A1. Re-run of Top-quark’s Accuracy Phase Solution

Author - Sharad Chitlangia, BITS Pilani, K. K. Birla Goa Campus, Goa, India

A2. Result in the competition

- Competition Name: TrackML
- Username: Sharad
- Leaderboard Place: 7th
- Leaderboard Score: 0.2918

A3. Summary

Top-quark’s approach is not a very heavily machine learning algorithm reliant model. It emphasizes on the usage of simple statistics and 3d geometry. The only part where a supervised learning algorithm is used is when pruning candidates for finding pairs and triples. The algorithm being a simple logistic regression model. After which the model relies on an unsupervised pipeline for track reconstruction.

A4. High - level Description

- The algorithm sets itself apart in the fact that it is very highly efficient and chooses to apply the logistic regression model at the right stage. This is because exaple prediction of the whole trajectory is not at all feasible at once and at the same time.
- The simplistic approach of defining different logistic regression models for different pairs of layers increases the pair prediction accuracy by upto 20%. This can be attributed to particles behaving differently at different distances from the origin.
- The algorithm consists of the following steps:
    a. Selection of promising pairs
    b. Extending pairs to triplet by search
    c. Extending triples to tracks by estimating helix
    d. Addition of duplicate hits to tracks
    e. Assign hits to tracks

A5. Scientific Details

Top-quark’s solution can be divided into 5 stages as follows:-

1. Selection of Promising Pairs of Hits
    - Logistic Regression models are trained for all pairs of hits on adjacent layers. The more the number of adjacent layers that are taken into consideration, the higher is the run-time and higher is the number of promising pairs that are selected initially. It should be kept in mind if more number of hits are considered initially, the whole trajectory reconstruction algorithm ahead has to search through more number of hits, hence increasing the time by a lot. I found the considering only 5 pairs of adjacent layers initially, gives an accuracy score of 60%, with running time of the complete algorithm being only 10 seconds. Whereas, when considering 50 adjacent layers, the accuracy reaches 67%, run-time being 38 seconds.
2. Extension of Pairs to Triples
    - Two promising pairs are selected and a line through them is extended. 10 closest points to this are set as triple candidates. The best candidates are selected by a logistic regression model running on features such as logarithm of radius of the helix that is created by considering the candidate triple.
3. Track Extension from Triples
    - A helix is fitted through the candidate triples and extended to the adjacent layers. The closest hit to the intersection is added onto the trajectory.
4. Addition of duplicate Hits
    - Duplicate hits that were ignored in the first step to avoid considering the same hits are now added to the trajectories if the hits coincide with the trajectory that is reconsidered.
5. Removal of Overlapping tracks
    - A scoring function based on probability of finding an outlier assuming that it is possible to model outliers given density of hits on a layer, is used to rank the trajectories. If a trajectory is selected, corresponding hits of that trajectory present in other trajectories are removed.
6. Changes from Accuracy Phase
    - There are no changes that were made from top-quarks solution. I was actually working on developing my own solution, but due to the time constraint of 600 seconds, my algorithm was scored 0. I did test few of his parameters that he chose which can be further optimised as I will describe in the section below.

A6. Interesting Findings

- The solution developed by the Top-quark team did not actually require training on all the train data. One event was sufficient enough to get accuracies of more than 67%.
- Consideration of less number of hits initially means a faster algorithm. The optimum value can be found out by plotting a graph of the optimum score v/s the number of adjacent layers considered initially.
- The score can be increased to I think around 90% by better choosing trajectories in the last step. This is primarily because the original solution was developed keeping in mind that there are outliers present whereas in the throughput phase the conditions are a bit different. The score can be finally also increased by imposing the condition of a trajectory having at least 8 hits.
- The most important features are indeed the x, y and z coordinates of the hits. From finding out which detector layer the hit belongs to, to what the radius of the helix will be if three hits are considered, all the calculations and prediction can be done using just the three coordinates without much loss in the score.
- Restarting from an algorithm which already exists serves as a jump-start even though the final score might be less. From a second year undergraduate student's perspective, developing a solution from scratch seems like a very hard task to achieve initially but understanding such an efficient solution definitely allows for a better final algorithm to develop, which I plan to do.

A7. Simple Features and Methods

A very high score can be achieved by finding the optimum amount of adjacent layers to be considered.

A8. Model Training and Execution Time

Training is only required for training the logistic regression models and requires about almost no time. Execution time is about 38 seconds on the Codalab Docker.

A9. Outlook

This method has a lot of scope to be integrated with the solutions by the top 2 candidates especially as they are combinatorial ones, their parameters can essentially be optimally be set by learning algorithms rather than unsupervised ones. In essence, Sergey's solution is somewhat similar as to tries to build the trajectories bit by bit(Pairs then triplet and so on).

A10. References

1. https://github.com/top-quarks/top-quarks
2. https://competitions.codalab.org/competitions/20112#learn_the_details
3. https://www.kaggle.com/c/trackml-particle-identification/