Luan Dinarica, Nicolas Vuillod December 2022
T30-1 :(

Miniproject BIO-322 : Report

Introduction :

We have access to a data set that contains the normalized counts for 32285 genes in 5000 different cells together with the experimental condition under which each cell was measured. We think about reducing the number of predictors. However, if we reduce it too much, we will lose information and then accuracy too. It is an art to find the perfect balance.
Then, the main part of the project was to test linear and nonlinear methods to make those predictions. In the course, we talk about many techniques, so we need to decide the ones that seem the most appropriate with the configuration of our data set (there is not a single method that works best for every data set). Furthermore, the methods can be optimized with the help of hyperparameters that can be tuned, in order to fit the data as well as possible. Finding the best hyperparameters is also an art.

Clean the data :

The data set we received was already normalized, so we don't need to do it. Then, the first step was to check if there are no missing values in our data that we should remove, hopefully it was not the case. After that, we removed irrelevant predictors from the data set (constant and correlated predictors are not useful) and also provided a linear subspace close to the data with the analysis of the principal components (PCA). With the PCA, we can decide the cumulative proportion of variance that we want to keep from the global data set. Firstly, we tried to keep 90% of the total variance and we got about 3000 predictors left, but then we tried to keep 95% and we figured out that it does not increase a lot, because we got 3534 predictors left. So we can conclude that a major part of the genes was not relevant to determine the experimental conditions and these 3534 genes combinations look sufficient and effectives to correctly classify the cells. If we were interested to know the most relevant genes which create these combinations, we should have a look at the matrix (Φ) given by the PCA.

Linear methods :

For the linear methods, we tried two principal techniques : Ridge classification and Lasso classification. In both of these methods, there is a hyper-parameter (lambda), which is a regularization constant that helps to minimize the loss of the model.
With the results obtained, we quickly decided to use the Lasso classification which gives better predictions. Indeed, the Lasso predictions can drop coefficients to zero and so avoid having a too flexible method which would cause overfitting.
Then, we decided to optimize as much as possible the Lasso classification, by tuning the hyperparameter lambda. That was the trickiest part of this optimization because the value of lambda must be very precise, so we need to find strategies to figure out the one which gives the best predictions.
The first thing we performed was a general tuning model with many lambdas and a big range because we did not have any idea about the scale of it. Then with the best value we got, we ↕closed the range of search near this lambda and we determined its boundary by plotting our tuning model (**Figure 1**). Finally, by repeating this procedure a few times, we were able to

find a lambda that gives us a pretty good result for the predictions (91,747% of accuracy on kaggle).
To the question : Are linear methods sufficient to classify the data or are non-linear methods needed for high classification accuracy ? We would say yes because the best result we got was with a linear method (Lasso), but we need to consider the fact that it was easier to tune it and probably with a good tuning of nonlinear methods (like Neural Networks or Gradient Boosting), we could reach a better result.

Nonlinear methods :

In the course, we saw different nonlinear ways to make predictions and we choose to try some of them : Neuronal-Network, Random Forest, Gradient Boosting and some clustering methods (KMeans and DSCAN)

Neuronal-Network (our best nonlinear method) : We tried two kinds of Neuronal-Networks, one with one hidden layer and one with two hidden layers. Taking more layers means having more hyperparameters to tune, so it becomes harder to find the best combination of them. However, if you add another hidden layer, you need an exponentially lower number of units to get the same number of connections. Hence, you can have almost the same capacity with an exponentially smaller number of units if you use multiple layers. This method being complicated to tune and taking a long time to run, we decided to begin by fixing the number of neurons in each layer, the epoch and the dropout to try to find a good lambda (for that, we proceed the same way as for linear lasso). After that we arbitrarily fixed the dropout after discussion with assistants. Finally, we tuned together the number of epochs, and number of neurons in both layers to find the best combination because they seem correlated. With our best hyperparameters' values, we reached 89.644% accuracy on kaggle.
Random Forest : The result with this classic tree based method was not good at all; we didn't manage to surpass 63% predictions on kaggle hence we chose to give up this method and to focus our work on another tree based method (Gradient Boosting).
Gradient Boosting : With this technique, we obtain 86.57% as the best predictions on kaggle, despite the accuracy for the validation set being 89.9% (overfit ?). It does not give us our best result, but this method also has advantages : very fast, constant and easily tuned. However, we could reach better results with NeuralNetwork, so we did not choose this method.
Clustering : The clustering methods are unsupervised and can help us to visualize how the data are arranged in the space. Hopefully, the two clustering methods we used gave us consistent results. Indeed, we ran KMeans clustering on the train data (with different initial distributions) and we were looking for 3 clusters that should represent the 3 experimental conditions. We plot the associate confusion matrix and, most of the time, it could not separate the data point into three different clusters and make one big cluster with the main part the point inside (**Figure 2 : 1, 2, 3**). Rarely, it separates them into two equivalent clusters (**Figure 2 : 4**), but we think that it depends on the initial repartition. Furthermore, when we use the density-based approach (DBSCAN), the result of the confusion matrix is even more extreme, because all the points are classified in one single cluster every time (**Figure 3**). It occurs even when we take a small radius, this shows the high density.

Based on those results, we can say that the training data is not easily separable in different clusters, which means that the data are spatially close to each other. We can suppose that the gene expression is almost similar for every experimental condition, it does not have a difference big enough to separate them by clustering. We can see on the TSne projection (**Figure 4**) that all the data are superposed, even if this method tries to rearrange the data such that all point clouds are distinguishable in the 2D plane.
We can conclude that unsupervised learning methods are not efficient for our data set.

Annexes :

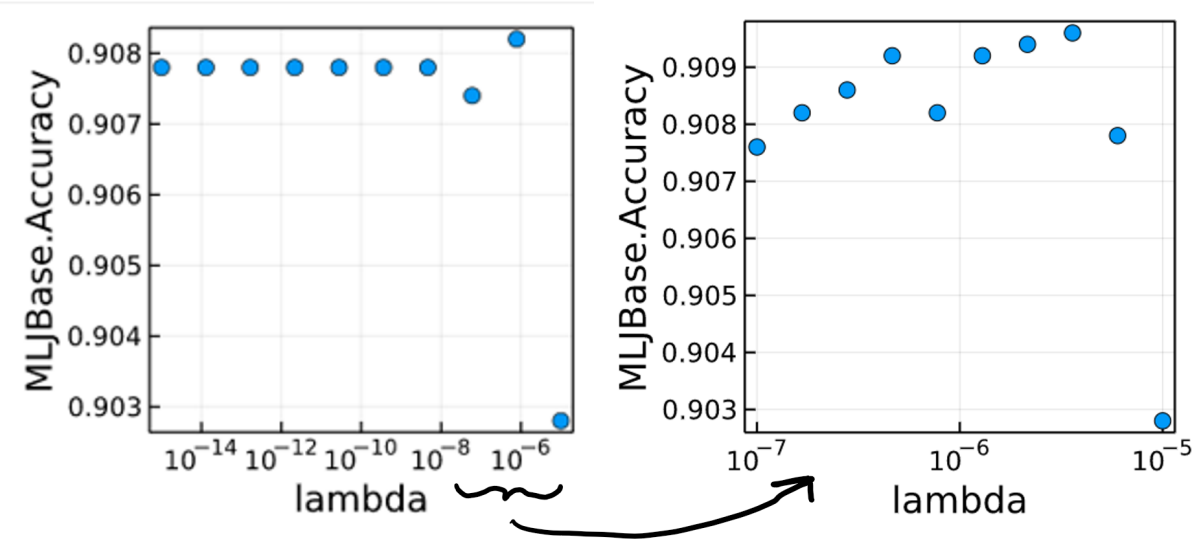

Figure 1 : Illustration of the technique we have done to find the best hyperparameter.

1

	Ground Truth		
Predicted	CBP	KAT5	eGFP
CBP	145	209	200
KAT5	0	1	0
eGFP	1707	1346	1392

```
confusion_matrix(KMeans_predict, clean_data.labels)
```

2

	Ground Truth		
Predicted	CBP	KAT5	eGFP
CBP	1	0	0
KAT5	1851	1555	1592
eGFP	0	1	0

```
confusion_matrix(KMeans_predict, clean_data.labels)
```

3

	Ground Truth		
Predicted	CBP	KAT5	eGFP
CBP	1852	1554	1592
KAT5	0	1	0
eGFP	0	1	0

```
confusion_matrix(KMeans_predict, clean_data.labels)
```

4

	Ground Truth		
Predicted	CBP	KAT5	eGFP
CBP	769	772	1138
KAT5	1	0	0
eGFP	1082	784	454

```
confusion_matrix(KMeans_predict, clean_data.labels)
```

Figure 2 : Illustration of four different confusion matrices, run with KMeans with different initial conditions and looking for 3 clusters.

The classes are unordered, so the labels do not have real meaning, it is the distribution of the data that we want to underline : in matrices 1, 2 and 3, the data are almost all in one group, but in the matrix 4, the repartition is almost equal in two clusters.

	Ground Truth		
Predicted	CBP	KAT5	eGFP
CBP	1852	1556	1592
KAT5	0	0	0
eGFP	0	0	0

```
confusion_matrix(DBSCAN_pred, clean_data.labels)
```

Figure 3 : Confusion matrix from the density-based approach (DSCAN), all the data are in one cluster.
The classes are unordered, so they do not have a real meaning, it is the distribution of the data that we want to underline.

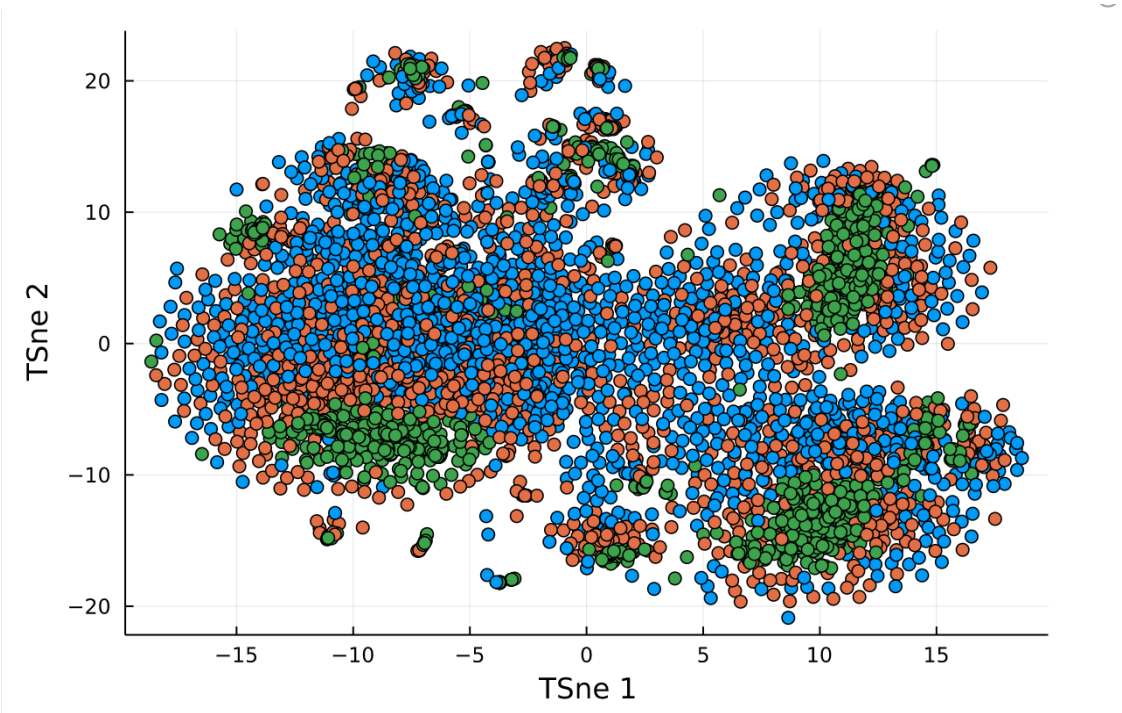

Figure 4 : Projection of the training data in a 2D plane with TSne method.

Luan Dinarica, Nicolas Vuillod December 2022
T30-1 :(