Measure Higgs CP via tth Channel

Ren-Qi Pan

Zhejiang Institute of Modern Physics

renqipan@zju.edu.cn

June 28, 2019

Overview

1. Towards Machine Learning
   - Decision tree
   - Boosting decision tree
   - Keras-DNN

2. Apply Machine Learning to Measure Higgs CP via tth
   - Determine input variables
   - Response value and ROC curve

Machine Learning

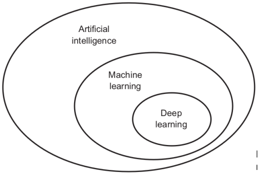


Figure: Artificial intelligence

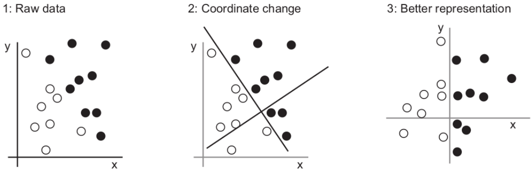


Figure: Coordinate change. Learning, in the context of machine learning, describes an automatic search process for better representations.

Common Models of Machine Learning

- Support vector machines
- Bayesian networks
- Boosting decision trees: AdaBoost, Gradient Boosting, XGBoost
- Artificial neural network

Decision tree

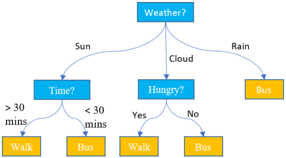


Figure: A decision tree

A decision tree takes a set of input features and splits input data recursively based on those features.

- Nodes
  - The data is split based on a value of one of features at each node.
  - Sometime called interior nodes
- Leaves
  - Terminal nodes
  - Represent a class label or probability
  - If the outcome is a continuous variable it's considered a regression tree

Decision tree

Learning:

- Each split at a node is chosen to maximize information gain(rate)
- The splits are created recursively

Note:

- Information gain is the difference in entropy before and after the potential split
- The process is repeated until some stop condition is met. Ex: depth of tree, no more information gain, etc...

Information Gain

category information entropy:

$$Info(D) = -\sum_{i}^{m} p_i log_2(p_i)$$

feature information entropy:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} * Info(D_j)$$

information gain: $Gain(A) = Info(D) - Info_A(D)$

Information Gain Rate

feature splitting information entropy:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} log_2 \frac{|D_j|}{|D|}$$

information gain rate:

$$GainRate(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

Boosting decision tree

Usually, a single tree is not strong enough to be used in practice. What is actually used is the ensemble model, which sums the prediction of multiple trees together.

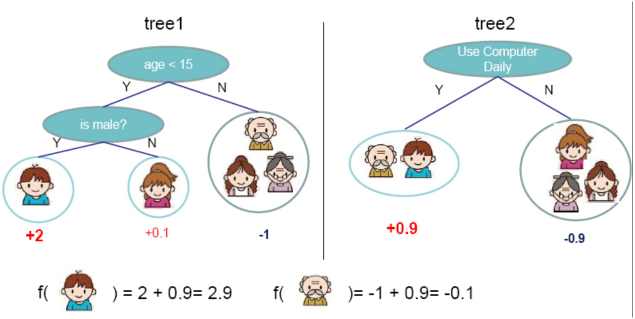


Figure: Here is an example of a tree ensemble of two trees. The prediction scores of each individual tree are summed up to get the final score

Tree boosting

Boosting is a method of combining many weak learners (trees) into a strong classifier.

- Each tree is created iteratively
- The trees output (h(x)) is given a weight (w) relative to its accuracy
- Use residual to create a new tree
- The ensemble output is the weighted sum: $\hat{y} = \sum_t w_t h_t(x)$
- After each iteration each data sample is given a weight based on its misclassification
  Note: The more often a data sample is misclassified, the more important it becomes
- The goal is to minimize an objective function

Types of boosting

There are many different ways of iteratively adding learners to minimize a loss function. Some of the most common:

- AdaBoost
  - Adaptive Boosting
  - One of the originals
  - residual=prediction- real value
- Gradient Boosting
  - Uses gradient descent to create new learners
  - The loss function is differentiable
  - residual: minus gradient of loss function
- XGBoost
  - eXtreme Gradient Boosting
  - Type of gradient boosting
  - Objective Function: Training Loss + Regularization
  - Taylor expansion of the loss function up to the second order
  - Shrinkage and Column Subsampling

Keras and TensorFlow

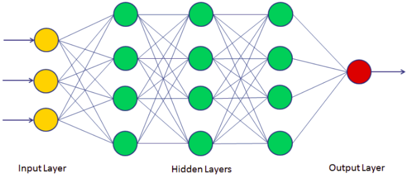
Keras:

- model-level library(released in 27 March 2015)
- an open-source neural-network library written in Python
- providing high-level building blocks for developing deep-learning models

TensorFlow:

- TensorFlow is developed by Google(released in November 9, 2015)
- serving as the backend engine of Keras
- handle low-level operations such as tensor manipulation and differentiation

Mechanism of DNN



(a) mechanism of artificial neural network

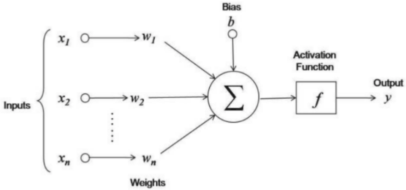


Image Source

$$Y = \sum (weight * input) + bias$$

(b) Here's a diagram of what one node might look like.

Activation function

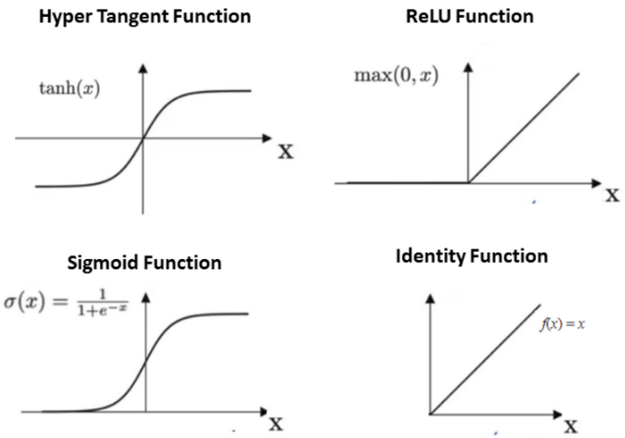


Figure: common activation function used in artificial neural network.

Components of neural network

Training a neural network revolves around the following objects:

- **Layers**, which are combined into a network (or model)
- The **input data** and corresponding **targets**
- The **loss function**, which defines the feedback signal used for learning
- The **optimizer**, which determines how to update weights

How neural network work?

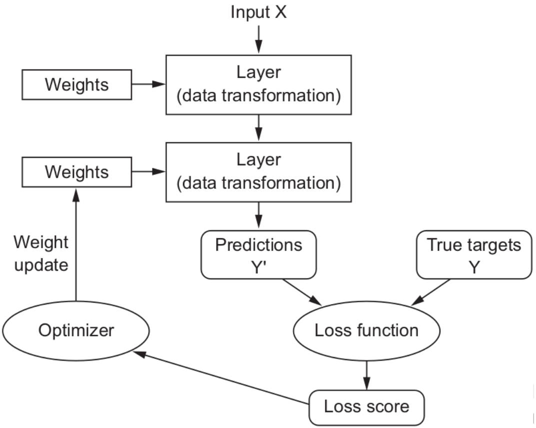
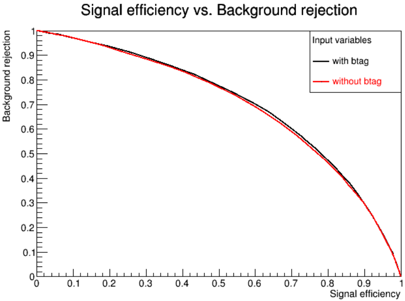


Figure: Relationship between the network, layers, loss function, and optimizer

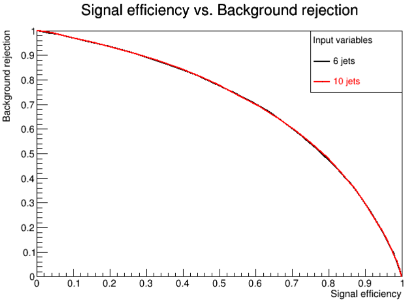
An neural network model

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(ndim,))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

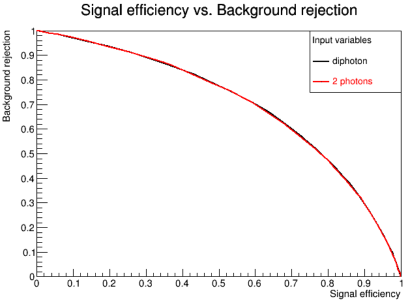
Figure: An example of artificial neural network model with 4 layers.



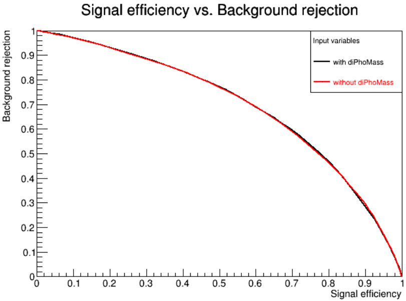
(a) compare the ROC curves between with and without btag



(b) compare the ROC curves between 6 jets and 10 jets



(c) diphoton information vs. 2 photons information



(d) with or without diPhoMass

Different input variables

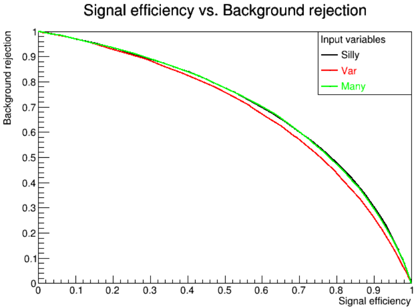


Figure: compare the ROC curves between three different types of input variables.

Input variables

From here on the input variables are :

- 6 jets: Pt, E, Eta, Phi, btag
- diPhoPhi, diPhoEta, diPhoPtoM
- in total 33 variables

	jetPt_1	jetEta_1	jetPhi_1	jetE_1	btag_1	jetPt_2	jetEta_2	jetPhi_2	jetE_2	btag_2	...	jetPt_6	jetEta_6	jetPhi_6	jetE_6
0	179.1340	0.129419	-0.770991	184.053	0.920134	75.2335	-0.047224	-1.281490	76.1059	0.999764	...	37.2568	-0.299217	-1.762450	39.8590
1	161.2780	1.071580	-0.156882	263.612	0.913174	126.8670	0.646219	2.747790	155.3490	0.999908	...	30.6574	1.363930	-0.648138	64.0560
2	157.1970	-0.333851	0.212655	166.653	0.005083	68.6658	-1.467650	0.122107	157.2250	0.999957	...	50.2075	-0.554880	3.135870	58.4124
3	143.9920	-0.899087	-0.825905	206.971	0.993589	112.7960	0.119058	0.629660	114.7540	0.132374	...	30.1873	-0.606200	0.356470	36.3707
4	85.5433	0.699251	-2.586590	108.438	0.999996	74.1270	0.099549	-0.422806	75.8854	0.054680	...	33.7513	-1.301440	2.364960	66.7972

Figure: partial input data

TMVA response for classifier: AdaBoost

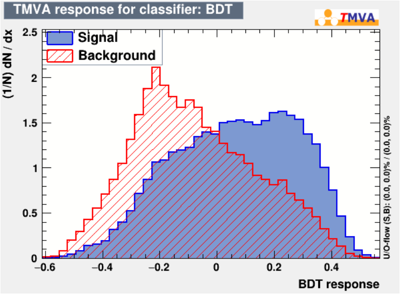
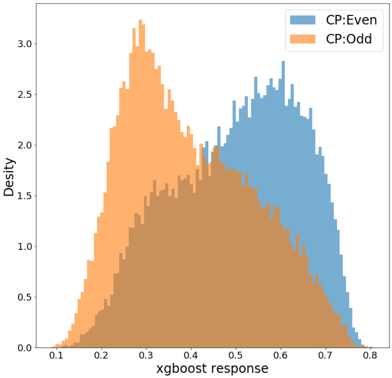
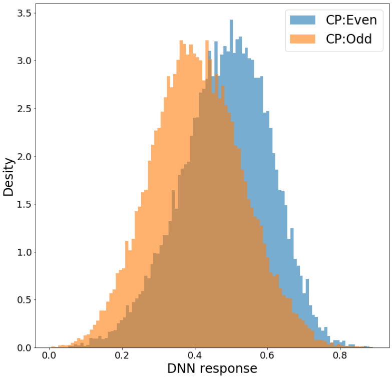


Figure: TMVA-AdaBoost response under Higgs CP even and odd.

xgboost and DNN



(a) xgboost response



(b) keras-DNN response

Different models

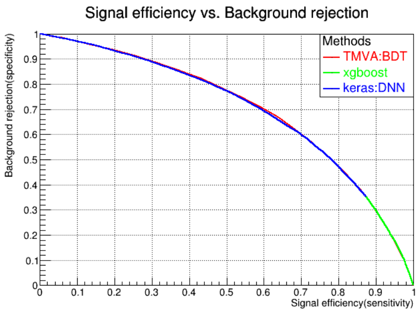


Figure: compare the ROC curves of three different models.

The End