ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Convex Optimization and Applications : Logistic regression

William Attache, Lorinde Westerberg Knoops, Juliane Dervaux

June 21, 2018

Introduction

We have chosen to work on the topic of *Logistic regression* because we thought it to be an interesting problem that also gives visually interpretable solutions. The model of Logistic regression is a statistical one, that is often used to find the probability distribution of a binary variable, depending on a set of explanatory variables. It thus becomes an optimization problem to find the distribution of this variable that suits reality the best, thus optimizing the statistical estimation. These problems are rather easy to solve because they are convex.
Logistic regression is a mathematical optimization tool used in many fields. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by among others, using the logistic regression.

In this project, we have chosen to apply it on 2 specific examples:

- case 1 : estimate the probability to pass an exam with respect to the number of studying hours.
- case 2 : estimate whether a wine is a good wine or not based on physico-chemical features.

The Logistic Regression model

A logistic model is based on a vector $Y \in \mathbb{R}^m$ and on a matrix $X = (x_1, x_2, ..., x_m) \in \mathbb{R}^{n\times m}$, where m is the number of samples and n is the number of descriptors. Each example x_k is associated to an output $y_k \in \{0,1\}$: thus, every output y_k in Y depends on the values of the *descriptor variables* $(x_{k,i})_{i\in[\![1,n]\!]}$. Our goal is now more clear : estimate a probability p depending on the descriptor variables such that, given a new example x_{new}, we can compute p the probability for x_{new} to be associated with an output $y = 1$. Recognizing a *Bernouilli scheme*, x_{new} is associated to the output $y = 0$ with the probability $1 - p$. Note that that descriptor variables do not need to be independent from each other [1], [2]. Illustrations of possible descriptor variables will be given in the following examples.

So the model is set up as such that it converts a Bernoulli variable Y, into a continuous one that can take on any real value.In the following, we will estimate parameters $a \in \mathbb{R}^n$ and b such that, for all $k \in [\![1,m]\!]$:

$$p = \mathbf{prob}(y_k = 1) = \frac{exp(a^T x_k + b)}{1 + exp(a^T x_k + b)} \tag{1}$$

$$1 - p = \mathbf{prob}(y_k = 0) = \frac{1}{1 + exp(a^T x_k + b)} \tag{2}$$

$$\tag{3}$$

This illustrates the fact that the probability distribution of the binary variable Y depends on a linear combination of our descriptor variables.

Theoretical facts and optimization problem arising:

We define the odds as the probability that a specific outcome is a case divided by the probability that it is a noncase [1].

$$odds = \frac{p}{1-p} \tag{4}$$

If we take the logarithm of the above we obtain the logit (log odds) function [1] :

$$log\ odds = \log(\frac{p}{1-p}) \tag{5}$$

Replacing by the corresponding probability, one can verify that log-odds in this model indeed a linear combination of the explanatory variable, which is an important property of the model :

$$log\ odds = a^T x + b \tag{6}$$

By ordering the data set such that the q first (x_i, y_i) pairs are 1 outcomes, whereas the (m-q) remaining pairs are 0 outcomes, we can compute the likelihood function given by :

$$L = \prod_{i=1}^{q} p_i \prod_{i=q+1}^{m} (1-p_i) \tag{7}$$

$$\tag{8}$$

Taking the log of this function we get the log-likelihood function $l : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}$:

$$\begin{aligned} l(a,b) = log(L) &= \sum_{i=1}^{q} \log p_i + \sum_{i=q+1}^{m} \log(1-p_i) && (9) \\ &= \sum_{i=1}^{q} (a^T x_i + b) - \sum_{i=1}^{m} \log(1 + exp(a^T x_i + b)) && (10) \end{aligned}$$

The last line equation has been computed by inserting the value of the logistic function. Finally, this function is maximized with respect to the parameters a and b through a maximum likelihood estimation and a convex optimization problem arises [2]:

$$\max_{a,b \in \mathbb{R}^n \times \mathbb{R}} l(a,b)$$

, which is well convex because

- $\mathtt{C} = \mathbb{R}^n \times \mathbb{R}$ is convex.
- $l : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}$ is concave as the sum of an affine function and of the opposite of a sum-log-exp function, which is convex.

Note that using the logistic function does not classify the data (it only gives the probability of a certain output). In order to obtain a classifier, one should set up a threshold value of p$(y = 1)$ to be compared with the estimated probability, and be able to assign input data to one of the two classes.

Formulation and experimental setup

Datasets.
We would like to apply the model on two different examples. The logistic regression is first run on a smaller dataset, **Case 1**, that indicates for $m = 20$ students their time of study for an exam and their results at the exam, pass or fail. In this case, $n = 1$: the only descriptor variable is the `number of studying hours`. The data for this example can be found on the Logistic regression page of Wikipedia [1]. The input data looks as follows:

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

Table 1: First data set : exam result (0 if failed, 1 if passed)
with respect to the number of hours of study

As a second example, **Case 2** studies an example with more data. The data was found on kaggle [3] and contains information about $n = 11$ different physicochemical properties of $m = 1599$ red wine varieties, all produced in a specific area of Portugal [4]. These physiochemical properties are our descriptor variables. The initial output vector Y corresponds to the ranking of each wine, a grade between 1 and 10, given by tasters. To make this problem binomial we mapped the grades to 0 or 1, where 0 was assigned to wines with a grade inferior or equal to a given threshold, and 1 was assigned to a wine with grade greater or equal to that *threshold*. This thus means 0 signals "bad wine", 1 signals "good wine".

Experiments.
First of all we will solve the above optimization problems, thus estimating the regression coefficients and finding an accurate estimation of the probability distributions. Then, trough changing the output vector Y, we look at the different coefficients a and b and thus probability distributions that we obtain.

Matlab/cvx code

The code we use is adapted for CVX by Argyris Zymnis [6], from [2], section 7.1.1.

`Code explanation:` The code itself can be found at the end of the document. We quickly explain here the interesting part using *CVX*. We work in CVX region, delimited by **cvx_begin** and **cvx_end** [5]. First, one needs to notify CVX about the number of parameters in the problem to solve. In our case, this is a column vector of size $n + 1$. Then , the instruction *maximize* permits to solve our problem, before giving our parameters back.

```
cvx_expert true
cvx_begin
    variables x(x_size)
    maximize(y'*U*x-sum(log_sum_exp([zeros(1,m); x'*U'])))
cvx_end
a = x(2:x_size)
b = x(1)
optval = cvx_optval;
```

For the rest of the code, one can find the following files in the archives :

- *regression.m* : this file permits to run three simulations, the original one of the Zymnis' code, the case 1 and the case 2. First data are generated / extracted, then problem is solved and then results are gave back, and curves plot when it is possible.
- *testCase1.m* : different output vectors y are generated, and simulations are run over each one of them. Curves are ploted if the user wants. All results are stored in a table, which is locally recorded in the *testCase1.xls* file.
- *testCase2.m* : simulations are run for different thresholds $\in [\![5, 10]\!]$. No plots because of the dimensions involved, but results are stored in a table and locally recorded in the *testCase2.xls* file.
- *regProba.m* : plot different curves to study the influence of a and b in the expression of p.

Case 1 : Results and Analysis

Results and Basic Interpretation

We first ran basic simulations for the **case 1** on the given output vector Y.

The results we obtained were the following:

Optimal value	a	b
-8.0299	1.5047	-4.0778

Above, the optimal value has no direct interpretation. However, a is the coefficient for the hours of studying and b for the intercept of the affine line that represents the logg odd function. Thus entering these in the logistic regression equation or in the equation above, one can estimate the odds or probability of passing the exam.

$$log\ odds_{pass} = \log(\frac{p}{1-p}) = a * hours + b = 1.5047 * hours + -4.0778 \tag{11}$$

$$odds_{pass} = \frac{p}{1-p} = exp(\log(\frac{p}{1-p})) = exp(1.5047 * hours + -4.0778) \tag{12}$$

$$p_{pass} = odds_{pass} * 1 - p = \frac{1}{1 + exp(-(1.5047 * hours + -4.0778)} \tag{13}$$

For example if you study 4 hours you obtain:

$$p_{pass}(4) = \frac{1}{1 + exp(-(1.5047 * 4 + -4.0778)} = 0.87 \tag{14}$$

For studying 4 hours the probability of passing the exam is 87%.

In the graph below we can see the logistic regression curve fitted to the output data Y. The graph shows the probability of passing the exam versus the number of hours studying.

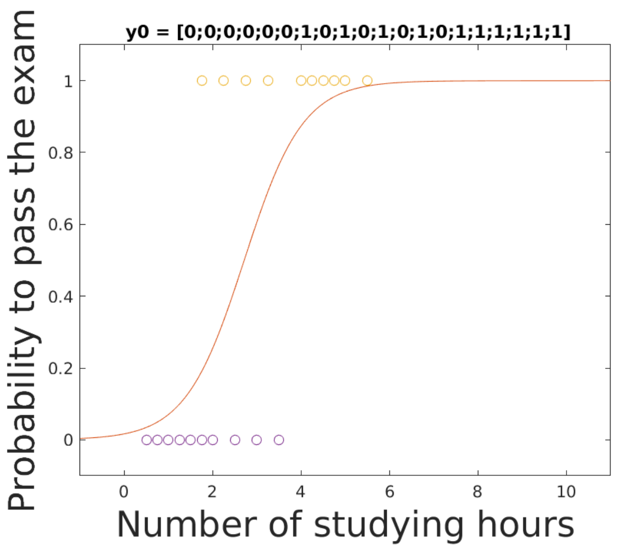


Figure 1: Probability of passing the exam with respect to the number of ours studying

Then we tested the influence of the output vector Y. More precisely, we changed its corresponding distribution from a simulation to another. The following table refers to the output testing vectors and the associated behaviour we wanted to test.

`Output vector`	`Associated behaviour to test`
y_0 = [00000010101010111111]	Initial *Wikipedia* vector
y_1 = [00000000001010111111]	Influence of the "random 1" part
y_2 = [00000000000000111111]	Influence of the "random 1" part
y_3 = [10000010101010111111]	Influence of the position of the first 1
y_4 = [00000000101010111111]	Influence of the position of the first 1
y_5 = [00001111000000001111]	Influence of to clouds of 1
y_6 = [10000000000000111111]	Influence of a "perturbation" in the output

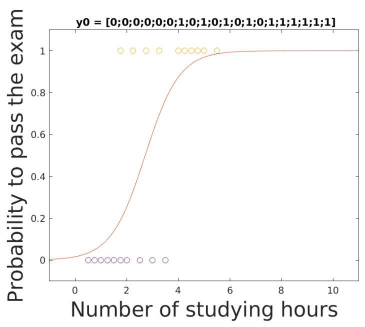
We present in the following table the results we got.

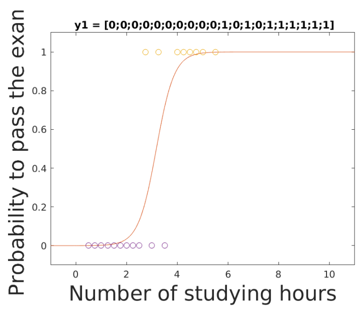
Vector	Optimal value	a	b
y_0	-8.0299	1.5047	-4.0778
y_1	-4.2656	2.8252	-8.9591
y_2	-6.961610^{-8}	4.227110^{5}	-1.564610^{6}
y_3	-10.524	0.94246	-2.3037
y_4	-6.1629	2.0194	-5.9333
y_5	-12.482	0.45228	-1.7048
y_6	-7.5059	1.4554	-5.1567

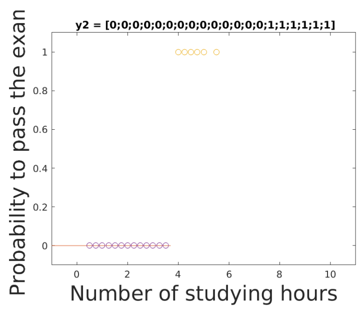
Analysis

In the following, we denote $\mathcal{C}_p$ the curve of the density of probability p.

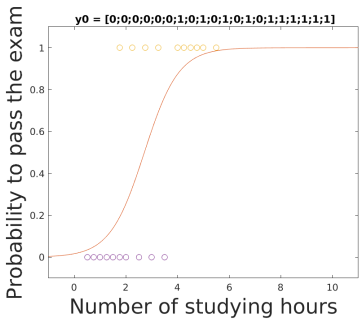
- **y_0 :** this is our reference element, give by the *Wikipedia* page. It will be the comparison point for the following interpretations.
- **y_1 & y_2 :** Reducing the size of the "random 1" area has the effect to divide by half the optimal value, thus increasing it. This implies an increase of the likelihood. When we totally remove this "random 1" area, we observe that the optimal value is almost zero, and that coefficients a and b highly increase in absolute value. In fact, the problem becomes more and more binary : there is a threshold under which you can not pass the exam. This is graphically translated by the fact that the slope of $\mathcal{C}_p$ increases for y_1 and "disappears" for y_2, just because the software did not plot a vertical slope. p became an `Heaviside` function.

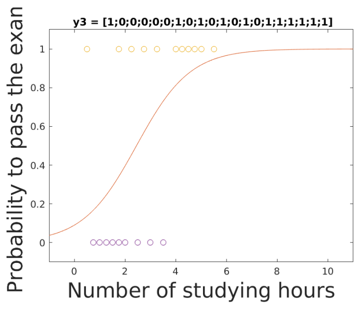


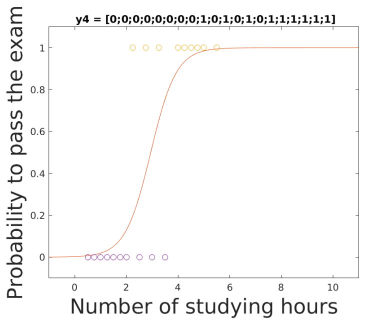




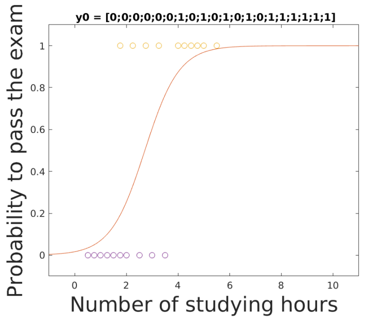
- **y_3 & y_4 :** When running with y_3, the slope of $\mathcal{C}_p$ decreases, which illustrates the fact that it is possible to pass the exam having worked a few time. p differs from the `Heaviside` function, in correspondence to the fact that the problem loses in binarity. When we shift the first 1 of the beginning of the output, as expected, the slope of $\mathcal{C}_p$ increases compared to the one of y_0 and is shifted to the right : the problem wins in binarity and one has to study more time than in the initial situation to pass the exam. The likelihood is higher for y_4 than for y_0, which is in line with the fact that y_4 increases the binarity of the output.

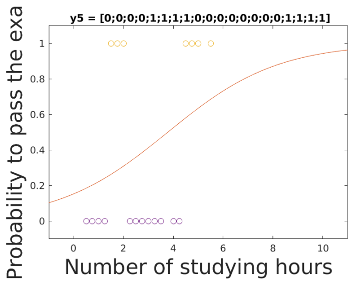




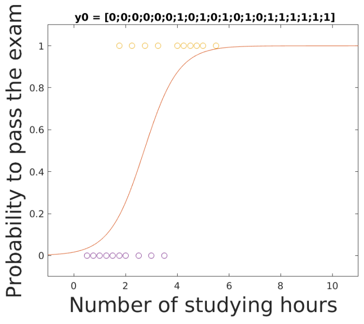


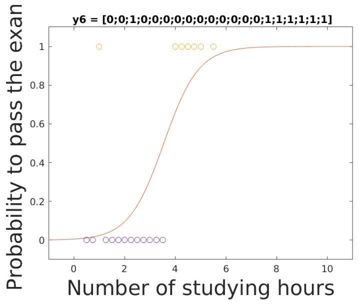
- y_5 **:** This time, we put two clouds of 1. Students are divided in two parts : those who studied between 1.5 and 2 hours passed and those who studied between 4.5 and 5.5 hours passed as well. As expected, a and b decrease in absolute value, and the slope of $\mathcal{C}_p$ decreases a lot. $\mathcal{C}_p$ tends to be an affine function. This is related to the fact that it is almost impossible to predict the success of a random student : it is possible to pass having studied less than two hours or more than 5 hours. In this case, a logistic model allowing "bumps" would be better, and using this, we would see appear camel bumps on our plot. This is the output vector which gives the lowest likelihood, which is consistent with the fact that the model is less adapted than for the other y_i.





- y_6 **:** This perturbation did not change the parameters and the optimal value a lot. Here we can refer to the "robustness" of the model and the estimation method : a single isolated value has a limited influence on the results (here, we put the perturbation at the very beginning of the output vector, what is intuitively supposed to have the biggest influence on the results). $\mathcal{C}_p$ remains almost unchanged.





In this case, we have used a single explanatory variable. An interesting tweak on case 1 could be to consider that not only number of hours studied, but also other properties such as hours of sleep night before exam, pre-requisite knowledge, food intake last 24h, e.g. could influence the exam result. This would imply several explanatory variables.

Case 2 : Results and Analysis

Results

For the experiments, we ran the algorithm for different values of the threshold, and observe the variation of parameters a and b. Note that now, given that input points $(x_i)_{i=1,..,m}$ have each one n coordinates, $a \in \mathbb{R}^n$. In order to get significant information, we also computed the norm 2 of a, given that analyzing it component-wise would be painful and lead to not necessarily relevant information. One can find every vector a in the file *case2_ tests.xls* provide in the archive, which is generated when running *case2_ tests.m.*

Threshold	Optimal value	$\|a\|$	b
5	-217.49	114.44	-101.62
6	-828.77	37.21	-41.45
7	-437.84	70.82	61.76
8	-70.68	311.53	304.59
9	-0.0058267	71161153.69	-3352302.94
10	-0.0058267	71161153.69	-3352302.94

Analysis

First, it is immediate to see that from 6 to 9, the optimal value increases. It reaches approximately 0 for threshold 9 and 10. Inversely, the norm of a increases, such as the norm of b. However, we can not conclude to anything, since it is not the norm 2 of a that intervenes in the expression of our logistic regression function but $a^T * x$. Looking in the *.xls* file, we see that a has between 5 and 7 negative components each time. Considering the fact that $\nabla p(x) = \frac{exp(a^T x+b)}{(1+exp(a^T x+b))^2} a^T$, we realize that a simple slope interpretation is not enough in that case to extract any information of our results. We can notice that $\nabla p(x)$ has both negative and positive slopes, and the higher the threshold is, the higher these slopes are, what it similar to previously : the model becomes more and more selective, and thus (absolute) slopes increase. A last thing we notice is that threshold 9 and 10 give the same value of parameters. An interpretation we could give to that is that the number of wines with a rating above 9 does not really differ from the number of wines with a rating above 10. Thus, go from threshold 9 to threshold 10 does not really increase the selectivity of our model, and then parameters do not change (it should be a slight difference that numerically disappears). The following table confirms this intuition.

Threshold	Number of wine with quality above threshold
5	1536
6	855
7	217
8	18
9	0
10	0

Moreover, for these thresholds, the likelihood is almost zero. It is clear, given that there is no wine with a rating over 9 or 10, that a constant function would have been more adapted to this situation.
Interpretation of the signs in a : We can interpret signs in a as an information on the impact that a *descriptor variable* has on our prediction. A negative component in a would reduce the quantity $a^T x + b$ and thus bring $exp(a^T x + b)$ closer to 0, which means that the slope in that direction decreases : the descriptor in question does not say a lot about the prediction or outcoem we can expect. For example, the fourth component of a is always negative, which means that the "residual sugar" property is not crucial (looking in the data, we realize that values corresponding to that descriptor are approximately all the same). The last component in a being always positive, one can think that "pH" is a relevant information for our model.

Conclusion

To conclude, we can say that logistic regression can be very useful to find the corresponding probability distribution of a binary variable, and thus to predict the outcomes of new cases. However, it obviously does not give accurate results/distributions every time since not every behaviour can be interpreted with a sigmoid function as was shown

in our tests.
As an opening, the above binary logistic regression model can if slightly modified also be applied to more than two levels of the dependent variable, thus making it non-binary. This is then called a multinomial logistic regression, which is a more complex statistical classification problem. [7].

References

[1] Logistic regression, wikipedia online,
https://en.wikipedia.org/wiki/Logistic_regression

[2] ConvexOptimization, Stephen Boyd and Lieven Vandenberghe, Cambridge Univeristy Press

[3] Kaggle, online,
https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

[4] Analysis of wine quality data, PennState College, Applied Data MIining and Statistical Learning, online,
https://onlinecourses.science.psu.edu/stat857/node/223/

[5] cvx website, online,
http://cvxr.com/cvx/examples

[6] Reused CVX code, Argyris Zymnis,
http://web.cvxr.com/cvx/examples/cvxbook/Ch07_statistical_estim/html/logistics.html

[7] Multinomial Logistic regression, wikipedia online,

Appendices

We provide here the MATLAB code used to run the tests above.

Case 1 tests

```
%% Test script to study the influence of the values in Y on the probability found

% ------ Building of the table of descriptors, i.e. the number of studying hours

u1 = 0.5:0.25:1.75;
u2 = 1.75:0.25:3.5;
u3 = 4.0:0.25:5.0;
u = [u1 u2 u3 5.5]';

% ------ Building the different corresponding outputs

y0 = [0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1]';

% -> Influence of the middle 1 in the ouput
y1 = [0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1]';
y2 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1]';

% -> Influence of the position of the first 1 in the ouput
y3 = [1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1]';
y4 = [0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1]';

% -> What if two "masses" in the ouput ?
y5 = [0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1]';

% -> What about a "perturbation" in the output ?
y6 = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1]';


% ------ Table to store all the results and outputs
res = {};
outputs = [y0 y1 y2 y3 y4 y5 y6];
```

```matlab


% ------Start the tests

disp("Would you like to plot the probability curves ?");
disp(" 0 : No ");
disp(" 1 : Yes ");
choice = input('Please, make you choise : ');



for i = 1:length(outputs(1,:))
    disp(["*********** Test for y" num2str(i-1)]);
    y = outputs(:,i);
    [a,b,optval] = solveur(u,y);
    res = [res ; {optval,a,b}];

    if (choice)
        clear plot
        figure()
        plot(u,y,'o')
        axis([-1,11,-0.1, 1.1]);
        ind1 = find(y==1);
        ind2 = find(y==0);

        us = linspace(-1,11,1000)';
        ps = exp(a*us + b)./(1+exp(a*us+b));

        hold on
        plot(us,ps,'-', u(ind1),y(ind1),'o',...
                      u(ind2),y(ind2),'o');

        axis([-1, 11,-0.1,1.1]);
        chr = " y" + num2str(i-1) + " = " + mat2str(y);
        title(chr);
        xlabel('Number of studying hours', 'fontsize', 22);
        ylabel('Probability to pass the exam', 'fontsize', 22);

        % Save the plot
        filename = "y" + num2str(i-1);
        print(filename,'-dpng');

    end
end

cres = cell2table(res);
cres.Properties.VariableNames = {'Optimal_value', 'a', 'b'};
writetable(cres,'case1_tests.xls','Sheet',1,'Range','A1');
disp(cres);





%% Solve the minimization problem

function [a,b,optval] = solveur(u,y)
    m = length(y(:,1));
    U = [ones(m,1) u];
    x_size = length(U(1,:));
    cvx_expert true
    cvx_begin
        variables x(x_size)
        maximize(y'*U*x-sum(log_sum_exp([zeros(1,m); x'*U'])))
    cvx_end
    a = x(2)
    b = x(1)
    optval = cvx_optval;
end
```

Case 2 tests

```
%% Test script to study the influence of the threshold from which we consider that
    a wine is "good"


thresholdInf = 5;
thresholdSup = 10;


% ------ Table to store all the results and outputs
res = {};


% ------Start the tests

for i = thresholdInf:thresholdSup
    disp(["*********** Test for threshold = " num2str(i)]);
    [u,y,c] = wineInfo(i);
    disp(['The number of wines with a quality above ' num2str(i) ' is ' num2str(c)
        ]);
    [a,b,optval] = solveur(u,y);
    res = [res ; {optval,strjoin(string(a)),norm(a,2),b}];
end

cres = cell2table(res);
cres.Properties.VariableNames = {'Optimal_value', 'a','norm_of_a' 'b'};
writetable(cres,'case2_tests.xls','Sheet',1,'Range','A1');
disp(cres);





%% Solve the minimization problem

function [a,b,optval] = solveur(u,y)
    m = length(y(:,1));
    U = [ones(m,1) u];
    x_size = length(U(1,:));
    cvx_expert true
    cvx_begin
        variables x(x_size)
        maximize(y'*U*x-sum(log_sum_exp([zeros(1,m); x'*U'])))
    cvx_end
    a = x(2:x_size)
    b = x(1)
    optval = cvx_optval;
```

```
end




%% Get the descriptors and the associated output vector from red wine data csv
    file

function [descriptors,scores,count] =  wineInfo(threshold)

    M = csvread('/home/kiwi974/cours/epfl/convex_opti/project/data/winequality-red
        .csv',1,1);

     descriptors = M(:,1:10);  % Seem to be not bad : 2,
     scores = M(:,11);

    % If quality >= threshold then the wine is good (new quality = 1) else it is
        not
    % (new quality = 0)

    count = 0;
    for i = 1:length(scores)
        if (scores(i) >= threshold)
            scores(i) = 1;
            count = count +1;
        else
            scores(i) = 0;
        end
    end

end
```