Time Walk Correction via Artificial Neural Networks

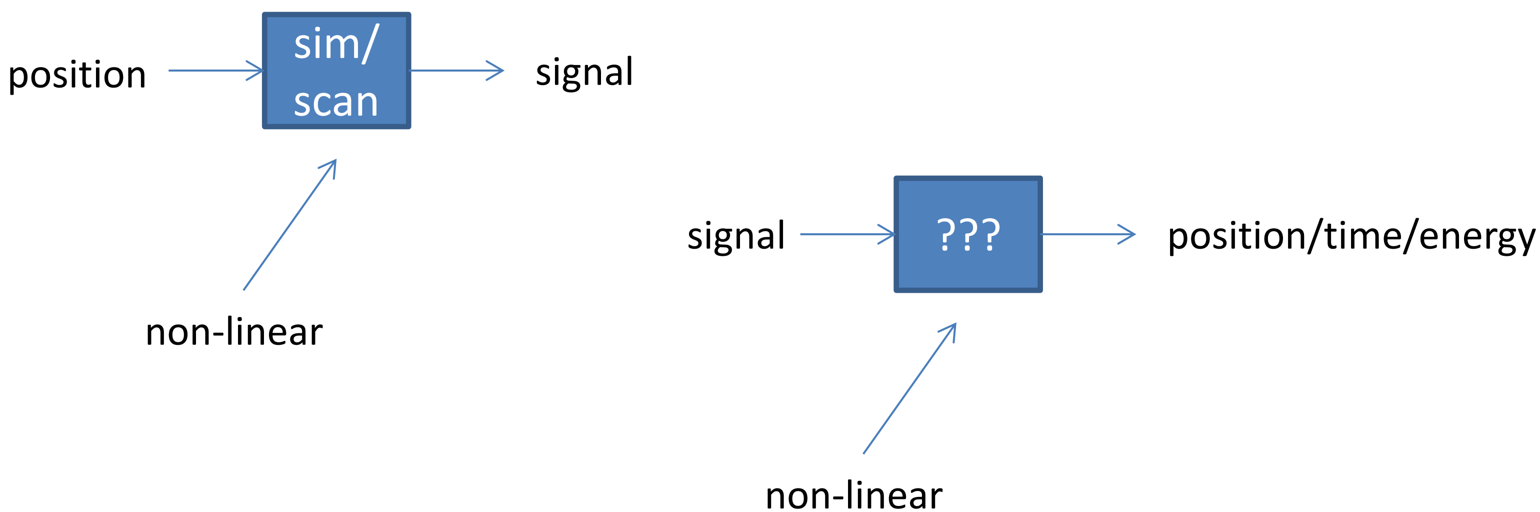
T.Habermann

10.02.2015

- Artificial Neural Networks (ANN)
- Time walk correction via ANN
- Outlook

Why Artificial Neural Networks?

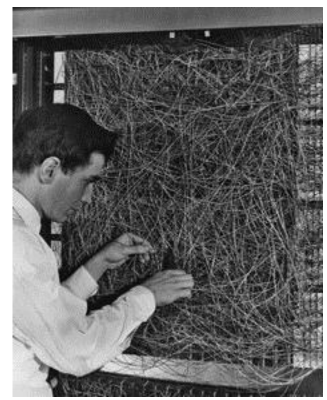
PSA = Inverse Problem



Non-linear models are usually difficult to work with ☹
ANNs are rather easy to use ☺

Artificial Neural Networks – A bit of history

1943 "A logical calculus of the ideas immanent in nervous activity" (W.S.McCulloch, W.Pitts)



1958 Mark I Perceptron (F.Rosenblatt)

1969 "Perceptrons" (M.Minsky, S.Papert)

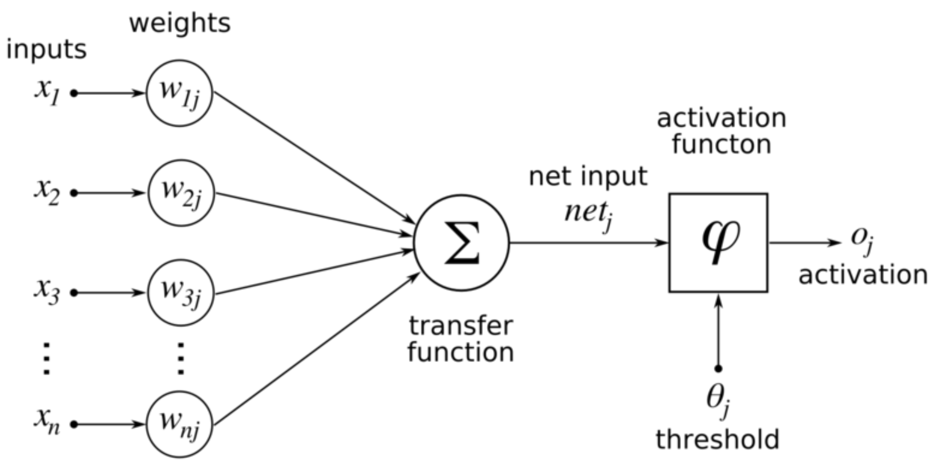
1986 "*Parallel Distributed Processing: Explorations in the Microstructure of Cognition*" (D.E.Rumelhart, J.L.McClelland)

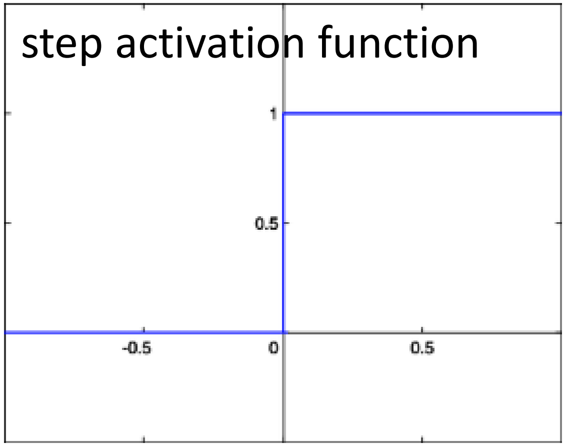
1987 First IEEE annual ANN conference

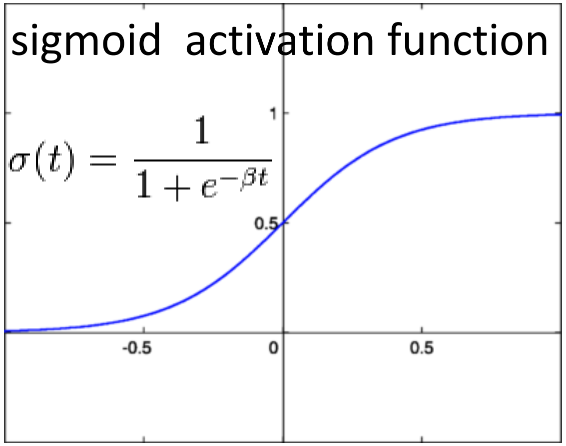
1988 International Neural Network Society (INNS)

Artificial Neurons

Single Neuron

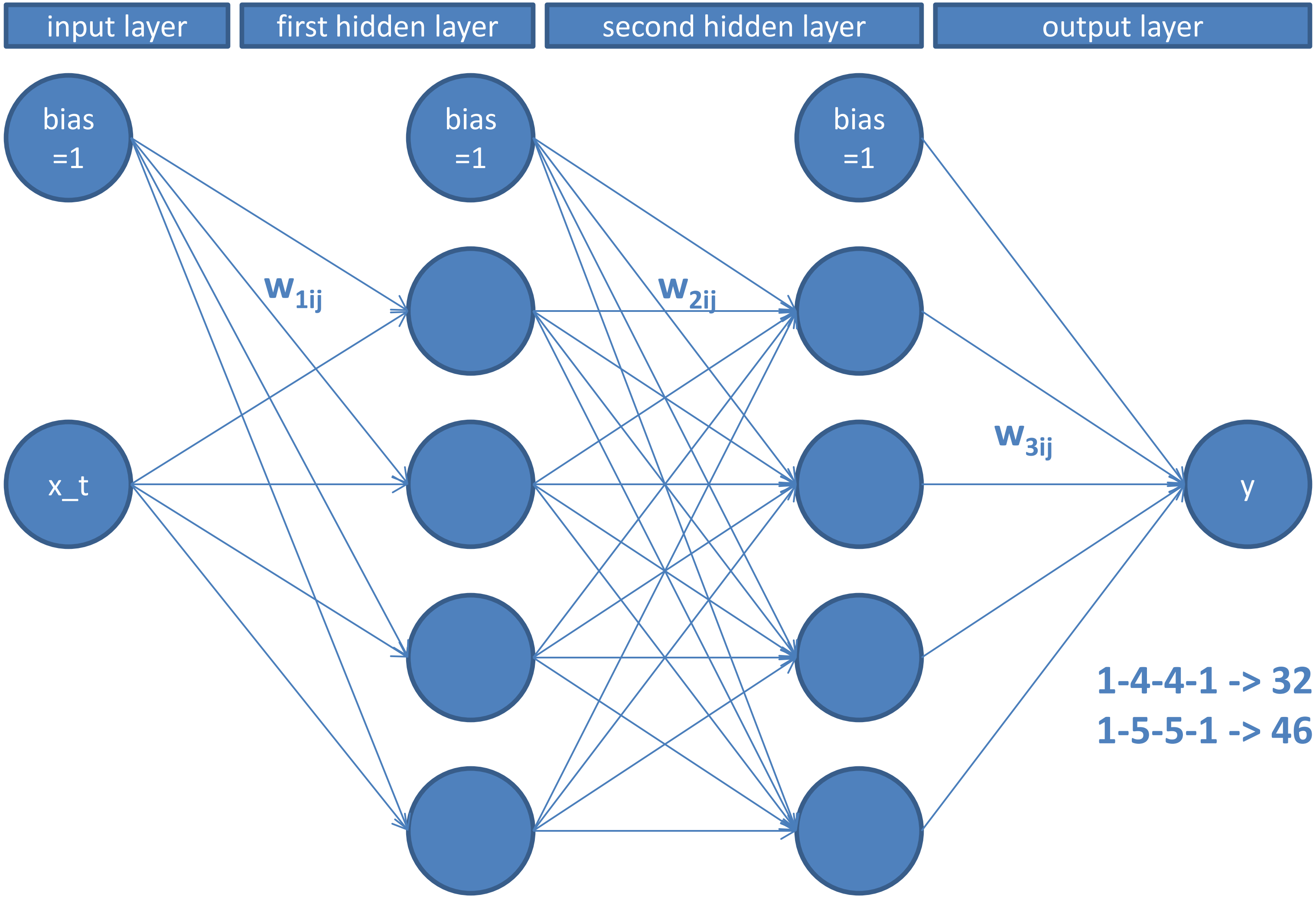






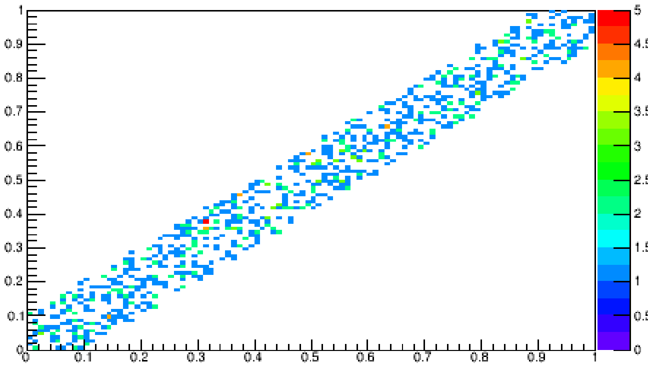
http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions

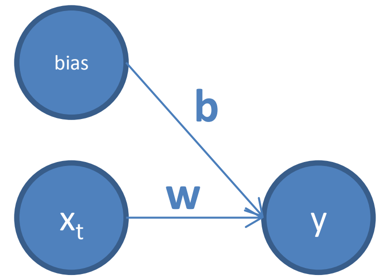
Network Topology



1-4-4-1 -> 32 parameters
1-5-5-1 -> 46 parameters

Linear vs Non-Linear Model

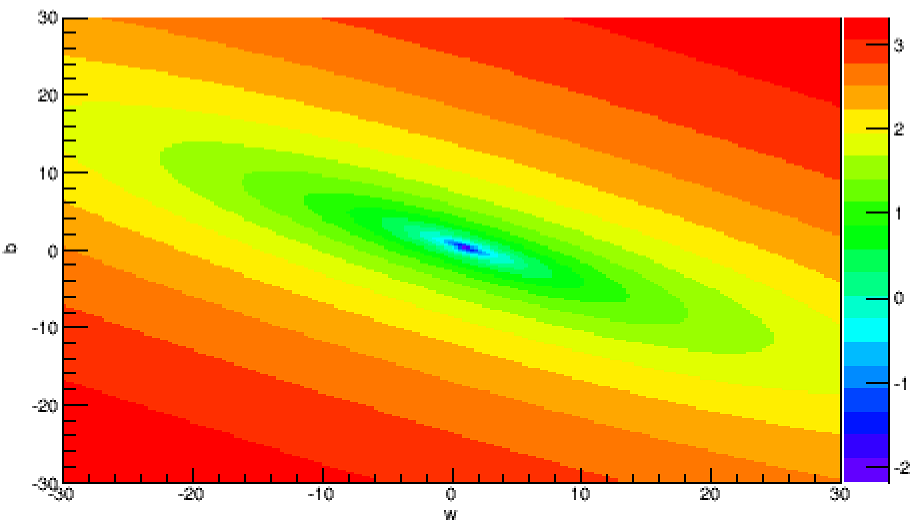


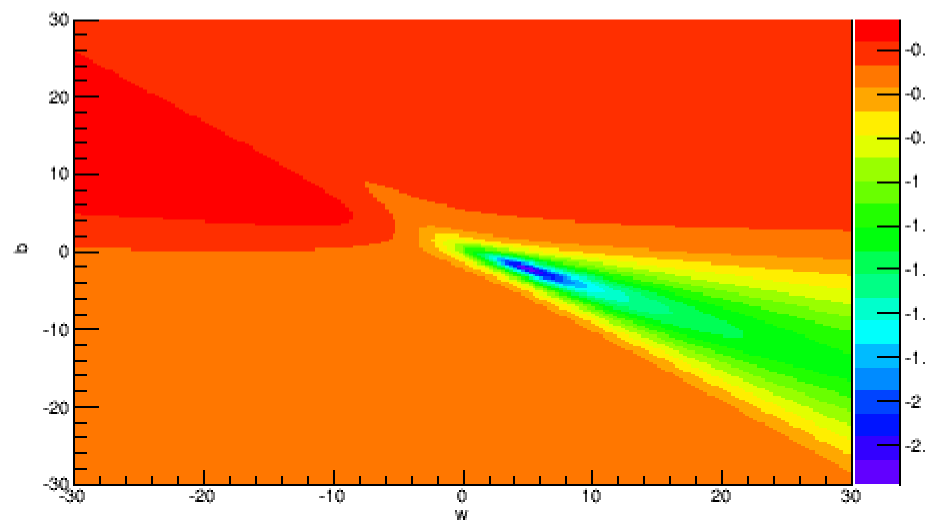


lin.reg.: $y(x_t) = wx_t + b$

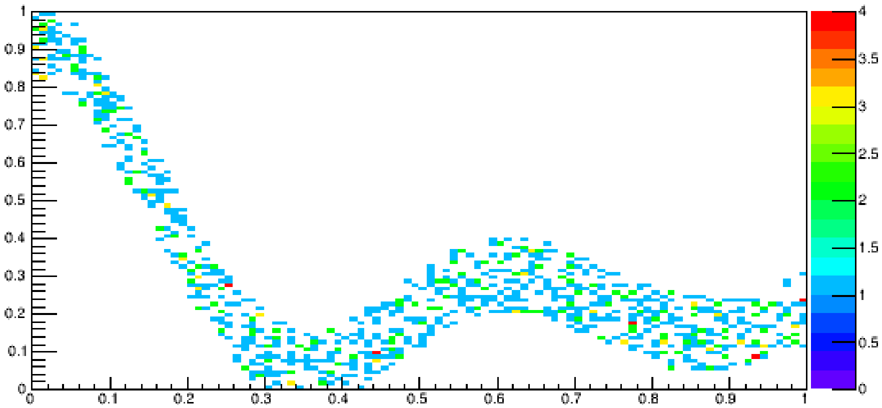
non-linear: $y(x_t) = \text{sigmoid}(wx_t+b)$

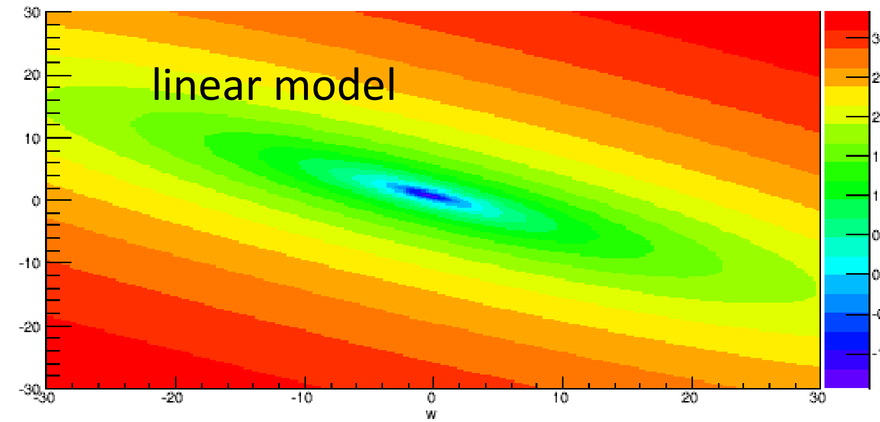
minimize: $E(w,b) = \sum (y_t - y(x_t))^2$ for the given data $\{x_t,y_t\}$

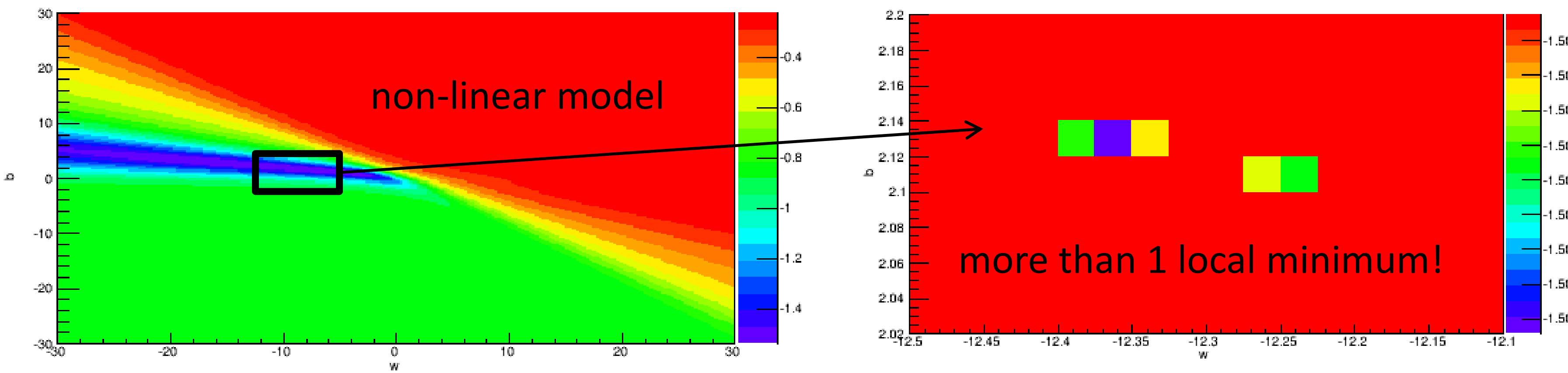




Linear vs Non-Linear Model







Gradient Descent Method

$y = F(\Sigma w_i x_i) = F(e) \qquad \mathrm{e} = \Sigma w_i x_i$

$\varepsilon = (y^* - F(e))^2$

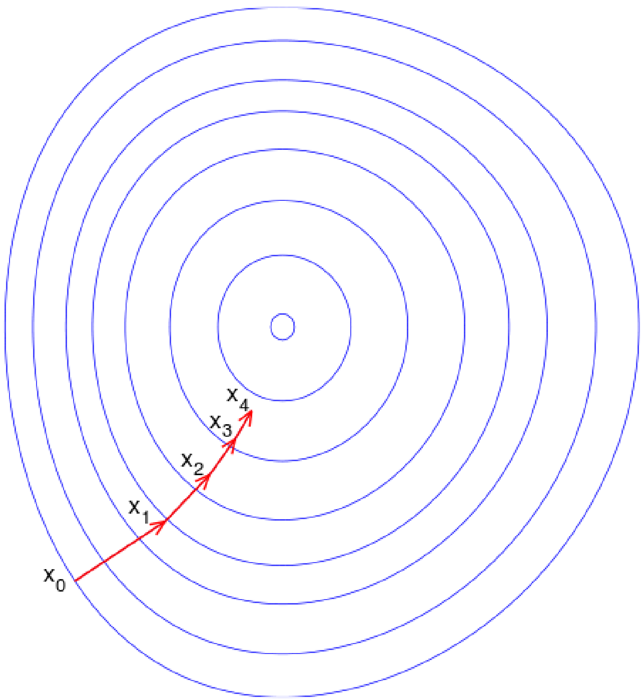
Find minimum error via Iteration:

$w_i{}^{t+1} = w_i{}^{t} + \Delta w_i{}^{t}$

Gradient descent:

$\Delta w_i{}^{t} = -\eta \dfrac{\partial \varepsilon}{\partial w_i}$ η – learning parameter

$\Delta w_i{}^{t} = -\eta \dfrac{\partial \varepsilon}{\partial w_i} + \alpha \Delta w_i{}^{t-1}$ α – momentum parameter

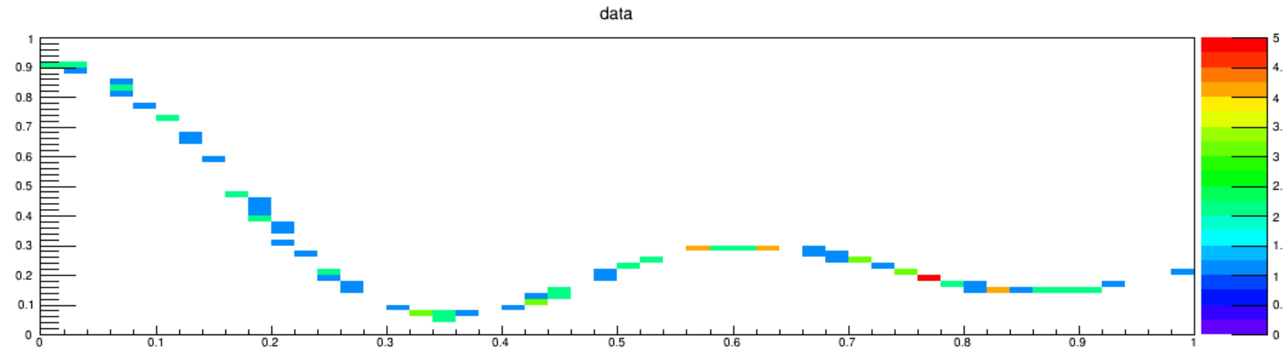


http://en.wikipedia.org/wiki/Gradient_descent

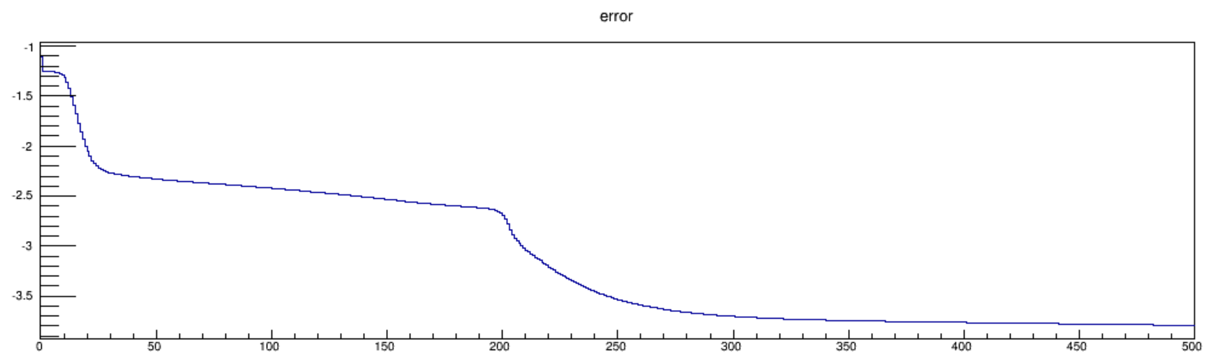
$$\frac{\partial \varepsilon}{\partial w_i} = 2(y^* - F(e)) * \left(-\frac{\partial F}{\partial e}\frac{\partial e}{\partial w_i}\right) = -2dF'(e)x_i$$

First Trial

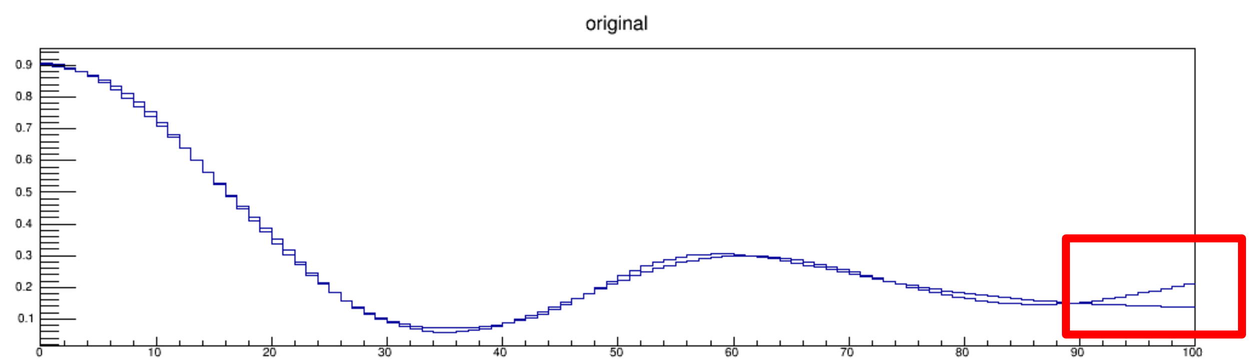
$$y = \frac{\sin(4\pi x)}{4\pi x}$$



training data:
100 random
samples



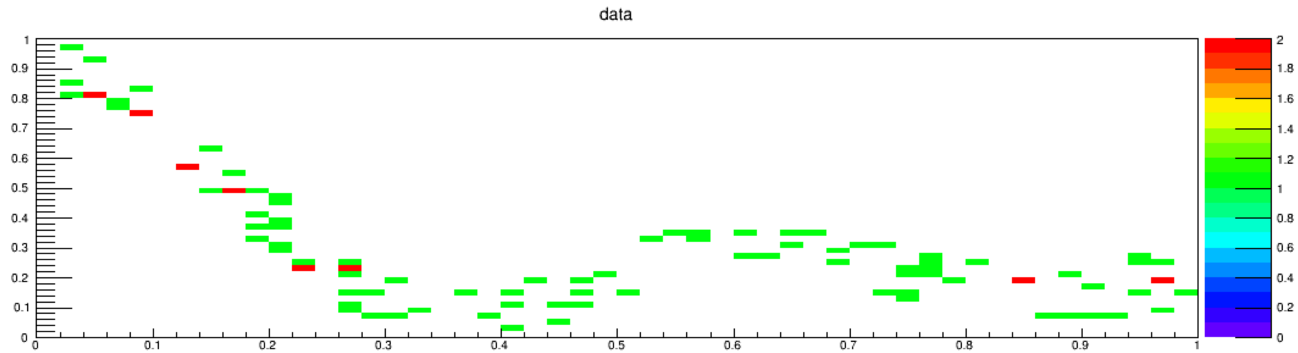
average error
converges nice
and smoothly



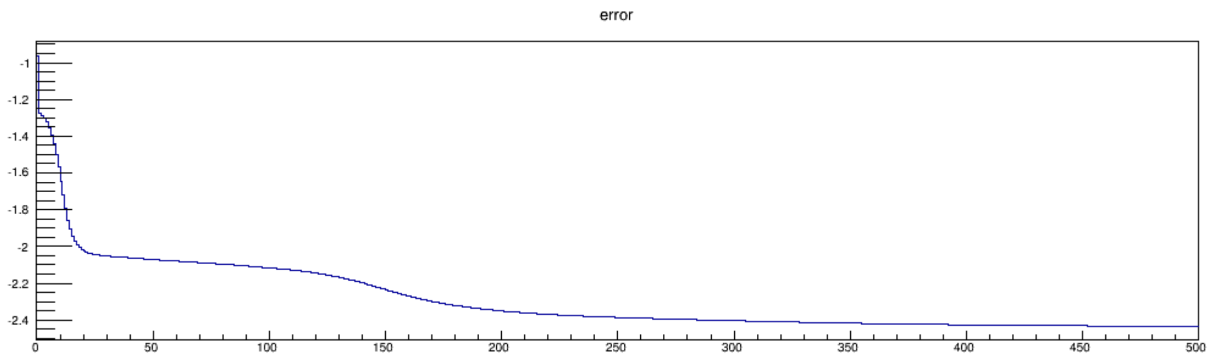
only one small
problem...
...bug?

Second Trial: Noisy Data

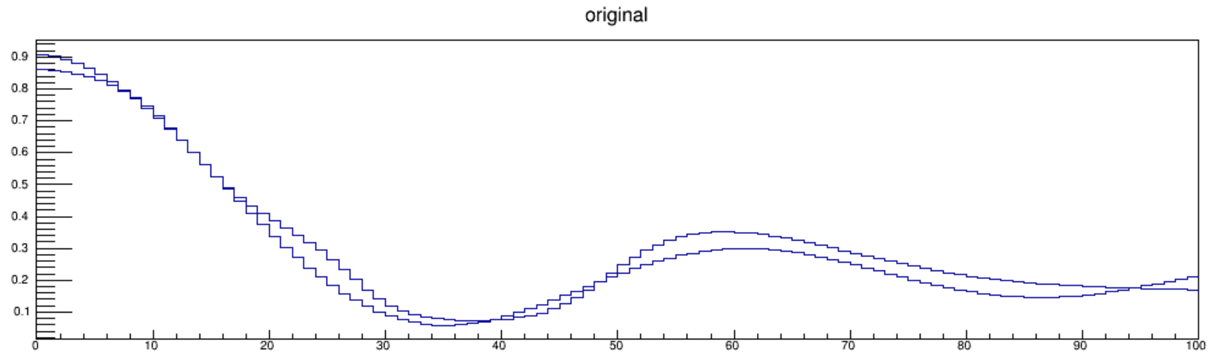
training data:
100 random
samples ± 10% noise



still nice
convergence

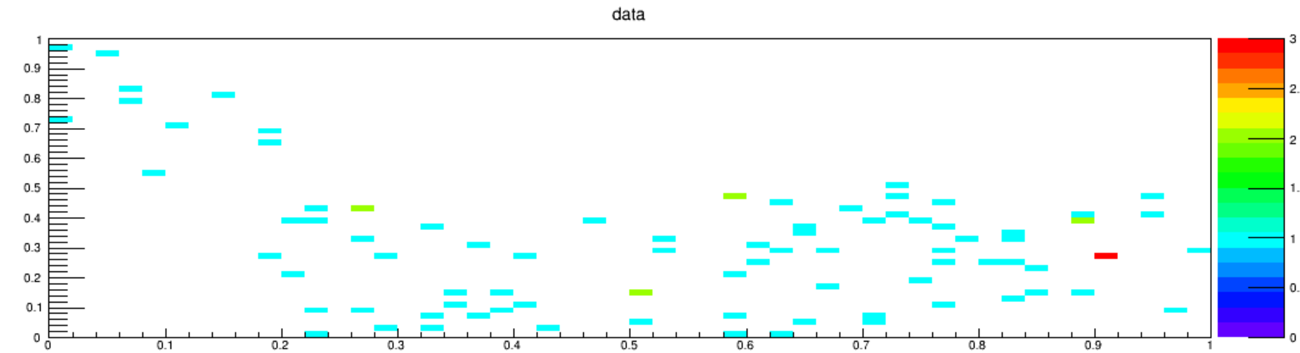


more or less “ok”

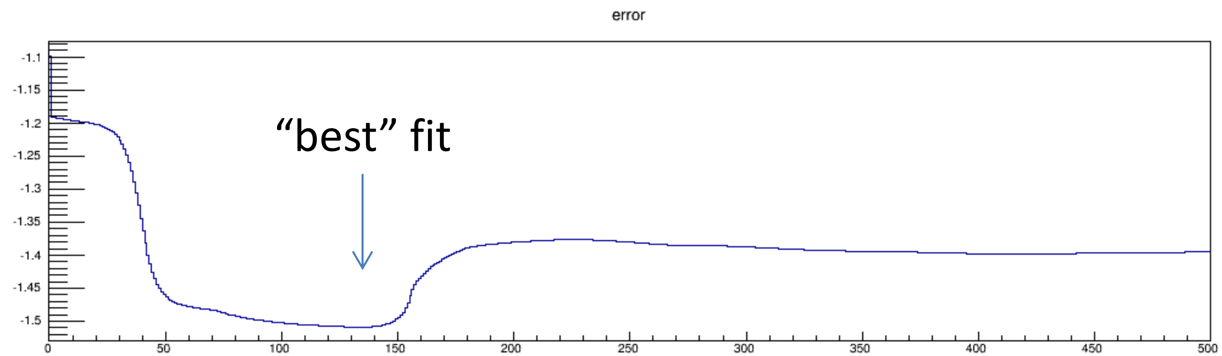


Second Trial: Very Noisy Data

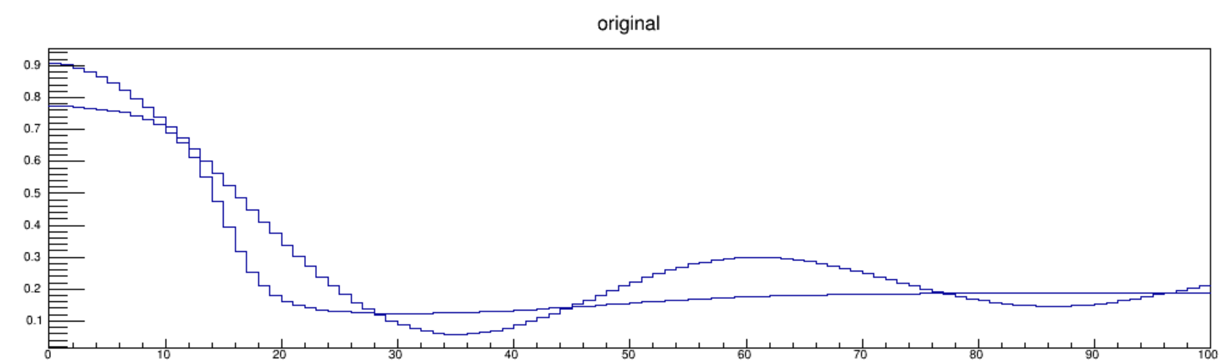
training data:
100 random
samples ± 25% noise



gradient descent
is not stable!!

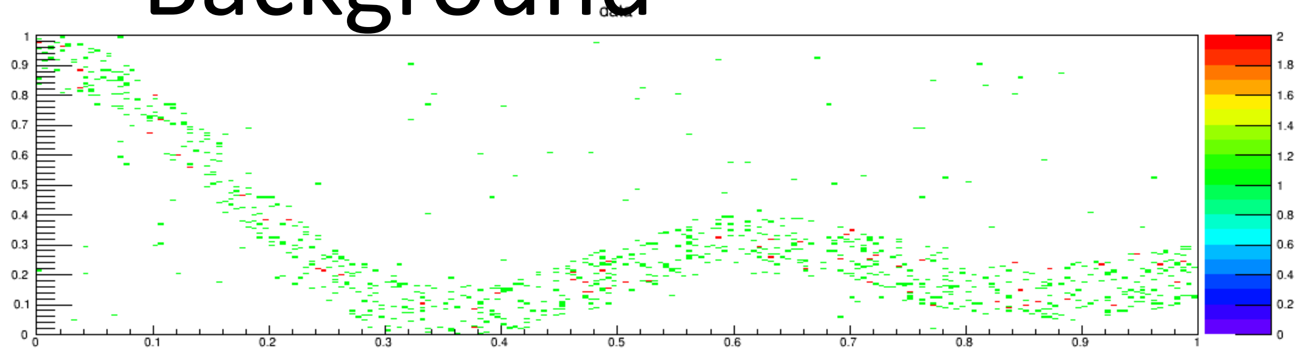


for the given data,
this is still “ok”

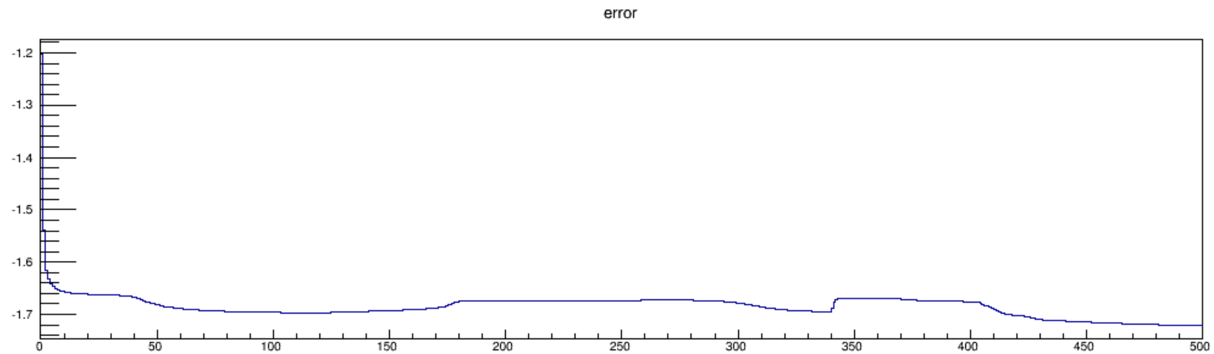


Third Trial: Noisy data on top of Background

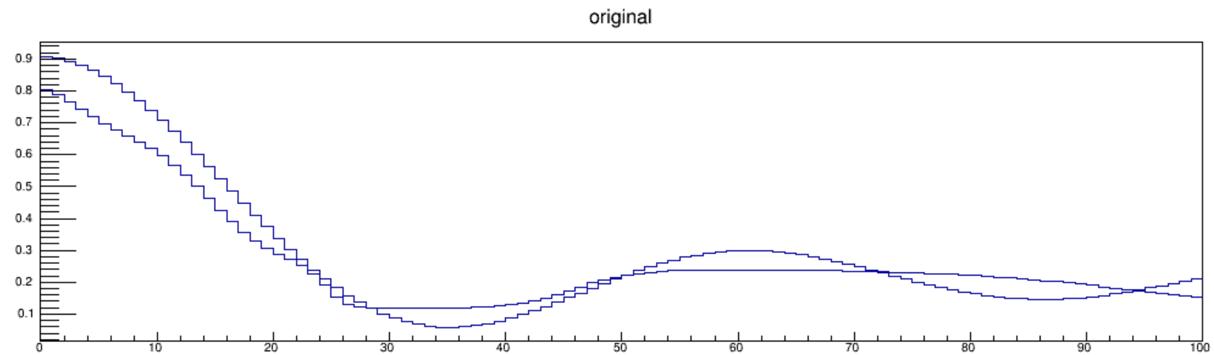
training data:
1000 random
samples ± 10% noise
with 10% background



converges
“somehow”

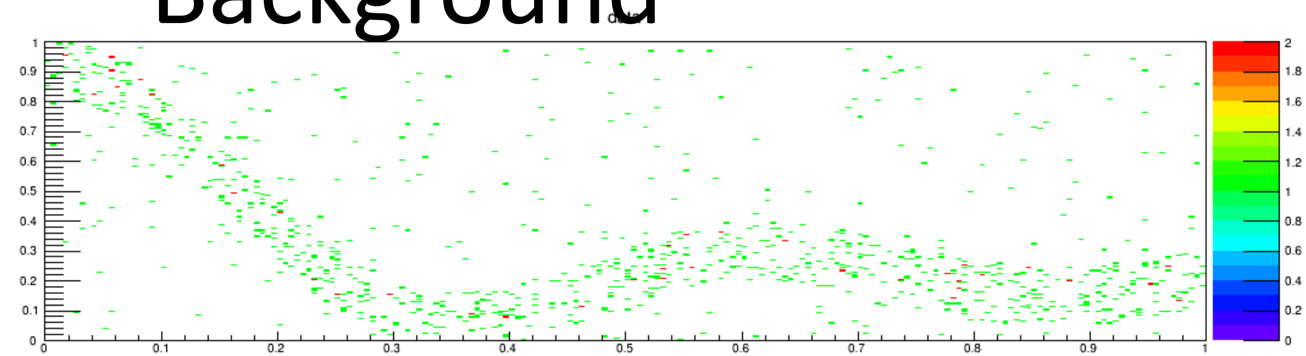


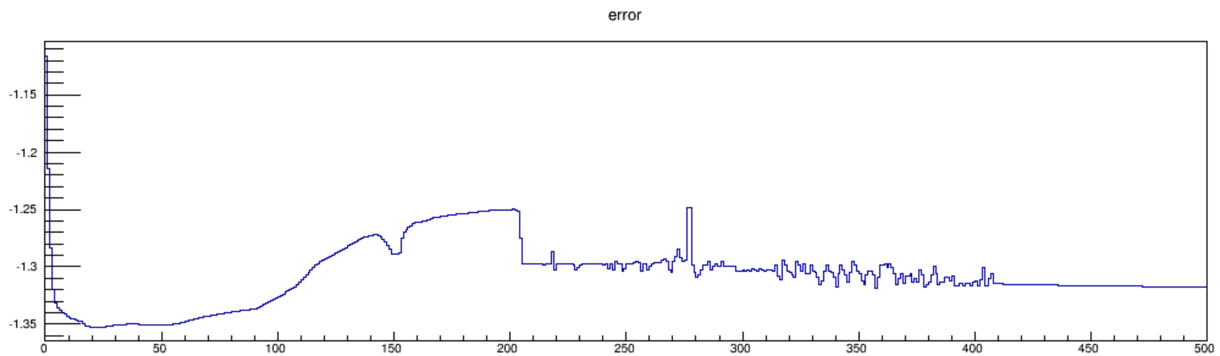
not so nice,
but at least close



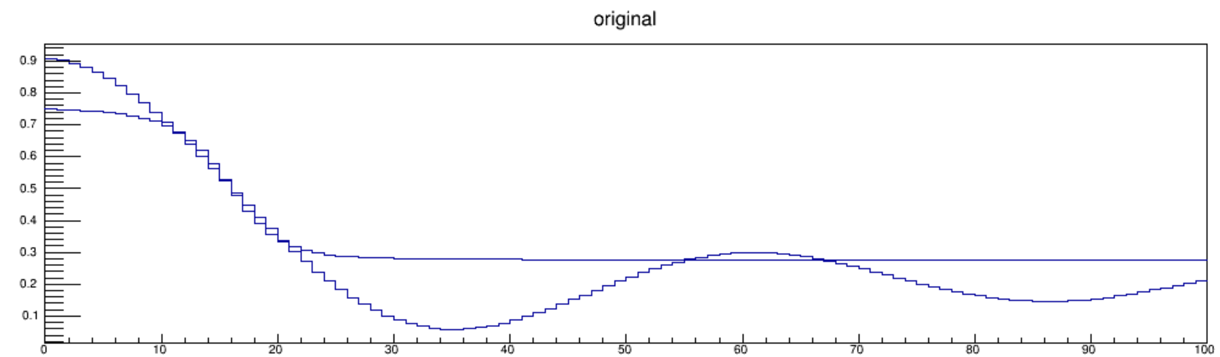
Third Trial: Noisy data on top of more Background

training data:
1000 random
samples ± 10% noise
with 30% background





no chance to get
detailed features

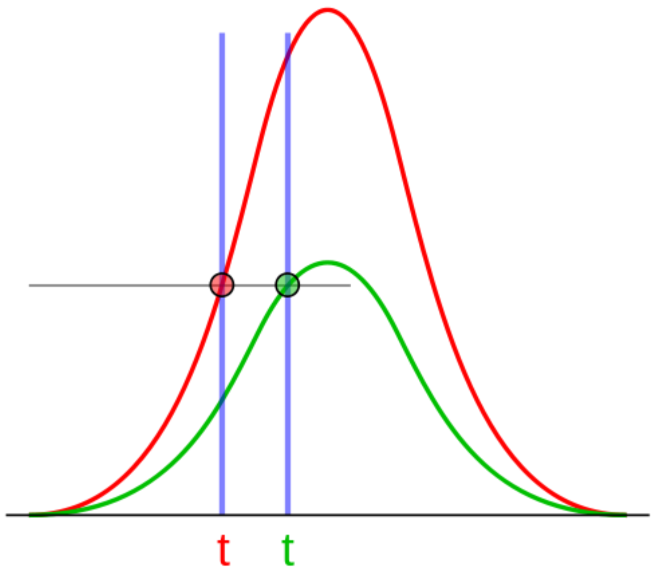


Conclusions

- ANNs should NOT be used as black box
- always have to monitor average error
- have to be aware that there might be more than one minimum
- quality of the fit is mainly determined by quality of training data
- stability is an issue, but we do not have to care too much (once we have a solution that fits)

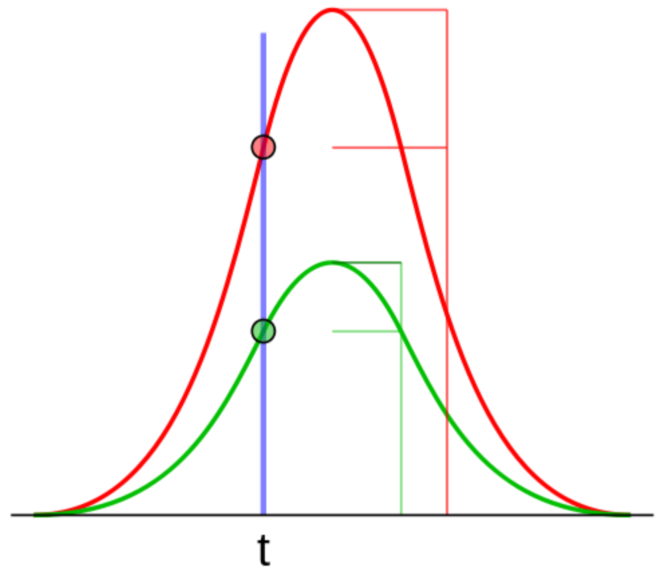
Time Walk

Leading edge discriminator



different amplitudes -> time walk

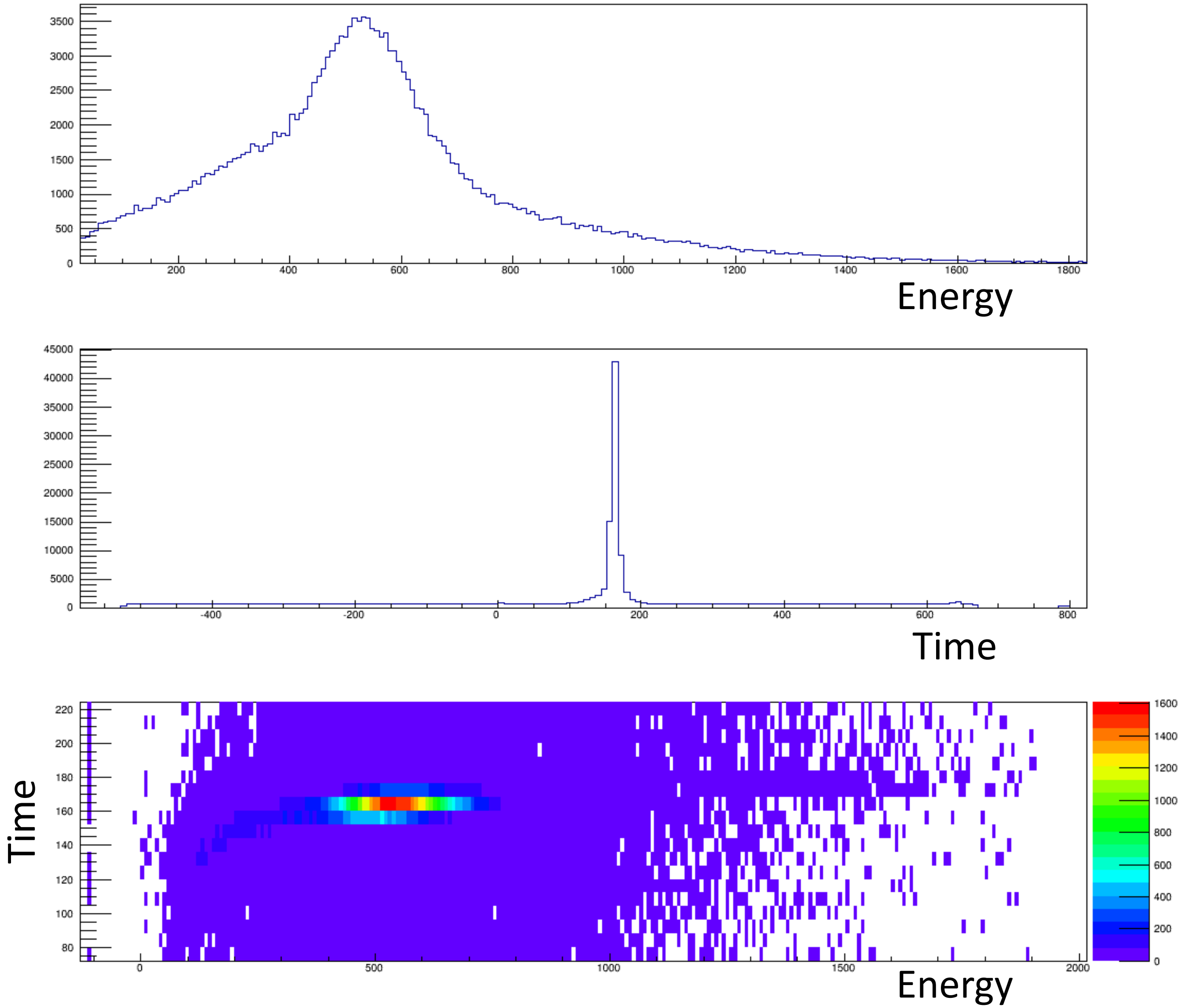
Constant fraction discriminator



no (energy dependent) time walk

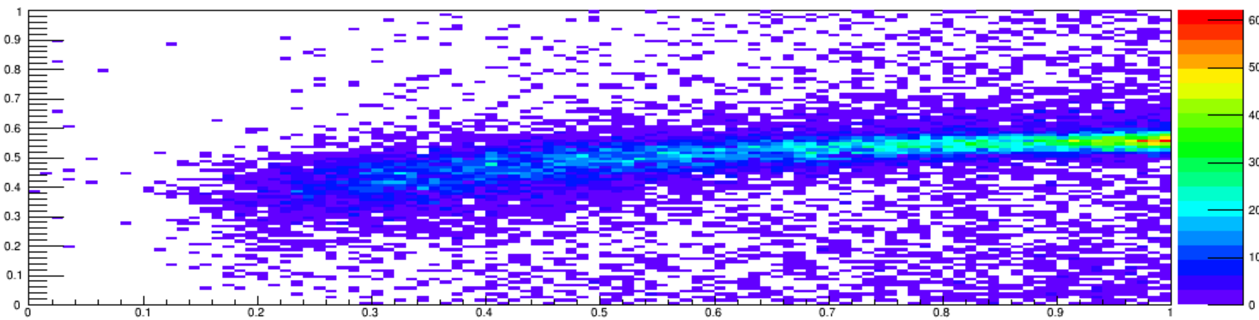
http://en.wikipedia.org/wiki/File:Constant_fraction_1.svg

Energy dependent Time Walk

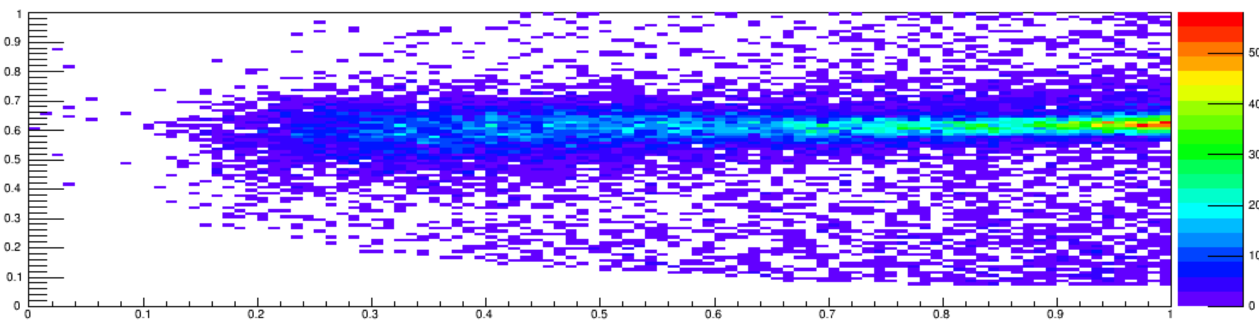


Correcting the Walk after Fitting

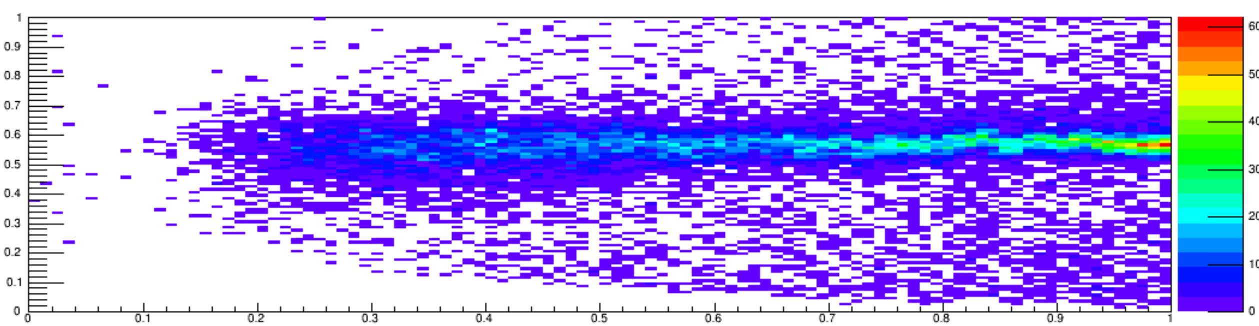
original



corrected after
ANN fit
(1-5-5-1)

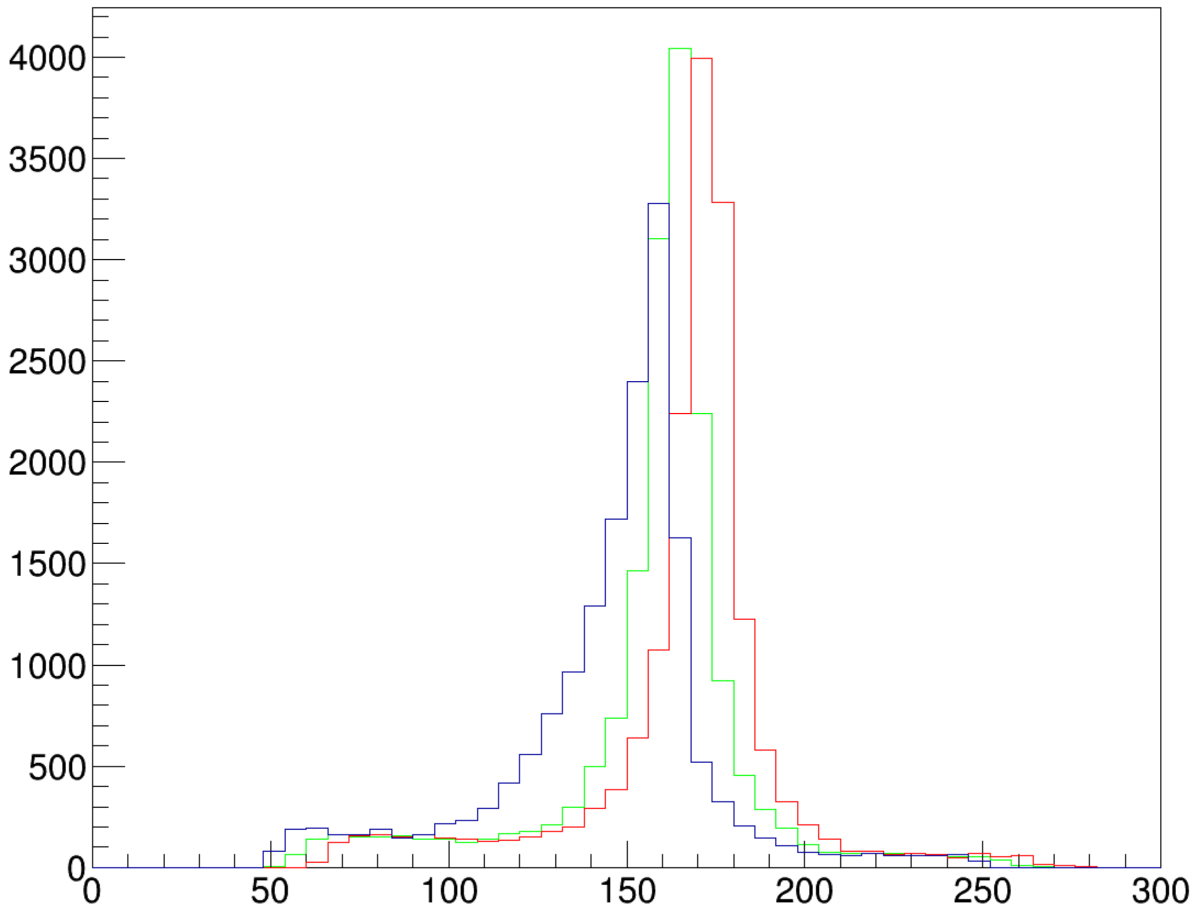


corrected after
binned mean value
(46 bins)



Corrected Time Spectra

	σ (bins)
org.	16.94
ANN	10.09
AVG	10.19

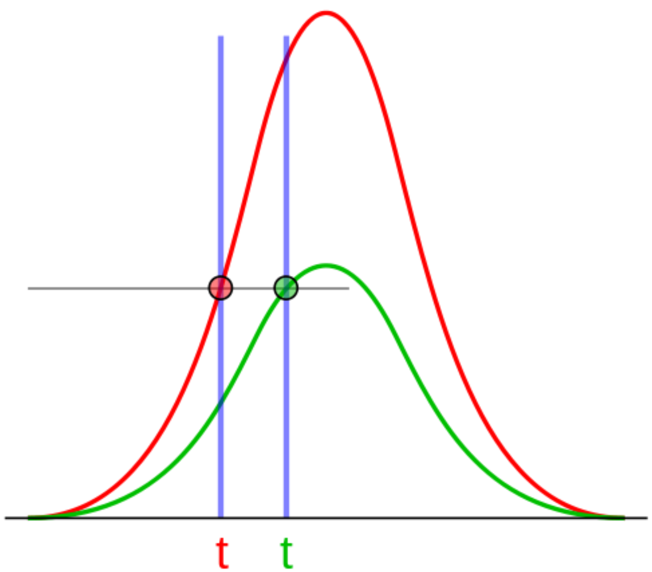


Conclusions

- for low dimensional problems we do not need ANNs
- the same ANN can be extended to work with n inputs and m outputs easily
- ANN results should be compared with results from linear methods

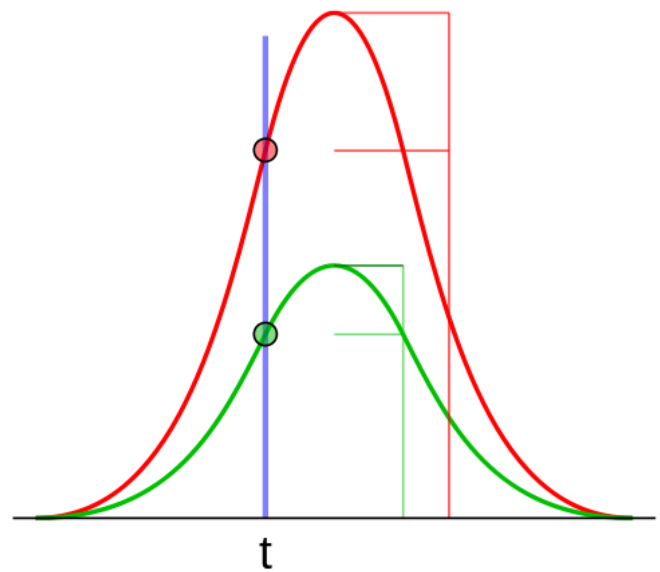
Outlook

Leading edge discriminator



amplitude dependent time walk

Constant fraction discriminator



pulse shape dependent time walk

http://en.wikipedia.org/wiki/File:Constant_fraction_1.svg

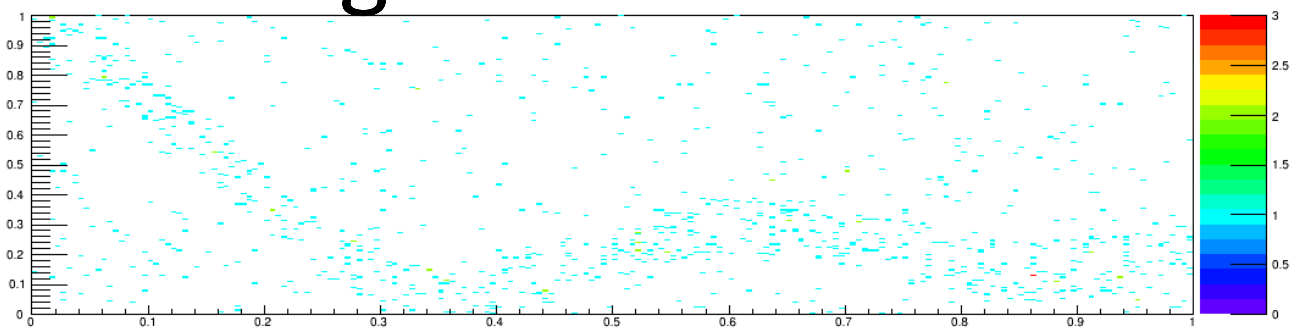
Nice explanation of backpropagation (careful: bias nodes are missing)
http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

My source and doc will be available at
https://g-wiki.gsi.de/foswiki/bin/view/SWiki/PulSAr

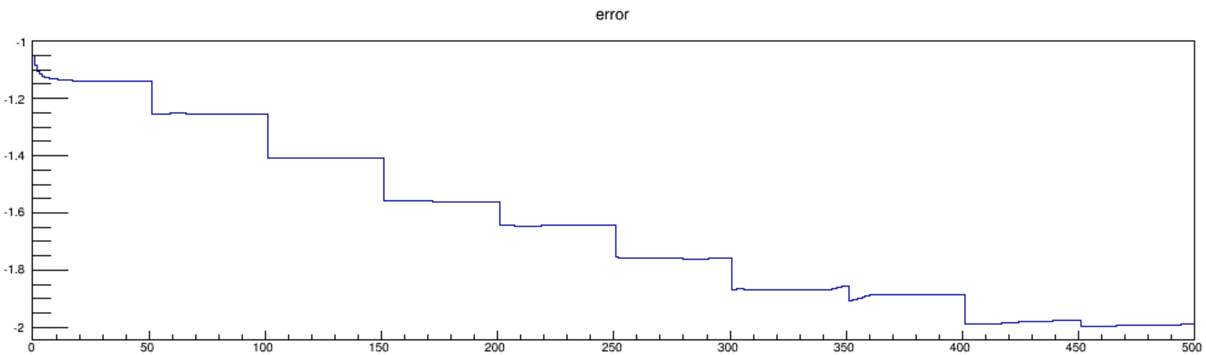
source code / libraries (just a random collection, I didn't read them)
http://www.heatonresearch.com/encog (C#,Java)
https://takinginitiative.wordpress.com/2008/04/23/basic-neural-network-tutorial-c-implementation-and-source-code/ (C++)
http://www.codeproject.com/Articles/14342/Designing-And-Implementing-A-Neural-Network-Librar (.NET)
http://www.codeproject.com/Articles/21171/Backpropagation-Artificial-Neural-Network-in-C (C++)

Third Trial: Noisy data on top of huge Background

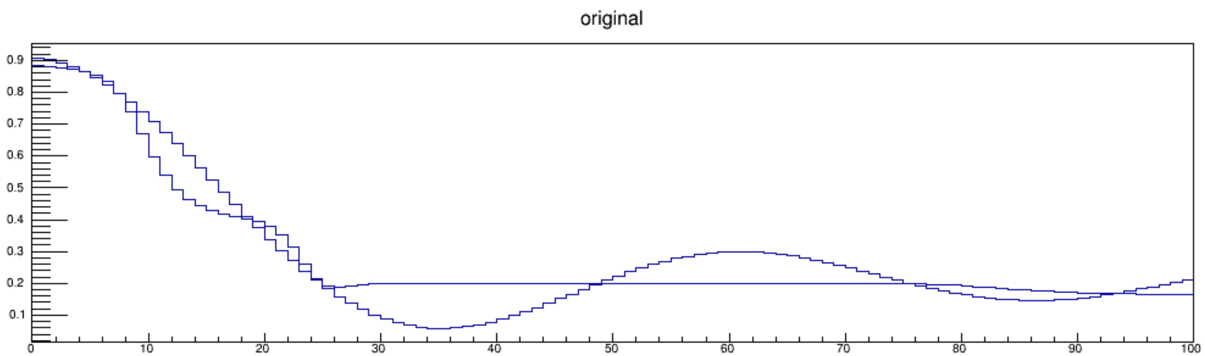
training data:
1000 random
samples ± 10% noise
with **50%** background



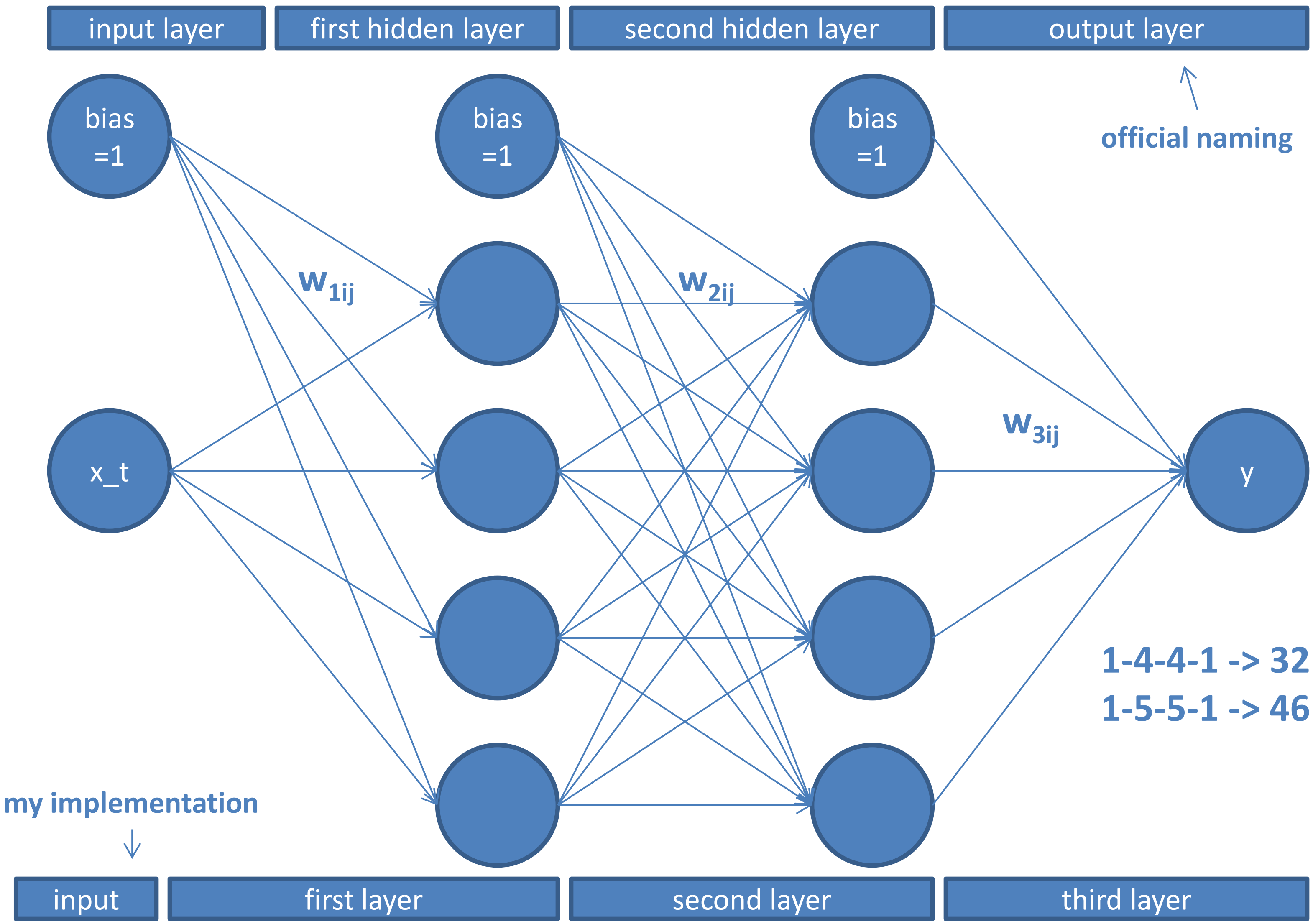
may converge...
...to something



still possible to
get close



Network Topology



1-4-4-1 -> 32 parameters

1-5-5-1 -> 46 parameters