Data Generation

UVA Spin Physics Group

April 2020

Abstract

The purpose of the project is to develop tools for information extraction. We are at present working on a flexible method of fitting multiple functions with multiple parameters simultaneously over a dynamic range. This provides a way to extract the parameters as well as the trends in parameters for multiple functions that work together to add constraints difficult to impose with standard statistical methods. The first task is to start simple and then add in the complexity gradually. Our first task is to provide a parametrization of the of generic form factor based on *experimental* data from scattering experiments. A critical part of this approach is the data generation which requires producing Monte Carlos data based on the experimental data so that the uncertainties can be propagated accurately to the extracted parameters.

1 Introduction

Here we intend to use ANN to a specific type of optimization task. In the approach, you can do the extraction with any software suite that can do the job. To some extent the philosophy of approach in machine learning is to through as many different types of methods at a problem and see which does the best job.

In the beginning it is suggested to start with deep neural network using TensorFlow but we may discover that there are better options. You can also use whatever computing system you like. One possibility is UVAs research computing cluster:

https://www.rc.virginia.edu/userinfo/rivanna/allocations/

TensorFlow and many other ANN and analysis frameworks are already installed and available.

Another option is using AWS Deep Learning amazon machine images:

https://aws.amazon.com/machine-learning/amis/

2 The Approach

The approach we use starts with a data set that is assumed to be high quality experimental data. We use this data to generate our own Monte Carlos data to feed into our analysis. The steps generally goes something like this:

1. We are give a function F and some outputs of the function. The outputs of this function F are the *experimental* data we start with. We are also given an error associated with each experimental data point. This error can be assumed to be Gaussian. The function F depends on three kinematic values t, Q^2, and x_b that the parameters are dependent on. The free variable in the function F is ϕ.

2. Kinematic values of the first data point are presented to the input-layer of the ANN.

3. These values are propagated through the network according to weights and activation functions. In the first iteration, weights are set to some random values.

4. As a result, the network produces some resulting values of output in its output-layer. The results are the parameters we are extracting which are real valued form factors $\mathcal{H}$, $\mathcal{E}$, and $\tilde{\mathcal{H}}$.

5. These output values are used to calculate the numeric output of F for the given ϕ in the data set. The results are compared to actually *experimental* values, with squared error used for making the standard χ^2 function.

6. The obtained error is then used to modify the network: It is, possibly weighted by the inverse uncertainty of the experimental measurement, propagated backwards through the layers of the network and each weight is adjusted such that this error is decreased.

7. This procedure is then repeated with the next data point, until the whole training set is exhausted.

In our approach it is very important to preserve statistical analysis of uncertainty as much as possible. It should also be clear that the *experimental* data is not used other than to generate your own data, or replica data. This is done by sampling from the experimental data points within the Gaussian error for each point in the set. In this way it is possible to generate extensive data to feed into the ANN. Taking a large number of of replica (N) data sets makes it possible to produce a large resulting family of trained neural networks with outputs $\mathcal{H}_1$, $\mathcal{E}_1$, and $\tilde{\mathcal{H}}_1$,....$\mathcal{H}_N$, $\mathcal{E}_N$, and $\tilde{\mathcal{H}}_N$ defining a provability distribution. The would provide an obvious way to calculate the mean value and variance.

Another important point is to setup your analysis to prevent overfitting. After a certain number of iterations the network will not only describe the general dependence of observables on kinematic variables, but will also adjust to the random fluctuations of data. This unwanted behavior is prevented by the

cross-validation procedure in which initial data is divided into two sets: training set and validation set. Then performance of the network is continuously checked on the validation set. Since this set is not used for training, after the onset of overfitting, the error of the network's description of validation sets will increase. At the onset of this divergence in the mean square error of the validation set from the training data is the cutoff.

3 Data Generation

https://confluence.its.virginia.edu/display/twist/ANN+Fitting+Project

Clone or download from Github ptgroupneuralnet.

In there you should find a text file: DVCS_cross.csv. This is the so called *experimental* data as outlined above. Use this to generate your own replica data sets and then start to run and play with the software.