LAMOUREUX Thibaut

OUSTRAIN Edgar

VBA projet: The perceptron in finance

Report

First Step

First to start the project, we need to prepare the data. We create an array where we put the stock price of the excel sheet to use it after. We calculate the daily returns and forward returns with 0 and 1 which are our true output. Then, we calculate the Simple moving average over 20 days and the Bollinger upper and lower bands which are going to be our inputs for the perceptron. All this is done thanks to functions called in the main. After that we write the SMA, Upper and Lower bands on the sheet to see the values and if it is consistent.

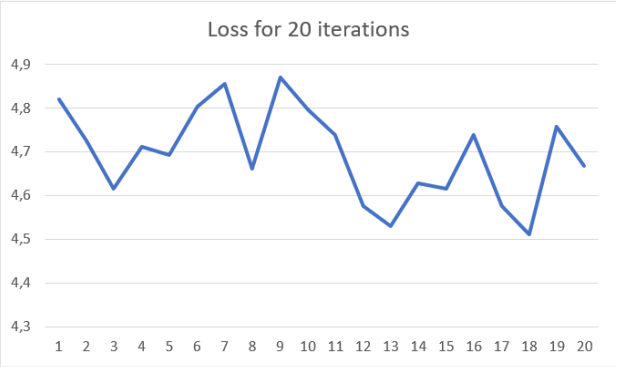
Second Step

Now we create a class Simple Perceptron to implement the perceptron. We create a training array which represents 80% of our data. We initialize the output, the weights, the number of iterations, an array for the inputs, the error and the loss. We make a loop on the number of iterations. Then, we create a function that initialize the weights randomly. We have 3 weights for the 3 inputs which are SMA, Upper Band and Lower Band and also a bias. We make a loop on the number of rows in the training set. We fill the inputs array with values calculated during the preparation of the data. We use the function Forwardsig to calculate the predictions which should be a 0 or 1. First, we calculate the value of z by multiplying inputs and weights. We had a condition in the function so that the value of z does not exceed 700 or -700 which is due to the calculation weakness of VBA. After we use the sigmoid formula and establish if the output is a 0 or 1 depending on the value of the sigmoid. If it is superior to 0.5 it is a 1 else it is a 0. We implement an array of the predictions to see that they look like. Then, we calculate the error between the output (prediction) and the true output (target). We call the function loss to calculate the loss. We use the formula given in the subject. Because log(0) does not exist we need to add a small number(Here we choose 1x10^-4) so that the loss could only be very close to 0 but not 0. Then, we calculate the difference of the weights in order to calculate the new weights. We have a function that calculate the weights using inputs and error just like in the pseudo code in the subject. Now, we can calculate the new weights and bias by using the backward function with a learning rate of 0.1. This the main part of our perceptron algorithm.

We need to repeat these steps as many times as there are rows in the training set. Finally, we obtain, 3 features which are loss, weights and bias. We need to divide them by the number of rows in order to find the mean of these values. After, we do that 20 times to find the minimization of the loss by using the gradient descent method. We have a function which

calculate the confusion matrix for each iteration to where the model is the best. After 20 iterations we should find a loss smaller than the first iteration which shows that our model improved.

Third Step



1) We plot the graph of the loss at the end of the learning process,and we observe that over the time the perceptron is converging. In fact, we see that the curve tend slowly tends to 4.7. We also see that the loss is decreasing over the time which is due to the gradient descent algorithm.
2) The values of the weights represent the importance of the different variables which are SMA, Upper band and Lower band. So, we will take the weights where the loss is minimal because they are the best to predict our model.
3)

```
Matrice de confusion:
            Prédiction
           | 1 | 0 |
-----------|---|---|
 Cible 1   | 374 | 346 |
 Cible 0   | 344 | 346 |
```

We obtain the confusion matrix above. We calculate the precision:
374/(344+374)=0.521
We have a precision of 52.1%. The model is not really good.

4)

```
Matrice de confusion:
            Prédiction
           | 1 | 0 |
-----------|---|---|
 Cible 1   | 374 | 346 |
 Cible 0   | 344 | 346 |
```

With the hyperbolic tangent activation method, we obtain the same result of precision than the sigmoid activation. 374/(344+374)=0.521

5) With the Glorot Initialization we obtain a precision of 51.3% which is close to the first one. However, we should obtain a greater result because this is a better way of initialization. There must be an error in our data.

We tried to do a Multiple Perceptron but we are not sure about our results.

AR-Garch Part

First of all, we started by putting the classical values that we knew, such as the returns and the data we had in the first place. After that we tried to put the different values that we had in the screen shot in order to initialize the basis of the optimization with the Ar-Garch model. We choose to try the model with the values we had in the screenshot.

We understood the model like that:

- First, we had the formula for the GARCH(1,1) part with the alpha 0, alpha 1 and beta 1 values. These values would be useful for the calculation of the sigmas of the model. The epsilon would be initialized with a function on excel to represent the Standard Gaussian Variable. We also had the first sigma but we knew it could change depending on the optimization of the model.
- After that we had to understand clearly the part about the AR(n) part in order to initialize this part with the different eta values and the other values. We first put the values on the screenshot for eta 1 and eta 2 to be able to calculate the other values such as the predicted results, the residuals of the AR-part and the log-likelihood values.
- Then, we understood that we needed to minimize the log-likelihood value to have the best model. It would be done by optimizing all the precedent values we mentioned (etas, alphas, beta and sigma0). We decided to put some text in a cell of excel to avoid using an userform module for the choice of the n etas in the AR(n) model.
- Finally, we finished by trying different macros. In the end, we decided to use some constraints on different variables and values in the end. The principal constraints were on the sum of all the different likelihood for the predicted returns and the returns. We also put a constraint on the variance and the beta 1. It resulted so that we have the sum of the likelihood superior to 0 and inferior to 1700 so that we have a good likelihood. We used 3 different macros 1 to optimize the log likelihood, 1 to delete all the etas values and 1 to create the different values for the etas.

To conclude, we can say that these macros can be optimized but we think that they are efficient for this type of work.

We used a reference to the solver in order to execute the macro.

Just to let you know, we find this project relatively difficult and spend over 60 hours to get to this result.