Machine learning - Project

Foivos Lambadaridis Kokkinakis

June 2024

0.1 Abstract and data classification

In this project, we attempt to classify events recorded in the Large Hadron Collider (LHC) and the Tevatron colliders as background noise or signals that could lead to discovering new particles. We are given a .csv file containing the necessary data, where the first column classifies the events as background noise:0 or as signals:1. For each event we are given two sets of quantities, the twenty-seven low-level quantities (from column 1 until column 27) and the seven high-level quantities (from column 22 to column 28).

Firstly we use two basic classifiers, the knn and the random forest classifier to achieve this. Then we will create an artificial neural network (ANN) for the same purpose and finally, these tree models will be compared based on their accuracy. Each one of these models will be made twice, one time with the low-level and the other with the high-level quantities as input. Finally, we will compare the accuracy of the results in both cases.

Our first step is to import our data from the .csv file using the pandas library. After doing so, we notice that the low-level quantity in column 17 is in a string format; thus, to complete computations with it, we turn it into the float format. After that, we separate our data into three categories

1. signal: Holds the decision on whether an event was a legitimate signal or just background noise
2. low_level: Holds the values of each one out of the 21 low-level quantities for each event (from column 1 until column 27).
3. high_level: Holds the values of each one out of the 7 high-level quantities for each event (from column 22 to column 28).

To be able to train each model we have to split our data into a training half that will be used to train our model and a testing half that will be used to assess our models' accuracy. We made the spit such that 80% of the data will be into the training half and 20% of it will be tested upon.

It is important to recognize that this problem is a binary classification problem, and thus, we have to use the appropriate algorithms that can be applied here.

0.2 Classifiers

0.2.1 Knn classifier

In both cases (when using high-level or low-level quantities) before applying the knn algorithm one must first scale their data appropriately. This is because the algorithm uses Euclidean distance to make each prediction. The only parameter we have to specify to create our classifier is the number of neighbors that are taken into account. We went with 10 since we have a fairly large amount of events in our data.

Low-level quantities

Using our scaled values of the training half of the low-level quantities and the corresponding values of the signal training half, we train our knn classifier. Using it we can make predictions by feeding the scaled values of the training half of the low-level quantities. We compare those guesses with the actual ones given to us from the original testing half of the low-level quantities. **The accuracy of the low-level knn model is 0.5721424,**

meaning that only 57,21% of the predictions of our model are correct, making it a bit better than guessing.

High-level quantities

We follow the same steps as before with the only difference being that we use the high-level instead of the low-scaled quantities (scaled where need be). **The accuracy of the high-level knn model is 0.6608369, meaning that 66,08% of the predictions of our model are correct, making it a lot better than the low-level knn model.**

0.2.2 Random forest classifier

To create our Random forest we need to specify the number of threes that are going to be created and the function that will judge the quality of each split. For both cases we choose fifty trees and the function entropy.

Low-level quantities

Using the values of the training half of the low-level quantities and the corresponding values of the signal training half, we train our random forest classifier. Using this classifier we can make predictions by feeding the values of the training half of the low-level quantities. We compare those guesses with the actual ones given to us from the original testing half of the low-level quantities. **The accuracy of the low-level random forest model is 0.6027482, meaning that 60,27% of the predictions of our model are correct. We see that the low-level random forest classifier's accuracy is bigger than the low-level knn's classifiers.**

High-level quantities

We follow the same steps as before with the only difference being that we use the high-level instead of the low-scaled quantities. **The accuracy of the high-level random forest model is 0.6826983, meaning that 68,27% of the predictions of our model are correct, making it a lot better than both the low-level knn and random forest models and a bit better than the knn high-level models. We can thus conclude that the random forest algorithm is better suited to this problem than the knn algorithm.**

0.3 Artificial Neural Networks

We have to create our neural network layer by layer. The first layer has as many nodes as different quantities, thus the low-level network has 21 and the high-level network has 7. The layer structure of both networks is the same from the 2nd layer and onwards, 30 nodes lead to 21 nodes which lead to one node, the output node. Since this is a binary classification problem firstly every layer uses the relu activation function except the output layer which uses the sigmoid function and secondly, the 'binary_crossentropy' loss function, the 'binary_accuracy' metric, and the 'adam' optimizer are chosen. Finally, we chose our network to be trained with a batch size of 15 and 150 epochs.

0.3.1 Low-level quantities

Using our scaled values of the training half of the low-level quantities and the corresponding values of the signal training half, we train our ANN. Using this network we can make predictions by feeding the scaled values of the training half of the low-level quantities.

These guesses are a number between zero and one. We assume that if a number is larger than 0.5 the final guess is 1 and if it isn't 0. We compare those guesses with the actual ones given to us from the original testing half of the low-level quantities. **The accuracy of the low-level artificial neural network is 0.5852592, meaning that only 58,52% of the predictions of our model are correct, making it slightly better than the low-level knn model and worse than the low-level random forest model.**

0.3.2 High-level quantities

We follow the same steps as before with the only difference being that we use the high-level instead of the low-scaled quantities (scaled where need be). **The accuracy of the high-level artificial neural network is 0.6826983, meaning that 68,26% of the predictions of our model are correct, making it identical to the high-level random forest model**

In conclusion, we can see that when a classifier or network is trained with the high-level quantities it is more capable of classifying events as background noise or signals (it is more accurate). It is important to note how the two best classifiers were equally accurate, the high-level random forest model, and the high-level ANN. This phenomenon could be explained by a theoretical limitation of the high-level quantities to 'give' more information to the classifiers so they can in turn make better decisions, but it also could be a structural limitation of our models. If we gave our random forest more trees or if we created a neural network with a different structure, we may have been able to create a more accurate prediction model.