Identification of the correct hard-scatter vertex at the Large Hadron Collider

Pratik Kumar, Neel Mani Singh

pratikk@stanford.edu, neelmani@stanford.edu

Under the guidance of Prof. Ariel Schwartzman(sch@slac.stanford.edu)

I. Introduction

The Large Hadron collider(LHC) is the largest and most powerful particle collider ever built. Every 25ns, bunches of protons collide at the highest energies, producing 100 megapixel pictures of collision events that may contain signatures of new subatomic particles or forces. One key challenge for the analysis of LHC events is pile-up i.e. multiple overlapping proton-proton collisions in a single event picture. Under high luminosity conditions, each event picture at the LHC will contain, on average, 200 overlapping interactions. Out of which 60 interactions are reconstructed on an average. Only one interaction is of interest. All others are noise that contaminate the event. One important problem for physics event reconstruction is the identification of the correct hard-scatter primary vertex among all the overlapping pile-up interactions on an every by event basis. Selecting the correct interaction is important to correctly reconstruct the full event characteristics.

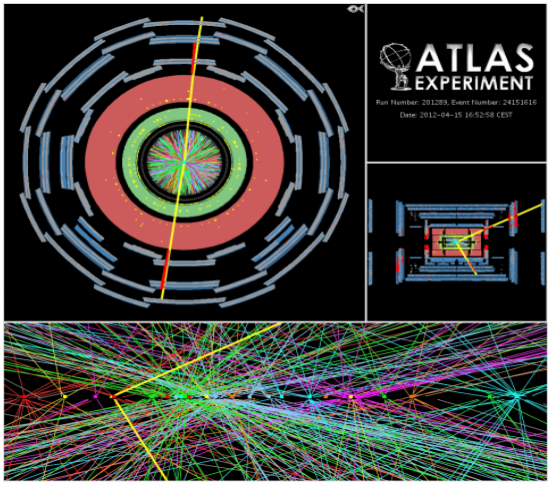

Figure 1[1]: An event with 25 interactions

The total momentum of a particle can be decomposed in two directions: longitudinal to the beam direction (z direction), and transverse (x-y). Since the beams move along z, when protons collide, one cannot know what is the fraction of the momentum (or speed) that the constituent quarks/gluons of the protons will have along the z axis. So, the resulting particles can have any value of momentum along z. On the other hand, we know that the quark/gluons do not have any speed (momentum) in x-y as they only move along z. So, the energy is conserved in the transverse plane. One starts with 0 transverse momentum, and ends with 0 transverse momentum. So, at LHC, the magnitude of interest is the transverse momentum: p_T.

Energy is approximately equal to momentum for particles that travel close to the speed of light. At LHC one utilizes transverse momentum as the measure of the total energy of the collision. In this text we will be using momentum and energy of particle interchangeably.

II. Problem - Input and Output

We propose to create a supervised machine learning algorithm to identify the correct primary vertex and distinguish it from all the overlapping pileup vertices. Our input to the problem will be attributes related to the proton-proton collision namely energies of the particles after collision, angles with the transverse plane at which particles go after collision, missing energy of a collision and sum of the energies of the particles created in a collision.

III. Current Vertex Selection Algorithm

The current technique[6] for the identification of the primary vertex selects the vertex with the highest total energy. The total energy is computed as the scalar sum of all particle tracks associated to the vertex. This method has a very poor performance when the number of pileup interactions is large, selecting the wrong vertex 40% of the time.

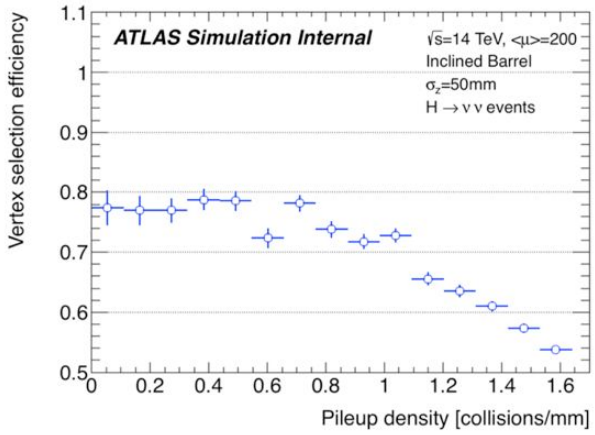

Figure 2: Vertex selection efficiency vs Pileup density of the current algorithm

IV. Data

The data consist of computer simulated events of Higgs bosons. Each event picture consists of a list of vertices (60 on average) and each vertex consists of a list of particle tracks. Each track is represented by a direction in 3D space, an origin (given by the vertex it belongs to), and its energy. These are used to generate features for our classifier.

V. Features

The information of all tracks in each vertex will be used to define features. We define the following features -

sumP_T each vertex is made of many tracks (the colored lines in Figure 1). Each track is a particle. Each particle has a p_T, which is the transverse momentum. It is basically the energy of the particle. This gives us the total scalar sum of track p_T of the vertex.

sumP_Tw is the weighted sum[2] of track p_T^2.

MET is the missing transverse energy. This is the vector sum of track p_T's of each vertex. In a pileup vertex, we expect energy to be conserved (in the transverse plane), so MET, on average, should be 0. For signal, since the Higgs boson decays to particles that escape the detector without being detected, there should be a significant imbalance of energy. So, MET should peak at a positive value.

pt1, pt2, pt3 is the transverse momentum of top 3 energetic particle track.

eta1, eta2, eta3 is the angle between top 3 energetic particle track and transverse (x-y) plane.

Each vertex has a truth label that says if the vertex is the correct hard-scatter primary vertex or if it is a pile-up vertex. Class 1 or positive class denotes class for primary vertex whereas Class 0 or negative class deontes pile-up background vertex.

VI. Unbalanced Data and Metrics

The data we have is inherently unbalanced because for each experiment 60 interaction, on an average, of particles are reconstructed and only one of those are point of interest. There are some techniques that can be used to handle the unbalanced data like -

Undersampling randomly downsamples the majority class. For example, Tomek[3] links are pairs of instances of opposite classes who are their own nearest neighbors. In other words, they are pairs of opposing instances that are very close together. Tomek's algorithm looks for such pairs and removes the majority instance of the pair.

Oversampling randomly replicates minority instances to increase their population. Another way of oversampling involves synthesizing new instances. The idea is to create new minority examples by interpolating between existing ones. For example, SMOTE[4] system creates new instances halfway between the instance and its neighbours.

Balanced Bagging[5] approach involves Bagging and Undersampling/Oversampling. The trick is to create balanced subset of training set for each model in the process of bagging.

VII. Methods

We used two different methods of evaluation. In this section and going forward we will use notation for data points as.

$x^{(i)}$: i^{th} *data point i.e.* $i \in \{1, 2, ..., m\}$

First method evaluates each vertex, i.e. $x^{(i)}$, and classifies it as primary vertex or pile-up background vertex one at time. This is a typical machine learning way of classifying input data point. We have used the following classification techniques:

Logistic Regression(LR) is a type of regression where response variable is dependent on linear combination of predictor variable. The hypothesis function is

$$h_\theta(x) = \sigma(\theta^T x)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function

Balanced Bagging with Logistic Regression(BBLR) Bagging or Bootstrap aggregating is ensemble meta-algorithm used in statistical classification and regression. Given a training set X bagging creates k new training set X_i by sampling from X uniformly with replacement. The k models are fitted using the above k bootstrap samples and some classification algorithm, combined by averaging the output (for regression) or voting (for classification). Balanced Bagging is a bagging technique which is same as bagging except in creating new training set from given training set. It creates bootstrap samples such that each class has approximately same number of data points.

Artificial Neural Network(NN) is a model of computation which consist of layers of artificial neuron and connections among them. We have used the simplest one where neuron in one layer is connected only to the next one. Artificial neuron is a computation unit like a LR where each one has some input features and output is defined by hypothesis function $h_\theta(x) = g(\theta^T x)$ where $g(z)$ is activation function. We have used ReLU for hidden layer and sigmoid for output layer.

Where ReLU is $g(z) = max(0, z)$

Second method is more closer to the experiment. In the above method, we treated each of the vertex as a single data input. But for our original problem, we need to select a vertex from a group of vertices in an experiment. We consider a single experiment as a single data now. We evaluate our model on vertices per experiment and chose the vertex that gives the highest probability.

$E = \{e^{(i)}\}$ *set of experiments*
$e^{(i)} = \{x^{(i)}\}$ *set of vertex in* i^{th} *experiment*
$v^{(i)}$: *vertex selected from* i^{th} *experiment*
$v^{(i)} = argmax_{x \in e^{(i)}} h_\theta(x)$

VIII. Metrics

The accuracy, precision and recall doesn't give correct estimate of the classifiers performance in case of unbalanced data because a small changes in classification of the majority class cause a big change in the precision and the accuracy. On the other hand, a big change in classification of the minority class does not have any major impact on accuracy and precision.

We use weighted F1-score, weighted by support and AUC_ROC score to evaluate our classifier for the first method.

We also used accuracy score based on vertex selection criteria. We calculate the fraction of experiment that picks the correct vertex. We also calculate this fraction as a function of pileup density.

IX. Experiments and Results

We started our experimentation with Logistic Regression with L2 regularization. The model achieved accuracy of 98.85%, precision of 82.65% and recall of 37.92%. The confusion matrix of this experiment revealed that most of the misclassification occurred in the positive class. Even though the accuracy is high, the precision and recall are low which is due to the class imbalance in our data.

Confusion Matrix Logistic Regression

	Predicted Class 0	Predicted Class 1
True Class 0	349366	465
True Class 1	3627	2215

Although the model was learning the negative class well but not the positive class. The data is unbalanced and logistic regression is not able to model the imbalanced data properly.

We tried Bagging with Logistic Regression. Number models used is 50 and each one was trained on 0.02% of data. The model achieved accuracy of 86.63%, precision of 0.18% and recall of 1.3%

Confusion Matrix Logistic Regression(L2)

	Predicted Class 0	Predicted Class 1
True Class 0	308065	41766
True Class 1	5766	76

We tried Balanced Bagging with Logistic Regression and L2 regularization. Number models used is 50 and each one was trained on 0.02% of data. The model achieved accuracy of 93.17%, precision of 17% and recall of 81.41%. Here the precision is disproportionately low because there is a small misclassification in the negative class which is quite high compared to the total number of positive class samples.

Confusion Matrix Balanced Bagging Logistic Regression

	Predicted Class 0	Predicted Class 1
True Class 0	326611	23220
True Class 1	1086	4756

We also tried Neural Networks and the results were similar to BBLR. The NN we used had one hidden layer with 100 neuron and α(regularization param) 0.001.

Confusion Matrix Neural Network

	Predicted Class 0	Predicted Class 1
True Class 0	329953	19878
True Class 1	1183	4659

Based on the above results, we see that the metrics like precision, accuracy does not work well for imbalanced data. Therefore we have used weighted F1-score and AUC_ROC score as described in the Metrics section.

Metrics for our Methods

Models	**Metrics**			
	F1 (train)	**F1 (test)**	**AUC_ROC (train)**	**AUC_ROC (test)**
LR	98.62	98.63	68.90	68.89
BBLR	96.37	96.32	87.49	87.39
NN	95.79	95.82	87.17	87.03

Based on AUC_ROC score, we see that BBLR performs the best as it takes into account the unbalanced nature of the data. BBLR creates multiple model on balanced subset of data. NN also works similar to BBLR. LR does not have a good performance because of unbalanced nature of data.

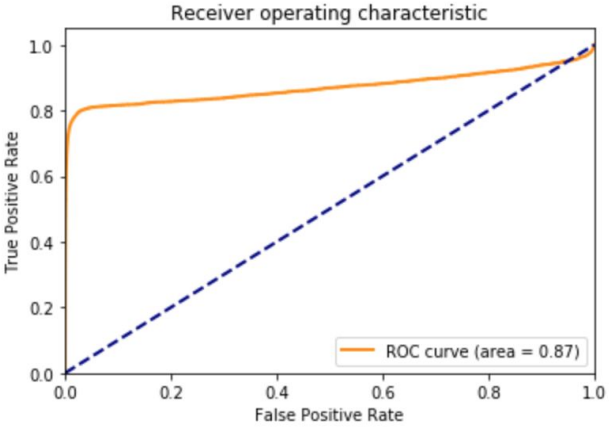

Figure 3: ROC Curve for BBLR

Let us now move to the second method of evaluation i.e. vertex selection. **Vertex Selection Efficiency** is the fraction of experiment in which correct vertex was classified as primary vertex.

Comparison for Vertex Selection Efficiency

Models	Metrics
	Vertex Selection Efficiency
LR	40.41
BBLR	78.24
NN	63.00
Current Method	57.90

Based on the results, we see that LR performs worse than the current technique. BBLR and NN performs better than the current technique with BBLR having an edge over NN. We also see the variation of vertex selection efficiency with pileup density in Figure 4 below. The result are similar to the overall vertex selection efficiency.

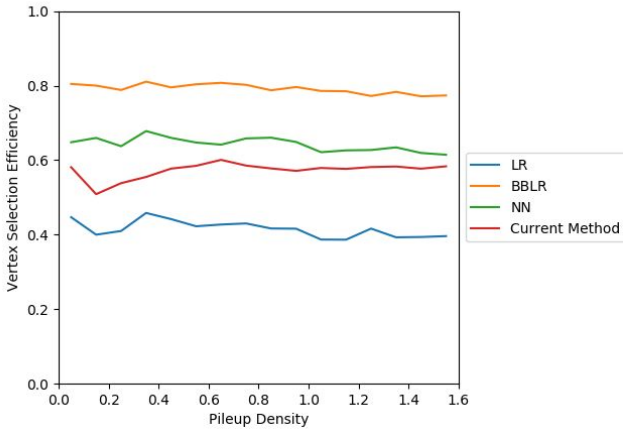

Figure 4: Comparison of Vertex selection efficiency vs Pileup density for different approach

Feature Selection: Since the total number of different combination of features are 512 and each one was not taking very long train we did exhaustive search on feature space for BBLR. We found that the subset of eta1, eta2, pt1, pt2 gave the best result.

Confusion Matrix Balanced Bagging Logistic Regression

	Predicted Class 0	Predicted Class 1
True Class 0	334380	15451
True Class 1	1072	4770

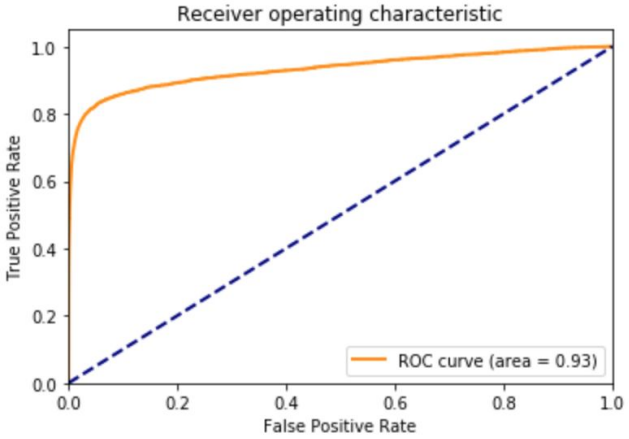

Figure 5: BBLR ROC Curve with subset feature

X. Future Work

We were expecting a better result with neural network because of its capability to learn complex nature of data. We would like to tune various parameters like regularization coefficient, number of hidden layers, neurons in the hidden layers, etc. We would also like explore other types of NN like Convolution NN, LSTM.

We would like to explore more on feature selection to see how it perform with vertex selection algorithm.

We tried various models like gaussian naive bayes, linear GDA, quadratic GDA, k-nearest neighbour classifiers. We did not get time to analyze each of them which we would like to do.

Contributions

Both of us worked together collaboratively most of the time. So there is no clear distinction in the contributions.

References

[1] https://atlas.cern/sites/atlas-public.web.cern.ch/files/field/image/2012_highPileup.png.

[2] Taken from slides of Prof. Ariel Schwartzman.

[3] Debashree Devi, Saroj kr. Biswas and Biswajit Purkayastha, "*Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance*", 2016.

[4] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall and W. Philip Kegelmeyer, "*SMOTE: Synthetic Minority Over-sampling Technique*", CoRR, 2011.

[5] https://svds.com/learning-imbalanced-classes/.

[6] Strandlie, Are et al. "Track and vertex reconstruction: From classical to adaptive methods" Rev.Mod.Phys. 82 (2010) 1419-1458