Dauphine | PSL
UNIVERSITÉ PARIS

Report

Optimisation project report

Student :
Hassan El mansouri Khoudari

10 janvier 2021

Table des matières

1 Introduction **2**

2 Presentation of our dataset **2**

3 Problematic **2**
3.1 Aim of the project 2
3.2 Pre-processing 2

4 Data exploration **3**
4.1 Histogram of features 3
4.2 Target w.r.t Education 4
4.3 Target w.r.t Gender and Age 5
4.4 Visualisation of samples using PCA 6

5 Modelisation of the problem **6**
5.1 Important : Convergence criteria 7

6 Constant gradient descent **7**

7 Accelerated gradient descent **9**

8 Stochastic gradient descent **9**

9 Diagonal Scaling **11**

10 Regularisation **12**

11 Variance Reduction Technique (SAGA) **13**

12 Second-order methods (Newton Raphson) **13**

13 Conclusion **14**

1 Introduction

The goal of this document is to report the results I found in the optimisation assignment. In this project, I have chosen a data set on which I have applied the algorithms of optimisation seen during the course. I will try to gather the results I have found with a comparative study.

At first, we will present our data set as well as the goal of our modelisation. Then, we will look into our data using exploration and visualisation techniques before diving into the optimisation part. The latter will contain constant step gradient descent, accelerated methods, stochastic methods, etc.

2 Presentation of our dataset

The database contains records for 1885 respondents. For each respondent 12 attributes are known : Personality measurements which include NEO-FFI-R (neuroticism, extraversion, openness to experience, agreeableness, and conscientiousness), BIS-11 (impulsivity), and ImpSS (sensation seeking), level of education, age, gender, country of residence and ethnicity. All input attributes are originally categorical and are quantified. After quantification values of all input features can be considered as real-valued. In addition, participants were questioned concerning their use of 18 legal and illegal drugs (alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine and volatile substance abuse and one fictitious drug (Semeron) which was introduced to identify over-claimers. For each drug they have to select one of the answers : never used the drug, used it over a decade ago, or in the last decade, year, month, week, or day. Database contains 18 classification problems. Each of independent label variables contains seven classes : "Never Used", "Used over a Decade Ago", "Used in Last Decade", "Used in Last Year", "Used in Last Month", "Used in Last Week", and "Used in Last Day".

3 Problematic

3.1 Aim of the project

The aim of the project is to predict whether a person has an 'addictive' personality and therefore whether they are likely to be addicted to an illegal substance.

Our original data set gives us information on the degree of addiction of individuals to different substances (legal and non-legal). The notation used is - CL0 : The person has never used. - CL1 : Person used over a decade ago. - CL 2,3,4,5,6 : for uses at different frequencies, which we are going to confuse within the framework of this project.

We are only looking at illegal substances.

3.2 Pre-processing

The pre-processing part will therefore consist of : 1. Identifying illegal substances and keep only those in the table (nicotine, alcohol and benzodiazepine are also removed). 2. Remove individuals who did not respond with CL0 to the Semer substance which is just an invention to detect over-claimers. 3. Add an 'Addict' column with 1s and 0s to

differentiate between people who are addicted to one of the substances determined in the first step and those who have never used any CL0 substance or who have used over a decade ago (CL1) (maybe just to try and so presume it wasn't an addiction).

4 Data exploration

4.1 Histogram of features

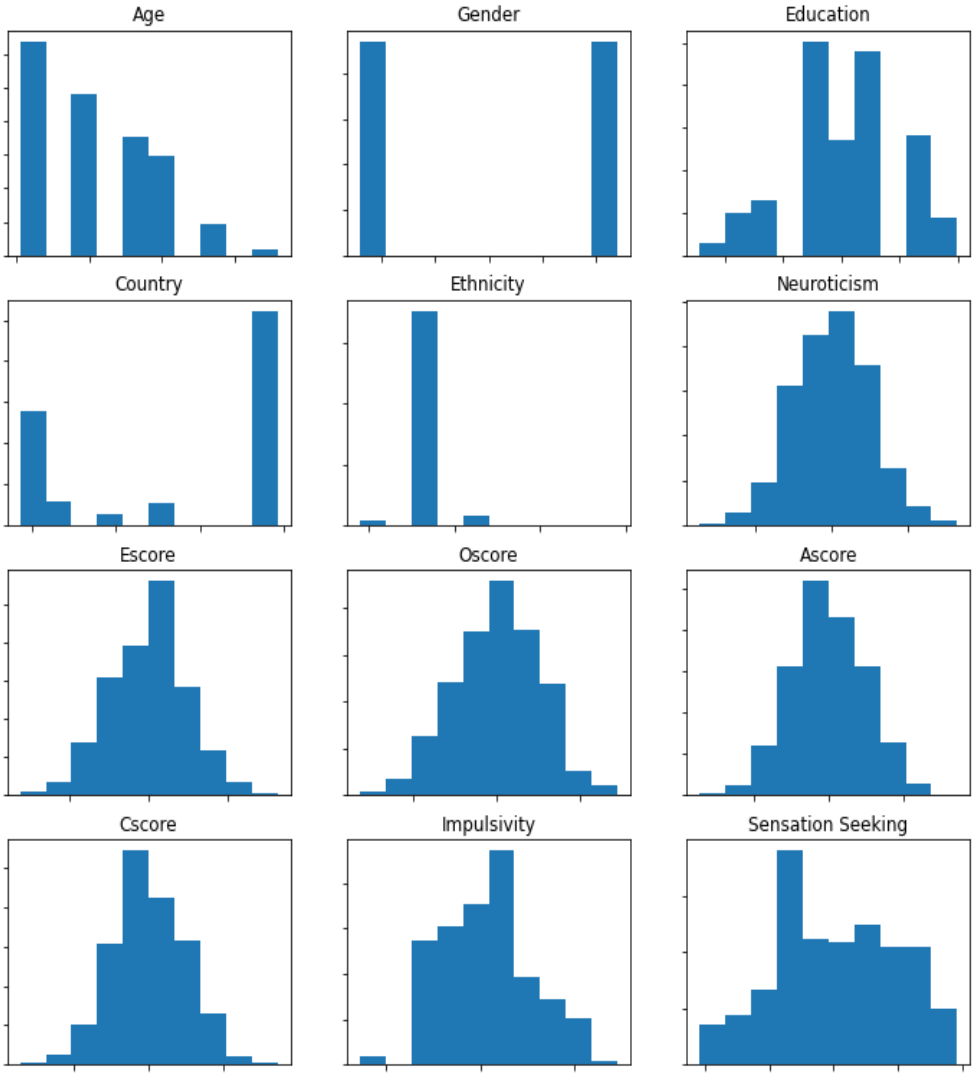


FIGURE 1 – Histogram of features

From the histograms and our data description, we notice the following :

— The data is skewed towards people that are addicted. Approximately 2/3 of our data set is constituted from people that are using a type of illegal drugs.

— We also notice that all of the features have been categorised into buckets from more useful insight. We have a sparse setting where every person that took the survey had multiple choices to pick from.

— We also notice that the features that represents a personality trait (Escore, Oscore, Impulsivity ...) all seem to follow a Gaussian distribution which is what we would expect from a non biased survey, that is extreme values are less likely to occur.
— We have as many females as males in this survey.
— Younger people (most in category 1) are represented more in the data set.

4.2 Target w.r.t Education

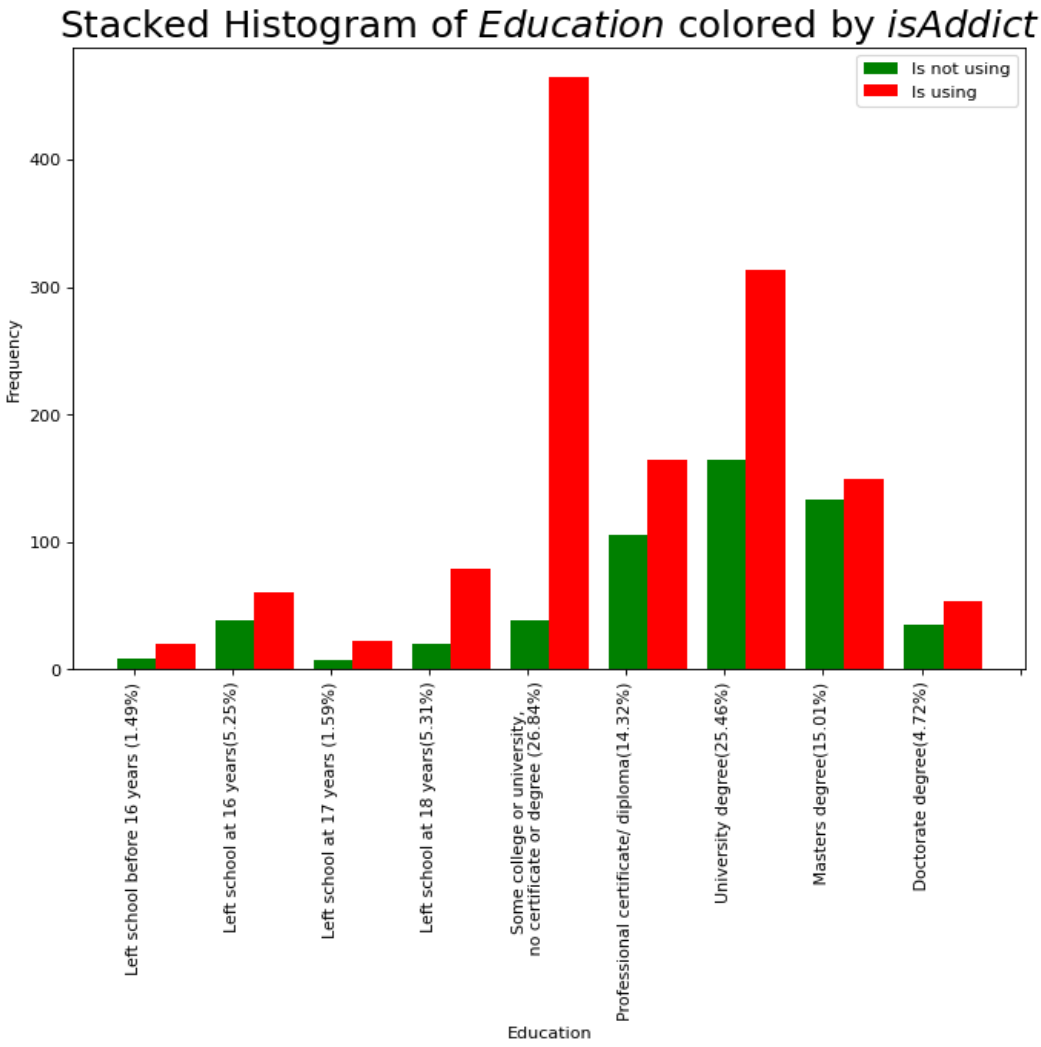
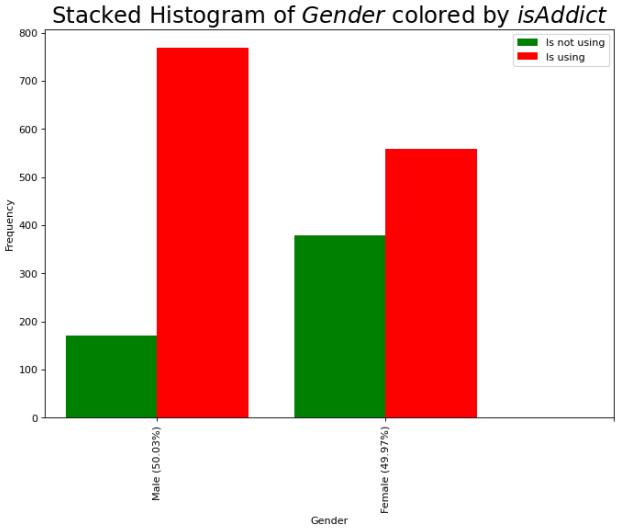


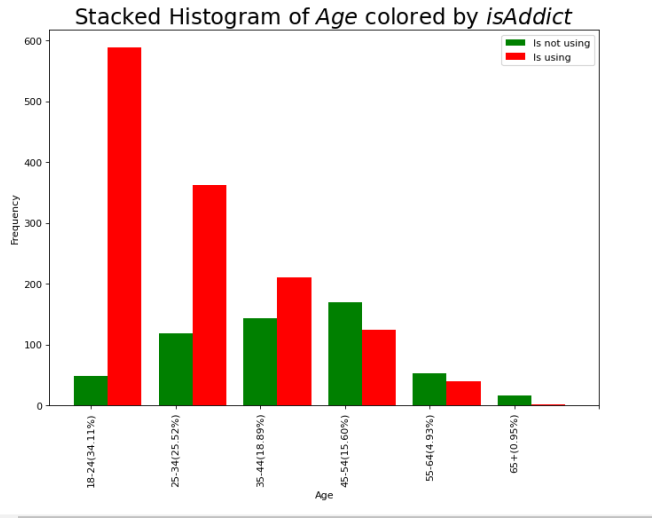
FIGURE 2 – Target w.r.t Education

Taking into account the prior on the number of people in every category, we should not directly compare the numbers as it is biased. However, we can still get some insight :
— The rate of using drops drastically between the category of people who had some college education without degree (college dropout) compared to those who have a university degree. (Those categories having the same amount of people, this conclusion is not biased)
⟶ The greater the education the lower the ratio using/not-using

4.3 Target w.r.t Gender and Age



(a) Target w.r.t Gender



(b) Target w.r.t Age

FIGURE 3 – Target w.rt to other attributes effect

⟶ Male consumes clearly more drugs than females. This will be confirmed later when doing Lasso feature selection.
⟶ The number of people using decreases as the age of the individual is high. Young subjects (18-24yo) consume the most. While elder have minimal use rates.

4.4 Visualisation of samples using PCA

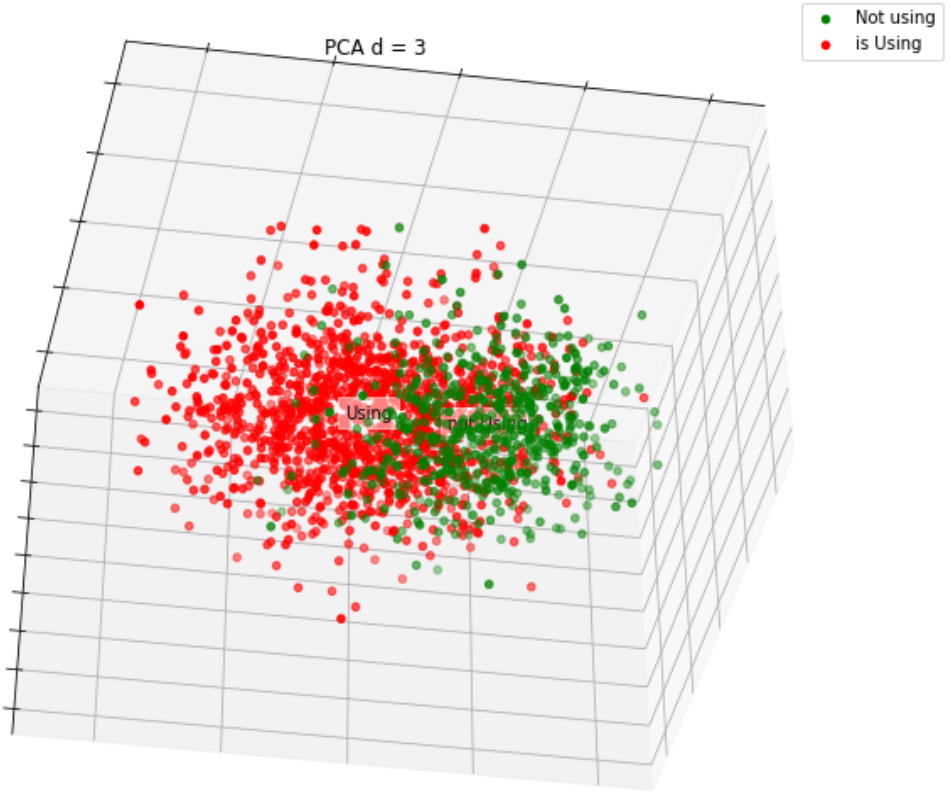


FIGURE 4 – Data distribution using PCA (dim = 3)

⟶ The PCA figure shows us there is distinctive regions for every category. We can imagine there being a hyperplane separating the two clusters especially that in higher dimension the separation might be more evident than the 3-d case. Thus, we can expect a relatively high accuracy.

5 Modelisation of the problem

We use the following soft classifier :

$$h_w(x) = \sigma(w^T x)$$

The result of the classification will be :

$$h_w(x) = \begin{cases} 1 & \text{if : } h_w(x) > \frac{1}{2} \\ 0 & \text{else} \end{cases}$$

⟶ We consider the matrix X as our training data, Y as the training labels.

The loss function that is suitable in our case is the binary cross-entropy :

$$\tilde{f}(x, y, w) = -y \cdot log(\sigma(w^T x)) - (1 - y) \cdot log(1 - \sigma(w^T x))$$

$$f(w) = \frac{1}{N} sum(-Y \odot log(\sigma(w^T x^T)) - (1 - Y) \odot log(1 - \sigma(w^T X^T)))$$

By using the Taylor approximation, we obtain the gradient of the loss function for one data point :

$$\nabla_w \tilde{f}(x, y, w) = -y \cdot x + x \cdot \sigma(w^T x)$$

Thus, for the mean of data points :

$$\nabla_w L(w) = \frac{1}{N} \sum_{i=1}^{N} -y_i \cdot x_i + x_i \cdot \sigma(w^T x_i) = \frac{1}{N} sum(-Y \odot X + X \odot \sigma(w^T X^T))$$

The Hessian w.r.t w is :

$$Hessian_w f(x) = x x^T \cdot (1 - \sigma(w^T x))$$

Thus, for the mean of data points, we have :

$$Hessian_w L(w) = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T \cdot (1 - \sigma(w^T x_i)) = \frac{1}{N} [(ones(N, 1) - \sigma(w^T X^T)) X^T X]$$

$\longrightarrow$ **L is convex because** $X^T X \in S^+$ **and** $1 - \sigma(w^T X^T) > 0$**, thus** $\nabla_w^2 L(w) \geq 0$
Now, that we have layed down terminology, we can start experimenting with our optimisation algorithms.

5.1 Important : Convergence criteria

In order to judge our algorithms in an neutral way, we will compute the time needed to reach the following condition :

$$\|\nabla L(w)\|_\infty < 0.05$$

Also, we will give all the algorithms a budget of 200 epochs.

6 Constant gradient descent

By noticing that the hessian depends on w, we can't adopt a constant step $= \frac{1}{L(w)}$. However, we notice that :

$$Hessian_w L(w) < \frac{1}{N} X^T X$$

Thus, the L we take into account is the maximum eigenvalue of the positive definite matrix $X^T X$ (the upper bound of our Hessian).

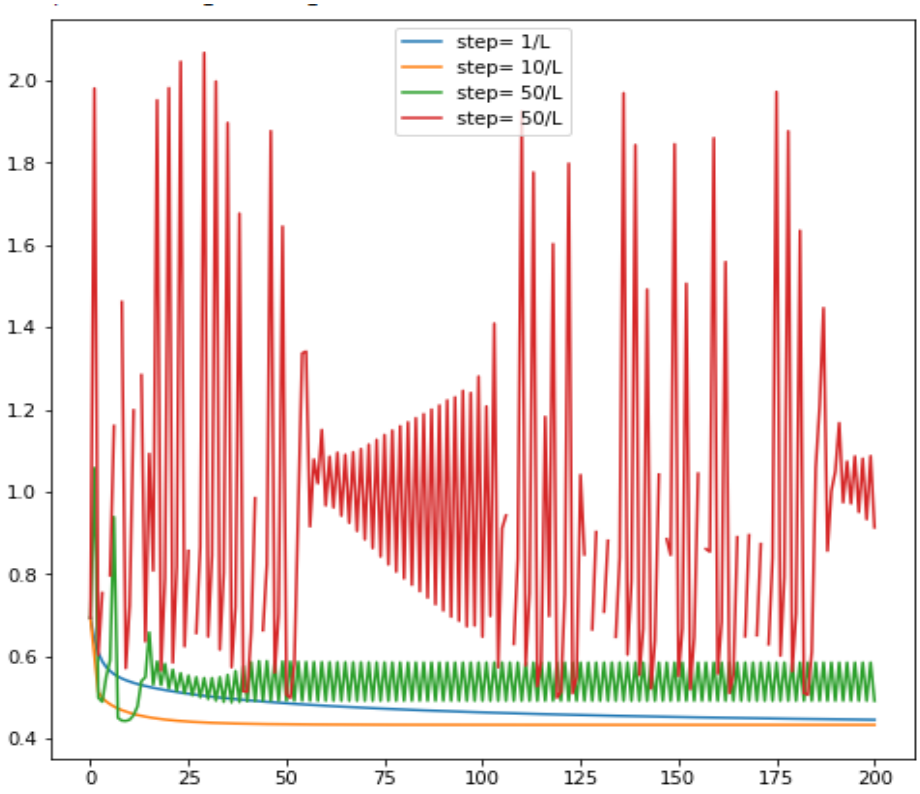


FIGURE 5 – Step size choice impact on convergence

	GD($\alpha = \frac{1}{L}$)	GD($\alpha = \frac{10}{L}$)	GD ($\alpha = \frac{50}{L}$)
Time to reach cv	0.026s	0.024s	Not reached

TABLE 1 – Time to reach convergence

Analysis :

For $\alpha \in \{\frac{1}{L}, \frac{10}{L}\}$, the algorithm converges to a constant point.

For the case $\alpha = \frac{1}{L}$, we have the convergence guarantee in the course, and for $\alpha = \frac{10}{L}$, we notice that we still have convergence but it's slightly faster as we have increased the step size.

Now, for $\alpha \in \{\frac{50}{L}, \frac{100}{L}\}$, we don't converge and we have a rather chaotic behavior especially for the latter step size, that is because near the minimum we make steps too large that we go passed the minimum and 'keep swinging' around the minimum.

7 Accelerated gradient descent

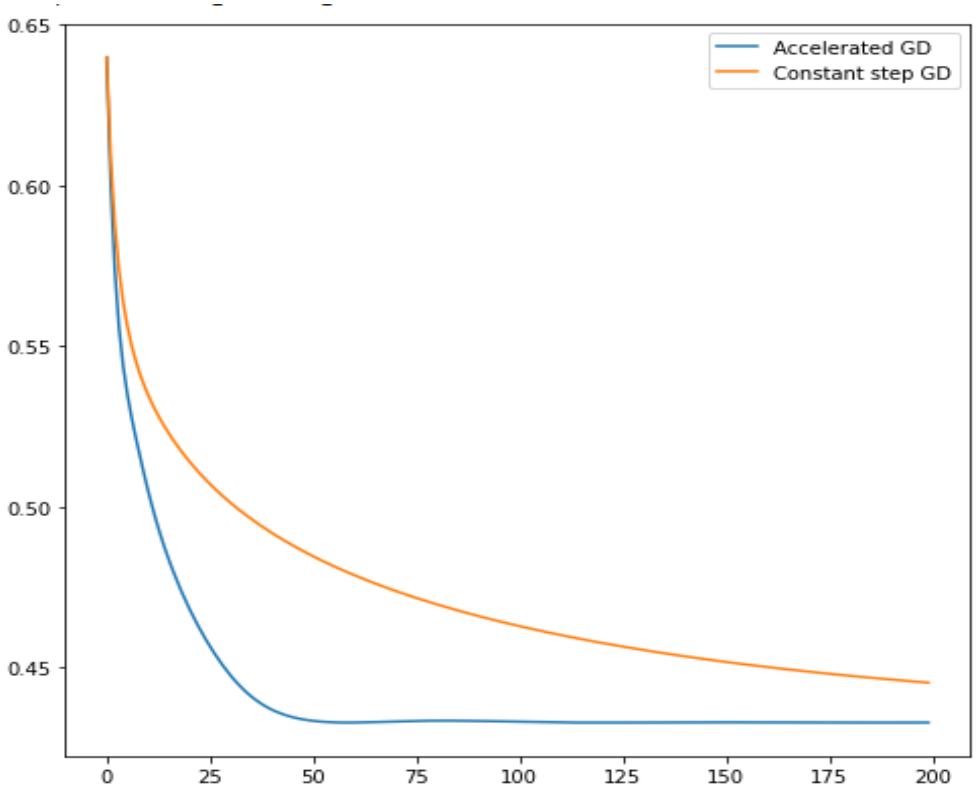


FIGURE 6 – Nesterov GD vs classical GD

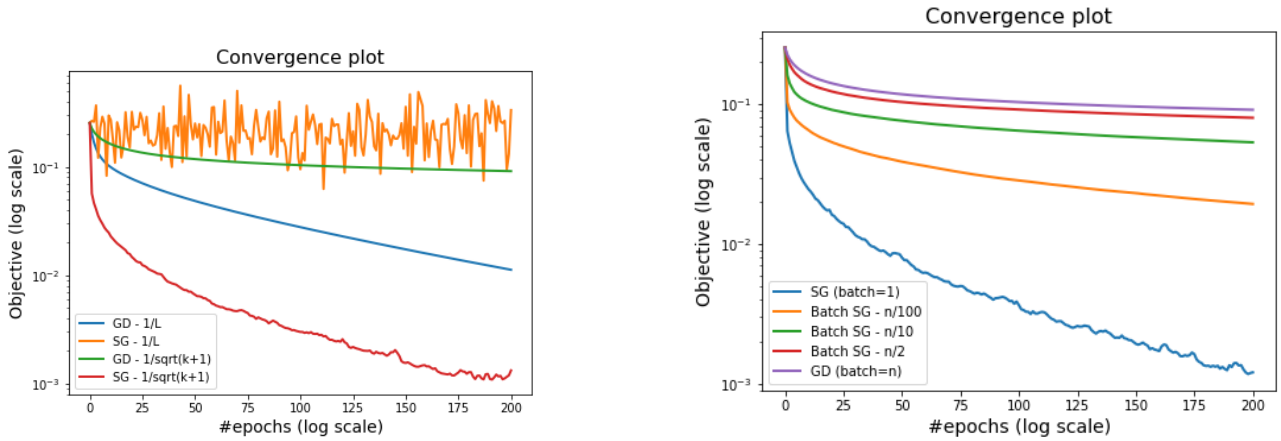
Methods	Accelerated GD	Classic GD
Time to reach cv	0.0091s	0.0269s

TABLE 2 – Time to reach convergence (in seconds)

As we would expect, the Nesterov algorithm is faster than it's classical counter part. That is because it uses information about previous iterates to move w_{k+1} both in the gradient direction and according to the direction of momentum. That is very useful and works in practice specially for L-smooth convex functions which is our case. The algorithm converges at 0.009s compared to the classical GD which converges at 0.02s

8 Stochastic gradient descent

The main problem with Batch Gradient Descent is the fact that it uses the whole training set to compute the gradients at every step, which makes it very slow when the training set is large. At the opposite extreme, Stochastic Gradient Descent just picks a random instance in the training set at every step and computes the gradients based only on that single instance. There is a compromise that can be made using mini-batch Gradient methods.



(a) Step size effect

(b) Batch size effect

FIGURE 7 – Hyper-parameters effect on convergence

In the next figure, we explore the step size scheduling and batch size and its effect on convergence :

We can notice :

— Stochastic Gradient descent with fixed step size oscillates away from the optimal value due to the noise that's inherent to the random choice of the sample.

— Randomness is good to escape from local optima, but bad because it means that the algorithm can never settle at the minimum. One solution to this dilemma is to gradually reduce the learning rate. The steps start out large (which helps make quick progress and escape local minima), then get smaller and smaller, allowing the algorithm to better approach the global minimum.

— We can notice that by decreasing the batch size, we have a faster convergence (we kept the decreasing step size setting). This is due to more frequent updates (for every batch size samples). However, we can see some noise on the blue plot in the right figure (due to the randomness of the chosen point)

9 Diagonal Scaling

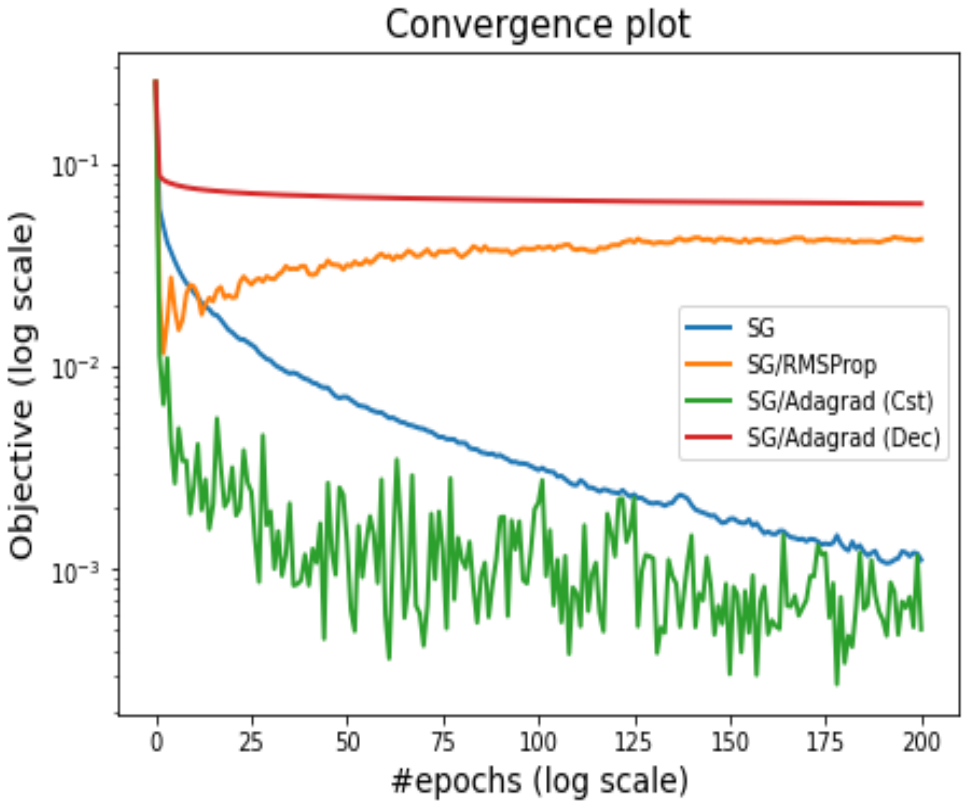


FIGURE 8 – Diagonal Scaling

Accuracy of	SGD bs=1	GD bs=n/100	GD bs=n/10	Full batch GD	RMPSProp	AdaGrad (Cons)
Training	0.787	0.776	0.7585	0.732	0.802	0.790
Testing	0.785	0.774	0.740	0.740	0.819	0.768

TABLE 3 – Time to reach convergence

In the diagonal scaling methods, we do not have particularly better results than Stochastic gradient. Except for Adagrad method with decreasing step size that seem to be faster at converging but with higher noise.

However, in terms of accuracy w.r.t training and testing sets, the diagonal scaling methods (in particular RMSProp) yields better results at around 81.9% on test versus 77-78% for stochastic methods while the full batch gradient method has the lowest performance at around 73%.

Due to our limited epochs budget, one can understand that faster methods will be the ones that have the better accuracy on the data distribution.

10 Regularisation

A good way to reduce over-fitting is to regularize the model (i.e., to constrain it) : the fewer degrees of freedom it has, the harder it will be for it to over-fit the data.

We will experiment with Lasso Regression which tends to completely eliminate the weights of the least important features (i.e., set them to zero). In other words, it automatically performs feature selection and outputs a sparse model (i.e., with few nonzero feature weights).

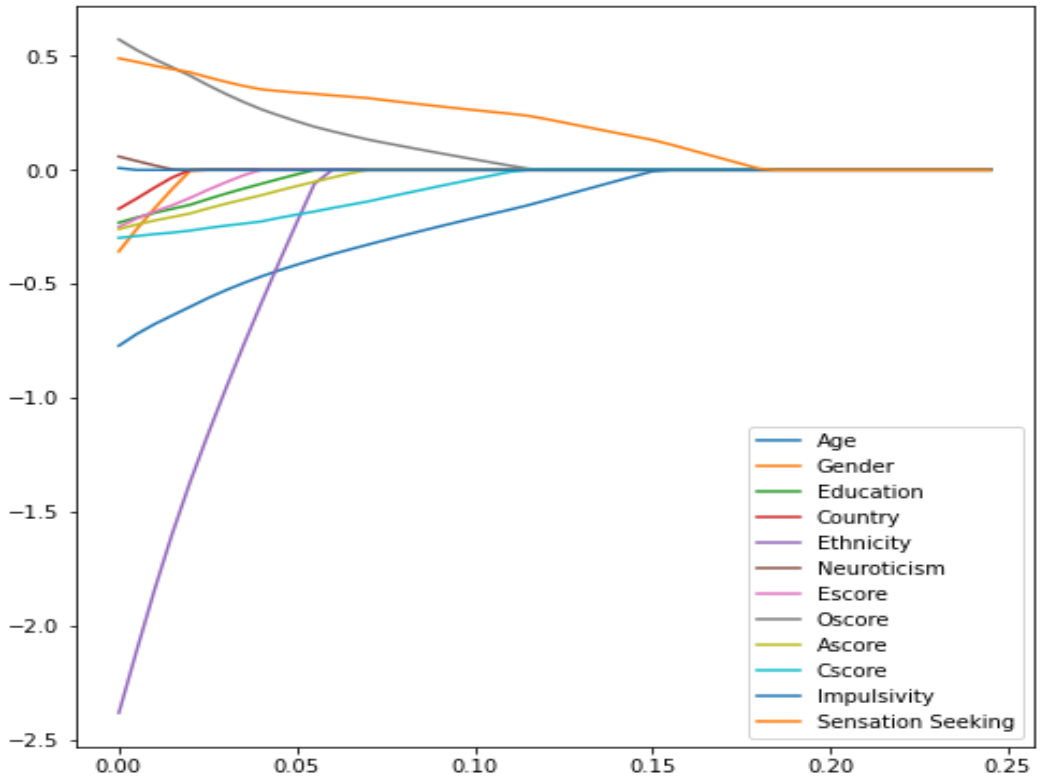


FIGURE 9 – Lasso feature selection

Analysis :

We can get qualitative insight from this graph, because features that become equal to zero the latest are the most representative :

- The gender and Age features are the most indicative of a person's addictiveness, which confirms what we said previously in the data exploration part.
- The Cscore(Conscientiousness) and Oscore (Openness to experience) have the same importance and come approximately in third place.

Accuracy of	GD with $\lambda = 0$	GD with $\lambda = 0.05$	GD with $\lambda = 0.1$
Training	0.781	0.732	0.732
Validation	0.785	0.745	0.740

TABLE 4 – Performance of model w/, w/o regularisation

Analysis :

We have an unusual setting here where the validation score always outperforms the training score for all values of λ. As we have a sparse data set with a relatively small size, one can intuitively think that regularization isn't necessary especially that we have N»d.

11 Variance Reduction Technique (SAGA)

In order to reduce variance of our estimator, one can use variance reduction techniques such as SVRG, SAGA that yield to unbiased estimators. In this section we will experiment with the SAGA algorithm.

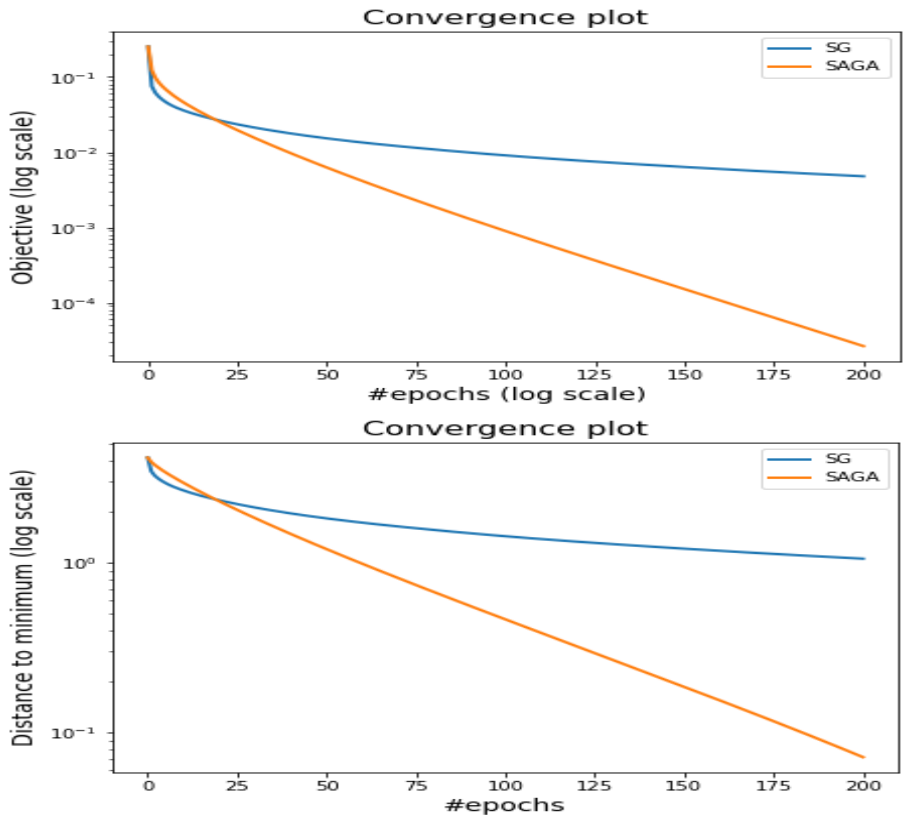


FIGURE 10 – Stochastic gradient vs SAGA

Analysis :
Using SAGA, we seem to have an exponential descent of $f(x_k) - f(x^*)$, that is because we overcome the noise that prohibits the classical stochastic methods to reach convergence.

12 Second-order methods (Newton Raphson)

In second order methods, we not only put gradients to use but also the hessian matrix. The newton-Raphson algorithm uses the following update step :

$$x_{k+1} = x_k - (Hess(x_k))^{-1} \nabla f(x_k)$$

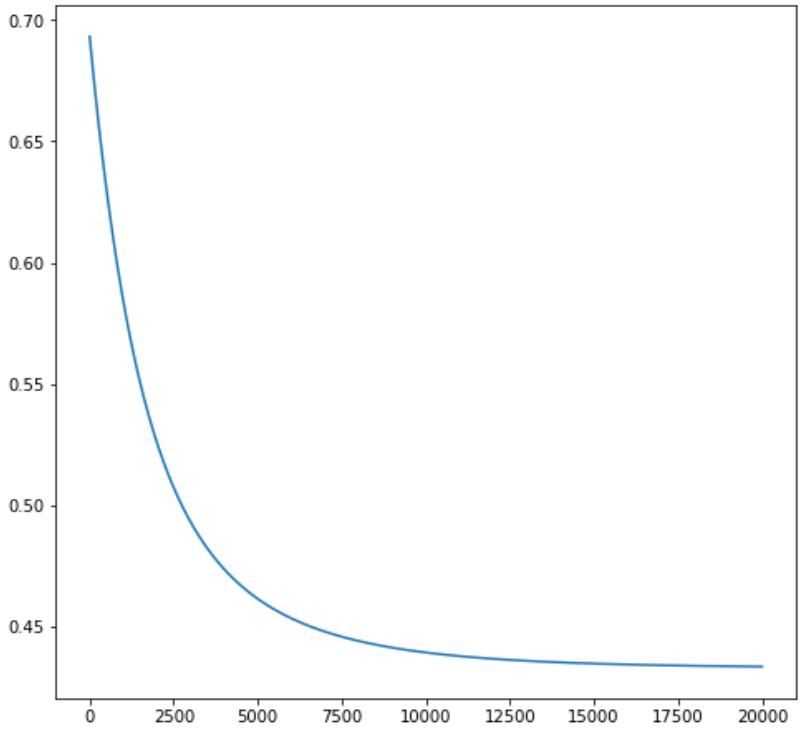


FIGURE 11 – Second order method : Newton

Analysis :

The algorithm is very slow compared to the other algorithms (I had to give him a budget of 20000 epochs in order for the convergence to occur, and took 3.1s to converge).

Normally, we have a quadratic convergence guarantee, however, I did not witness this rate here. It might be an odd setting in which the convergence is slower than usual.

I have checked that the implementation is correct but it remained a mystery to me why the algorithm was very slow to converge.

In any case, second order methods are not necessary here and are rather impractical as they require a high memory overload to calculate the hessian matrix (size d*d)

13 Conclusion

In our specific setting, it is the stochastic methods with diagonal scaling that gave the best performance both on training and testing sets. They have also presented great convergence speeds and accuracy compared to more classical methods like full batch gradient descent.

One can wonder if the results would be any different with data sets that presents less sparsity and with higher volume.

In response to our initial goal, we have been able to correctly classify non-users from users and have found that certain personality traits, age and gender are parameters that are sensible to our classification task.