Polarized DIS Structure Functions and Polarized PDFs from Neural Networks

Alberto Guffanti

University of Edinburgh

In collaboration with:

L. Del Debbio (Edinburgh) and A. Piccione (Torino)

SPIN 2006

The 17th International Spin Physics Symposium

Kyoto Japan Oct. 2-7 '06

Outline

1. Introduction
2. The Neural Network Approach
3. g_1 from Neural Networks

Motivation

- The growth in statistics and improving precision of polarized data allow us to reduce errors in the extraction of polarized parton distributions.
- Experience in unpolarized case showed that sometimes a discrepancy between theory and experiments is not a signal of ”new physics” but ”old physics” we do not fully understand
  - High-E_T jets at the Tevatron,
  - B production,
  - ...
- Need for faithful estimation of errors on polarized parton distribution functions (PDF).

Problem
Faithful estimation of errors

- Single quantity: 1-σ error
- Multiple quantities: 1-σ contours
- Function: need an ”error band” in the space of functions (*i.e.* the probability density $\mathcal{P}[f]$ in the space of functions $f(x)$)

Expectation values are Functional integrals

$$\langle \mathcal{F}[f(x)] \rangle = \int \mathcal{D}f \mathcal{F}[f(x)] \mathcal{P}[f(x)]$$

Problem
Faithful estimation of errors

- Single quantity: 1-σ error
- Multiple quantities: 1-σ contours
- Function: need an "error band" in the space of functions (*i.e.* the probability density $\mathcal{P}[f]$ in the space of functions $f(x)$)

Expectation values are Functional integrals

$$\langle \mathcal{F}[f(x)] \rangle = \int \mathcal{D}f \mathcal{F}[f(x)] \mathcal{P}[f(x)]$$

Determine an infinite-dimensional object (a function) from a finite set of data points ... mathematically ill-defined problem.

Solution
Standard Approach

- Introduce a simple functional form with enough free parameters

$$q(x, Q^2) = x^\alpha (1-x)^\beta P(x; \lambda_1, ..., \lambda_n).$$

- Fit parameters minimizing χ^2.

Solution
Standard Approach

- Introduce a simple functional form with enough free parameters

$$q(x, Q^2) = x^{\alpha}(1-x)^{\beta} P(x; \lambda_1, ..., \lambda_n).$$

- Fit parameters minimizing χ^2.

Open problems:

- Error propagation from data to parameters and from parameters to observables is not trivial.
- Theoretical bias due to the chosen parametrization is difficult to assess.

The Strategy
Bayesian Inference Method

[Giele, Keller and Kosower, hep-ph/0104052]

- Generate a Monte-Carlo sampling of the function space according to a *reasonable* prior distribution.
- Compute observables as functional integrals with the probability measure defined by the sampling.
- Update probability using Bayesian inference on the MC sample.
- Iterate until convergence is reached.

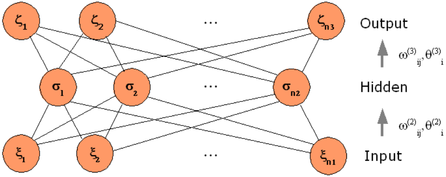
The originally ”infinite dimensional” problem is made finite by choosing a prior, but the final result should not depend on this choice.

The Neural Network Approach

1. Generate N_{rep} Monte-Carlo replicas of the experimental data.
2. *Train* a Neural Network on any of the replicas, defining a probability density on the space of the observable.
3. Expectation values for observables are sums over nets

$$\langle \mathcal{F}[g_1(x,Q^2)] \rangle = \frac{1}{N_{rep}} \sum_{k=1}^{N_{rep}} \mathcal{F}\left(g_1^{(net)(k)}(x,Q^2) \right)$$

Neural Networks



- Neural Networks are a class of algorithms suitable to fit noisy or incomplete data.

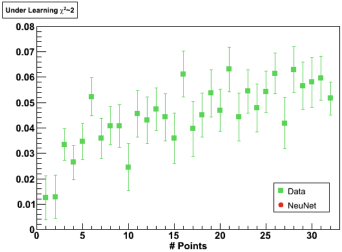
[for HEP applications see ACAT 2005]

- Any continuous function can be approximated with neural network with one internal layer and non-linear neuron activation function.

[G. Cybenko (1989)]

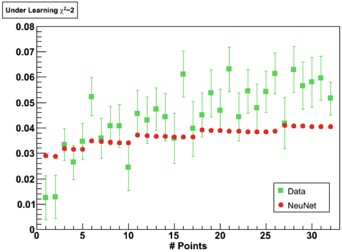
Neural Networks
Training

- Set network parameters randomly.
- If there are different inputs, normalize them.
- Define a *figure of merit* E (*i.e.* χ^2).
- Define a criterion of convergence (*i.e.* $\chi^2 \sim 1$).



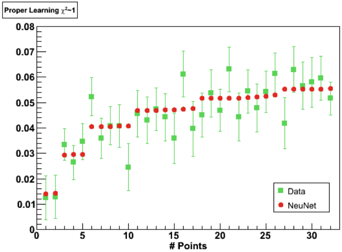
Neural Networks
Training

- Set network parameters randomly.
- If there are different inputs, normalize them.
- Define a *figure of merit* *E* (*i.e.* χ^2).
- Define a criterion of convergence (*i.e.* $\chi^2 \sim 1$).



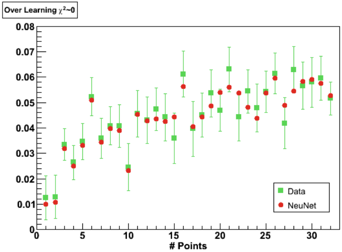
Neural Networks
Training

- Set network parameters randomly.
- If there are different inputs, normalize them.
- Define a *figure of merit* E (*i.e.* χ^2).
- Define a criterion of convergence (*i.e.* $\chi^2 \sim 1$).



Neural Networks
Training

- Set network parameters randomly.
- If there are different inputs, normalize them.
- Define a *figure of merit* *E* (*i.e.* χ^2).
- Define a criterion of convergence (*i.e.* $\chi^2 \sim 1$).



Neural Networks
Training Methods

- Back Propagation
  1. Set network parameters randomly.
  2. Present and input and compute the output.
  3. Evaluate χ^2.
  4. Modify the weights according to

$$\omega_{ij}^{(l)} \longrightarrow \omega_{ij}^{(l)} - \eta \frac{\partial \chi^2}{\partial \omega_{ij}^{(l)}}$$

  5. Back to 2, until stability in χ^2 is reached.

Neural Networks
Training Methods

- Back Propagation
  1. Set network parameters randomly.
  2. Present and input and compute the output.
  3. Evaluate χ^2.
  4. Modify the weights according to
  $$\omega_{ij}^{(l)} \to \omega_{ij}^{(l)} - \eta \frac{\partial \chi^2}{\partial \omega_{ij}^{(l)}}$$
  5. Back to 2, until stability in χ^2 is reached.
- Genetic Algorithm
  1. Set network parameters randomly.
  2. Make *clones* of the set of parameters.
  3. Mutate each clone.
  4. Evaluate χ^2 for all the clones.
  5. Select the clone that has the lowest χ^2.
  6. Back to 2, until stability in χ^2 is reached.

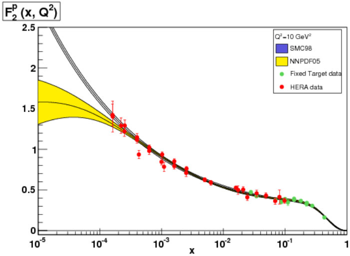
g_1 from Neural Networks
Data & MC replicas

Data in the present analysis

Experiment	x range	Q^2 range	# of points
E80	0.110 – 0.510	1.02 – 4.09	7
E130	0.380 – 0.580	1.45 – 1.70	5
EMC	0.015 – 0.466	3.50 – 29.5	10
SMC	0.005 – 0.480	1.30 – 58.0	12
E143	0.031 – 0.749	1.27 – 9.52	28
E155	0.015 – 0.750	1.22 – 34.72	24
HERMES	0.023 – 0.660	0.92 – 7.36	20
Total	0.005 – 0.750	0.92 – 58.0	106

- Only statistical and (when available) uncorrelated systematic errors.
- g_1 is extracted from A_1 data using the NNPDF parmetrization of F_2

[Del Debbio *et al.*, hep-ph/0501067]

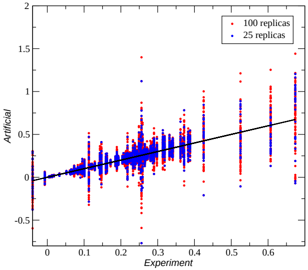


g_1 from Neural Networks
Data & MC replicas

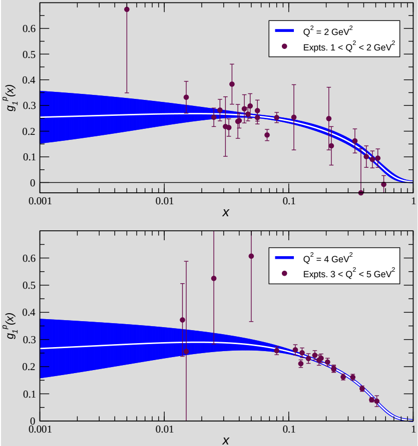
- Generate N_{rep} Monte-Carlo replicas of the data according to:

$$g_1^{(art),i}(x, Q^2) = g_1^{(exp)}(x, Q^2) + r_i \sigma_t^i$$

- Validate Monte-Carlo replicas against experimental data. (statistical estimators, faithful representation of uncertainties, convergence rate increasing N_{rep})



g_1^P from Neural Networks
Preliminary Fit

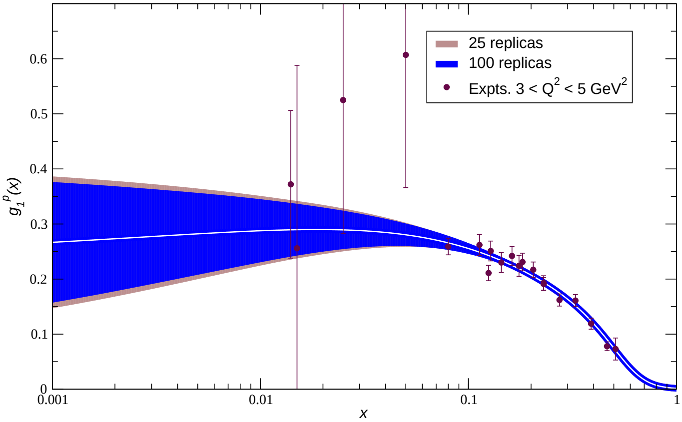


Network architecture: 4-3-1
$N_{rep} = 100$
Training: *Genetic Algorithm*

Input: x, Q^2, $\ln x$, $\ln Q^2$
Output: $g_1(x, Q^2)$

g_1^P from Neural Networks

Preliminary Fit



Summary

Where we are ...

- We derived a parametrization of the structure function g_1^p with faithful error estimation, based on Monte-Carlo techniques and Neural Networks.
- It could be used as input in a (Factorization-)Scheme-Invariant analysis to determine α_s. ([Blüemlein and Böttcher])
- Inclusion of new data and finalization of the analysis before the end of the year.

Instead of Conclusions
The way to NN Polarized PDFs

The general strategy is the same as in the structure function case but

- Each PDF is parametrized by a different neural network
  $(\Delta u_v^{(net)}(x, Q_0^2), \Delta d_v^{(net)}(x, Q_0^2), \Delta \overline{q}^{(net)}(x, Q_0^2), \Delta g^{(net)}(x, Q_0^2))$.
- The training of neural networks on experimental data involves DGLAP evolution and convolution with Wilson Coefficients.
- Include other observables ($g_1^{d,n}$, SIDIS, polarized Drell-Yan).