

Sapienza University of Rome

Master in Artificial Intelligence and Robotics  
Master in Engineering in Computer Science

# Machine Learning

A.Y. 2019/2020

Prof. L. Iocchi, F. Patrizi, V. Ntouskos

## 2. Classification Evaluation

L. Iocchi, F. Patrizi, V. Ntouskos

# Overview

- Statistical evaluation
- Performance metrics

## References

T. Mitchell. Machine Learning. Chapter 5

## Statistical methods for estimating accuracy

Performance evaluation in classification based on *accuracy* or *error rate*.

### Questions:

- How to estimate accuracy of a hypothesis  $h$ ?
- Given accuracy of  $h$  over a limited sample of data, how well does this estimate its accuracy over additional examples?
- Given that  $h$  outperforms  $h'$  over some sample of data, how probable is it that  $h$  is more accurate in general?
- When data is limited what is the best way to use data to both learn  $h$  and estimate its accuracy?
- Is accuracy the unique performance metric to evaluate classification methods?

## Example

Consider a typical classification problem:

$$f : X \rightarrow Y$$

$\mathcal{D}$  : probability distribution over  $X$

$S$  : sample of  $n$  instances drawn from  $X$  (according to distribution  $\mathcal{D}$ ) and for which we know  $f(x)$

Consider a hypothesis  $h$ , solution of a learning algorithm obtained from  $S$ .

What is the best estimate of the accuracy of  $h$  over future instances drawn from the same distribution?

What is the probable error in this accuracy estimate?

## Two Definitions of Error/Accuracy

The **true error** of hypothesis  $h$  with respect to target function  $f$  and distribution  $\mathcal{D}$  is the probability that  $h$  will misclassify an instance drawn at random according to  $\mathcal{D}$ .

$$\text{error}_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}} [f(x) \neq h(x)]$$

The **sample error** of  $h$  with respect to target function  $f$  and data sample  $S$  is the proportion of examples  $h$  misclassifies

$$\text{error}_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x) \neq h(x))$$

where  $\delta(f(x) \neq h(x))$  is 1 if  $f(x) \neq h(x)$ , and 0 otherwise.

Note:  $\text{accuracy}(h) \equiv 1 - \text{error}(h)$

## Two Definitions of Error

The **true error** cannot be computed, the **sample error** is computed only on a small data sample.

How well does  $error_S(h)$  estimate  $error_{\mathcal{D}}(h)$ ?

Note: the goal of a learning system is to be accurate in  $h(x)$ ,  $\forall x \notin S$

If  $accuracy_S(h)$  is very high, but  $accuracy_{\mathcal{D}}(h)$  is poor, then our system would not be very useful.

## Problems in Estimating the True Error

Estimation bias

$$bias \equiv E[error_S(h)] - error_{\mathcal{D}}(h)$$

- ① If  $S$  is the training set used to compute  $h$ ,  $error_S(h)$  is optimistically biased
- ② For unbiased estimate,  $h$  and  $S$  must be chosen independently  
 $E[error_S(h)] = error_{\mathcal{D}}(h)$
- ③ Even with unbiased  $S$ ,  $error_S(h)$  may still vary from  $error_{\mathcal{D}}(h)$ .  
The smaller the set  $S$ , the greater the expected variance.

## Confidence Intervals

If

- $S$  contains  $n$  examples, drawn independently of  $h$  and each other
- $n \geq 30$

Then

- With approximately  $N\%$  probability,  $error_{\mathcal{D}}(h)$  lies in interval

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

where

$N\%$ :	50%	68%	80%	90%	95%	98%	99%
$z_N$ :	0.67	1.00	1.28	1.64	1.96	2.33	2.58

## Estimators

How to compute  $error_S(h)$

- 1 Partition the data set  $D$  ( $D = T \cup S$ ,  $T \cap S = \emptyset$ ,  $|T| = 2/3|D|$ )
- 2 Compute a hypothesis  $h$  using training set  $T$
- 3 Evaluate  $error_S(h) = \frac{1}{n} \sum_{x \in S} \delta(f(x) \neq h(x))$

$error_S(h)$  is a random variable (i.e., result of an experiment)

$error_S(h)$  is an unbiased *estimator* for  $error_{\mathcal{D}}(h)$

Given observed  $error_S(h)$  what can we conclude about  $error_{\mathcal{D}}(h)$ ?

## Comparing two hypotheses

Given two hypotheses  $h_1, h_2$ , the true comparison is

$$d \equiv \text{error}_{\mathcal{D}}(h_1) - \text{error}_{\mathcal{D}}(h_2)$$

and its estimator is

$$\hat{d} \equiv \text{error}_{S_1}(h_1) - \text{error}_{S_2}(h_2)$$

$\hat{d}$  is an *unbiased estimator* for  $d$ , iff  $h_1, h_2, S_1$  and  $S_2$  are independent each other.

$$E[\hat{d}] = d$$

Note: still valid if  $S_1 = S_2 = S$ .

## Overfitting

Consider error of hypothesis  $h$  over

- training data:  $\text{error}_S(h)$
- entire distribution  $\mathcal{D}$  of data:  $\text{error}_{\mathcal{D}}(h)$

Hypothesis  $h \in H$  **overfits** training data if there is an alternative hypothesis  $h' \in H$  such that

$$\text{error}_S(h) < \text{error}_S(h')$$

and

$$\text{error}_{\mathcal{D}}(h) > \text{error}_{\mathcal{D}}(h')$$

# Evaluation of a learning algorithm

How can we evaluate the performance of a learning algorithm?

$h$  is the solution of learning algorithm  $L$  when using a training set  $T$   
 $h = L(T)$

$error_S(h)$  is the result of only one experiment and the confidence interval can be large.

We can perform many experiments and compute  $error_{S_i}(h)$  for different independent sample data  $S_i$ .

⇒ **K-Fold Cross Validation** method

## K-Fold Cross Validation

- 1 Partition data set  $D$  into  $k$  disjoint sets  $S_1, S_2, \dots, S_k$  ( $|S_i| > 30$ )
- 2 For  $i = 1, \dots, k$  do
  - use  $S_i$  as test set, and the remaining data as training set  $T_i$
  - $T_i \leftarrow \{D - S_i\}$
  - $h_i \leftarrow L(T_i)$
  - $\delta_i \leftarrow error_{S_i}(h_i)$

- 3 Return

$$error_{L,D} \equiv \frac{1}{k} \sum_{i=1}^k \delta_i$$

Note:  $accuracy_{L,D} = 1 - error_{L,D}$

## Comparing learning algorithms $L_A$ and $L_B$

Which algorithm is better?

We would like to estimate:

$$E_{S \subset \mathcal{D}}[\text{error}_{\mathcal{D}}(L_A(S)) - \text{error}_{\mathcal{D}}(L_B(S))]$$

where  $L(S)$  is the hypothesis output by learner  $L$  using training set  $S$

i.e., the expected difference in true error between hypotheses output by learners  $L_A$  and  $L_B$ , when trained using randomly selected training sets  $S$  drawn according to distribution  $\mathcal{D}$ .

This measure can be again approximated by a K-Fold Cross Validation.

## Comparing learning algorithms $L_A$ and $L_B$

Use K-Fold Cross Validation to compare algorithms  $L_A$  and  $L_B$ .

- 1 Partition data set  $D$  into  $k$  disjoint sets  $S_1, S_2, \dots, S_k$  ( $|S_i| > 30$ )
- 2 For  $i$  from 1 to  $k$ , do
  - use  $S_i$  as test set, and the remaining data as training set  $T_i$
  - $T_i \leftarrow \{D - S_i\}$
  - $h_A \leftarrow L_A(T_i)$
  - $h_B \leftarrow L_B(T_i)$
  - $\delta_i \leftarrow \text{error}_{S_i}(h_A) - \text{error}_{S_i}(h_B)$
- 3 Return

$$\bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^k \delta_i$$

Note: if  $\bar{\delta} < 0$  we can estimate that  $L_A$  is better than  $L_B$ .



## Other performance metrics in classification

Is accuracy always a good performance metric?

Binary classification  $f : X \rightarrow \{-, +\}$ , with training set  $D$  containing 90% of positive samples.

$h_1(x)$  has 90% of accuracy,  $h_2(x)$  has 85% of accuracy.

Which is better?

## Other performance metrics in classification

$h_1(x) = +$  (most common value in  $D$ )

$h_2(x)$  is the result of a classification algorithm

In some cases, accuracy only is not enough to assess the performance of a classification method.

Unbalanced data sets are very common in problems related to anomaly detection (e.g, malware analysis, fraud detection, etc.)

## Other performance metrics in classification

	Predicted class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

Error rate =  $\frac{|\text{errors}|}{|\text{instances}|} = \frac{(\text{FN} + \text{FP})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}$

Accuracy =  $1 - \text{Error rate} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}$

Problems when datasets are unbalanced.

## Other performance metrics in classification

	Predicted class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

Recall =  $\frac{|\text{true positives}|}{|\text{real positives}|} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$   
ability to avoid false negatives (1 if FN = 0)

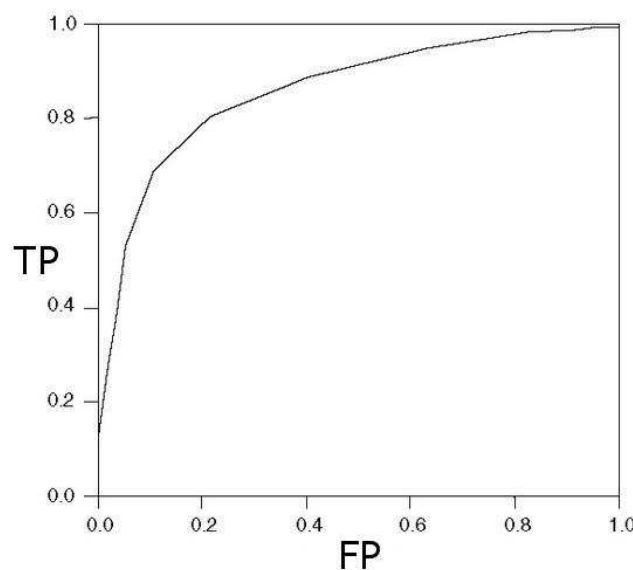
Precision =  $\frac{|\text{true positives}|}{|\text{predicted positives}|} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$   
ability to avoid false positives (1 if FP = 0)

Impact of false negatives and false positives depend on the application.

F1-score =  $2(\text{Precision} \cdot \text{Recall}) / (\text{Precision} + \text{Recall})$

## ROC curve

ROC curve: plot of (FP, TP) varying some parameters of the algorithm



ROC Area: area under the ROC curve.

## Confusion Matrix

In a classification problem with many classes, we can compute how many times an instance of class  $C_i$  is classified in class  $C_j$ .

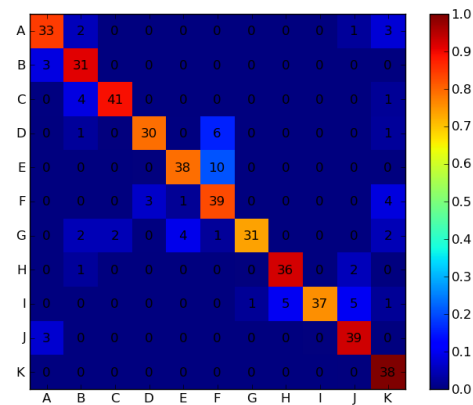
	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$
$C_1$					
$C_2$					
$C_3$					
$C_4$					
$C_5$					

Main diagonal contains accuracy for each class.

Outside the diagonal, the errors. It is possible to see which classes are more often confused.

# Confusion Matrix

Often represented with color-maps



## Summary

- Performance evaluation of machine learning methods is important and tricky.
- k-Fold Cross Validation is a general prototype method to evaluate classification methods.
- Several performance metrics can be considered and in some cases best metrics to use depend on the application.
- Performance estimation is very useful also during the execution of an algorithm.