Part 6:
Elements of Performance Evaluation

Battista Biggio
battista.biggio@unica.it

Introduction

- In Part 2, we pointed out that the exact, analytical computation of error probability is feasible in a limited number of special cases
  - In general, the decision regions are not known exactly (in closed form), and the computation of the integrals required to calculate the error probability is very complicated

- However, the error probability of a given classifier can be estimated by experiments using the "design set" D.
  - The next slides illustrate the basic concepts for such a performance evaluation by experiments

Preliminary Remarks

- In real applications, we can never collect a design dataset that contains all, or most of, the possible examples of the objects we want to recognize: this limitation refers to the so-called **Issue of Generalization** (error)

- Indeed, if we could collect examples of any object to be recognized, we could assess classifier error using the same data set D used to train the classifier!

- Instead, we must estimate error using "patterns" out of the design dataset so that we can have a reliable estimate of error on "future" (unknown) patterns that are the ones that our pattern recognition machine will receive as inputs in the operational conditions (in the "wild" environment)

Apparent Error

- Given the design data set $\boldsymbol{D}=[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, the most straightforward method for error computation is to apply our classifier to all the n patterns in $\boldsymbol{D}$ and compute the rate of misclassified patterns:

$$\textit{Apparent Error} = n_{err} / n$$

- This is called *apparent* error (also called "resubstitution" error) because it is an *apparent,* very optimistic estimate of the true error that our classifier will make on future, unknown patterns (the "test" data) within the real-world operational environment
  - Remember that $\boldsymbol{D}$ has been used in the design stage!

Apparent Error

- It is easy to see that classifier parameters computed with ***D*** can be close to optimality for the data ***D*** itself (one could say that they have been "over-fitted" on ***D***)
  - but it is unlikely that such parameters are optimal for future unknown patterns ("test" data) unless ***D*** contains all the possible objects! Not realistic!

- Note that it is (usually) easy to design a classifier with zero error for data in D (also called an "over-trained" classifier), but what we want is to create a classifier with a low error on future, unknown patterns ("test" data).

Techniques for Performance (Error) Evaluation

Hold-out

- We subdivide the set D into two disjoint subsets called training set and test set.
  - The training set is used to "design" the classifier
  - The test set is used to assess the error probability

- This technique usually provides a pessimistic error evaluation. The two sets can have different sizes, for example, 70%/30%

- To improve the reliability of the assessment, we can do different trials by subdividing D randomly into two sets, and then we compute the mean value and the standard deviation of the error estimation

Techniques for Performance (Error) Evaluation

K-fold Cross-validation

- We subdivide ***D*** into K subsets of size n/K
  - We design the classifier using the union of K-1 subsets that we use as a "training set"
  - We estimate the error on the subset we have left out
  - We repeat this procedure by K times
  - We finally compute the mean value and the standard deviation of the error estimation

- When K=n, this technique is called leave-one-out.

Techniques for Performance (Error) Evaluation

Bootstrap

- We generate L subsets of size n by random sampling from ***D*** with replacement
- We compute the mean value and the standard deviation of the error using these L subsets

Some notable issues

- Which method should we use? It's not easy to answer, in general.
- If D is *small*, K-fold cross-validation or Bootstrap should work reasonably well, but they are computationally expensive
- If D is large, hold-out should be OK
- We can also use tests for statistical significance of estimation to assess if and how much the difference in performance between two classifiers is statistically significant

A Quick Note on the Validation Data

- If ***D*** is large enough we can subdivide it into three subsets: the **training** set, the **validation** set, and the **test** set
  - The validation set is used during the design phase of the classifier to avoid the so-called over-training; e.g., to tune classifier parameters while avoiding overfitting the training data

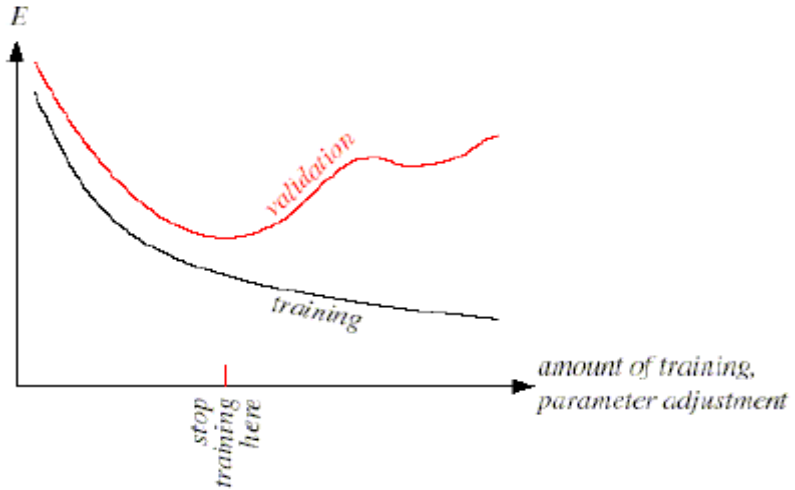


Illustration of the joint use of training and validation sets to control the learning phase of a given classifier (e.g., a neural network) to avoid "over-fitting"

Main Factors Influencing Performance Evaluation

1. The choice of "test" set data
   - Different "test" sets bring different evaluations of performance; that is, we usually have a clear dependency on the test data
2. The choice of "training" set data
   - Some classifiers are *unstable*, i.e., small changes in the training data can cause large changes in the test error
3. Intrinsic "randomness" of some classifier parameters (initialization)
   - Some classifiers have parameters that have to be specified randomly (e.g., the initial weights of a neural network)

- Therefore, it is mandatory to assess performance using different training and test sets, and different classifier initializations

Some Practical Remarks

- Performance evaluation methods aim to estimate the error probability on future, unknown data

- The designer uses these techniques to assess the performance that the classifier will exhibit when it works in the real-world operational conditions

- If the **standard deviation** of the classifier error, assessed with these techniques, is acceptable, then the designer will use the whole "design set" D to design and train the classifier!
  - The assumption is that the error on new data will be close to the estimated one

- If the **standard deviation** of the classifier error is large, different methods should be used. For example, the designer could train different classifiers and then combine them into a "multiple classifier system"

The Confusion Matrix

- A global evaluation of classifier performance can be obtained using the so-called confusion matrix computed on the test data

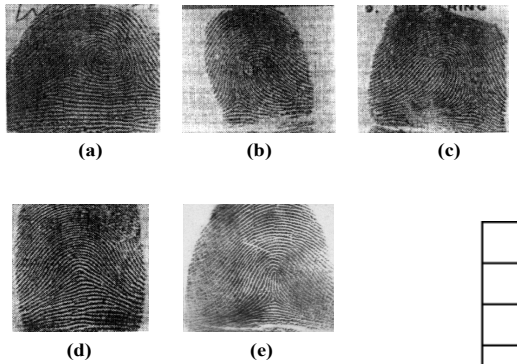
- It is a matrix of size (c x c), being c the number of classes
  - The rows of the confusion matrix represent the true classes
  - Its columns represent the predicted classes

- The confusion matrix's element (i, j) provides a value that estimates the probability of a sample of class ωi being predicted as class ωj

- The diagonal elements provide the correct classification probabilities for the c classes

An Example of Confusion Matrix

- Classification of fingeprint images within five classes:
  - Whorl (W), Right Loop (R), Left Loop (L), Arch (A), Tented Arch (T)



	A	L	R	T	W
A	**79.8**	1.0	1.9	17.3	0.0
L	1.6	**91.8**	0.5	4.0	2.1
R	1.6	0.0	**89.3**	7.0	2.1
T	14.0	3.1	2.0	**80.4**	0.5
W	1.0	3.3	6.0	0.3	**89.4**
Overall accuracy: 86.0%					