Optimizing classifier performance
(Adapted from the Finnish course material "Luokittelijan optimointi")

In this chapter we will talk about optimal feature selection, optimizing classifiers and evaluating performance of the classifier

A. Methods for selecting features

Generating suitable features for the problem is crucial part of an machine learning system. However, after features have been generated, we need to select the best features for future processing. It is not feasible to select all the generated features, because each feature adds more parameters to the classifier, which requires more training data. Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of the classifier or may in fact decrease the accuracy. Fewer parameters is also desirable because it reduces the complexity of the classifier, and a simpler classifier is simpler to understand and explain.

A.1. Full search

If the amount of features is smallish, we can test all different lengths of feature vectors and feature combinations and select the best one. Full search feature selection has high time complexity ($O(2^n)$) so it is not practical for large amounts of data. This is because on each cycle of the algorithm we need to train the classifier and evaluate its accuracy.

A.2. Sequential forward floating selection (SFFS)

Let's first assume that we have an original set of features Y and X is the selected subset. In sequential forward selection (SFS) the idea is to start with an empty set of features (X=0) and repeatedly add the most significant feature with respect to X. This, however, may lead to a nesting problem. That is, once the feature is retained in cannot be discarded.

In sequential backward selection (SBS) we start with a full set of features ($X=Y$) and repeatedly delete the least significant feature. This leads also to the nesting problem; once discarded, the feature cannot be reclaimed.

Next we look at a popular feature selection method: sequential forward floating selection (SFFS) [Pudil et al. 1994]. The idea is to start with the SBS, followed by a series of successive conditional exclusion of the worst feature in the newly updated set. The algorithm proceeds by adding and removing features (floating) until improvement cannot be made anymore.

Algorithm for SFFS

Begin with empty set of features, X=0.

Step 1: Inclusion. Use the basic SFS method to select the most significant feature with respect to X and include it in X. Stop if d features have been selected, otherwise go to step 2.

Step 2: Conditional exclusion. Find the least significant feature in X. If its removal will (a) leave X with at least 2 features, and (b) the performance of the classifier is better than before, then remove it and repeat step 2. When these two conditions cease to be satisfied, return to step 1.

Source and more detailed description of the algorithm: Pudil, P., Novovičová, J. & Kittler J. (1994) Floating search methods in feature selection. Pattern Recognition Letters 15 (11), 1119–1125.

B. Testing the classifier

The parameters of the classifier are determined with training data and the performance evaluated with test data. The main idea is that the training set is as broad as possible so that the classifier parameters can be determined as well as possible and the classifier is good at generalization. On the other hand, the test set should also be as large as possible. Data can be divided into training and test sets with multiple ways.

B.1. Hold-out test

Data is divided into two parts, training set and test set with, e.g., 2/3 and 1/3 of the data, respectively. This works when there is a lot of data to be used.

B.2. N-fold cross-validation (sometimes K is used instead of N)

Data is divided into N equal sized parts and each is used separately for classifier testing and other parts for training. In the end, the performance values are averaged. This way, each data sample is taken along as training sample and testing sample, and the data is well utilized. Often N=10.

```
Algorithm for N-fold cross-validation
1. Divide data D randomly into N equal sized non overlapping parts $D_1,\ldots, D_n$.
2. Execute for each $i = 1,\ldots,N$
    2.1. Take $D_i$ as a test set
    2.2  Take other parts $D_j$ as training set ($j=1,\ldots.N, j\neq i$) and train the classifier
    2.3  Test the classifier with data $D_i$ and save the performance value $P_i$
3. Calculate the average of the performance values $P = (P_1 + \ldots + P_n)/N$
```

B.3 Leave-one-out

Leave-one-out is N-fold cross-validation taken as far as possible. In leave-one-out, one data point is left as test set and the rest are used as training set.