Statistical Interpretation of ML Discriminator Results

Mohamed Elbeltagi

DANCE Workshop

OCT 2019

Introduction

- Machine learning (ML) algorithms are playing an increasingly important role in analysis of particle physics experiments

- Deep learning methods starting to outperform conventional approaches in Energy/Position reconstruction, Signal/Bkgrnd discrimination

- Analyses using ML already exist but mostly play a small role within the traditional analysis (exo-200 PRL paper arXiv:1906.02723), interest in having end-to-end ML analysis.

- Despite its improved performance some skepticism remains in parts of the nuclear and particle physics communities due to :
    1. Lack on interpretability (“Black Box”)
    2. Scarce evidence of performance on real detector data
    3. Absence of rigorous treatment of statistical/systematic errors

- We aim to address the third point in the context of discriminators for rare event searches

- For a typical discriminator between a number of different classes (event types), an “event” is passed to the algorithm with set of features, and the algorithm predicts which class it belongs to

$$X_i \xrightarrow{\text{MLA}} X_i^C$$

- We can represent this in the following way

$$\vec{X}^C = \tilde{P}\vec{X} \quad \text{or} \quad X_i^C = \sum_{j=1}^{M} P_{ij} X_j$$

- Matrix $\tilde{P}$ represents the true action of the algorithm on the vector $\vec{X}$ containing the true number of events of each type, to give the predicted number of events in each class $\overrightarrow{X^C}$

- For simplification, use the confusion matrix $\tilde{B}$ (average performance) to represent the algorithm, averages

$$\tilde{B} = \begin{bmatrix} b_{11} & \dots & b_{1M} \\ \dots & \ddots & \dots \\ b_{M1} & \dots & b_{MM} \end{bmatrix}$$

- Vector $\vec{X}$ (unknown) is the true number of events of each type occurring in the detector for a particular run with the true event numbers following a multinomial

$$f(X_i; p_i; N, M) = N! \prod_{i=1}^{M} \left(\frac{{p_i}^{X_i}}{X_i!} \right)$$

- N total events, M event types, we're interested in estimating the true probabilities of events occurring p_i
- Transform the previous distribution to find the pdf of classified events in each type (known) as a function of the p_i's

$$h(\vec{X}^C; p_i; N, M) = \frac{1}{\left|\det \tilde{B}\right|} f\left(\tilde{B}^{-1}\vec{X}^C; p_i; M, N\right) = \frac{N!}{\left|\det \tilde{B}\right|} \prod_{i=1}^{M} \left[\frac{{p_i}^{\left(\sum_{j=1}^{M} (\tilde{B}^{-1})_{ij} X_j^C\right)}}{\left(\sum_{j=1}^{M} (\tilde{B}^{-1})_{ij} X_j^C\right)!} \right]$$

- For rare event searches, data usually segmented into independent measurements, “runs”.
- Given a single run: $\overrightarrow{X^C} = \overrightarrow{x^C}$, the Likelihood function is

$$L(p_i; \vec{x}^C; N, M) = \frac{N!}{\left|\det \tilde{B}\right|} \prod_{i=1}^{M} \left[\frac{p_i^{\left(\sum_{j=1}^{M} (\tilde{B}^{-1})_{ij} x_j^C\right)}}{\left(\sum_{j=1}^{M} (\tilde{B}^{-1})_{ij} x_j^C\right)!} \right]$$

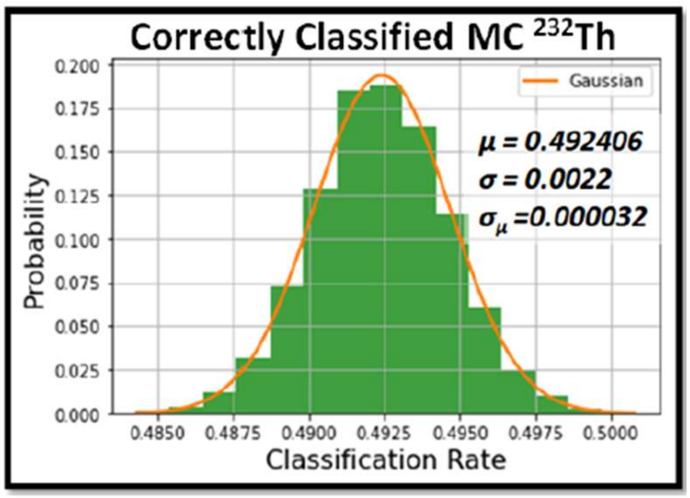
- The log likelihood is

$$\mathcal{L}(p_i; \vec{x}^C; N, M) = \ln N! - \ln \left|\det \tilde{B}\right| + \underline{\sum_{i=1}^{M} \left[\left(\sum_{j=1}^{M} (\tilde{B}^{-1})_{ij} x_j^C \right) \ln p_i \right]} - \sum_{i=1}^{M} \ln \left[\left(\sum_{j=1}^{M} (\tilde{B}^{-1})_{ij} x_j^C \right)! \right]$$

- Extremizing this log likelihood using Lagrange multipliers, the true probabilities of events can be estimated analytically

$$p_i^0 = \frac{1}{N}\sum_{j=1}^{M}(\tilde{B}^{-1})_{ij}x_j^C$$

- This is incorrect, because cannot assume average confusion matrix
- We can assume the matrix elements follow a Gaussian distribution



(from https://curve.carleton.ca/03c9a2b2-d4ae-443c-9bd8-7d5080c89fcd)

- So to find the distribution of the classified events $X_i^C = \sum_{j=1}^{M} P_{ij} X_j$, 2 types of convolution must be performed
- (From Statistical Data Analysis, Cowan)For multiplication of variables:

$$xy = z \qquad \begin{aligned} f(z) &= \int_{-\infty}^{\infty} g(x) h(z/x) \frac{dx}{|x|} \\ &= \int_{-\infty}^{\infty} g(z/y) h(y) \frac{dy}{|y|}, \end{aligned}$$

- For addition of variables:

$$z = x + y \qquad \begin{aligned} f(z) &= \int_{-\infty}^{\infty} g(x) h(z - x) dx \\ &= \int_{-\infty}^{\infty} g(z - y) h(y) dy. \end{aligned}$$

- For P_{ij} Gaussian, X_j multinomial, weren't able to solve analytically

- We turn instead to a numerical solution
- The log likelihood incorporates the confusion matrix elements, so we can account for statistical variation in algorithm performance by pulling the matrix elements randomly from a Gaussian distribution
- Systematic errors, found by comparing confusion matrix to calibration data, can then be added to randomly pulled matrix elements
- The log likelihood can then be extremized numerically to find the estimates

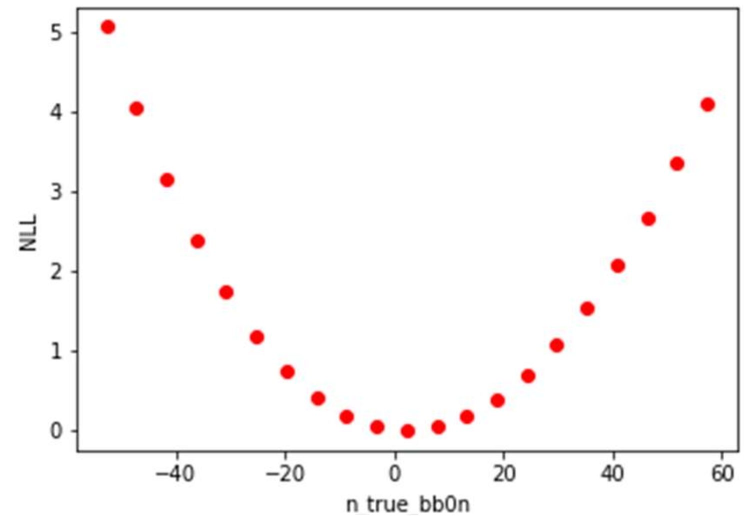
- The software algorithm implemented in Python (using Scipy for minimization with constraints and bounds), minimizes –ve log likelihood to find best estimates of p_i's

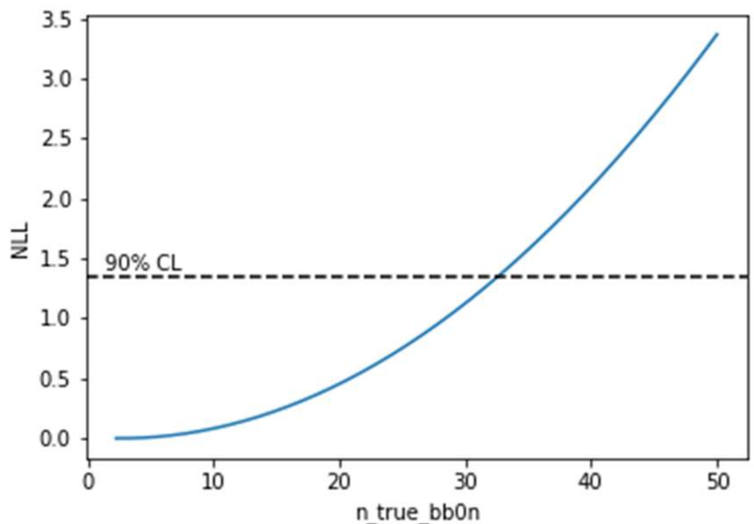
- Constraints on p_i's:

$$0 \leq p_{true,i} \leq 1 \quad , \quad \sum_{i=1}^{m} p_{true,i} = 1$$

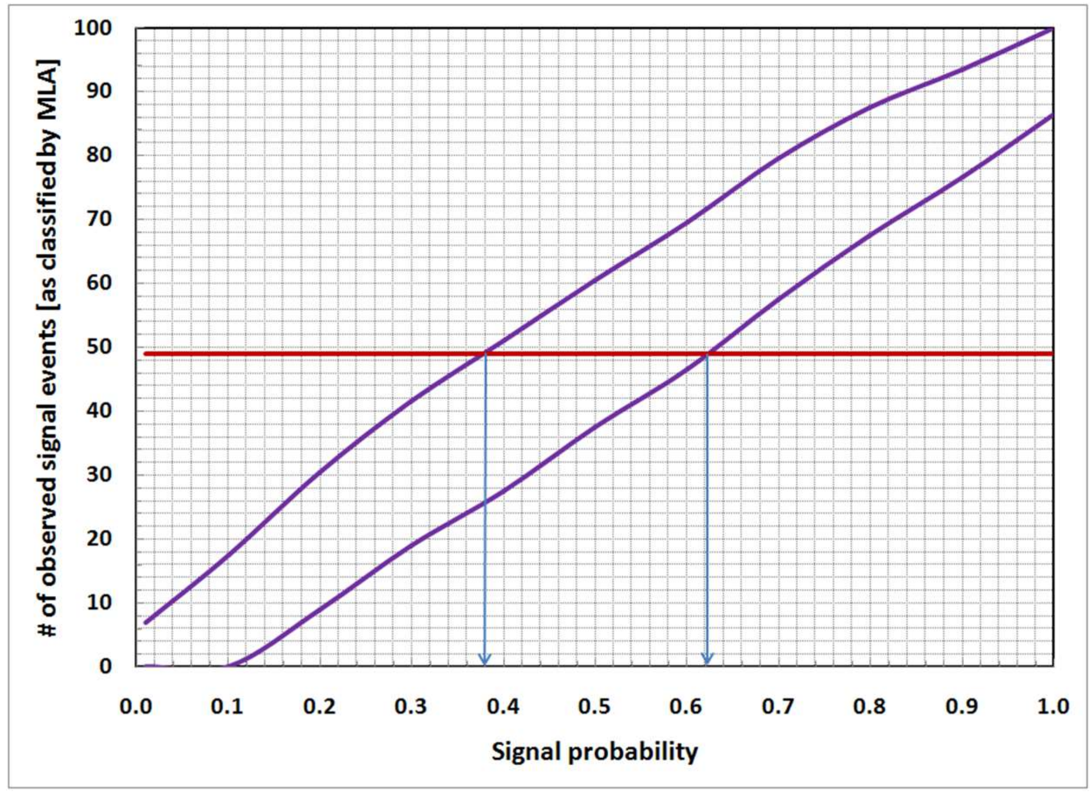
- Ensure normalization and bounds are respected in the minimizer output, and outputs are unique

- Confidence interval for signal is built by incrementing signal estimate and fixing it, then re-minimizing to find backgrounds and calculating log likelihood at these points, then incrementing again and repeating to build likelihood profile (add plot , make own slide, see if matches other approach)





- Alternatively can build acceptance regions using a likelihood ratio (and ordering principle, following Feldman-Cousins) and obtain confidence belts



Conclusion

- Goal is to get direct physics results from low level information
- We treated the problem analytically for the simplified case
- We developed a software package to find estimates and build confidence intervals
- This takes us closer to an end-to-end ML analysis
- We have a paper in preparation with more technical details