NNV$_{\mathrm{ub}}$: A Method to Extract Information on Shape Functions Using Artifical Neural Networks

Working Toward an Unbiased Inclusive V_{ub} Extraction

Kristopher J. Healey

Università degli Studi di Torino, INFN Torino

MITP : Challenges in Semileptonic B Decays

April 22, 2015

A Quick Outline

- 1. Brief overview of the V_{ub} extraction method
  [Gambino, Giordano, Ossola, Uraltsevc][GGOU]
  (JHEP 0710:058,2007, arXiv:0707.2493)
- 2. Introduction to Artificial Neural Networks
- 3. Applying ANN to V_{ub} extraction
- 4. Obstacles and Feasibility

Inclusive $B \to X_u \ell \bar{\nu}$ Determination of $|V_{ub}|$

Starting Point is the triple-differential distribution:

$$\frac{d^3\Gamma}{dq^2\, dq_0\, dE_\ell} = \frac{G_F^2 |V_{ub}|^2}{8\pi^3} \Big\{ q^2 W_1 - \left[2E_\ell^2 - 2q_0 E_\ell + \frac{q^2}{2} \right] W_2 + q^2 (2E_\ell - q_0) W_3 \Big\} \times$$
$$\times \theta \left(q_0 - E_\ell - \frac{q^2}{4E_\ell} \right) \, \theta(E_\ell) \, \theta(q^2) \, \theta(q_0 - \sqrt{q^2})$$

Structure Functions in the valid range of the local OPE :

$$W_i(q_0, q^2, \mu) = m_b(\mu)^{n_i} \left[W_i^{pert}(\hat{q}_0, \hat{q}^2, \mu) + W_i^{pow}(\hat{q}_0, \hat{q}^2, \mu) \right]$$

The structure functions are expressed as convolutions with the light-cone distribution function :

$$W_i(q_0, q^2) = m_b^{n_i}(\mu) \int dk_+ \; F_i(k_+, q^2, \mu) \; W_i^{pert} \left[q_0 - \frac{k_+}{2} \left(1 - \frac{q^2}{m_b M_B} \right), q^2, \mu \right]$$

Inclusive $B \to X_u \ell \bar{\nu}$ Determination of $|V_{ub}|$

- Include sub-leading effects to all orders; as a result they are non-universal, with one shape function corresponding to each structure function.
- The k_+^n moments can be computed in the OPE and related to observables $(\mu_{\pi,G}, \rho_{D,LS})$ and to the shape functions defined.

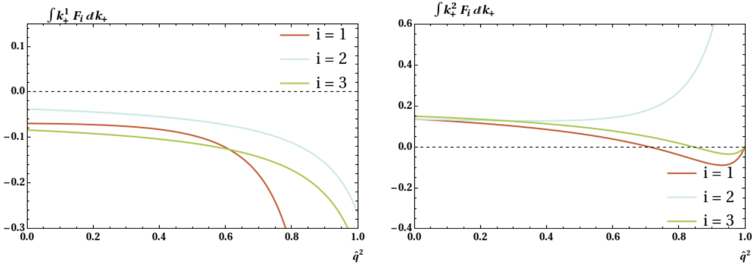
$$\int dq_0 (q_0 - a)^n \, W_i(q_0, q^2) = m_b^{n_i} \int dk_+ \, F_i(k_+, q^2) \int dq_0 (q_0 - a)^n \, W_i^{(0)}(q_0 - k_+ \frac{\Delta}{2}, q^2)$$

LHS calculated inc. power corrections; RHS simplified and can be rewritten as:

$$\int dk_+ \; k_+^n \; F_i(k_+, q^2) = \left(\frac{2\, m_b}{\Delta}\right)^n \left[\delta_{n0} + \frac{I_i^{(n),pow}}{I_i^{(0),\mathrm{tree}}}\right]$$

Shape Functions

We know the q^2 dependence of the first few moments of each structure function from the OPE :

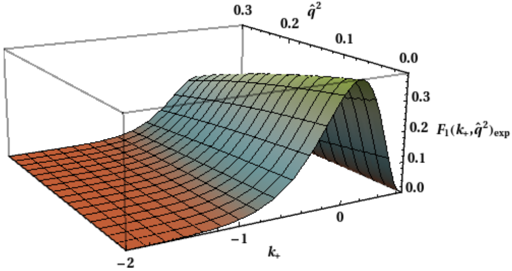


We expect a breakdown at high q^2

Previous Modeling of Shape Functions

A few ansatz for $F_i(k_+)$:

- Exponential suppression at $-k_+$
- Positive Definite
- Explicit $\theta(\bar{\Lambda} - k_+)$
- Non-universal $(F_{1,2,3})$



- Modeled with mutliple 2-parameter forms :

$$F_i(k_+) = N_i\,(\bar{\Lambda} - k_+)^{a_i}\,e^{b_i\,k_+}\;\theta(\bar{\Lambda} - k_+) \qquad (\textit{exponential})$$

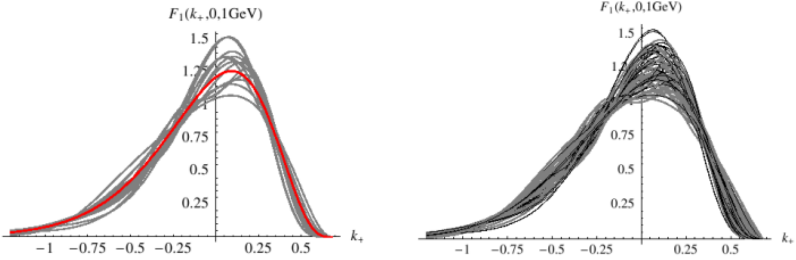
$$F_i(k_+) = N_i\,(\bar{\Lambda} - k_+)^{a_i}\,e^{-b_i\,(\bar{\Lambda}-k_+)^2}\;\theta(\bar{\Lambda} - k_+) \qquad (\textit{gaussian})$$

$$F_i(k_+) = N_i e^{-a_i\left(\bar{\Lambda}-k_+ + \frac{b_i}{\bar{\Lambda}-k_+}\right)^2}\;\theta(\bar{\Lambda} - k_+) \qquad (\textit{roman})$$

- Model dependent, parameters solved by binning for qs, etc.
- also introduces bias, error estimated by varying parameters

Previous Modeling of Shape Functions

Can also modify forms and/or include functional distortions



- Goal of Neural Networks : Remove this model dependence/uncertainty

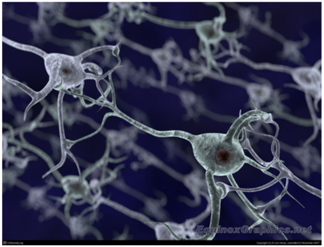
Previous Modeling of Shape Functions

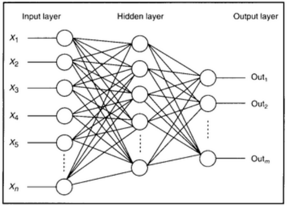
- Two different methods used for tail at high q^2
  - Freezout : Fix q^2 dependence at some cutoff q^{2*}, use in convolutions $q^2 > q^{2*}$
  - Damping of Singularity :

$$\frac{d\Gamma}{d\hat{q}^2} \sim \left\{ \frac{\rho_D^3}{6m_b^3} \left[20\,\hat{q}^6 + 66\,\hat{q}^4 + 48\,\hat{q}^2 + 74 - \frac{96\,(1 - e^{-\frac{(1-\hat{q}^2)^2}{b^2}})}{1-\hat{q}^2} \right] + X_{WA}\,\delta(1-\hat{q}^2) + \ldots \right\}$$

- Neural Network method *may* predict into high q region
- Any experimental results in this region can help train NN

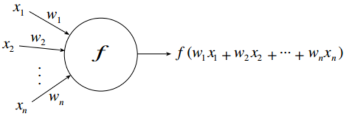
An Introduction to Neural Networks

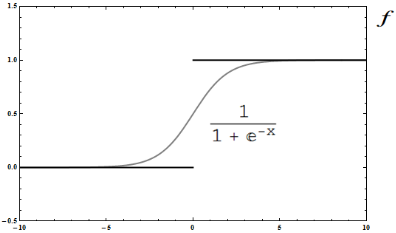




- Trained to reproduce outputs given specific inputs
- Commonly used in pattern recognition (OCR/Facial Recog/etc.)
- Successful in HEP/EXP [NNPDF : Ball, Bertone, Carrazza, Deans, Del Debbio, Forte, Guffanti, Hartland, Latorre, Rojo, Ubiali]
- Provides a predictive, unbiased, and statistically controlled output.
- *Computationally expensive

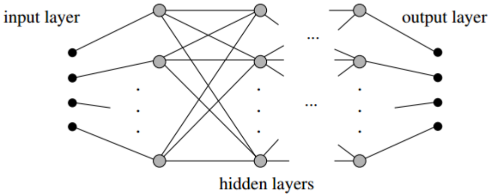
An Introduction to Neural Networks





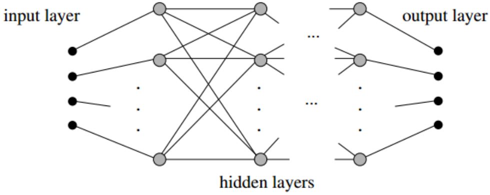
- General Principle : Neurons "Activate" if Σ weighted inputs > 0
- *Use sigmoidal activation function for smoother activation
- Train "weights" to achieve desired response
- Develop a network of Neurons to mimic a continuous function

Feed Forward Neural Networks



- Increased function complexity requires more neurons.
- Proven : any continuous function $\rightarrow \infty$ Neurons in 1 hidden layer
- Multiple hidden layers [2], speeds up training
- Variable number of neurons in each layer [2-5-5-1]

Training Process



NN Adjustable Parameters are the weights connecting neurons.

- 1. Randomize All Weights
- 2. Feed inputs, get outputs
- 3. Use a "Goodness Of Fit" (compare with desired output)
- 4. Adjust weights [Genetic Algorithm/ Backpropagation]
- 5. GOTO 2
- 6. Stop training (N Epochs or validation set required)

Our Training Process

- Three Neural Networks : $F_{1,2,3}(k_+, q^2)$ (2 in/ 1 out)
- Goodness Of Fit from moments:

$$\int dk_+ \; k_+^n \; F_i(k_+, q^2) = \left(\frac{2\, m_b}{\Delta}\right)^n \left[\delta_{n0} + \frac{I_i^{(n),pow}}{I_i^{(0),\mathrm{tree}}}\right]$$

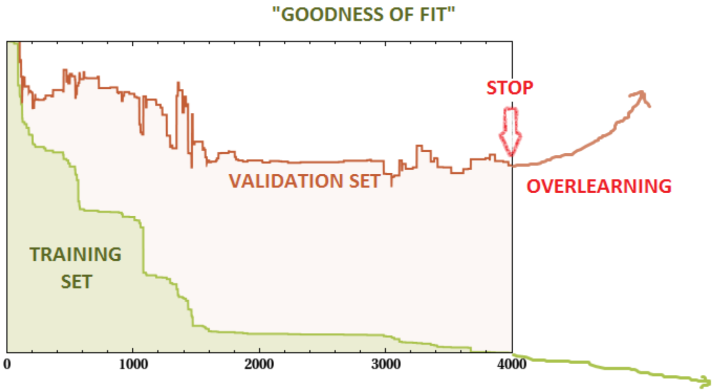
Generated over a sampling of $q^2 \in [0, 13\mathrm{GeV}^2]$
*Also experiment input ($< M_X^n >$ spectra)

- Embedded in V_{ub} fitting code (all available pow+pert corr)
- Validation set for finishing from $q^2 \in [0, 13\mathrm{GeV}^2]$

Training Methods

- Back propagation :
  Work backwards from output finding how each weight affects error, and adjust in the correct direction every at every node.
- Genetic Algorithm :
  Randomly (MC) choose weights and vary them to create new children. Vary many times, choosing "best child", it becomes parent, repeat.
  - Randomly initialize $N_P[10]$ parents
  - Randomly choose one parent network
    Vary $n = (1..4)$ nodes. Save 1000 children.
  - Choose 10 best, replace parents. End of Epoch.
  - 1000+ Epochs (NNPDF - 1000x10000)
  - Allows complete probing of the space (avoids local minima)

Stopping



- If run too long will overlearn the data (no predictive power)
- Stop when Error[Validation] Diverges
- Programmatically : Automatically stop when test error increases

Error Analysis

- Need full correlation matrix/dependence : F_i vs $(m_b, \mu_{\pi,G}, \rho_{D,LS})$
- Create a space of "replicas" with varying input parameters
- For full description see [Rojo : 0607122]
- Very computationally expensive
- Can speed up by assuming shape conditions
  $F_i = e^{k_+} NN_i(k_+)$ or even $F_i = a_0 e^{a_1 k_+}(k_+ - \Lambda)^{a_2} NN_i(k_+)$
  - More computation time needed to validate range of validity. (All NN's should converge to similar forms regardless of initial shapes)

OPTIMIZATION

- Need to vary nodes/layer (requires entire training)
- Need to vary "random" variation in weights (learning parameter)
- Need to vary over number of children per epoch

Benefits and Issues

- Neural Networks are a powerful tool for fitting shape functions without modeling the form
- High-q^2 tail (beyond OPE)
  - As $q^2 \to m_b^2$, process is no longer hard.
  - WA/Higher Dimensional Operators appear.
  - Could predict into high-q^2 region if trained on relative exp. data
- Need to complete full error analysis before any meaningful results can be obtained.