Sapienza University of Rome

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

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

2. Classification Evaluation

L. Iocchi, F. Patrizi

Overview

in many case it's important to evaluate performance of learned hp to understand whether to use the hp or in other case evaluating hp is an integral component of many learning methods

we assume we have a function, a dataset...how good is this hp?? This is the goal of this lecture

- Statistical evaluation
- Performance metrics

References

T. Mitchell. Machine Learning. Chapter 5

Statistical methods for estimating accuracy

When evaluating a learned hypothesis we are most often interested in estimating the accuracy with which it will classify future instances. At the same time, we would like to know the probable error in this accuracy estimate

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

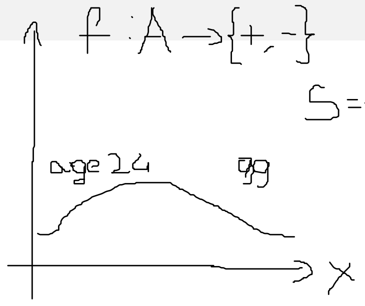
Questions:

- 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

$f : A \to \{+,-\}$

$S = \{(x_i, f(x_i))\}$



Consider a typical classification problem:

$f : X \to Y$

$\mathcal{D}$: *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?

we need to distinguish carefully between two notions of accuracy or, equivalently, error. One is the error rate of the hypothesis over the sample of data that is available. The other is the error rate of the hypothesis over the entire unknown distribution D of examples. We will call these the sample error and the true error respectively

Two Definitions of Error/Accuracy

it refers to any sample extracted from the dataset

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 $\mathcal{D}$. this D is different from the D with which we indicate the dataset (this stands for distribution)

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

defined on a dataset S for which we have both input and output

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

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

normalization factor

where $\delta(f(x) \neq h(x))$ is 1 if $f(x) \neq h(x)$, and 0 otherwise. delta = 1 if the condition is true and 0 if it is false

Note: $\boxed{accuracy(h) \equiv 1 - error(h)}$ we cannot compute the true error but we compute the sample error

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

difference between the expected value of the estimator and the true value of the parameter

Estimation bias

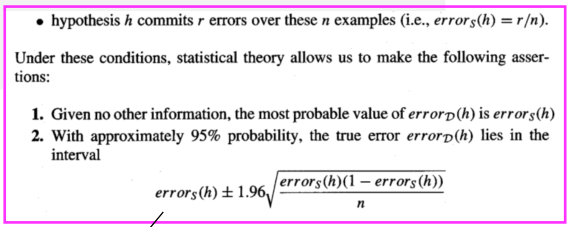
errorS(h) is an estimator of errorD(h)

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

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
   if $E[error_S(h)] = error_{\mathcal{D}}(h)$ errorsS(h) is a good estimator of errorD(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.

If the estimation bias is zero, we say that errorS(h) is an unbiased estimator for errorD(h)

Confidence Intervals



- hypothesis h commits r errors over these n examples (i.e., $error_S(h) = r/n$).

Under these conditions, statistical theory allows us to make the following assertions:

1. Given no other information, the most probable value of $error_{\mathcal{D}}(h)$ is $error_S(h)$
2. With approximately 95% probability, the true error $error_{\mathcal{D}}(h)$ lies in the interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

If

- S contains n examples, drawn independently of h and each other
- $n \geq 30$ increasing n is always good because reduces the variance and so guaranties that the true error falls in an interval

Then

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

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

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

errorD(h): true error
errorS(h): sample error

Estimators

it is a good balance, not a precise value

How to compute $error_S(h)$

the split must be done in a random way, in this way we obtain a different number for errorS(h)

split D in T and S so that the intersection is empty, sometimes also in 3 subsets

1. Partition the data set D ($D = T \cup S$, $T \cap S = \emptyset$, $\underline{|T| = 2/3|D|}$)
2. Compute a hypothesis h using training set T

   the split guaranties an unbiased estimator for the error
3. Evaluate $error_S(h) = \frac{1}{n}\sum_{x \in S} \delta(f(x) \neq h(x))$

if T =0.9*D errorD(h) is high while errorS(h) will be not a good value because the range in which it can fall is too large. On the contrary the true error is low, but errorS(h) will have a very good estimation

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

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

NOTE:

- $error_{\mathcal{D}}(h)$ is needed to evaluate hypothesis h, but *cannot be computed!*
- We must use an estimate
- The best we can do is to use $error_S(h)$ (suitably computed)

the sample error is a good estimator for the true error if bias=0

Trade off between training and testing

In general

if T is small, the variance is bigger, we can obtain potential good solution but potential

- Having more samples for training and less for testing improves performance of the model:
  potentially better model, but $error_S(h)$ does not approximate well $error_{\mathcal{D}}(h)$
- Having more samples for evaluation and less for training reduces variance of estimation:
  $error_S(h)$ approximates well $error_{\mathcal{D}}(h)$, but this value may be not satisfactory.

Trade off for medium sized datasets: 2/3 for training, 1/3 for testing.

Comparing two hypotheses

Given two hypotheses h_1, h_2, the true comparison is

$$d \equiv error_{\mathcal{D}}(h_1) - error_{\mathcal{D}}(h_2)$$

and its estimator is

$$\hat{d} \equiv error_{S_1}(h_1) - error_{S_2}(h_2)$$

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

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

$bias = E[\hat{d}] - d = d - d = 0$

$\hat{d}$ IS UNBIASED EST FOR d

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

Overfitting

for ex. if the accuracy in the training is higher than in the test

it is very easy to overfit

Consider error of hypothesis h over

- training data: $error_S(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

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

$$error_S(h) < error_S(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Evaluation of a learning algorithm

Often we are interested in comparing the performance of two learning algorithms La and Lb , rather than two specific hypotheses

How can we evaluate the performance of a learning algorithm?

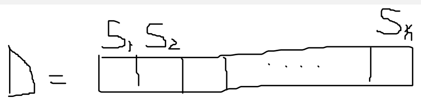
h is the solution of learning algorithm L when using a training set T
$h = L(T)$

$error_S(h)$ 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.

$\Rightarrow$ **K-Fold Cross Validation** method most common used method for evaluating classification problem

K-Fold Cross Validation



generate k subsets

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
   - $T_i \leftarrow \{D - S_i\}$
   - $h_i \leftarrow L(T_i)$ I apply a learning algo to obtain the hp
   - $\delta_i \leftarrow error_{S_i}(h_i)$

   if I repeat this process several times, I will obtain different hp and different values of errorSi(hi)

   $error_{S_i}(h_i)$
3. Return

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

the error will be the mean value of all the errors computed

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

Comparing learning algorithms L_A and L_B

one obvious approach to estimating the above quantity is to divide Do into a training set T and a disjoint test set S. The training data can be used to train both La and Lb,and the test data can be used to compare the accuracy of the two learned hypotheses

$$error_{T_0}(L_A(S_0)) - error_{T_0}(L_B(S_0))$$

Which algorithm is better?

We would like to estimate:

One way to improve this estimator is to repeatedly partition the data D into disjoint training and test sets and to take the mean of the test set errors for these different experiments. This procedure first partitions the data into k disjoint subsets of equal size, where this size is at least 30. It then trains and tests the learning algorithms k times, using each of the k subsets in turn as the test set, and using all remaining data as the training set (SLIDE 17)

$$\boxed{E_{S \subset \mathcal{D}}[error_{\mathcal{D}}(L_A(S)) - error_{\mathcal{D}}(L_B(S))]}$$

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.

In this way, the learning algorithms are tested on k independent test sets, and the mean difference in errors delta is returned as an estimate of the difference between the two learning algorithms.

Comparing learning algorithms L_A and L_B

extension of the previous case

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

1. Partition data set D into k disjoint sets $S_1, S_2, \ldots, S_k$ ($|S_i| > 30$)
2. For i from 1 to k, do

   use S_i as test set, and the remaining data as training set T_i
   - $T_i \leftarrow \{D - S_i\}$
   - $h_A \leftarrow L_A(T_i)$
   - $h_B \leftarrow L_B(T_i)$
   - $\delta_i \leftarrow error_{S_i}(h_A) - error_{S_i}(h_B)$
3. Return

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

this delta is an estimate of the formula in SLIDE 16

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

Performance metrics in classification

Is accuracy always a good performance metric?

$$D = \{(x_i, +)\ 10\%,\ (x_j, -)\ 90\%\}$$

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

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

Which one is better?

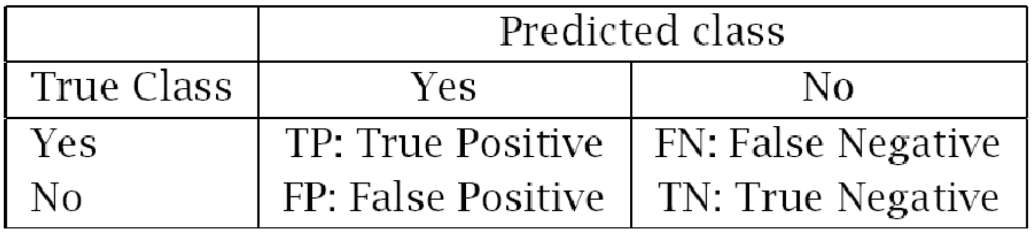
Performance metrics in classification

$h_1(x) = -$ (most common value of Y 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, medical tests, etc.)

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 = | errors | / | instances | = (FN + FP) / (TP + TN + FP + FN)
Accuracy = 1 - Error rate = (TP + TN) / (TP + TN + FP + 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

we need to balance precision and recall. If we have FP=0 and FN=10 or viceversa it is not good, because they need to be balanced

Recall = | true positives | / | real positives | = TP / (TP + FN)
ability to avoid false negatives (1 if FN = 0)

Precision = | true positives | / | predicted positives | = TP / (TP + FP)
ability to avoid false positives (1 if FP = 0)

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

F1-score $= 2(Precision \cdot Recall)/(Precision + Recall)$
a good value of F1-scores depends on the problem

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.

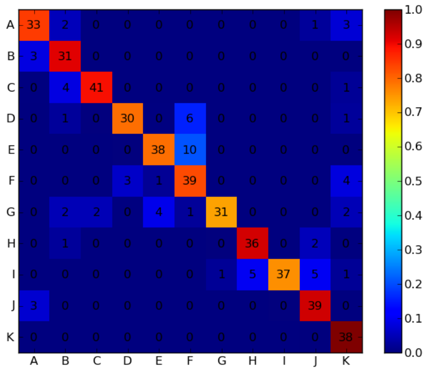
	C_1	C_2	C_3	C_4	C_5
C_1					
C_2					
C_3					
C_4					
C_5					

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



	A	B	C	D	E	F	G	H	I	J	K
A	33	2	0	0	0	0	0	0	0	1	3
B	3	31	0	0	0	0	0	0	0	0	0
C	0	4	41	0	0	0	0	0	0	0	1
D	0	1	0	30	0	6	0	0	0	0	1
E	0	0	0	0	38	10	0	0	0	0	0
F	0	0	0	3	1	39	0	0	0	0	4
G	0	2	2	0	4	1	31	0	0	0	2
H	0	1	0	0	0	0	0	36	0	2	0
I	0	0	0	0	0	0	1	5	37	5	1
J	3	0	0	0	0	0	0	0	0	39	0
K	0	0	0	0	0	0	0	0	0	0	38

Other performance measures

- Recall, Sensitivity, True Positive Rate
  $TPR = TP/P = TP/(TP + FN)$
- Specificity, True Negative Rate
  $TNR = TN/N = TN/(TN + FP)$
- False Positive Rate
  $FPR = FP/N = TP/(TN + FP)$
- False Negative Rate
  $FNR = FN/P = FN/(TP + FN)$
- ROC curve: plot TPR vs FPR varying classification threshold
- AUC (Area Under the Curve)

Summary

- Performance evaluation of machine learning methods is important and tricky.
- k-Fold Cross Validation is a general prototype method to evaluate classification methods. using this we have an unbalanced estimation
- 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.