Calibrated, Parametrized Learning

Kyle Cranmer and Daniel Whiteson

October 22, 2014

1   Introduction

Notation:

- x: a vector of observed quantities for each event (energy, momenta, angles of particles)

- θ: parameters of a statistical model

- $f_c(x|\theta)$: probability density function (statistical model) for x for category $c \in \{0, 1\}$.

- $s(x)$: real-valued score from a machine learning classification algorithm (or any map $s : X \to \mathbb{R}$)

- $s(x; \theta)$: real-valued score from a parametrized machine learning classification algorithm (or any map $s : X \otimes \Theta \to \mathbb{R}$)

- $f_c(s|\theta)$ The probability density function for s implied by $f_c(x|\theta)$ and $s(x)$

- $f_c(s(\cdot; \theta)|\theta)$ The probability density function for s implied by $f_c(x|\theta)$ and $s(x; \theta)$

1.1   Typical usage of machine learning in HEP

In high-energy physics (HEP) we typically are searching for or measuring the properties of some class of events, generically referred to as *signal*, in the presence of a separate class of *background* events. For each event we measure some quantities x that have corresponding distributions $f_0(x|\theta)$ for background and $f_1(x|\theta)$ for signal. Often machine learning classification algorithms are trained on large samples of synthetic data $\{x_i, c_i\}$ generated with some nominal values of the parameters θ_0, where $c = 0$ corresponds to background and $c = 1$ corresponds to signal. The resulting classifier is denoted $s(x)$. Based on this classifier and large samples of synthetic data drawn from $f_c(x|theta)$ we construct the distribution $f_c(s|\theta)$. An example of the distributions of the distribution of s for the signal and background events with $\theta = \theta_0$ is shown in Figure 1.
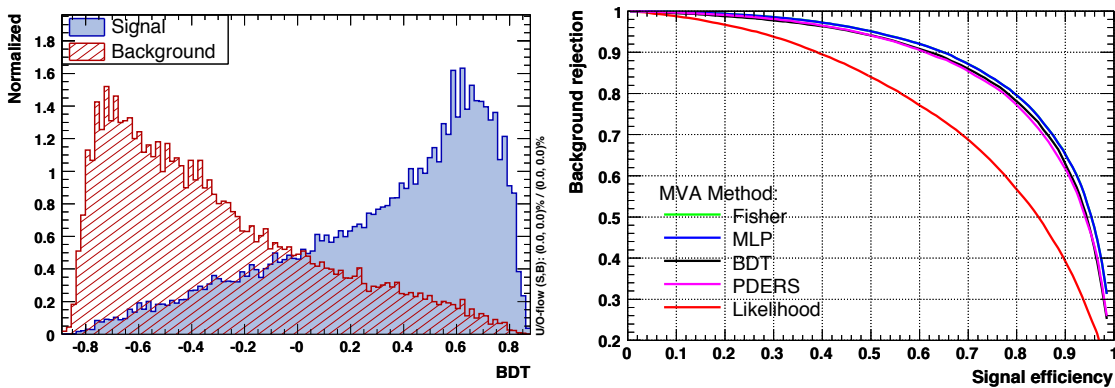
Figure 1: Left: an example of the distributions $f_0(s|\theta)$ and $f_1(s|\theta)$ when the classifier s is a boosted-decision tree (BDT). Right: the corresponding ROC curve (right) for this and other classifiers. Figures taken from TMVA manual.

These steps lead to a subsequent statistical analysis where one observes in data $\{x_e\}$, where e is an event index running from 1 to n. For each event, the classifier is evaluated and one performs inference on a parameter μ related to the presence of the signal contribution. In particular, one forms the statistical model

$$p(\{x_e\}\,|\,\mu, \theta) = \prod_{e=1}^{n} \left[\, \mu f_1(s(x_e)\,|\,\theta) + (1 - \mu)\, f_0(s(x_e)\,|\,\theta) \,\right] \,, \tag{1}$$

where $\mu = 0$ is the null (background-only) hypothesis and $\mu > 0$ is the alternate (signal-plus-background) hypothesis.[1] Typically, we are interested in inference on μ and θ are nuisance parameters; though, sometimes θ may include some components that we are also wish to infer (like the mass of a new particle that affects the distribution x for the signal events).

1.2 Comments on typical usage of machine learning in HEP

Nuisance parameters are an after thought in the typical usage of machine learning in HEP. In fact, most machine learning discussions would only consider $f_0(x)$ and $f_1(x)$. However, as experimentalists we know that we must account for various forms of systematic uncertainty, parametrized by θ. In practice, we take the classifier as fixed and then propagate uncertainty through the classifier as in Eq. 1. Building the distribution $f(s(x)|\theta)$ for values of θ other than the nominal θ_0 used to train the classifier can be thought of as a calibration

---

[1]Sometimes there is an additional Poisson term when expected number of signal and background events is known.

necessary for classical statistical inference; however, this classifier is clearly not optimal for $\theta \neq \theta_0$.

1.3 A more powerful approach

The idea here is to combine the calibration of the distributions of the classifier output and a more optimal family of classifiers $s(x; \theta)$. Creating the family of classifiers is straight forward, one simply augments the training data with x examples drawn from several values of θ and then includes the corresponding value of θ as an input to the classifier. Thus $\{x_e, c_e\} \rightarrow \{x_e, \theta_e, c_e\}$ leading to a parametrized learner $s(x) \rightarrow s(x; \theta)$. This leads to a complication: one does not know the value of θ to use when evaluating the parametrized classifier $s(x; \theta)$. However, when performing statistical inference one will evaluate the likelihood of the data at a given value of θ, so this is a natural choice. Thus one arrives at the generalization of Eq. 1

$$p(\{x_e\} \,|\, \mu, \theta) = \prod_{e=1}^{n} \left[\, \mu f_1(s(x_e; \theta) \,|\, \theta) + (1 - \mu) \, f_0(s(x_e; \theta) \,|\, \theta) \,\right] \;. \tag{2}$$

The construction in Eq. 2 presents a computational challenge as for each value of θ one has a new mapping $s : x \rightarrow \mathbb{R}$. The most naive realization of this would be to generate synthetic data $\{x_e\}$ according to $f_c(x|\theta)$, evaluate $s(x_e; \theta)$, and then use a histogram, kernel density, or other non-parametric density estimation procedure to estimate $f_c(s|\theta)$. This approach is not realistic in situations where $f_c(x|\theta)$ is realized with expensive computer simulations; thus, some interpolation strategy or emulator over a fixed set of parameter points $\{\theta_i\}$ would typically be used.