Particle Classification from Accelerator Data

Jack Agren and Max Ramsdell

Github Repo: https://github.com/3lbsofSalt/DataScience8/tree/master
Slideshow: Project 8 presentation

Introduction

Current experimental physics research relies on particle accelerators to understand the behavior of matter at the subatomic level. Particles are accelerated to near the speed of light, then smashed into each other, where the energy of the collision is converted into mass in the form of new particles. These particles are detected with hypersensitive detectors measuring things like charge and momentum of the particles that strike them. Cataloging these particles and understanding their properties can help us better understand the fundamental nature of matter and the origins of the universe.

Dataset

This dataset contains 1.2 million data points of 49 features with a corresponding label of Muon, Ghost, Pion, Proton, Kaon, and Electron. The features contain information about which sensors got hit, the momentum of the particle, the calculated trajectory and its uncertainty, and the calculated probabilities for each particle based on the measurements. It might sound like the calculated probabilities would simply give the answer each time, but this is probability based only on information from "sub detectors", which are part of the larger detectors. It doesn't incorporate the other information included in the dataset, which will likely influence the model's decision.

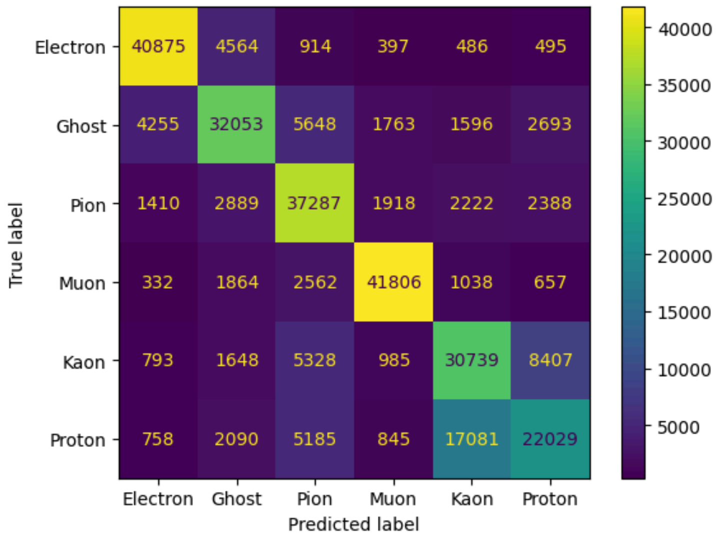
Analysis

In this analysis we used both a decision tree and a neural network to get different results. A decision tree will be suitable for this analysis because the goal is classification, which makes it a candidate. The features though are a little bit arcane and so a decision tree will help to inform us as to the most important features in the dataset. Understanding the nature of the features will better allow us to understand the nature of the particles that the dataset is concerned with. A neural network is capable of finding subtle patterns in the data without overfitting, which is useful with such small particles. However, it is less interpretable, and requires much more time to train than a decision tree.

Results

For the decision trees, several depths of trees were attempted. Depths below 3 were absolutely ineffective because they didn't even have enough leaf nodes to have a classification for each of the six types of particle. However, oddly enough, even the tree of depth 4 would not classify either the pion or the electron, telling me that the pion and electron are too similar to the other particles to be able to be classified on even a depth of 4 (with 8 leaf nodes to classify from). This seems to be confirmed as the algorithm continues to struggle with these classifications as the depth is increased.
We think that the best decision tree had a depth of 6. This decision tree did reasonably well at predicting the type of all particles with no f1 scores below .5, but excelled especially at predicting Muons and Electrons which were generally the best throughout the entirety of the models with f1 scores of generally around .8.

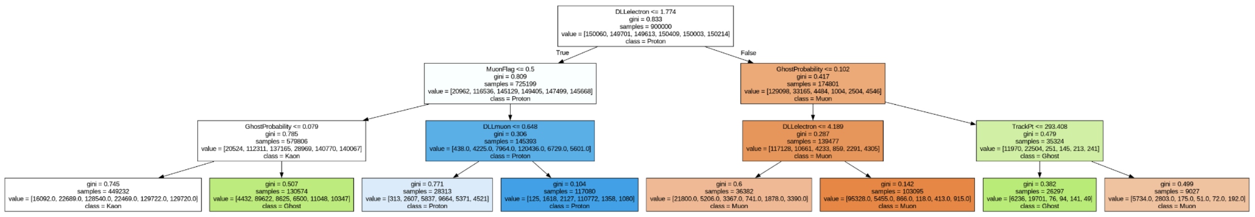
With our final neural net, we got an average f1 score of .7, which we were very happy with for having so many classes. In the confusion matrix below, we can see which particles were difficult for the model to pin down. For reference, each particle had about 48000 instances in the test dataset. The model generally had difficulty distinguishing between protons and Kaons. It also misclassified many of the samples as pions, while electrons and muons were easier to distinguish.



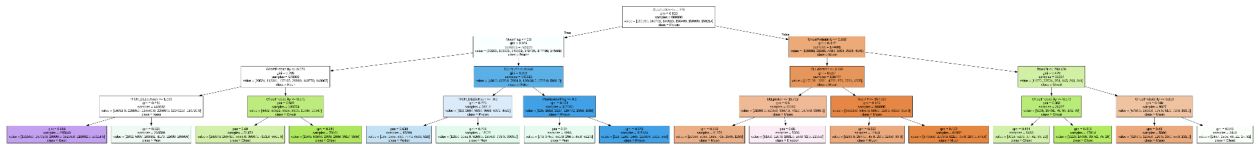
	Electron	Ghost	Pion	Muon	Kaon	Proton
Electron	40875	4564	914	397	486	495
Ghost	4255	32053	5648	1763	1596	2693
Pion	1410	2889	37287	1918	2222	2388
Muon	332	1864	2562	41806	1038	657
Kaon	793	1648	5328	985	30739	8407
Proton	758	2090	5185	845	17081	22029

Technical

Here are some of the visualizations for the different decision trees.

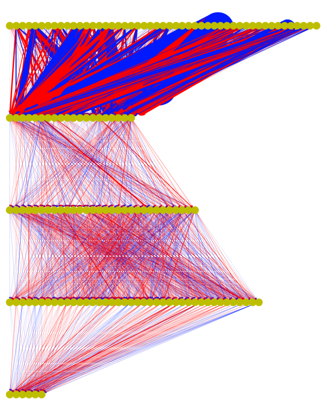

DLLelectron <= 1.774
gini = 0.833
samples = 900000
value = [150060, 149701, 149613, 150409, 150003, 150214]
class = Proton
True
False
MuonFlag <= 0.5
gini = 0.809
samples = 725199
value = [20962, 116536, 145129, 149405, 147499, 145668]
class = Proton
GhostProbability <= 0.102
gini = 0.417
samples = 174801
value = [129098, 33165, 4484, 1004, 2504, 4546]
class = Muon
GhostProbability <= 0.079
gini = 0.785
samples = 579806
value = [20524, 112311, 137165, 28969, 140770, 140067]
class = Kaon
DLLmuon <= 0.648
gini = 0.306
samples = 145393
value = [438.0, 4225.0, 7964.0, 120436.0, 6729.0, 5601.0]
class = Proton
DLLelectron <= 4.189
gini = 0.287
samples = 139477
value = [117128, 10661, 4233, 859, 2291, 4305]
class = Muon
TrackPt <= 293.408
gini = 0.479
samples = 35324
value = [11970, 22504, 251, 145, 213, 241]
class = Ghost
gini = 0.745
samples = 449232
value = [16092.0, 22689.0, 128540.0, 22469.0, 129722.0, 129720.0]
class = Kaon
gini = 0.507
samples = 130574
value = [4432, 89622, 8625, 6500, 11048, 10347]
class = Ghost
gini = 0.771
samples = 28313
value = [313, 2607, 5837, 9664, 5371, 4521]
class = Proton
gini = 0.104
samples = 117080
value = [125, 1618, 2127, 110772, 1358, 1080]
class = Proton
gini = 0.6
samples = 36382
value = [21800.0, 5206.0, 3367.0, 741.0, 1878.0, 3390.0]
class = Muon
gini = 0.142
samples = 103095
value = [95328.0, 5455.0, 866.0, 118.0, 413.0, 915.0]
class = Muon
gini = 0.382
samples = 26297
value = [6236, 19701, 76, 94, 141, 49]
class = Ghost
gini = 0.499
samples = 9027
value = [5734.0, 2803.0, 175.0, 51.0, 72.0, 192.0]
class = Muon

Depth of 3

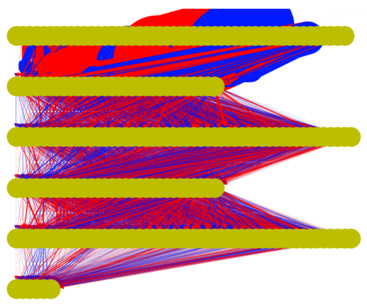


Depth of 4

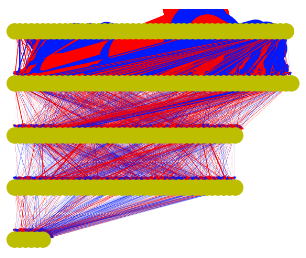
The important difference between the lower depth and the greater depth is that the lower depth has a greater bias. It acts more like a linear regression in that it is less susceptible to noise in the data, but that it doesn't model things perfectly. The greater depth is in danger of overfitting and being deceived by the noise data.
The decision tree always decided to predict on the DLLelectron feature for the first decision. This means that that was likely the most differentiating feature, as on the right you generally have predictions that move towards Muons and on the left (lower DLLelectron) the predictions move towards Protons. This makes sense because a high electronDLL indicates the probability that it is an electron and a quick wikipedia search will tell you that Muon's are very similar to electrons, while Protons could be considered as an opposite particle, as it has the opposite charge and is much larger.



[50, 20, 30, 40, 6]
Score:



[50, 30, 50, 30, 50, 6]



[50, 50, 40, 40, 6] (Final Model)

We tried other architectures, but these show the general ideas we tried. In general, having more nodes in the early hidden layers then decreasing the layer size. We also found that the model performance was extremely inconsistent until we gave it enough data (about 200k datapoints).