Lupus prognosis: a clinical study

YANN BRULE

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2000

Acknowledgements

I would like to thank Dr C. GORDON for the database she has provided to the department of Neural Computing and Research Group and for the time she has allocated to give advice and to help me in the analysis of the behaviour of the disease.

My sincere thanks to my supervisor Dr I. NABNEY for his availability, guidance and patience. I would like to add thanks for having provided the library of matlab functions NETLAB.

Last but not least, thanks to Miss V. BOND for her help all along the year in the administrative tasks which has permitted me to be concentrated on my work.

Contents

List of Figures

List of Tables

Introduction

This thesis relies on a partnership between Dr. C. GORDON from the Department of Rheumatology of the Faculty of Medicine and Dentistry and the Neural Computation Research Group. Dr. GORDON works on a disease which is called **Lupus**. A lot of data has been collected throughout different hospitals from the Midlands on people who suffer from this disease and has been put together in a database. At each visit the patients undergo some clinical tests and answer questionnaires to better follow the evolution of the disease. My goal was to know if, using the database and some neural networks, it is possible to predict the global flares of the disease and moreover in which part of his body the patient is most likely to have complications.

In this kind of medical problem, you can usually think of two different ways to treat it: either you adopt graphical models or you use neural networks. In this study, this is the second method which had been used for simple reasons: the first and most important one is that in graphical models, you need the expert to work with you in order to build the model. This task is quite long and the doctor must give a lot of his/her time, which is not always possible. Moreover, the computations to run on the graphical models for learning are very time-consuming. For these two main reasons, neural networks have been used in this study.

Keeping in mind that we wanted to predict the flares of the disease, I have worked on different problems which will be detailed in this thesis. The first one was to find out which variables were the most reliable to help us in the prognosis. I have tried several different techniques, the first ones based on linear dependencies while the last one is a Bayesian technique called Automatic Relevance Determination.

Then we tried to find out using some variables among the clinical tests, if it is possible to predict the score[1] of the general state of health of the patient. This problem is a *regression* problem because, here, we try to predict the exact score. We wanted, with the help of the neural networks, to give an approximation of this score at the next visit knowing only the results of some clinical assessments.

Last but not least, we have tried to *classify* the visits into two groups: the visits for which there will be a flare up of the disease at the next visit and the ones which will stay normal. This problem is quite different from the first one: we try to predict if the score will be greater than a threshold or not. The prediction does not inform us of the exact value of the score which can be slightly greater than the threshold or very high! It is a binary problem: we try to predict if there will or will not be a flare up in the disease.

So at the start of this thesis, you will find first a quick introduction to lupus. After this, I have presented the results I have obtained working on the different problems quoted above: the selection of variables, regression and classification problems.

[1]Total BILAG score as will be later described

Chapter 1

Lupus

1.1 What is lupus?

Lupus is an autoimmune disease in which the patient's immune system creates antibodies which, instead of protecting the body from bacteria, viruses or other foreign matter, attack the person's own body tissues. This causes symptoms of fatigue, joint pain, muscle aches, anaemia, general malaise, and possibly destruction of vital organs.

Lupus is neither infectious nor contagious, the cause is not known though research has provided evidence implicating heredity, hormones, and infections. including viruses. The disease lies dormant in the body until some trigger from outside sets the process in motion.

Lupus mainly attacks women during their child-bearing years but men and even children can be affected. In the U.K., 1 in 750 women suffer from lupus, with the ratio of women to men being 9:1.

There is at present no cure for lupus but careful monitoring of the disease and a treatment program with medication adjusted as appropriate enables the condition to be controlled, so most patients are able to live a normal life span.

1.2 The Database

1.2.1 Overview

The database contain the data of more than 7000 visits concerning 430 patients. Each visit is represented by 78 variables. Some of these variables are categorical while others are continuous:

- variables which characterise the patient,
- variables which characterise the visit,
- clinical values which are divided into (with their number into brackets):
  - Points values (9),
  - SLICC values (14),
  - SF36 values (12),
  - Fxxx values (35) [each x represents a number].

It is important to notice that not all these fields are filled for every visit. A particular test may not be carried out because it is expensive, inconvenient for the patient or it is only done for the people who have certain symptoms.

1.2.2 The patients

The patients are defined using a number of fields they have to fill in as to their date of birth, sex or race. As said above, the number of patients is 430. Most of them (90%) are women and they are born between 1914 and 1980; the majority are between 30 and 40 years old. Finally, the most numreous race is coded 3 which corresponds to the Causcasian. So the "modal individual" is a women between 30 and 40 who is Caucasian.

1.2.3 The visits

The fields about the visits inform us of the location of the assessment tests, the person who did the test, the date of the visit... It seems to be interesting to speak at this point of the thesis about the frequency of these visits. As you can see in the histogram (Figure 1.1) this delay can vary between 3 weeks and several years! The distribution of the delays lower than 6 months can be found in Figure 1.2. This variety in the range of delays has several different causes. The first reason is that when a patient comes to a consultation, the doctor usually asks him either to come back in three months if everything is fine or if the disease does not show any evolution, or in one month if he thinks there is a risk of flare or if the disease is active for a short time. Moreover, these patients do not always come on regular basis because of professional reasons, travelling, etc ...

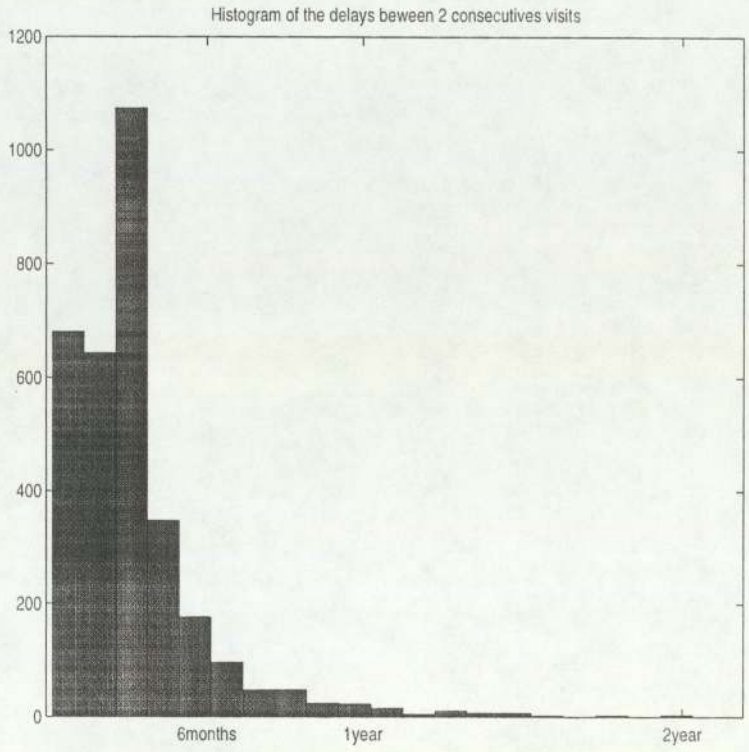


Figure 1.1: Histogram of the delays between two visits

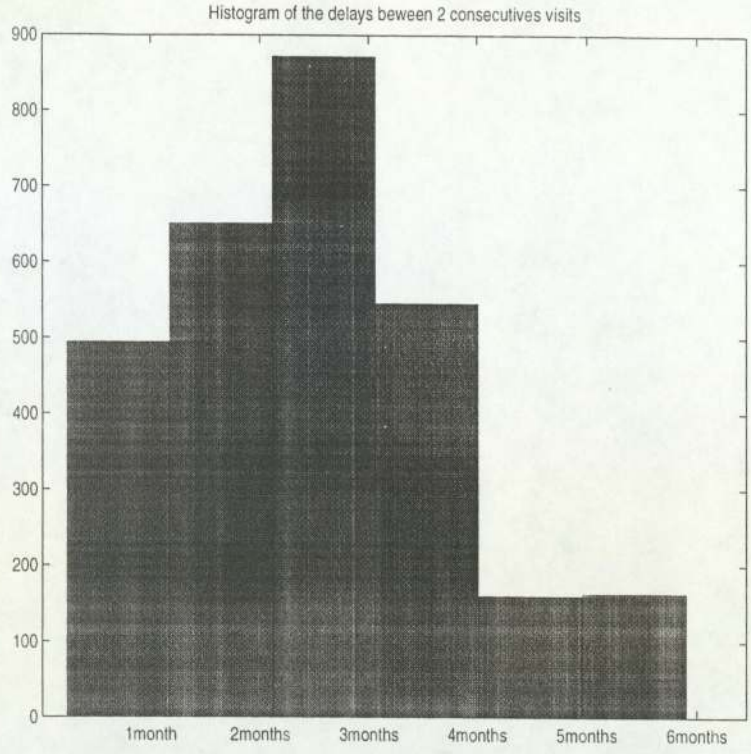


Figure 1.2: Histogram of the delays less than 6 months

1.2.4 The clinical values

These variables are the most interesting because they are the ones which are in direct relation with the disease.

First of all, there are the Points values. They assess (thanks to a score which uses the scale 0, 1, 3, 9) the *state of health* of different parts of the body: general, mucocutaneous (related to the skin), neurological, musculoskeletal, cardio-respiratory, vasculitis (related to the blood vessels), renal and haematological. Each system is evaluated thanks to a categorical score assessed by the BILAG index. The last variable of this group is the total BILAG score which represents the sum of all the individual BILAG systems. Throughout this thesis you will see that either the total BILAG score or the individual systems have been considered. Section 1.3 describes in details what the BILAG index is and how the doctor can give a score which represents the state of health of parts of the body.

The SLICC index is a measure of disease damage acquired over time either due

to disease activity, therapy, or other types of disease. This assessment is done every 6 months because a clinical feature has to be present for at least 6 months in order to score as damage. As for the BILAG score, there are individual systems which are evaluated separately. This variable is a categorical variable.

After this, there are the SF36 (or Short Form 36) variables which represent a general health survey. In agreement with the doctor, this group of variables has not been studied.

And finally the Fxxx variables. They are the clinical tests. Some of these variables are continuous, others categorical. They give us different information:

- they are markers and/or "predictors" of the disease;
- they identify particular types of clinical disease activity instead of the amount of disease activity;
- they keep a track of the drug treatment (if there is a treatment, the change of dose...).

1.3 The BILAG index

For a better understanding of what's going on, it seems to be important to explain clearly what the BILAG index is. First of all, BILAG means British Isles Lupus Assessment Group. It was developed in 1984 in United Kingdom according to the principle of the physician's "intention to treat" on the premise that, while physicians might not agree about the significance of individual clinical features or laboratory tests, there was broad agreement about when to treat lupus.

The index allocates separate alphabetic scores to each of eight organ-based systems (general, mucocutaneous, neurological, muscoskeletal, cardiorespiratory, vasculitis and thrombosis, renal, haematological). These scores and their meaning are described in the Table 1.1. To make this index reliable, all the doctors have to answer some questions for each system which will determine the score attributed. You will find more information about the BILAG index in [5].

In our study, each alphabetical score has been replaced by a numerical value wich lies between 0 and 9. You can see the exact correspondence on the Table 1.2. These scores had been chosen by the doctor and are based on her judgement and experience: it can be admitted that three B's (3) are equivalent to an A (9), and three C's (1) to a B. This scale seems coherent so we have not modified it.

Category	Meaning
A	Denotes disease thought to be sufficiently active to require disease-modifying treatment
B	Denotes disease which is less active than in "A"; mild reversible problems requiring only symptomatic therapy
C	Indicates stable mild disease
D	System previously affected but currently inactive
E	Indicates system never involved

Table 1.1: Scoring system for the BILAG index

Alphabetical value	A	B	C	D	E
Numerical value	9	3	1	0	0

Table 1.2: Correspondences alphabetical/numerical values for the BILAG index

Knowing this background, the problem appears to be clearer: is it possible to make some forecasts of the BILAG scores (total or individuals) knowing the values of the clinical tests? But, as you will see in the next chapter, there was a preliminary task to select the right variables to use for this prediction.

Chapter 2

Variable Selection

A big issue in the study was that we had two problems in one: finding the variables which could help us to predict the general state of health of the patients means, without even being sure that it was possible to predict anything, selecting the *right* variables among all of them (35 in total). We could not use all the variables for two reasons: the first one is if we do so, the data sets are too small after the preprocessing to estimate parameters reliabily. The second reason is that we want to detect which tests are the most relevant so as to minimize the number of tests carried out on the blood samples.

To find a *good* selection, we tried different techniques: we begun with computing the correlation between the inputs (clinical tests) and the outputs (total BILAG score). After this we tried Principal Component Analysis and finally we used a Bayesian technique: Automatic Relevance Determination.

2.1 Correlation

It is usually a good thing to begin with computing the correlation matrix between the inputs and the outputs because it measures the linear dependencies between variables. But before going further, you have to keep in mind that even if there are no good correlations, a Neural Network may give good results! Indeed, their structure allows them to estimate non-linear function which means that even if there is no linear relation between the outputs and the inputs (measured by the correlation), the estima-

tion of the function may be very good! The correlation was computed because it could show some linear relations among the inputs or between the inputs and the outputs that would allow us to take account of only the relevant variables . The correlation matrix can be found in the Appendix A where you will find only very poor results i.e. there are only small correlations.

2.2 Cross-correlation

The goal of this part was to detect if there was a time lag between the assessment of the test variables and the response of the disease. A way to confirm such an hypothesis for 2 time series is to compute their cross-correlation coefficients. These coefficients can show when 2 time series are correlated with a lag. The coefficient of the cross-correlation at lag k between 2 time-series x and y which have N points are given in the Equation 2.1. You can notice that for the lag k=0. it corresponds to the correlation coefficient.

$$c_{xy}(k) = \begin{cases} \sum_{t=1}^{N-k}(x_t - \overline{x})(y_{t+k} - \overline{y})/N & k \in \{0, 1, \ldots, N-1\} \\ \sum_{t=1-k}^{N}(x_t - \overline{x})(y_{t+k} - \overline{y})/N & k \in \{-1, -2, \ldots, -(N-1)\} \end{cases} \tag{2.1}$$

For this part of the study, we only considered the relation between the tests variables and the total BILAG score. Because the cross-correlation is applied on time series, we have picked up a sample of 7 interesting patients indicated by Dr GORDON. These patients are considered interesting because we can observe some relations between the tests and the corresponding BILAG scores. We have computed for every patients the cross-correlation between their total BILAG score and around ten variables. The results, such as the one shown in Figure 2.1, were not very interesting once more. As you can observe, the graphs, for all the patients and all the variables, are nearly constant at zero. It would have been interesting if we had found a common peak at a lag τ_0 considering one variable and all the individuals of the sample. You will find the other graphs in Appendix B.

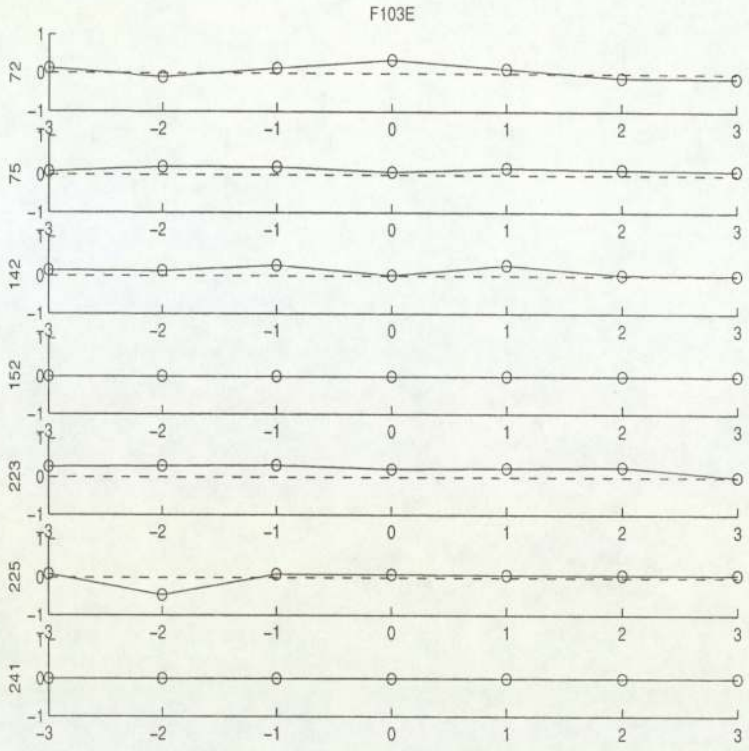


Figure 2.1: Cross-correlation between the total BILAG score categorized and F103E

2.3 Principal Component Analysis

The Principal Component Analysis is a simple technique to project the data linearly to a lower dimensional space by maximizing the variance of the observation along the new axes. This technique is based on the computation of the eigenvalues of the correlation matrix. Doing this, we hoped to see some variables which would be more relevant than the others thanks to the percentage of variance explained by the axes. We computed PCA on 2 data sets: the first one had all the clinical tests while the second one only contained the variables advised by Dr GORDON. As you can see in the Appendix C, we did not obtain useful information with this process. That seems correct since this process relies on the correlation matrix!

2.4 ARD

2.4.1 Introduction

Automatic relevance determination is a Bayesian technique based on the *evidence* framework introduced by MacKay in [9]. Each input is associated with a hyperparameter. A hyperparameter is a parameter which belongs to a higher level of inference: it controls the distribution of others parameters. In our case, when one of these hyperparameters acquires a large value, the corresponding input can be discarded because it is considered as irrelevant. The problem is now to estimate these hyperparameters.

2.4.2 The evidence framework

Here we will have to introduce a lot of notions about neural networks. Some will be defined here while the others will be seen in Section 3.3. A complete definition of a neural network and all its attributes can be found in this part. Futhermore, as I said in the introduction, two different problems have been considered: the regression and the classification problems. Because they are different, we need to distinguish them to explain the way the hyperparameters are computed.

Nevertheless, in the two cases, the goal is the same: we will analyse the probability of the data given the hyperparameters. Once this is done, we can find a way to compute the values of the hyperparameters which maximize this probability.

Classification problem

First, you have to know that the goal of neural networks is to learn the mapping which is defined by some pairs of input and target vectors $D = (x^n, t^n)$. The output of the network for the given input x^n is noted y^n. For a 2-class problem, the target vectors use 1 (for class C_1) and 0 (for class C_2). These represent the probabilities that the corresponding input pattern belongs to class C_1. Given this, the probability of the

data (likelihood) is written as:

$$p(D|w) = \prod_n p(t^n|x^n) \tag{2.2}$$

$$p(D|w) = \prod_n (y^n)^{t^n}(1-y^n)^{1-t^n} \tag{2.3}$$

Furthermore, if we take negative logarithms

$$E_D(D|w) = -\sum_n (t^n \ln y^n + (1-t^n)\ln(1-y^n)) \tag{2.4}$$

we obtain:

$$p(D|w) = \exp(-E_D(D|w)) \tag{2.5}$$

$E_D(D|w)$ is an error function: we can use it to assess the performance of our network.

On the other hand, we assign a prior over the weights, i.e. we assume that the weights are generated from a particular distribution, of the form:

$$p(w|\alpha) = \frac{\exp(-\alpha E_W(w))}{Z_W(\alpha)} \tag{2.6}$$

where $Z_W = \int \exp(-\alpha E_W)$ is a normalization factor to have the property $\int p(w|\alpha)dw = 1$. By writing the error $E_W(w)$ as

$$E_W(w) = \sum_i \frac{1}{2} w_i^2 \tag{2.7}$$

where the w_i are the weights of the networks, we can see that large weights are penalized. When they actually are large, the error E_W is large and the probability $p(w|\alpha)$ is small. Large weights must be penalized because they may lead to poor generalisation as they can be caused by the network overfitting[1] to the noise in the training data.

Then, using Bayes' rule, we can compute the posterior distribution of the weights:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} \tag{2.8}$$

$$p(w|D) = \frac{\exp(-E_D(D|w) - \alpha E_W(W))}{Z_M} \tag{2.9}$$

where Z_M is again a normalization coefficient defined by $Z_M = \int \exp(-E_D(D|w) - \alpha E_W(W)dw)$. Our goal is to maximize this probability i.e. find the weights of the

[1] cf Section 3.3.1

network which are the most probable given the data. At this point, you can notice that maximizing this probability is equivalent to minimizing the function:

$$M(w) = \alpha E_W(w) + E_D(D|w) \tag{2.10}$$

This function is nothing but the global error of the network. The term $\alpha E_W(w)$ is a weight decay term which penalized the large weights as seen above. This way we regularize[2] the model by favouring small values of the weights.

Now, concerning the probability of having the hyperparameter given the data, one can write it using Bayes' rule:

$$p(\alpha|D) = \frac{p(D|\alpha)p(\alpha)}{p(D)} \tag{2.11}$$

Then, if $p(\alpha)$, the prior distribution of the hyperparameter α, is chosen to be very insensitive to the value α and because the denominator does not depend on α, it can be noticed that the probability $p(\alpha|D)$ is maximum when the quantity $p(D|\alpha)$ is maximum. Thus, we call this latter probability the *evidence*. An other way to write it is:

$$p(D|\alpha) = \int p(D|w,\alpha)p(w|\alpha)dw \tag{2.12}$$

then,

$$p(D|\alpha) = \frac{\int \exp(-E_D(D|w) - \alpha E_W(w))dw}{Z} \tag{2.13}$$

$$\text{or} p(D|\alpha) = \frac{Z_M(w)}{Z} \tag{2.14}$$

$$\text{or} p(D|\alpha) = \frac{\int \exp(-M(w))}{Z} \tag{2.15}$$

Z is again a normalizing constant. Finally, maximizing this probability is equivalent to maximizing the function $Z_M(w)$

Thanks to this last equality and after some computations, we now can find the best α. Let assume that M has a single minimum at reached at the point w_{MP}. Then if A is the Hessian matrix of M evaluated at w_{MP}, $\nabla\nabla M|_{w_{MP}}$, and if its eigenvalues are

[2]see Section 3.3

described as $(\lambda_i + \alpha)_{i \in \{1..W\}}$, we have γ, the number of parameters determined by the data, which can be obtained by:

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \alpha} \tag{2.16}$$

And then the hyperparameter α can be computed at each iteration of the training with the formula:

$$\alpha^{new} = \frac{\gamma}{2E_W} \tag{2.17}$$

Regression

A major difference between the classification and the regression problems is in the choice of their error function. In the latter case, it is the following error which is considered:

$$E_D(D|w) = \sum_n \frac{1}{2}(y^n - t^n)^2 \tag{2.18}$$

The performance is so measured by the distance between the target vectors and the outputs of the network. Then, the outputs of the networks can be considered as the probability to have the target vector given its input value with the distribution:

$$P(t^n|x^n, w, \beta) = \frac{\exp(-\beta E(t^n|x^n, w))}{Z_n(\beta)} \tag{2.19}$$

where $Z_n(\beta) = \int \exp(-\beta E)$ is once more a normalization factor. E is the error for a single pattern and β is a new hyperparameter which represents the inverse variance of the gaussian noise. Considering the whole data set, we obtain:

$$P(D|w, \alpha) = \frac{\exp(-\beta E_D(D|w))}{Z_D(\beta)} \tag{2.20}$$

In the same way that in the part above, we write:

$$M(w) = \alpha E_w(w) + \beta E_D(D|w) \tag{2.21}$$

and then the hyperparameters can be computed thanks to the formulas:

$$\alpha^{new} = \frac{\gamma}{2E_w} \tag{2.22}$$

$$\beta^{new} = \frac{N - \gamma}{2E_D} \tag{2.23}$$

Summary

We can summarize the estimation of the hyperparameters in both cases with the steps:

1. Initialize the α's parameters using some priors knowledge or at random (respectively β).
2. Train the network to minimize the function M(w).
3. Compute the hessian matrix.
4. Re-estimate the α parameter using 2.22. (respectively β).
5. Return to 2. until α (respectively β) converges.

2.4.3 ARD

The goal here is to see which inputs are the most relevant. To determine this, we use Automatic Relevance Determination. As said in the introduction above, this process consists in associating a hyperparameter α_i for each input. The values of the α_i are then computed during the training process using the evidence framework described above. At the end, if the value of any of the α_i's is large, we can discard the corresponding input. If it is large, it means that penalty associated to the corresponding input is large to prevent the corresponding output from causing overfitting.

So, I have first run the ARD process considering the variables which are assumed to be predictors of the disease by Dr GORDON. To these variables, I have added 3 other variables, F108, F133 and F75 which I thought were irrelevant. The regression problem has first been considered. The results are in the Table 2.1 and show that all the variables, except F108, are relevant. So this result does not give us additional information and has been discarded.

As regards the classification problem, the result is different. After the first process, six variables were pointed out as relevant whilst the others were uninteresting. Then, a second ARD process has been run based on the results of the first one. We have

finally obtained a selection of four variables: F112, F113, F119C and F141. It is quite reassuring that the added variables do not belong to this selection. Let describe what these variables represent: F112 is the C4 complement test which goes down with disease activity. F113 is the C3 complement degradation product, C3d, which was thought by Dr Gordon to be an accurate predictor of the disease since it reflects consumption of complement due to disease activity and is independant of the rate of synthesis of this protein. F119C is the C-reactive protein (CRP). Finnaly, F141 is the change of steroid dose and follow the coding system: 1 for reduced, 2 for the same, 3 for increased and 4 for start. This group of 4 variables is coherent in terms of following the disease: we have information on the evolution of the disease with the three first variables whilst the last one indicates the treatment received by the patients which influneces the evolution of the disease.

To assess the pertinence of this result, we have tried the classification problem using these four variables. As you will be able to judge after having seen Chapter 4, the results shown in Figure 2.2 are surprisingly good! The percentage of detection of future flares is between 90 and 100% while the false positive rate is around 10%. These are the best results obtained for the classification task. So a normal thing to do at this point would have been to run the ARD with each system to see which variables could be interesting for each classification task. Unfortunately, this has not been done because of time constraints. It could constitute a good basis for future work.

	F103E	F111	F112	F113	F108	F119C	F87	F121	F141	F133	F75
α	0.00	0.00	0.00	0.00	0.37	0.00	0.00	0.00	0.00	0.00	0.00

Table 2.1: Regresssion problem: the results of the ARD process

	F103E	F111	F112	F113	F108	F119C	F87	F121	F141	F133	F75
α	1800	133	35	55	341152	39	99	18	6	26	222

Table 2.2: Classification problem: first ARD process

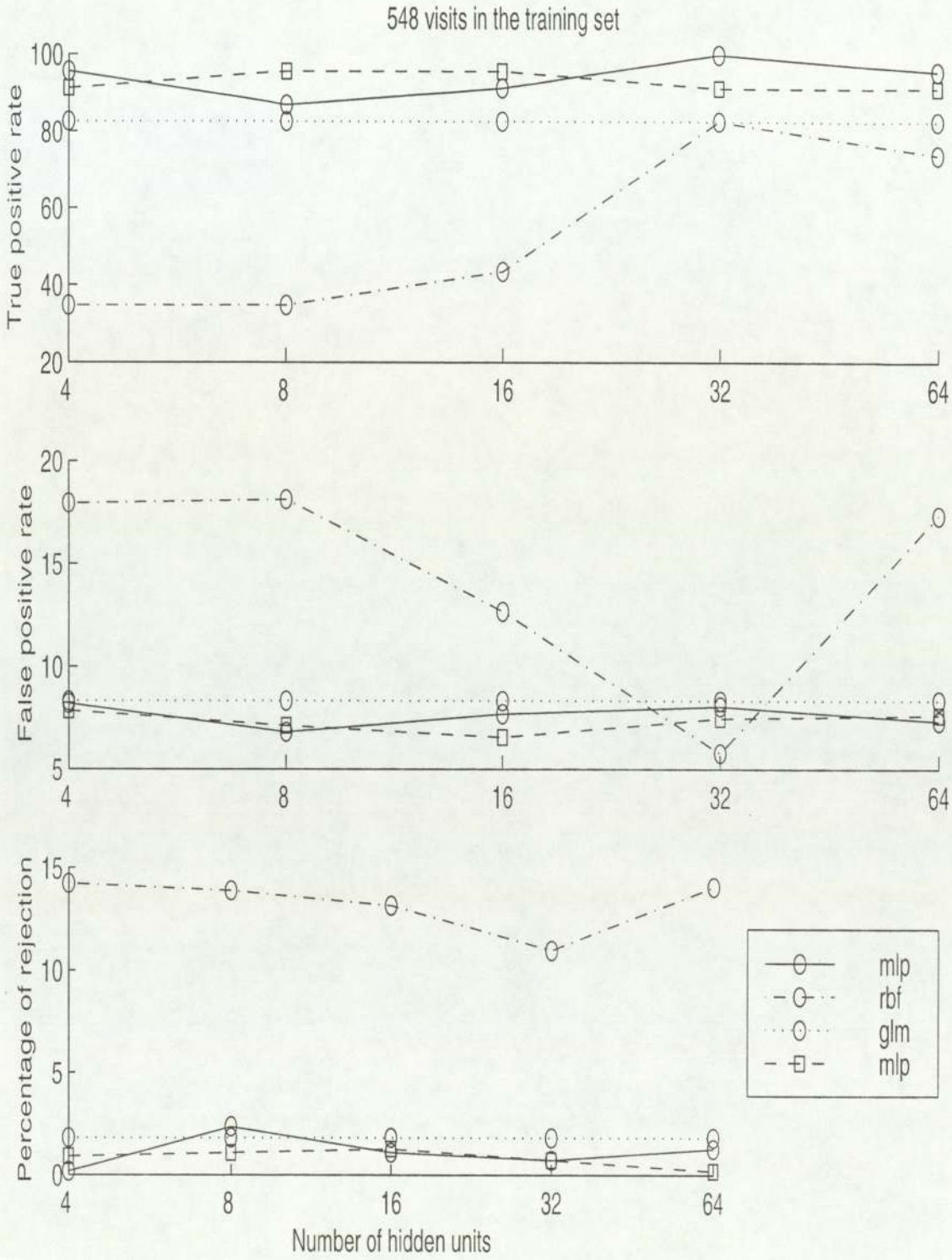


Figure 2.2: Results for the classification problem for the total BILAG score. The inputs variables are F112, F113, F119C and F141.

	F112	F113	F119C	F121	F141	F133
α	5.58	3.12	10.08	27.84	4.65	99.79

Table 2.3: Classification problem: second ARD process

Chapter 3

Regression problem

3.1 The goal

The problem mainly consisted in forecasting the value of the BILAG score (total or individual) of the next visit given a selection of clinical tests. This problem is a regression problem, because knowing some variables, we try to find the corresponding value which can be considered as the result of a function f applied on these first variables. The clinical tests used were chosen thanks to different criteria studied in the Chapter 2 so we will not speak about these choices here.

Since the nature of the inputs are very different, some preprocessing has been done to make them have equal weight for the neural networks used. Moreover different data sets have been created to obtain the maximum of information from our data. Finally, we have used several types of networks to see which one could give the best results.

3.2 Preprocessing of the data

The first procedure performed on the data has always been the same: remove the visits where there was any missing value. Most of the time, this had not been a problem in term of *size* of the data sets. Because the database was really big, we usually had enough data to work on (it depends on the variables we had selected and on the type of data set used). But another issue arose here: by removing some visits, we lose

the trend of the disease, we remove some steps in the disease which can be important! Another point is that we had to remove 3 variables because they have too many missing values: F103, F110, F88. The problem was that the variable F103 was believed to be a good predictor of the disease. But, because we had no time to work on the *missing value* problem, we have discarded it. A histogram of the percentage of missing values for every variable can be found in Figure 3.1. We have plotted these histograms for a sample of variables in Figure 3.2.

Moreover, because an input with large values would have more weight for the neural network, all the data are standardized before any work is done. For each input, we apply the following transformation ($\overline{x_i}$ is the mean of the input x_i and σ_i its variance):

$$\widetilde{x_i^n} = \frac{x_i^n - \overline{x_i}}{\sigma_i}$$

Finally, we wanted neural networks to learn the standard response of the organism in response to the disease, so we have removed 2% of the extreme values (at the top and at the bottom) for every data sets.

Some further preprocessing has been carried out on the data. This way, we obtained, for each selection of variables, 4 different data sets which are the *Normal* set, the *Pred* set, the *Diff* set and the *Evol* set. The *Normal* set has no further transformation while each of the others uses a different representation.

The *diff* set

We decided to create this data set to answer the question: "Can a neural network learn what the difference is between two BILAG scores from two successive visits knowing the differences in the blood tests for these two visits?". Schematically here is what is done for the matrix of all the visits of a patient after having done the standardization: x_i is the input (one variable) and stands for a test undegone at time t_i while y_i is the output and stands for the corresponding BILAG score obtained at the same visit.

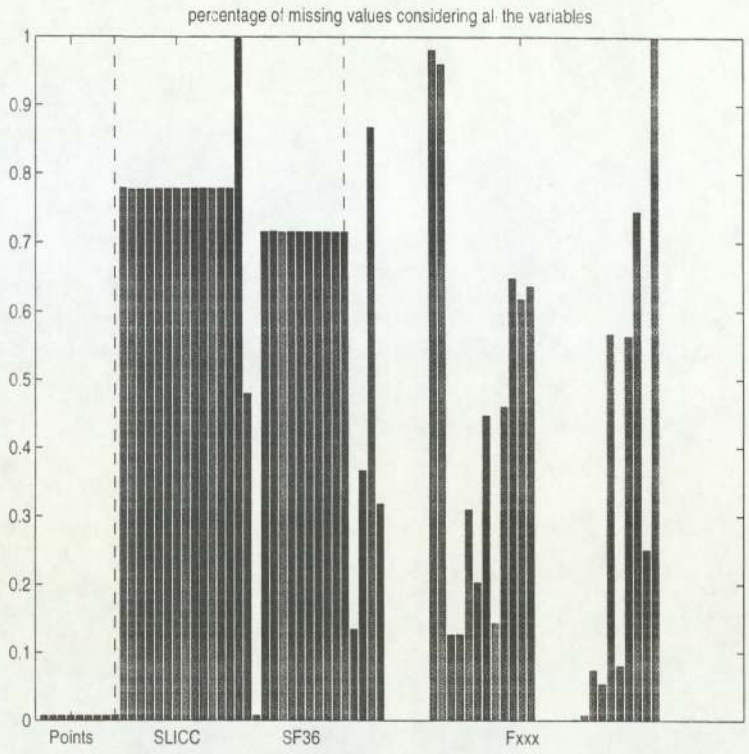


Figure 3.1: Histogram of the percentage of missing values considering all the variables

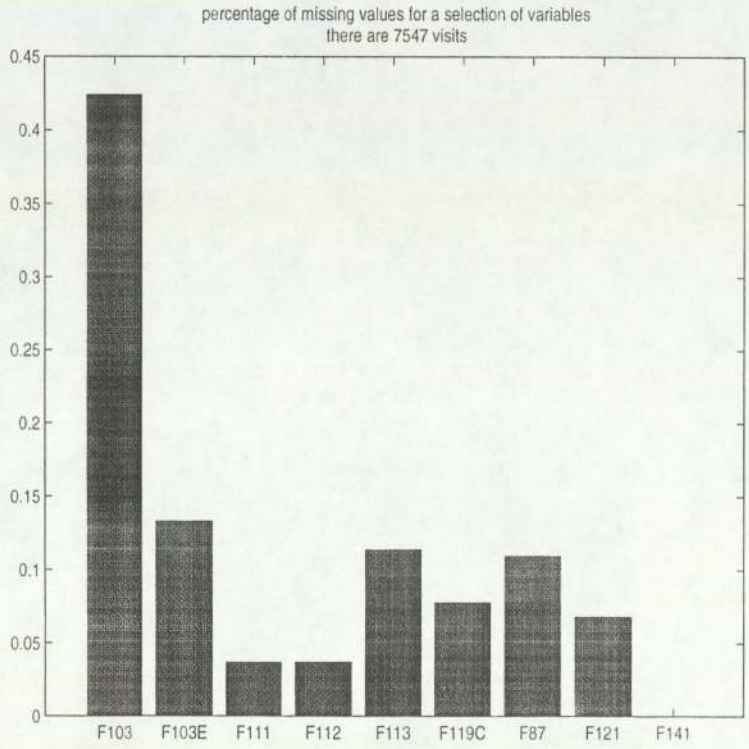


Figure 3.2: Histogram of the percentage of missing values considering a selection of variables.

$$\begin{bmatrix} x_0 & y_0 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \rightarrow \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \\ \vdots & \vdots \\ x_n - x_{n-1} & y_n - y_{n-1} \end{bmatrix}$$

the *pred* set

The goal of the study being to predict the flares of the disease, it was natural to build the following set: given the test variables at time t we try to predict the BILAG score at time $t+1$.

$$\begin{bmatrix} x_0 & y_0 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \rightarrow \begin{bmatrix} x_0 & y_1 \\ \vdots & \vdots \\ x_{n-1} & y_n \end{bmatrix}$$

The *evol* set

The last set is a combination of the 2 latest transformations. It just means that using the evolution in the blood tests (differences between 2 visits for a test), we try to predict the evolution (raise) of the BILAG score.

$$\begin{bmatrix} x_0 & y_0 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \rightarrow \begin{bmatrix} x_1 - x_0 & y_2 - y_1 \\ \vdots & \vdots \\ x_{n-1} - x_{n-2} & y_{n-1} - y_{n-2} \end{bmatrix}$$

3.3 Neural Networks Used

3.3.1 Introduction

In this section, we will study the use of neural networks in the regression problem. All that is said below can be extended for the classification problem; if there are some differences it will be explained in the corresponding section.

A regression problem consists in finding a scalar variable y given other variables x. In our context, y is called the output while x is a vector whose each dimension corresponds to a variable. In our model, y is considered as the output of a function f evaluated at the point x plus some noise added:

$$y = f(x) + noise$$

A neural network is a structure which is flexible and which should estimate the function f. Before seeing how we make it learn the function, we describe what a neural network is.

First of all, as its name shows, it is a set of neurons. Each of them takes something as input and computes the output with its own *activation function*. The activation function allows the neurons to have different outputs as a function of the inputs. Some simple examples follow which are the linear (Equation 3.1) and the logistic (Equation 3.2) activation functions. A graphical representation of a neuron is presented on the Figure 3.3.

$$f : a \rightarrow a \tag{3.1}$$

$$f : a \rightarrow \frac{1}{1 - e^{-a}} \tag{3.2}$$

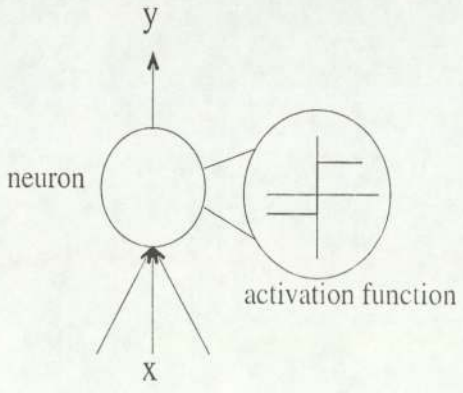


Figure 3.3: Representation of a neuron

Because it is a network, all the neurons are connected to other neurons. The architecture of the network can vary: all the neurons can be connected together, it can

be feed-forward (the outputs are well-determined as a function of the inputs) or not have a particular structure. In the network, each connection possesses a weight. This weight represents the importance of the contribution of the neuron in the response (output): if the weight is high, it contributes a lot whereas if it is low, it participates less.

The weights are adjusted to allow the network to *learn* the function f. This is possible because we *train* the network with some examples. The training process corresponds to an update of the weights to make the network better approximate the function.

In fact, we take all the data (inputs and outputs) and divide them in three different sets. The first one is called the training set and is used during the training process described above. The second one is the validation set which permits us to detect when there is *overfitting*. We say there is *overfitting* when the network learns the noise. It just means that it fits quite well the training data but does not give a good approximation of the function f which generates the output data. Indeed, it is said that the network does not generalize: it gives good answer for new points which belonged to the trainind data set while for others it will give an answer far beyond the range of the training data. An illustration is in Figure 3.4. The validation set is used to detect when the network begins to overfit the data. We compute the global error on this set and controls that it goes down. When it starts increasing, we stop the training and prevent this way the network from overfitting. This technique is called the *early stopping* method.

Finally, the last set is the test set which is used to assess the performances of the network. With this last set, we obtain an unbiased estimate of the *generalization error*. In a regression problem, the performances are assessed by the computation of an *error*. The error is a global error and it is measured with all the points of the test set. The error depends on the kind of problem treated. The one chosen in the regression problem

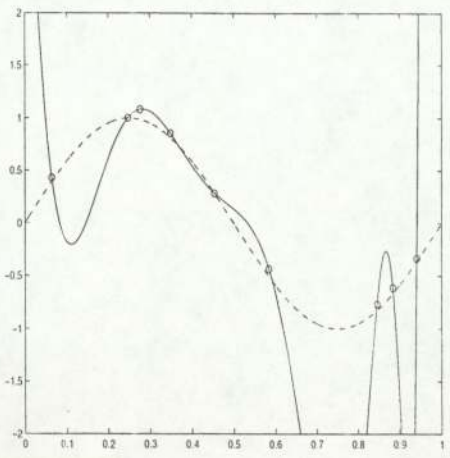


Figure 3.4: Illustration of overfitting: the network fits the training points (circles) well, but the underlying data generator (dashed lines) poorly

is the mean square error:

$$E = \frac{\sum_{n=1}^{N}\sum_{i=1}^{d_{out}}(y_i^n - t_i^n)^2}{2N} \tag{3.3}$$

N : number of data in the test set

d_{out} : number of dimensions of the output

y_i^n : i^{th} component of the output for the example number n

t_i^n : i^{th} component of the target for the example number n

There exist many different kinds of neural network. In the following, we will explain briefly the one we have used, their parameters and training methods.

3.3.2 Multi-Layer Perceptron

As you have seen above, a neural network is a flexible structure which can approximate a given function. A Multi-Layer Perceptron (MLP) is a feed-forward network composed of a succession of L layers of weights. To make it simpler, let's take the case where L=2 (the case used during the study and illustrated in Figure 3.5). We have two layers of weights, so there are three levels of nodes. We call them the inputs, the hidden neurons and the outputs. The outputs are the final layer of nodes, while the

hidden nodes are all layers of neurons between the first layer (inputs) and the outputs.

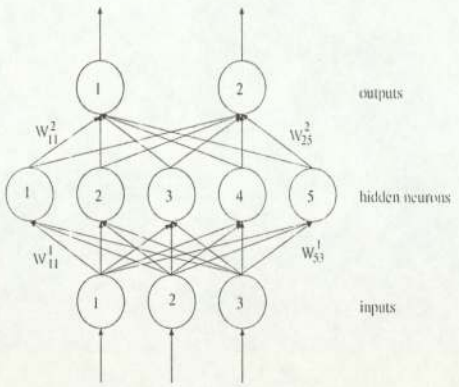


Figure 3.5: MLP with 2 layers of weights

Multi-Layer Perceptrons can have different number of units for their inputs, hidden neurons and outputs. Usually, the number of inputs and outputs are well-determined by the problem studied. Regarding to the number of hidden neurons, it is different. It controls the complexity of the mapping: the more hidden units you have, the more degrees of freedom your network has. You can make this parameter vary to obtain different answers. Now, let's define mathematically the functionality of the network.

If $(x^n, t^n)_n$ are the pairs of input and target vectors represented by vectors of dimension d_{in} and d_{out} respectively. Let g and h denote the activation functions of the first and second layer of nodes, and if w_{kj} is the weight associated at the connection between the k^{th} neuron of the layer $l+1$ and the j^{th} neuron of the layer l, we have, for a neuron of the hidden layer:

$$z_j = g\left(\sum_{i=1}^{d_{in}} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) \tag{3.4}$$

$$z_j = g\left(\sum_{i=0}^{d_{in}} w_{ji}^{(1)} x_i\right) \tag{3.5}$$

In the second equation we have just incorporated the *bias* (w_{j0}) into the sum thanks to an extra dimension for x, x_0, which takes the value 1. To have an interpretation of

this bias, let's take a simpler case. If we consider the problem of finding the space of solutions of the equation:

$$y(x) = w^T x + w_0 \tag{3.6}$$

Then, the space of solutions is an hyperplane of dimension $d-1$ perpendicular to the vector w (if x_1 and x_2 are solutions, $y(x_1) = y(x_2) = 0$ and $w^T(x_2 - x_1) = 0$). Moreover the distance between this hyperplane and the origin is determined by:

$$l = \frac{w^T x}{||w||} = -\frac{w_0}{||w||} \tag{3.7}$$

You can then observe that w_0 or the bias is representative of the distance between the space of solution and the origin (to convince yourself, you can try the case $y = x + b$). Moreover, to make the notation easier to read, we have always included the bias term in the following equations adding one dimension to the weights and to the input vector such that w becomes (w_0, w) and x $(1, x)$. We then obtain for the outputs of a MLP if *hid_out* is the number of hidden neurons connected to the outputs:

$$y_k = h\left(\sum_{j=0}^{hid\_out} w_{kj}^{(2)} z_j\right) \tag{3.8}$$

$$y_k = h\left[\sum_{j=0}^{hid\_out} w_{kj}^{(2)} g\left(\sum_{i=1}^{d_{in}} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right)\right] \tag{3.9}$$

You can then notice that it is not a linear model if g and h are not linear.

To complete the definition of the structure of this kind of network, the activation functions are *tanh* for the hidden nodes and linear function for the outputs. The error chosen is the sum-of-squares error function described in Equation 3.3.

Training

Now, it is interesting to see how this kind of network can be trained. The first thing to notice is that the training process is just a way to adjust the weights in the network in order to minimize the error chosen. To do so, we first have to compute the derivatives of the error with respect to the weights and, when it is done, we can

use a nonlinear optimization algorithm to compute the new values of the weights. The back-propagation algorithm is just a means to compute the derivatives of the error function.

Consider the error as the sum of all the errors obtained after the presentation of each pattern:

$$E = \sum_n E^n \tag{3.10}$$

where n is the number of patterns in the training set. Furthermore, we assume that we have already computed the activation functions for all the hidden and the outputs units. If we write

$$z_j = g(a_j) \tag{3.11}$$

$$a_j = \sum_i w_{ij} z_i \tag{3.12}$$

where z_j is the output of a neuron. We can write, using the chain rule,

$$\frac{\partial E^n}{\partial w_{ij}} = \frac{\partial E^n}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \tag{3.13}$$

$$\frac{\partial E^n}{\partial w_{ij}} = \delta_j z_i \tag{3.14}$$

where the terms $\delta_i = \partial E^n / \partial a_j$ depend on the node considered.

First, if the neuron is an output, $z_j = y_j$ and it is clear that:

$$\delta_j = \frac{\partial E^n}{\partial a_j} \tag{3.15}$$

$$\delta_j = \frac{\partial E^n}{\partial z_j} \frac{\partial z_j}{\partial a_j} \tag{3.16}$$

$$\delta_j = \frac{\partial E^n}{\partial y_j} g'(a_j) \tag{3.17}$$

For a sum of squares error, we have:

$$\frac{\partial E^n}{\partial y_j} = y_j - t_j \tag{3.18}$$

If the neuron considered is a hidden neuron:

$$\delta_j = \frac{\partial E^n}{\partial a_j} \tag{3.19}$$

$$\delta_j = \sum_k \frac{\partial E^n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{3.20}$$

where k represents the index of the units to which j sends a connection. It can be noticed that this latest equation is true because a variation in a_j only implies variation for the a_k. We then obtain the *back-propagation formula*:

$$\delta_j = g'(a_j) \sum_k w_{kj} \delta_k \tag{3.21}$$

The reason of the name of this algorithm is now obvious: we can compute the δ_j of a neuron only if we have computed the ones of the following layer (see Figure 3.6)! To be clear, this algorithm used for evaluating the derivatives of E^n with respect to the weights can be summarized in 4 steps:

1. Apply the input x and compute the activations of all the hidden and the outputs units.

2. Evaluate the δ's of the outputs.

3. Back-propagate the δ.

4. Compute the Error derivatives using Equation 3.14.

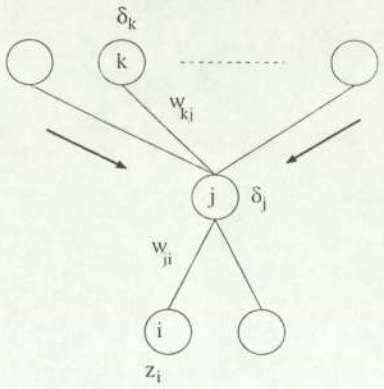


Figure 3.6: Illustration of the Back-propagation formula

Once this part is done, we only are half-way to the end of the training iteration: once the derivatives of the error have been computed, it remains to use them to adjust the values of the weights.

A lot of different techniques can be used for this computation. As far I am concerned, I have used the one known as "scaled conjugate gradients". This one is a

development of the "conjugate gradients" method. None of these algorithms will be detailed here but if you want further information, you can consult [2].

As you have seen in Section 3.3.1, we use a data set in order to control if there is overfitting or not. This is called the *early stopping* method. It simply consists in keeping a data set, the validation set, and computing the error on this set after a part of the training has been performed. If this error is lower than the one obtained at the previous step, we carry on, otherwise we stop the training. Thanks to this method we avoid overfitting in a simple and low-cost manner.

MLP with weight decay

To complete the description of this kind of neural network, I have to speak about the MLP with weight decay. This is an extension of the MLP where we try to control the smoothness of the response of the network to improve its generalization. This can be done if we write the error of the network as the sum of 2 terms: the first one controls the training set error while the second controls the smoothness of the response. If this second term is not introduced, we can obtain a response of the type of the one shown on Figure 3.4: the network has large weights, high curvature and so poor generalization.

$$S(w) = \frac{1}{2}\sum_{n=1}^{N}(y(x^n; w) - t^n)^2 + \frac{\alpha}{2}\sum_{i=1}^{W} w_i^2 \tag{3.22}$$

with W total number of weight in the network. The parameter α is called hyperparameters and have been kept constant once chosen. It controls the importance given to the regularity of the function we will obtain. These parameters can be re-evaluated during the training process to give a better response in a framework which had been called the *evidence* framework. You will find further details of this framework in the Section 2.4 or in [9].

In this study, every time we have worked with this kind of MLP, we have tried different values for the parameters and select the ones which give the lowest final error on the validation set.

3.3.3 Generalized Linear Model

Because this is a regression problem and assuming we work with target represented in 1-dimension, we want to approximate a function h such that for all $(x^n, t^n)_n, n \in \{1, \ldots, N\}$:

$$h(x^n) = t^n \tag{3.23}$$

with

$$h : \mathbb{R}^n \to \mathbb{R} \tag{3.24}$$

This function h can be modeled as the following sum:

$$h(x) = g\left(\sum_{j=1}^{d_{in}} w_j x_j\right) \tag{3.25}$$

where g is the linear activation function. This definition can be extended easily to the case where the dimension of the output is greater than one. This model is the Generalized Linear Model. A graphical representation can be found in Figure 3.7. The interest of this model is to see if there was an improvement by using non-linear model such as the MLP or RBF. The activation function for the output is defined in the same way than for the MLP. The training process used here is an algorithm called the Iterative Reweighted Least-Square (IRLS) which will not be described here but which can be found in [12].

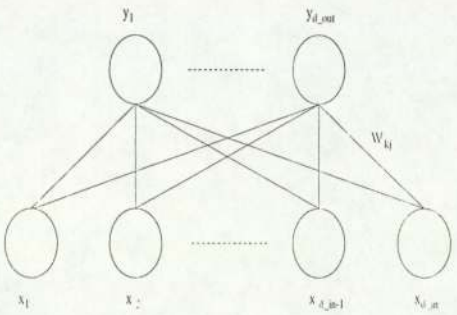


Figure 3.7: Graphical representation of our GLM

3.3.4 Radial Basis Function

At this point, we introduce a new kind of neural network where the response of a neuron is a function of the distance between the input vector and the output vector. One way to approximate the function h in Equation 3.23 is to write it as a weighted sum of basis functions:

$$h(x) = \sum_{j=1}^{M} w_{kj}\Phi_j(x) \tag{3.26}$$

where the Φ_j are defined as following:

$$\Phi_j(x) = \exp\left\{-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\right\} \tag{3.27}$$

and

$$\Sigma_j = \mathrm{cov}(\Phi_j) \tag{3.28}$$

The parameters μ_j and Σ_j are the center and the width (or variance) of the basis functions. They are different for each of them. Here, we can observe that it is the distance between the input vector and the center of the basis function which determines the amplitude of the response. You can see the corresponding graphical representation in Figure 3.8.

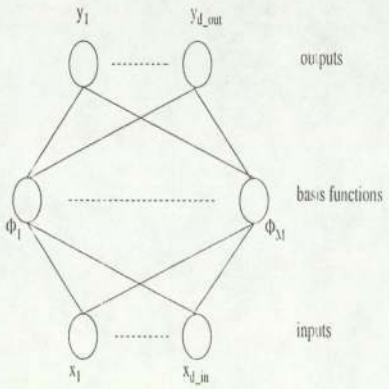


Figure 3.8: A graphical representation of Radial Basis Functions

Moreover, it is important to ensure that the number M of basis functions is lower than N, the number of patterns. It is a parameter of the network and it plays the

same role as the number of hidden nodes for a MLP: it controls the complexity of the response of the network.

The training process for the Radial Basis Functions is composed of two steps. First the centers μ_j and the width Σ_j are computed thanks to an iterative algorithm which is called the Expectation-Maximization (EM) algorithm. This part of the training is an *unsupervised* technique because we do not use the target values. During the second step the parameters of the basis functions are kept constant while we compute the values of the weights. Assuming that the error function for the regression problem is:

$$E = \frac{1}{2}\sum_n \sum_k (y_k(x^n) - t_k^n)^2 \tag{3.29}$$

then if Equation 3.26 is written in the form

$$\Phi W^T = T \tag{3.30}$$

we can find the weights which minimize this error with the equations:

$$\Phi^T \Phi W^T = \Phi^T T \tag{3.31}$$

$$\text{so } W^T = \Phi^\dagger T \tag{3.32}$$

with

$$\Phi^\dagger = (\Phi^T \Phi)^{-1} \Phi^T \tag{3.33}$$

and is called pseudo-inverse of the matrix Φ. The problem to find the optimum weights is then a linear problem and can be solved easily.

3.4 The results

To assess the results in a simple and quick way, the value of the BILAG score predicted by the network has been plotted against its true value. With a correct behavior for the network we should observe all the points near the line $y = x$.

Regarding the experiments, we have used the 4 data sets described above and trained a lot of different networks to see if they could learn what the value of the

BILAG index of the next visit will be. For each network where it was possible, we have used 4, 8, 16, 32 and 64 hidden neurons to see if there was some improvements in the results.

Concerning the MLP with weight decay, we have used the values $10^{-6}, 10^{-3}, 1, 10^{3}$ and 10^{6} for α. The network choosen as the best network is the one with the lowest validation error, and the results are plotted on the test set.

The plots in Figure 3.9, 3.10 and 3.11 show the results obtained when we tried to approximate the BILAG score at the next visit (*pred* set) using MLP with 16 hidden neurons, GLM and RBF with 16 hidden neurons. As you observe, the networks are far from giving us the good scores (points near the line $y = x$). We observe 4 series of points parallel to the axis (Oy). This corresponds to the 4 different values a BILAG score can take (0, 1, 3 and 9) taking account of the fact that these values have been normalized. For these scores, the answers of the networks seem to be at random!

Thus it appears that a new way to model the problem has to be found. At this moment we, logically, decided to look at a classification problem. Because the outputs of the networks we are using are categorized, why not use these categories to work on a classification problem?

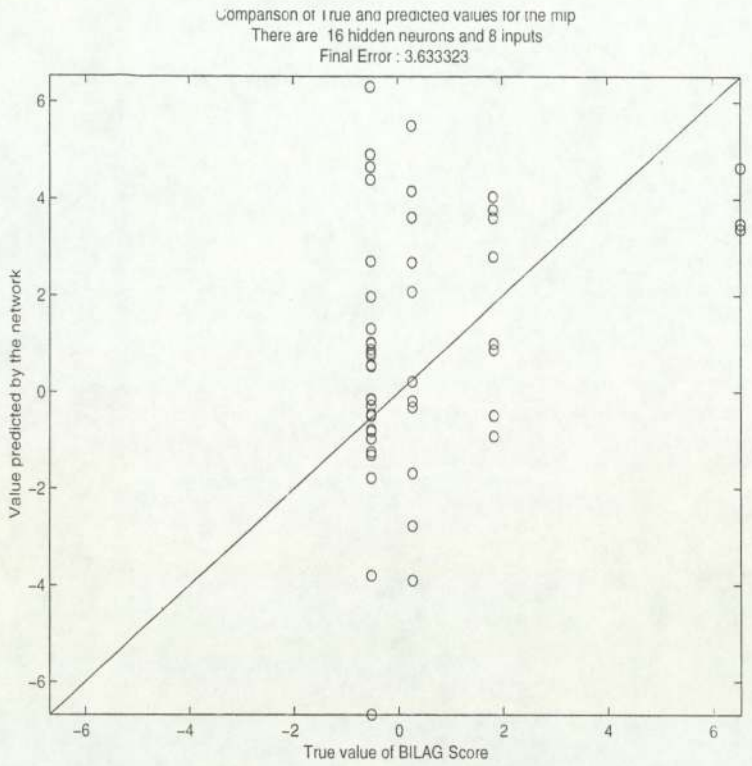


Figure 3.9: Result obtained for the Renal score with the MLP with 16 hidden neurons

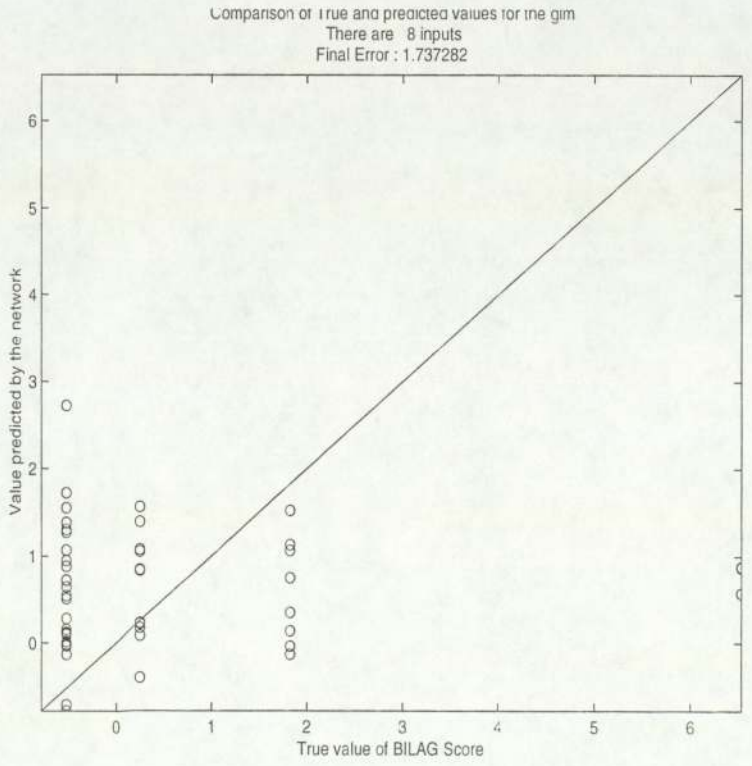


Figure 3.10: Result obtained for the Renal score with the GLM

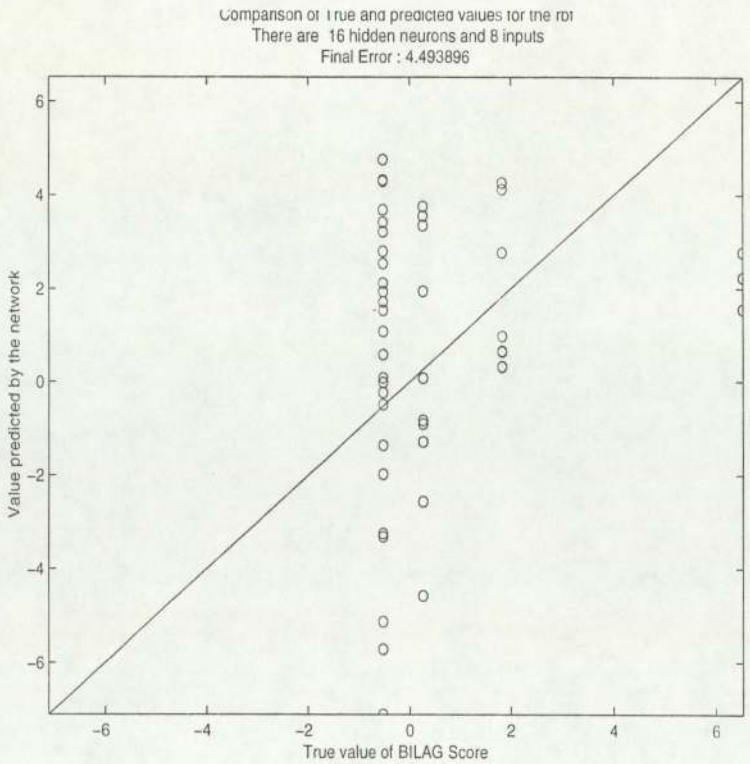


Figure 3.11: Result obtained for the Renal score with the RBF with 16 hidden neurons

Chapter 4

Classification problem

4.1 Introduction

The results of the regression problem being very poor, we worked on a different representation of the problem. Roughly, using the neural networks described in the Section 3.3, we decided to work on a classification problem. The goal was to classify the visits in two subgroups: those for which the next visit will be normal and those for which there is a flare of the disease at the next visit.

4.2 Modeling the problem

Remembering that the goal was to predict the flares up of the disease, we tried the classification problem. The BILAG scores can take 4 different values: 0, 1, 3 or 9. These 4 scores could have been used to compose 4 classes but the results would not have been so accurate: a good way to model the problem is to consider only 2 categories, the visits with a flare up at the next appointment with the doctor and the others. This way the model seems more reliable because while 2 different doctors could give a 0 and a 1 for the same observations, it less likely that their judgement differs when a 3 or a 9 is scored. So the score 3 appears naturally as a good limit above which we can consider there is flare up of the disease.

Our first approach had been to consider that a score of 3 for an individual score

and of 10 for the total BILAG score could be considered as flares. After a while, on the advice of Dr GORDON, we decided that having 3 systems scores greater than 3 should be a better way to represent a flare of the total BILAG score. Once more, considering that it is the doctors who give these scores, by doing this it was less likely that the subjectivity of the judgement would affect our results. At last, the goal of this classification problem was, given the clinical results at the visit v_t, to predict the BILAG score at the visit v_{t+1}. So we have only considered the *pred* set. During all this part, only one kind of problem has been treated: considering one, and only one, BILAG score, we try to separate the visits into 2 subgroups.

4.3 Implementation

4.3.1 Further preprocessing

Because in the original data set, there are only a few points with a high score, we had to balance the training set. This just means that we had to select our targets such that there were as many points with a high score as with a low score. As you can see on the Table 4.1, the percentage of visits with a high value for the BILAG scores varies between 2% and 19%.

	General	Muco	Neuro	Muscu	Cardio	Vascu	Renal	Haema	Total
Fraction	0.05	0.13	0.02	0.17	0.03	0.04	0.11	0.19	0.09

Table 4.1: Fraction of visits with flare of the disease

So, to make the Neural Network have an answer different to "all the visits are normal", we gave it in inputs as many "normal visits" as "visits with flare of the disease". This had been done knowing that after we would have to counter-balance the probability[1] obtained. To compute the right posterior probability we use Bayes'

[1] As explained in Section 4.3.2, the outputs of a neural network in a classification problem are probabilities

theorem:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} \tag{4.1}$$

where $p(C_k)$ is the prior probability and $p(x)$ the normalization term. Then you have, if $p(C_k^{train})$ and $p(C_k^{test}|x)$ are respectively the prior and the posterior probabilities after having balanced the data set,

$$p(C_k^{real}|x) = \frac{p(C_k^{test}|x)p(C_k^{real})}{p(C_k^{train})p(x)} \tag{4.2}$$

The prior $p(C_k^{real})$ is just obtained by counting the number of flares in the original data set. By applying this operation to the outputs of the network, we re-scale them to estimate the true posterior probability. Let's see how it works on a simple numerical example:

If we have 2 classes with the ratios 1/10 and 9/10 respectively for the classes C_1 and C_2. We assume than our network gives us the probability $p(C_1|x)$. As described above, we take as many points from C_1 as from C_2 for the training. Then, $p(C_1^{train}) = 1/2 = p(C_2^{train})$. Now, assuming that for an input x^n we obtain $p(C_1^{test}|x^n) = 0.8$. We then have to correct this probability using the formula 4.2. We begin with computing the quantities:

$$p = \frac{p(C_1^{test}|x^n)p(C_1^{real})}{p(C_1^{train})} \tag{4.3}$$

$$p = \frac{0.8 * 0.1}{0.5} \tag{4.4}$$

$$p = 0.16 \tag{4.5}$$

and

$$q = \frac{p(C_2^{test}|x^n)p(C_2^{real})}{p(C_2^{train})} \tag{4.6}$$

$$q = \frac{(1 - 0.8) * 0.9}{0.5} \tag{4.7}$$

$$q = 0.36 \tag{4.8}$$

then, to keep the property that $p(C_1^{test}|x^n) + p(C_2^{test}|x^n) = 1$, we can compute $p(C_1|x^n)$

by:

$$p(C_1|x^n) = \frac{p}{p+q} \tag{4.9}$$

$$p(C_1|x^n) = \frac{0.16}{0.16+0.36} \tag{4.10}$$

$$p(C_1|x^n) = 0.3 \tag{4.11}$$

So, the probability $p(C_1|x^n)$ has been rescaled from 0.8 to 0.3. This operation is a correction which makes some points move from class C_1 to C_2 because we have not taken account to the original distribution of the points.

Moreover, we remove some points: if the lag between the date when the tests had been undergone and the date the next score has been assessed is greater than 6 months, we remove the point. We can reasonably assume that the prediction will not be very accurate in this case.

4.3.2 The Neural networks

While in the regression problem, the target variables were the values of the BILAG score, here it is the probabilities of class membership. Indeed, the outputs represent the posterior probability of class membership $p(C_k|x)$ where C_k is the kth class and x the input vector.

In a classification problem with 2 classes, there exist 2 ways for dealing with the data. On the first hand, you can consider a single output for the neuron which represents the probability of belonging to the class C_1. On the other hand, the dimension of the output is 2, it is a 1-of-n matrix: each dimension of the output corresponds to the probability of the point belonging to the class C_i.

One output

Here we consider a neural network where the output corresponds to the posterior probability $p(C1|x^n)$. It just means that it represents the probability that the point presented to the network belong to the class C_1. The posterior probability of the class C_2 is then given by the quantity $p(C_2|x^n) = 1 - p(C_1|x^n) = 1 - y^n$. If the target t^n is

1 when the vector belongs to class C_1 and 0 otherwise, we can write the probability of observing the target value as:

$$p(t^n|x^n) = (y^n)^{t^n}(1-y^n)^{1-t^n} \tag{4.12}$$

The the joint probability density of the whole data set is:

$$p(t^1,\ldots,t^n|x^1,\ldots,x^n) = \prod_n p(t^n|x^n) \tag{4.13}$$

$$p(t^1,\ldots,t^n|x^1,\ldots,x^n) = \prod_n (y^n)^{t^n}(1-y^n)^{1-t^n} \tag{4.14}$$

To obtain the best performances for our network, we want to maximize this function which is called the *likelihood*. By taking the negative logarithm of this function, this is equivalent to minimizing the following function:

$$E = -\sum_n (t^n \ln y^n + (1-t^n)\ln(1-y^n)) \tag{4.15}$$

This is called the *cross-entropy* function and is the error chosen in the classification problem when the dimension of the output is 1.

The activation function for the outputs is the *logistic* function. This has been chosen because it can be demonstrated under the assumption that the class-conditional densities ($p(C_i|x)$ can be approximated by a normal distribution that the output has the property of being a probability function. Its definition is:

$$y = \frac{1}{1-e^{-a}} \tag{4.16}$$

Multiple outputs

In this case, we consider the 1-of-n matrix: each output represents the probability to belong to a class given the observation (class-conditional density). If the points are drawn independently, we then have:

$$p(t^n|x^n) = \prod_{k=1}^{c} (y_k^n)^{t_k^n} \tag{4.17}$$

In the same manner as before, we obtain the likelihood and the error functions:

$$p(t^1, \ldots, t^n | x^1, \ldots, x^n) = \prod_n \prod_{k=1}^{c} (y_k^n)^{t_k^n} \tag{4.18}$$

$$E = -\sum_{k=1}^{c} t_k^n \ln y_k^n \tag{4.19}$$

$$\tag{4.20}$$

As far as the activation function is concerned, we just use a generalization of the one used in one dimension. This one is called the *softmax* and is defined as following:

$$y_k = \frac{e^{a_k}}{\sum_j e^{a_j}} \tag{4.21}$$

4.4 Results

4.4.1 Additional background

Now, because it was only a 2 class problem, and because the results were rather the same in both cases, we only show the result in the case of one output. This output is then the probability $p(C_1|x)$. The value $p(C_2|x)$ is obtained by computing $1 - p(C_1|x)$.

Moreover, we have introduced a quantity which represents the doubt in our decision: we have used a *threshold of rejection*. This threshold is the limit for the probability $p(C_1|x)$ or $p(C_2|x)$ below which we will not classify the point because we are not sure enough of its class. In this case, the decision is let to the expert point of view that is to say the doctor.

Finally, we should have taken account of the *risk* of misclassification. Indeed, until now, we have made no distinction in mis-classifying an individual in the group of those who will have a flare up and the opposite. Because these 2 diagnoses do not have the same consequences in a human and a financial point of view, a *cost matrix* can be introduced. This cost matrix can only be defined by the experts and represents the *penalties* associated to the different misclassifications. If this matrix is R, we have R_{ij} which represents the cost of misclassifying a pattern of class C_i in class C_j. It comes into play when a decision is made. This information is very difficult to quantify and is left for future work.

4.4.2 Presentation of the results

Finnaly, to have an overview of the results, 4 graphs are plotted for each BILAG score. These graphs give a quick idea of the performances of the networks. On each of them, you will find 4 curves which correspond to the 4 kinds of networks used: MLP, RBF, GLM and MLP with weight decay (square points). Moreover, these networks have been trained with different numbers of hidden neurons (when it was possible, i.e. not for GLM). So, for each graph, the horizontal axis represents the number of hidden neurons of the network. The numbers tried are 4, 8, 16, 32 and 64. Concerning the variables selected as inputs, they are the ones adviced by Dr GORDON: F103E, F111,F112,F113,F119C,F87,F121,F141.

Now let see what these graphs represent:

- First, on the right bottom, you can find the percentage of rejected values. Because of the threshold of rejection, not all the points are classified.

- Secondly, on the left bottom, the false positive rate is presented. This rate is the percentage of people classified as "will have a flare up" while they actually will not. This quantity is quite interesting since for example, we can be able to detect all the flares but if at the time there is a false positive rate of 80%, the results are useless. It is equivalent to saying that everybody will have a flare up.

- Now, the most interesting graphs. On the top right side, is shown the percentage of correct classification for the flares considering only the classified visits (true positive rate).

- Last, on the top left side, you will find the percentage of detection of flares considering all the visits. This had to be plotted because of the threshold of rejection. We can have a percentage of 100% on the last graph because only a few flares had a probability high enough to be classified, while a lot of points were rejected. Then, with this graph, we will be able to detect this kind of situation

So, to summarize, a way to read the results is the following: you first look at the bottom right graph to see if a lot of points are classified or not. Then, among the

classified points, you can look if the false positive rate is high. Last, you look at the percentages of detection in the top graphs.

Figure 4.1 and Figure 4.2 show the results obtained for the General and the Renal scores. The others can be found in the Appendix D. You will observe that 2 scores are missing: the cardiorespiratory and the neurological scores. This is due to the way we build our training and test sets. As a matter of fact, we take at random 50 individuals. The data collected on these 50 individuals will not be included in the training set, but kept for the test set. Doing this, this happens sometimes that only a few flares actually are in the test set. When it happens, the results can not be interpreted since the percentages of detection are based on too few values: 5 for these two scores. The Table 4.2 presents a summary of the results: it shows (in order) the true positive rate, the true positive rate regarding all the points, the false positive rate and lastly the percentage of points which are rejected. The figures are the ones obtained with the best model for each system.

	TP	TP total	FP	Rejected
General	0.86	0.60	0.33	0.28
Renal	0.73	0.73	0.39	0.00
Haema	0.82	0.73	0.37	0.20
Muco	0.44	0.28	0.43	0.46
Muscu	0.62	0.35	0.54	0.36
Vascu	0.54	0.43	0.45	0.16

Table 4.2: Table of the results of the best model for each system

4.4.3 Comments

Role played by the threshold

First, we have to speak here about the role played by the threshold. For all the results, we have made this threshold take the values 0.5, 0.6 up to 1 by adding 0.1 at each time. Then the percentage of rejected points varies between 0 and 1. The threshold influences it in the following manner. When the threshold is 0.5, none of the points are rejected because we have either $P(C_1|x) > 0.5$ or $p(C_2|x) = 1-p(C_1|x) > 0.5$.

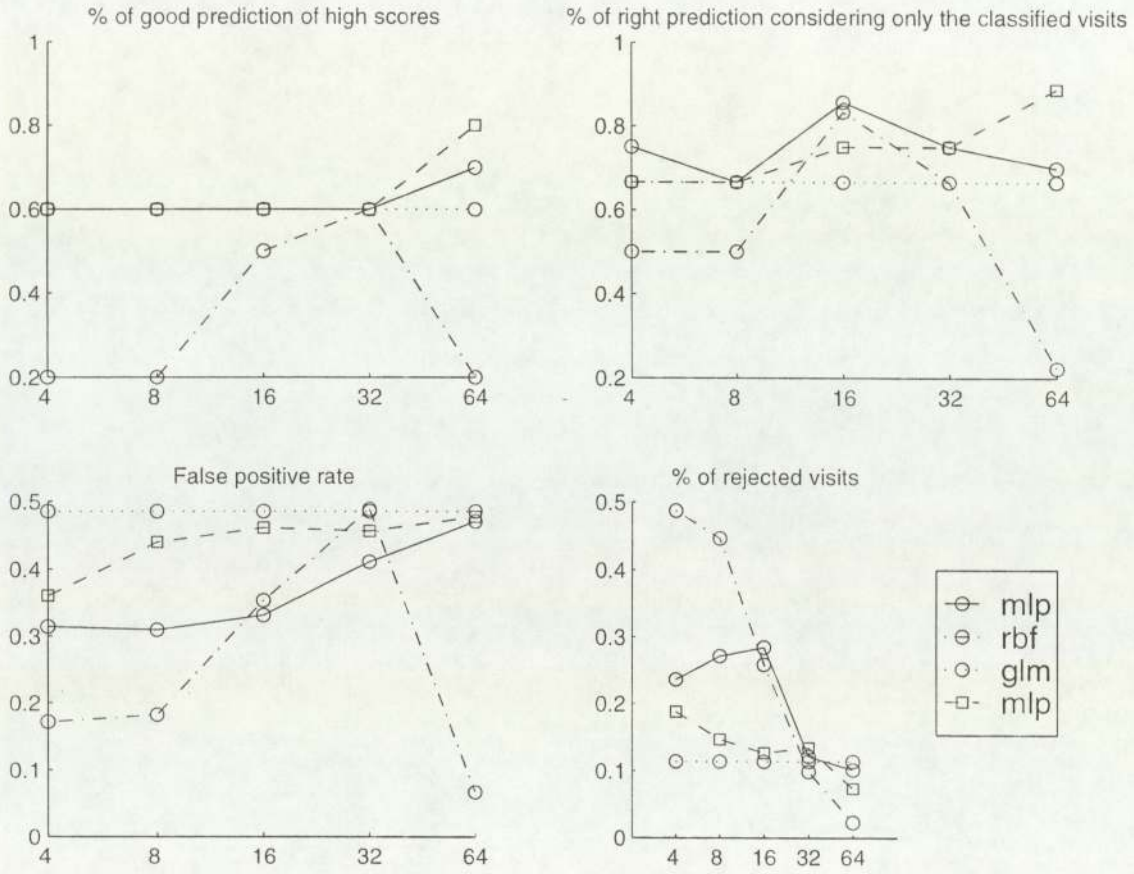


Figure 4.1: General score. There are 156 and 394 visits for the train and the test set among which 78 and 10 are high scores. The best model is MLP with 16 neurons

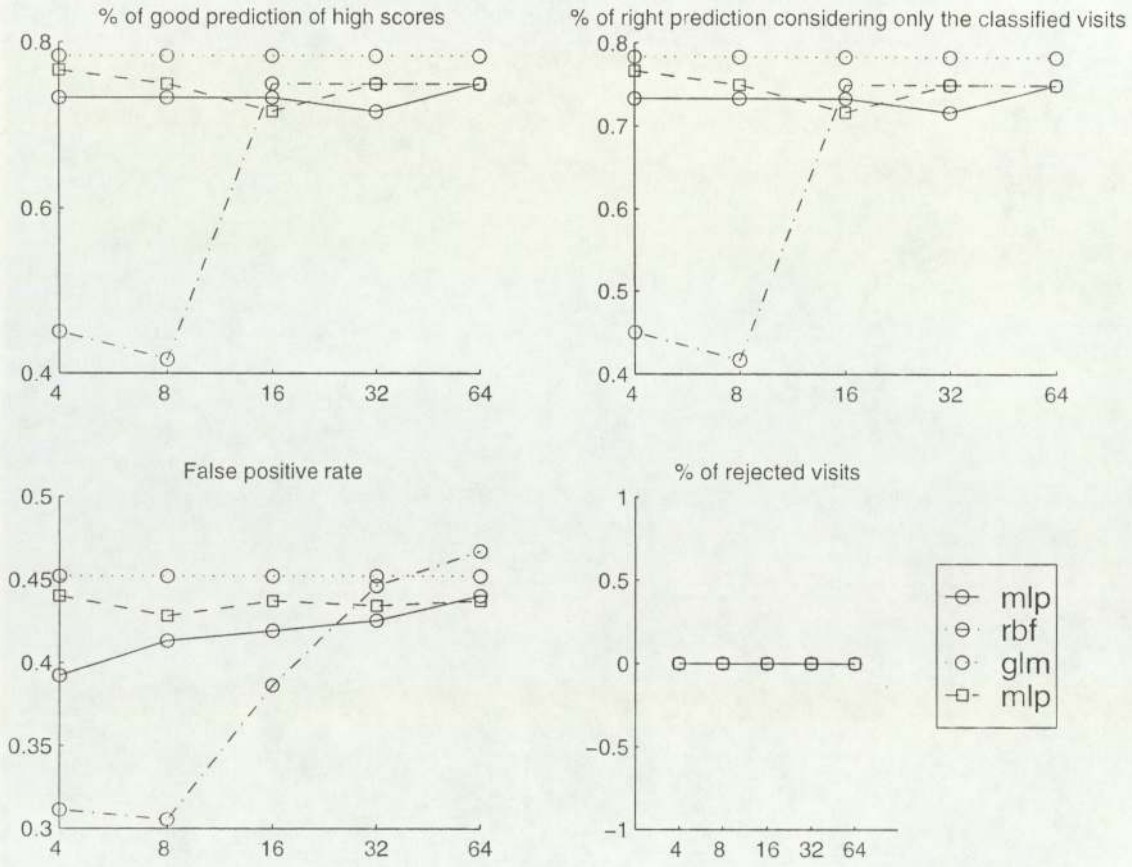


Figure 4.2: Renal score. There are 638 and 394 visits for the train and the test set among which 319 and 60 are high scores. The best model is MLP with 4 neurons (lower false positive rate for nearly same percentage of detection than GLM)

Now if the threshold τ lies in the interval $(0.5, 1)$, we have to consider 3 cases (they are exclusive):

1. $p(C_1|x) > \tau$ then the visit belongs to the class of the people who will have a flare.
2. $p(C_2|x) > \tau$ then we do not predict any flare.
3. $p(C_1|x) < \tau$ and $p(C_2|x) < \tau$ then we do not take any decision. The point is left to the expert judgement.

Moreover, this threshold influences in a opposite way the false positive and the false negative rates. To illustrate it, let's take a simple example. Let imagine that we have 2 kinds of points p1 and p2 which are between 0 and 10. If we have the following partition (cumulative percentage):

	[0;2.5[	[2.5;5[	[5;7.5[	[7.5;10[
p1	10	15	35	100
p2	80	85	90	100

Table 4.3: Cumulative percentage of points

Then if the threshold is $\tau_1 = 2.5$, 10% of the p1 points will be classified in p2 (false negative rate) while 20% of the p2 points will be classified as p1 points (false positive rate). Now if $\tau_2 = 7.5$ we will a false negative rate of 35% and a false positive rate of 10%. So what it gains in one statistic, is lost in the other. Our threshold acts in exactly the same way.

So, by making this threshold vary, we obtain different performances. For each score we have tried to find a compromise between obtaining the lowest false positive rate and rejected percentage on one hand, and, on the other hand, the best percentage of detection. The results presented are the ones we assumed to be the best; the scores do not have the same threshold.

The graphs

Now let us speak about the results themselves: we can say that they are rather interesting. Roughly, we can detect between 60% and 80% of the classified visits which do have a flare. On the other side, we usually have a false positive rate which is between 30% and 40%. This means that 1 patient over 3 will be declared to be at the beginning of a flare of the disease while it is not true! This will imply some further tests on these patients although they do not need them. You then have to define some priority to give a help in the decision making. A way to do so is to fill in the loss matrix and it would be very interesting to look at this in the future.

The time dimension

Last but not least, there is one more thing to consider in this problem: the time dimension. As a matter of fact, we do not take account of the time in our prediction problem. The inputs are the values of some variables taken at time t, the outputs are a BILAG score taken a time $t+1$ but we do not know how many days, weeks or months have elapsed between these two visits. To have something more coherent, we have removed the points where the time spent between the date when the tests have been done and the date of the score was greater than 6 months. In order to see the influence of time on the performances of the network, we have plotted some graphs. These graphs show the true positive rate as a function of the time. At each point is associated a number in brackets which is the false positive rate. These curves have been plotted for three different systems, Haema., Renal and Vascu., with two kinds of neural networks which are the MLP and the RBF. The number of hidden neurons selected is 16 for both of them.

In Figure 4.3, the first thing we can notice is that for the visits which have a time lag between 3 and 4 months there is a big fall in the performances of the network. This trend is also true for the other curves you can find in Appendix E. Moreover, the performances are slightly growing with the time between 0 and 3 months but this is not true for the other curves.

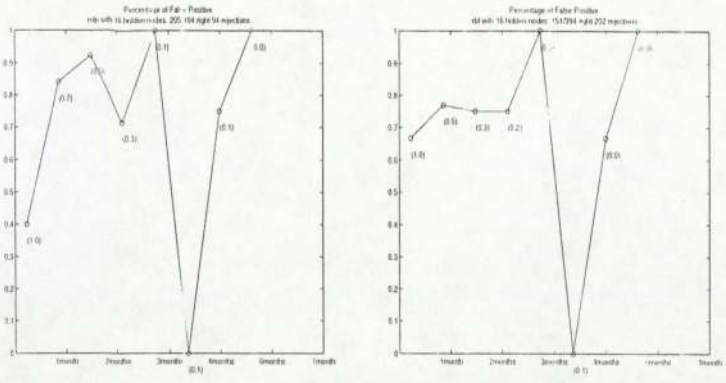


Figure 4.3: Results for the Haema score as a function of the time

Another common point beween all these curves to notice is that the false positive rate goes down with the time when the lag is between 0 and 3 months. This just means that if the visits are too close we have more chances to declare a patient without any problem as a patient who is likely to have complications. It is in agreement with the usual interval of time adviced by the doctors between the visits and confirms it as a good lag. It would have been then a good idea to apply our classification problem using only the points where the lag is 3 months. Unfortunately, we could not do everything we wanted in the time allocated to this thesis!

Conclusion

As you have seen throughout this thesis, we did not immediately obtain some pertinent results. Some issues arose because of the data themselves with the important number of variables or missing values by instance, while, at the same time, we did not immediately find a good way to model the problem.

The results obtained for regression problem and the selection of variables with linear dependencies were not very satisfying. This could probably be explained by the nature and the complexity of the problem itself: we tried to model the evolution of a human body. Furthermore these poor results in looking for some linear dependencies may be a reason of the difficulties to obtain systematic results on Lupus in the previous studies.

Nevertheless, it has finally been possible to obtain a good percentage of detection of the future flares of the disease. Considering only the classified visits, we can predict between 60 and 90 percent of future high scores for six over eight of the BILAG systems and between 90 and 100 percents of the total number of increases of the total BILAG score! This last being obtained thanks to the conjunction of two non-linear process: ARD and Neural Networks.

To conclude, I will say that the classication problem seemed to be a good approach for our prediction problem and could be the start of some further work. The most interesting thing would be to look at the ARD. It can be thought that it is a key process in order to determine the relevance of the variables, and so of the blood tests, in terms of prediction of a BILAG score. It pointed out F119C as a relevant variable in the determination of the total BILAG score whereas this variable was usually not thought as a good predictor of the disease. Finnaly, the introduction of a loss matrix

will help a lot in decision making for this problem.

Appendix A

Correlation matrices

These correlation matrices have been computed by taking account to all the variables except the ones with too many missing values (F103, F109, F110, F88). Moreover, after the preprocessing (we had to remove the visits where any value was missing), one variable (F108) had a null variance. Because of this we can not compute the correlation between this variable and the others so we have discarded it.

There are 4 result matrices, each of them corresponding to a data set described in section 3.2. The target variable (total BILAG score) is the last variable. The results are clearly not interesting, since the magnitude of the correlation is too small.

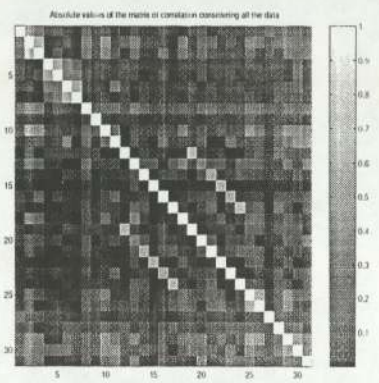


Figure A.1: Correlation matrix of all the data

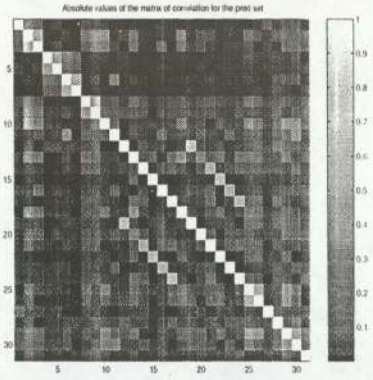


Figure A.2: Correlation matrix of all the data for the pred set

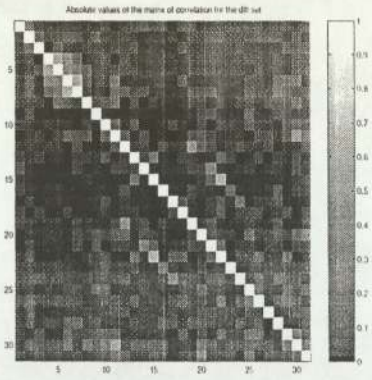


Figure A.3: Correlation matrix of all the data for the diff set

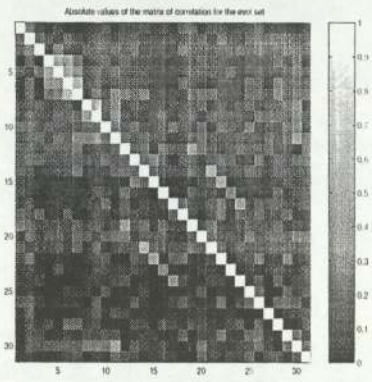
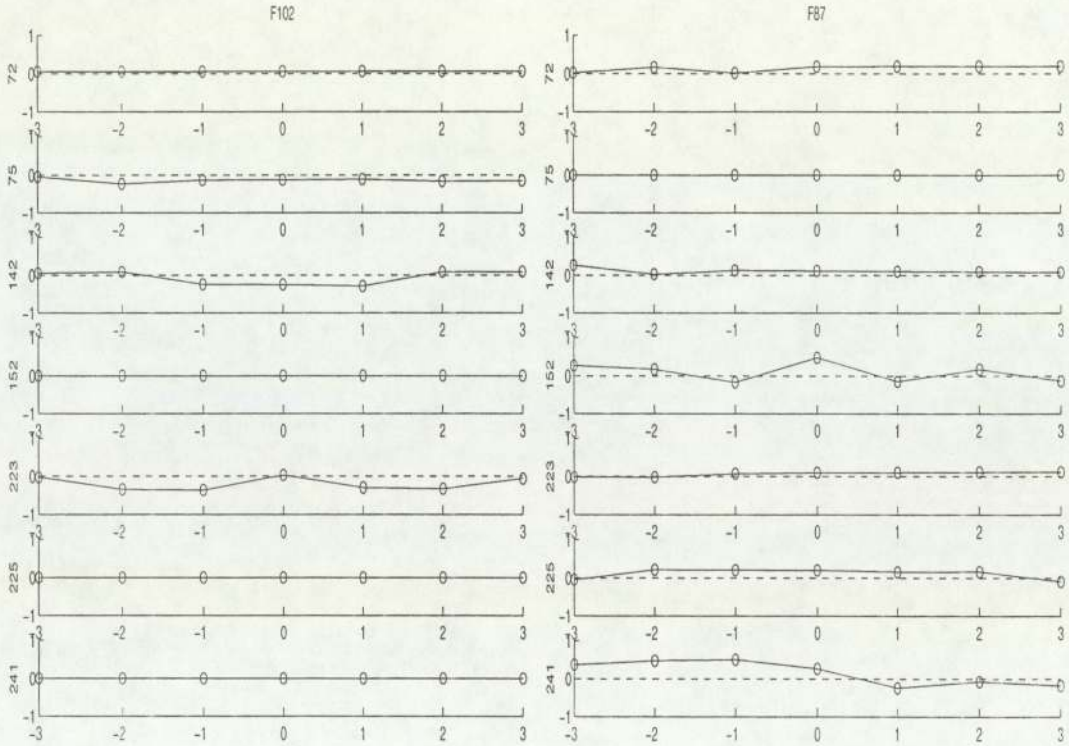
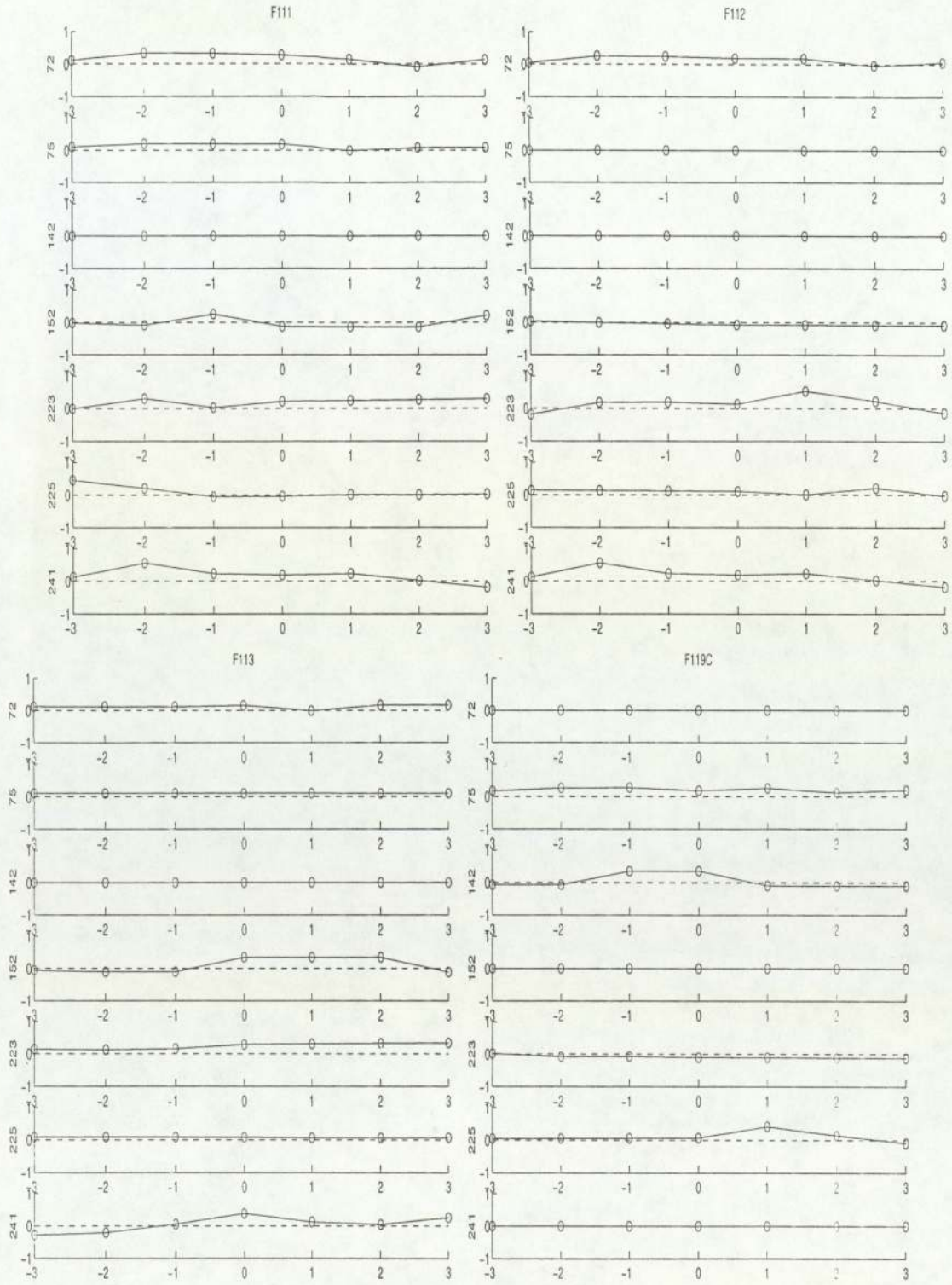


Figure A.4: Correlation matrix of all the data for the evol set

Appendix B

Cross-correlation




F111
F112
F113
F119C
72
75
142
152
223
225
241

Appendix C

Results of the PCA

F101	F102	F103E	F104	F105	F106	F107	F108	F111	F112	
1	2	3	4	5	6	7	8	9	10	
F113	F119C	F120	F121	F122	F123	F124	F125	F133	F140	F141
11	12	13	14	15	16	17	18	19	20	21
F142	F143	F144	F145	F71	F72	F75	F76	F90	F87	Points BILAG
22	23	24	25	26	27	28	29	30	31	32

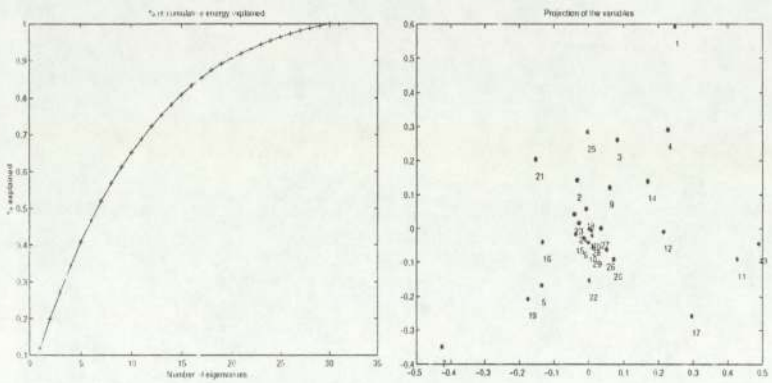


Figure C.1: **PCA with all variables.** In right figure, we see that 15 variables are needed to explain 80% of the variance. As soon as it is noticed, we can stop the interpretation since the number of variables is too much great! The left graph could show some correlations between the variables but they are noticeable only if their value is greater than 0.7. Because this not the case, we can not give extract additional information from this figure.

103E	F111	F112	F113	F119C	F87	F121	F141	Points BILAG
1	2	3	4	5	6	7	8	9

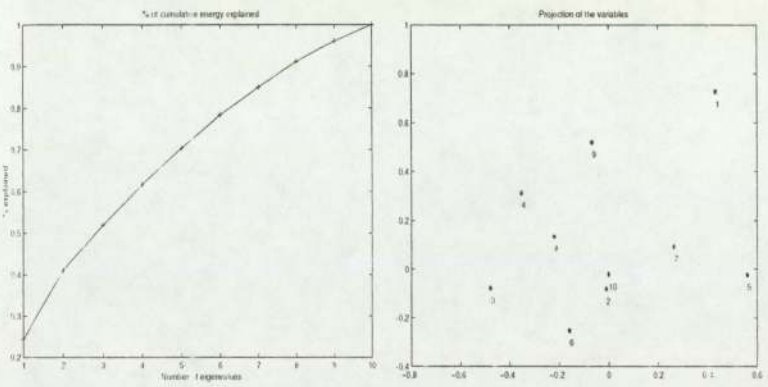


Figure C.2: **PCA with variables adviced by Dr GORDON**. For the same reasons than before and because there is no noticeable knee in the cumulative percentage of variance explained, the results of this PCA are not interesting

Appendix D

Results of the classification problem

You will find for every systems the model which I think is the best. The model chosen is the best because it has a good percentage a prediction (figures at the top) and low false positive rate and percentage of rejected value.

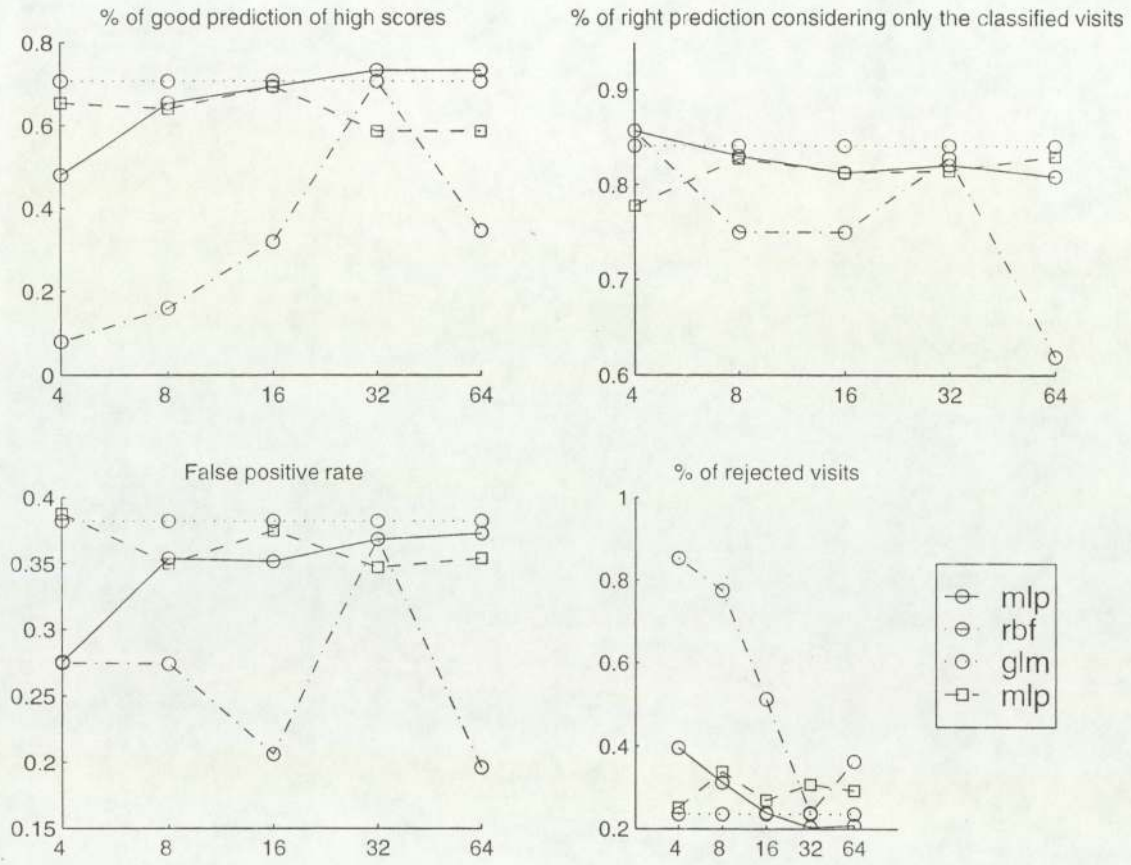


Figure D.1: Haema score. There are 858 and 394 visits for the train and the test set among which 429 and 75 are high scores. The best model is the MLP 32 hidden neurons.

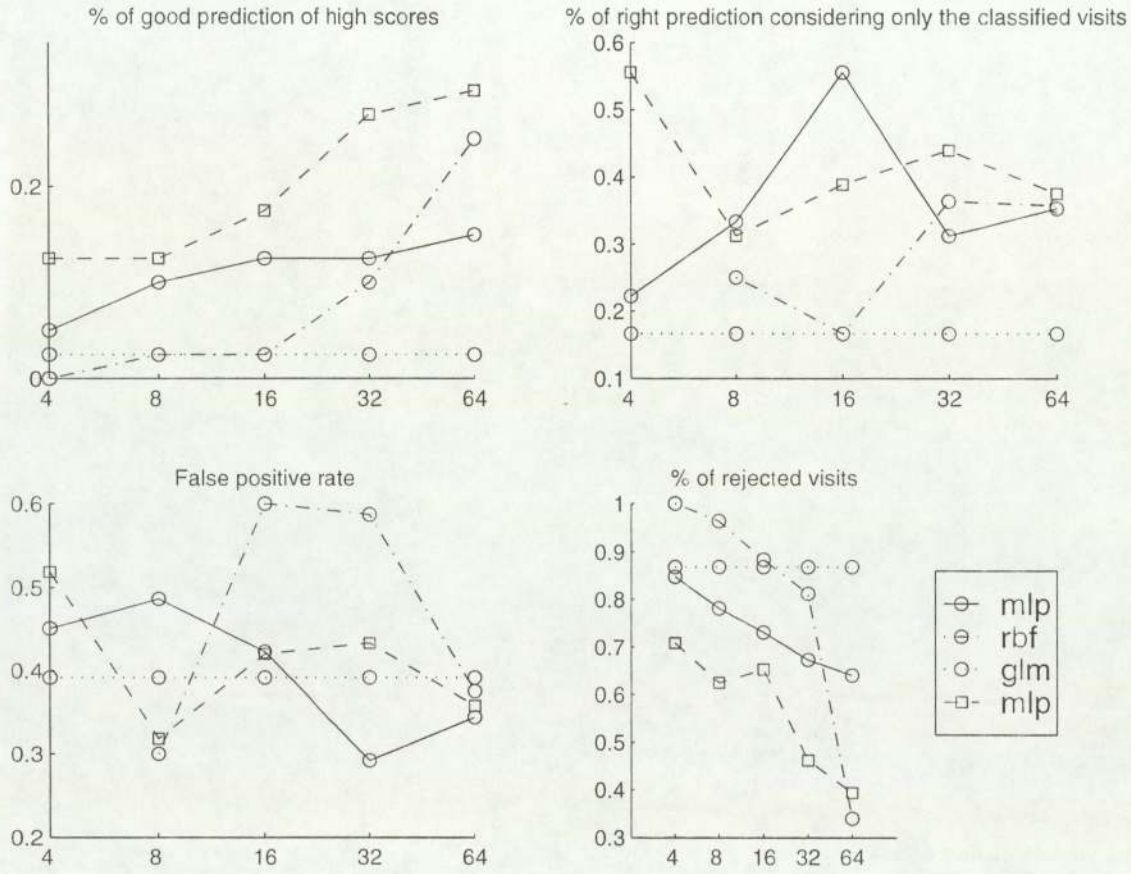


Figure D.2: Mucocutaneous score. There are 540 and 394 visits for the train and the test set among which 270 and 40 are high scores. The best is the MLP with weight decay with 32 hidden neurons.

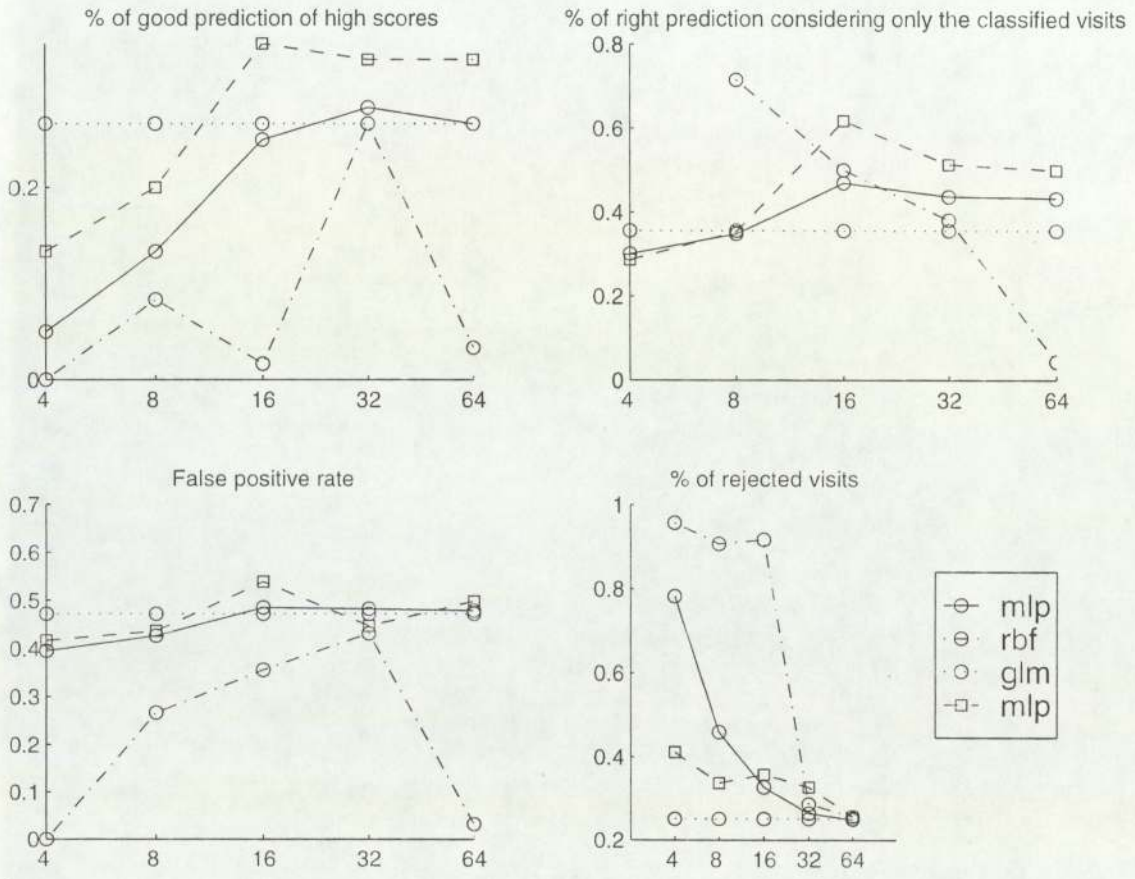


Figure D.3: Musculoskeletal score. There are 614 and 394 visits for the train and the test set among which 307 and 60 are high scores. The best model is the MLP with weight decay with 16 hidden neurons.

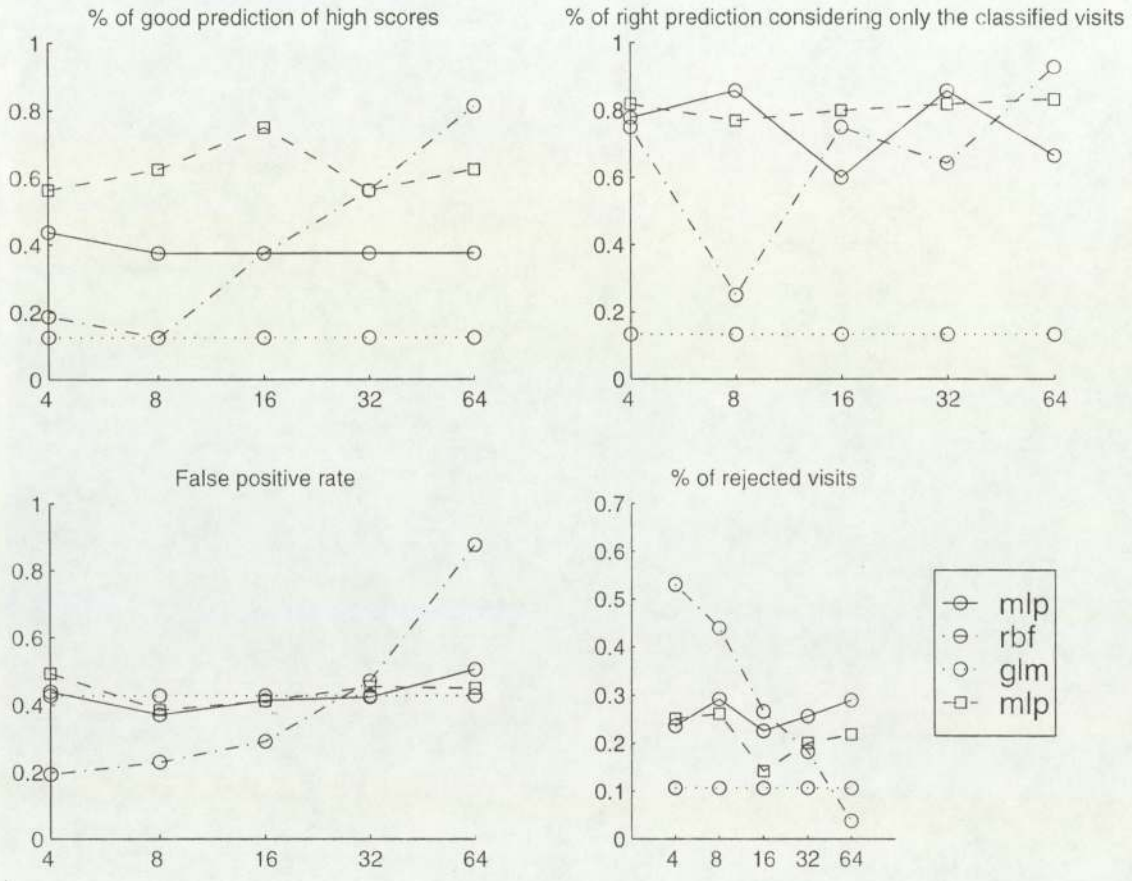


Figure D.4: Vasculitis score. There are 148 and 394 visits for the train and the test set among which 74 and 16 are high scores. The best model is the MLP with weight decay with 32 hidden neurons.

Appendix E

Study of the influence of time on classification results

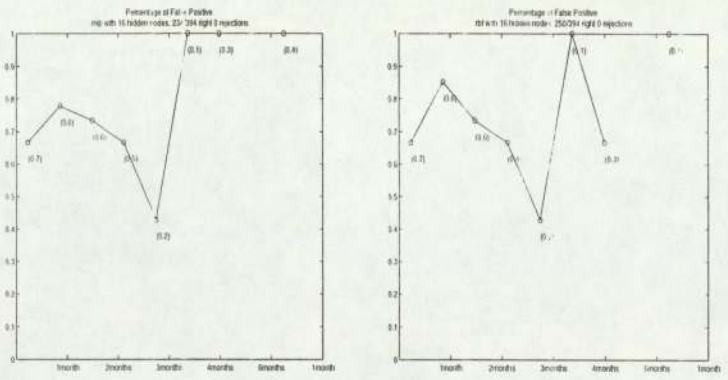


Figure E.1: Results for the Renal score in function of the time. None of the network reject any point. 238 and 250 points over 394 are classified in their good class for MLP and RBF.

First, the graphs in Figure E.1 and in Figure E.2 do not have the same number of points because some of the points could have been rejected. So, there is no point for some lags. Moreover, we can observe in these two figures a decrease of the performances when the lag is 3 months. Finally, the false positive rate decreases with the time when the lag is between 0 and 3 months.

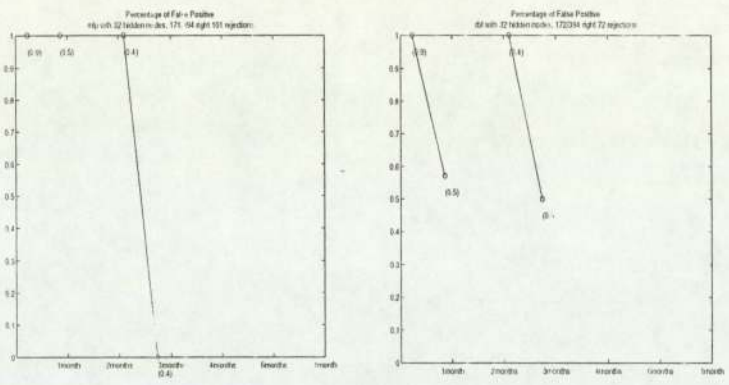


Figure E.2: Results for the Vasculitis score in function of the time. MLP rejects 101 and right classifies 171 over a total of 394 points. RBF rejects 72 points and makes 172 good prediction for the same number of points.

Bibliography

[1] W.G. Baxt. Use of an artificial neural network for data analysis in clinical decission-making: The diagnosis of acute coronary occlusion. *Neural Computation*, 2(480-489), 1990.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] C Chatfield. *The analysis of times series*, chapter 8. Chapmann and Hall Ltd, 1980.

[4] K. Gurney. *An introduction to neural networks*. UCL Press, 1997.

[5] E.M. Hay, P.A. Bacon, C. Gordon, D.A. Isenberg, P. Maddison, M.L. Snaith, D.P.M. Symmons, N. Vinner, and A. Zoma. The bilag index: a reliable and valid instrument for measuring clinical disease activity in systemic lupus erythematosus. *Quarterly Journal of Medicine*, 86:447–458, 1993.

[6] D. Lowe and A.R. Webb. Exploiting prior knowledge in network optimization: an illustration for medical prognosis. *Network: Computation in Neural Systems*, 1(3):299–323, 1990.

[7] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[8] D.J.C MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.

[9] D.J.C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[10] D.J.C. MacKay. *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, chapter Bayesian Non-Linear Modeling for the Prediction Competition. Dordezcht: Kluwer, 1995.

[11] S. Magnus and T.J. Sejnowski. A mixture model system for medical and machine diagnosis. *Advances in Neural Information Processing Systems*, 7, 1995.

[12] P. McCullagh and J.A. Nelder. *Generalized Linear Model*. Chapman & Hall, 2nd edition, 1989.

[13] M.D. Richard and R.P. Lippmann. Neural networks classifiers estimate bayesian a posteriori probabilities. *Neural Computations*, 3(4):461, 1991.