A Short Course in Neural Networks

Ralph Grothmann, PhD
Ralph.Grothmann@siemens.com
Neural Networks @ Siemens: 25 Years of Research, Development, Innovation

Software Environment

Complex Systems

Mathematics of Neural Networks

Process Control

Technical Applications

Forecasting Renewables

for Neural Networks

Neural Networks

Risk Analysis

Price Forecasting

Economical Applications

High Performance Computing

Uncertainty Analysis

Prior Rule Information

State Space Modeling

Scripted Modeling

Decision Support

Learning Algorithms

Optimal Control

Topology Design

Economical Applications

Power Mgmt. for HEVs

Energy Mgt. for HEVs

Energy Price Forecasts

LME Copper Forecasts

Clinical Trial Analysis

Quality Surveillance

Process Surveillance

Customer Relation Mgt.

Scripted Modeling

High Performance Computing

Uncertainty Analysis

Optimal Control

Topology Design

Learning Algorithms

Decision Support

Energy Mgmt. for HEVs

Global Footprint Simulator

Global Footprint Simulator

Graphical User Interface

Data processing

Fuzzy Rules

Feedforward Neural Nets

Neuro-Fuzzy Networks

Recurrent Neural Nets

Forecasting & Diagnosis

Forecasting Transfer Losses

Renewables

Renewables

Process Modeling

Demand Forecasting for DB Energie

Load Forecasts for DB Energie

Demand Forecasting

Wind Turbines

Solar Power

Forecasting

Economical Applications

Intelligent Systems & Control

© Siemens AG, Corporate Technology
Distinct to linear superpositions of basis functions, NN are composed substructures.
Mathematical Neural Networks in Nonlinear Regression

**Basics on Neural Networks**

- Calculus
- Linear Algebra
- Neural Networks
- $y = W_2 f(W_1x)$
- Non-linearity
- # variables

**Nonlinear Regression**

Based on data identify an input-output relation

$$y = W_2 f(W_1x)$$

$$\sum_{t=1}^{T} (y_t - y_d)^2 \rightarrow \min_{W_1, W_2}$$

**Existence Theorem:**
(Hornik, Stinchcombe, White 1989)

3-layer neural networks can approximate any continuous function on a compact domain.

NN imply a Correspondence of Equations, Architectures, Local Algorithms.

Page 4
The Curse of Dimensionality in Approximation Theory

The curse of dimensionality in Standard Approximation:

\[ f(x) \approx \sum_{j=1}^{m} v_j b_j(x) \quad \text{with} \quad \|v_j\| \approx c^{\dim(x)} \]

This is a linear superposition of basis functions – their number & the number of parameters increase exponentially with \( \dim(x) \).

Neural Networks escape the curse of dimensionality:

\[ f(x) \approx \sum_{j=1}^{m} v_j b(w_j, x) \quad \text{with} \quad \|v_j, w_j\| \approx \text{Var}(f) \]

The independence of the number of parameters from the input dimension is paid with nonlinear optimization.

Support Vector Machines offer an alternative remedy:

\[ f(x) \approx \sum_{j=1}^{m} v_j b(x - x_j) \quad \text{with} \quad \|v_j\| \approx \|\text{data}\| \& \text{Var}(f) \]

Here we have a linear superposition of basis functions, which are chosen as part of the data - which can be a drawback.
Error Backpropagation - Correspondence between Architecture & Algorithm

\[ \text{target out dev} = 22 \]

\[ \text{dev netin f} = \partial T = 221 W_{\text{dev}} \]

\[ \text{dev netin f} = \partial T = 110 W_{\text{dev}} \]

\[ \text{netin fout} = 221 \]

\[ \text{netin fout} = 011 \]

\[ \text{netin out} = 00 \]

\[ E = \frac{1}{T} \sum_{t=1}^{T} E_t = \frac{1}{T} \sum_{t=1}^{T} (y_t - y_d^t)^2 \]

- By the forward & backward flows, are efficiently computed.

- Because of the local algorithm, we can easily extend the network.

- In case of \( f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \), we get

\[ f'(\text{netin}) = 1 - (f(\text{netin}))^2 = 1 - \text{out}^2 \]

- In case of \( f(z) = \text{logistic}(z) = \frac{1}{1 + e^{-z}} \), we get

\[ f'(\text{netin}) = f(\text{netin})(1 - f(\text{netin})) = \text{out}(1 - \text{out}) \]
Uncertainty in Model Building

- In nonlinear modeling, different weight initializations may end up in different local minima. This differentiation shows up on the training as well as on the test set.
- Large networks may be underdetermined. The random substructures do not cause problems on the training set but cause a differentiation on the test set.
- The over-parameterization is even helpful to lower the local minima problem, but we pay the price by the uncertainty of the generalization behavior.
Backpropagation allows an efficient computation of gradients, but how to do the weight update to get a stable model? Can we use the model to evaluate the data quality and filter corrupted data.

Given an ensemble of models, Occam’s razor defines the best model as the most parsimonious. Bayesian analysis defines the best solution as the average solution of the model ensemble.
Learning Structure from Data - Learning Rules for Stochastic Search

Task: \( E = \frac{1}{T} \sum_{t=1}^{T} E_t = \frac{1}{T} \sum_{t=1}^{T} \left( NN(x_t, w) - y_t^d \right)^2 \Rightarrow \min_w \)

Notation: \( g_t = \frac{\partial E_t}{\partial w} , \quad g = \frac{1}{T} \sum_{t=1}^{T} g_t \)

Steepest descent learning: \( \Delta w = \eta \cdot (-g) \) = step length \cdot search direction

Pattern by pattern learning:

\[
\begin{align*}
\Delta w_t &= -\eta g_t \\
&= -\eta g - \eta (g_t - g) \\
&= \text{steepest descent} + \text{stochastic search}
\end{align*}
\]

Vario Eta Learning:

\[
\Delta w_t = -\eta \frac{g_t}{\sqrt{\frac{1}{T} \sum (g_t - g)^2}}
\]

Vario-Eta is a stochastic approx. of the Newton method
Data meet Structure: The Observer - Observation Dilemma

Psychological Dilemma:
How far should observations determine our picture of the world?
&
How far should our picture of the world evaluate observations?

Technical Dilemma:
How far should observations determine a model?
&
How far should a model evaluate observations?

Use the model to clean the data
Data cleaning implies data uncertainty
Use the data uncertainty to harden the learning

Geometry of Cleaning:

\[
E_{yx} = E_y + E_x = \frac{1}{2} \sum_t \left[ (y_t - y_t^d)^2 + (x_t - x_t^d)^2 \right] \rightarrow \min_{w,x}
\]

Assumption of input uncertainty and a small correction in \( x \) may decrease the error in \( y \) dramatically.
Occam's Razor: Search for a Parsimonious Network

**Advise: Late Stopping & Weight Pruning**

Define a criterion for the weight importance:
\[
\text{test}_w = w^2
\]

**Weight Pruning Procedure:**

1. Train the Neural Network
2. Rank weights by importance
3. Prune lower ranked weights

Procedure is bias free towards linear models

Pruning methods split the training data in learning data & validation data, used in the trace.

Data = learning + validation + generalization

data, available in training
Occam’s Training Procedure for Feedforward Neural Networks

0) Analyze relevant model priors
   Setup a model architecture.

1) Learning to *minimal training error*
   a) Use pattern-by-pattern learning.
   b) Cleaning noise forces training from coarse to detailed data features.

2) Learning / Pruning to minimal test error
   a) Prune weights and relearn iteratively.
   b) Trace the best solution on a test set.
      Unneeded inputs & hiddens are pruned automatically.
      Weight pruning has no prior towards linear models.

3) Evaluate the input – output relationships.
Decreasing Model Uncertainty by Averaging I

The sub-networks learn different solutions of the same task. In case of large averages \((m > 20)\) an equal weighting is superior, in case of small averages it is superior to freeze the subnets and optimize the weighting factors.

\[
E_{\text{aver}} = \frac{1}{T} \sum_{T} \left[ \text{out}_{\text{aver}} - \text{tar} \right]^2
\]

\[
= \frac{1}{T} \sum_{T} \left[ \frac{1}{m} \sum_{i} \text{out}_{i} - \text{tar} \right]^2
\]

\[
= \frac{1}{T} \sum_{i} \left[ \frac{1}{m} \sum_{i} (\text{out}_{i} - \text{tar}) \right]^2
\]

\[
= \frac{1}{m^2} \frac{1}{T} \sum_{i} \sum_{i} (\text{out}_{i} - \text{tar})^2
\]

\[
= \frac{1}{m} \frac{1}{m} \sum_{i} \frac{1}{T} \sum_{i} (\text{out}_{i} - \text{tar})^2
\]

\[
= \frac{1}{m} \text{aver}(E_i)
\]

\[
\frac{1}{T} \sum_{T} (\text{out}_{i} - \text{tar})(\text{out}_{j} - \text{tar}) = 0 \quad \forall i \neq j
\]

(covariance of the errors of the submodels)
Forecasting Expectation & Risk with Ensemble Neural Networks

Following Bayes, the expected value of the forecast is computed as the ensemble average.

Uncertainty shows up because we do not know the true scenario. Stochasticity is not seen as a feature of the real world, but as a consequence of partial observability.
Forecasting by Pattern Recognition versus State Space Modeling

\[ y_{t+1} = h(u_t, u_{t-1}, u_{t-2}, \ldots) \]

\[ y_{t+1} = V \tanh(W x_t + w_0) \]

\[ x_t = (u_t, u_{t-1}, u_{t-2}, \ldots) \]

Taken’s Theorem allows an explicit representation of the internal state by a sequence of past observations, which are used as inputs of a feedforward neural network. Multi-step forecasting is possible by using the computed forecasts as inputs.

\[ s_{t+1} = f(s_t, u_t) \]

\[ y_t = g(s_t) \]

\[ s_{t+1} = \tanh(As_t + Bu_t) \]

\[ y_t = Cs_t \]

Recurrent neural networks try to identify an optimal state representation which acts as a dynamical memory. Multi-step forecasting iterates this implicit state into the future.
Finite unfolding in time transforms time into a spatial architecture. We assume, that $x_t=\text{const}$ in the future.

The analysis of open systems by RNNs allows a decomposition of its autonomous & external driven subsystems.

Long-term predictability depends on the extraction of a strong autonomous subsystem.

$y_{t+1} = f(s_t, u_t)$

$y_t = g(s_t)$

$s_{t+1} = \tanh(As_t + Bu_t)$

$y_t = Cs_t$

$\sum_{t=1}^{T} (y_t - y^d_t)^2 \rightarrow \min_{A,B,C}$

Preprocessing:

$u_t = x_t - x_{t-1}$
Feedforward versus Recurrent Neural Networks Architectures

Unfolding in time allows the representation of a temporal process in form of a recurrent neural network, if the matrices A, B and C are constant over time.

In contrast to a feedforward network, the recurrent structure depends on fewer parameters and provides more gradient information.

Each vertical branch of the recurrent net is a 3-layer MLP.
Modeling in form of open dynamical systems is only consistent if the external drivers are nearly constant from present time on. Otherwise the learning works only for small RNNs because, following the regression paradigm, the optimization finds an intermediate solution between the active & the constant environment.
From Standard to Normalized Recurrent Neural Networks

W.l.o.g. we can model a dynamics by one matrix $A$:

$$
\begin{align*}
\tau \leq t &: \quad s_\tau = \tanh(As_{\tau-1} + \begin{bmatrix} 0 \\ 0 \\ \text{Id} \end{bmatrix} u_\tau) \\
\tau > t &: \quad s_\tau = \tanh(As_{\tau-1}) \\
y_\tau &= \begin{bmatrix} \text{Id} & 0 & 0 \end{bmatrix} s_\tau, \quad \sum_{t=m}^{T-n} \sum_{\tau=t-m}^{t+n} (y_\tau - y^d_\tau)^2 \rightarrow \min_A
\end{align*}
$$

It is philosophically implausible, to model the real world by various matrices $A$, $B$, $C$. 
Modeling the Dynamics of Observables

Inputs and targets are merged to observables. In the net we refer to observations $y_t^d$ and expectations $y_t$ of these observables.

Due to the network design, the input to output relation is delayed.

\[ \tau \leq t : \quad s_\tau = \tanh(As_{\tau-1} + \begin{bmatrix} 0 \\ 0 \\ \text{Id} \end{bmatrix} y_\tau^d) \]
\[ \tau > t : \quad s_\tau = \tanh(As_{\tau-1}) \]

\[ y_\tau = \begin{bmatrix} \text{Id} & 0 & 0 \end{bmatrix} s_\tau, \quad \sum_{t=m}^{T-n} \sum_{\tau=t-m}^{t+n} (y_\tau - y_\tau^d)^2 \to \min \]

Now we have a contradiction: The modeling is dynamical inconsistent.
Modeling Dynamical Systems with Dynamical Consistent Neural Networks

Missing future observations are substituted by the models own expectations.

DCNNs overcome the schizophrenic learning of only 1 matrix $A$ for both a changing and a constant environment.

$$\tau \leq t : \quad s_{\tau} = \begin{bmatrix} \text{Id} & 0 & 0 \\ 0 & \text{Id} & 0 \\ 0 & 0 & 0 \end{bmatrix}\tanh(As_{\tau-1}) + \begin{bmatrix} 0 \\ 0 \\ \text{Id} \end{bmatrix}y_{\tau}^{d} = C_{\leq}$$

$$\tau > t : \quad s_{\tau} = \begin{bmatrix} \text{Id} & 0 & 0 \\ 0 & \text{Id} & 0 \\ \text{Id} & 0 & 0 \end{bmatrix}\tanh(As_{\tau-1}) = C_{>\tau}$$

All $\tau$: $y_{\tau} = \begin{bmatrix} \text{Id} & 0 & 0 \end{bmatrix}s_{\tau}$

$$\sum_{i=m}^{T-n} \sum_{\tau=i-m}^{t+n} (y_{\tau} - y_{\tau}^{d})^{2} \rightarrow \min_{A}$$

Consistency is a necessary condition for the learning of large systems.
We describe the evolvement of the dynamics by a sequence of states \( s_{t} \in \Re^{n} \). The observables \( y_{t} \) are a subset of the states.

The modeling is dynamical consistent:
- we do not assume the constancy of any variable in the future,
- present time has no special meaning between past & future,
- all variables are handled in a symmetric way by one matrix \( A \).

\[
\begin{align*}
\text{target}_{t+i} &= y_{t+i}^{d}, \quad i \leq 0 \\
\sum_{\tau=t-m}^{t} (y_{\tau} - y_{\tau}^{d})^{2} &\rightarrow \min_{A, s_{0}}
\end{align*}
\]

The model is unique along the history \( \rightarrow \) we have only one training example \( = \) history.

Given a finite history, the initial state of a HCNN can be learned in form of a bias vector.
The Identification of Dynamical Systems in Closed Form (HCNN)

- To solve the learning of HCNNs we use teacher-forcing (replace expectations \( y_\tau \) by observations \( y_\tau^d \)).
- For large HCNNs, the extended architecture converges during the learning to the basic architecture.

\[
\begin{align*}
\tau \leq t : & \quad s_{\tau+1} = \tanh(A(s_\tau - \begin{bmatrix} \text{Id} \\ 0 \end{bmatrix}(y_\tau - y_\tau^d))) \\
\tau > t : & \quad s_{\tau+1} = \tanh(As_\tau) \\
\text{all } \tau : & \quad y_\tau = [\text{Id}, 0]s_\tau \quad \sum_{\tau=t-m}^{t}(y_\tau - y_\tau^d)^2 \rightarrow \min_{A, s_0}
\end{align*}
\]

The error flow is identical

\[ r_{\tau}^{upper} = y_\tau - (y_\tau - y_\tau^d) = y_\tau^d \]
A Comparison between Open & Closed Systems

Small recurrent neural nets describe an input – output relationship over time. The open system is modeled as a superposition of an autonomous & an external driven subsystem. The concept works only in the frame of regression paradigms.

Large recurrent neural nets allow an embedding of the observables into a large state vector. These models are dynamical consistent, symmetric in all variables and present time does not play any special role.
In the frame of regression theory we use small forecasting models, which do not explain every movement of the observed data. The residual error is interpreted as additive noise, and should have a similar behavior on training and test data. Thus, it is a measure of the model uncertainty.

We shift from the paradigm of regression theory (under-parameterized models) to large systems modeling (over-parameterized models).

Given a finite data set, there exist many perfect explanations of the past, showing different future scenarios - caused by different reconstructions of the hidden states. One of these scenarios is the true dynamics – but we do not know which one! We take the ensemble average as expected forecast and the ensemble distribution as its risk.
Forecasting LME Copper Prices for Optimal Procurement

- Forecasts are used to hedge the price risks
- Intelligent risk measures for the price volatility.
- SAG: Global purchasing vol: ~120k tons p.a.
- Model impact is evaluated by savings per ton and hit rate of the model.
Neural Networks in Finance

Rational Decision Making in markets is based on an estimation of Expectation & Risk.

Paradigms for the Computation of Market Expectations:
> Neural nets as interacting decision models are micro-economically reasonable market models & can be fitted to macro-data.
> Market dynamics is essentially nonlinear: a) it is a superposition of jumps, b) economic relations are mostly concave (saturation).
> A linear model as a first, easy approach to any function (Taylor expansion) is only valid if the system stays near an equilibrium.

Paradigms for the Computation of Uncertainty / Risk:
> Volatility as a risk measure implies a naive, constant forecast (a high frequency wave has high volatility without uncertainty).
> Backward risk is the error between model and real-world. Here distributional features of the risk depends on the chosen model.
> Forward risk can be estimated by an ensemble forecast: many models fit the past perfectly and still give diversifying forecasts.


