Pattern Analysis and Machine Intelligence
(Homework 06-06-2011 – v0.1)

Matteo Matteucci, Davide Eynard, Luigi Malagò

Note: homework should be turned in by 29/06/2011 as a digital document (you can also scan your handwritten assignment into a pdf) through email (cc all the three authors of this document). Please, not to flood our mailboxes, send us the link to the document and we will acknowledge its receipt to you. For ***any*** doubt on the text ask by email the teachers, we will publish the answers on the website of the course!

Exercise 0: Matlab tutorial (2 Points)

For the homework you will need to code some algorithms and to do that we suggest to use the Matlab environment (if you do not have it, as Politecnico di Milano students, you can use it for free with a campus licence http://www.asict.polimi.it/software/matlab.html).

Your first exercise is thus to practice with the Matlab environment though any of the tutorial you can find online on the mathworks website

http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

A quite complete one is the getstart guide

http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

a shorter one is the following:

http://www.maths.dundee.ac.uk/~ftp/na-reports/MatlabNotes.pdf

You don not need to "master" the Matlab environment, knowing how to write a function, plot a graph, write a *for* loop, and compute a traspose/inverse of a matrix should cover the 80% of the requirements.

Exercise 1: Classifiers & Co. (8 Points)

Decision rules and classification rules are effective ways to extract human readable knowledge from data. Use the following dataset to extract some knowledge about factor affecting sunburn; in particular answer the following requests:

1. Build a full *decision tree* using the algorithm described during classes (explain also why you should NOT use the *Name* attribute); then turn the decision tree into a rule set and prune it using the chi-squared test with alpha 0.05 (multiply by 4 the number of records to allow for its use ...)
2. Build a brand new rule set using the *sequential covering algorithm* described during classes (predict *sunburned* class)
3. Build a *Naive Bayes classifier* using the easiest probability estimator for the discrete attributes (i.e., multinomial distribution).

Name	*Hair*	*Height*	*Weight*	*Lotion*	*Result*
Sarah	blonde	average	light	no	sunburned (positive)
Dana	blonde	tall	average	yes	none (negative)
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

Note: make all passages clear in he homework since the grading will be base on those not just on the final results!

Exercise 2: Linear classifiers hands-on (8 Points)

Using the Matlab environment **implement the following functions** according to the algorithms described in the texbook "The elements of statistical learning" and **compare them** on the "South African Heart Disease" dataset (avalable online on the book website http://www-stat.stanford.edu/~tibs/ElemStatLearn/data.html) by producing a table similar to Table 4.1 on page 107 of the same book.

Hints: generate a training and a test set from the 463 records by holding out 30% of data from each class randomly; code with 0/1 the Absent/Present values; do not use row.names as an attribute!

Linear regression on an indicator matrix: following the simple formulas on Section 4.2 of the textbook implement the simplest linear classification algorithm. Use the following headers for the functions, being X the input data, Y the classes, and beta the regression parameters:

```
function [beta]=linearRegression_train(X, Y)
function [Y]=linearRegression_test(X, beta)
```

Linear discriminant analysis: now build the classical LDA algorithm as described on Section 4.3 (without and with Fisher projection). The header of the functions should be:

```
function [mu_0,mu_1, sigma, pi_0, pi_1]=linearDiscriminantAnalysis_train(X, Y)
function [Y]=linearDiscriminantAnalysis_test(X, mu_0, mu_1, sigma, pi_0, pi_1)
function [mu_0,mu_1, sigma, pi_0, pi_1,a]=FisherLinearDiscriminantAnalysis_train(X, Y)
function [Y]=FisherLinearDiscriminantAnalysis_test(X, mu_0, mu_1, sigma, pi_0, pi_1,a)
```

being X the input data, Y the classes, mu_k the mean vectors for the attributes, pi_k the prior probabilities for the classes, sigma the pooled covariance, and a the Fisher projection matrix.

Notes: With only 2 classes what should be the optimal rank for the projection "matrix" a? After Fisher projection using a what should be the dimensions of mu_k, and sigma?

Quadratic discriminant analysis: build the QDA algorithm as described on Section 4.3 either in its original quadratic version and in its simplified version (i.e., linear regression on the extended quadratic input space). The header of the functions should be:

```
function [mu_0, mu_1, sigma_0, sigma_1 pi_0,
pi_1]=quadraticDiscriminantAnalysis_train(X, Y)
function [Y]=quadraticDiscriminantAnalysis_test(X,mu_0,mu_1,sigma_0,sigma_1,pi_0,pi_1)
function [mu_0, mu_1, sigma, pi_0, pi_1]=quadraticDiscriminantAnalysisEZ_train(X, Y)
function [Y]=quadraticDiscriminantAnalysisEZ_test(X, mu_0, mu_1, sigma, pi_0, pi_1)
```

being X the input data, Y the classes, mu_k the mean vectors for the attributes in each class, pi_k the prior probabilities for the classes, sigma_k the pooled attributes covariance for each class, sigma the full pooled covariance.

Note: In the EZ version what should be the dimension of mu_k, and sigma?

Hints: In Matlab you can easily add a column to a matrix by using X = [X new_x] being new_x a column vector. In Matlab, the element by element product of two vectors is x.*x

Logistic regression (OPTIONAL): by following the algorithm of Section 4.4 in the textbook implement the logistic regression classifier with the following header

```
function [beta]=logisticRegression_train(X, Y, beta_init, iterations)
function [Y]=logisticRegression_test(X, Y, beta)
```

being X the input data, beta_init the initial vector of parameters (including the constant term), iterations the number of Newton steps, and Y the classes.

Hint: if no convergence is obtained, consider halving the Newton step size and iterate again ...

Exercise 4: LAR and LASSO Algorithms (8 Points)

Code the LAR algorithm in Matlab , as described in Section 3.4.4 of the textbook "The Elements of Statistical Learning". Additional details can be found in the Least Angle Regression paper (http://www.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf), Section 1-3.

Note: Since the solutions are not equivariant under scaling of the inputs, remember to first standardize the predictors in the dataset, and fit a model without an intercept (as for ridge regression, see comment at the end of page 63 of the textbook).

1) Implement the following functions:

- `function beta_coeff = lars(X,y,n-steps)`

  Parameters
  `X: predictors matrix`
  `y: output vector`
  `n-steps: number of steps of the algorithm`

  Return value:
  `beta_coeff: coefficient vector`

- `function t = larRegressionPath(X,y)`

  Parameters
  `X: predictors matrix`
  `y: output vector`

  Return value:
  `t: vector of values (t=\sum|\hat beta_i|) that compose the regression path`

2) Using the dataset "prostate" at http://www-stat.stanford.edu/~tibs/ElemStatLearn/data.html, test the LAR algorithm, and create the following plot:

   y-asis: the value of each of the \hat beta_i
   x-axis: t=\sum|\hat beta_i|,

   as in Least Angle Regression paper, Figure 1

3) Starting from the LAR implementation, add the LASSO modification described in Algorithm 3.2a of the textbook, write the corresponding lasso functions as in 1)

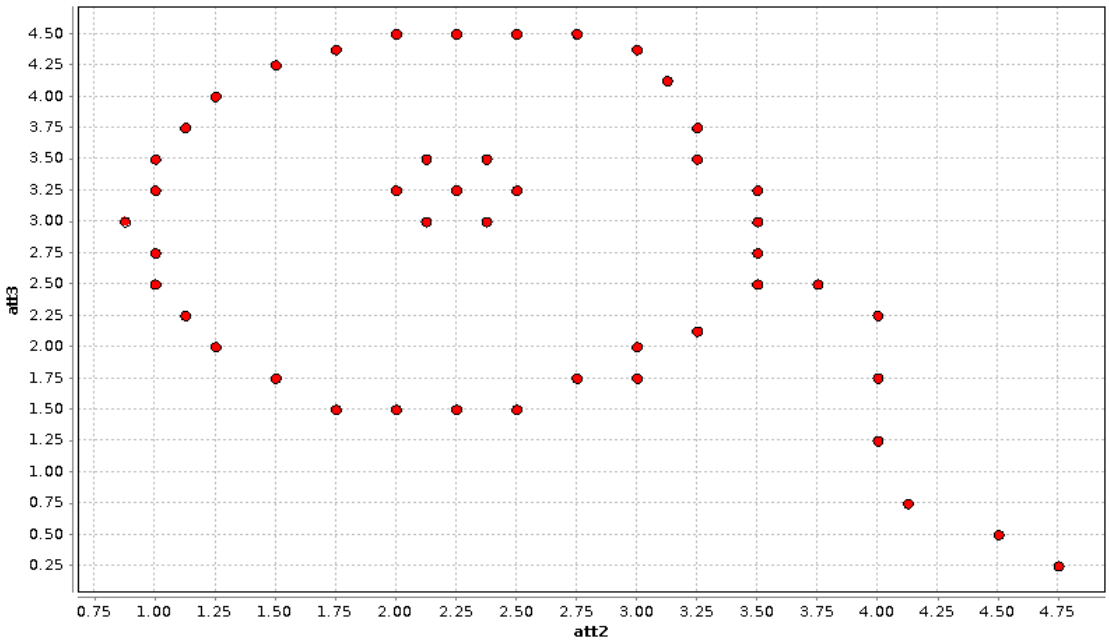
```
function beta_coeff = lasso(X,y,n-steps)
function t = lassoRegressionPath(X,y)
```

   and test the algorithm over the prostate dataset as in 2)

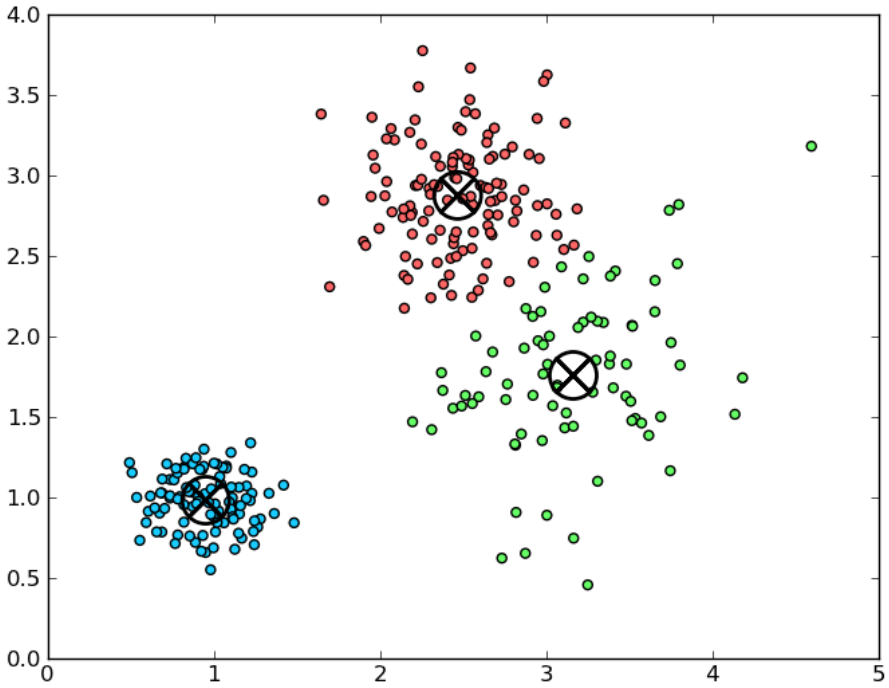
4) Compare the results with those obtained with LAR. Can you see any difference? Comment on the maximum value for n-steps for both algorithms.

Exercise 4: Clustering and friends (8 Points)

Given the dataset shown in figure (and available as a CSV file on the course website) perform the following operations:



1) Comment on the disposition of points in the dataset. What is peculiar in this dataset and how does it differ with respect to another dataset like the one in the following figure? How would you group points "naturally" if you had two clusters? And if you had three?



2) Cluster the data with a K-Means algorithm, choosing K=2 and K=3. How do the two executions perform? Does increasing the number of clusters allow you to better group data in "natural" clusters?

3) Cluster the data with an agglomerative hierarchical algorithm (use single linkage to calculate the distance between clusters). How is the resulting clustering in this case? Is it more similar to the "natural" clustering of the dataset?

4) For the three different clustering results you obtained (K-Means with K=2, K-Means with K=3, Hierarchical) evaluate their quality in terms of their WSS. Which one is higher and which one is lower? What does these differences mean? Which is the best performing algorithm according to the calculations you did? Is it coherent with your idea of "natural" clusters?

5) (OPTIONAL) Repeat the clustering using another algorithm chosen between DBSCAN and Jarvis-Patrick (feel free to try different choices for the parameters of both the algorithms), evaluate it as you did with the others in point 4, and comment the results.

Note: You can freely choose to solve this exercise either manually or automatically, by using Matlab, RapidMiner or similar tools. In any case, be as verbose as you can providing the steps you have performed to do the actual clustering, the results (well commented, to show you did not just copied and pasted some text from the screen), and your personal notes (i.e. which tool you have used, with which parameters, source code if you have developed a tool, and so on).