UNIVERSITÉ LIBRE DE BRUXELLES
Computer Science Department

INFO-F-422 : Statistical Foundations of Machine Learning

Project 2010-11

Alexandre Balon-Perin

Professor: Gianluca Bontempi

Année académique 2007 - 2008

Contents

Chapter 1

Introduction

We have been asked to realize a project as part of the course ”Statistical foundations of machine learning”.

The goal of this project are:

1. to implement and assess different supervised learning algorithms and different methods of feature selection in a regression task,
2. to select among the learning and feature selection techniques the ones which appear to be the most accurate and use them for predicting the target in a test set.

The learning task consist in predicting the percentage of time that the cpu runs in an user mode on the basis of a set of measurement of the activity of a computer system. These measurement refer to quantities like the number of system calls per second, the number of system fork per seconds or the process run queue size.

Chapter 2

Feature Selection

2.1 Principal Component Analysis [1]

The feature selection procedure used in this assignment is the Principal Component Analysis (PCA). PCA applies a transformation on the original dataset in order to reduce the number of variables taken into account for the prediction. With less variables, the computation time will be reduced dramatically. The principle is to create a new set of orthogonal axis sorted from the biggest variance to the lowest one. Those axis are called the eigenvectors and correspond to the vector a = $[a_1, ..., a_n] \in \mathbb{R}$ such that the variable

$$z = a_1 x_1 + ... + a_n x_n = a^T x$$

has the largest variance. The axis with the highest variance will be the first component, the one with the next highest variance will be the second component,etc.

2.2 Algorithm of the PCA [1]

The PCA consist of the following steps.

1. The matrix is normalized and transformed to a matrix $\tilde{X}[,i], i = 1, ..., n$, with mean 0 and variance 1.

2. The Singular Value Decomposition (SVD) of $\tilde{X}$ is computed

   $$\tilde{X} = UDV^T$$

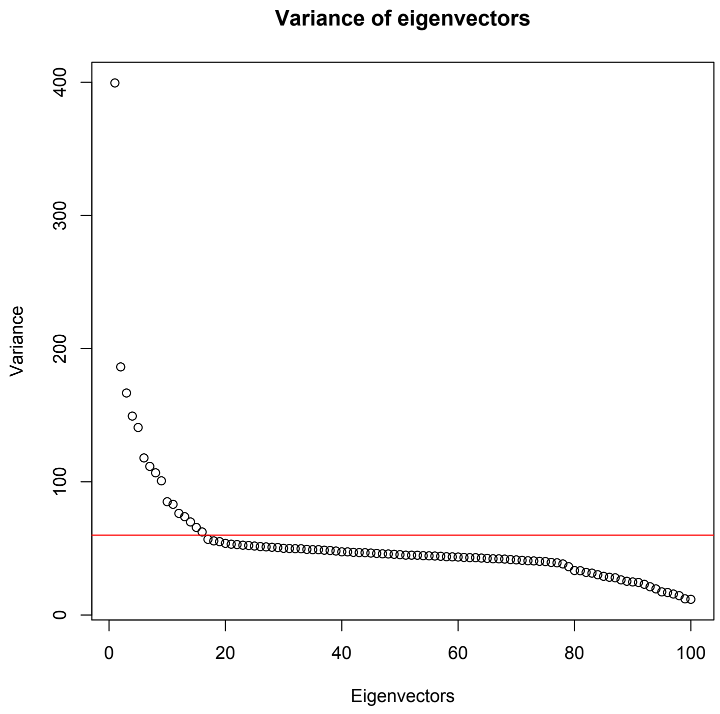
   where U is an orthogonal [N,N] matrix, D is a [N,N] diagonal matrix with diagonal elements $d_1 \geq d_2 \geq ... \geq d_N$ and V is an orthogonal matrix [N,n].

3. The matrix $\tilde{X}$ can be transformed into a new set of coordinates
   $Z = \tilde{X}V = UD$ where UD is a [N,N] matrix, where each column is a linear combination of the original features and its importance is diminishing. The first $h \leq n$ columns of Z (aka eigen-genes) may be chosen to represent the dataset.

2.3 selection

Because of the matrix product UDV^T, the columns of Z are not a permutation of the columns of X but the new dataset is a transformation of the original one. Therefore, it is not possible to show the selected variable with respect to the original dataset. Indeed, the new data can be used as it is for the training. However, the number of variables that will be used must still be decided. A graphical method is applied in order to find this number. The following plot shows the values of the diagonal of the matrix D in function of the column number.

These values correspond to the variance of x, they are also known as the eigenvalues.



The graph is close to the shape of a decreasing exponential. The goal is to find the eigenvectors which maximize the variance, so if an horizontal red line is placed where the slope decreases suddenly, we can find the limit for which the variance is too low to give good results. The intersection between the horizontal red line and the graph gives the number of variables that will be kept from the new dataset. This procedure ensures that the information loss is reduced and that we have filtered mainly noise (irrelevant information for the training) from the original dataset. A total of 17 variables have been kept in the new dataset.

Chapter 3

Model Selection

3.1 Cross-validation

The model selection procedure chosen in this work is the 10-fold cross-validation. The idea is to slice the 5000 observations of the training set in 10 parts of equals dimensions (here 500 observations).

Then the following pseudo-code is applied.

Algorithm 1: 10-fold cross-validation

```
input  : The set of data X.tr, Y.tr and X.ts
output: Two arrays containing the MSE and NMSE of the predictions for
         each fold of the cross-validation
sizeOfCrossValidation ← floor(length(X.tr)/10)
for $i$ ← 1 to 10 do
    X.tr.tr ← getTrainingSetX(X.tr, sizeOfCrossValidation)
    Y.tr.tr ← getTrainingSetY(Y.tr, sizeOfCrossValidation)
    X.tr.ts ← getTestSetX(X.tr, sizeOfCrossValidation)
    Y.tr.ts ← getTestSetY(Y.tr, sizeOfCrossValidation)
    model ← getModel(Y.tr.tr .,X.tr.tr)
    prediction ← computePrediction(model,X.tr.ts)
    MSE ← computeMSE(prediction,Y.tr.ts,sizeOfCrossValidation)
    NMSE ← computeNMSE(prediction,Y.tr.ts,sizeOfCrossValidation)
end
MSE.mean ← mean(MSE)
NMSE.mean ← mean(NMSE)
```

The final choice of the model was made by calculating the mean of the mean squared error (MSE)

$$E = \frac{1}{N_{ts}} \sum_{i=1}^{N_{ts}} (y_i - \hat{y_i})^2$$

and the mean of the normalized mean squared error (NMSE)

$$E = \frac{\sum_{i=1}^{N_{ts}} (y_i - \hat{y_i})^2}{\sum_{i=1}^{N_{ts}} (y_i - \bar{y_i})^2}$$

over the ten repetitions of the algorithm.

3.2 results

Five models were applied on the new dataset produced by the PCA.

1. Linear Model
2. Lazy Learning for local regression
3. Random Forest
4. Decision Tree
5. Support Vector Machines

The following table shows the different errors calculated for each learning method.

Model	Empirical Error	Mean Squared Error	Normalized Mean Squared Error
Linear	0.9226	40.9242	0.1162
Lazy	0.7786	10.5731	0.0301
Random Forest	0.6282	16.213	0.0455
Decision Tree	0.9096	40.6591	0.1142
SVM	0.8572	28.8098	0.0797

The lazy method is the one minimizing the mean squared error. That is why this method was selected for predicting the percentage of time that the cpu runs in an user mode.

Chapter 4

Confidence Interval

The 95% confidence interval is calculated using the following method.

$$\hat{y} - z_{\alpha/2}\frac{\sigma}{\sqrt{N}} \leq y \leq \hat{y} + z_{\alpha/2}\frac{\sigma}{\sqrt{N}}$$

where

- $\hat{y}$ is the prediction vector
- σ is the standard deviation of this vector
- y is the target value for the percentage of time that the cpu runs in a user mode
- N is the number of observations in the test set 3000
- α is equal to 0.05
- $z_{\alpha/2}$ is the quantile for the 95% confidence interval

The training set and the test set are the result of the feature selection procedure. The first step of the procedure to infer the confidence interval is to use these new data sets to compute the prediction with the lazy learning for local regression. Then, the standard deviation of the prediction vector is calculated. Afterwards, $z_{\alpha/2}$ is computed using the qnorm function of R. Eventually, the confidence interval is calculated with the formula given above.

Chapter 5

Combination of models

5.1 The Lazy Forest

The combination of models strategy is based on the idea of cross-validation. First, the data set for the training and the test are split in ten smaller datasets. So there are 2700 observations for the training and 300 observations for the test. Second, a random number between 0 and 10 is generated. Third, the Lazy Learning model is applied 50% of the time and the Random Forest model is applied the other 50% of the time according to the random number previously generated. Eventually, the prediction vector construction is a combination of the predictions of both methods. Random Forest performs better than Lazy Learning for some observations. That is why sometimes the predictions will be more accurate than with the Lazy Learning alone. However sometimes the predictions might as well be worse. The mean square error of this new kind of predictions will oscillate around the MSE of the Lazy Learning method.

The procedure used to infer the confidence interval is the same as the one explained in the chapter 4.

Algorithm 2: The Lazy Forest

```
input  : The set of data X.tr, Y.tr and X.ts
output: Two arrays containing the MSE and NMSE of the predictions for
        each fold of the cross-validation
sizeOfCrossValidation ← floor(length(X.ts)/10)
for $i$ ← 1 to 10 do
    X.tr.tr ← getTrainingSetX(X.tr, sizeOfCrossValidation)
    Y.tr.tr ← getTrainingSetY(Y.tr, sizeOfCrossValidation)
    X.tr.ts ← getTestSetX(X.ts, sizeOfCrossValidation)
    if rand < 5 then
        model.lazy ← getModel(Y.tr.tr .,X.tr.tr) current.predict ←
        computePrediction(model.lazy,X.tr.ts)
    else
        model.forest ← getModel(Y.tr.tr .,X.tr.tr)
        current.predict ← computePrediction(model.forest,X.tr.ts)
    end
    predict.vector ← (predict.vector,current.predict)
end
```

Bibliography

[1] Gianluca Bontempi. Methodes d'apprentissage automatique pour la bioinformatique, 2011.