Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2025/2026

Prof. Luca Iocchi

2. Performance Evaluation

Luca Iocchi

Overview

- Statistical evaluation
- Performance metrics for classification
- Performance metrics for regression

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?

Problem definition

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?

Example

You want to develop an App for university students and predict whether it will be successful.

$WillBuyMyApp : X \to \{Yes, No\}$

X: features about university students, including age

Probability distribution over age in X is not uniform.

Sampling $x \in X$ (i.e., pick up a random university student)
$Pr(age(x) = 22) > Pr(age(x) = 27)$

A sample set S of university students will contain more values with age 22 than values with age 27.

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 $\mathcal{D}$.

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

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

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

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

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

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

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.

Expected Value of Sample Error

$error_S(h)$ is a random variable depending on sampling S from $\mathcal{D}$
Sampling two different sets S and S' from $\mathcal{D}$ returns different values $error_S(h)$ and $error_{S'}(h)$

$E_S[error_S(h)]$ is the expected value of the sample error, i.e., the weighted average over all the possible samples S.

Note: the expected value $E[V]$ of a random variable V is the wighted average of all the possible outcomes, weighted by the probability that any outcome occurs.

Example: V = outcome of rolling a 6-sided die,
$E[V] = 1/6 \cdot 1 + 1/6 \cdot 2 + \ldots + 1/6 \cdot 6 = 3.5$

Problems in Estimating the True Error

Estimation bias

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

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**
   $E_S[error_S(h)] = error_{\mathcal{D}}(h)$
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

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Unbiased Estimators

How to compute an unbiased estimation $error_S(h)$

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

Using $error_S(h)$, suitably computed, is the best we can do!

Trade off between training and testing

In general

- 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.

Estimating the expected value

Computing $error_S(h)$ is not enough (it's just one outcome of the random variable)

$E_S[error_S(h)]$ can be approximated by averaging the computed values of $error_{S_i}(h)$ for several subsets S_i.

Example: to compute $E[$ *outcome of a die* $]$ we can average the values observed when rolling the die many times.

Evaluation of a learning algorithm

How can we evaluate the performance of a learning algorithm?

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

With different training sets T and T', we have different solutions $h = L(T)$ and $h' = L(T')$

How to compute an **unbiased estimation** of the **expected value** of the **sample error** of a learning algorithm L ?

$\Rightarrow$ **K-Fold Cross Validation** algorithm

K-Fold Cross Validation

1. Partition data set D into k disjoint sets $S_1, S_2, \ldots, S_k$ ($|S_i| > 30$)
2. For $i = 1, \ldots, k$ 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 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. Still valid if $S_1 = S_2 = S$.

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

Overfitting

Consider error of hypothesis h over

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

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

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

and

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

Comparing learning algorithms L_A and L_B

Which algorithm is better?

We would like to estimate:

$$error_{\mathcal{D}}(L_A(T)) - error_{\mathcal{D}}(L_B(T))$$

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

using

$$E_S[error_S(L_A(T)) - error_S(L_B(T))]$$

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 $S_1, S_2, \ldots, S_k$ ($|S_i| > 30$)
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

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

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

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.

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Performance metrics in classification

Is accuracy always a good performance metric?

Example:
Binary classification $f : X \to \{-, +\}$, with test 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.)

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 = | 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 $= 2(Precision \cdot Recall)/(Precision + Recall)$

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

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Domain-dependent targets

$PredictDesease : X \rightarrow \{T, F\}$

FP: disease is predicted, but it is not true
Impact: undue medical treatment
FN: disease is not predicted, but it is true
Impact: lack of medical treatment

$DetectPedestrian : X \rightarrow \{T, F\}$

FP: pedestrian is predicted, but it is not present
Impact: car brakes without reason
FN: pedestrian is not predicted, but it is present
Impact: possible injury of a person

Multi-class Confusion Matrix

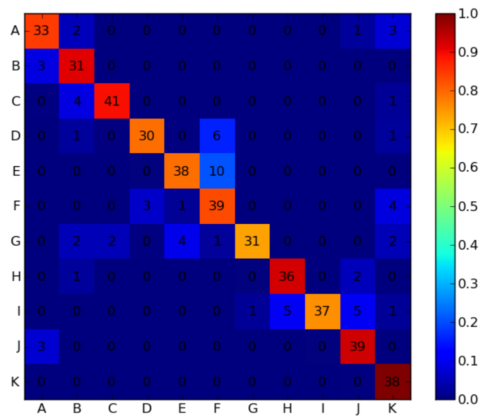
Report how many times an instance of class C_i is classified in class C_j.

T \ P	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: which classes are more often confused.
Sum of row i = total number of samples fo class C_i in dataset
When using percentages, each row is normalized to 1 (100 %)

Confusion Matrix

Often represented with color-maps



Performance metrics for regression

For regression problems $f : X \to \Re^d$, with test set $S = \{(x_i, t_i)_{i=1}^n\}$, performance measured in terms of

$$|\hat{f}(x_i) - t_i| \quad \text{for } (x_i, t_i) \in S$$

Mean Absolute Error (MAE)

$$\frac{1}{n}\sum_{i=1}^{n} |\hat{f}(x_i) - t_i|$$

Mean Squared Error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n} (\hat{f}(x_i) - t_i)^2$$

Root Mean Squared Error (RMSE) $\sqrt{MSE}$

Performance metrics for regression

... or in terms of percentage error

$$\frac{|\hat{f}(x_i) - t_i|}{t_i}$$

Mean Absolute Percentage Error (MAPE)

Mean Squared Percentage Error (MSPE)

Root Mean Squared Percentage Error (RMSPE)

Performance evaluation for regression

Regression score R^2 (coefficient of determination)

$$R^2 = 1 - \frac{\sum_{n=1}^{N}(y_n - t_n)^2}{\sum_{n=1}^{N}(t_n - \mu)^2}$$

with $\mu = \frac{1}{N}\sum_{n=1}^{N} t_n$

For $y(x) = \mu$, $R^2 = 0$.

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Performance evaluation for regression

k-Fold Cross-Validation can be extended to regression problems using appropiate metrics.

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

   use S_i as test set, and the remaining data as training set T_i
   - $T_i \leftarrow \{D - S_i\}$
   - $h_i \leftarrow L(T_i)$
   - $\delta_i \leftarrow MAE_{S_i}(h_i)$
3. Return

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

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.