Model Assesment

Formation ENSTA ParisTech
Conférence IA

Florent Chatelain* Olivier Michel*

*GIPSA-lab, Univ. Grenoble Alpes,

2-5 février 2021
ENSTA, France

Performance criterion

We seek a regression function/classification rule f for accurate predictions $f(X)$ of target Y given X.

Problem : accurate ? ?

Solution : Introduction of a loss function :

$$L : \quad \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+,$$

in order to penalize the prediction errors :

$$L(Y, f(X)) \begin{cases} = 0 \text{ if } f(X) = Y, \\ \geq 0 \text{ otherwise,} \end{cases}$$

Performance criterion (Cont'd)

Two useful loss functions

1. Quadratic cost : $L(Y, f(X)) = (Y - f(X))^2$,
2. $0 - 1$ cost :

$$L(Y, f(X)) = \begin{cases} 0 \text{ if } |Y - f(X)| < \epsilon & (\epsilon > 0 \text{ arbitrarily small}), \\ 1 \text{ otherwise} \end{cases}$$

Thus for a classification problem, $L(Y, f(X)) = 0$ if $Y = f(X)$, 1 otherwise

Remark

These two loss functions are the most popular. But other loss functions exist and may be useful depending on the prediction method (e.g., *cross-entropy* to train neural nets, or *hinge loss* for SVM)

Error rate

True error/Test error

The *true error rate* w.r.t. a prediction function f and a loss function L is defined as

$$\begin{aligned}\mathcal{E}[f] &= E_{X,Y}\left[L\left(Y, f(X)\right)\right], \\ &= \int L\left(y, f(x)\right) dP(x, y),\end{aligned}$$

where $P(x, y)$ is the joint probability measure of (X, Y).

☞ measure the average error rate for new data independent from f, aka Test error.

Training error

The *training error rate* w.r.t. a prediction function f, a loss function L, and a training dataset $(X_1, Y_1), \dots, (X_N, Y_N)$ is defined as the empirical mean

$$\hat{\mathcal{E}}_N[f] = \frac{1}{N}\sum_{i=1}^{N} L\left(Y_i, f(X_i)\right)$$

☞ in practice f is learned and may over-fit the training set : not a faithful estimator of the True error !

Error rates for usual loss functions

Quadratic cost (rather for regression)

- True error rate →Mean Squared Error (MSE)

$$\mathcal{E}[f] = E_{X,Y}\left[(Y - f(X))^2\right] = \int (y - f(x))^2\, dP(x, y),$$

- Training error rate → Residual Sum of Squares(RSS)

$$\hat{\mathcal{E}}_N[f] = \frac{1}{N}\sum_{i=1}^{N}(Y_i - f(X_i))^2$$

Error rates for usual loss functions (Cont'd)

$0-1$ cost (rather for classification)

For a classication problem,

- True error rate :

$$\begin{aligned}\mathcal{E}[f] &= \Pr\left(|Y - f(X)| > \epsilon\right), \\ &= \Pr\left(Y \neq f(X)\right) \leftarrow \text{misclassification probability}\end{aligned}$$

- Training error rate :

$$\begin{aligned}\hat{\mathcal{E}}_N[f] &= \frac{1}{N}\sum_{i=1}^{N} 1_{|Y_i - f(X_i)| > \epsilon}, \\ &= \frac{1}{N}\sum_{i=1}^{N} 1_{Y_i \neq f(X_i)} \leftarrow \text{misclassification rate}\end{aligned}$$

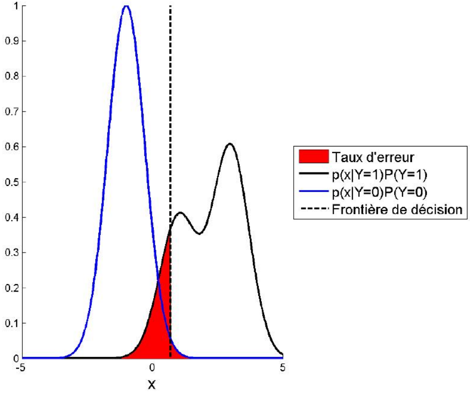
Exercise : True error rate

Consider a binary classification problem $Y \in \{0, 1\}$ with scalar data $X \in \mathbb{R}$.

For a prediction rule $f_1(X) = 0$ if $X \leq t$, 1 otherwise, where t is a given threshold, and based on the

- class weights $\Pr(Y = 0)$, $\Pr(Y = 1)$
- class pdf's $p(x|Y = 0)$, $p(x|Y = 1)$

how can we compute the true error rate $(0-1$ cost) ?



Recap on Train and Test Errors

- Loss-function
  - Classification : $L(y, \hat{y}) = 0$ if $y = \hat{y}$ else $1 \leftarrow$ 0-1 loss
  - Regression : $L(y, \hat{y}) = (y - \hat{y})^2 \leftarrow$ quadratic loss
- **Train error** : average loss over the training sample

$$\text{Err}_{\text{train}} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{y}_i)$$

- **Test error** : average loss over a new test sample $\rightarrow$ Generalization error
- General picture :

$$\text{Err}_{\text{test}} \approx \text{Err}_{\text{train}} + O$$

O would be the average *optimism* (overfitting problem !)

Model Assesment/Validation

Objective

Estimate the generalization error, i.e. the predictive performance on unseen/test data

Common Applications

- Hyperparameter tuning : estimate the best set of hyperparameters
- Model selection : compare the performance of differents models

Validation procedures

Validation set

Create a validation set, i.e. fresh data not used during the training set, to estimate the predictive performances/test error rate

- Best solution : collect a large data set of new test data... But in practice, this is generally not available/possible !
- Simple solution : split the data set. One part is removed from the training set and will be kept as a validation set only

  $\longleftarrow$ Full Dataset $\longrightarrow$

Train	Validation

  But

  - only part of the data is used to train the model (introduces a bias)
  - the role of the data is not symmetrical : some are used only for training, others only for testing (increases the variance)
- standard solution : cross-validation, e.g. K-fold Cross-Validation

K-fold Cross-Validation (CV) : Principle

- Method to estimate prediction error using the only training sample
- Based on splitting the data in K-folds, here $K = 5$:

$\text{Err}_1(\hat{f}_1, \lambda)$	Validation	Train	Train	Train	Train
$\text{Err}_2(\hat{f}_2, \lambda)$	Train	Validation	Train	Train	Train
$\text{Err}_3(\hat{f}_3, \lambda)$	Train	Train	Validation	Train	Train
$\text{Err}_4(\hat{f}_4, \lambda)$	Train	Train	Train	Validation	Train
$\text{Err}_5(\hat{f}_5, \lambda)$	Train	Train	Train	Train	Validation

where λ are some hyperparameters of the model/method

- Estimate of Test error :

$$\mathrm{CV}(\hat{f}, \lambda) = \sum_{k=1}^{K} \text{Err}_k(\hat{f}_k, \lambda)$$

K-fold Cross-Validation (CV) : Algorithm

`Input` : input variables X (dimension $n \times p$), responses y (dim. n), number of folds k

`Divide` randomly the set $\{1, 2, \ldots, n\}$ in k subsets (i.e., folds) of roughly equal sizes (e.g., size equals to the integer part of n/k with a little smaller last part if n is not a multiple of k) denoted as $F_1, \ldots, F_k$

`for i = 1 to k:`

- `Form` the validation set (X_{val}, y_{val}) where the indexes of the X and y variables belongs to the ith fold F_i
- `Form` the training set (X_{train}, y_{train}) where the indexes of the X and y variables belongs to all the folds except F_i
- `Train` the algorithm/model on the training set (X_{train}, y_{train})
- `Apply` the resulting prediction rule on the input X_{val} of the validation set
- `Compute` the error rate on the validation set based on the predictions and the true responses y_{val}

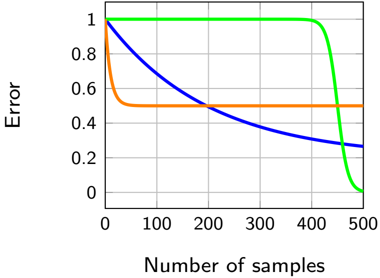
`Output` : the average error rate computed over all the k folds

Pratical advices

- K ? Usually K=5 or 10 is a good trade-off (K=n is called leave-one-out)

	Bias	Variance
K low	High	Low
K high	Low	High
$K = n$	Low	Very High

- Be careful to the learning curve



- Model should be trained completely for each fold (i.e., data normalization, optimization, etc ...)
- Notebooks : `N1_validation_and_model_selection.ipynb`

Conclusions

Model assesment methods are an essential tool for data analysis, especially for big datasets involving many predictors

Cross-validation

- generic and simple procedure that can be used for any supervised problem
- provides a direct estimate of the test error.
- Can be used for model selection and/or hyperparameter tuning
- K-fold (with $K = 5$, or $K = 10$) is a standard choice, but there exists many variants depending on the problem, e.g.
  - *stratified* K-fold to ensure that all the folds have roughly the same average response value $\leftarrow$ useful for classification to be sure that each fold contains roughly the same proportions of class labels.
  - *hold-out* cross-validation for time-series where a subset (split temporally) of the data is reserved for validating the model performance

Supplementary materials

Cross-validation

- Coursera MOOC short (12mn) video https://fr.coursera.org/lecture/machine-learning/model-selection-and-train-validation-test-sets-QGKbr
- Wikipedia page https://en.wikipedia.org/wiki/Cross-validation_(statistics)
- Scikit-learn documentation (with examples) :
  - https://scikit-learn.org/stable/modules/cross_validation.html#
  - https://scikit-learn.org/stable/modules/grid_search.html#grid-search