Introduction to Nonlinear Statistics and Neural Networks

Vladimir Krasnopol'sky
NCEP/NOAA & ESSIC/UMD

http://polar.ncep.noaa.gov/mmab/people/kvladimir.html
Outline

- Introduction: Regression Analysis
- Regression Models (Linear & Nonlinear)
- NN Tutorial
- Some Atmospheric & Oceanic Applications
  - Accurate and fast emulations of model physics
  - NN Multi-Model Ensemble
- How to Apply NNs
- Conclusions
• Problems for Classical Paradigm:
  – Nonlinearity & Complexity
  – High Dimensionality - Curse of Dimensionality

• New Paradigm under Construction:
  – Is still quite fragmentary
  – Has many different names and gurus
  – NNs are one of the tools developed inside this paradigm
Problem:
Information exists in the form of finite sets of values of several *related variables* (sample or training set) – a part of the population:

\[
\mathcal{N} = \{(x_1, x_2, ..., x_n)_p, z_p\}_{p=1,2,...,N}
\]

- \(x_1, x_2, ..., x_n\) - independent variables (accurate),
- \(z\) - response variable (may contain observation errors \(\varepsilon\))

We want to find responses \(z'_q\) for another set of independent variables

\[
\mathcal{N}' = \{(x'_1, x'_2, ..., x'_n)_q\}_{q=1,..,M}
\]

\(\mathcal{N}' \notin \mathcal{N}\)
Find mathematical function $f$ which describes this relationship:

1. Identify the unknown function $f$
2. Imitate or emulate the unknown function $f$
Regression Analysis (2): A Generic Solution

- The effect of independent variables on the response is expressed mathematically by the regression or response function $f$:
  $$y = f(x_1, x_2, ..., x_n; a_1, a_2, ..., a_q)$$
- $y$ - dependent variable
- $a_1, a_2, ..., a_q$ - regression parameters (unknown!)
- $f$ - the form is usually assumed to be known
- Regression model for observed response variable:
  $$z = y + \varepsilon = f(x_1, x_2, ..., x_n; a_1, a_2, ..., a_q) + \varepsilon$$
- $\varepsilon$ - error in observed value $z$
Regression Models (1):
Maximum Likelihood

• Fischer suggested to determine unknown regression parameters \( \{a_i\}_{i=1,\ldots,q} \) maximising the functional:

\[
L(a) = \sum_{p=1}^{N} \ln \left[ \rho(z_p - y_p) \right]; \quad \text{where } y_p = f(x_i)
\]

here \( \rho(\varepsilon) \) is the probability density function of errors \( \varepsilon_i \).

• In a case when \( \rho(\varepsilon) \) is a normal distribution

\[
\rho(z - y) = \alpha \cdot \exp\left(-\frac{(z - y)^2}{\sigma^2}\right)
\]

the maximum likelihood \( \Rightarrow \) least squares

\[
L(a) = \sum_{p=1}^{N} \ln \left[ \alpha \cdot \exp\left(-\frac{(z_p - y_p)^2}{\sigma^2}\right) \right] = A - B \cdot \sum_{p=1}^{N} (z_p - y_p)^2
\]

\[
\max L \Rightarrow \min \sum_{p=1}^{N} (z_p - y_p)^2
\]
Regression Models (2): Method of Least Squares

• To find unknown regression parameters \{a_i\}_{i=1,2,...,q}, the method of least squares can be applied:

\[ E(a_1, a_2, ..., a_q) = \sum_{p=1}^{N} (z_p - y_p)^2 = \sum_{p=1}^{N} \left[ z_p - f((x_1, ..., x_n)_p; a_1, a_2, ..., a_q) \right]^2 \]

• \( E(a_1, ..., a_q) \) - error function = the sum of squared deviations.

• To estimate \{a_i\}_{i=1,2,...,q} => minimize \( E \) => solve the system of equations:

\[ \frac{\partial E}{\partial a_i} = 0; \quad i = 1,2, ..., q \]

• Linear and nonlinear cases.
Regression Models (3):
Examples of Linear Regressions

- **Simple** Linear Regression:
  \[ z = a_0 + a_1 x_1 + \varepsilon \]

- **Multiple** Linear Regression:
  \[ z = a_0 + a_1 x_1 + a_2 x_2 + \ldots + \varepsilon = a_0 + \sum_{i=1}^{n} a_i x_i + \varepsilon \]

- **Generalized** Linear Regression:
  \[ z = a_0 + a_1 f_1(x_1) + a_2 f_2(x_2) + \ldots + \varepsilon = a_0 + \sum_{i=1}^{n} a_i f_i(x_i) + \varepsilon \]
  - **Polynomial** regression, \( f_i(x) = x^i \),
    \[ z = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + \varepsilon \]
  - **Trigonometric** regression, \( f_i(x) = \cos(ix) \)
    \[ z = a_0 + a_1 \cos(x) + a_1 \cos(2x) + \ldots + \varepsilon \]
Regression Models (4): Examples of Nonlinear Regressions

• Response Transformation Regression:
  \[ G(z) = a_0 + a_1 x_1 + \varepsilon \]
  
  • Example:
    \[ z = \exp(a_0 + a_1 x_1) \]
    \[ G(z) = \ln(z) = a_0 + a_1 x_1 \]

• Projection-Pursuit Regression:

  \[ y = a_0 + \sum_{j=1}^{k} a_j f \left( \sum_{i=1}^{n} \Omega_{ji} x_i \right) \]

  • Example:
    \[ z = a_0 + \sum_{j=1}^{k} a_j \tanh(b_j + \sum_{i=1}^{n} \Omega_{ji} x_i) + \varepsilon \]

  Free nonlinear parameters
NN Tutorial:  
*Introduction to Artificial NNs*

- **NNs as Continuous Input/Output Mappings**
  - Continuous Mappings: definition and some examples
  - NN Building Blocks: neurons, activation functions, layers
  - Some Important Theorems
- **NN Training**
- **Major Advantages of NNs**
- **Some Problems of Nonlinear Approaches**
Mapping
Generalization of Function

• **Mapping:** A *rule of correspondence established between vectors in vector spaces* $\mathbb{R}^n$ and $\mathbb{R}^m$ *that associates each vector* $X$ *of a vector space* $\mathbb{R}^n$ *with a vector* $Y$ *in another vector space* $\mathbb{R}^m$.

\[
Y = F(X) \\
X = \{x_1, x_2, \ldots, x_n\}, \in \mathbb{R}^n \neq \left\{ \begin{array}{c} y_1 = f_1(x_1, x_2, \ldots, x_n) \\
y_2 = f_2(x_1, x_2, \ldots, x_n) \\
\vdots \\
y_m = f_m(x_1, x_2, \ldots, x_n) \end{array} \right\} \\
Y = \{y_1, y_2, \ldots, y_m\}, \in \mathbb{R}^m
\]
## Mapping \( Y = F(X) \): examples

- **Time series prediction:**
  \[
  X = \{x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-n}\}, \quad \text{- Lag vector}
  \]
  \[
  Y = \{x_{t+1}, x_{t+2}, \ldots, x_{t+m}\}, \quad \text{- Prediction vector}
  \]
  (Weigend & Gershenfeld, “Time series prediction”, 1994)

- **Calculation of precipitation climatology:**
  \[
  X = \{\text{Cloud parameters, Atmospheric parameters}\}
  \]
  \[
  Y = \{\text{Precipitation climatology}\}
  \]
  (Kondragunta & Gruber, 1998)

- **Retrieving surface wind speed over the ocean from satellite data (SSM/I):**
  \[
  X = \{\text{SSM/I brightness temperatures}\}
  \]
  \[
  Y = \{W, V, L, SST\}
  \]
  (Krasnopolsky, et al., 1999; operational since 1998)

- **Calculation of long wave atmospheric radiation:**
  \[
  X = \{\text{Temperature, moisture, } O_3, \ CO_2, \ \text{cloud parameters profiles, surface fluxes, etc.}\}
  \]
  \[
  Y = \{\text{Heating rates profile, radiation fluxes}\}
  \]
  (Krasnopolsky et al., 2005)
NN - Continuous Input to Output Mapping

**Multilayer Perceptron**: Feed Forward, Fully Connected

\[ Y = F_{NN}(X) \]

**Jacobian!**

\[
Y = a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot t_j = a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot \phi(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i) = \\
= a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot \tanh(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i) \quad q = 1,2,...,m
\]

\[ t_j = \phi(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i) = \\
= \tanh(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i) \]

3/9/2011

Meto 630; V.Krasnopolsky, "Nonlinear Statistics and NNs"
Some Popular Activation Functions

- **tanh(x)**
- **Sigmoid, \((1 + \exp(-x))^{-1}\)**
- **Hard Limiter**
- **Ramp Function**
NN as a Universal Tool for Approximation of Continuous & Almost Continuous Mappings

Some Basic Theorems:

Any function or mapping \( Z = F (X) \), continuous on a compact subset, can be approximately represented by a \( p \) (\( p \geq 3 \)) layer \( \text{NN in the sense of uniform convergence} \) (e.g., Chen & Chen, 1995; Blum and Li, 1991, Hornik, 1991; Funahashi, 1989, etc.)

The error bounds for the uniform approximation on compact sets (Attali & Pagès, 1997):

\[
\| Z - Y \| = \| F (X) - F_{NN} (X) \| \sim C/k
\]

\( k \) -number of neurons in the hidden layer

\( C \) – does not depend on \( n \) (avoiding \textbf{Curse of Dimensionality!})
NN training (1)

• For the mapping $Z = F(X)$ create a training set - set of matchups $\{X_i, Z_i\}_{i=1,...,N}$, where $X_i$ is input vector and $Z_i$ - desired output vector.

• Introduce an error or cost function $E$: 

$$E(a,b) = \|Z - Y\| = \sum_{i=1}^{N} |Z_i - F_{NN}(X_i)|,$$

where $Y = F_{NN}(X)$ is neural network.

• Minimize the cost function: $\min\{E(a,b)\}$ and find optimal weights $(a_0, b_0)$.

• Notation: $W = \{a, b\}$ - all weights.
NN Training (2)

One Training Iteration

Training Set

Input

Output

\[ \{W\} \]

\[ X \]

\[ Y \]

\[ Z \]

Desired Output

Error

\[ E = ||Z - Y|| \]

Weight Adjustments

\[ \Delta W \]

BP

End Training

Yes

No

End Training

3/9/2011

Meto 630; V.Krasnopolsky, "Nonlinear Statistics and NNs"
Backpropagation (BP) Training Algorithm

- BP is a simplified steepest descent:
  \[
  \Delta W = -\eta \frac{\partial E}{\partial W}
  \]
  where \( W \) - any weight, \( E \) - error function, \( \eta \) - learning rate, and \( \Delta W \) - weight increment.

- Derivative can be calculated analytically:
  \[
  \frac{\partial E}{\partial W} = -2 \sum_{i=1}^{N} [Z_i - F_{NN}(X_i)] \cdot \frac{\partial F_{NN}(X_i)}{\partial W}
  \]

- Weight adjustment after \( r \)-th iteration:
  \[
  W^{r+1} = W^r + \Delta W
  \]

- BP training algorithm is robust but slow.
Generic Neural Network
FORTRAN Code:

DATA W1/.../, W2/.../, B1/.../, B2/.../, A/.../, B/.../ ! Task specific part

!--------------------------------------------------------------

DO K = 1,OUT

!

DO I = 1, HID
X1(I) = tanh(sum(X * W1(:,I) + B1(I))
ENDDO ! I

! X2(K) = tanh(sum(W2(:,K)*X1) + B2(K))
Y(K) = A(K) * X2(K) + B(K)

! XY = A(K) * (1. -X2(K) * X2(K))
DO J = 1, IN
DUM = sum((1. -X1 * X1) * W1(J,:)* W2(:,K))
DYDX(K,J) = DUM * XY
ENDDO ! J

ENDDO ! K

NN Output
Jacobian
**Major Advantages of NNs:**

- NNs are very **generic, accurate and convenient** mathematical (statistical) models which are able to emulate **numerical model components**, which are complicated nonlinear input/output relationships (continuous or almost continuous mappings).

- NNs avoid **Curse of Dimensionality**

- NNs are **robust** with respect to random noise and fault-tolerant.

- NNs are **analytically differentiable** (training, error and sensitivity analyses): almost free Jacobian!

- NNs emulations are **accurate and fast** but **NO FREE LUNCH**!

- Training is complicated and time consuming nonlinear optimization task; **however, training should be done only once for a particular application!**

- Possibility of online adjustment

- NNs are **well-suited for parallel and vector processing**
NNs & Nonlinear Regressions: Limitations (1)

- Flexibility and Interpolation:

- Overfitting, Extrapolation:
• **Consistency of estimators:** \( \hat{a} \) is a **consistent estimator** of parameter \( A \), if \( \hat{a} \to A \) as the size of the sample \( n \to N \), where \( N \) is the size of the population.

• **For NNs and Nonlinear Regressions** **consistency** can be usually “proven” only **numerically**.

• **Additional independent** data sets are required for test (demonstrating **consistency of estimates**).
ARTIFICIAL NEURAL NETWORKS: BRIEF HISTORY

• 1943 - McCulloch and Pitts introduced a model of the neuron

Modeling the single neuron

• 1962 - Rosenblat introduced the one layer "perceptrons", the model neurons, connected up in a simple fashion.

• 1969 - Minsky and Papert published the book which practically “closed the field”
ARTIFICIAL NEURAL NETWORKS: BRIEF HISTORY

• 1986 - Rumelhart and McClelland proposed the "multilayer perceptron" (MLP) and showed that it is a perfect application for parallel distributed processing.

The multilayer perceptron

• From the end of the 80's there has been explosive growth in applying NNs to various problems in different fields of science and technology
Atmospheric and Oceanic NN Applications

• Satellite Meteorology and Oceanography
  – Classification Algorithms
  – Pattern Recognition, Feature Extraction Algorithms
  – Change Detection & Feature Tracking Algorithms
  – Fast Forward Models for Direct Assimilation
  – Accurate Transfer Functions (Retrieval Algorithms)

• Predictions
  – Geophysical time series
  – Regional climate
  – Time dependent processes

• NN Ensembles
  – Fast NN ensemble
  – Multi-model NN ensemble
  – NN Stochastic Physics

• Fast NN Model Physics

• Data Fusion & Data Mining

• Interpolation, Extrapolation & Downscaling

• Nonlinear Multivariate Statistical Analysis

• Hydrological Applications
Developing Fast NN Emulations for Parameterizations of Model Physics

Atmospheric Long & Short Wave Radiations
General Circulation Model

The set of conservation laws (mass, energy, momentum, water vapor, ozone, etc.)

- **First Principles/Prediction 3-D Equations on the Sphere:**

\[
\frac{\partial \psi}{\partial t} + D(\psi, x) = P(\psi, x)
\]

- \( \psi \) - a 3-D prognostic/dependent variable, e.g., temperature
- \( x \) - a 3-D independent variable: \( x, y, z & t \)
- \( D \) - dynamics (spectral or gridpoint)
- \( P \) - physics or parameterization of physical processes (1-D vertical r.h.s. forcing)

- **Continuity Equation**
- **Thermodynamic Equation**
- **Momentum Equations**
General Circulation Model

Physics – P, represented by 1-D (vertical) parameterizations

- Major components of $P = \{R, W, C, T, S\}$:
  - $R$ - radiation (long & short wave processes)
  - $W$ – convection, and large scale precipitation processes
  - $C$ - clouds
  - $T$ – turbulence
  - $S$ – surface model (land, ocean, ice – air interaction)

- Each component of $P$ is a 1-D parameterization of complicated set of multi-scale theoretical and empirical physical process models simplified for computational reasons

- $P$ is the most time consuming part of GCMs!
Current NCAR Climate Model (T42 x L26): ~ 3° x 3.5°

Near-Term Upcoming Climate Models (estimated) : ~ 1° x 1°
Generic Situation in Numerical Models
Parameterizations of Physics are Mappings

\[ Y = F(X) \]
Generic Solution – “NeuroPhysics”
Accurate and Fast NN Emulation for Physics Parameterizations

Learning from Data

GCM

Original Parameterization

\[ F_{NN} \]

Training Set

\[ \{X_i, Y_i\}, \ldots \]

\[ \forall X_i \in D_{phys} \]

NN Emulation

\[ F_{NN} \]

NN Emulation

\[ X \rightarrow F_{NN} \rightarrow Y \]

\[ \text{Set} \]

\[ \ldots, \{X_i, Y_i\}, \ldots \]

\[ \forall X_i \in D_{phys} \]

\[ X \rightarrow F_{NN} \rightarrow Y \]
NN for NCAR CAM Physics

**CAM Long Wave Radiation**

- **Long Wave Radiative Transfer:**
  \[
  F^\downarrow(p) = B(p_t) \cdot \varepsilon(p_t, p) + \int_{p_t}^{p} \alpha(p_t, p) \cdot dB(p')
  \]
  \[
  F^\uparrow(p) = B(p_s) - \int_{p}^{p_s} \alpha(p, p') \cdot dB(p')
  \]
  \[
  B(p) = \sigma \cdot T^4(p) \quad – \text{the Stefan–Boltzman relation}
  \]

- **Absorptivity & Emissivity (optical properties):**
  \[
  \alpha(p, p') = \frac{\int_{0}^{\infty} \{dB_v(p')/dT(p')\} \cdot (1 - \tau_v(p, p')) \cdot dv}{dB(p)/dT(p)}
  \]
  \[
  \varepsilon(p_t, p) = \frac{\int_{0}^{\infty} B_v(p_t) \cdot (1 - \tau_v(p_t, p)) \cdot dv}{B(p_t)}
  \]
  \[
  B_v(p) \quad – \text{the Plank function}
  \]
The Magic of NN Performance

Input/Output Dependency:  \( Y = F(X) \)

NN Emulation of Input/Output Dependency:
\[ Y_{NN} = F_{NN}(X) \]

Input/Output Dependency:  \( \{X_i, Y_i\}_{i=1}^{N} \)

Mathematical Representation of Physical Processes

Original Parameterization

\[ F^+(p) = B(p) \cdot \varepsilon(p,p) + \int_{p'} \alpha(p,p') \cdot dB(p') \]

\[ F^-(p) = B(p) - \int_{p'} \alpha(p',p) \cdot dB(p') \]

\[ B(p) = \sigma \cdot T^4(p) \quad \text{the Stefan–Boltzmann relation} \]

\[ \alpha(p,p') = \frac{\int dB(p'/dT(p')) \cdot (1 - \tau_{e}(p,p')) \cdot dv}{dB(p)/dT(p)} \]

\[ \varepsilon(p,p) = \frac{\int B_e(p) \cdot (1 - \tau_e(p,p)) \cdot dv}{B(p)} \]

\[ B_e(p) \quad \text{the Plank function} \]
Neural Networks for NCAR (NCEP) LW Radiation
NN characteristics

- **220 (612 for NCEP) Inputs:**
  - **10 Profiles:** temperature, humidity, ozone, methane, cfc11, cfc12, & N₂O mixing ratios, pressure, cloudiness, emissivity
  - **Relevant surface characteristics:** surface pressure, upward LW flux on a surface - \textit{flwupcgs}

- **33 (69 for NCEP) Outputs:**
  - Profile of heating rates (26)

- **Hidden Layer:** One layer with 50 to 300 neurons

- **Training:** \textit{nonlinear optimization in the space with dimensionality of 15,000 to 100,000}
  - **Training Data Set:** Subset of about 200,000 instantaneous profiles simulated by CAM for the 1-st year
  - Training time: about 1 to several days (SGI workstation)
  - Training iterations: 1,500 to 8,000

- **Validation on Independent Data:**
  - **Validation Data Set (independent data):** about 200,000 instantaneous profiles simulated by CAM for the 2-nd year
Neural Networks for NCAR (NCEP) SW Radiation
NN characteristics

- **451 (650 NCEP) Inputs:**
  - **21 Profiles:** specific humidity, ozone concentration, pressure, cloudiness, aerosol mass mixing ratios, etc
  - **7 Relevant surface characteristics**

- **33 (73 NCEP) Outputs:**
  - Profile of heating rates (26)
  - 7 LW radiation fluxes: $f_{sns}$, $f_{snt}$, $f_{sdc}$, $sols$, $soll$, $solsd$, $solld$

- **Hidden Layer:** One layer with 50 to 200 neurons

- **Training:** nonlinear optimization in the space with dimensionality of 25,000 to 130,000
  - Training Data Set: Subset of about 200,000 instantaneous profiles simulated by CAM for the 1-st year
  - Training time: about 1 to several days (SGI workstation)
  - Training iterations: 1,500 to 8,000

- **Validation on Independent Data:**
  - Validation Data Set (independent data): about 100,000 instantaneous profiles simulated by CAM for the 2-nd year
### NN Approximation Accuracy and Performance vs. Original Parameterization (on an independent data set)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Model</th>
<th>Bias</th>
<th>RMSE</th>
<th>Mean</th>
<th>σ</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWR (°K/day)</td>
<td>NASA M-D. Chou</td>
<td>1. 10^{-4}</td>
<td>0.32</td>
<td>-1.52</td>
<td>1.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NCEP AER rtm2</td>
<td>7. 10^{-5}</td>
<td>0.40</td>
<td>-1.88</td>
<td>2.28</td>
<td>~ 100 times faster</td>
</tr>
<tr>
<td></td>
<td>NCAR W.D. Collins</td>
<td>3. 10^{-5}</td>
<td>0.28</td>
<td>-1.40</td>
<td>1.98</td>
<td>~ 150 times faster</td>
</tr>
<tr>
<td>SWR (°K/day)</td>
<td>NCAR W.D. Collins</td>
<td>6. 10^{-4}</td>
<td>0.19</td>
<td>1.47</td>
<td>1.89</td>
<td>~ 20 times faster</td>
</tr>
<tr>
<td></td>
<td>NCEP AER rtm2</td>
<td>1. 10^{-3}</td>
<td>0.21</td>
<td>1.45</td>
<td>1.96</td>
<td>~ 40 times faster</td>
</tr>
</tbody>
</table>
Individual Profiles

PRMSE = 0.18 & 0.10 K/day  
PRMSE = 0.11 & 0.06 K/day  
PRMSE = 0.05 & 0.04 K/day
NCAR CAM-2: 50 YEAR EXPERIMENTS
NCEP CFS: 17 YEAR EXPERIMENTS

• CONTROL RUN: the standard NCAR CAM or NCEP CFS versions with the original Radiation (LWR and SWR)

• NN RUN: the hybrid version of NCAR CAM or NCEP CFS with NN emulation of the LWR & SWR
NCAR CAM-2 Zonal Mean U 50 Year Average

(a)– Original LWR Parameterization
(b)- NN Approximation
(c)- Difference (a) – (b), contour 0.2 m/sec
all in m/sec
NCAR CAM-2 Zonal Mean Temperature 50 Year Average

(a)– Original LWR Parameterization
(b)– NN Approximation
(c)– Difference (a) – (b), contour 0.1 K

all in K
Application of the Neural Network Technique to Develop a Nonlinear Multi-Model Ensemble for Precipitations over ConUS
Available data for precipitations over ConUS

- Precipitation forecasts available from 8 operational models:
  - NCEP's mesoscale & global models (NAM & GFS)
  - the Canadian Meteorological Center regional & global models (CMC & CMCGLB)
  - global models from the Deutscher Wetterdienst (DWD)
  - the European Centre for Medium-Range Weather Forecasts (ECMWF) global model
  - the Japan Meteorological Agency (JMA) global model
  - the UK Met Office (UKMO) global model
- Also NCEP Climate Prediction Center (CPC) precipitation analysis is available over ConUS.
Data & Products for Comparisons

• Forecasts:
  – MEDLEY multi-model ensemble: simple average of 8 models (24 hr forecasts)
  – NN multi-model ensemble (experimental, 24 hr forecast)
  – Hydrometeorological Prediction Center (HPC) human 24 hr forecast, produced by human analyst using models, satellite images, and other available data

• Validation: CPC analysis over ConUS
Advantages: better placement of precipitation areas

Disadvantages (because of simple linear averaging) Motivation for NN developments:

• Smoothes, diffuse features, reduces gradients
  – High bias for low level precip – large areas of false low precip
  – Low bias in high level precip – highs smoothed out and reduced
24h Forecast Ending 07/24/2010 at 12Z

Verifying CPC analysis
A NN Multi-Model Ensemble

• Use past data (model forecasts and verifying analysis data) to train NN
  – For NN Inputs: precip amounts (8 model 24 hr forecasts), lat, lon, and day of the year
  – For NN output: CPC verification analysis for the corresponding time

• Data for 2009 have been used for training

\[ \text{NN}_{ens} = a_0 + \sum_{j=1}^{k} a_j \cdot \phi(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i) \quad ; \quad n = 12; \quad k = 7 \]
Sample NN forecast: example 1 (1)
Sample NN forecast: example 1 (2)

Verifying CPC analysis

MEDLEY

NN

HPC
Sample NN forecast: example 2

Verifying CPC analysis

MEDLEY

Sample NN forecast: example 2

Verifying CPC analysis

MEDLEY

Sample NN forecast: example 2

Verifying CPC analysis

MEDLEY
Sample NN forecast: example 3
How to Develop NNs:
An Outline of the Approach (1)

- **Problem Analysis:**
  - Are traditional approaches unable to solve your problem?
    - At all
    - With desired accuracy
    - With desired speed, etc.
  - Are NNs well-suited for solving your problem?
    - Nonlinear mapping
    - Classification
    - Clusterization, etc.
  - Do you have a first guess for NN architecture?
    - Number of inputs and outputs
    - Number of hidden neurons
How to Develop NNs: An Outline of the Approach (2)

- **Data Analysis**
  - How noisy are your data?
    - May change architecture or even technique
  - Do you have enough data?
  - For selected architecture:
    - 1) Statistics => $N^1_A > n_W$
    - 2) Geometry => $N^2_A > 2^n$
    - $N^1_A < N_A < N^2_A$
    - To represent all possible patterns => $N_R$
      - $N_{TR} = \max(N_A, N_R)$
  - Add for test set: $N = N_{TR} \times (1 + \tau); \ \tau > 0.5$
  - Add for validation: $N = N_{TR} \times (1 + \tau + \upsilon); \ \upsilon > 0.5$
How to Develop NNs: An Outline of the Approach (3)

• **Training**
  – Try different initializations
  – If results are not satisfactory, then goto Data Analysis or Problem Analysis

• **Validation (must for any nonlinear tool!)**
  – Apply trained NN to independent validation data
  – If statistics are not consistent with those for training and test sets, go back to Training or Data Analysis
Conclusions

• There is an obvious trend in scientific studies:
  – From simple, linear, single-disciplinary, low dimensional systems
  – To complex, nonlinear, multi-disciplinary, high dimensional systems

• There is a corresponding trend in math & statistical tools:
  – From simple, linear, single-disciplinary, low dimensional tools and models
  – To complex, nonlinear, multi-disciplinary, high dimensional tools and models

• Complex, nonlinear tools have advantages & limitations: learn how to use advantages & avoid limitations!

• Check your toolbox and follow the trend, otherwise you may miss the train!
Recommended Reading

- **Regression Models:**

- **NNs, Introduction:**

- **NNs, Advanced:**
  - V. Cherkassky and F. Muller, 2007: Learning from Data: Concepts, Theory, and Methods, J. Wiley and Sons, Inc

- **NNs in Environmental Sciences:**