Introduction to machine learning

Eric Gaussier

LIG - MIAI
Univ. Grenoble Alpes
Eric.Gaussier@imag.fr

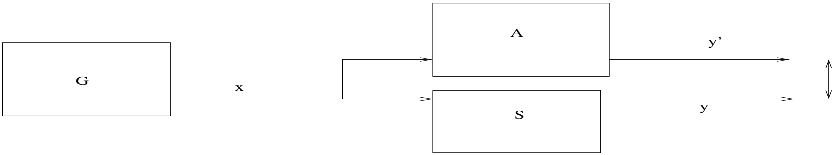
Table des matières

What's machine learning ?

- Unsupervised learning
- Supervised learning (weakly supervised, semi-supervised)
- Reinforcement learning

Focus today on supervised learning

Supervised learning (1)



- *input* x, *output* y - $y = f^*(x)$, f^* (function/process/algorithm) unknown
- One observes a series of input-output pairs
- From these observations, the learner A aims to identify, within a family of functions, the best function to relate inputs to outputs

Supervised learning (2)

Input : training set

- $\mathcal{D} = ((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)}))$
- x real vector - $x \in \mathbb{R}^p$
- $y \in \mathcal{Y}$ - binary classification : $\mathcal{Y} = \{0, 1\}$; simple linear regression : $\mathcal{Y} \subseteq \mathbb{R}$

Learning model

- Family of functions $\mathcal{F}$ - example : set of linear functions
- Cost function : measures the error made by the learned model (error between y, desired output, and the predicted output $y' = f(x)$, $f \in \mathcal{F}$)
- Objective function : function to be optimized (minimized) - cost function plus additional terms (regularization)
- Optimization method (to identify the "best" function acc. to the objective function)

How to measure the quality of a learned model ?

Loss (cost) function to evaluate the errors made by a learned model on known input-output pairs

Loss function

$$L : \mathcal{Y}x\mathcal{Y} \rightarrow \mathbb{R}^{+}, \text{such that } L(y, y') > 0 \text{ for } y \neq y'$$

Illustration

- $0 - 1$ loss :

$$L(y, y') = \begin{cases} 0 & \text{if } y = y', \\ 1 & \text{otherwise} \end{cases}$$

- Quadratic loss :

$$L(y, y') = (y - y')^2$$

Selecting $f \in \mathcal{F}$

Looking for the function that minimizes the prediction errors

1. *Ideal case* - Functional risk minimization :

$$\arg\min_{f \in \mathcal{F}} \underbrace{\int_x \int_y P(x,y) L(y, f(x)) dx dy}_{R(f) = \mathbb{E}_{P(x,y)}[L(y, f(x))]}$$

2. *Realistic case* - Empirical risk minimization :

$$\arg\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))}_{\text{Remp}(f;\mathcal{D})} = \arg\min_{f \in \mathcal{F}} \text{Remp}(f; \mathcal{D})$$

Selecting $f \in \mathcal{F}$

Looking for the function that minimizes the prediction errors

1. *Ideal case* - Functional risk minimization :

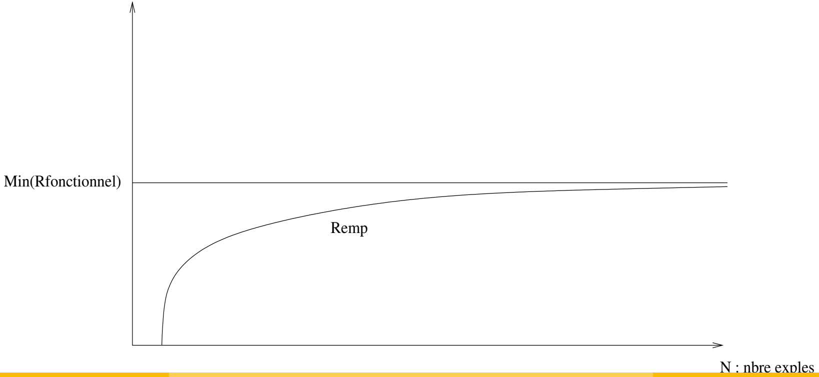
$$\underset{f \in \mathcal{F}}{\arg\min} \underbrace{\int_x \int_y P(x,y) L(y, f(x)) dxdy}_{R(f) = \mathbb{E}_{P(x,y)}[L(y, f(x))]}$$

2. *Realistic case* - Empirical risk minimization :

$$\underset{f \in \mathcal{F}}{\arg\min} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))}_{\text{Remp}(f;\mathcal{D})} = \underset{f \in \mathcal{F}}{\arg\min} \, \text{Remp}(f; \mathcal{D})$$

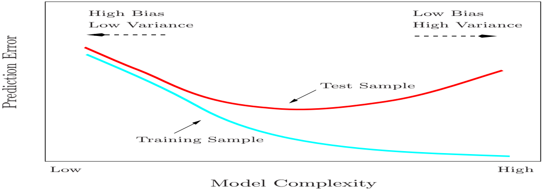
Intuitive justification of the empirical risk minimization prinicple

For $f \in \mathcal{F}$ fixed, the empirical risk tends towards the true risk when the number of training examples tends to infinity



However, in practice ...

... when the number of examples is limited :

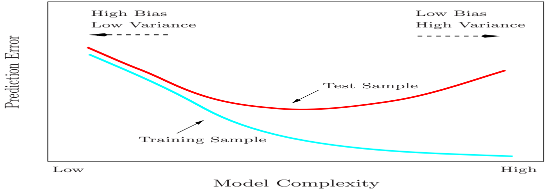


Solution : $\arg\min_{f \in \mathcal{F}} \text{Remp}(f) + \lambda\Omega(f)$
$\Omega(f)$ is a measure of the complexity of f

Image from "Elements of statistical learning". Hastie, Tibshirani, Friedman. Springer

However, in practice ...

... when the number of examples is limited :



Solution : $\arg\min_{f \in \mathcal{F}} \text{Remp}(f) + \lambda\Omega(f)$
$\Omega(f)$ is a measure of the complexity of f

Image from "Elements of statistical learning". Hastie, Tibshirani, Friedman. Springer

Regularization : complexity, knowledge, constraints

$$\arg\min_{f \in \mathcal{F}} \text{Remp}(f) + \overbrace{\underbrace{\lambda}_{\text{regularization parameter}} \quad \Omega(f)}^{\text{regularization}}$$

Regularization allows one to :

- Avoid selecting too complex functions
- Integrate prior knowledge and constraints

Learning model (1)

A learning model :

- Has access to a set of functions $\mathcal{F}$
- Selects the "best" function from the training set and the objective function defined by the user/designer
- Operates this selection following optimization methods (*stochastic gradient descent (SGD)*)

Learning model (2)

The user/designer defines or selects :

- The loss function adapted to the task addressed
- The regularization terms (L_1, L_2, ... regularization)

What about original representation of examples ?

Learning model (2)

The user/designer defines or selects :

- The loss function adapted to the task addressed
- The regularization terms (L_1, L_2, ... regularization)

What about original representation of examples ?

Feature engineering vs representation learning

1. Before deep learning : huge effort devoted to pre-processing and the selection and extraction of appropriate features

2. Deep learning : adequate choice of the architecture that will lead to learn an appropriate representation (still need original representation)

Which family of functions ?

Let R^* be the minimal functional over all possible functions. Let $R_{\mathcal{F}}(f_{\text{min}})$ be the minimal functional risk over the functions in $\mathcal{F}$ and let $R_{\mathcal{F}}(f)$ be the functional risk of the function f in $\mathcal{F}$.
One has :

$$R_{\mathcal{F}}(f) - R^* = \underbrace{(R_{\mathcal{F}}(f) - R_{\mathcal{F}}(f_{\text{min}}))}_{\text{estimation error}} + \underbrace{(R_{\mathcal{F}}(f_{\text{min}}) - R^*)}_{\text{approximation error}}$$

Remark (this is just a trend !)

- *The simpler the family is, the smaller the estimation error and the bigger the approximation error are*
- *Inversely, the more complex the family is, the bigger the estimation error and the smaller the approximation error are*

Which family of functions ?

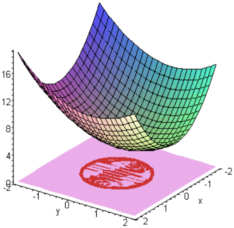
Let R^* be the minimal functional over all possible functions. Let $R_{\mathcal{F}}(f_{\text{min}})$ be the minimal functional risk over the functions in $\mathcal{F}$ and let $R_{\mathcal{F}}(f)$ be the functional risk of the function f in $\mathcal{F}$.
One has :

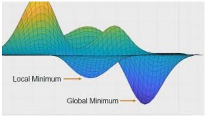
$$R_{\mathcal{F}}(f) - R^* = \underbrace{(R_{\mathcal{F}}(f) - R_{\mathcal{F}}(f_{\text{min}}))}_{\text{estimation error}} + \underbrace{(R_{\mathcal{F}}(f_{\text{min}}) - R^*)}_{\text{approximation error}}$$

Remark (this is just a trend !)

- *The simpler the family is, the smaller the estimation error and the bigger the approximation error are*
- *Inversely, the more complex the family is, the bigger the estimation error and the smaller the approximation error are*

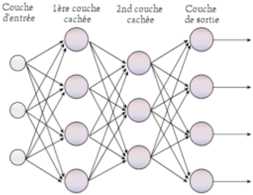
Tradeoff estimation-approximation





Multilayer perceptron - MLP (1)

- $\mathbf{y} \in \mathbb{R}^4, \mathbf{x} \in \mathbb{R}^3$
- $\mathbf{y} = f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$
- Depth of the network (number of layers), dimensionality of each layer



MLP (2)

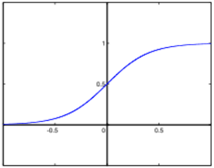
Which functions f^i at each layer ?

Let $\mathbf{h^{i-1}}$ be the input of f^i ($\mathbf{h^0} = \mathbf{x}$) :

$$f^i(\mathbf{h^{i-1}}) = \sigma(\mathbf{W}^i\mathbf{h^{i-1}} + \mathbf{b}^i)$$

with $\mathbf{h^{i-1}} \in \mathbb{R}^{p_i}$, $\mathbf{W}^i \in \mathbb{R}^{p_{i+1} \times p_i}$, $\mathbf{b}^i \in \mathbb{R}^{p_{i+1}}$

The function σ is a non-linear (in general) function called an activation function (sigmoïd, tanh, RELU)



MLP (3)

- An MLP is a universal approximator
- Rich family of functions : good approximation but estimation more complex
- Number of parameters
- Number of training examples
- Regularization : L_1-, L_2-, ... norm, dropout, max pooling
- Quality of local minima ?

Table des matières

How to evaluate a learned model ?

Train/test split

- Size of the annotated set, the training and test sets
- Train/test plit : 80-20, 70-30
- Random split, sometimes with constraints (time series)
- The model is learned on the training set and evaluated on the test set - you should not even glance at the test set

How to evaluate a learned model ?

Train/validation/test split

- Validation set to determine hyperparameter values (degree of a polynomial function, number of neurons on each layer, ...)
- Random split 64-16-20 or 49-21-30
- For possible hyperparameter values (e.g., degree = 1, 2 or 3), learn model on training set, evaluate it on validaiton set
- The select the best hyperparameter values and learn the associated model on train+validation
- Finally evaluate this model on test set - you should not even glance at the test set

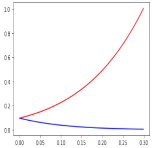
How to evaluate a learned model ?

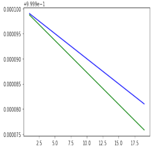
x-flod cross-validation

- Randomly partition data in k groups of equal size $\{g_1, \cdots, g_k\}$ (*k-fold cross-validation*) - $k = 3, 5, 10$
- Construct k sets training-validation-test
  - Set 1 : train=$\{g_1, \cdots, g_{k-2}\}$; valid.=g_{k-1} ; test = g_k
  - Set 2 : train.=$\{g_2, \cdots, g_{k-1}\}$; valid.=g_k ; test = g_1
  - ...
- Training, validation and evaluation on each set as before
- Compute average (over all sets) performance and associated standard deviation
- Advantage : avge, std deviation, and use of all training examples for both training and testing

Some remarks

Scale effects





Significant differences

- Is a system B which improves a system A by 0.008 pt really better ?
- Statistical significance tests

Table des matières

Data annotation is often a costly and difficult process

Annotated data may however be easily available in some contexts

- Machine translation ; pre-training LLMs
- Relevance of a web page for information retrieval
- Objects in images, actions in videos

Data annotation is often a costly and difficult process

Annotated data may however be easily available in some contexts

- Machine translation ; pre-training LLMs
- Relevance of a web page for information retrieval
- Objects in images, actions in videos

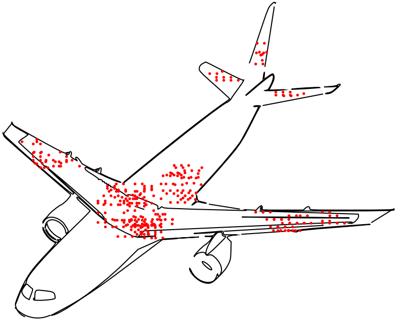


Table des matières

Conclusion

- A rich, reactive domain opened to many actors
- Many questions still open
  - Local minima
  - Number of examples
  - Generalization properties
  - Adversarial examples, ...