CEPC NOTE

March 19, 2018

中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

W/Z Classification at CEPC

Yongfeng Zhu

Abstract

Could we find a way to classify w and z boson when they all decay to quarks? This note will introduce two ways to classify w and z boson, one is Invariant-mass method, another is deep-neural-network method.

E-mail address: li.gang@mail.ihep.ac.cn

Contents

1 Introduction

A project on a high energy Circular Electron Positron Collider (CEPC) as a Higgs and/or Z factory and a subsequent Super Proton-proton Collider (SPPC) is at the stage of Conceptual Design Report study. When $e^+e^- \to ww \to 4quarks$ or $e^+e^- \to zz \to 4quarks$, we want to find a powerful way to classify w and z. Until know, we had tried two ways, Invariant-mass and deep-neural-network, to classify w and z at generation level (not through simulation or reconstruction).

2 Samples

The samples we use are all at generation level(green rhombus showed in Figure 1), The detailed information is listed in Table 1.

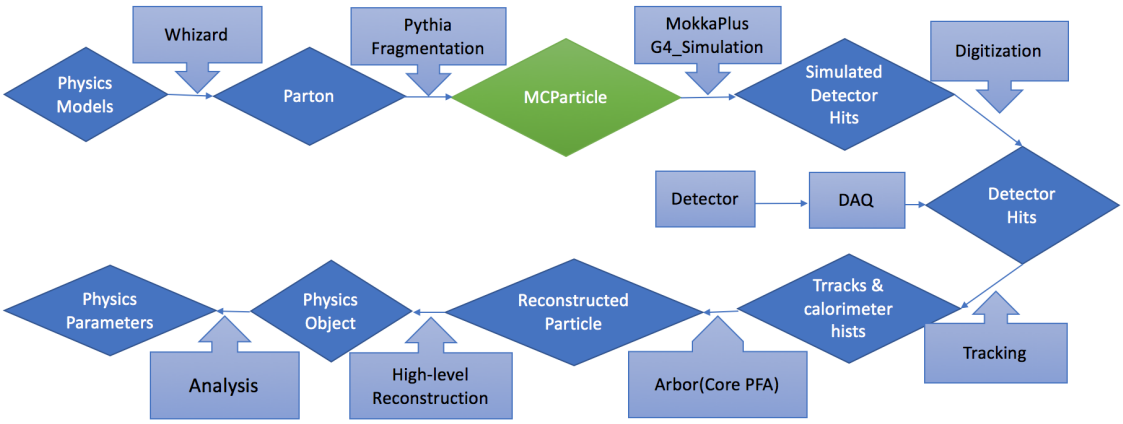

Figure 1: the working process

Table 1: The channels and the number of events used in our experiment

ww channel	the number of ww event	zz channel	the number of zz event
$ww \to ccbs$	100 000	$zz \to utut$	199 800
$ww \to ccds$	100 000	$zz \to dtdt$	199 593
$ww \to cusd$	99 990		
$ww \to uubd$	100 000		
total number	399 990		399 393

3 the definition of accuracy

We need to make a rule to compare the classification result of two methods introduced above. In this note, we use the following rule: assumpt the number of w and z boson is wtruth and ztruth separately, correctly classifying the number of w and z boson is wprediction and zprediction separately, so the incorrectly classifying number of w and z boson is (wtruth-wprediction) and (ztruth-zprediction) separately since there are two classes,The definition shows in Table 2.

Table 2: The definition of classification accuracy

accuracy	w prediction	z prediction
w truth	$waccuracy = \frac{wprediction}{wtruth}$	$wnonaccuracy = \frac{wtruth-wprediction}{wtruth}$
z truth.	$znonaccuracy = \frac{ztruth-zprediction}{ztruth}$	$zaccuracy = \frac{zprediction}{ztruth}$

4 Invariant-mass method

4.1 introduction of Invariant-mass method

We all know that the rest-mass of w and z boson is 80Gev and 91Gev separately, we can use this character to classify w and z. When ww or zz decay to 4 quarks, two among of them are decayed from one of two w bosons or two z bosons. For z boson, the sum of the value of PDG of two quarks is zero. For w boson, the absolute value of the sum of the charge of two quarks is one. We can use these features to correctly pairing quarks. The Figure 2 shows the code of correctly pairing quarks decay from z boson. The Figure 3 shows w.

```
int q1, q2, q3, q4;
q1=2;
for(q2 = 3; q2 < 6; ++q2)
{
    if(zzpdgid[q1] + zzpdgid[q2] == 0)
   { break;}
}
for(q3 = 3; q3 < 6; ++q3)
{
    if(q3 != q1 && q3 != q2){break;}
}
for(q4 = 3; q4 < 6; ++q4)
{
    if(q4 != q1 && q4 != q2 && q4! = q
}
```

Figure 2: the code for pairing quarks decayed from zz

```
int q1, q2, q3, q4;
q1=2;
for(q2 = 3; q2 < 6; ++q2)
{
    if(abs(charge[q1] + charge[q2]) == 1)
       { break;}
}
for(q3 = 3; q3 < 6; ++q3)
{
    if(q3 != q1 && q3 != q2){break;}
}
for(q4 = 3; q4 < 6; ++q4)
{
    if(q4 != q1 && q4 != q2 && q4! = q3){break;}
}
```

Figure 3: the code for pairing quarks decayed from ww

4.2 classification result

We can calculate the invariant mass of two quarks when we correctly pairing quarks, in other words, the value of invariant mass is also the rest mass of w or z. We can classify w and z by the value of invariant mass. The distribution of invariant mass calculated through the four momentum of two quarks shows in Figure 4. We can see there is some overlapping, that's why we can't classify w and z exactly. We can find a cut to separate w and z. When cut move from 0Gev to 250Gev, the classification accuracy show in Figure 5. We need the classification accuracy to reach to maximum meanwhile, but the accuracy can't exceed 90% meanwhile.

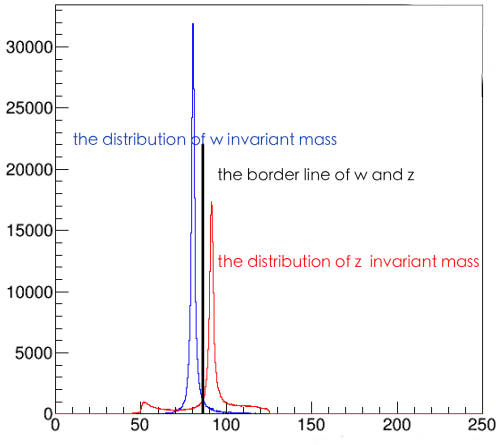

Figure 4: the distribution of the invariant mass

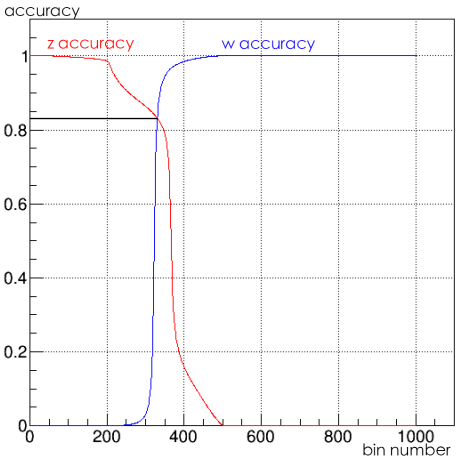

Figure 5: the classification accuracy when cut move from zero to 250Gev

5 Deep-neural-network method

5.1 Introduction of deep-neural-network

Since we living in a big-data era, there are lots of information around us. Deep-learning can use these information effectively to get general structure. Each information is a sample, a sample has more or less features, these features used to describe these samples. Each sample belongs to a class or has a value, the class or the value is the sample's label. For our problem, the class is w or z, every sample is a ww or zz event, the features of every sample is the four momentum or invariant mass of four quarks. We have about 800 000 samples, 600 000 of them divided to training-set, 100 000 of them divided to validation-set and 80 000 fo them divided to testing-set. Training-set used to train model, validation-set used to avoid over-fitting and testing-set used to test the efficient of model.

The Figure 6 is a simple neural-network, it has three layers: input-layer, hidden-layer and output-layer. The circle represent neuron, the connection line represent weight(a vector) which value got through training. The neuron in neural-network seems like biological neuron, it can accept information and process these information and get output result. Figure 7 is a typical neuron structure. The neural network is constituted with lots of neurons. It can accept input from neurons in the last layer and multiply input with each weight and sum them. The box represent a non-linear activation function, it is Relu in this note. The working process can be expressed with $y = f(\sum_{i=1}^{n} w_i x_i + b_i)$.

The deep-neural-network(DNN) is one of the deep-learning method. DNN has another name called multi-layer-perceptron, it's name comes from that it has more than two layers: input layer, output layer

and hidden layers. The structure of DNN seems like Figure 6, the number of hidden layers can be more than one. The number of input layer and output layer is only one. The number of neurons of input layer is same as the number of features of sample. The number of neurons in output layer is the number of classes. The number of neurons in each hidden layer based on our experience. The working process of neurons is same as Figure 7. The information transit through input layer to output layer and get the final classification result.

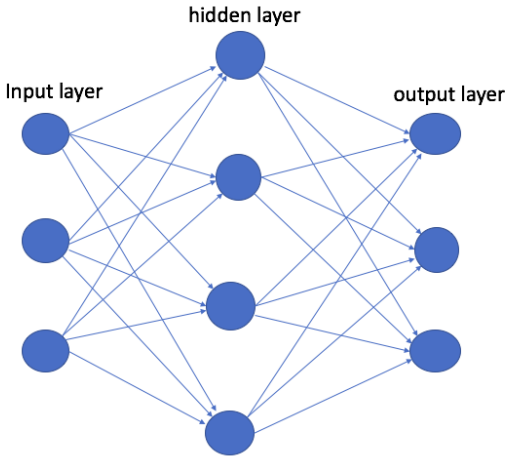

Figure 6: the distribution of the invariant mass

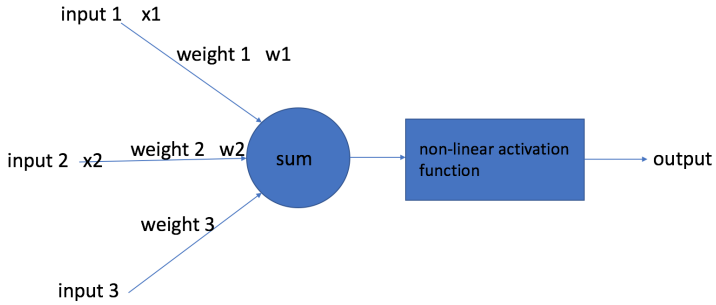

Figure 7: the classification accuracy when cut move from zero to 250Gev

5.2 the structure of DNN model in this note

```
model.add(Dense(units=32, input_dim=16))
model.add(Activation("relu"))
model.add(Dense(units=64))
model.add(Activation("relu"))
model.add(Dense(units=128))
model.add(Activation("relu"))
model.add(Dense(units=64))
model.add(Activation("relu"))
model.add(Dense(units=32))
model.add(Activation("relu"))
model.add(Dense(units=16))
model.add(Activation("relu"))
model.add(Dense(units=1))
model.add(Activation("sigmoid"))
model.summary()
model.compile(loss='binary_crossentropy',optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08),  \
              metrics=['accuracy'])
history=LossHistory()
early_stopping = EarlyStopping(monitor='val_loss', patience=5)
```

Figure 8: the structure of DNN model

For our problem, we have designed a DNN model. The detail shows in Figure 8. Since that we have four type samples, the number of features for every type sample is different, so we need four models, the structure of every models is similar except that the number of neurons need to change with the number of features of sample. We can change the number of neurons of each layer to train model to get the best output. About the code of model, we give the following introduction:

- $model.add(Dense(units = 32, input\_dim = 16))$: the value of $input_dim$ is the number of features of our sample, the value of units represents the number of neurons in the first hidden layer.
- $model.add(Activation(\text{"}relu\text{"}))$:the function "relu" is the non-linear activation function since our problem is non-linearly separable, under the action of "rely", our problem would become linearly separable. Figure 9 is the function graph of "relu".
- $model.add(Activation(\text{"}sigmoid\text{"}))$: the function "sigmoid" is another non-linear activation function, its's function graph shows in Figure 10. The output of last hidden layer is a value distribution from zero to one, while our class label is zero or one. We need to make a cut, the value above the cut turn to integer one while under the cut turn to integer zero. For our problem, the function sigmoid can does it.
- $model.compile(loss =' binary_crossentropy', optimizer = Adam(lr = 0.001, beta_1 = 0.9...), metrics = ['accuracy'])$:loss represents $loss_function$, it's a method to calculate the difference between the result of prediction and real. At this note, we use "$binary\_crossentropy$". Optimizer represents a type of back-propagation function to train model, we use "Adam" in this note, the values in brackets is the parameters of Adam. The metrics is a method to definite classification accuracy as we had definite above.
- $early\_stopping = EarlyStoping(monitor =' val\_loss', patience = 5)$: EarlyStoping is used to avoid $over_fitting$, the training process would stop when val_loss(the value of $loss_function$ in $validate_set$) don't decrease in five continuous training loops.
- $model.summary()$: this command will record log information of the training process for our inspection.

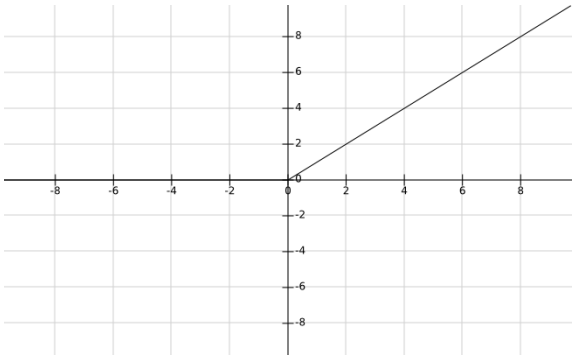

Figure 9: "relu" function

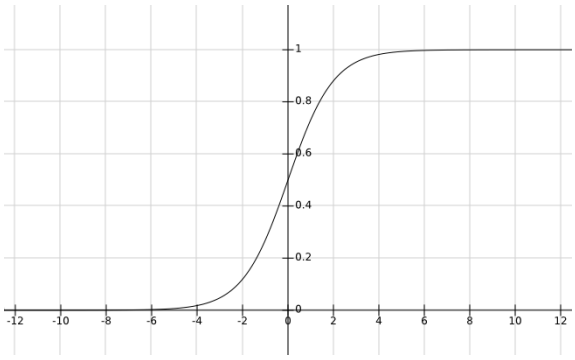

Figure 10: sigmoid function

5.3 classification result

Every column corresponds a special condition, the type of features different between each condition, the concrete information described in the last row of Table 3. The first two rows list the classification accuracy, which definition as Table 2.

Table 3: The classification accuracy for four condition

0.9665	0.0335	0.9627	0.0373	0.995075	0.004925	0.975875	0.02415
0.034825	0.965175	0.038075	0.961925	0.002475	0.997525	0.033025	0.966975
two features: two correct invariant mass of four quarks		sixteen features: four momentum of four quarks		eighteen features: four momentum and two correct invariant mass of four quarks		twenty-two features: four momentum and six invariant mass of four quarks	

6 conclusion

We had classified w and z at generation level. If the effect of invariant-mass method is good, then we can say the deep-neural-network method is perfect. From the analyses results, It's possible that the model hadn't learn to calculate the mass-energy relation, but after training the model had extract more useful information, which is much beyond invariant mass to classify w and z. The deep-learning is a powerful box, it can finds a special way to attain our goal automatically.