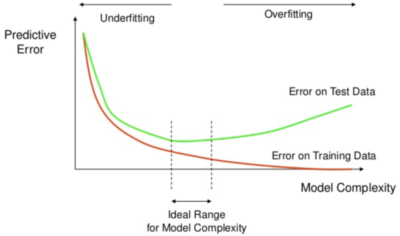
Training and Test sets

Don't trust a priori...

Once the model is trained, there is a need to test if it is reliable and performs well.

$$Z = Z_L \cup Z_T$$

- Z_L: training set (training error)
- Z_T: test set (generalization error)



Training, Test and Validation sets

Comparing several models.

Need for an additional set when comparing several algorithms

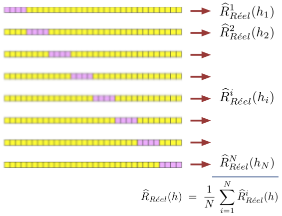
$$Z = Z_L \cup Z_T \cup Z_V$$

- Z_L: training set (training error)
- Z_T: test set (generalization error)
- Z_V: validation set (hyperparameters tuning)

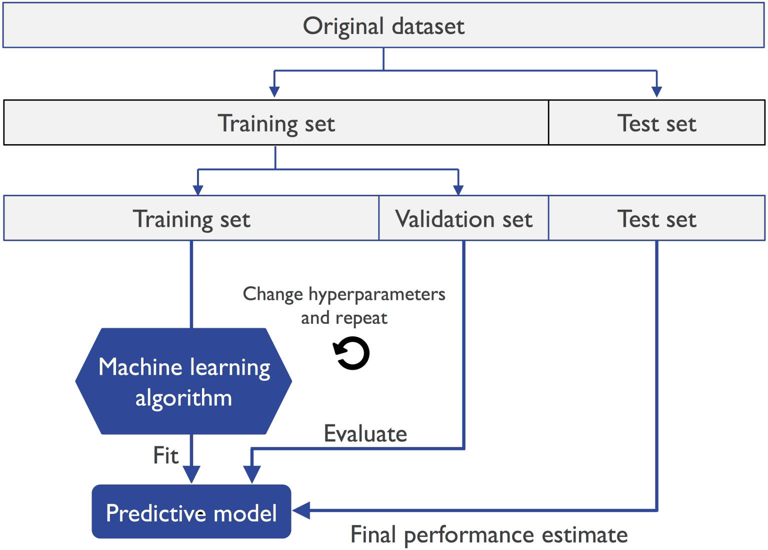
Risk: learning Z_V.

Solution: Cross validation.

- Use different partitions $Z = Z_L \cup Z_T \cup Z_V$
- keep the best model



Training, Test and Validation sets



Performance measure

Measuring the performance of a classifier is generally harder than for a regression algorithm.

- cross validation (can be difficult if the classes are non equilibrated)
- confusion matrix (binary and multiclass cases)

Performance measure

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Example: binary confusion matrix C

- $C_{1,1}$: true positives (TP)
- $C_{2,2}$: true negatives (TN)
- $C_{1,2}$: false positives (FP)
- $C_{2,1}$: false negatives (FN)

Performance measure

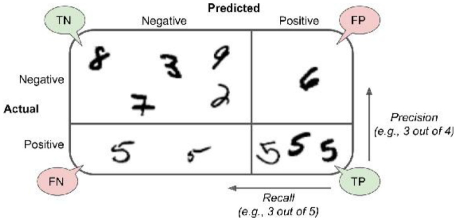
$$C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Example: binary confusion matrix C

- $C_{1,1}$: true positives (TP)
- $C_{2,2}$: true negatives (TN)
- $C_{1,2}$: false positives (FP)
- $C_{2,1}$: false negatives (FN)

Precision / Recall

- precision $P = \frac{TP}{TP+FP}$:
- recall $\frac{TP}{TP+FN}$



Performance measure

F_1 score

$$F_1 = 2\frac{P.R}{P+R} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Performance measure

F_1 score

$$F_1 = 2\frac{P.R}{P+R} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Harmonic mean

Good performances for classifiers with similar P and R values

Performance measure

F_1 score

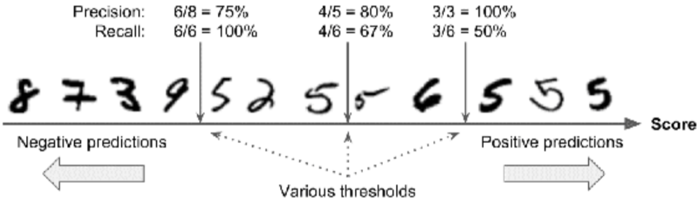
$$F_1 = 2\frac{P.R}{P+R} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Harmonic mean

Good performances for classifiers with similar P and R values

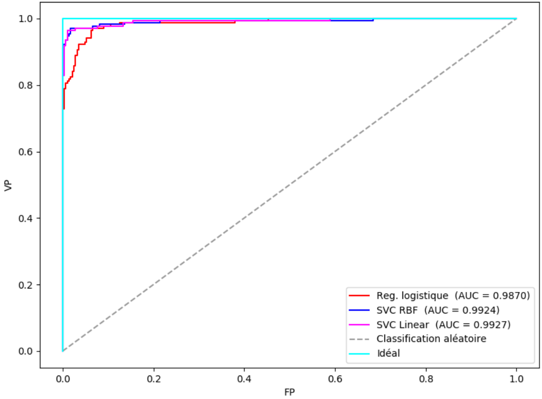
P/R compromise

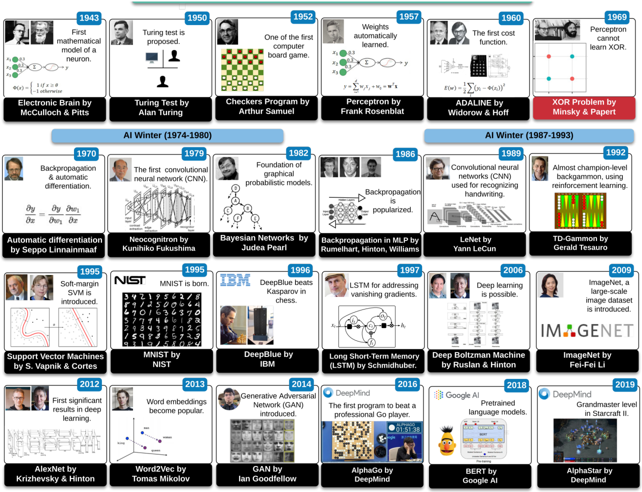
- In general, improving P lowers R and vice versa.
- Decision function, returning a value compared to a threshold



Performance measure

ROC curves




1943
First mathematical model of a neuron.
Electronic Brain by McCulloch & Pitts
1950
Turing test is proposed.
Turing Test by Alan Turing
1952
One of the first computer board game.
Checkers Program by Arthur Samuel
1957
Weights automatically learned.
Perceptron by Frank Rosenblatt
1960
The first cost function.
ADALINE by Widorow & Hoff
1969
Perceptron cannot learn XOR.
XOR Problem by Minsky & Papert
AI Winter (1974-1980)
AI Winter (1987-1993)
1970
Backpropagation & automatic differentiation.
Automatic differentiation by Seppo Linnainmaa
1979
The first convolutional neural network (CNN).
Neocognitron by Kunihiko Fukushima
1982
Foundation of graphical probabilistic models.
Bayesian Networks by Judea Pearl
1986
Backpropagation is popularized.
Backpropagation in MLP by Rumelhart, Hinton, Williams
1989
Convolutional neural networks (CNN) used for recognizing handwriting.
LeNet by Yann LeCun
1992
Almost champion-level backgammon, using reinforcement learning.
TD-Gammon by Gerald Tesauro
1995
Soft-margin SVM is introduced.
Support Vector Machines by S. Vapnik & Cortes
1995
NIST
MNIST is born.
MNIST by NIST
1996
IBM
DeepBlue beats Kasparov in chess.
DeepBlue by IBM
1997
LSTM for addressing vanishing gradients.
Long Short-Term Memory (LSTM) by Schmidhuber.
2006
Deep learning is possible.
Deep Boltzman Machine by Ruslan & Hinton
2009
ImageNet, a large-scale image dataset is introduced.
IMAGENET
ImageNet by Fei-Fei Li
2012
First significant results in deep learning.
AlexNet by Krizhevsky & Hinton
2013
Word embeddings become popular.
man
woman
king
queen
Word2Vec by Tomas Mikolov
2014
Generative Adversarial Network (GAN) introduced.
GAN by Ian Goodfellow
2016
DeepMind
The first program to beat a professional Go player.
AlphaGo by DeepMind
2018
Google AI
Pretrained language models.
BERT
BERT by Google AI
2019
DeepMind
Grandmaster level in Starcraft II.
AlphaStar by DeepMind

AND NOW, SOME MATHS...

Statistical Learning theory

Learning Model (Vapnik)

1. A generator (G) of random vectors $x \in D$, i.i.d, $P(x)$ fixed but unknown;
2. A supervisor (S) giving for each input x a value $y \in C$ drawned from $P(y|x)$ fixed but unknown;
3. A Learning Machine (LM) implementing a set of functions $\mathcal{F}$.

Statistical Learning theory

Problem statement

Find $f \in \mathcal{F}$ that best fit the supervisor S

Training Set

$$Z = \{(x_1, y_1), \ldots, (x_l, y_l)\}$$

l observations i.i.d from $P(x, y) = P(x) P(y|x)$.

Find $f : D \to C$ such as $R(f) = P(y \neq f(x))$ is minimal.

Loss function and Error

Loss function

$$L(y, f(x)) = \mathbb{1}_{y \neq f(x)}$$

Difference between S (y) and LM $(f(x))$

Risk or Error

$$R(f) = \int L(y, f(x))\, dP(x, y) = P(y \neq f(x))$$

$\Rightarrow$ Expected value of the loss function = probability that f predicts a different value of S.

Classification, regression and density estimation

Problem statement

Knowing Z, find $f \in \mathcal{F}$ such that

$$f = Arg \min_{g \in \mathcal{F}} R(g)$$

Some comments

Depending on L, this formulation allows to address classification, regression and density estimation problems, e.g.

- Classification: $L(y, f(x)) = \mathbb{1}_{y \neq f(x)}$
- Regression: $L(y, f(x)) = (y - f(x))^2$
- Density estimation: $L(y, f(x)) = -log(f(x))$

Minimum risk function: Bayes' function

For classification problem, $\exists$ a minimum risk function

$$f_{Bayes}\left(x\right) = Arg\max_{y} P\left(y|x\right)$$

f_{Bayes} :"Ideal" function to reach (no hypothesis on the underlying distributions)

Problem statement

Knowing Z, approximate f_{Bayes} with $f \in \mathcal{F}$.[a]

[a] a priori $f_{Bayes} \notin \mathcal{F}$

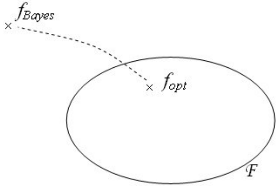
Minimum risk function: Bayes' function

Let suppose there exists $f_{opt} \in \mathcal{F}$ of minimal risk:

$$0 \leq R\left(f_{Bayes}\right) \leq R\left(f_{opt}\right) = \underbrace{R\left(f_{Bayes}\right)}_{\text{non-deterministic}} + \underbrace{\left(R\left(f_{opt}\right) - R\left(f_{Bayes}\right)\right)}_{\text{structural error}}$$

Choice of $\mathcal{F}$

- Using expressive $\mathcal{F}$ spaces to allow $R\left(f_{opt}\right) \approx R\left(f_{Bayes}\right)$
- Not too rich, otherwise risk of overfitting



Empirical risk

Natural idea: find $f \in \mathcal{F}$ that best classify Z

Empirical risk

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l} L(y_i, f(x_i)) = \frac{Card\{i | f(x_i) \neq y_i\}}{l}$$

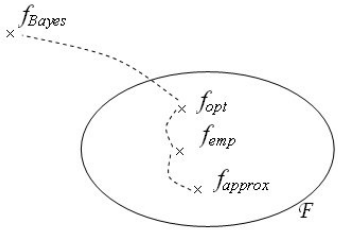
Empirical Risk Minimization(ERM)

Find $f \in \mathcal{F}$ (f_{emp}) minimizing $R_{emp}(f)$

$$R(f_{emp}) = R(f_{Bayes}) + (R(f_{opt}) - R(f_{Bayes})) + (R(f_{emp}) - R(f_{opt}))$$

Empirical risk

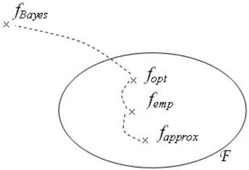
One cannot expect to compute f_{emp} is a reasonable time
$\rightarrow$ Approximation f_{approx} of f_{emp}.



EMPIRICAL RISK

At least four reasons altering the results of a classfication method:

- *Nature of the problem* : minimum $\Rightarrow$ Bayes and can be important;
- *low expressivity of $\mathcal{F}$* : structural error;
- *Non consistancy of ERM principle* : do we get close to f_{opt} with Z? (be careful: learning by heart !!);
- *Minimization can be computationaly hard/unstable.*



Uniform convergence of the empirical risk

ERM does not necesseraly get close to rhe real risk (Z is randomly drawned)

Serious problem !!

- f_{opt}close to $f_{bayes} \Rightarrow$ rich$\mathcal{F}$
- To find f_{opt} by ERM, $\mathcal{F}$ not too rich....

Uniform convergence of the empirical risk

ERM does not necesseraly get close to rhe real risk (Z is randomly drawned)

Extreme cases

- $\mathcal{F} = \{f_{opt}\}$: easy to find but probably high R_{emp}
- $\mathcal{F}$: set of all possible functions $\Rightarrow f_{bayes} \in \mathcal{F}$ but also all f minimizing R_{emp} and in particular $f_{byheart}$.

Uniform convergence of the empirical risk

ERM does not necesseraly get close to rhe real risk (Z is randomly drawned)

Bias-variance tradefoff

- Biais $\approx$ distance between f_{bayes} andf_{opt}
- Variance $\approx$ distance between f_{opt} and f_{emp}

Some results

Definition

The Empirical risk uniformly converges (in probability) to the real risk in $\mathcal{F}$ iff

$$(\forall \epsilon > 0) \lim_{l \to \infty} Pr\left\{Max_{f \in \mathcal{F}} \left| R_{emp}^{l}(f) - R(f) \right| \geq \epsilon \right\} = 0$$

Some results

Proposition:

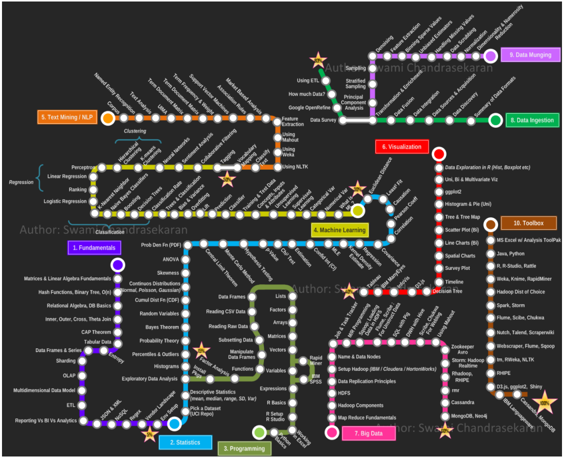
If the empirical risk uniformly converges to the real risk then a LM based on ERM converges in probability to f_{opt}:

$$\lim_{l \to \infty} Pr \left\{ \left| R\left(f^l_{emp}\right) - R\left(f_{opt}\right) \right| \geq \epsilon \right\} = 0$$

The VP dimension D_f allows to precise some points: if $D_f < \infty$, and $|Z| = l$, then with probability at least $1 - \eta$:

$$\left| R^l_{emp}\left(f\right) - R\left(f\right) \right| \leq \sqrt{\frac{D_f}{l}\left(1 + ln\left(\frac{2l}{D_f}\right)\right) - \frac{1}{l} log \frac{\eta}{4}}$$

$\Rightarrow$ if $D_f < \infty$, ERM allows to converge to a function of minimum risk in $\mathcal{F}$.


1. Fundamentals
2. Statistics
3. Programming
4. Machine Learning
5. Text Mining / NLP
6. Visualization
7. Big Data
8. Data Ingestion
9. Data Munging
10. Toolbox
Matrices & Linear Algebra Fundamentals
Hash Functions, Binary Tree, O(n)
Relational Algebra, DB Basics
Inner, Outer, Cross, Theta Join
CAP Theorem
Tabular Data
Data Frames & Series
Entropy
Sharding
OLAP
Multidimensional Data Model
ETL
Reporting Vs BI Vs Analytics
JSON & XML
NoSQL
Regex
Vendor Landscape
Env Setup
Prob Den Fn (PDF)
ANOVA
Skewness
Continuos Distributions (Normal, Poisson, Gaussian)
Cumul Dist Fn (CDF)
Random Variables
Bayes Theorem
Probability Theory
Percentiles & Outliers
Histograms
Exploratory Data Analysis
Descriptive Statistics (mean, median, range, SD, Var)
Pick a Dataset (UCI Repo)
Central Limit Theorem
Monte Carlo Method
Hypothesis Testing
p-Value
Chi² Test
Estimation
Confid Int (CI)
MLE
Kernel Density Estimate
Regression
Covariance
Install Pkgs
Factor Analysis
Data Frames
Reading CSV Data
Reading Raw Data
Subsetting Data
Manipulate Data Frames
Functions
Lists
Factors
Arrays
Matrices
Vectors
Variables
Expressions
R Basics
R Setup R Studio
Rapid Miner
IBM SPSS
Python Basics
Working in Excel
Regression
Perceptron
Linear Regression
Ranking
Logistic Regression
Clustering
Hierarchical Clustering
K-means Clustering
Neural Networks
Sentiment Analysis
Collaborative Filtering
Tagging
Vocabulary Mapping
Classify Text
Using NLTK
K-Nearest Neighbor
Naive Bayes Classifiers
Boosting
Decision Trees
Classification Rate
Trees & Classification
Bias & Variance
Overfitting
Lift
Prediction
Classifier
Training & Test Data
Concepts, Inputs & Attributes
Unsupervised Learning
Supervised Learning
Categorical Var
Numerical Var
What is ML?
Euclidean Distance
Least² Fit
Causation
Pearson Coeff
Correlation
Classification
Corpus
Named Entity Recognition
Text Analysis
UIMA
Term Document Matrix
Term Frequency & Weight
Support Vector Machines
Association Rules
Market Based Analysis
Feature Extraction
Using Mahout
Using Weka
Data Survey
Google OpenRefine
How much Data?
Using ETL
Sampling
Stratified Sampling
Principal Component Analysis
Transformation & Enrichment
Data Fusion
Data Integration
Data Sources & Acquisition
Data Discovery
Summary of Data Formats
Denoising
Feature Extraction
Binning Sparse Values
Unbiased Estimators
Handling Missing Values
Data Scrubbing
Normalization
Dimensionality & Numerosity Reduction
Data Exploration in R (Hist, Boxplot etc)
Uni, Bi & Multivariate Viz
ggplot2
Histogram & Pie (Uni)
Tree & Tree Map
Scatter Plot (Bi)
Line Charts (Bi)
Spatial Charts
Survey Plot
Timeline
Decision Tree
D3.js
InfoVis
IBM ManyEyes
Tableau
Job & Task Tracker
M/R Programming
Sqoop: Loading Data in HDFS
Flume, Scribe: For Unstruct Data
SQL with Pig
DWH with Hive
Scribe, Chukwa For Weblog
Using Mahout
Zookeeper Avro
Storm: Hadoop Realtime
Rhadoop, RHIPE
rmr
Cassandra
MongoDB, Neo4j
Name & Data Nodes
Setup Hadoop (IBM / Cloudera / HortonWorks)
Data Replication Principles
HDFS
Hadoop Components
Map Reduce Fundamentals
MS Excel w/ Analysis ToolPak
Java, Python
R, R-Studio, Rattle
Weka, Knime, RapidMiner
Hadoop Dist of Choice
Spark, Storm
Flume, Scibe, Chukwa
Nutch, Talend, Scraperwiki
Webscraper, Flume, Sqoop
tm, RWeka, NLTK
RHIPE
D3.js, ggplot2, Shiny
IBM Languageware
Cassandra, MongoDB
Author: Swami Chandrasekaran