Sapienza University of Rome

Master in Engineering in Computer Science

Artificial Intelligence & Machine Learning

A.Y. 2024/2025

Prof. Fabio Patrizi

4. Classification Evaluation

Fabio Patrizi

Overview

- Statistical evaluation
- Performance metrics

References

- Lecture notes and slides
- [AIMA] 19.4.1
- T. Mitchell. Machine Learning. Chapter 5.

Statistical Methods for Perfomance Evaluation

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 new examples?
- Given that h outperforms h' over some samples, 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 accuracy?
- Is accuracy the unique performance metric to evaluate classification methods?

Example

$P(h(x)=f(x))$?

Consider a classification problem (supervised, discrete X, Y):

- $f : X \to Y$
- d : probability distribution over X
- S: n samples $x \in X$ according to d (written $S \sim d$), known $f(x)$

Consider a hypothesis h returned by a learning algorithm on S

- What is the best estimate of h accuracy over future instances $x \sim d$?
- What is the probable error in this accuracy estimate?

True/Sample Error/Accuracy

- **True Error** of h wrt f and d:

$$error_d(h) = \Pr[f(x) \neq h(x)]$$

probability that h misclassifies random instance $x \sim d$

- **Sample Error** of h wrt f and S:

$$error_S(h) = \frac{1}{n} \sum_{x \in S} \delta(x), \text{ with } \delta(x) = \begin{cases} 1, \text{ if } f(x) \neq h(x) \\ 0, \text{ otherwise} \end{cases}$$

proportion of examples from S that h misclassifies

- **True Accuracy**: $accuracy_d(h) = 1 - error_d(h)$
- **Sample Accuracy**: $accuracy_S(h) = 1 - error_S(h)$

True/Sample Error

- True error $error_d(h)$ cannot be computed: need d and f
- Sample error $error_S(h)$ is computed on small data sample S

How well does $error_S(h)$ estimate $error_d(h)$?

Observe:

- Our goal is high accuracy of h over $X \setminus S$ (on S, we have f!)
- If $accuracy_S(h)$ is high but $accuracy_d(h)$ poor, h is not useful

Problems in Estimating the True Error

Estimation Bias: $bias = E[error_S(h)] - error_d(h)$

1. If S used to compute h, $error_S(h)$ is optimistically biased
2. For unbiased estimate, h and S must be chosen independently
   $E[error_S(h)] = error_d(h)$
3. $error_S(h) \neq error_d(h)$, even with unbiased S
   The smaller S, the greater the expected variance

Training Set, Test Set, Error Estimation

How to compute $error_S(h)$

1. Partition data set as $D = \{T, S\}$ ($T \cap S = \emptyset$), where:
    - T is the *Training Set*
    - S is the *Test Set*
    - $|T| = 2/3|D|$ (rule of thumb)
2. Compute h using training set T
3. Evaluate error on test set S: $error_S(h) = \frac{1}{n}\sum_{x \in S} \delta(x)$

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

$error_S(h)$ is an unbiased *estimator* for $error_d(h)$

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

Training vs Testing Trade-off

- More training samples (and less for testing) improve performance: better model, but $error_S(h)$ does not approximate well $error_d(h)$
- More evaluation samples (and less for training) improves estimation: $error_S(h)$ approximates well $error_d(h)$, but may be unsatisfactory

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

Overfitting

Consider error of hypothesis h over

- training data: $error_T(h)$
- entire instance space: $error_d(h)$ (estimated by $error_S(h)$)

Hypothesis $h \in H$ **overfits** training data if for some alternative $h' \in H$:

$$error_T(h) < error_T(h') \text{ (h' performs worse than h on T)}$$

but

$$error_d(h) > error_d(h') \text{ (estimated as: } error_S(h) > error_S(h'))$$
$$\text{(h' performs better than h on unseen instances)}$$

Intuitively, h is (too much) tailored to training data

Hypothesis Comparison and Selection

Given two hypotheses h_1, h_2, the true comparison is

$$d = error_d(h_1) - error_d(h_2)$$

and its estimator is

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

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

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

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

Learning Algorithm (and Model Class) Evaluation

How to evaluate performance of a learning algorithm L wrt target f?

- $h = L(T) \in H$ learnt with algorithm L and training set $T \sim d$
- $error_S(h)$: result of single experiment
  - may not approximate well $E_{T\sim d}[error_d(L(T))]$

K-Fold Cross Validation:

- Perform many experiments and compute $error_{S_i}(h)$ for different independent test sets S_i

K-Fold Cross Validation

K-Fold Cross Validation:

1. Partition dataset $D = \{S_1, S_2, \ldots, S_k\}$
2. For $i = 1, \ldots, k$ do

   use S_i as test set, and remaining data as training set T_i:
   - $T_i = \{D - S_i\}$
   - $h_i = L(T_i)$
   - $\eta_i = error_{S_i}(h_i)$
3. Return $\eta = \frac{1}{k}\sum_{i=1}^{k} \eta_i$

- $error_{L,D}$ is an unbiased estimator for $E_{T \sim d}[error_d(L(T))]$
- Note: $accuracy_{L,D} = 1 - error_{L,D}$

Comparing Learning Algorithms

Given:

- Learning algorithms L_A and L_B
- Target function f
- Dataset D

Which algorithm is better wrt f?

Need to estimate: $E_{T \sim d}[error_d(L_A(T)) - error_d(L_B(T))]$

(expected difference in true error between hypotheses output by L_A and L_B, when trained both on random training set T, drawn according to d)

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

Comparing Learning Algorithms

K-Fold Cross Validation to compare algorithms L_A and L_B

1. Partition dataset $D = \{S_1, S_2, \ldots, S_k\}$
2. For $i = 1, \ldots, k$ do

   use S_i as test set, and remaining data as training set T_i:
   - $T_i = \{D - S_i\}$
   - $h_A = L_A(T_i)$
   - $h_B = L_B(T_i)$
   - $\eta_i = errors_{S_i}(h_A) - errors_{S_i}(h_B)$
3. Return $\eta = \frac{1}{k}\sum_{i=1}^{k} \delta_i$

η is an estimator for $E_{T \sim d}[error_d(L_A(T)) - error_d(L_B(T))]$

If $\eta < 0$, we expect L_A to perform better than L_B in approximating f

Performance Metrics in Classification

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

- $error_S(h) = \frac{\#errors}{\#instances} = \frac{FN+FP}{TP+TN+FP+FN}$
- $accuracy_S(h) = 1 - error_S(h) = \frac{TP+TN}{TP+TN+FP+FN}$

Problems when datasets are unbalanced

Performance Metrics in Classification

Is accuracy always a good performance metric?

Example:

- Binary classification $f : X \rightarrow \{+, -\}$
- Test set S with 90% negative samples

Consider two hypotheses:

- $h_1(x)$ with $accuracy_S(h) = .9$
- $h_2(x)$ with $accuracy_S(h) = .85$

Which one is better?

Performance Metrics in Classification

- $h_1(x) = -$ (most common value of Y in training set T)
- $h_2(x)$: result of learning algorithm

Accuracy alone not always enough to assess performance of hypothesis

Unbalanced datasets very common in problems related to anomaly detection, e.g.:

- malware analysis, fraud detection, medical tests, etc.

Precision and Recall

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

- $precision_S(h) = \frac{TP}{TP+FP} = \frac{\#true\ \ positives}{\#predicted\ \ positives}$
  (ability to avoid false pos)
- $recall_S(h) = \frac{TP}{TP+FN} = \frac{\#true\ \ positives}{\#real\ \ positives}$
  (ability to avoid false neg)
- $F1{-}score_S(h) = 2\frac{precision_S(h) \cdot recall_S(h)}{precision_S(h) + recall_S(h)}$
  (harmonic mean of $precision_S(h)$ and $recall_S(h)$)

Impact of false negatives and false positives depends on application

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)

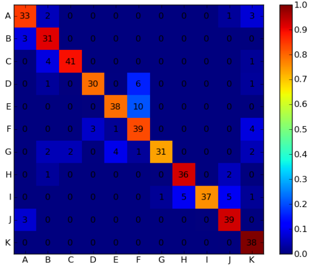
Confusion Matrix (Multi-Class)

	c_1	c_2	c_3	c_4	c_5
c_1					
c_2					
c_3					
c_4					
c_5					

- Element (c_i, c_j): # (or proportion) of c_i-instances classified as c_j
- Main diagonal contains accuracy for each class
- Errors are outside main diagonal

Confusion Matrix

Often represented with color-maps



Summary

- Performance metrics are critical in ML to evaluate and compare solutions and algorithms
- True metrics can only be estimated
- k-Fold Cross Validation allows to compute unbiased metric estimators
- Error/Accuracy not always reliable, need for additional metrics: precision, recall, confusion matrix (and other)