Third-year ISAE-SUPAERO engineering students
Research Area: Neuro & AI
December, 2020

Neuro & AI: Methods and Tools for Neuroergonomics

Introduction to Machine Learning

Nicolas Drougard[1]

[1]ISAE-SUPAERO DCAS, Toulouse, FRANCE

nicolas.drougard@isae-supaero.fr

Machine Learning in this third module (M3) of the NeuroAI domain

- 15 hours spread over 5 days: 7/01, 8/01, 14/01, 15/01 and 21/01.
- The main points:
  - Supervised Learning,
  - Preprocessing and dimensionality reduction,
  - Unsupervised Learning,
  - Theoretical Results,
  - Machine Learning in practice and for BCIs.

Outline

References

- [F.C18] F.Chollet, *Deep learning with python*, MANNING, 2018, download link **here**.
- [GS20] X. Gendre and F. Simatos, *Probabilités et statistique*, Tronc Commun Scientifique 1A, Formation Ingénieur ISAE-SUPAERO, 2020.
- [HMS20] G. Haine, D. Matignon, and M. Salaün, *Mathématiques déterministes*, Tronc Commun Scientifique 1A, Formation Ingénieur ISAE-SUPAERO, 2020.
- [HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Springer Science & Business Media, 2009, download link **here**.

Outline

What is Artificial Intelligence?

Recent successes of AI applications

- ATTOL project with fully autonomous flight tests.

- Autonomous vehicles (*cf.* class of Caroline Chanel).

Recent successes of AI applications

- Deep fakes, filters.

- Artificially generated pictures `https://thispersondoesnotexist.com` (Generative Adversarial Networks, GANs).

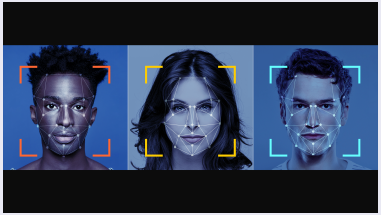
Recent successes of AI applications

- Artificially generated artworks.

- Machine translation: google translate, deepl (based on linguee database).

Recent successes of AI applications

- Facial recognition, assessment of emotions in facial expressions.



- Personnal assistant (*e.g.* siri, alexia, google home), music recognition (*e.g.* shazam).

Recent successes of AI applications

- Alphago

- recommandation systems, smart picture management, chatbots, fraud detection, speech synthesis, etc.

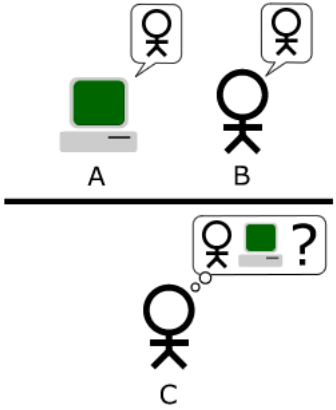
Artificial Intelligence, Machine Learning and Big Data?

Attempt to define "Artificial Intelligence"

A research domain, or a set of methods, tools, formalisms, algorithms and theories, with the aim of making machines able to realize intelligent tasks generally performed by animals and humans.

- Example of **intelligent tasks generally performed by animals and humans**: using language, recognizing objects, people, emotions, playing games, learning from experience, deducing facts, creating art, piloting vehicles, etc.
- When allowing algorithms/machines to do these tasks, *i.e.* to act, we talk about **artificial agents**.
- Example of **automation tasks that are not considered to be AI**: compute the solution of a complex equation, a random or very large number, give a precise quantitative value to make a system stable or make a trajectory optimal using classical mechanics, ... not naturally performed by human/animals!

- **Turing test**: test of a machine's ability to exhibit intelligent behaviour equivalent to that of a human.



A field constantly evolving

- **In the 1950s**, *mathematical functions* $\xrightarrow{transformation}$ *computer-executable instructions* was artificial intelligence.
  Today it is just a simple functionality of compilers.
- Same remark on *route planners*, that use AI planning algorithm like A^* (*cf.* class of Caroline Chanel).

Artificial Intelligence, Machine Learning and Big Data?

Different visions

- AI is somewhere in the intersection between **Mathematics and Computer Science**.
- Note that until now, artificial agents are more or less complex algorithms, basically **optimizing criteria**.
- **Video Games**: AI can used to refer to a Non Player Characters (NPC).
- **Science Fiction**: AI can be used to refer to a fully autonomous virtual super-human able to understand and compute almost everything, to finally rule us or save the world.
- In **french**: Intelligence is often linked to a human quality, and even to conciousness, *i.e.* human beings more intelligent than animals.
- In **english**: Business Intelligence / Central Intelligence Agency (CIA). Intelligence is related to the tasks (intelligent tasks).
- In **recent years**: *AI* sometimes refers to *Neural Networks* and *Deep Learning*.

Short history

- Naturally **starts with computers**, roughly in the 1950's,
  e.g. Dynamic Programming (R.Bellman) or Turing test.
- Then a lot of effort is made in the direction of **automatic reasoning** with expert system managing knowledge, ontologies, set of rules, infering facts, using modal logic etc.
  Most impressive contributions: proofs of theorems, natural language processing (NLP) and game opponents.
- Referred to **Symbolic AI**, or GOFAI (good old-fashioned artificial intelligence).
  e.g. verification of accuracy & reliability of online aircraft control software (Airbus).
- Another AI field also emerges: **Machine learning**, or data-driven AI, coming from Statistics, Computer Science, Connexionnism (neural networks to model brains).

Machine Learning thanks to data

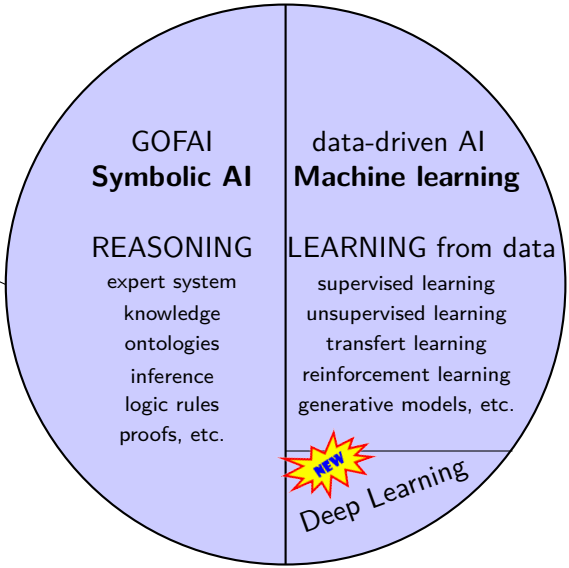
- Machine Learning (ML) is boosted by access to **more and more databases** (Internet, Big Data) and by the increasing available **computational power**.
- Trendiest framework in ML is **Deep Learning** (Neural Networks with a lot of layers).
- Deep Learning: hard to tune, hard to derive theoretical guaranties and needs a lot of data. Researchers in this field were perceived as alchemists until **recently** (around 2010).
- As a friend of yours put it so well: "*If you are not paying for it, you're not the customer; you're the product being sold.*".

- Data allows to create models, to better understand and control future.

Artificial Intelligence, Machine Learning and Big Data?

Representation of the research field of

Artificial Intelligence



Remark: the previous AI application examples were often a mix of GOFAI & Machine Learning.

Some remarks from AI leaders

- Yann LeCun: *Machines learn slower than we do.*

- Luc Julia: *It is not the intelligence that characterizes today's AI systems, but their ability to recognize through machine learning.*

Next challenges of AI

- Increase **trust** in AI: explanability, certification,
- **Speed** up learning,
- **Re-unite** GOFAI & Machine Learning,
- More generally, and as usual with AI, **reduce** the need for **human intervention**, *i.e.* increase the autonomy of artificial agents.

Machine Learning

- Learning from data: dataset $d_n = \{x_i\}_{i=1}^n \in \mathcal{X}^n$, with $dim(\mathcal{X}) = d$ (usually $\mathcal{X} = \mathbb{R}^d$). An element of d_n, $x_i \in \mathcal{X}$, is called a **sample**.

  Matrix representation of the dataset: $d_n = \mathbb{X} = \begin{pmatrix} x_{1,1} & \dots & x_{1,d} \\ x_{2,1} & \dots & x_{2,d} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,d} \end{pmatrix}$.

- **Supervised learning**
  - Settings: each sample x_i is associated with a target $y_i \in \mathcal{Y}$.
  - Goal: compute a function (predictor) $c : \mathcal{X} \to \mathcal{Y}$ that predicts the target given a sample. Sample values are called **features**, and $\mathcal{X}$ the feature space.
  - Two main problems:
    classification ($\mathcal{Y}$ is finite, y_i are called **labels**),
    and **regression** ($\mathcal{Y} = \mathbb{R}$).

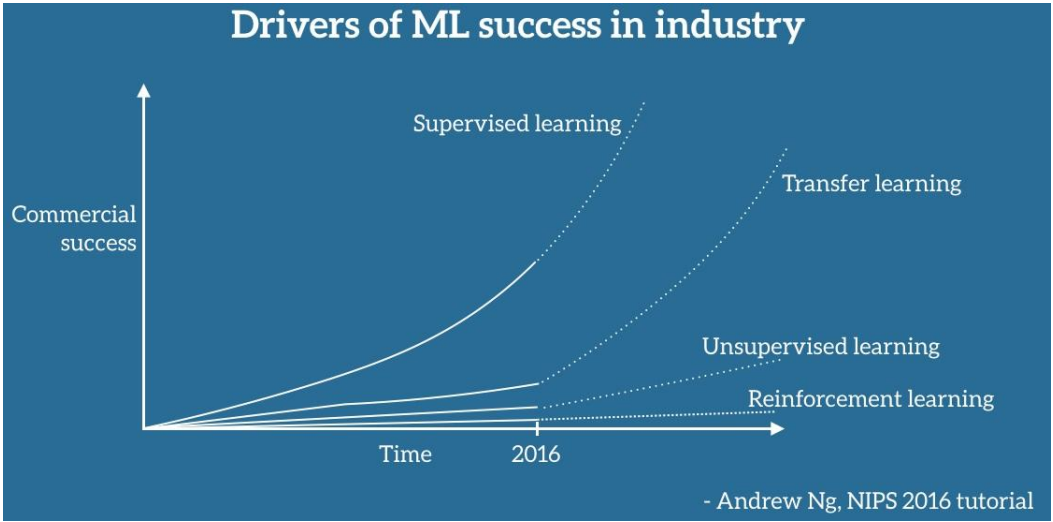
Machine Learning

- **Unsupervised Learning**
  - Settings: samples are not associated with any target.
  - Goal: anomaly/novelty detection, clustering, discovering structures in the data d_n.
- **Transfert Learning**
  - Settings: two different domains, *i.e.* two different couples (feature space, target space): source domain $(\mathcal{X}_S, \mathcal{Y}_S)$ and target domain $(\mathcal{X}_T, \mathcal{Y}_T)$.
  - Goal: Use a dataset from source domain to help making predictions in the target domain.
- **Semi-supervised Learning**
  - Settings: the samples are not always associated with target.
  - Goal: predict the target given a sample.
- **Self-supervised Learning**
  - Settings: the samples are associated with targets predicted by another predictor.
  - Goal: predict the target given a sample.

Machine Learning

- **Reinforcement Learning**
  - Settings: environment simulator returning an observation $s_t \in \mathcal{S}$, and a reward $r_t \in \mathbb{R}$ for each action $a_t \in \mathcal{A}$ sent to it.
  - Goal: compute an optimal strategy $\pi : \mathcal{S} \to \mathcal{A}$ by interacting with the simulator, *i.e.* maximizing the expectation of the sum of future rewards, *cf.* class of Planning (C.Chanel) and Reinforcement Learning (N.Drougard).
- **Generative Models**
  - Settings: samples are not associated with any target.
  - Goal: generation of samples that could be in the dataset d_n *i.e.* that look like samples in the dataset.

In any of these Machine Learning problems, the computations on data, leading to desired outputs, are called **training**.


Drivers of ML success in industry
Supervised learning
Transfer learning
Commercial success
Unsupervised learning
Reinforcement learning
Time
2016
- Andrew Ng, NIPS 2016 tutorial

The Machine Learning library

- **Scikit-learn** (sklearn) is one of the most popular machine learning libraries.
- Written in python, designed to interoperate with NumPy and SciPy, first release in 2010.
- Sklearn defined the formalism
  - `model.fit(data)` to train on a dataset,
  - `model.predict(data)` to do prediction on new data,

  used by most machine learning libraries (*e.g.* Deep Learning libraries pytorch, tensorflow, keras, etc.).

Outline

Random variable

- Random variable $X \in \mathcal{X}$, uncertainty model for each sample $x_i \in \mathcal{X}$.
- Distribution of the random variable defines the probability $\mathbb{P}(X \in A)$ for $A \subseteq \mathcal{X}$.
- Independant variables: X_1 and X_2 such that $\forall A, B \subseteq \mathcal{X}$,

$$\mathbb{P}\Big((X_1, X_2) \in A \times B\Big) = \mathbb{P}(X_1 \in A)\,\mathbb{P}(X_2 \in B)\,.$$

Expectation and variance

- Expectation $\mathbb{E}$ of a real-valued random variable X:
  - $\mathbb{E}[X] = \int_{\mathcal{X}} x f_X(x) dx$ if X is a continuous random variable,
  - $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \mathbb{P}(X = x)$ if X is a discrete variable.
- $\mathbb{E}$ is linear, *i.e.* $\mathbb{E}[aX_1 + bX_2] = a\mathbb{E}[X_1] + b\mathbb{E}[X_2]$, $\forall a, b \in \mathbb{R}$, X_1, X_2 random variables.
- Variance of a real-valued random variable X: $Var(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$.

Dataset distribution and likelihood

- To reason on the dataset before observation of it,
  we also consider the random dataset $D_n = \{ X_i \}_{i=1}^n$.
- If the samples are independent,
  - $f_{D_n}(d_n) = \prod_{i=1}^n f_{X_i}(x_i)$ if X continuous,
  - $p_{D_n}(d_n) = \prod_{i=1}^n p_{X_i}(x_i)$ if X discrete.
- When we assume that the variable distribution depends on parameters $\theta \in \Theta$
  ($e.g.$ $\theta = (\mu, \sigma)$ for $\mathcal{N}(\mu, \sigma^2)$, or $\theta = p$ for $Ber(p)$)
  the Likelihood is the random variable $\mathcal{L}(\theta, X) = f_\theta(X)$, or $\mathcal{L}(\theta, X) = p_\theta(X)$ if discrete).
- $\mathcal{L}(\theta, D_n) = \prod_{i=1}^n \mathcal{L}(\theta, X_i)$ if variables X_i are independent.

Notations

- Random variable $X \sim P_X$ with values in $\mathcal{X}$.
- Random dataset $D_n = \{ X_i \}_{i=1}^n$.
- Independant $X_i \Rightarrow D_n \sim P_X^n$.
- Consider a real random variable $\mathcal{X} = \mathbb{R}$.
- Cumulative Distribution Function (CDF): $F_X(t) = \mathbb{P}(X \leqslant t)$.
- Empirical CDF: $F_{X,n}(t) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{\{ X_i \leqslant t \}}$.
- $\forall t \in \mathbb{R}$, $F_{X,n}(t) \xrightarrow[n \to +\infty]{a.s.} F_X(t)$ (Strong Law of Large Numbers SLLN, see Theorem 3.1 [GS20]).
- Quantile function: $F_X^{-1}(s) = \inf \{ t \in \mathbb{R} \mid F(t) \geqslant s \}$.

Outline

Notations

- **Input (features)**: X (random) or x (realization).
- **Output (targets)**: Y (random) or y (realization).
- **Random dataset**: $D_n = (X_i, Y_i)_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$
- usually, features $\mathcal{X} = \mathbb{R}^d$
- target for **classification**: $y \in \mathcal{Y} = \{y_i\}_{i=1}^m$. Binary classification: $\{y_+, y_-\}$ $(m = 2)$.
- target for **regression**: $y \in \mathcal{Y} = \mathbb{R}$.
- **Classifier/regressor**: function $c : \mathcal{X} \to \mathcal{Y}$.
- **Dataset**, samples, realization of D_n: $d_n = (x_i, y_i)_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$.
- **Prediction**: $\hat{y}_i = c(x_i)$.

Scores/Loss

How to evaluate a Classification/Regression?
"Good classifiers" have a high score and low loss/error.

- Classification accuracy $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{\hat{y}_i = y_i\}} = 1 -$ error rate $\approx \mathbb{E}\left[\mathbb{1}_{\{Y = c(X)\}}\right]$
  - sklearn: `metrics.accuracy_score(y_true,y_pred)`

  & error rate = zero-one classification loss
  - sklearn: `metrics.zero_one_loss(y_true, y_pred)`
- Regression: mean squared error $\frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$
  - sklearn: `metrics.mean_squared_error(y_true,y_pred)`

Some famous loss functions

- Hinge loss (used for SVM algorithm): $L_{Hinge}(y, v) = \max\left(0, 1 - sgn(y) \times v(x)\right)$ with $sgn(y_+) = 1$, $sgn(y_-) = -1$, and the **decision function** $v(x) \in \mathbb{R}$ defining prediction by $\hat{y} = c(x) = \begin{cases} y_+ & v(x) > 0 \\ y_- & \text{otherwise .} \end{cases}$

  A missclassification means $v(x)$ and $sgn(y)$ have opposite signs $\Rightarrow L_{Hinge}(y, v) \geqslant 1$: $\frac{1}{n}\sum_{i=1}^{n} L_{Hinge}(y_i, v(x_i)) \approx \mathbb{E}\left[L_{Hinge}(Y, v(X))\right]$ is a upper bound of the accuracy.
  - sklearn: `metrics.hinge_loss(y_true, pred_decision)`
- Log (logistic, or cross-entropy) loss (used for logistic regression and neural networks): $L_{log}(y, \eta) = -\Big(y \log(\eta(x)) + (1 - y) \log(1 - \eta(x))\Big)$ with $y_+ = 1$, $y_- = -1$, and the **posterior probability function** $\eta(x)$ defining prediction by $\hat{y} = c(x) = \begin{cases} y_+ & \eta(x) > \frac{1}{2} \\ y_- & \text{otherwise .} \end{cases}$ Opposite of the log-likelihood of the Bernoulli distribution with parameter $\eta(x) = \mathbb{P}(Y = y_+ \mid X = x)$ we want η to maximize.
  - sklearn: `metrics.log_loss(y_true, y_pred)`

Binary classification (2-class)

- $\mathcal{Y} = \{y_+, y_-\}$
- Samples with $y = y_+$ are called positive: $Pos = \sum_{i=1}^{n} \mathbb{1}_{\{y_i = y_A\}}$.
- Samples with $y = y_-$ are called negative: $Neg = \sum_{i=1}^{n} \mathbb{1}_{\{y_i = y_B\}} = n - Pos$.

The accuracy of c that returns always y_+ (resp. y_-) is $\frac{Pos}{n}$ (resp. $\frac{Neg}{n}$).
Unbalanced dataset with $Pos > Neg$ (resp. $Pos < Neg$), $\Rightarrow c$'s accuracy can be good!!
E.g. if $Pos = 9 \times Neg$, the accuracy of $c(x) \overset{const.}{=} y_+$ is $\frac{Pos}{10 \times Neg} = \frac{9}{10} = 90\%$...
The balanced accuracy take this issue into account:

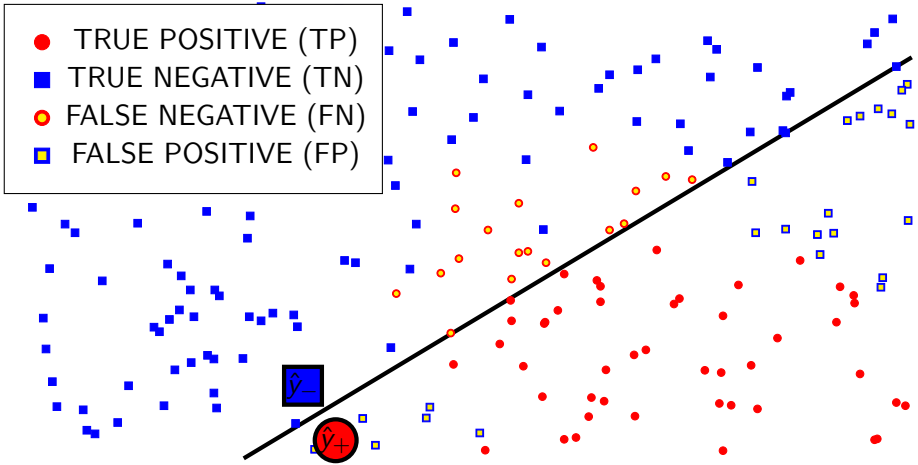
Balanced accuracy

$$\frac{1}{2}\left(\frac{1}{Pos}\sum_{i=1}^{n}\mathbb{1}_{\{\hat{y}_i = y_i = y_+\}} + \frac{1}{Neg}\sum_{i=1}^{n}\mathbb{1}_{\{\hat{y}_i = y_i = y_-\}}\right) = \frac{1}{2}\left(\frac{TP}{Pos} + \frac{TN}{Neg}\right) = \frac{1}{2}(TPR + TNR)$$

with $TP = \sum_{i=1}^{n}\mathbb{1}_{\{\hat{y}_i = y_i = y_+\}}$ the number of "True Positive", (TPR for "True Positive Rate"),
$TN = \sum_{i=1}^{n}\mathbb{1}_{\{\hat{y}_i = y_i = y_-\}}$ the number of "True Negative", (TNR for "True Negative Rate").

- sklearn: `metrics.balanced_accuracy_score(y_true, y_pred)`

Binary classification (2-class)

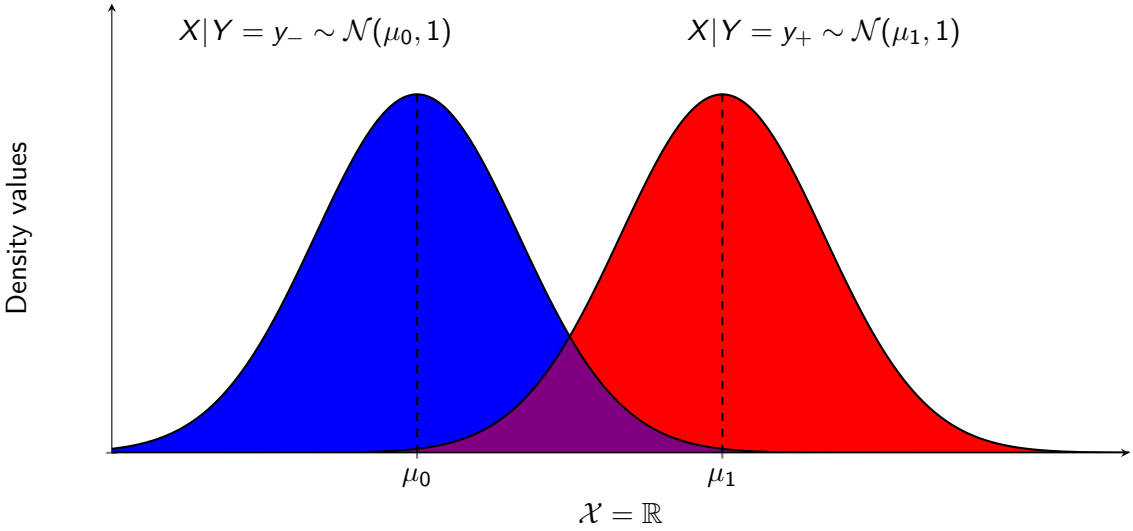


Terminology: conditioning by reality y (samples)

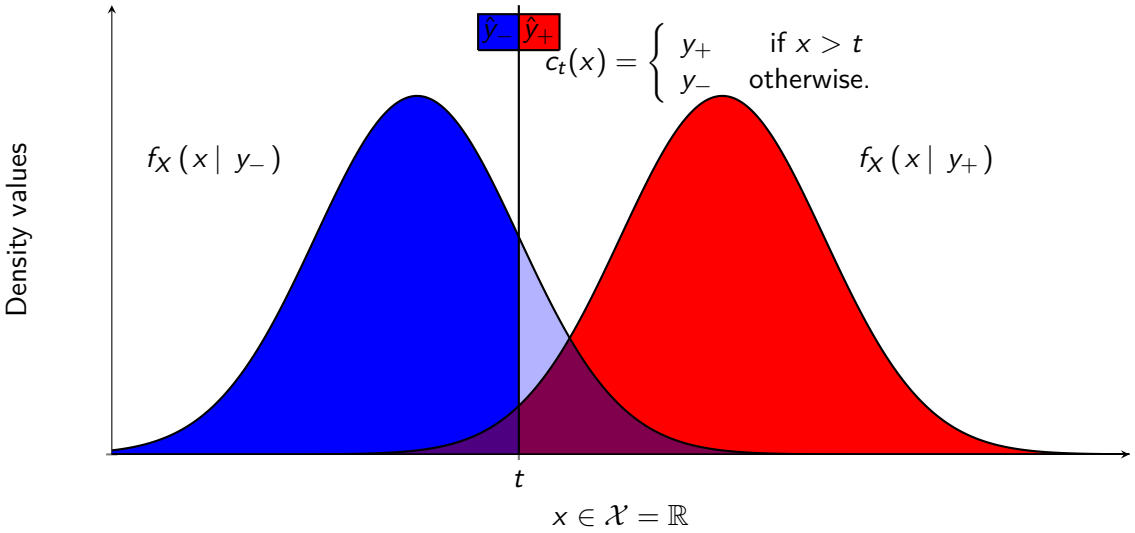
Proportions among

NEGATIVE samples y_- ■	**POSITIVE samples** y_+ ●
$TNR =$ True Negative Rate	$TPR =$ True Positive Rate
$=$ **specificity/selectivity**	$=$ **sensitivity/recall/hit rate**
$= \frac{TN}{Neg} = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{y_i = \hat{y}_i = y_-\}}}{\sum_{i=1}^{n} \mathbb{1}_{\{y_i = y_-\}}}$	$= \frac{TP}{Pos} = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{y_i = \hat{y}_i = y_+\}}}{\sum_{i=1}^{n} \mathbb{1}_{\{y_i = y_+\}}}$
$= 1 - \frac{FP}{Neg} = 1 - FPR$	$= 1 - \frac{FN}{Pos} = 1 - FNR$
$= 1 -$ False Pos. Rate (or fall-out)	$= 1 -$ False Neg. Rate (or miss rate)
Links with statistical tests	
H_0	H_1
$\approx p(\hat{y}_- \mid y_-) \overset{def.}{=} \mathbb{P}(Y = y_- \mid c(X) = y_-)$	$\approx p(\hat{y}_+ \mid y_+) = \gamma$ (power)
$= 1 - p(\hat{y}_+ \mid y_-)$	$= 1 - p(\hat{y}_- \mid y_+)$
$= 1 - \alpha$ ($\alpha =$ type I error)	$= 1 - \beta$ ($\beta =$ type II error)

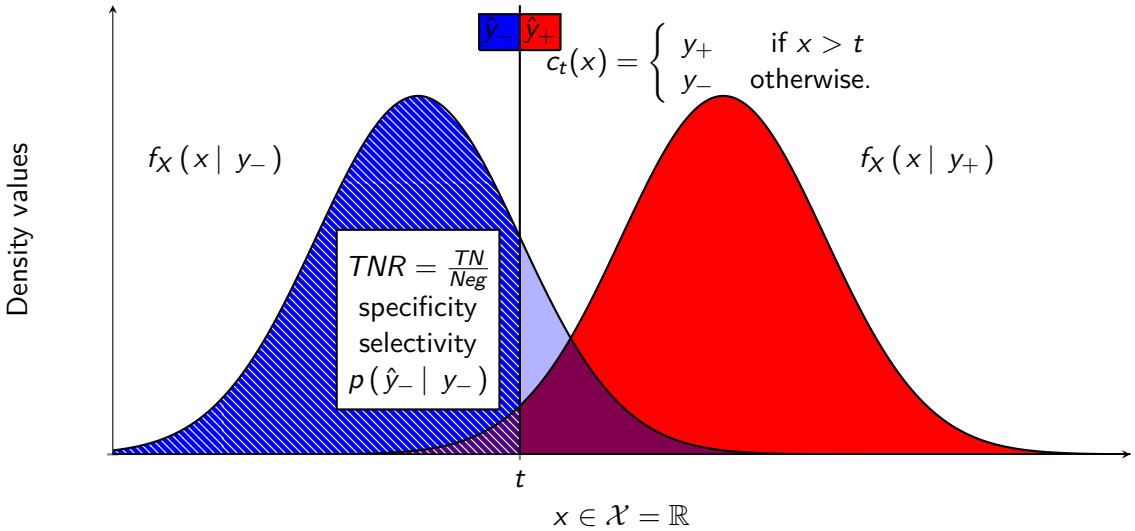
Visualization with densities



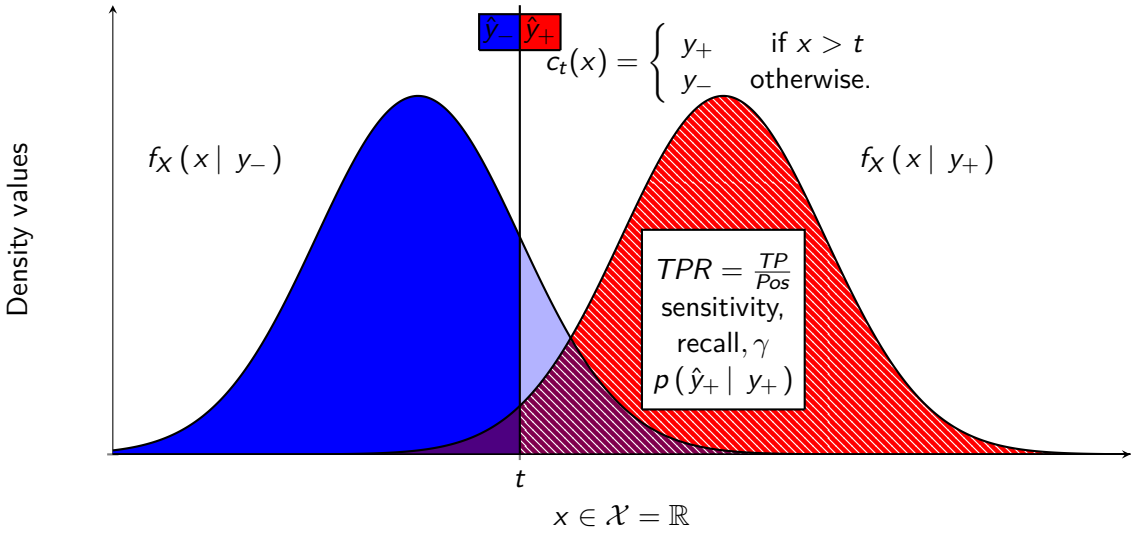
Visualization with densities



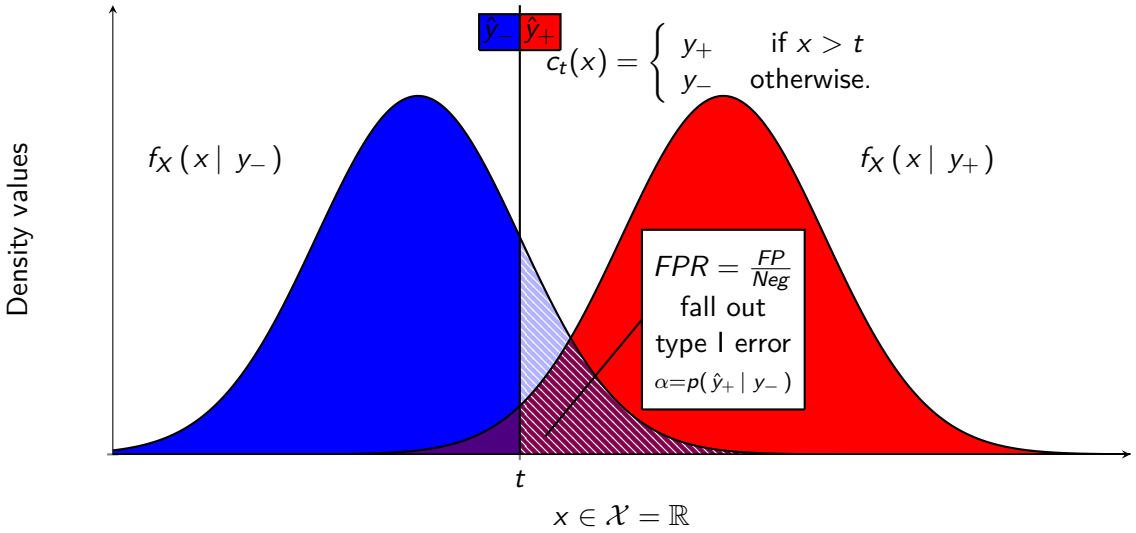
Visualization with densities



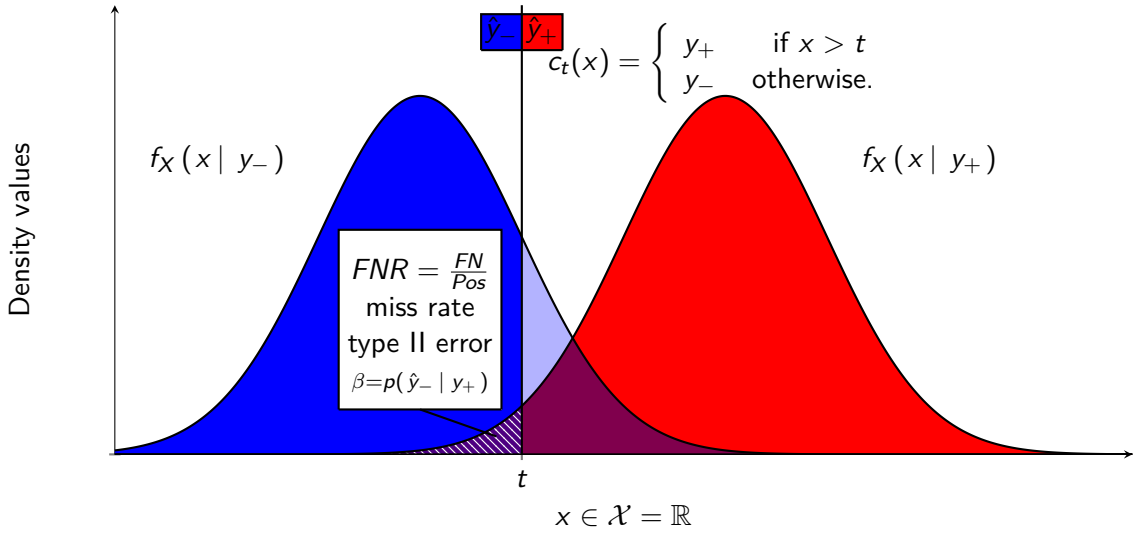
Visualization with densities



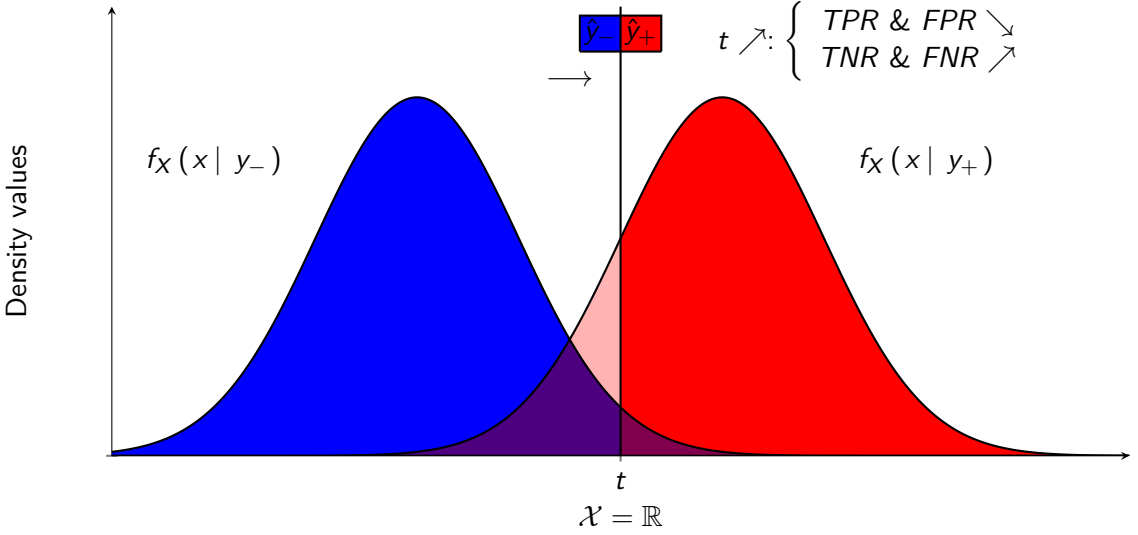
Visualization with densities



Visualization with densities



Visualization with densities



Binary classification (2-class)

Consider a classifier $c_t(x) = \begin{cases} y_+ & \text{if } s(x) > t \\ y_- & \text{otherwise.} \end{cases}$ with $s : \mathcal{X} \to \mathbb{R}$ (called score) and $t \in \mathbb{R}$.

For instance, $s(x)$ can be an approximation of $\mathbb{P}(Y = y_+ \mid X = x) = \eta(x)$.

Note that when $t \nearrow$,

- $TP(t) \searrow$, $\lim_{t \to -\infty} TP(t) = Pos$, $\lim_{t \to +\infty} TP(t) = 0$.
- $FP(t) \nearrow$, $\lim_{t \to -\infty} FP(t) = Neg$, $\lim_{t \to +\infty} FP(t) = 0$.

ROC curve

Receiver operating characteristic (ROC): $\Big(FPR(t), TPR(t)\Big)$

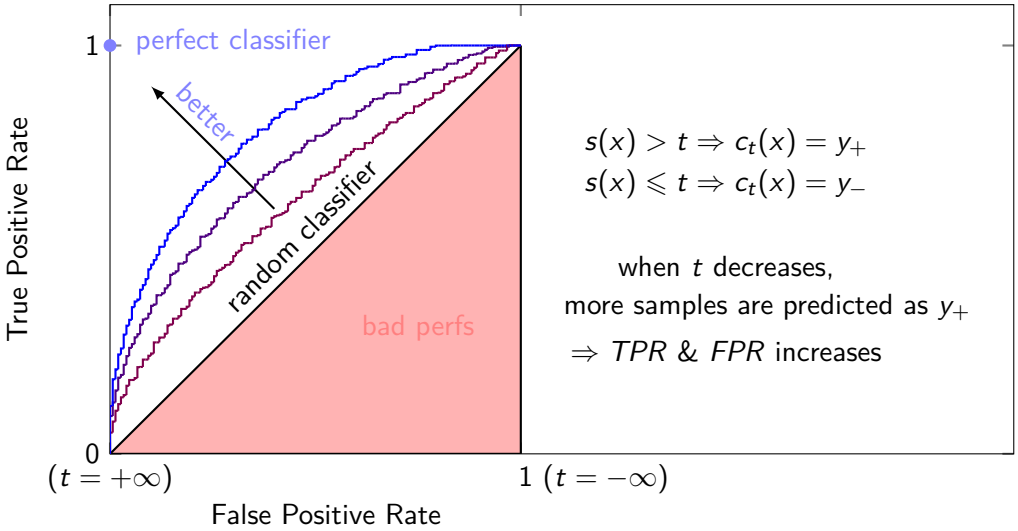
- sklearn: `metrics.roc_curve(y_true, y_score)`

Resulting performance metric: Area Under the ROC Curve (AUC).

- sklearn: `metrics.roc_auc_score(y_true, y_score)`

Evaluation of a prediction function.

Receiver operating characteristic (ROC): $\Big(FPR(t), TPR(t)\Big)$



$s(x) > t \Rightarrow c_t(x) = y_+$
$s(x) \leqslant t \Rightarrow c_t(x) = y_-$

when t decreases,
more samples are predicted as y_+
$\Rightarrow$ *TPR* & *FPR* increases

Terminology: conditioning by prediction $\hat{y}$

Proportions among

NEGATIVE predictions $\hat{y}_-$ (above line)	**POSITIVE predictions** $\hat{y}_+$ (below line)
$NPV =$ Negative Predictive Value	$PPV =$ Positive Predictive Value
$= \dfrac{TN}{TN+FN} = \dfrac{\sum_{i=1}^{n} \mathbb{1}_{\{y_i=\hat{y}_i=y_-\}}}{\sum_{i=1}^{n} \mathbb{1}_{\{\hat{y}_i=y_-\}}}$	$=$ **precision** $= \dfrac{TP}{TP+FP} = \dfrac{\sum_{i=1}^{n} \mathbb{1}_{\{y_i=\hat{y}_i=y_+\}}}{\sum_{i=1}^{n} \mathbb{1}_{\{\hat{y}_i=y_+\}}}$
$\approx p(y_- \mid \hat{y}_-)$	$\approx p(y_+ \mid \hat{y}_+)$
$\overset{def.}{=} \mathbb{P}(c(X) = y_- \mid Y = y_-)$	$\overset{def.}{=} \mathbb{P}(c(X) = y_+ \mid Y = y_+)$
$= 1 - \dfrac{FN}{TN+FN}$	$= 1 - \dfrac{FP}{TP+FP}$
$= 1 - FOR$	$= 1 - FDR$
(for "**False Omission Rate**")	(for "**False Discovery Rate**")

- sklearn: `metrics.precision_score(y_true, y_pred)`

Binary classification (2-class)

Consider again $c_t(x) = \begin{cases} y_+ & \text{if } s(x) > t \\ y_- & \text{otherwise.} \end{cases}$ with $s : \mathcal{X} \to \mathbb{R}$ (called score) and $t \in \mathbb{R}$.

Note that when $t \searrow$, positive predictions $TP(t) + FP(t) \nearrow$.

The precision $PPV(t) = \frac{TP}{TP+FP}$ $\begin{cases} \bullet & \text{increases when } TP(t) \nearrow \\ \bullet & \text{decreases when } FP(t) \nearrow . \end{cases}$

Recall that when $t \searrow$ $TP(t) \nearrow$, $\lim_{t \to -\infty} TP(t) = Pos$, $\lim_{t \to +\infty} TP(t) = 0$.

Precision-Recall (PR) curve

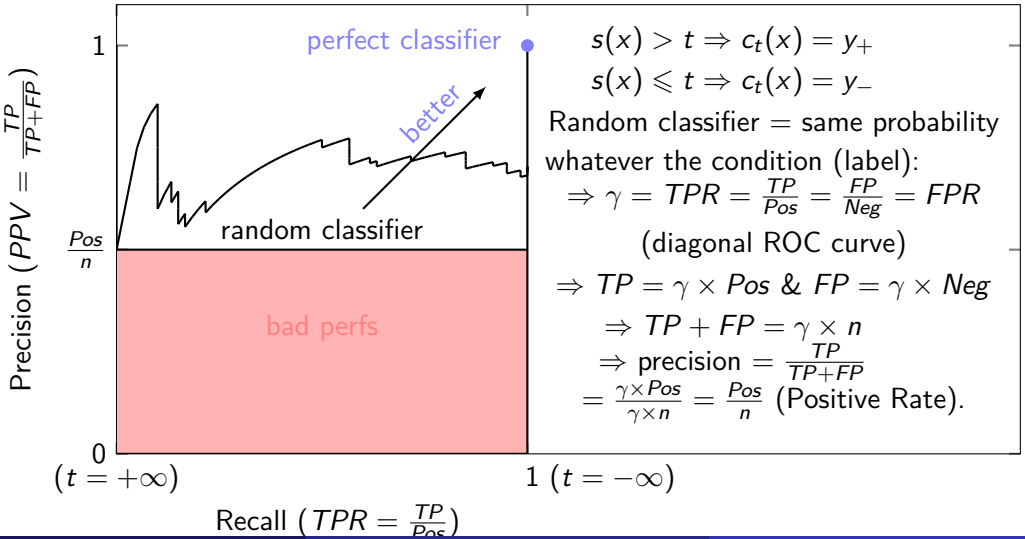
$t \mapsto \Big(TPR(t), PPV(t)\Big)$. The y-axis of the ROC curve is now the x-axis.

The y-axis becomes the precision $PPV = \frac{TP}{TP+FP}$.

- sklearn: `metrics.precision_recall_curve(y_true, probas_pred)`
- sklearn: `metrics.recall_score(y_true, y_pred)`

Evaluation of a prediction function.

Precision Recall (PR): $\Big(TPR(t), PPV(t)\Big)$



$s(x) > t \Rightarrow c_t(x) = y_+$

$s(x) \leqslant t \Rightarrow c_t(x) = y_-$

Random classifier = same probability whatever the condition (label):

$\Rightarrow \gamma = TPR = \frac{TP}{Pos} = \frac{FP}{Neg} = FPR$

(diagonal ROC curve)

$\Rightarrow TP = \gamma \times Pos \;\&\; FP = \gamma \times Neg$

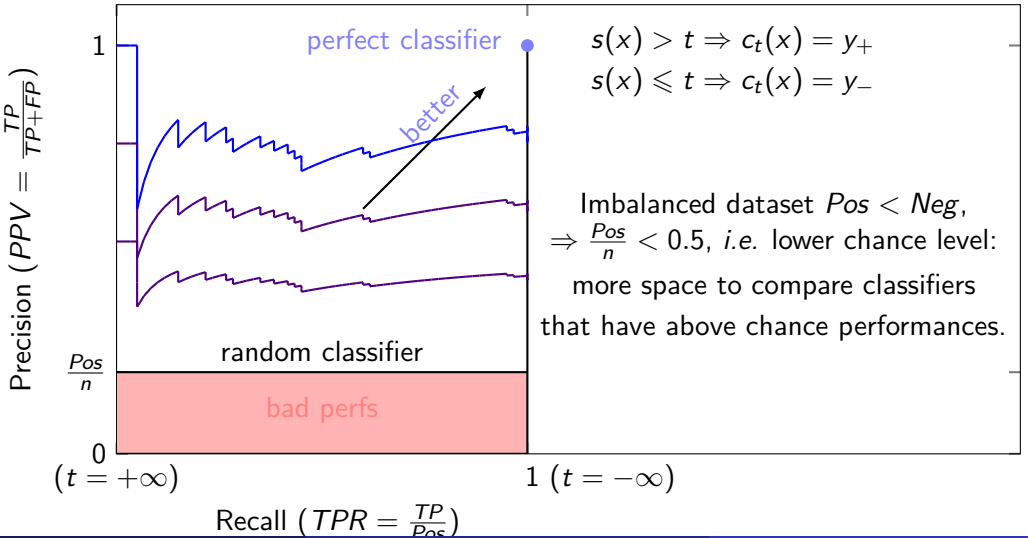
$\Rightarrow TP + FP = \gamma \times n$

$\Rightarrow$ precision $= \frac{TP}{TP+FP}$

$= \frac{\gamma \times Pos}{\gamma \times n} = \frac{Pos}{n}$ (Positive Rate).

Evaluation of a prediction function.

Precision Recall (PR): $\Big(TPR(t), PPV(t)\Big)$



$s(x) > t \Rightarrow c_t(x) = y_+$

$s(x) \leqslant t \Rightarrow c_t(x) = y_-$

Imbalanced dataset $Pos < Neg$,
$\Rightarrow \frac{Pos}{n} < 0.5$, *i.e.* lower chance level:
more space to compare classifiers
that have above chance performances.

Resulting performance metrics

- Area Under the Curve (AUC) just like for the ROC curve:
  - sklearn: `metrics.auc(x,y)`
- Average precision: $AP = \sum_{i=1}^{n} (R_{i+1} - R_i) P_{i+1}$
  with $(R_i)_{i=1}^{n+1}$ the non-decreasing values of Recall $\in \left\{ \frac{i}{Pos} \,\middle|\, i \in \{1, \ldots, Pos\} \right\}$ when $t \searrow$, and P_n the associated Precision values. Same idea as AUC, but less optimistic:
  - sklearn: `metrics.average_precision_score(y_true)`
- F-measure or balanced F-score (F1 score)
  = harmonic mean of precision and recall: $2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
  - sklearn: `metrics.f1_score(y_true, y_pred)`
  - sklearn: `metrics.precision_recall_fscore_support`
- statistics on the accuracy distribution, and many other metrics:
  - sklearn: `Classification metrics`

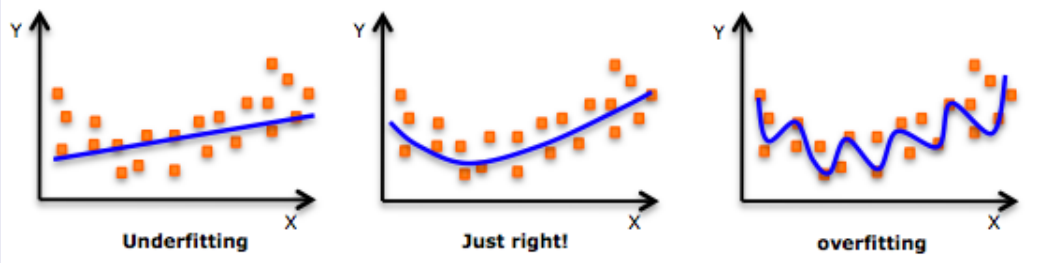
Multiclass classification (m-classes)

- **Confusion matrix** $\begin{pmatrix} TPR & FPR \\ FNR & TNR \end{pmatrix}$ with T/F = True/False, P/N = Positive/Negative and R = Rate.
- Generalization for multi-class labels $\mathcal{Y} = \{\lambda_1, \ldots, \lambda_m\}$: confusion matrix $\forall (i,j) \in \mathcal{Y}$, $M_{i,j} = \hat{p}\left(\hat{y} = \lambda_i \mid y = \lambda_j\right) = \frac{\sum_{k=1}^{n} \mathbb{1}_{\left\{\hat{y}_k = \lambda_i, y_k = \lambda_j\right\}}}{\sum_{k=1}^{n} \mathbb{1}_{\left\{y_k = \lambda_j\right\}}}$.
  - sklearn: `metrics.confusion_matrix(y_true, y_pred)`

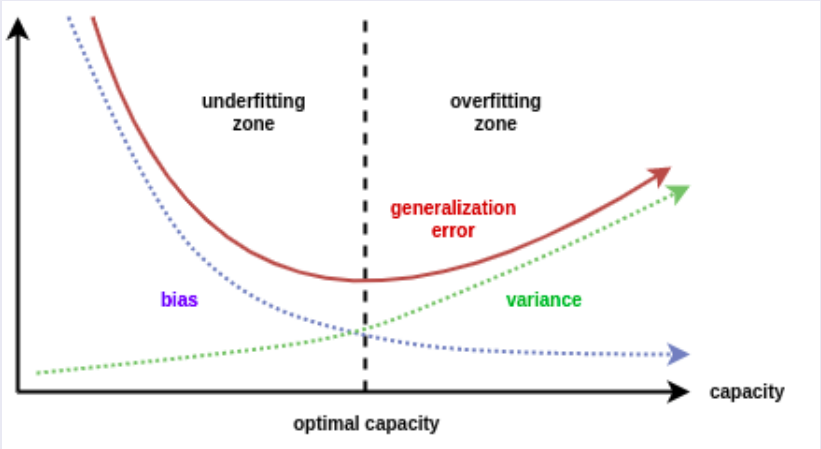
Outline

Overfitting/Underfitting: classification

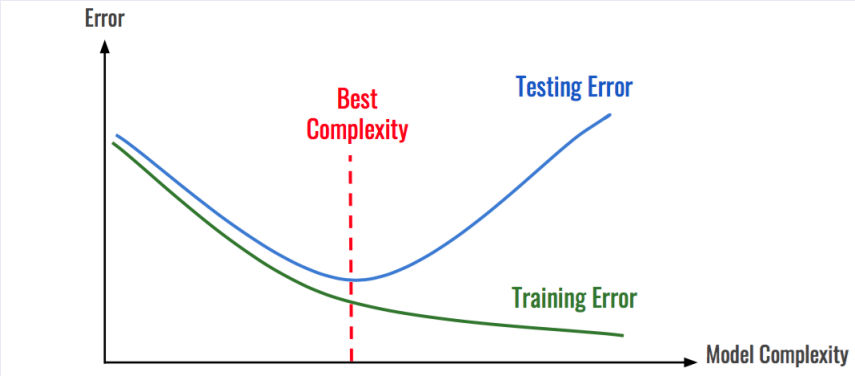
Overfitting/Underfitting: regression



Overfitting/Underfitting



Overfitting/Underfitting



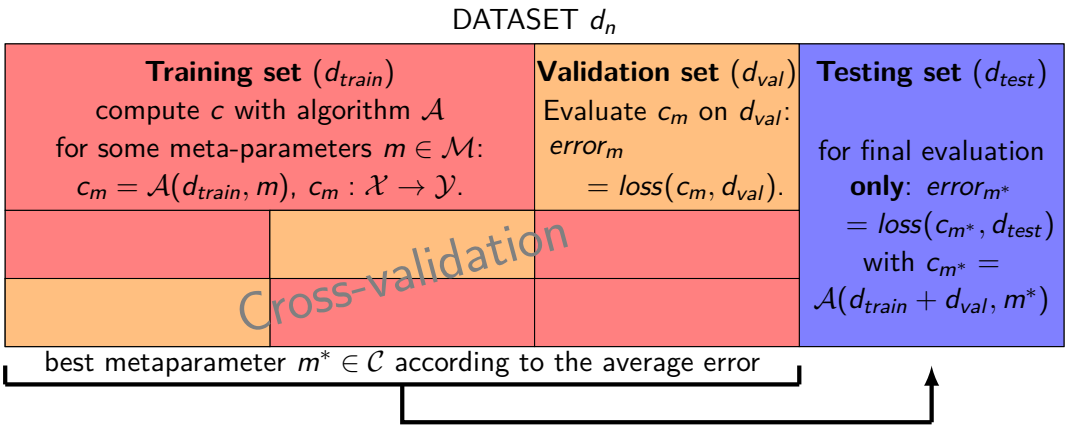
Overfitting/Underfitting

Overfitting is avoided by

- penalizing the model complextity (typically the number of parameters)
- if training is sequential, stop when the error on the **testing set** increases.

General process

- training (optimization)
- validation (meta optimization, cross-validation)
- testing (final evaluation)



Linear Regression

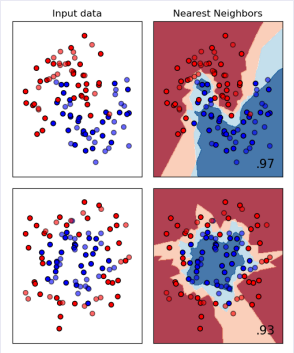
General Linear Model:

$$Y = \mathbb{X}\theta = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)} & \dots & x_1^{(d)} \\ x_2^{(1)} & \dots & x_2^{(d)} \\ \vdots & & \vdots \\ x_n^{(1)} & \dots & x_n^{(d)} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

- the MLE is the linear least square parameters $\widehat{\theta_n} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T Y$.
- Given the target $y \in \mathbb{R}^n$, the regression is $c(x) = x^T\widehat{\theta_n}$ with $\widehat{\theta_n} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T y$, for feature vector $x \in \mathcal{X}$.
  - sklearn: `linear_model.LinearRegression`

k nearest neighbors (k-NN)

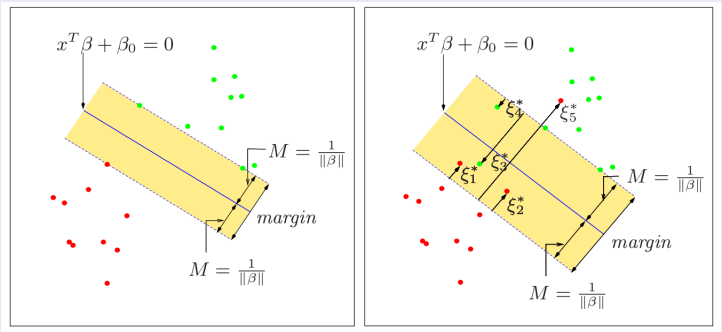
- Let x be a feature vector. We want to predict the associated label y.
- Let $N_k(x)$ be the k closest feature vectors in the dataset
- Prediction: $c(x) = \arg\max_y \#\{y_i \mid x_i \in N_k(x)\}$.



- sklearn: `neighbors.KNeighborsClassifier`

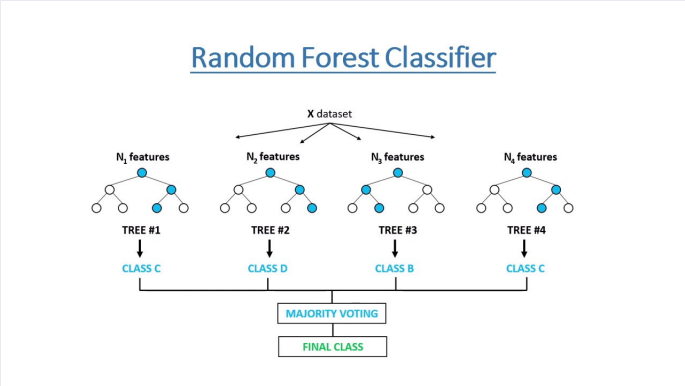
Support Vector Machine (SVM)

- Maximizing the margin



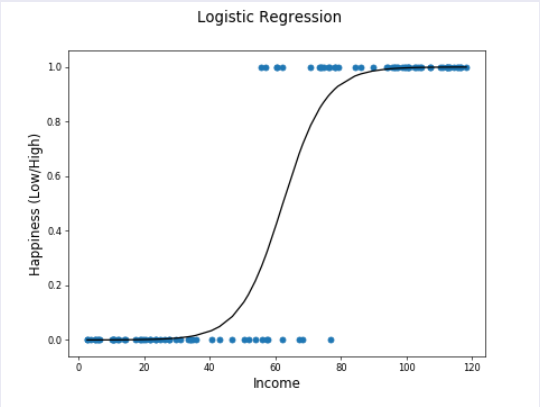
- sklearn: `svm.SVC`

Random Forest (RF)



- sklearn: `ensemble.RandomForestClassifier`

Logistic Regression



- sklearn: `linear_model.LogisticRegression`

Outline

In the past few years, artificial intelligence (AI) has been a subject of intense media hype. Machine learning, deep learning, and AI come up in countless articles, often outside of technology-minded publications. We're promised a future of intelligent chatbots, self-driving cars, and virtual assistants—a future sometimes painted in a grim light and other times as utopian, where human jobs will be scarce and most economic activity will be handled by robots or AI agents. For a future or current practitioner of machine learning, it's important to be able to recognize the signal in the noise so that you can tell world-changing developments from overhyped press releases. [F.C18]

Single layer perceptron a.k.a. *single hidden layer back-propagation network*

inputs/features	$x \in \mathbb{R}^d$	
neurons/hidden layer	$z_i = \sigma(\alpha_{i0} + \alpha_i^T x)$	$\forall i \in \{1, \dots, p\}$
network outputs	$w_j = \beta_{j0} + \beta_j^T z$	$\forall j \in \{1, \dots, m\}$
prediction/outputs	$\widehat{y}_j = \tau_j(w)$	$\forall j \in \{1, \dots, m\}$

with

- $\alpha_{i0} \in \mathbb{R}$ and $\beta_{j0} \in \mathbb{R}$ **biases**,
- $\alpha_i \in \mathbb{R}^d$ and $\beta_j \in \mathbb{R}^p$ **weights**,
- $\sigma : \mathbb{R} \to \mathbb{R}$ non-linear **activation function**,
- and $\tau : \mathbb{R}^m \to \mathbb{R}^m$ the **output function**.

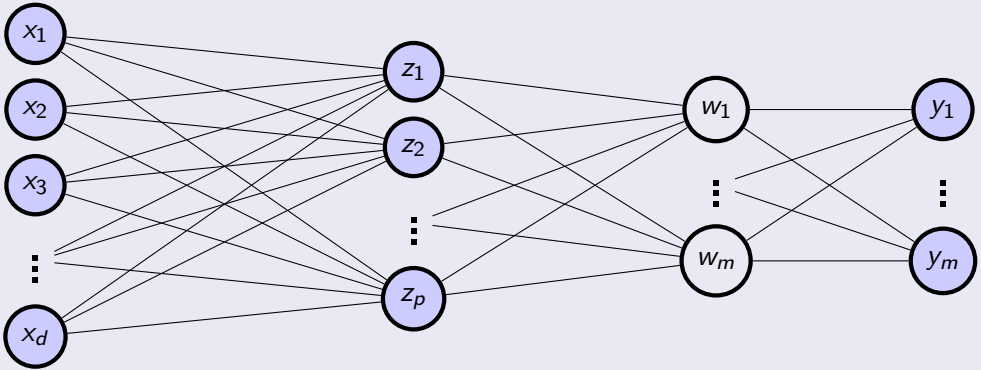
Single layer perceptron

inputs $x \in \mathbb{R}^d$

neurons $z_i = \sigma(\alpha_{i0} + \alpha_i^T x)$

network outputs $w_j = \beta_{j0} + \beta_j^T z$

predictions $\widehat{y}_j = \tau_j(w)$

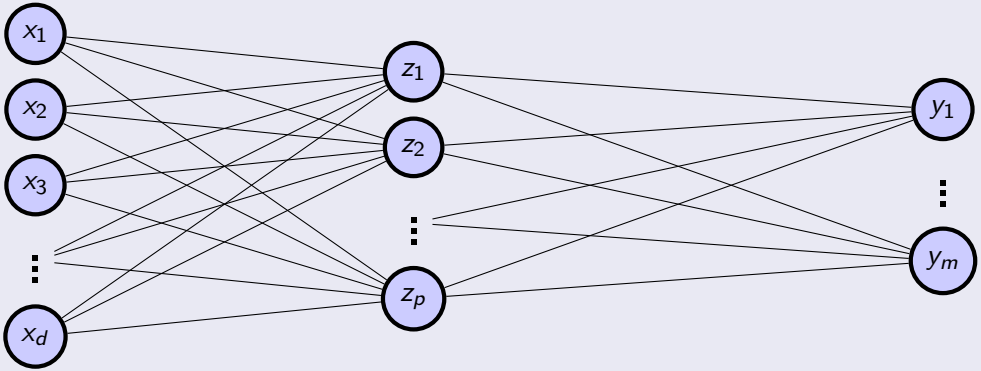


Single layer perceptron

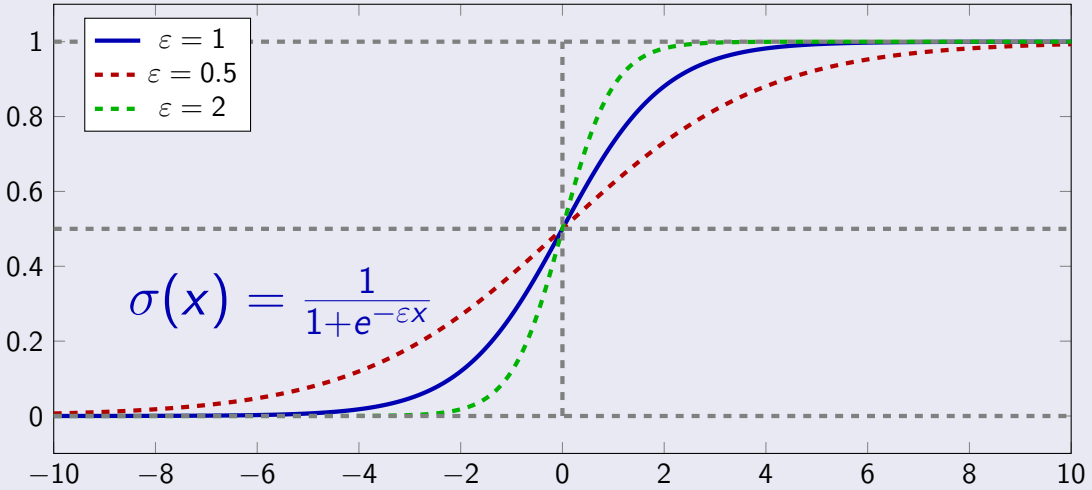
inputs
$x \in \mathbb{R}^d$

neurons
$z_i = \sigma(\alpha_{0,i} + \alpha_i^T x)$

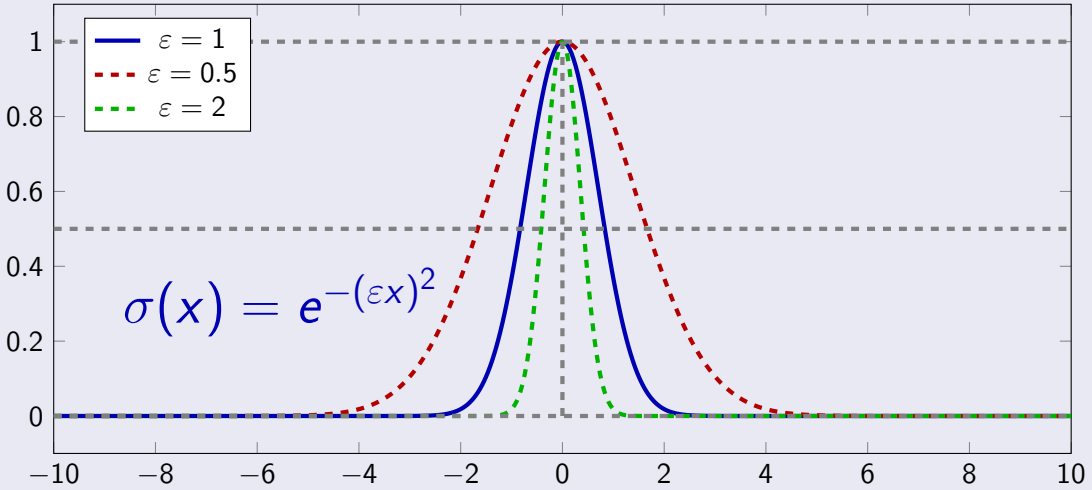
predictions
$\widehat{y}_j = \tau_j(w)$



Typical activation function: the sigmoid



Another activation function: the gaussian radial basis function



Typical output functions

- the **softmax function**

$$\forall j \in \{1, \ldots, m\} \qquad \tau_j : \mathbb{R}^m \to \mathbb{R}$$
$$\tau_j(w) = \frac{e^{w_j}}{\sum_{k=1}^{m} e^{w_k}} \in]0, 1[$$

  i.e. the prediction $\widehat{y} = \tau(w)$ can be seen as a **probability distribution** on $\mathcal{Y}$:
  $\rightarrow$ actual prediction is then $\arg\max_{1 \leqslant j \leqslant m} \widehat{y}_j$.

- the **identity function**

$$\widehat{y} = w \in \mathbb{R}^m,$$

  i.e. the prediction $\widehat{y}$ is the network output w.

Fitting Neural Networks

Network's parameters θ consist in:

- $\{\alpha_{0,i}, \alpha_{1,i}, \ldots, \alpha_{d,i}\}\ \forall i \in \{1, \ldots, p\}$ *i.e.* $p(d+1)$ weights,
- and $\{\beta_{0,i}, \beta_{1,i}, \ldots, \beta_{p,i}\}\ \forall i \in \{1, \ldots, m\}$ *i.e.* $m(p+1)$ weights.

Usual loss functions (errors measures)

- regression: squared error $L(\theta) = \sum_{i=1}^{n} \|y_i - \widehat{y_i}\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (y_{ij} - \widehat{y_{ij}})^2$.
- classification: squared error or cross-entropy (deviance, logistic loss)
  $L(\theta) = -\sum_{i=1}^{n} L_{log}(y_i, \widehat{y_i}) = -\sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \log(\widehat{y_{ij}})$.

Generic approach to minimize $L(\theta)$:

$\rightarrow$ gradient descent, called **back-propagation**.

Avoid overfitting using:

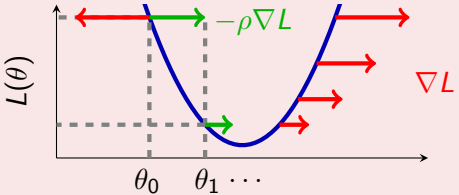
- a penalty term,
- or an early stop.

Gradient descent

Problem: minimize a regular function $L : \begin{cases} \mathbb{R}^q & \to & \mathbb{R} \\ \theta & \mapsto & L(\theta) \end{cases}$.

Solution: follow the (opposite of) the gradient!

$$L(\theta) \geqslant L\Big(\theta - \rho \frac{d}{d\theta} L(\theta)\Big) = L\Big(\theta - \rho \nabla L(\theta)\Big),$$



with ρ called *step size* or *learning rate* in ML.

If L is strictly convex, gradient methods lead to a point θ^* close to the global minimum *cf.* Section 5.6.2.1 of [HMS20].

Back-propagation

Compute a better θ with gradient descent

$\rightarrow$ computation of the gradient *w.r.t.* the parameters $\theta = \{\alpha, \beta\}$ using the **chain rule** (*cf.* Theorem 3.7 in [HMS20]).

Reminder: $\left(\nabla_\theta L(\widetilde{\theta})\right)^T = \frac{\partial}{\partial\theta} L(\widetilde{\theta}) = \left(\frac{\partial}{\partial\theta_1} L(\widetilde{\theta}), \frac{\partial}{\partial\theta_2} L(\widetilde{\theta}), \ldots, \frac{\partial}{\partial\theta_{dim(\theta)}} L(\widetilde{\theta})\right)$

Remember that

$$
\begin{aligned}
z_i &= \sigma(\alpha_{0i} + \alpha_i^T x), && \forall i \in \{1, \ldots, p\}, \\
w_j &= \beta_{0j} + \beta_j^T z, && \forall j \in \{1, \ldots, m\}, \\
\widehat{y}_j &= \tau_j(w), && \forall j \in \{1, \ldots, m\}, \\
L(\theta) &= \sum_{i=1}^{n} L(y_i, \widehat{y}_i). && \text{with } y_i \in \mathbb{R}^m \text{ and } \widehat{y}_i \in \mathbb{R}^m.
\end{aligned}
$$

Gradient computation: loss

$$
\begin{aligned}
(\nabla_\theta L)^T = \frac{\partial}{\partial \theta} \sum_{i=1}^{n} L(y_i, \widehat{y_i}) &= \frac{\partial}{\partial \theta} \sum_{i=1}^{n} \sum_{j=1}^{m} L(y_{ij}, \widehat{y_{ij}}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\partial}{\partial \theta} L(y_{ij}, \widehat{y_{ij}}) \\
\text{(chain rule)} &= \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \theta} \widehat{y_{ij}}
\end{aligned}
$$

Gradient computation: loss

$$(\nabla_\theta L)^T = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \theta} \widehat{y_{ij}}$$

Squared error

$$\frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) = \frac{\partial}{\partial \widehat{y_{ij}}} (y_{ij} - \widehat{y_{ij}})^2 \quad = \quad -2(y_{ij} - \widehat{y_{ij}}).$$

Gradient computation: loss

$$(\nabla_\theta L)^T = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \theta} \widehat{y_{ij}}$$

Cross entropy

$$\frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) = \frac{\partial}{\partial \widehat{y_{ij}}} y_{ij} \log(\widehat{y_{ij}}) = y_{ij} \frac{\partial}{\partial \widehat{y_{ij}}} \log(\widehat{y_{ij}}) \quad = \quad \frac{y_{ij}}{\widehat{y_{ij}}}.$$

Gradient computation: loss

$$(\nabla_\theta L)^T = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \theta} \widehat{y_{ij}}$$

So now, we fix a sample $\widehat{y} \in \mathbb{R}^m$, and we have to deal with $\frac{\partial}{\partial \theta}\widehat{y}_i = \frac{\partial}{\partial \theta}\tau_i(w)$, where $\widehat{y}_i \in \mathbb{R}$ and $i \in \{1, \ldots, m\}$.

Gradient computation: β

$$
\begin{aligned}
\blacksquare \forall i \in \{1,\dots,m\}, \qquad w_i &= \beta_{i0} + \beta_i^T z = \beta_{i0} + \sum_{k=1}^{p} \beta_{ik} z_k \\
\frac{\partial}{\partial \beta_{i0}} w_i &= \frac{\partial}{\partial \beta_{i0}} \beta_{i0} = 1, \text{ and } \frac{\partial}{\partial \beta_{k0}} w_i = 0 \text{ if } k \neq i, \\
\frac{\partial}{\partial \beta_{ij}} w_i &= \frac{\partial}{\partial \beta_{ij}} \beta_{ij} z_j = z_j, \text{ and } \frac{\partial}{\partial \beta_{kj}} w_i = 0 \text{ if } k \neq i. \\
\blacksquare \forall i \in \{1,\dots,m\}, \qquad \widehat{y}_i &= \tau_i(w) \\
\forall k \in \{1,\dots,m\}, \qquad \frac{\partial}{\partial \beta_{k0}} \widehat{y}_i &= \frac{\partial}{\partial \beta_{k0}} \tau_i(w) = \nabla \tau_i(w)^T \frac{\partial}{\partial \beta_{k0}} w = \frac{\partial}{\partial w_k} \tau_i(w), \\
\forall j \in \{1,\dots,p\} \qquad \frac{\partial}{\partial \beta_{kj}} \widehat{y}_i &= \frac{\partial}{\partial \beta_{kj}} \tau_i(w) = \nabla \tau_i(w)^T \frac{\partial}{\partial \beta_{kj}} w = \frac{\partial}{\partial w_k} \tau_i(w) z_j.
\end{aligned}
$$

Gradient computation: β

- We have $\forall i \in \{1, \ldots, m\}, \quad \forall \quad k \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, p\},$

$$\begin{aligned} \frac{\partial}{\partial \beta_{k0}} \widehat{y}_i &= \frac{\partial}{\partial w_k} \tau_i(w), \\ \frac{\partial}{\partial \beta_{kj}} \widehat{y}_i &= \frac{\partial}{\partial w_k} \tau_i(w) z_j. \end{aligned}$$

Let's compute the partial derivatives with respect to α components.

Gradient computation: α

First, the neurons $\{z_i\}_{i=1}^{p}$:

$$
\begin{aligned}
\blacksquare \forall i \in \{1,\dots,p\}, \qquad z_i &= \sigma(\alpha_{i0} + \alpha_i^T x) = \sigma\Big(\alpha_{i0} + \sum_{k=1}^{d} \alpha_{ik} x_k\Big) \\
\frac{\partial}{\partial \alpha_{i0}} z_i &= \sigma'(\alpha_{i0} + \alpha_i^T x) \frac{\partial}{\partial \alpha_{i0}} \alpha_{i0} = \sigma'(\alpha_{i0} + \alpha_i^T x), \\
\forall j \in \{1,\dots,d\}, \quad \frac{\partial}{\partial \alpha_{ij}} z_i &= \sigma'(\alpha_{i0} + \alpha_i^T x) \frac{\partial}{\partial \alpha_{ij}} \alpha_{ij} x_j = \sigma'(\alpha_{i0} + \alpha_i^T x) x_j, \\
\frac{\partial}{\partial \alpha_{k0}} z_i &= \frac{\partial}{\partial \alpha_{kj}} z_i = 0 \text{ if } k \neq i.
\end{aligned}
$$

Now, what are the partial derivatives of w with respect to α components?

Gradient computation: α

- We have $\forall i \in \{1,\dots,p\}$, $\dfrac{\partial}{\partial \alpha_{i0}} z_i = \sigma'(\alpha_{i0} + \alpha_i^T x)$,

  $\forall j \in \{1,\dots,d\}$, $\dfrac{\partial}{\partial \alpha_{ij}} z_i = \sigma'(\alpha_{i0} + \alpha_i^T x) x_j$ and $\dfrac{\partial}{\partial \alpha_{k0}} z_i = \dfrac{\partial}{\partial \alpha_{kj}} z_i = 0$ if $k \neq i$.

- $\forall i \in \{1,\dots,m\}$, $w_i = \beta_{i0} + \beta_i^T z = \beta_{i0} + \sum_{l=1}^{p} \beta_{il} z_l$

  $\forall k \in \{1,\dots,p\}$, $\dfrac{\partial}{\partial \alpha_{k0}} w_i = \sum_{l=1}^{p} \beta_{il} \dfrac{\partial}{\partial \alpha_{k0}} z_l = \beta_{ik} \dfrac{\partial}{\partial \alpha_{k0}} z_k = \beta_{ik} \sigma'(\alpha_{k0} + \alpha_k^T x)$,

  $\forall j \in \{1,\dots,d\}$, $\dfrac{\partial}{\partial \alpha_{kj}} w_i = \sum_{l=1}^{p} \beta_{il} \dfrac{\partial}{\partial \alpha_{ij}} z_l = \beta_{ik} \dfrac{\partial}{\partial \alpha_{kj}} z_k = \beta_{ik} \sigma'(\alpha_{k0} + \alpha_k^T x) x_j$.

Now, what are the partial derivatives of $\widehat{y}$ with respect to α components?

Gradient computation: α

- We have $\forall i \in \{1,\dots,m\}, \quad \forall \quad k \in \{1,\dots,p\}, \qquad \forall j \in \{1,\dots,d\},$

$$\frac{\partial}{\partial \alpha_{k0}} w_i = \beta_{ik}\sigma'(\alpha_{k0} + \alpha_k^T x), \text{ and } \frac{\partial}{\partial \alpha_{kj}} w_i = \beta_{ik}\sigma'(\alpha_{k0} + \alpha_k^T x)x_j.$$

- $\forall i \in \{1,\dots,m\}, \quad \widehat{y}_i = \tau_i(w)$

$$\begin{aligned}
\forall k \in \{1,\dots,p\}, \frac{\partial}{\partial \alpha_{k0}} \widehat{y}_i &= \frac{\partial}{\partial \alpha_{k0}} \tau_i(w) = \nabla \tau_i(w)^T \frac{\partial}{\partial \alpha_{k0}} w \\
&= \sum_{l=1}^{m} \frac{\partial}{\partial w_l} \tau_i(w) \frac{\partial}{\partial \alpha_{k0}} w_l = \sum_{l=1}^{m} \frac{\partial}{\partial w_l} \tau_i(w) \beta_{lk}\sigma'(\alpha_{k0} + \alpha_k^T x), \\
\forall j \in \{1,\dots,d\}, \frac{\partial}{\partial \alpha_{kj}} \widehat{y}_i &= \frac{\partial}{\partial \alpha_{kj}} \tau_i(w) = \nabla \tau_i(w)^T \frac{\partial}{\partial \alpha_{kj}} w \\
&= \sum_{l=1}^{m} \frac{\partial}{\partial w_l} \tau_i(w) \frac{\partial}{\partial \alpha_{kj}} w_l = \sum_{l=1}^{m} \frac{\partial}{\partial w_l} \tau_i(w) \beta_{lk}\sigma'(\alpha_{k0} + \alpha_k^T x) x_j.
\end{aligned}$$

Gradient computation: α

- We have $\forall i \in \{1, \ldots, m\}, \quad \forall \quad k \in \{1, \ldots, p\}, \forall j \in \{1, \ldots, d\},$

$$\frac{\partial}{\partial \alpha_{k0}} \widehat{y}_i = \left(\sum_{l=1}^{m} \frac{\partial}{\partial w_l} \tau_i(w) \beta_{lk} \right) \sigma'(\alpha_{k0} + \alpha_k^T x),$$

$$\frac{\partial}{\partial \alpha_{kj}} \widehat{y}_i = \left(\sum_{l=1}^{m} \frac{\partial}{\partial w_l} \tau_i(w) \beta_{lk} \right) \sigma'(\alpha_{k0} + \alpha_k^T x) x_j.$$

Here is where we see backpropagation.

Put it all together: $\frac{\partial}{\partial\theta}\widehat{y}_i = (\nabla_\theta \widehat{y}_i)^T$

$$
\begin{aligned}
\blacksquare \forall i \in \{1,\dots,m\}, \quad & \forall \quad k \in \{1,\dots,m\}, \forall j \in \{1,\dots,p\}, \\
\frac{\partial}{\partial \beta_{k0}}\widehat{y}_i &= \frac{\partial}{\partial w_k}\tau_i(w) \stackrel{def}{=} \delta_k^\beta, \\
\frac{\partial}{\partial \beta_{kj}}\widehat{y}_i &= \frac{\partial}{\partial w_k}\tau_i(w) z_j = \delta_k^\beta z_j. \\
\blacksquare \forall i \in \{1,\dots,m\}, \quad & \forall \quad k \in \{1,\dots,p\}, \forall j \in \{1,\dots,d\}, \\
\frac{\partial}{\partial \alpha_{k0}}\widehat{y}_i &= \left(\sum_{l=1}^{m} \frac{\partial}{\partial w_l}\tau_i(w)\beta_{lk}\right)\sigma'(\alpha_{k0} + \alpha_k^T x) \stackrel{def}{=} \delta_k^\alpha, \\
\frac{\partial}{\partial \alpha_{kj}}\widehat{y}_i &= \left(\sum_{l=1}^{m} \frac{\partial}{\partial w_l}\tau_i(w)\beta_{lk}\right)\sigma'(\alpha_{k0} + \alpha_k^T x) x_j = \delta_k^\alpha x_j.
\end{aligned}
$$

Put it all together: $\frac{\partial}{\partial\theta}\widehat{y}_i = (\nabla_\theta \widehat{y}_i)^T$

- $\forall i \in \{1,\dots,m\}, \quad \forall \quad k \in \{1,\dots,m\}, \forall j \in \{1,\dots,p\},$

$$\begin{aligned}
\frac{\partial}{\partial\beta_{k0}}\widehat{y}_i &= \frac{\partial}{\partial w_k}\tau_i(w) \stackrel{def}{=} \delta_k^\beta, \\
\frac{\partial}{\partial\beta_{kj}}\widehat{y}_i &= \frac{\partial}{\partial w_k}\tau_i(w) z_j = \delta_k^\beta z_j.
\end{aligned}$$

- $\forall i \in \{1,\dots,m\}, \quad \forall \quad k \in \{1,\dots,p\}, \forall j \in \{1,\dots,d\},$

$$\begin{aligned}
\frac{\partial}{\partial\alpha_{k0}}\widehat{y}_i &= \left(\sum_{l=1}^m \delta_l^\beta \beta_{lk}\right)\sigma'(\alpha_{k0}+\alpha_k^T x) \stackrel{def}{=} \delta_k^\alpha, \\
\frac{\partial}{\partial\alpha_{kj}}\widehat{y}_i &= \left(\sum_{l=1}^m \delta_l^\beta \beta_{lk}\right)\sigma'(\alpha_{k0}+\alpha_k^T x) x_j = \delta_k^\alpha x_j. \\
\Rightarrow \delta_k^\alpha &= \left(\sum_{l=1}^m \delta_l^\beta \beta_{lk}\right)\sigma'(\alpha_{k0}+\alpha_k^T x) \text{ a function of } \left\{\delta_l^\beta\right\}_{l=1}^m.
\end{aligned}$$

Put it all together: $\frac{\partial}{\partial\theta}\widehat{y}_i = (\nabla_\theta \widehat{y}_i)^T$

- $\forall i \in \{1,\ldots,m\}, \quad \forall \quad k \in \{1,\ldots,m\}, \forall j \in \{1,\ldots,p\},$

$$\begin{aligned}
\frac{\partial}{\partial\beta_{k0}}\widehat{y}_i &= \frac{\partial}{\partial w_k}\tau_i(w) \stackrel{def}{=} \delta_k^\beta, \\
\frac{\partial}{\partial\beta_{kj}}\widehat{y}_i &= \frac{\partial}{\partial w_k}\tau_i(w) z_j = \delta_k^\beta z_j.
\end{aligned}$$

- $\forall i \in \{1,\ldots,m\}, \quad \forall \quad k \in \{1,\ldots,p\}, \forall j \in \{1,\ldots,d\},$

$$\begin{aligned}
\frac{\partial}{\partial\alpha_{k0}}\widehat{y}_i &= \left(\sum_{l=1}^m \delta_l^\beta \beta_{lk}\right)\sigma'(\alpha_{k0} + \alpha_k^T x) \stackrel{def}{=} \delta_k^\alpha, \\
\frac{\partial}{\partial\alpha_{kj}}\widehat{y}_i &= \left(\sum_{l=1}^m \delta_l^\beta \beta_{lk}\right)\sigma'(\alpha_{k0} + \alpha_k^T x) x_j = \delta_k^\alpha x_j. \\
\Rightarrow \delta_k^\alpha &= \left(\sum_{l=1}^m \delta_l^\beta \beta_{lk}\right)\sigma'(\alpha_{k0} + \alpha_k^T x) \text{ a function of } \left\{\delta_l^\beta\right\}_{l=1}^m.
\end{aligned}$$

Backpropagation equations

$$\forall k \in \{1, \ldots, p\} \quad \delta_k^\alpha = \left(\sum_{l=1}^{m} \delta_l^\beta \beta_{lk} \right) \sigma'(\alpha_{k0} + \alpha_k^T x).$$

A two-pass algorithm

- Forward pass: computation of $\widehat{y}$
- Backward pass:
  - computation of δ^β
  - computation of δ^α from δ^β
- Computation of $\nabla_\theta L$ with δ^α and δ^β.
- Gradient descent *i.e.* update of the weights.

Computation of $\nabla_\theta L$

$$\frac{\partial}{\partial \beta_{k0}} L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \beta_{k0}} \widehat{y_{ij}} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \delta_k^\beta$$

$$\frac{\partial}{\partial \beta_{kl}} L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \beta_{kl}} \widehat{y_{ij}} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \delta_k^\beta z_l$$

$$\frac{\partial}{\partial \alpha_{k0}} L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \alpha_{k0}} \widehat{y_{ij}} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \delta_k^\alpha$$

$$\frac{\partial}{\partial \alpha_{kl}} L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \frac{\partial}{\partial \alpha_{kl}} \widehat{y_{ij}} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial}{\partial \widehat{y_{ij}}} L(y_{ij}, \widehat{y_{ij}}) \delta_k^\alpha x_l$$

Update of the weights

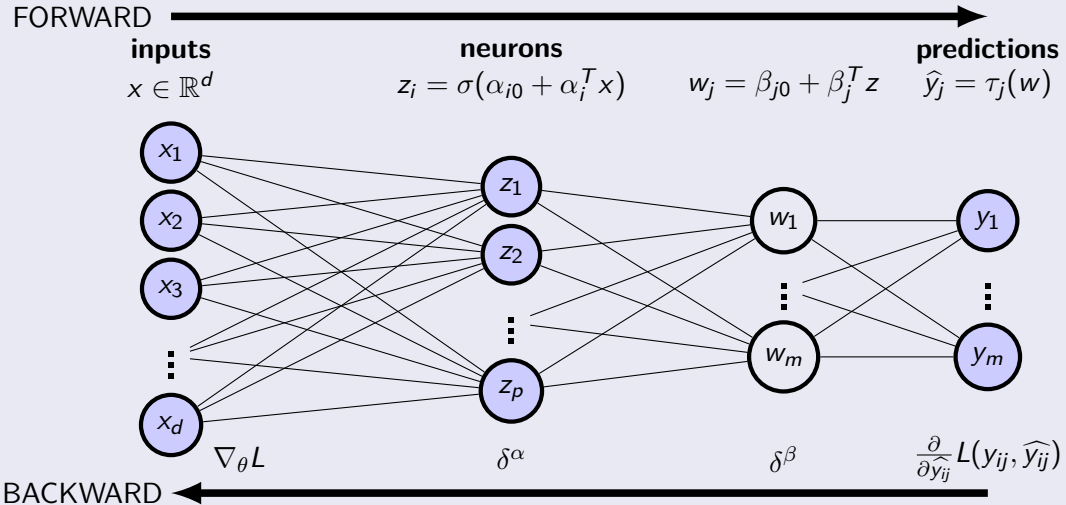
$$\beta_{k0}^{new} = \beta_{k0} - \rho \frac{\partial}{\partial \beta_{k0}} L(\theta)$$

$$\beta_{kj}^{new} = \beta_{kj} - \rho \frac{\partial}{\partial \beta_{kj}} L(\theta)$$

$$\alpha_{k0}^{new} = \alpha_{k0} - \rho \frac{\partial}{\partial \alpha_{k0}} L(\theta)$$

$$\alpha_{kj}^{new} = \alpha_{kj} - \rho \frac{\partial}{\partial \alpha_{kj}} L(\theta)$$

Forward/Backward pass



Advantages of back-propagation

- simple and local nature
- information shared only with connected units $\Rightarrow$ efficient implementation on paralell architecture computer (*e.g.* GPU).

Batch versus Stochastic/Online learning

- We used the entire dataset (batch) to compute the gradient.
- We can also use a random subset (Stochastic Gradient Descent).
- In this case, the learning rate ρ should tend to zero. $\Rightarrow$ efficient implementation on paralell architecture computer (*e.g.* GPU).

Advantages of back-propagation

- simple and local nature
- information shared only with connected units $\Rightarrow$ efficient implementation on paralell architecture computer (*e.g.* GPU).

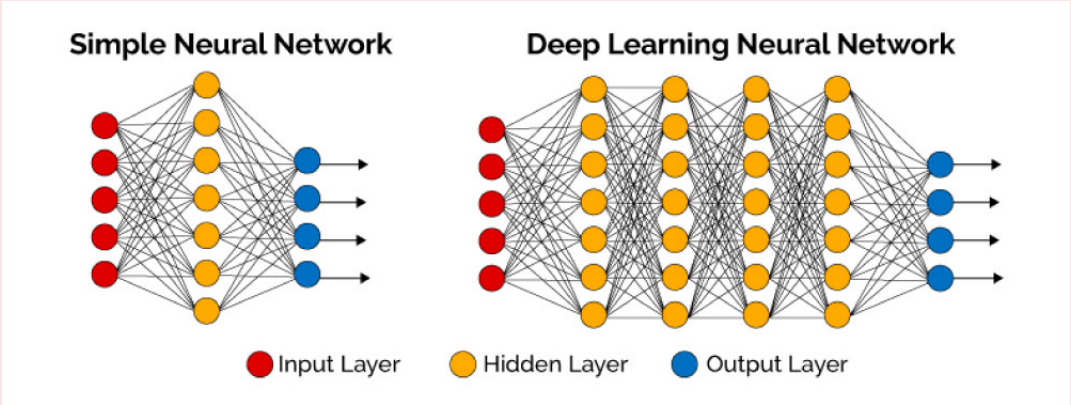
Batch versus Stochastic/Online learning

- We used the entire dataset (batch) to compute the gradient.
- We can also use a random subset (Stochastic Gradient Descent).
- In this case, the learning rate ρ should tend to zero. $\Rightarrow$ efficient implementation on paralell architecture computer (*e.g.* GPU).

Issues

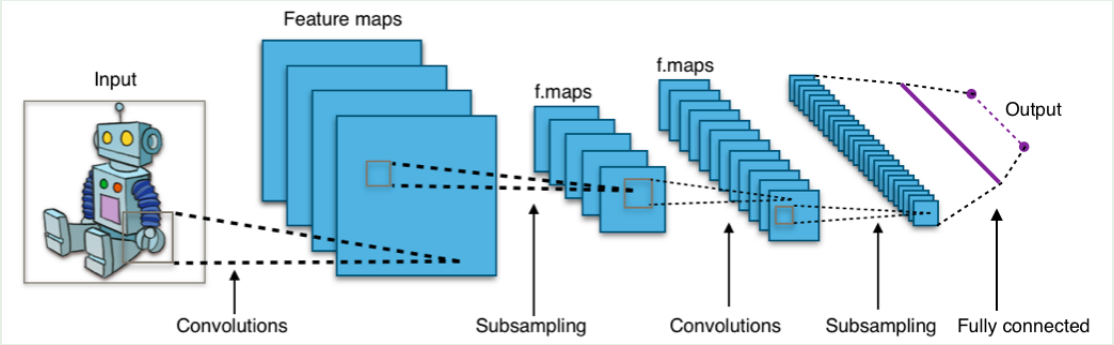
- starting values
- overfitting (penalizes weights, early stoping)
- design number of hidden units and layes
- $\theta \to L(\theta)$ is not strictly convex
  $\Rightarrow$ convergence of the gradient descent is not ensured!

Simple Neural Networks versus Deep Learning



;

Convolutional Neural Network



;

- the most famous libraries are pytorch, keras, tensorflow
- Pro:
  - no feature extraction (end to end learning, learning from the raw data)
- Cons:
  - no guaranties of convergence in a global minimum
  - large amount of training data needed
  - large number of hyperparameters (layer/units number, activation functions...)
  - "black box" model, “not very useful for producing an understandable model for the data” [HTF09]

Outline

Reminder: Andrey Markov's Inequality

Markov's Inequality

Let Z be a non negative random variable and $\varepsilon > 0$ a positive real number. Then,

$$\mathbb{P}(X \geqslant \varepsilon) \leqslant \frac{\mathbb{E}[X]}{\varepsilon}.$$

Proof

$$\begin{aligned} X \geqslant 0 \quad &\Rightarrow \quad \varepsilon \cdot \mathbb{1}_{\{X \geqslant \varepsilon\}} \leqslant X \\ &\Rightarrow \quad \varepsilon \cdot \mathbb{P}(X \geqslant \varepsilon) \leqslant \mathbb{E}[X] \\ &\Rightarrow \quad \mathbb{P}(X \geqslant \varepsilon) \leqslant \frac{\mathbb{E}[X]}{\varepsilon} \end{aligned}$$

(see Theorem 1.3.11 in Section 1.3.4 of [GS20]).

Reminder: Pafnuty Chebyshev's inequality

Chebyshev's inequality

Let Z be a random variable and $\varepsilon > 0$ a positive real number. Then,

$$\mathbb{P}(|Z - \mathbb{E}[Z]| \geqslant \varepsilon) \leqslant \frac{Var(Z)}{\varepsilon^2}.$$

Proof

By using Markov's inequality on the non negative random variable $(Z - \mathbb{E}[Z])^2$,

$$\mathbb{P}(|Z - \mathbb{E}[Z]| \geqslant \varepsilon) = \mathbb{P}\left((Z - \mathbb{E}[Z])^2 \geqslant \varepsilon^2\right) \leqslant \frac{\mathbb{E}\left[(Z - \mathbb{E}[Z])^2\right]}{\varepsilon^2} = \frac{Var(Z)}{\varepsilon^2}.$$

□

(see also [GS20]).

Another famous inequality

Hoeffding's inequality

Let $(X_i)_{i=1}^n$ be n independent random variables. Assume that $\forall i \in \{1, \dots, n\}$, $\exists (a_i, b_i) \in \mathbb{R}^2$ such that $a_i \leqslant X_i \leqslant b_i$. Then, $\forall \varepsilon > 0$,

$$\mathbb{P}\left(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) > \varepsilon\right) \leqslant \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

In order to show this inequality, we'll need another result called **Hoeffding's Lemma**...

Hoeffding's Lemma

Let Z be a centered random variable (*i.e.* $\mathbb{E}[Z] = 0$) such that $a \leqslant Z \leqslant b$.

$$\text{Then}, \forall \lambda > 0, \qquad \mathbb{E}\left[e^{\lambda Z}\right] \leqslant \exp\left(\frac{\lambda^2 (b-a)^2}{8}\right).$$

Proof – 1

Let's consider the function $\Psi : \lambda \mapsto \ln\left(\mathbb{E}\left[e^{\lambda Z}\right]\right)$ with $\lambda > 0$. Note that $\omega \mapsto e^{\lambda Z(\omega)}$ and $\omega \mapsto \frac{\partial^k}{\partial \lambda^k} e^{\lambda Z(\omega)} = Z(\omega) e^{\lambda Z(\omega)}$ $\forall k \in \mathbb{N}$ are bounded. Then they are integrable ($\mathbb{P}$ is a finite measure) and the theorem of *differentiation under the integral sign [HMS20]* gives: $\frac{d^k}{d\lambda^k}\mathbb{E}\left[e^{\lambda Z}\right] = \mathbb{E}\left[\frac{\partial^k}{\partial \lambda^k} e^{\lambda Z}\right] = \mathbb{E}\left[Z^k e^{\lambda Z}\right]$.

$$\Rightarrow \Psi'(\lambda) = \frac{d}{d\lambda} \ln\left(\mathbb{E}\left[e^{\lambda Z}\right]\right) = \frac{d}{d\lambda}\left(\mathbb{E}\left[e^{\lambda Z}\right]\right) \frac{1}{\mathbb{E}\left[e^{\lambda Z}\right]} = \frac{\mathbb{E}\left[Z e^{\lambda Z}\right]}{\mathbb{E}\left[e^{\lambda Z}\right]}$$

Hoeffding's Lemma: $\mathbb{E}\left[e^{\lambda Z}\right] \leqslant \exp\left(\frac{\lambda^2(b-a)^2}{8}\right)$.

Proof – 2

$\Psi : \lambda \mapsto \ln\left(\mathbb{E}\left[e^{\lambda Z}\right]\right)$ with $\lambda > 0$. $\Rightarrow \Psi'(\lambda) = \frac{d}{d\lambda}\left(\mathbb{E}\left[e^{\lambda Z}\right]\right)\frac{1}{\mathbb{E}\left[e^{\lambda Z}\right]} = \frac{\mathbb{E}\left[Ze^{\lambda Z}\right]}{\mathbb{E}\left[e^{\lambda Z}\right]}$

$$\Rightarrow \Psi''(\lambda) = \frac{d}{d\lambda}\left(\frac{\mathbb{E}\left[Ze^{\lambda Z}\right]}{\mathbb{E}\left[e^{\lambda Z}\right]}\right) = \frac{\frac{d}{d\lambda}\left(\mathbb{E}\left[Ze^{\lambda Z}\right]\right)\mathbb{E}\left[e^{\lambda Z}\right] - \mathbb{E}\left[Ze^{\lambda Z}\right]\frac{d}{d\lambda}\left(\mathbb{E}\left[e^{\lambda Z}\right]\right)}{\mathbb{E}\left[e^{\lambda Z}\right]^2}$$

$$= \frac{\mathbb{E}\left[Z^2e^{\lambda Z}\right]\mathbb{E}\left[e^{\lambda Z}\right] - \mathbb{E}\left[Ze^{\lambda Z}\right]^2}{\mathbb{E}\left[e^{\lambda Z}\right]^2}$$

$$= \mathbb{E}\left[Z^2\frac{e^{\lambda Z}}{\mathbb{E}\left[e^{\lambda Z}\right]}\right] - \left(\mathbb{E}\left[Z\frac{e^{\lambda Z}}{\mathbb{E}\left[e^{\lambda Z}\right]}\right]\right)^2$$

Hoeffding's Lemma: $\mathbb{E}\left[e^{\lambda Z}\right] \leqslant \exp\left(\frac{\lambda^2(b-a)^2}{8}\right)$.

Proof – 3

$\Psi : \lambda \mapsto \ln\left(\mathbb{E}\left[e^{\lambda Z}\right]\right)$ with $\lambda > 0$. $\Rightarrow \Psi''(\lambda) = \mathbb{E}\left[Z^2 \frac{e^{\lambda Z}}{\mathbb{E}\left[e^{\lambda Z}\right]}\right] - \left(\mathbb{E}\left[Z \frac{e^{\lambda Z}}{\mathbb{E}\left[e^{\lambda Z}\right]}\right]\right)^2$.

Note that $\omega \mapsto \frac{e^{\lambda Z(\omega)}}{\mathbb{E}\left[e^{\lambda Z}\right]}$ is a density on $(\Omega, \mathbb{P})$ (non negative & $\int_\Omega \frac{e^{\lambda Z(\omega)}}{\mathbb{E}\left[e^{\lambda Z}\right]} d\mathbb{P}(\omega) = \frac{\mathbb{E}\left[e^{\lambda Z}\right]}{\mathbb{E}\left[e^{\lambda Z}\right]} = 1$).

Let's consider the probability measure $\mathbb{Q}_\lambda$ defined by $\forall A \subset \Omega$, $\mathbb{Q}_\lambda(A) = \int_A \frac{e^{\lambda Z(\omega)}}{\mathbb{E}\left[e^{\lambda Z}\right]} d\mathbb{P}(\omega)$.

$$\Rightarrow \Psi''(\lambda) = \int_\Omega Z(\omega)^2 dQ_\lambda(\omega) - \left(\int_\Omega Z(\omega) dQ_\lambda(\omega)\right)^2 = \mathbb{E}_{\mathbb{Q}_\lambda}\left[Z^2\right] - \mathbb{E}_{\mathbb{Q}_\lambda}\left[Z\right]^2 = Var_{\mathbb{Q}_\lambda}\left(Z\right).$$

Note also that $a \leqslant Z \leqslant b \Rightarrow \frac{a-b}{2} \leqslant Z - \frac{b+a}{2} \leqslant \frac{b-a}{2} \Rightarrow |Z - \frac{b+a}{2}| \leqslant \frac{b-a}{2}$.

Then $\forall \lambda > 0, \Psi''(\lambda) = Var_{\mathbb{Q}_\lambda}\left(Z - \frac{b+a}{2}\right) \leqslant \mathbb{E}_{\mathbb{Q}_\lambda}\left[\left(Z - \frac{b+a}{2}\right)^2\right] \leqslant \left(\frac{b-a}{2}\right)^2$.

Hoeffding's Lemma: $\mathbb{E}\left[e^{\lambda Z}\right] \leqslant \exp\left(\frac{\lambda^2(b-a)^2}{8}\right)$.

Proof – 4

$\Psi : \lambda \mapsto \ln\left(\mathbb{E}\left[e^{\lambda Z}\right]\right)$ with $\lambda > 0$. $\Rightarrow \Psi''(\lambda) \leqslant \left(\frac{b-a}{2}\right)^2$.

- $\Psi'(0) = \frac{\mathbb{E}\left[Ze^{0\times Z}\right]}{\mathbb{E}\left[e^{0\times Z}\right]} = \mathbb{E}[Z] = 0$ (Z is centered) $\Rightarrow \Psi'(\lambda) = \int_0^\lambda \Psi''(u)du \leqslant \lambda\frac{(b-a)^2}{4}$.
- $\Psi(0) = \ln\left(\mathbb{E}\left[e^{0\times Z}\right]\right) = \ln(1) = 0$, thus

$$\Psi(\lambda) = \int_0^\lambda \Psi'(u)du \leqslant \int_0^\lambda \frac{u(b-a)^2}{4}du = \frac{\lambda^2(b-a)^2}{8}.$$

□

Hoeffding's inequality

Let $(X_i)_{i=1}^n$ be n independent random variables. Assume that $\forall i \in \{1, \dots, n\}$, $\exists (a_i, b_i) \in \mathbb{R}^2$ such that $a_i \leqslant X_i \leqslant b_i$. Then, $\forall \varepsilon > 0$,

$$\mathbb{P}\left(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) > \varepsilon\right) \leqslant \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Proof – 1

Let's consider the centered random variable $Z_i = X_i - \mathbb{E}[X_i]$. Then $\forall \lambda > 0$,

$$\mathbb{P}\left(\sum_{i=1}^{n} Z_i > \varepsilon\right) = \mathbb{P}\left(e^{\lambda \sum_{i=1}^{n} Z_i} > e^{\lambda\varepsilon}\right) \overset{\text{(Markov's ineq.)}}{\leqslant} \frac{\mathbb{E}\left[e^{\lambda \sum_{i=1}^{n} Z_i}\right]}{e^{\lambda\varepsilon}} =$$

$$= \frac{\mathbb{E}\left[\prod_{i=1}^{n} e^{\lambda Z_i}\right]}{e^{\lambda\varepsilon}} = \frac{\prod_{i=1}^{n} \mathbb{E}\left[e^{\lambda Z_i}\right]}{e^{\lambda\varepsilon}} \quad \text{(indep. vars., Prop. 1.20 [GS20])}$$

Hoeffding's inequality: $\mathbb{P}\left(\sum_{i=1}^n (X_i - \mathbb{E}[X_i]) > \varepsilon\right) \leqslant \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$.

Proof – 2

$$\mathbb{P}\left(\sum_{i=1}^n Z_i > \varepsilon\right) \leqslant \frac{\prod_{i=1}^n \mathbb{E}\left[e^{\lambda Z_i}\right]}{e^{\lambda\varepsilon}} \overset{\text{(Hoeffding's Lemma)}}{\leqslant} \frac{\prod_{i=1}^n \exp\left(\frac{\lambda^2 (b_i - a_i)^2}{8}\right)}{e^{\lambda\varepsilon}} = \exp\left(-\lambda\varepsilon + \frac{\lambda^2}{8}\sum_{i=1}^n (b_i - a_i)^2\right)$$

Note that $\lambda \mapsto -\lambda\varepsilon + \frac{\lambda^2}{8}\sum_{i=1}^n (b_i - a_i)^2$ is minimal for $\lambda_* = \frac{4\varepsilon}{\sum_{i=1}^n (b_i - a_i)^2}$.

$$\Rightarrow \mathbb{P}\left(\sum_{i=1}^n Z_i > \varepsilon\right) \leqslant \exp\left(-\lambda_*\varepsilon + \frac{\lambda_*^2}{8}\sum_{i=1}^n (b_i - a_i)^2\right)$$

$$= \exp\left(-\frac{4\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2} + \frac{16\varepsilon^2}{8\sum_{i=1}^n (b_i - a_i)^2}\right) = \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

□

Hoeffding's inequality

Let $(X_i)_{i=1}^n$ be n independent random variables. Assume that $\forall i \in \{1, \dots, n\}$, $\exists (a_i, b_i) \in \mathbb{R}^2$ such that $a_i \leqslant X_i \leqslant b_i$. Then, $\forall \varepsilon > 0$,

$$\mathbb{P}\left(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) > \varepsilon\right) \leqslant \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Corollary

$$\mathbb{P}\left(|\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i])| > \varepsilon\right) \leqslant 2\exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Hoeffding's inequality

Corollary

$$\mathbb{P}\left(|\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i])| > \varepsilon\right) \leqslant 2\delta \quad \text{with} \quad \delta = \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Proof

$(-X_i)_{i=1}^n$ are also n independent random variables with $\forall i \in \{1, \dots, n\}$, $-b_i \leqslant -X_i \leqslant -a_i$ and $(-b_i, -a_i) \in \mathbb{R}^2$. Since $(a_i - b_i)^2 = (b_i - a_i)^2$, we get $\mathbb{P}(-\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) > \varepsilon) \leqslant \delta$. Finally, since the events are incompatible,

$$\mathbb{P}\left(|\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i])| > \varepsilon\right) = \mathbb{P}\left(\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) > \varepsilon\right) + \mathbb{P}\left(-\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i]) > \varepsilon\right) \leqslant 2\delta.$$

□

Another way to write the inequality

Let $(X_i)_{i=1}^n$ be n independent random variables. Assume that $\forall i \in \{1, \dots, n\}$, $\exists (a_i, b_i) \in \mathbb{R}^2$ such that $a_i \leqslant X_i \leqslant b_i$. Then, $\forall \varepsilon > 0$,

$$\mathbb{P}\left(\left| \frac{1}{n} \sum_{i=1}^{n} (X_i - \mathbb{E}[X_i]) \right| > \varepsilon \right) \leqslant 2\delta \quad \text{with} \quad \delta = \exp\left(- \frac{2n^2\varepsilon^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right).$$

by simply replacing the variable $\varepsilon \to n\varepsilon$.

Outline

.

Optimal classifier

- X and Y are random variables distributed according to $\mathbb{P}$.
- We look for a **classifier** that predicts Y, denoted by $c : \mathcal{X} \to \mathcal{Y} = \{y_+, y_-\}$.
- The probability of error, or **risk** is $R(c) = \mathbb{P}(c(X) \neq Y) = \mathbb{E}\left[\mathbb{1}_{\{c(X) \neq Y\}}\right]$

$$
\begin{aligned}
&= \mathbb{E}\left[\mathbb{1}_{\{c(X)=y_+\}}\mathbb{1}_{\{Y=y_-\}} + \mathbb{1}_{\{c(X)=y_-\}}\mathbb{1}_{\{Y=y_+\}}\right] \\
&= \mathbb{E}\left[\mathbb{1}_{\{c(X)=y_+\}}\mathbb{E}\left[\mathbb{1}_{\{Y=y_-\}} \,\middle|\, X\right] + \mathbb{1}_{\{c(X)=y_-\}}\mathbb{E}\left[\mathbb{1}_{\{Y=y_+\}} \,\middle|\, X\right]\right] \\
(\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y \mid X]] \,\&\, \mathbb{E}[\phi(X)Y \mid X] &\overset{a.s}{=} \phi(X)\mathbb{E}[Y \mid X] \text{ see Prop. 1.31 in [GS20]}) \\
&= \mathbb{E}\left[\mathbb{1}_{\{c(X)=y_+\}}\mathbb{P}(Y = y_- \mid X) + \mathbb{1}_{\{c(X)=y_-\}}\mathbb{P}(Y = y_+ \mid X)\right] \\
&= \mathbb{E}\left[\mathbb{1}_{\{c(X)=y_+\}}(1 - \eta(X)) + \mathbb{1}_{\{c(X)=y_-\}}\eta(X)\right]
\end{aligned}
$$

with $\eta(X) = \mathbb{P}(Y = y_+ \mid X) \in [0, 1]$.

Optimal classifier

$$R(c) = \mathbb{E}\left[\mathbb{1}_{\{\,c(X)=y_+\,\}}\Big(1-\eta(X)\,\Big) + \mathbb{1}_{\{\,c(X)=y_+\,\}}\eta(X)\right]$$

Since $0 \leqslant \eta(X) \leqslant 1$, the optimal classifier $c : \mathcal{X} \to \mathcal{Y}$, *i.e.* minimizing $R(c)$, should be

- equal to y_+ when $\eta(X) > \frac{1}{2}$
- equal to y_- otherwise.

t is called the **Bayes classifier** or **target function**:

$$t(x) = \begin{cases} y_+ & \text{if } \eta(x) > \frac{1}{2}, \\ y_- & \text{if } \eta(x) \leqslant \frac{1}{2}. \end{cases}$$

The associated risk is then called the **Bayes risk**

$$R^* = \inf_{c:\mathcal{X}\to\mathcal{Y}} R(c) = R(t) = \mathbb{E}\Big[\min\{\,\eta(X), 1-\eta(X)\,\}\Big].$$

Note on *Bayes* term.

- Thomas Bayes (1701-1761)
- $\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A)\mathbb{P}(A)}{\mathbb{P}(B)}$.
- prior $\mathbb{P}(A)$, posterior $\mathbb{P}(A \mid B)$.
- *no data*, information in the prior, parameters have a probability distribution
- prior knowledge on the parameters $p_{prior}(\theta)$, θ becomes a random variable.
- the set of distributions of the dataset $\mathbb{P}_\theta(D_n = d_n)$ **parameterized by** $\theta \in \Theta$ considered as the **conditional probability distribution given** $\theta \in \Theta$: $\mathbb{P}(D_n = d_n \mid \theta)$.
- $p_{post}(\theta) = \mathbb{P}(\theta \mid D_n = d_n) = \frac{\mathbb{P}_\theta(D_n = d_n)}{\int_\theta \mathbb{P}_\theta(D_n = d_n) p_{prior}(\theta) d\theta}$

Optimal classifier

Deterministic case

Let's suppose $(X, Y) = \Big(X, f(X)\Big)$ with $X \sim \mathbb{P}$ and $f : \mathcal{X} \to \mathcal{Y}$.

$$\begin{aligned} \eta(X) &= \mathbb{P}(Y = y_+ \mid X) = \mathbb{E}\Big[\mathbb{1}_{\{Y=y_+\}} \,\Big|\, X\Big] \\ &= \mathbb{E}\Big[\mathbb{1}_{\{f(X)=y_+\}} \,\Big|\, X\Big] \\ &= \mathbb{1}_{\{f(X)=y_+\}}. \end{aligned}$$

$\Rightarrow R^* = \mathbb{E}[\min\{\eta(X), 1-\eta(X)\}] = 0$, *i.e.* the Bayes risk is equal to zero.

General case

We can then define the **noise level** $s(x) = \min\{\eta(x), 1-\eta(x)\}$,
and the Bayes risk can be rewritten $R^* = \mathbb{E}\Big[s(X)\Big]$.

Notations

- **Input (features)**: X (random) or x (realization).
- **Output (targets)**: Y (random) or y (realization).
- **Random dataset**: $D_n = (X_i, Y_i)_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$
- usually, features $\mathcal{X} = \mathbb{R}^d$
- set of targets/labels $\{y_k\}_{k=1}^m$. Binary classification: $\{y_+, y_-\}$ ($m = 2$).

Empirical risk & the loss class

Empirical risk

$$R_n(c) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{ c(X_i) = Y_i \}}.$$

Loss class

$$\mathcal{F} = \left\{ f : (x, y) \mapsto \mathbb{1}_{\{ c(x) \neq y \}} \;\middle|\; c \in \mathcal{C} \right\} :$$

Let's denote $f_c(x, y) = \mathbb{1}_{\{ c(x) \neq y \}}$:

$$R(c) - R_n(c) = \mathbb{E}\left[f_c(X, Y) \right] - \frac{1}{n} \sum_{i=1}^{n} f_c(X, Y).$$

Some definitions

- $\mathcal{C}$ is the set of **candidate functions** or **classifiers**: if $c \in \mathcal{C}$, $c : \mathcal{X} \to \mathcal{Y}$.
- c^* denotes the classifier that minimizes the **theoretical risk**:

$$c^* \in \arg\min_{c \in \mathcal{C}} R(c).$$

  Note that if the Bayes classifier t is in $\mathcal{C}$, then $R(c^*) = R^* \overset{def}{=} R(t) = \mathbb{E}\left[\mathbb{1}_{\{t(x) \neq y\}}\right]$.
- c_n denotes the classifier that minimizes the **empirical risk**:

$$c_n \in \arg\min_{c \in \mathcal{C}} R_n(c).$$

First result

$\forall c \in \mathcal{C},\ R_n(c) \xrightarrow[n\to+\infty]{} R(c)$ almost surely.

Proof

Strong law of the large numbers (see Theorem 3.1 in [GS20]) applied on integrable (bounded) independent random variables $(f_c(X_i, Y_i))_{i=1}^n$:

$$\mathbb{P}\left(\lim_{n\to+\infty} \frac{1}{n}\sum_{i=1}^{n} f_c(X_i, Y_i) = \mathbb{E}\left[f_c(X, Y)\right]\right) = 1,$$

i.e.

$$\mathbb{P}\left(\lim_{n\to+\infty} R_n(c) = R(c)\right) = 1.$$

Bound for a classifier's risk

Property: Hoeffding's inequality

Let $(X_i)_{i=1}^n$ be n independent random variables. Assume that $\forall i \in \{1, \dots, n\}$, $\exists (a_i, b_i) \in \mathbb{R}^2$ such that $a_i \leqslant X_i \leqslant b_i$. Then, $\forall \varepsilon > 0$,

$$\mathbb{P}\left(\left| \frac{1}{n} \sum_{i=1}^{n} (X_i - \mathbb{E}[X_i]) \right| > \varepsilon \right) \leqslant 2 \exp\left(- \frac{2n^2 \varepsilon^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right).$$

Let us apply this inequality with f_c: $(f_c(X_i, Y_i))_{i=1}^n$ that are independent random variables such that $0 \leqslant f_c(X_i, Y_i) \leqslant 1$: $\mathbb{P}\left(\left| \frac{1}{n} \sum_{i=1}^{n} f_c(X_i, Y_i) - \mathbb{E}[f_c(X, Y)] \right| > \varepsilon \right) \leqslant 2e^{-2n\varepsilon^2}$.

Result: probabilistic bound on the risk estimation error

$$\mathbb{P}\left(|R_n(c) - R(c)| > \varepsilon \right) \leqslant 2e^{-2n\varepsilon^2} \qquad \& \qquad \mathbb{P}\left(R_n(c) - R(c) > \varepsilon \right) \leqslant e^{-2n\varepsilon^2}.$$

Bound for a classifier's risk

Probabilistic bound on the risk estimation error

$$\mathbb{P}\left(|R_n(c) - R(c)| > \varepsilon\right) \leqslant 2e^{-2n\varepsilon^2} \qquad \& \qquad \mathbb{P}\left(R(c) - R_n(c) > \varepsilon\right) \leqslant e^{-2n\varepsilon^2}.$$

By writting $\delta = 2e^{-2n\varepsilon^2}$, we have $\varepsilon = \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2n}}$ (on the right $\varepsilon = \sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2n}}$), so, we can write:

- with probability at most δ, $|R_n(c) - R(c)| > \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2n}}$, or also:
- with probability at least 1 - δ, $|R_n(c) - R(c)| < \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2n}}$.

Bound on the true risk

In practice, knowing n and $R_n(c)$, we have $\forall \delta \in]0, 1[$ a bound on the true risk with probability $1 - \delta$:

$$R(c) \leqslant R_n(c) + \sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2n}}.$$

Bound for a finite set of candidate functions

Let us consider a finite set of candidate classifiers $\mathcal{C}$ with $\#\mathcal{C} = N < +\infty$.

$$\begin{aligned}\mathbb{P}\left(\exists c \in \mathcal{C} \mid |R_n(c) - R(c)| > \varepsilon\right) &= \mathbb{P}\left(\cup_{c\in\mathcal{C}} \{|R_n(c) - R(c)| > \varepsilon\}\right) \\ &\leqslant \sum_{c\in\mathcal{C}} \mathbb{P}\left(|R_n(c) - R(c)| > \varepsilon\right) \\ &\leqslant 2Ne^{-2n\varepsilon^2}.\end{aligned}$$

Then, with $\delta = 2Ne^{-2n\varepsilon^2}$, $\varepsilon = \sqrt{\frac{\ln\left(\frac{2N}{\delta}\right)}{2n}}$, so,

with probability at least $1-\delta$, $\forall c \in \mathcal{C}$,

$$|R_n(c) - R(c)| \leqslant \sqrt{\frac{\ln(N) + \ln\left(\frac{2}{\delta}\right)}{2n}} \quad \text{and} \quad R(c) \leqslant R_n(c) + \sqrt{\frac{\ln(N) + \ln\left(\frac{1}{\delta}\right)}{2n}}.$$

The extra $\ln(N)$ accounts for the fact that we want N bounds to hold simultaneously.

Bound on the estimation error

We know that, with probability $1-\delta$, $\forall c \in \mathcal{C}$, $R_n(c) - R(c)$ is bounded. But we would like to know how to bound the actual risk of our estimation of the optimal classifier $c_n \in \arg\min_{c \in \mathcal{C}} R_n(c)$.

Approximation and estimation errors

$$R(c_n) - R^* = \underbrace{R(c^*) - R^*}_{\text{approximation error}} + \underbrace{R(c_n) - R(c^*)}_{\text{estimation error}}.$$

- The **approximation error** $R(c^*) - R^*$ decreases when the size of the set of candidates $\mathcal{C}$ increases. If it contains the Bayes classifier ($t \in \mathcal{C}$), this error is equal to zero.
- What about the **estimation error** $R(c_n) - R(c^*)$?
  i.e. the error due to the minimization of the empirical risk $R_n(c) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{c(X_i) \neq Y_i\}}$ instead of the actual one $R(c) = \mathbb{E}\left[\mathbb{1}_{\{c(X) \neq Y\}}\right]$?

Bound on the estimation error

Approximation and estimation errors

$$R(c_n) - R^* = \underbrace{R(c^*) - R^*}_{\text{approximation error}} + \underbrace{R(c_n) - R(c^*)}_{\text{estimation error}}.$$

$$\begin{aligned} R(c_n) &= R(c_n) - R(c^*) + R(c^*) \\ &\leqslant R(c_n) - R(c^*) + R(c^*) + R_n(c^*) - R_n(c_n) \quad (\text{since } R_n(c^*) - R_n(c_n) \geqslant 0) \\ &\leqslant |R(c_n) - R_n(c_n)| + |R_n(c^*) - R(c^*)| + R(c^*) \leqslant 2 \sup_{c \in \mathcal{C}} |R(c) - R_n(c)| + R(c^*). \end{aligned}$$

Bound for the estimation error

With probability at least $1 - \delta$, $\quad R(c_n) - R(c^*) \leqslant 2\sqrt{\dfrac{\ln(N) + \ln\left(\frac{2}{\delta}\right)}{2n}}.$

While $R(c^*)$ decreases with the size of the class $\mathcal{C}$, the bound increases ($\#\mathcal{C} = N$).

Infinite case

Let us consider a countable set of classifiers $\mathcal{C} = (c_i)_{i\in\mathbb{N}}$. Let us assume that we know the probability distribution on the classifiers computed by the considered algorithm: $\forall c \in \mathcal{C}$, $p(c) \in [0,1]$. We take the risk $\delta(c) = \delta p(c)$, $\forall c \ \in \mathcal{C}$.

$$\mathbb{P}\left(\exists c \in \mathcal{C} \;\middle|\; |R_n(c) - R(c)| > \sqrt{\frac{\ln\left(\frac{2}{\delta(c)}\right)}{2n}}\right) = \mathbb{P}\left(\cup_{c\in\mathcal{C}}\left\{|R_n(c) - R(c)| > \sqrt{\frac{\ln\left(\frac{2}{\delta(c)}\right)}{2n}}\right\}\right)$$

$$\leqslant \sum_{c\in\mathcal{C}} \mathbb{P}\left(|R_n(c) - R(c)| > \sqrt{\frac{\ln\left(\frac{2}{\delta(c)}\right)}{2n}}\right) \leqslant \sum_{c\in\mathcal{C}} \delta(c) = \sum_{c\in\mathcal{C}} \delta p(c) = \delta$$

A bound depending on a distribution on classifiers $p(c) > 0$ and $\sum_{c\in\mathcal{C}} p(c) = 1$.

With probability at least $1-\delta$, $|R_n(c) - R(c)| \leqslant \sqrt{\frac{\ln\left(\frac{2}{\delta(c)}\right)}{2n}} = \sqrt{\frac{\ln\left(\frac{1}{p(c)}\right)+\ln\left(\frac{2}{\delta}\right)}{2n}}$.

Note that we get back the finite case's bound by using a uniforme probability $p(c) = \frac{1}{N}$.

Figure: Vladimir Vapnik & Yann Lecun

Vapnik-Chervonenkis

For any $\delta > 0$, with probability at least $1 - \delta$,

$$\forall c \in \mathcal{C}, R(c) \leqslant R_n(c) + 2\sqrt{2\frac{\ln\left(S_{\mathcal{C}}(2n)\right) + \ln\left(\frac{2}{\delta}\right)}{n}}.$$

with $S_{\mathcal{C}}(n) = \sup_{(x_i, y_i)_{i=1}^n \in (\mathbb{R} \times \{y_1, y_2\})^n} \#\left\{\left(f_c(x_i, y_i)\right)_{i=1}^n \in \{0,1\}^n \mid c \in \mathcal{C}\right\}$, the maximum number of ways into which n points can be classified by the function class $\mathcal{F}$.

Since $S_{\mathcal{C}}(n) = \#A$ with $A \subset \{0,1\}^n$, $S_{\mathcal{C}}(n) \leqslant 2^n$.

VC dimension

Examples

The set of half-planes in $\mathbb{R}^d$ can shatter a set of $d+1$ points but no set of $d+2$ points: the VC dimension is $d+1$ + figure + Vayatis' examples

Lemma – Symmetrization

For any $t > 0$, such that $nt^2 \geqslant 2$,

$$\mathbb{P}\left(\sup_{c\in\mathcal{C}}\Big(R(c) - R_n(c)\Big) \geqslant t\right) \leqslant 2\mathbb{P}\left(\sup_{c\in\mathcal{C}}\Big(R_n(c) - R'_n(c)\Big) \geqslant \frac{t}{2}\right),$$

where $R'_n(c) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}_{\{c(X'_i)\neq Y'_i\}}$ is computed with an extra dataset $(X'_i, Y'_i)_{i=1}^n$ (called ghost dataset) independent from $(X_i, Y_i)_{i=1}^n$ (used for $R_n(c)$).

Proof – 1

For $c \in \mathcal{C}$, "$R(c) - R_n(c) \geqslant t$ and $R'_n(c) - R(c) \geqslant -\frac{t}{2}$" $\Rightarrow R'_n(c) - R_n(c) \geqslant \frac{t}{2}$,

$$\text{i.e. } \left\{R(c) - R_n(c) \geqslant t\right\} \cap \left\{R(c) - R'_n(c) \leqslant \frac{t}{2}\right\} \subseteq \left\{R'_n(c) - R_n(c) > \frac{t}{2}\right\},$$

or also $\mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}}\mathbb{1}_{\{R(c)-R'_n(c)\leqslant\frac{t}{2}\}} \leqslant \mathbb{1}_{\{R'_n(c)-R_n(c)\geqslant\frac{t}{2}\}}$.

Lemma – Symmetrization

$$\mathbb{P}\left(\sup_{c\in\mathcal{C}}\Big(R(c)-R_n(c)\Big)\geqslant t\right)\leqslant 2\mathbb{P}\left(\sup_{c\in\mathcal{C}}\Big(R_n(c)-R_n'(c)\Big)\geqslant\frac{t}{2}\right).$$

Proof – 2

For $c\in\mathcal{C}$, $\mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}}\mathbb{1}_{\left\{R(c)-R_n'(c)\leqslant\frac{t}{2}\right\}}\leqslant\mathbb{1}_{\left\{R_n'(c)-R_n(c)\geqslant\frac{t}{2}\right\}}$.
Let us assume that c depends on $(X_i,Y_i)_{i=1}^n$ but not on the ghost dataset $(X_i',Y_i')_{i=1}^n$.
By applying the conditional expectation, conditioned by the variables $(X_i,Y_i)_{i=1}^n$,

$$\mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}}\mathbb{P}\left(R(c)-R_n'(c)\leqslant\frac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right)\leqslant\mathbb{P}\left(R_n'(c)-R_n(c)\geqslant\frac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right),$$

since $\mathbb{E}[f(X)Z\,|\,X]=f(X)\mathbb{E}[Z\,|\,X]$ (see Prop. 1.31 in [GS20]).

Lemma – Symmetrization

$$\mathbb{P}\left(\sup_{c\in\mathcal{C}}\left(R(c)-R_n(c)\right)\geqslant t\right)\leqslant 2\mathbb{P}\left(\sup_{c\in\mathcal{C}}\left(R_n(c)-R'_n(c)\right)\geqslant\frac{t}{2}\right).$$

Proof – 3

$\mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}}\mathbb{P}\left(R(c)-R'_n(c)\leqslant\frac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right)\leqslant\mathbb{P}\left(R'_n(c)-R_n(c)\geqslant\frac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right)$.
In addition, using Chebyshev's inequality (slide 95)

$$\mathbb{P}\left(R(c)-R'_n(c)>\tfrac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right)$$
$$\leqslant\mathbb{P}\left(|R(c)-R'_n(c)|\geqslant\tfrac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right)\leqslant\frac{Var(R'_n(c))}{\left(\frac{t}{2}\right)^2}=\frac{4Var\left(\mathbb{1}_{\{c(X')\neq Y'\}}\right)}{nt^2}\leqslant\frac{1}{nt^2}$$

(since the variance of a random variable with range $[0,1]$ is less than $\frac{1}{4}$: $\mathbb{E}\left[X^2\right]-(\mathbb{E}[X])^2\leqslant\mathbb{E}[X](1-\mathbb{E}[X])\leqslant\frac{1}{4}$).

$$\Rightarrow\mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}}\left(1-\frac{1}{nt^2}\right)\leqslant\mathbb{P}\left(R'_n(c)-R_n(c)\geqslant\frac{t}{2}\,\middle|\,(X_i,Y_i)_{i=1}^n\right).$$

Lemma – Symmetrization

For any $t > 0$, such that $nt^2 \geqslant 2$,
$\mathbb{P}\left(\sup_{c\in\mathcal{C}}\left(R(c) - R_n(c)\right) \geqslant t\right) \leqslant 2\mathbb{P}\left(\sup_{c\in\mathcal{C}}\left(R_n(c) - R'_n(c)\right) \geqslant \frac{t}{2}\right).$

Proof – 4

$\mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}}\left(1 - \frac{1}{nt^2}\right) \leqslant \mathbb{P}\left(R'_n(c) - R_n(c) \geqslant \frac{t}{2} \mid (X_i, Y_i)_{i=1}^n\right).$
In addition, $nt^2 \geqslant 2 \Rightarrow \left(1 - \frac{1}{nt^2}\right) \geqslant \frac{1}{2}$

$$\Rightarrow \mathbb{1}_{\{R(c)-R_n(c)\geqslant t\}} \leqslant 2\mathbb{P}\left(R'_n(c) - R_n(c) \geqslant \frac{t}{2} \mid (X_i, Y_i)_{i=1}^n\right).$$

By applying the expectation $\mathbb{E}[.]$,

$$\Rightarrow \mathbb{P}\left(R(c) - R_n(c) \geqslant t\right) \leqslant 2\mathbb{P}\left(R'_n(c) - R_n(c) \geqslant \frac{t}{2}\right).$$

We get the "$\sup_{c\in\mathcal{C}}$" by using increasing sequences of events $\left\{R'_n(c) - R_n(c) \geqslant \frac{t}{2}\right\}$ and $\left\{R(c) - R_n(c) > t\right\}$ (Prop. 1.1 in [GS20]) whose limits are the "$\sup_{c\in\mathcal{C}}$" with classifiers depending on $(X_i, Y_i, X'_i, Y'_i)_{i=1}^n$ on the right, and depending on $(X_i, Y_i)_{i=1}^n$ on the left. □

Vapnik-Chervonenkis

For any $\delta > 0$, with probability at least $1 - \delta$,

$$\forall c \in \mathcal{C}, R(c) \leqslant R_n(c) + 2\sqrt{2\frac{\ln\left(S_{\mathcal{C}}(2n)\right) + \ln\left(\frac{2}{\delta}\right)}{n}}.$$

with $S_{\mathcal{C}}(n)$ the maximum number of ways into which n points can be classified by $\mathcal{C}$:
$S_{\mathcal{C}}(n) = \max_{(x_i,y_i)_{i=1}^n} \#\mathcal{F}\Big((x_i,y_i)_{i=1}^n\Big)$ with $\mathcal{F}\Big((x_i,y_i)_{i=1}^n\Big) = \Big\{(f_c(x_i,y_i))_{i=1}^n \in \{0,1\}^n \,\Big|\, c \in \mathcal{C}\Big\}$.

Proof – 1

Note that $\left\{\sup_{c\in\mathcal{C}}\Big(R_n(c) - R'_n(c)\Big) \geqslant \frac{t}{2}\right\} = \left\{\sup_{c\in\mathcal{C}}\frac{1}{n}\sum_{i=1}^n\Big(f_c(X_i,Y_i) - f_c(X'_i,Y'_i)\Big) \geqslant \frac{t}{2}\right\}$

$= \left\{\exists c \in \mathcal{C} \mid \frac{1}{n}\sum_{i=1}^n\Big(f_c(X_i,Y_i) - f_c(X'_i,Y'_i)\Big) \geqslant \frac{t}{2}\right\} = \cup_{c\in\mathcal{C}}\left\{\Big(R_n(c) - R'_n(c)\Big) \geqslant \frac{t}{2}\right\}$

$= \cup_{f\in\mathcal{F}\left((X_i,Y_i)_{i=1}^n,(X'_i,Y'_i)_{i=1}^n\right)}\left\{\frac{1}{n}\sum_{i=1}^n\Big(f_i - f_{i+n}\Big) \geqslant \frac{t}{2}\right\}$.

Vapnik-Chervonenkis

With probability $1-\delta, \forall c \in \mathcal{C}, R(c) \leqslant R_n(c) + 2\sqrt{2\dfrac{\ln\left(S_{\mathcal{C}}(2n)\right) + \ln\left(\frac{2}{\delta}\right)}{n}}$.

Proof – 2

By using the Symmetrization's Lemma (slide 122),

$$\mathbb{P}\left(\sup_{c\in\mathcal{C}}\left(R(c) - R_n(c)\right) \geqslant t\right) \leqslant 2\mathbb{P}\left(\sup_{c\in\mathcal{C}}\left(R_n(c) - R'_n(c)\right) \geqslant \frac{t}{2}\right)$$

$$= 2\mathbb{P}\left(\cup_{f\in\mathcal{F}\left((X_i,Y_i)_{i=1}^n,(X'_i,Y'_i)_{i=1}^n\right)}\left\{\frac{1}{n}\sum_{i=1}^n\left(f_i - f_{i+n}\right) \geqslant \frac{t}{2}\right\}\right)$$

$$\leqslant 2\mathbb{P}\left(\cup_{f\in\mathcal{F}\left((X_i,Y_i)_{i=1}^n,(X'_i,Y'_i)_{i=1}^n\right)}\left\{\frac{1}{n}\sum_{i=1}^n\left(f_i - f_{i+n}\right) \geqslant \frac{t}{2}\right\}\right)$$

□

Institut Supérieur de l'Aéronautique et de l'Espace

10 avenue Édouard Belin – BP 54032

31055 Toulouse Cedex 4 – France

Phone: +33 5 61 33 80 80

www.isae-supaero.fr

isae
Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO