Part 6
Elements of performance evaluation

Introduction

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

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

Some preliminary remarks

•In real applications, we cannot never collect a design data-set that contains all, or the 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 data set, 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 $\boldsymbol{D=[x_1, x_2, \ldots, x_n]}$, the most straightforward method for error computation is to apply our classifier to all the *n* patterns in ***D*** and compute the rate of misclassified patterns:

$$\text{Apparent error} = n_{err} / n$$

•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 do on future, unknown patterns (the "test" data) when the classifier will work in the operational environment (remember that ***D*** has been used in the design stage!!).

•It is easy to see that classifier parameters computed with ***D*** can be close to optimality for patterns in D (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 that ***D*** contains all the possible objects! Not realistic!.

➢Indeed, 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 design 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. Training set is used to "design" the classifier, the test set is used to assess error probability. This technique usually provides a **pessimistic** error evaluation. The two sets can have different sizes, for example 50/50. To improve the reliability of the evaluation, 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.

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", and we estimate the error on the subset we have left out. We repeat this procedure by K times, and then we compute the mean value and the standard deviation of the error estimation.

If K=n this techniques is called leave-one-out.

Techniques for performance (error) evaluation

Bootstrap

Using ***D*** we generate *L* subsets of size *n* by random sampling with replacement. Then we compute the mean value and the standard deviation of the error using these *L* subsets.

Some notable issues

✓Which method should we use? 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 can be OK.

➢We can also use tests for statistical significance of estimation, especially to assess if and how much the difference of performance between two classifiers is statistically significant.

A quick note on the validation set

If ***D*** is large enough we can subdivide it into three subsets called: training set, validation set, and test set. The validation set is used during the design phase of the classifier to avoid the so called “over-training”. For example, to select classifier parameters so that they are not “over-fitted” on data used for classifier design.

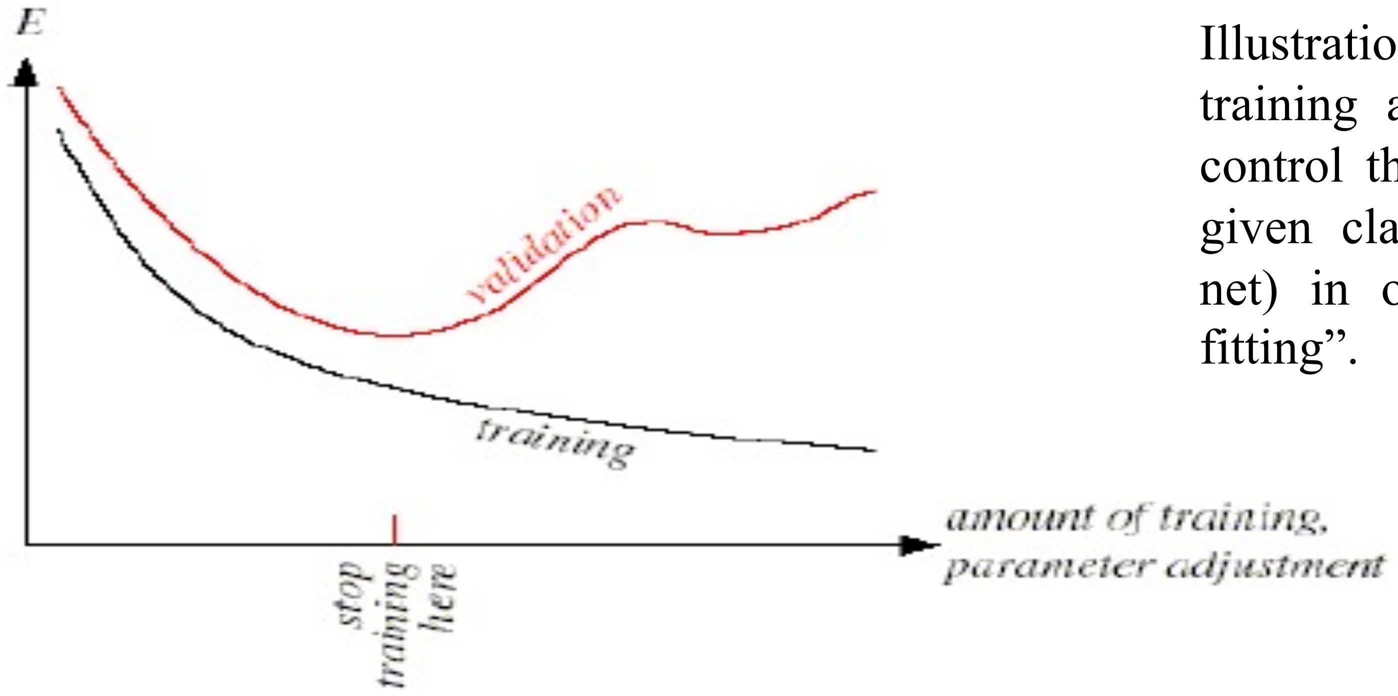


Illustration of the joint use of training and validation sets to control the learning phase of a given classifier (e.g., a neural net) in order to avoid “over-fitting”.

Main factors influencing performance evaluation

1. The choice of “test” set data

Different “test” sets bring to 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 called “instable”, namely, small changes of “training” set data cause large changes of error on test data.

3. Intrinsic “randomness” of some classifier parameters

Some classifiers have parameters which 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 values of classifier parameters.

Some practical remarks

✓It is worth noting that all the previous techniques are aimed to estimate the error probability on future, unknown, data

✓The designer use these techniques to assess performance that the classifier will exhibit when it will work in the 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 one estimated.

✓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".

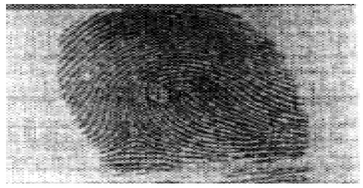
A note on the confusion matrix

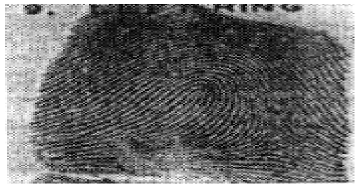
- A global evaluation of classifier performance can be obtained using the so called confusion matrix computed with test set data.
- It is a matrix with size (c x c), "c" is the number of data classes.
- Rows of the confusion matrix represent the true classes, columns represent classes assigned by the considered classifier.
- The element (i, j) of the confusion matrix provides a value that is an estimate of the probability that the classifier assigns one "pattern" belonging to the class ω_i to the class ω_j.
- The diagonal elements provide the correct classification probabilities for the "c" classes.

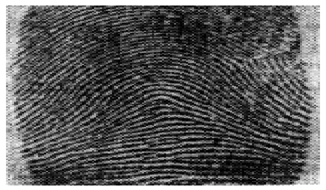
An example of confusion matrix

- Classification of fingeprint images, five classes are considered.

(a)

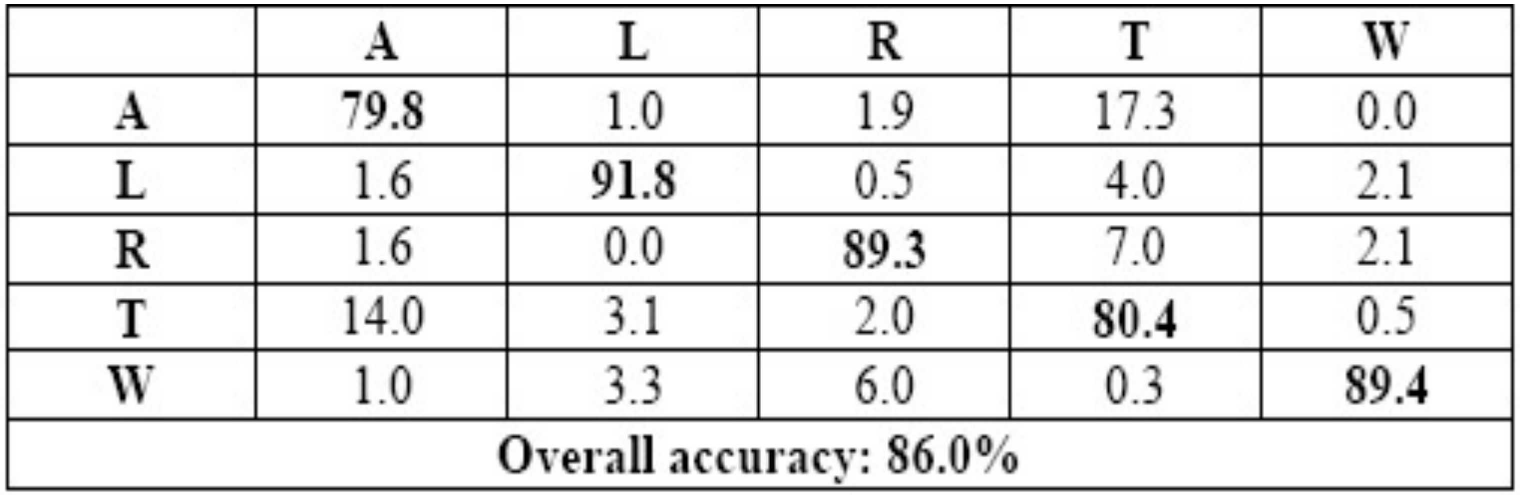

(b)


(c)


(d)

(e)

(a) Whorl (W)
(b) Right Loop (R)
(c) Left Loop (L),
(d) Arch (A)
(e) 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%					