Sydney Erickson

Advisor: Professor Junjie Zhu

Special thanks to: Yao Fu, Miaoran Lu, Prachi Atmasiddha

Improving Signal Separation of Longitudinally Polarized WZ Boson Events

Introduction

There are four fundamental interactions that govern the universe: the strong force, the weak force, the electromagnetic force, and the gravitational force. The interaction between two elementary particles is due to the exchange of force carrier particles, also called gauge bosons. In the Standard Model (SM) of particle physics, the electromagnetic force is mediated by photons, the weak force by W and Z bosons, and the strong force by gluons.

Just as the electric force and the magnetic force both derive from electromagnetism, the electromagnetic and weak forces are derived from the electroweak force in the SM. Even though these two forces appear different at low energies, they are two different aspects of the same force and merge into a single force above the unification energy of ~250 GeV. However, it is strange that even though they are all force carriers for the electroweak force, photons are massless, while the W and Z bosons are massive, with a mass of ~90 GeV. Furthermore, all elementary particles and force carriers are required to be massless in the SM due to intrinsic gauge symmetries, contradicting experimental observations that most particles have non-zero masses. Thus the acquisition of mass of the W and Z bosons is quite an interesting phenomenon to study.

The masses of the W and Z bosons are acquired through the Higgs mechanism as a part of electroweak symmetry breaking (EWSB). A Higgs field is introduced to break the electroweak symmetry, resulting in three massless Goldstone bosons, and one massive Higgs boson. Massless W and Z bosons absorb the Goldstone bosons as their longitudinal components. Massless particles have only two transverse polarizations, while massive particles can also have a third, longitudinal polarization. Since the bosons acquire a longitudinal component, this results in massive W+, W-, and Z bosons. In order to probe the mechanisms of EWSB, it is vital to study the interactions of longitudinally polarized vector bosons. Thus, the study of longitudinal-longitudinal diboson events is of great interest.

To study elementary particles and their interactions, we need to reach extremely high energies. The best way to achieve this is to accelerate particles to high speeds and collide them together, creating a shower of elementary particles in their wake. At present, the Large Hadron Collider (LHC) at CERN is the world's highest energy particle collider and serves as the best place to study these interactions. Two proton beams are accelerated to collide at a center-of-mass energy of 13-14 TeV. Two general-purpose particle detectors, ATLAS (A Toroidal LHC Apparatus) and CMS (Compact Muon Solenoid), act like cameras to record all particles produced during each collision. There are roughly 40 million proton-proton collisions every second, and it is impossible to store

information about each of these events. Only a small fraction of collisions will have two protons experiencing inelastic scattering, thus producing interesting detector signatures. A trigger system analyzes preliminary data in real time to filter out which events are of high interest, and which events are to be thrown out. This leaves about 1000 collisions per second, which still results in roughly 3 Gigabyte of data collected for each second the LHC is running.

The ATLAS detector is a multi-purpose particle physics detector with cylindrical geometry. A right-handed coordinate system with its origin at the nominal interaction point (IP) in the center of the detector is used. The x-axis points from the IP to the center of the LHC ring, the y-axis points upward, and the z-axis is along one of the proton beam directions. Cylindrical coordinates (r, ϕ) are used in the transverse plane with ϕ being the azimuthal angle around the beam pipe. The pseudorapidity η is defined in terms of the polar angle θ as $\eta=-\ln\tan(\theta/2)$. Transverse momentum (p_T) is defined relative to the beam axis and is calculated as $p_T=p\times\sin\theta$ where p is the momentum. The ATLAS detector consists of an inner tracker (ID) surrounded by a superconducting solenoid providing a 2 T axial magnetic field, sampling electromagnetic (EM) and hadronic (HAD) calorimeters using liquid argon as active material and lead/copper/tungsten as absorber material, and a muon spectrometer (MS) based on one barrel and two endcap air-core toroidal superconducting magnets providing a field of approximately 0.5 T and 1 T in the central and endcap regions, respectively. The ID covers the region $|\eta|<2.5$, the EM (HAD) calorimeter covers the region $|\eta|<3.2$ ($|\eta|<4.9$), and the MS covers $|\eta|<2.7$.

The diboson production process, pp → WZ, is used for the first time to study longitudinal-longitudinal (LL) WZ interactions. The fully-leptonic WZ decay channels, where both W and Z bosons decay leptonically (W→lν and Z→ll with l=e, μ), are used. The final event signature (pp → WZ →lνll) is three charged leptons (electron or muon) and one neutrino. Electrons are reconstructed from energy deposits in the EM calorimeter associated with tracks found in the ID. Muons are reconstructed by combining tracks reconstructed in the ID to tracks or track segments found in the MS. The detection of the neutrino is based on the missing transverse energy (MET) calculated using reconstructed physics objects and unclustered energies found in the calorimeters. Figure 1 shows the diagram of WZ→lνll in the WZ rest frame.

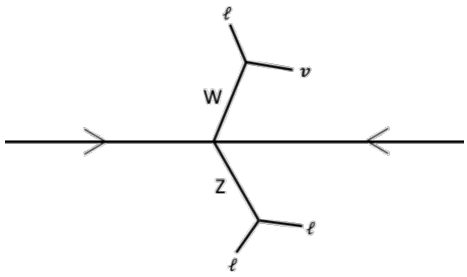

Figure 1: Diagram of WZ→lνll decay in the WZ rest frame

The production of pp → WZ occurs in three different channels: s-, u-, and t-, as shown in the Feynman diagrams in Figure 2.

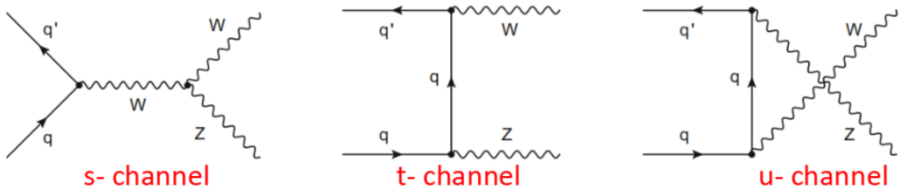

Figure 2: Feynman diagrams for pp→WZ production in s-, u-, and t- channels

One of the biggest challenges of this study is the separation of a miniscule signal from background processes. We are only interested in events in the s- channel, as this is the channel where W and Z bosons directly interact with each other. The s- channel makes up only ~10% of all WZ → lvll events. Additionally, there are three other polarizations of the W and Z bosons that can occur: longitudinal-transversal (LT), transversal-longitudinal (TL), and transversal-transversal (TT). LL polarization only occurs in ~6% of events. These two restraints result in a signal to background ratio of about 0.006. Due to its low cross section, LL WZ events have never been directly observed before. This study aims to develop techniques to improve separation of the LL WZ signal from background to make observation of these events possible.

The standard procedure to filter out background events is to apply a series of simple selection cuts on the events. There are many kinematic properties of different particles that could be used in addition to existing selection cuts due to the highly complex nature of collisions at the LHC. Thus, it is advantageous to develop multivariate techniques in order to combine the separation powers of multiple kinematic distributions at once. In this study, I show that multivariate methods can be used in tandem with selection cuts to improve the LL polarization fraction further.

Outline of Questions Addressed

The biggest challenge in this research is finding a way to overcome the massive background present in the WZ→lvll events when looking for longitudinally-polarized WZ interactions. The most significant background is from other polarizations: LT, TL, and TT. In this study, I investigate whether using a multivariate approach in tandem with a series of selection cuts will provide better performance than existing techniques. I look to find which kinematic variables serve as the best discriminants between LL and background polarizations. I also investigate which machine learning techniques, such as early stopping rounds and grid search over hyperparameters, can best improve model performance for this classification task. The most important result is the final model performance and understanding how the output from the final model can be used to improve signal separation of LL events.

Methods

The dataset for this study is simulated using Monte Carlo methods. First, WZ events with different polarizations were generated using the MadGraph event generator. For this sample, we used leading-order calculations in Quantum Chromodynamics (QCD) only. To emulate next-to-leading order effects, we generated WZ+0 and WZ+1 jet events. Then, we used Pythia and applied the CKKW-L merging algorithm to combine the two samples. MadGraph is then used to decay polarized W and Z bosons to lepton-neutrino or lepton-lepton pairs, where the lepton is either an electron or muon. This results in a final sample of WZ→lvll events. The datasets are stored in

ROOT tree files and then parsed into .csv files for the multivariate study. Sample weights are stored for each event such that when the weights are applied, the events are normalized to their cross sections times the integrated luminosity.

The first method applied to separate out signal events is filtering of events using a series of selection cuts. Events that satisfy all the initial cut requirements make up the LL-enhanced fiducial region, also called the signal region. The first cut is a group of three cuts on lepton properties, which must be satisfied by all three leptons from the WZ decay. The three lepton cuts require: transverse momentum (p_T) > 25 GeV, absolute value of eta ($|\eta|$) < 2.5, and missing transverse energy (MET) > 25 GeV. The second cut requires p_T of the Z boson to be greater than 200 GeV. The third cut requires p_T of the WZ system to be less than 70 GeV.

When analyzing the effectiveness of these selection cuts, we look at the LL polarization fraction. The LL polarization fraction is the number of LL events divided by the number of total events. We aim to maximize the LL polarization fraction.

$$\text{LL polarization fraction} = \frac{LL}{LL+LT+TL+TT}$$

Applying the selection cuts described above on the simulated dataset, we were able to increase the LL polarization fraction from 0.058 to 0.257, with results shown in the table below.

Region	LL Events	LT Events	TL Events	TT Events	LL Fraction
No Cuts	1413	3752	3792	15174	0.058
LL-Enhanced	43	9	8	107	0.257

Table 1: Number of events for each polarization state and LL polarization fraction before and after selection cuts are applied.

While this is a significant increase, we aim to further improve this LL polarization fraction. To do so, we developed a multivariate method and applied it to simulated in the LL-enhanced region.

We approached this problem as a binary classification problem. LL events were assigned a label of 1, while other polarizations were given a label of 0. Given a set of multiple variables describing an event, a machine learning model is able to predict whether the event has a label of 1 or 0.

The first step in this process was to identify which set of variables to use as features for classification. After analyzing the distributions of several variables, we chose 6 variables to use as features. These features all had distributions that showed good separation power between LL events and background. The distributions are included below, along with a table giving the definition of each variable that was chosen.

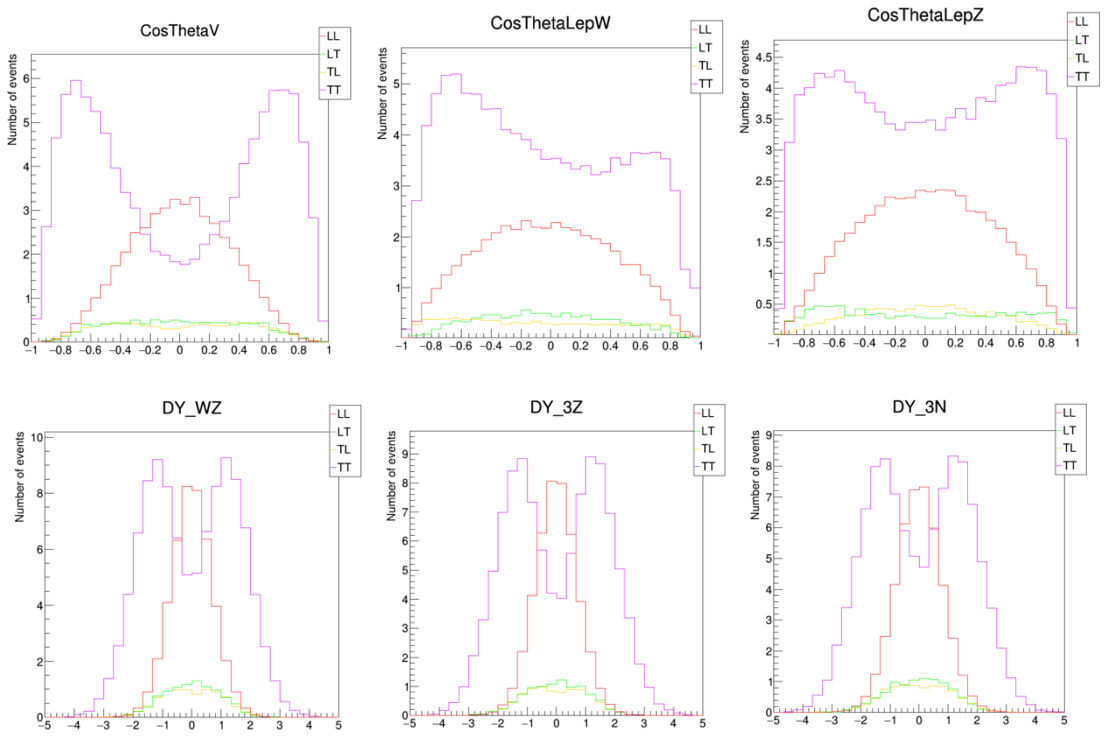

Figure 3: Distributions for the 6 kinematic variables chosen as features. Each polarization state is plotted separately. Note the LL polarization is in red, LT in green, TL, in orange, and TT in purple. We look for intervals in the distribution where the LL contribution increases, and the background contribution decreases simultaneously.

$\cos\theta_{W}$	W boson direction in the WZ rest frame w.r.t. the lab Z axis
$\cos\theta_{lepW}$	Direction of the lepton from W decay in the W rest frame w.r.t the W direction in the lab frame
$\cos\theta_{lepZ}$	Direction of the negatively charged lepton from Z decay in the Z rest frame w.r.t. the Z direction in the lab frame
ΔY(WZ)	Rapidity difference between the W and Z bosons
ΔY(3N)	Rapidity difference between the lepton from W decay and the negatively charged lepton from Z decay
ΔY(3Z)	Rapidity difference between the lepton from W decay and the Z boson

Table 2: List of features chosen for the multivariate study and their definitions.

The cosine variables are the cosine of the directions of different particles in the collision in particular rest frames. For example, θ_W is shown below in red on the WZ→lvll collision diagram.

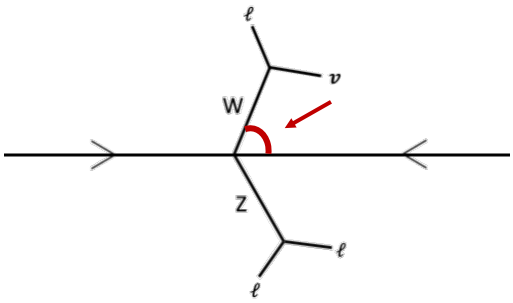

Figure 4: Diagram showing W boson direction in the WZ rest frame with respect to the lab Z axis.

Rapidity can be loosely thought of as the direction that a particle travels with respect to the particle beam from which a collision originates. If the rapidity is close to zero, the particle travels near perpendicular to the beam, while if the rapidity is large, the particle travels along the direction of the beam.

$$Y = \frac{1}{2}\ln\left(\frac{E+p_z c}{E-p_z c}\right)$$

Using these 6 features, the next step was to develop the binary classification model. For this study, we used the XGBoost boosted decision tree model. XGBoost was originally developed by Tianqi Chen for the Higgs Machine Learning competition, so it is particularly well suited for classification of events from the LHC.

XGBoost trains multiple trees and combines the prediction scores from all the trees to generate a final classification for each event. Training a forest of trees rather than a single instance makes the model more robust. The tree that XGBoost employs is a classification and regression tree (CART). CARTs assign a probability score to each leaf node in the tree rather than a classification label. XGBoost maps the scores from the forest of CARTs to {0,1} binary output.

To build the forest, we look for trees that minimize the objective function, obj(θ). The objective function is a mathematical definition of how well the trees classify events. The parameters of the objective function, θ, describe the structure of all the trees. Given a set of θ, we can build the trees and map each event to its prediction. The objective function optimizes two facets of the model: performance and complexity. The loss function, $l(y, \hat{y})$, measures the classification error. $l(y, \hat{y})$ compares y, the labels provided for each event, and $\hat{y}$, the labels predicted by the model. For this study, we use the binary logistic loss function. The regularization term, $\Omega(f)$, measures complexity. The input for the regularization term, f, is a function describing the structure of the CARTs. Given n is the number of training events, and K is the number of trees generated, we can define the objective function:

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Given a tree structure, we can directly solve for the leaf node weights, w, that optimize the objective function. XGBoost exploits this closed form solution to generate the forest. The trees are built individually by looking at one split at a time. The model attempts to separate a leaf node into two branches by making a new split on a certain feature. Then, it calculates w given the new structure. Given the new leaf weights, the score of the new structure is compared to the score of the previous structure. If the improvement of the score is above a certain threshold, γ, that cut is deemed optimal. In this case, a learning rate, η, is multiplied with the new feature weights to shrink their magnitude. This helps prevent the model from fitting too fast. Then, the algorithm starts searching for a new split. Otherwise, if the improvement of score does not pass the threshold, the branches are removed, and the model searches for a different split to make on the leaf node.

We used several methods to improve upon the performance of the default XGBoost model. First, we re-weighted training events. Then, we used early stopping of boosting rounds to ensure the model will always pick the optimal number of trees in the forest and avoid overfitting. Finally, we performed a grid search over different model parameters using 5-fold cross validation.

The metric we optimized when testing these methods was the AUROC metric. AUROC measures the area under the ROC curve. The receiver operating characteristic curve (ROC curve) plots the performance of the model to compare the tradeoff between the false positive rate (FPR) and the true positive rate (TPR). Using the area under this curve is a good metric for this study because it considers both true positives and false positives. An ideal model passes through the point (FPR=0, TPR=1) and has an area under the curve (AUROC) score of 1. The worst possible performance occurs when the model takes a random guess, which results in an AUROC score of 0.5. We hope to generate a model that approaches the perfect classification curve; thus, we look to maximize the AUROC score towards 1.

Before training events are passed to the classifier, the sample weights are re-normalized such that there is a 50% contribution from signal events and a 50% contribution from background events. This method is used to avoid overfitting. We want the classifier to train on a balanced dataset as opposed to the original distribution so that fluctuations in the relatively smaller signal portion don't dominate the trends that the classifier learns.

Another method we use to avoid overfitting is early stopping of boosting rounds. In each boosting round, a new tree is added to the ensemble. If too many trees are added, the model overfits to the noise in the training sample. We can measure when overfitting occurs by measuring performance after each boosting round on a held-out validation set. We reserve a portion of the training set as the validation set for this method. When the AUROC score of the classifier tested on the validation set starts to decrease, while the AUROC score tested on the training set continues to increase, we observe that the classifier has started to learn patterns from fluctuations in the training set rather than trends from the entire dataset. After each tree is built, we check the AUROC score on the validation set. If the score hasn't increased in 10 boosting rounds, we know that the model has become overfitted, and we remove the 10 additional trees that did not increase the score to arrive at the optimal forest size.

We performed a grid search using 5-fold cross validation and incorporating early stopping rounds to identify which settings of hyperparameters are the best for this classification problem. Two parameters described above, the threshold for adding a new branch, γ, and the learning rate, η, are user defined parameters, also called hyperparameters. Other hyperparameters include the maximum depth of each tree, max_depth, the fraction of features used for splits in each tree, colsample_bytree, and the minimum number of training events required to be in each node, min_child_weight. See the appendix for documentation of the full output from grid search. The results from the grid search led us to set max_depth = 3, colsample_bytree = 1, gamma = 0, min_child_weight = 5.

After implementing these methods, we tested to see how successful these methods are at improving performance from the default XGBoost settings. The table below shows the average AUROC score over 5 separate iterations of training and testing. We used multiple iterations of training and testing to account for differences in performance due to the random selection of subsets for training and testing.

	Avg. AUROC
Default Parameters	0.8250
Default Parameters with Early Stopping Rounds	0.8276
Grid Search Parameters with Early Stopping Rounds	0.8272

Table 3: Average AUROC score across 5 iterations for each training method.

Note that using the grid search parameters did not result in improved performance. This might be because the grid search was affected by noise from smaller subsections of the sample. Also note that XGBoost was designed for a particle physics machine learning competition, so it is reasonable to expect the default parameters to work well for this classification problem. For our final evaluation of model performance, we will use XGBoost's default parameter settings and early stopping of boosting rounds while training.

After implementing a technique using machine learning, it is paramount to understand what choices the model makes to reach its predictions. Decision trees are fairly interpretable, as we can analyze which features were split on the most and which of those splits provided the largest gain of information. Using these types of metrics, we can analyze which features were the most important in deciding the classification of events. The SHAP feature importance metric was found to be the best choice to analyze the features used in XGBoost training. SHAP metric results are included in the results section of this paper.

Results

A multivariate technique was employed to improve the separation of LL events from background polarizations: LT, TL, and TT. Before constructing a machine learning model, distributions of kinematic variables were analyzed to identify features with the best separation power. Six features were identified: $\cos\theta_{W}$, $\cos\theta_{lepW}$, $\cos\theta_{lepZ}$, $\Delta Y(WZ)$, $\Delta Y(3N)$, and $\Delta Y(3Z)$. XGBoost was used to perform binary classification on the polarization state of WZ→lvll events with these six variables used as features. Through this study, we learned which hyperparameter settings of XGBoost were

best suited for this classification task and which features the model valued the most in generating classifications. The resulting XGBoost model provided reasonable performance separating LL events from background polarization. Finally, we demonstrate how the probability scores output from XGBoost can be used to further separate LL events from background polarizations.

To test the model, we used a held-out test set containing events that were not used for any part of the model's training procedure. There are three important metrics: AUROC score, SHAP feature importance scores, and the LL polarization fraction in the LL-enhanced region after a final selection cut is applied using the multivariate output. We reserve only 20% of the dataset for testing, and the selection of the events for the test set is at random. Thus, the output metrics are all averaged over multiple runs of the algorithm to account for fluctuations in the distribution of the test set. Only plots from the first run are shown, as the major trends in the results remain the same across different runs.

AUROC

To evaluate the overall model performance, we measure the AUROC metric on a held-out test set. Below, we plot the ROC curve for our model from the first run and provide a table of results from all 5 runs.

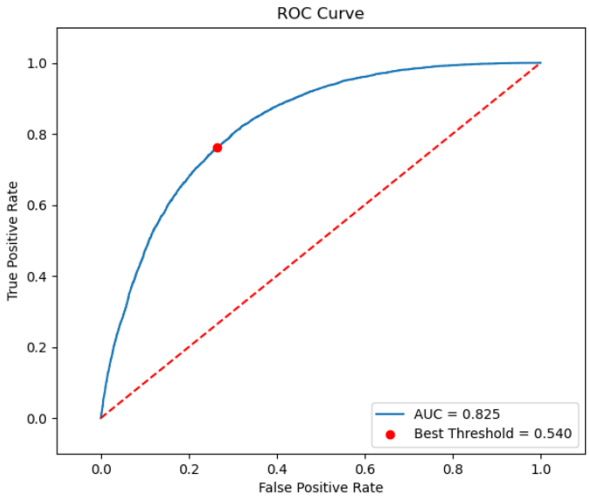

Figure 5: ROC curve plotted in blue for the 1st run of final output. The red dotted line plots the result for a random classifier, with AUROC = 0.5.

	AUROC
Run 1	0.825
Run 2	0.828
Run 3	0.826
Run 4	0.826
Run 5	0.831
Average	**0.827**

Table 3: Calculation of average AUROC score across 5 final runs.

Note how the ROC curve pulls away from the random classification curve and towards ideal performance, indicating that the classifier has some degree of success. The average AUROC score is 0.827. This means our model can provide reasonably accurate classification, as the score is much higher than 0.5.

SHAP Feature Importance

The SHAP metric measures how influential a feature was in the classification of each individual event. Thus, an average of the SHAP score across all events tells us how important each feature was. The plot shown below is for the first run. Note that while there are fluctuations in the ordering of importance in reach run, the importance of ΔY(3Z) remains the largest by a significant margin in all 5 runs.

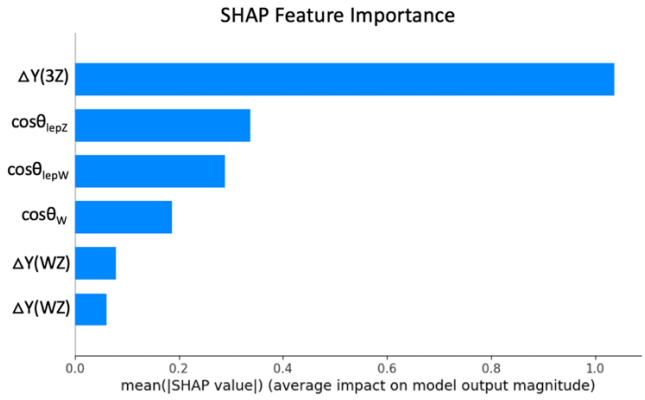

Figure 6: Bar plot of average SHAP metric magnitude for each feature.

It is somewhat unexpected that ΔY(3Z) has such a large impact on the classification of events compared to other variables that appear to have similar separation power given the distribution plots for the four polarization states (Figure 3). More analysis needs to be done to verify whether this result is due to fluctuations in this particular dataset or some kind of physical property that can be exploited to separate events.

LL Polarization Fraction

Given the model's AUROC score, we know that XGBoost is able to classify signal events with reasonable accuracy. Thus, we hope to use the output from XGBoost to help filter out more background events. We pass the test event set to the trained XGBoost model to generate probability scores for each event. The probability scores range between 0 and 1 and measure the probability that the label for an event is 1, which corresponds to LL polarization. Then, a final selection cut is taken, requiring the probability to be above a certain threshold (usually around 0.5).

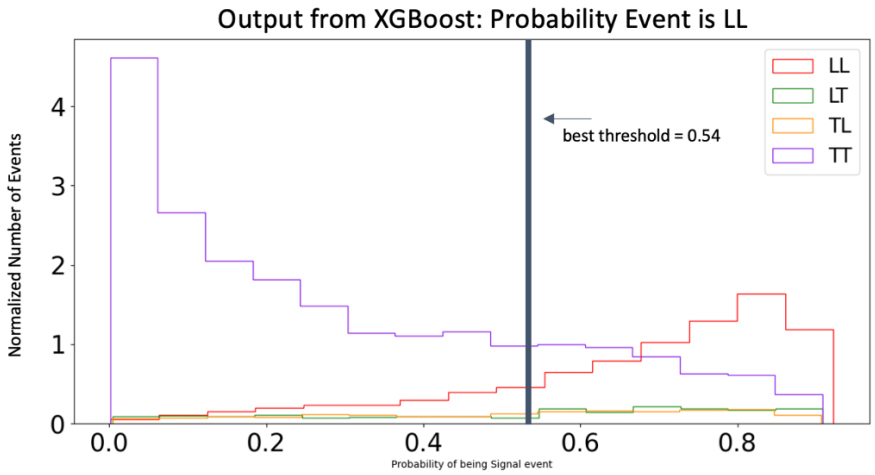

Figure 7: Histograms of the test events' probability scores are plotted for each polarization state. Events with a probability score lower than 0.54 are discarded in order to generate a final signal region.

	LL-Enhanced LL Fraction	**LL Fraction after requiring probability > 0.54**
Run 1	0.257	0.506
Run 2	0.260	0.545
Run 3	0.260	0.515
Run 4	0.254	0.500
Run 5	0.259	0.503
Average	**0.258**	**0.513**

Table 4: Calculation of average LL polarization fraction from each test set. One measurement is from the LL-enhanced region and the other measurement is made after the final selection cut on the XGBoost output probability is taken.

Adding the cut from the multivariate technique improves the LL polarization fraction from 0.258 to 0.513. This is an example of how the output from the pre-trained XGBoost classifier trained can be used on previously unseen events to further improve the signal to background ratio.

Discussion

A multivariate method using XGBoost shows promise for improving the separation of LL WZ boson events from background polarization states. Using truth-level samples, we were able to improve the LL polarization fraction in the LL-enhanced region from 0.258 to 0.513 using output from a pre-trained classifier. In the future, a multivariate method using boosted decision trees will be integrated into the existing analysis framework for longitudinal-longitudinal WZ interactions. With the added separation power from a multivariate technique, we hope that the separation methods will be robust enough to provide the capability to extract the W_LZ_L fraction from data for the first time.

Extracting the cross section of these events from data will allow us to compare this measurement to the prediction from the Standard Model. Precision measurements like this can help us identify whether our current model is accurate. This means we can either confirm what is predicted, or more rarely, get an unexpected result that may point to new physics. Ultimately, this measurement helps us better understand the EWSB sector of the Standard Model.

To validate the findings of this study, the next step will be to test this machine learning technique on other samples. We need to determine whether the classifier has learned trends from the physics defining the distribution, or from fluctuations in this sample that won't generalize to the overall dataset. One dataset that will likely be used next is a sample generated using next-to-leading order calculations, as opposed to this sample which uses leading-order calculations.

Future work on this study also includes writing code to link together the current Python implementation of the multivariate technique and the pre-existing C++ analysis framework for the WZ→lvll study. We also need to test whether a model trained on LT, TL, TT polarization backgrounds will be robust to other background processes, such as Z+ jets and $t\bar{t}$ processes, which also significantly contribute to the event set.

Sources

Brownlee, J. “Avoid Overfitting by Early Stopping with XGBoost in Python.” Accessed: https://machinelearningmastery.com/avoid-overfitting-by-early-stopping-with-xgboost-in-python/

Brownlee, J. “How to Develop Your First XGBoost Model in Python.” Accessed: https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/

Chen, T., Guestrin, C. “XGBoost: A Scalable Tree Boosting System.” Accessed: https://arxiv.org/abs/1603.02754

Daw, E. “Lecture 7: Rapidity and Pseudorapidity.” Accessed: http://www.hep.shef.ac.uk/edaw/PHY206/Site/2012_course_files/phy206rlec7.pdf

“Introduction to Boosted Trees.” Accessed: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Lundberg, S. “Interpretable Machine Learning with XGBoost.” Accessed: https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27

Tanedo, P. “Who ate the Higgs?” Accessed: https://www.quantumdiaries.org/2011/10/10/who-ate-the-higgs/

Tanedo, P. “Why do we expect a Higgs boson? Part I: Electroweak Symmetry Breaking.” Accessed: https://www.quantumdiaries.org/2011/11/21/why-do-we-expect-a-higgs-boson-part-i-electroweak-symmetry-breaking/

Vidal, X., Manzano, R. “LHC Data Analysis.” Accessed: https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.lhc_data_analysis

Appendix: Grid Search Results

Initial grid search:

max_depth = {3, 4, 5, 6}, gamma = {0, 0.1, 1, 10}, colsample_bytree = {0.$\bar{6}$, 1}

Parameter Settings	**Avg. AUROC Score Across 5-fold CV**
max_depth = 3, gamma = 0, colsample-bytree = 0.$\bar{6}$	0.82799
max_depth = 3, gamma = 0, colsample-bytree = 1	0.82825
max_depth = 3, gamma = 0.1, colsample-bytree = 0.$\bar{6}$	0.82801
max_depth = 3, gamma = 0.1, colsample-bytree = 1	0.82832
max_depth = 3, gamma = 1, colsample-bytree = 0.$\bar{6}$	0.82803
max_depth = 3, gamma = 1, colsample-bytree = 1	0.82817
max_depth = 3, gamma = 10, colsample-bytree = 0.$\bar{6}$	0.82757
max_depth = 3, gamma = 10, colsample-bytree = 1	0.82747
max_depth = 4, gamma = 0, colsample-bytree = 0.$\bar{6}$	0.82751
max_depth = 4, gamma = 0, colsample-bytree = 1	0.82770
max_depth = 4, gamma = 0.1, colsample-bytree = 0.$\bar{6}$	0.82753
max_depth = 4, gamma = 0.1, colsample-bytree = 1	0.82758
max_depth = 4, gamma = 1, colsample-bytree = 0.$\bar{6}$	0.82774
max_depth = 4, gamma = 1, colsample-bytree = 1	0.82768
max_depth = 4, gamma = 10, colsample-bytree = 0.$\bar{6}$	0.82736
max_depth = 4, gamma = 10, colsample-bytree = 1	0.82723
max_depth = 5, gamma = 0, colsample-bytree = 0.$\bar{6}$	0.82738
max_depth = 5, gamma = 0, colsample-bytree = 1	0.82742
max_depth = 5, gamma = 0.1, colsample-bytree = 0.$\bar{6}$	0.82737
max_depth = 5, gamma = 0.1, colsample-bytree = 1	0.82757
max_depth = 5, gamma = 1, colsample-bytree = 0.$\bar{6}$	0.82746
max_depth = 5, gamma = 1, colsample-bytree = 1	0.82732

max_depth = 5, gamma = 10, colsample-bytree = $0.\overline{6}$	0.82754
max_depth = 5, gamma = 10, colsample-bytree = 1	0.82759
max_depth = 6, gamma = 0, colsample-bytree = $0.\overline{6}$	0.82689
max_depth = 6, gamma = 0, colsample-bytree = 1	0.82691
max_depth = 6, gamma = 0.1, colsample-bytree = $0.\overline{6}$	0.82691
max_depth = 6, gamma = 0.1, colsample-bytree = 1	0.82703
max_depth = 6, gamma = 1, colsample-bytree = $0.\overline{6}$	0.82701
max_depth = 6, gamma = 1, colsample-bytree = 1	0.82687
max_depth = 6, gamma = 10, colsample-bytree = $0.\overline{6}$	0.82743
max_depth = 6, gamma = 10, colsample-bytree = 1	0.82722

Overall trends show that max_depth 3 is optimal and colsample-bytree = 1 is optimal when max_depth = 3. It is unclear whether it's worth changing gamma from the default value of 0. Based on the trends from the first round of grid search, fix max_depth = 3 and colsample-bytree=1, and narrow the range of gamma

2nd Grid Search:

max_depth = 3, colsample_bytree = 1, gamma = {0, 0.1, 1, 2}, min_child_weight = {1, 2, 5, 10}

Parameter Settings	**Avg. AUROC Score Across 5-fold CV**
gamma = 0, min_child_weight = 1	0.82702
gamma = 0, min_child_weight = 2	0.82736
gamma = 0, min_child_weight = 5	0.82778
gamma = 0, min_child_weight = 10	0.82730
gamma = 0.1, min_child_weight = 1	0.82708
gamma = 0.1, min_child_weight = 2	0.82742
gamma = 0.1, min_child_weight = 5	0.82745
gamma = 0.1, min_child_weight = 10	0.82730
gamma = 1, min_child_weight = 1	0.82717
gamma = 1, min_child_weight = 2	0.82742
gamma = 1, min_child_weight = 5	0.82723
gamma = 1, min_child_weight = 10	0.82737
gamma = 2, min_child_weight = 1	0.82744
gamma = 2, min_child_weight = 2	0.82743
gamma = 2, min_child_weight = 5	0.82751
gamma = 2, min_child_weight = 10	0.82741

The trends observed here are that min_child_weight = 5 seems to be optimal, while the value of gamma isn't having a significant impact on performance. Let's set gamma to the default value, and explicitly require min_child_weight = 5 moving forwards.

Final parameters from grid search:

max_depth = 3, colsample_bytree = 1, gamma = 0, min_child_weight = 5