Model selection and assessment

Gilles Gasso

INSA Rouen -Department ASI
Laboratory LITIS

December 19, 2019

Plan

The goal

Goal

- $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1\cdots n}$: set of labeled data
- $(\boldsymbol{x}, y) \sim \mathrm{p}(X, Y)$ with $\mathrm{p}(X, Y)$ the joint distribution generally unknown
- Goal : learn from $\mathcal{D}$ a function

$$\begin{aligned} f: \quad \mathcal{X} &\longrightarrow \mathcal{Y} \\ x &\longmapsto \hat{y} = f(\boldsymbol{x}) \end{aligned}$$

that predicts the output $\hat{y}$ associated to each point $\boldsymbol{x} \in \mathcal{X}$

Properties of the learning

- $\forall\ (\boldsymbol{x}_i, y_i) \in \mathcal{D}$, we want f to predict the correct label y_i
- f should correctly predict the labels of unseen sample $\boldsymbol{x}_j$

Example

Example : image classification

Classification methods

- K-NN
- Logistic Regression
- SVM (linear or non-linear)
- . . .

$\Longrightarrow$ Which model to select ? How to asess its ability to generalize to unseen data ?

Loss function

Loss function $\ell(Y, f(X))$

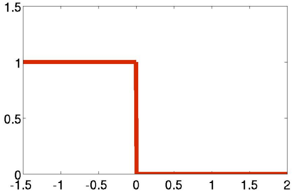
- evaluates how "close" is the prediction $f(\mathbf{x})$ to the true label y
- it penalizes errors: $\ell(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if} \quad y = f(\mathbf{x}) \\ \geq 0 & \text{if} \quad y \neq f(\mathbf{x}) \end{cases}$

For binary classification

- We suppose $\mathcal{Y} = \{-1, 1\}$
- 0 - 1 cost

$$\ell(y, f(\mathbf{x})) = \mathbb{I}_{yf(\mathbf{x}) \leq 0} = \begin{cases} 0 & \text{if} \quad yf(\mathbf{x}) > 0 \\ 1 & \text{if} \quad yf(\mathbf{x}) \leq 0 \end{cases}$$

measures the number of classification errors



Risk function and learning

Risk function

Assesses the expected error (generalization ability) of f

$$\begin{aligned} R(f) &= \mathbb{E}_{X,Y}\ell(Y, f(X)) \\ R(f) &= \int_{\mathcal{X},\mathcal{Y}} \ell(y, f(\boldsymbol{x}))\mathrm{p}(\boldsymbol{x}, y)d\boldsymbol{x}dy \end{aligned}$$

Statistical learning problem

Find the function f^* that minimises $R(f)$

$$f^* = \text{argmin}_f \mathbb{E}_{X,Y}\ell(Y, f(X))$$

However

f^* is not attainable as $\mathbb{P}(X, Y)$ is unknown

Empirical risk

We only have access to a finite set of samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots n}$.

Define the empirical risk

$$R_{\mathrm{n}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i))$$

Empirical risk minimization

- We are looking for a decision function

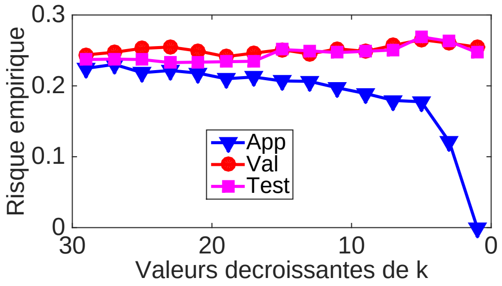
$$f_n \quad = \quad \mathrm{argmin}_f R_{\mathrm{n}}(f)$$

- $R_{\mathrm{n}}(f_n)$ is the empirical risk corresponding to f_n. It is an approximation of the real risk $R(f_n) = \mathbb{E}_{X,Y}\ell(Y, f_n(X))$

Empirical risk and over-fitting

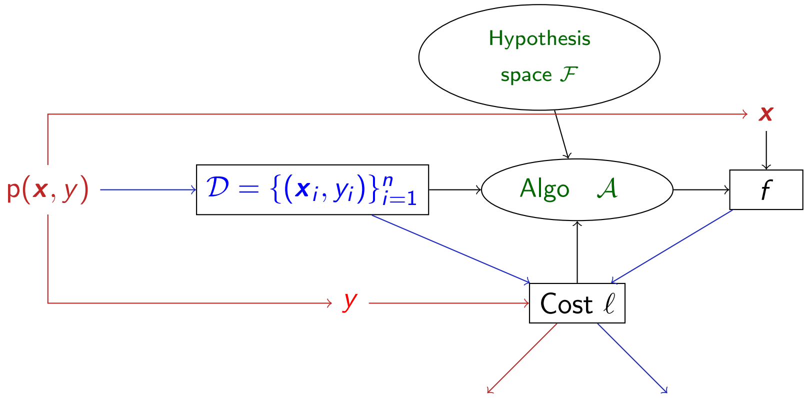
- Should we choose f based on $R_n(f_n)$? NO !
- as we can design a sufficiently complex function f_n such that $R_n(f_n) \rightarrow 0$ but with high risk $R(f_n)$

K-NN classification function



$\Longrightarrow$ Control the complexity of the function f

The paradigm of statistical learning


Hypothesis space $\mathcal{F}$
x
$p(x, y)$
$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
Algo $\mathcal{A}$
f
y
Cost ℓ

$$\mathbb{P}\Big(\underset{\mathbb{E}_{x,y}\ell(y, f(x))}{\overset{R(f)}{=}} > \underset{\frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)}{\overset{\text{Error } R_n(f)}{=}} + \epsilon \Big) \leq \delta$$

With given $\mathcal{D}$, find a model f in a family $\mathcal{F}$ (linear, kernel SVM . . .) with good generalization properties

Why the learning is possible

Supremum on generalization error

Let's $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots n}$ the dataset. Let $\mathcal{F}$ be a space of functions. For each $f \in \mathcal{F}$, with probability $1-\delta$ we have

$$R(f) \leq R_{\mathrm{n}}(f) + \mathcal{O}\left(\sqrt{\frac{h}{n}\log\frac{2en}{h} + \frac{\log 2/\delta}{n}}\right)$$

$h > 0$ measures the "complexity" of the functions class $\mathcal{F}$

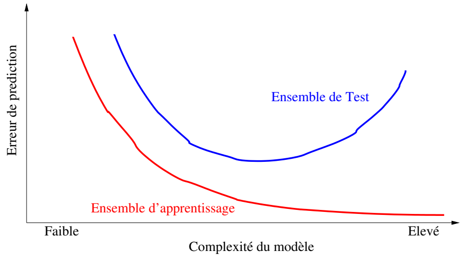
- Generalization occurs whenever $h < \infty$
- Bigger is n better it is ($n >> h$: the number of data increases with model complexity)
- Linear model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ with $w \in \mathbb{R}^d$, $h = d+1$

Illustration

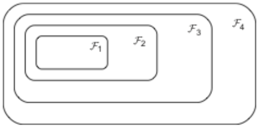
Generalization / over-fitting

$$R(f) \leq \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) + \text{term}(n, h(\mathcal{F}))$$

- $R_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i)$ is not a good estimator of generalization ability
- Over-fitting appears with the increasing complexity of f



Complexity control: regularisation



Let $k_1 < k_2 < k_3 < \cdots$

We define $\mathcal{F}_j = \{f : \Omega(f) \leq k_j\}$

$\Omega(f)$: regularisation function

Example : $\Omega(f) = \|f\|^2$

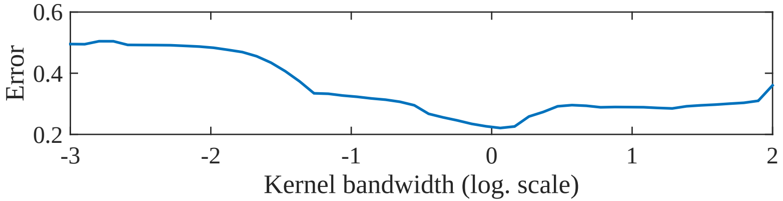
Minimization of the regularized empiric risk

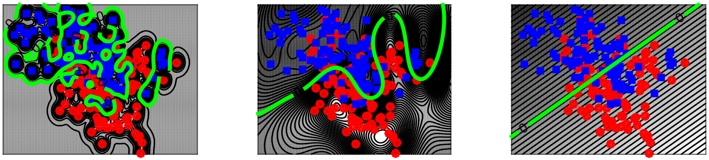
$$\min_f \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i) + \lambda \Omega(f)$$

- $\lambda > 0$: regularization hyper-parameter
- $\lambda >> 1 \rightarrow$ we encourage f to be of low complexity

Example : SVM $\min_f \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i) + \lambda \|f\|^2$ with cost function $\ell(y, f(\boldsymbol{x})) = \max(0, 1 - yf(\boldsymbol{x}))$ and $\lambda = 1/C$

Illustration: influence of model's hyper-parameters





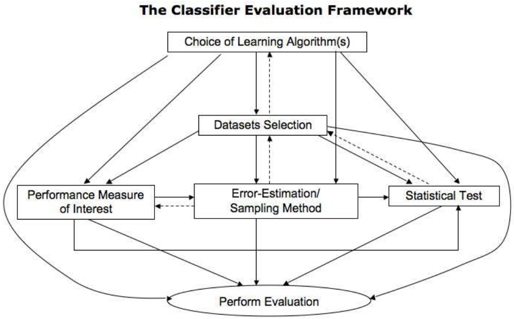
σ too small nice σ σ too large

- The choice of the hyper-parameter's value (hence of the model) impacts the quality of the prediction

Model selection and evaluation

Raised issues

- Model evaluation : what measure(s) of performance?
- Estimation of the generalisation capacity of the model
- Practical model selection procedures


The Classifier Evaluation Framework
Choice of Learning Algorithm(s)
Datasets Selection
Performance Measure of Interest
Error-Estimation/ Sampling Method
Statistical Test
Perform Evaluation

Plan

Assessing the quality of a model

The confusion matrix

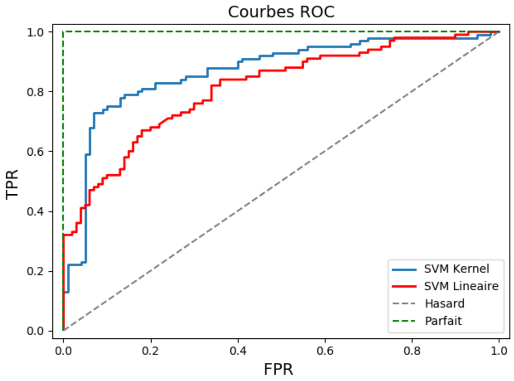
A matrix showing the predicted and actual classifications. A confusion matrix is of size $p \times p$, where p is the number of classes.

Predicted / Actual	**Positive**	**Negative**
Positive	TP	FP
Negative	FN	TN
	P = TP + FN	N = FP + TN

- Error rate $=$ (FP $+$ FN)/(P $+$ N) ($\searrow\searrow$)
- Accuracy $=$ 1 - Error rate $=$ (TP $+$ TN)/(P $+$ N) ($\nearrow\nearrow$)
- Precision $=$ TP/(TP $+$ FP)
- Recall, Sensitivity $=$ TP/P
- Specificity $=$ FP/N
- F-Measure $= 2\dfrac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ ($\nearrow\nearrow$)

ROC Curve

- It's the curve TPR = fonction(FPR)
- Allows graphical comparison of different models



Measure of performances

Area Under the ROC Curve (AUC)

- Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i = 1)\}_{i=1}^{P} \cup \{(\boldsymbol{x}_j, y_j = -1)\}_{i=1}^{N}$ and f be the decision function. The AUC is defined by

$$\text{AUC} = \sum_{i=1}^{P} \sum_{j=1}^{N} \frac{\mathbb{I}\left[f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)\right] + 0.5\, \mathbb{I}\left[f(\boldsymbol{x}_i) = f(\boldsymbol{x}_j)\right]}{P \times N}$$

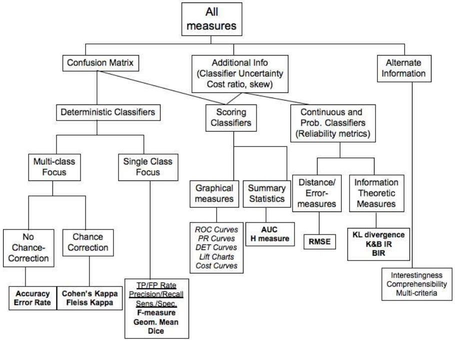
  with $\mathbb{I}$ the indicator function

- AUC is between 0 and 1 $(\nearrow\nearrow)$
- Favours the decision function such that $f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$ $\forall\ (y_i = 1, y_j = -1)$

Other performance measures

- Many performance measures exist
- Each classifier may be the best one according to a specific measure
- Keep in mind that your model may fail according to another measure

→ Choose wisely according to your problematic


All measures
Confusion Matrix
Additional Info (Classifier Uncertainty Cost ratio, skew)
Alternate Information
Deterministic Classifiers
Scoring Classifiers
Continuous and Prob. Classifiers (Reliability metrics)
Multi-class Focus
Single Class Focus
Graphical measures
Summary Statistics
Distance/ Error-measures
Information Theoretic Measures
No Chance-Correction
Chance Correction
ROC Curves PR Curves DET Curves Lift Charts Cost Curves
AUC H measure
RMSE
KL divergence K&B IR BIR
Accuracy Error Rate
Cohen's Kappa Fleiss Kappa
TP/FP Rate Precision/Recall Sens./Spec. F-measure Geom. Mean Dice
Interestingness Comprehensibility Multi-criteria

N. Japkowicz & M. Shah, "Evaluating Learning Algorithms: A Classification Perspective", Cambridge University Press, 2011

The model' generalization

- Let f be a decision-making function developed using the data $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1\cdots n}$
- We are looking at $R(\mathcal{D}_\infty, f)$ the theoretical performance of f on all possible future data

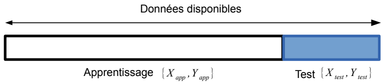
Generalisation Capacity

Capacity of f to perform well (measured with one of the previous metrics) when tested on data other than those used for training

How to estimate $R(\mathcal{D}_\infty, f)$ in practice ?

Paradigm test set/training set

Randomly split $\mathcal{D}_n$ into two disjoints sets $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$


Données disponibles
Apprentissage $\{X_{app}, Y_{app}\}$
Test $\{X_{test}, Y_{test}\}$

- $\mathcal{D}_{train} = \{(\boldsymbol{x}_i, y_i)\}_{i=1 \cdots n_{train}}$: data used for training f
- $\mathcal{D}_{test} = \{(\boldsymbol{x}_i, y_i)\}_{i=1 \cdots n_{test}}$: data used to evaluate the generalization capacity of f

Remark

- Bigger n_{train} is, better the training
- Bigger n_{test} is, better the estimation of performance is f
- $\mathcal{D}_{test}$ is used only one time !

Error bars on Bernoulli trials

Hypothesis

My new method classifies well 90 (n_S) examples over 100 (n). 10 (n_F) examples are mis-classified. What is my level of confidence?

Level of confidence α

success probability : $\widehat{p} = 0.9$

$$\hat{p}_\alpha = \widehat{p} \pm z\sqrt{\frac{\widehat{p}\,(1-\widehat{p})}{n}} = \frac{n_S}{n} \pm \frac{z}{n}\sqrt{\frac{n_S n_F}{n}}$$

with z is the $1 - \frac{\alpha}{2}$ quantile of a standard normal distribution.

- Consider $\alpha = 0.95$,
- z = scipy.stats.norm.ppf(0.975)∗np.sqrt(0.9∗(1−0.9)/100)
  $\widehat{p}_\alpha = 0.9 \pm 0.059$
- ie. 95% of time: $0.84 < \widehat{p} < 0.96$

http://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval

To improve the estimate

Dataset size

- If you increase the number of runs, your confidence increases.
- Check the confidence interval

Increase n

- Random Subsampling (The repeated holdout method)
- K-Fold Cross-Validation ($K = 10, 5, 2, \dots$)
- Leave-one-out Cross-Validation ($K = n$)
- Bootstrap (each sample can be in differents subsets)

Error bars: the gaussian approximation

The *repeated* holdout method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
- In each iteration, use a different random splitting
- Average the error rates on the different iterations

Statistics

- Mean error rate $e = \dfrac{1}{K}\displaystyle\sum_{k=1}^{K} e_k$
- Variance $\widehat{\sigma}^2 = \dfrac{1}{K-1}\displaystyle\sum_{k=1}^{K}(e_k - e)^2$
- Confidence: $e \pm t_{\alpha/2,K-1}\,\frac{\widehat{\sigma}^2}{\sqrt{K}}$

  $(t_{0.025,9} = 2.262)$

Conclusion

Good habits

- Simulate real conditions
- Avoid test set bias by adding it within learning procedure
- Look for stability rather than performance

What to do next ?

- What is the best method for my problem?
- How good is my learning algorithm?

Comparing two algorithms: Mc Nemar's test

Algo 1 \ Algo 2	Classified Well	Classified Wrong
Well	e_{00}	e_{01}
Wrong	e_{10}	e_{11}

Null Hypothesis H_0: No differences

$$\text{We expect :}\begin{cases} e_{00} + e_{10} = e_{00} + e_{01} \\ e_{11} + e_{10} = e_{11} + e_{01} \end{cases}$$

- $H_0 : e_{10} = e_{01}$

$$\frac{(e_{10} - e_{01})^2}{e_{10} + e_{01}} \sim \chi^2_1$$

- python: in statsmodel

J. L. Fleiss (1981) Statistical Methods for Rates and Proportions. Second Edition. Wiley.

Plan

Model Selection

Problem

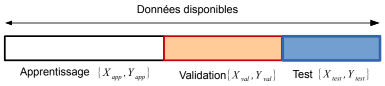
- Given a set of models $\mathcal{F} = \{f_1, f_2, \cdots\}$, choose the decision function giving the best performances on future data

Examples of function choice by classification type

- K-NN :choice of K
- Sparse Logistic Regression : number of selected variables
- SVM : choice of the hyper-parameter C, kernel tuning
- $\cdots$

Validation set

How to choose the "best" model without testing on $\mathcal{D}_{test}$?


Données disponibles
Apprentissage $\{X_{app}, Y_{app}\}$
Validation $\{X_{val}, Y_{val}\}$
Test $\{X_{test}, Y_{test}\}$

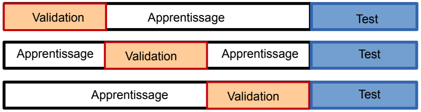
1. Randomly split $\mathcal{D}_n = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$
2. Train each possible model on $\mathcal{D}_{train}$
3. Evaluate the performance on $\mathcal{D}_{val}$
4. Select the model with the best performance on $\mathcal{D}_{val}$
5. Test the selected model on $\mathcal{D}_{test}$

Remark

- $\mathcal{D}_{test}$ is used only one time !

K-fold validation

What if the size of $\mathcal{D}_n$ is small ?



1. Randomly split $\mathcal{D}_n = \mathcal{D}_{train} \cup \mathcal{D}_{test}$
2. Then split randomly $\mathcal{D}_{train} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$ in K sets
3. For $k = 1$ to K
    1. Put aside $\mathcal{D}_k$
    2. Train the model f on the $K-1$ remaining sets
    3. Evaluate its performance R_k on generalizing to $\mathcal{D}_k$
4. Average the K measures of performance R_k

Practical procedure (1)

General Methodology

Input : hyper-parameters family $\mathcal{F} = \{p_1, p_2, \cdots\}$ and $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1\cdots n}$

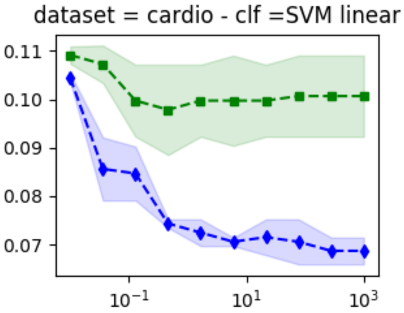
1. Split data $(\mathcal{D}_{train\_val}, \mathcal{D}_{test}) \leftarrow \texttt{SplitData}(\mathcal{D}, \texttt{options})$
2. Selecting the best model : $f^* \leftarrow \texttt{Selection}(\mathcal{D}_{train\_val}, \mathcal{F})$
3. $Perf \leftarrow \texttt{EvaluerPerf}(\mathcal{D}_{test}, f^*)$

Practical procedure (2nd part)

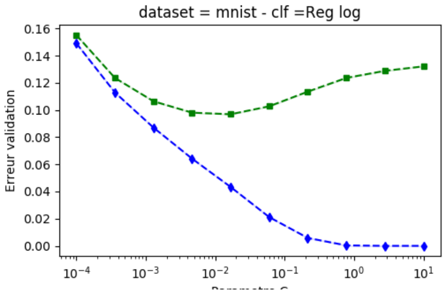
```
function $f^* \leftarrow \texttt{Selection}(\mathcal{D}_{train\_val}, \mathcal{F})$
  1 Split again the dataset $(\mathcal{D}_{train}, \mathcal{D}_{val}) \leftarrow \texttt{SplitData}(\mathcal{D}_{\texttt{train\_val}}, \texttt{options})$
  2 For $f_i \in \mathcal{F}$
      1 Train the model : $f_i \leftarrow \texttt{Model.fit}(\mathcal{D}_{train}, \mathrm{p}_i)$
      2 $Perf(i) \leftarrow \texttt{EvaluerPerf}(\mathcal{D}_{val}, f_i)$
  3 Select the performing model (best hyper-parameter) : $p^* \leftarrow \text{argmin } Perf$
  4 $f^* \leftarrow \texttt{Model.fit}(\mathcal{D}_{train\_val}, \mathrm{p}^*)$
```

Illustration

K-Fold Cross-Validation



Cross-Validation


dataset = mnist - clf =Reg log
Erreur validation
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}