Utilizing Neural Network-based Classifiers for the Extraction of DY Angular Coefficients

Dinupa Nawarathne, Vassili Papavassiliou,
Stephen Pate-Morales, Abinash Pun, Forhad Hossain

May 31, 2023

Introduction

- A classifier is an algorithm/model that is trained to assign labels/categories to input data based on patterns/features present in the data. It is a fundamental component of supervised machine learning, where the goal is to learn a mapping between input data and their corresponding labels.
- The likelihood ratio method is a statistical technique used for hypothesis testing and decision-making. It involves comparing the likelihoods of observed data under different hypotheses to determine the relative support for each hypothesis.
- Classifiers can be used in the likelihood ratio method as a way to discriminate between two different hypotheses/classes based on the observed data.

- The Classifier can be any supervised learning algorithms, such as logistic regression, decision trees, support vector machines, or neural networks.
- The training process involves adjusting the model parameters to minimize a specific loss function or maximize the likelihood of the observed data given the class labels.
- Given a new set of observed data, feed it to classifiers to obtain the predicted probabilities or scores for each class. These probabilities represent the likelihood of the observed data under each hypothesis.
- Calculate the likelihood ratio by dividing the likelihood under one hypothesis by the likelihood under the other hypothesis. In binary classification, this can be done by taking the ratio of the predicted probabilities for the two classes.

- Our primary objective is to utilize the likelihood ratio method with a neural network classifier to formulate a parametrized weight function. By achieving this, we can further apply this parametrized neural network to uncover the unknown features within a given dataset. [1]
- Suppose that f is a neural network trained to distinguish between two distinct classes. Using this neural network, we can compute the relative weights assigned to each of the two classes by;

$$w(x, \theta) = \frac{f(x)}{1 - f(x)} \tag{1}$$

where x and θ are input data and parameters of the model.

[1] Parameter Estimation using Neural Networks in the Presence of Detector Effects. arXiv:2010.03569 [hep-ph]

Simple Example: 1D Gaussian Fitting

- To demonstrate this approach, we aim to extract the unknown parameter μ from a one-dimensional Gaussian distribution $\mathcal{N}(\mu, \sigma)$.
- The neural network consists of three hidden linear layers, each consist of 50 nodes. Rectified Linear Units (ReLU) are employed to connect the hidden layers, and the final output is passed through a Sigmoid activation function.
- The neural network was trained for 200 epochs, employing early stopping with a patience of 10, to minimize the binary cross-entropy loss.
- The process of extracting the unknown parameter involves three steps. [2]

[2] This example closely follows the steps outlined in "Neural Networks for Full Phase-space Reweighting and Parameter Tuning" arXiv:1907.08209 [hep-ph]

Step 1: Compute the weights for the distribution $\mathcal{N}(1, 1.3)$ with respect to the distribution $\mathcal{N}(0, 1)$.

- We train our neural network to distinguish samples drawn from $\mathcal{N}(0, 1)$ and $\mathcal{N}(1, 1.3)$.
- We use equation 1 to calculate the weights.

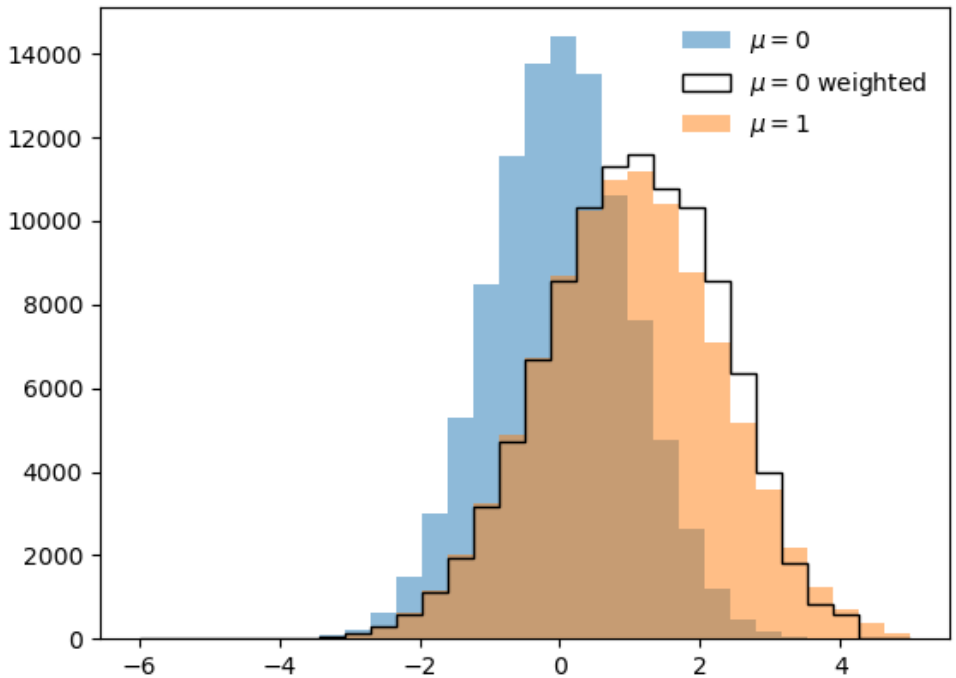

Figure 1: Weighted $\mathcal{N}(0, 1)$ distribution.

- Since weighted $\mathcal{N}(0, 1)$ distribution closely approximate the $\mathcal{N}(1, 1.3)$ distribution, we can use neural network-based classifier to determine the weights between any two $\mathcal{N}(\mu, \sigma)$.

Step 2: Compute the weights for the distribution $\mathcal{N}(\mu, 1)$ with respect to the distribution $\mathcal{N}(0, 1)$ with one Model for any μ

- We train our neural network to distinguish samples drawn from $\mathcal{N}(0, 1)$ and $\mathcal{N}(\mu, 1)$, where μ is uniformly sampled from the range of $(-2.0, 2.0)$.
- In this step, we have parametrized our neural network to calculate the weights for any μ within the range of $(-2.0, 2.0)$.

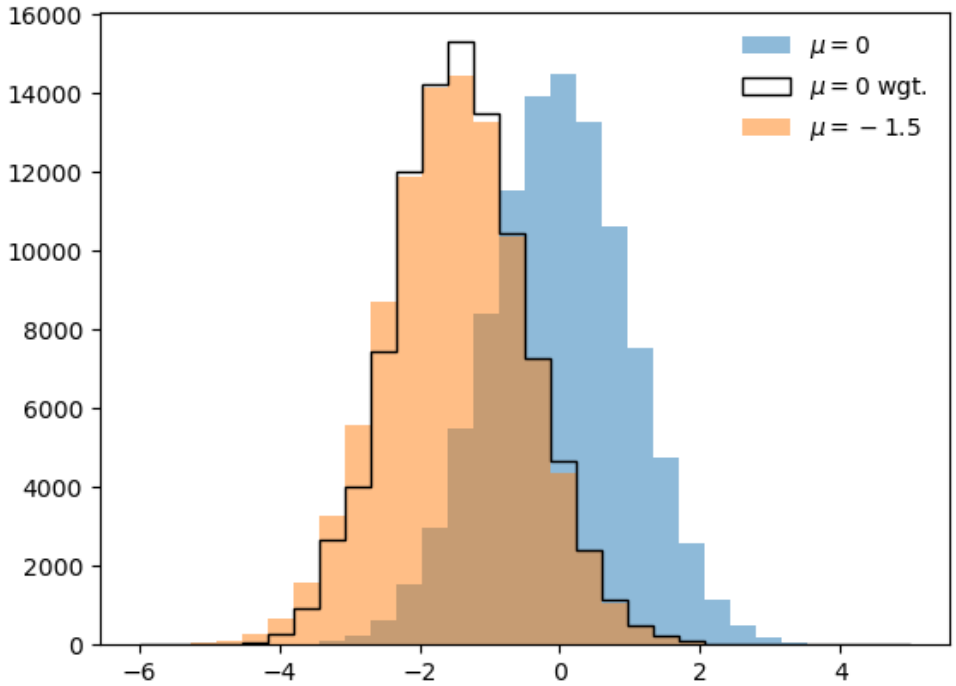

Figure 2: Weighted $\mathcal{N}(0, 1)$ distribution.

Step 3: Using the gradient descent algorithm to extract the unknown parameter μ.

- Now, we have parameterized our neural network to calculate weights for any μ value.
- In this step, we attempt to extract the unknown parameter μ by minimizing the loss using a gradient descent algorithm. [3]

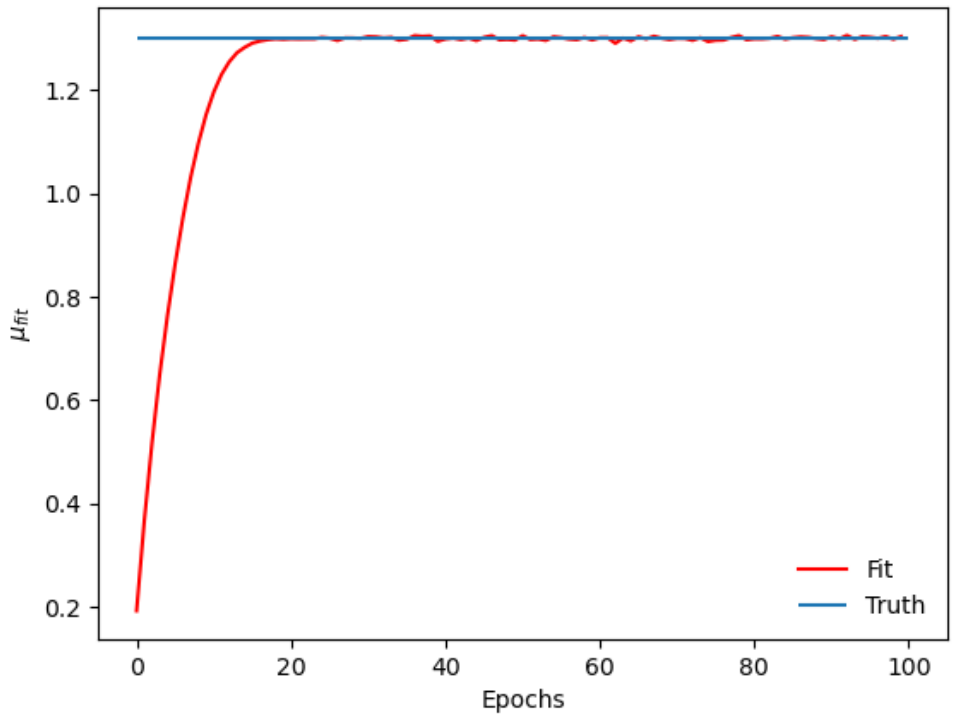

Figure 3: Fitted μ values with epochs.

[3] A similar approach has been proposed in "Parameter Estimation using Neural Networks in the Presence of Detector Effects". arXiv:2010.03569 [hep-ph]. However, in the last step, it employs a non-differentiable minimizer to minimize the AUC value.

- In this example, we utilized the Adam optimizer to determine the best-fit value of μ.
- In this particular example, the unknown μ value is $\mu_? = 1.3$, while the fitted μ value is $\mu_{\text{fit}} = 1.309$.

Applying this Method to E906 Simulated Data

- We utilize the messy MC reconstructed data to train the neural network, with ϕ and $\cos\theta$ serving as inputs. The neural network is parametrized using the values λ, μ, and ν.
- We sample the values of λ, μ, and ν uniformly from the ranges of $(0.5, 1.5)$, $(-0.5, 0.5)$, and $(-0.5, 0.5)$, respectively. [4]
- The neural network consists of three hidden linear layers, each consist of 100 nodes. Rectified Linear Units (ReLU) are employed to connect the hidden layers, and the final output is passed through a Sigmoid activation function.
- The neural network was trained for 200 epochs, employing early stopping with a patience of 20, to minimize the binary cross-entropy loss.

[4] These ranges were chosen based on the results of the E866 experiment "Measurement of Angular Distributions of Drell-Yan Dimuons in p + d Interaction at 800 GeV/c" arXiv:hep-ex/0609005

Performing a Sanity Check

- We conducted a sanity check using five different combinations of λ, μ, and ν. The fitted values are presented in table 1.

Combination	Coefficient	Injected	Fitted
1	λ	0.9	0.888 ± 0.202
	μ	0.1	0.112 ± 0.046
	ν	0.1	0.089 ± 0.039
2	λ	1.2	1.086 ± 0.166
	μ	0.2	0.189 ± 0.060
	ν	-0.1	-0.081 ± 0.034
3	λ	0.8	0.860 ± 0.185
	μ	0.0	0.035 ± 0.035
	ν	0.3	0.281 ± 0.045

Combination	Coefficient	Injected	Fitted
4	λ	1.15	1.109 ± 0.163
	μ	-0.35	-0.291 ± 0.072
	ν	0.45	0.378 ± 0.063
5	λ	0.84	0.852 ± 0.157
	μ	0.26	0.265 ± 0.062
	ν	-0.34	-0.305 ± 0.047

Table 1: Table showing the fitted values of λ, μ, and ν using the gradient descent algorithm. Each combination contains 30K messy MC events.

- In the five test samples, we were able to extract the injected parameters within a ± 1.5 standard deviation (σ) interval.

Summary

- This is an unbinned method capable of processing multi-dimensional feature spaces and parameter spaces.
- Unlike traditional methods that typically use low-dimensional unfolded data, this method allows for the direct extraction of unknown parameters using high-dimensional detector-level data.
- We plan to fine-tune our network architecture by increasing the input features, incorporating dropout layers, batch normalization, and other techniques to enhance the results.

A 2D Example

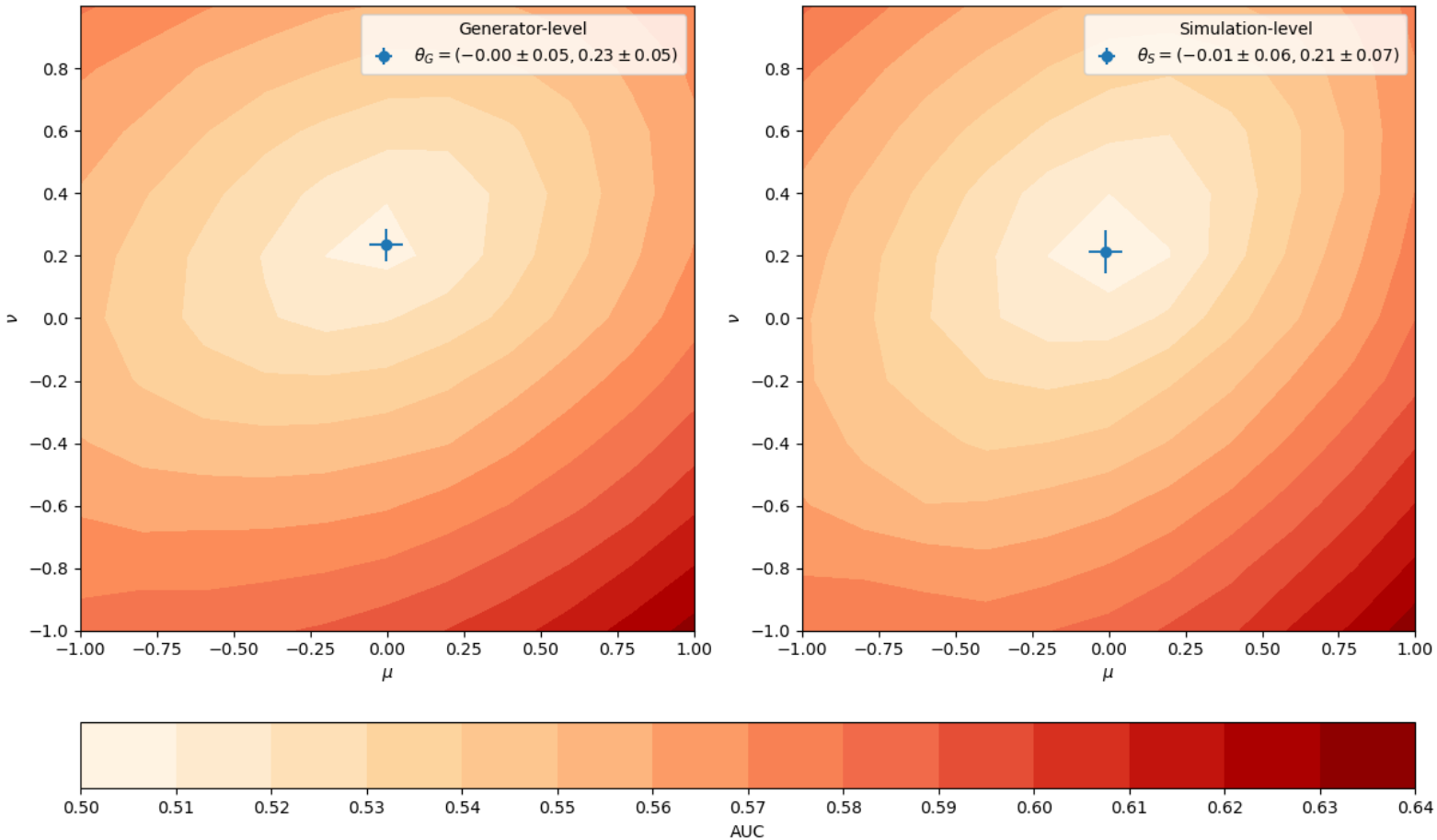

Figure 4: Extraction of μ and ν by fixing $\lambda = 0.0$. Injected values are $(\lambda, \mu, \nu) = (0.4, 0.0, 0.2)$. θ is the unknown parameter (ν and μ) in this example.

- In this example we fixed the λ and try to extract the μ and ν values.