Application of Neural Networks

practical remarks

or

What do neural networks actually. And how?

František Hakl , Institute of Computer Science, Prague

Zakopane, June 2003

Goal of this talk

Lot of people are using neural network as clasifier or as prediction machine. Neural networks behaviour differs to deterministic algorithms one. Some of these behaviour will be pointed out in this talk to clarify them to users.

Especially we pay attention to:

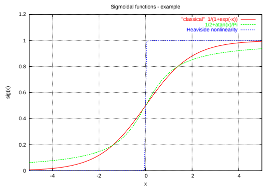
- universal approximation (Kolmogorov theorem, Stone-Weierstraß theorem)
- overlearning phenomena
- numerical behaviour of delta rule
- recomendations

Additional topic

- application of neural networks to Higgs bosson search

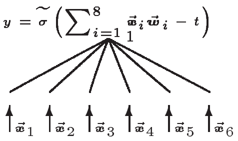
Introduction

Definition 0.1 *Sigmoidal function is any function* $\widetilde{\sigma} : \Re \to \Re$ *which is monotonous and* $\lim_{x \to -\infty} = 0$, $\lim_{x \to \infty} = 1$.



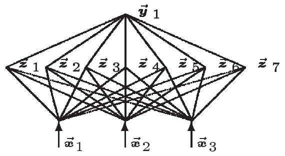
For example:

$$\frac{1}{1+e^{-x}}, \qquad \arctan(x), \qquad f(x) = \begin{cases} 0 & for \quad x < 0 \\ 1 & for \quad x \geq 0 \end{cases}$$



PERCEPTRON: divide input space by hyperplane. Can separate only linearly separated sets only. (Perhaps sipliest neural network.)

MULTILAYER PERCEPTRON (MLP):



$$output = \widetilde{\sigma}\left(\sum_{j=1}^{7} \omega_j \widetilde{\sigma}\left(\sum_{i=1}^{3} \vec{x}_i \vec{w}_i - t\right)\right)$$

Multilayer Perceptron is universal approximator

Theorem 0.1 (Kolmogorov superposition theorem (1957)) *For any natural number* $n \geq 2$ *there exist* $n \times (2n+1)$ *continuous functions* $\widetilde{\psi_{pq}}$ *such that for any real function* $\widetilde{f} : \Re^n \to \Re$ *there exists* $2n$ *continuous functions* $\widetilde{\Phi}_q$ *such that for arbitrary vector* $\vec{x} \in \Re^n$ *the following holds:*

$$\widetilde{f}(\vec{x}) = \sum_{q=0}^{2n} \widetilde{\Phi_q}\left(\sum_{p=1}^{n} \widetilde{\psi_{pq}}(\vec{x}_p)\right).$$

- Kolmogorov theorem is a great mathematical result with very sophisticated proof

but

- constructive proof based on recursive construction of stairwise functions $\widetilde{\psi_{pq}}(\vec{x}_p)$ dependent on the target function $\widetilde{f}$
- functions $\widetilde{\Phi}_q$ are recursive iterations of $\widetilde{f}$
- it is impossible to exploit Kolmogorov theorem in MLP analysis

$\Rightarrow$ do not pay attention to Kolmogorov theorem

Another methods to analyze neural networks:

- theory of dual spaces (show universal approximation in integral norms, e.g. $\mathcal{L}_p^q$, e.t.c.)
- application of convolution (density of sigmoidal functions with weak conditions in C_K spaces)
- Stone–Weierstraß theorem (straithforward method, easy to understand)

Application of Stone–Weierstraß theorem

Theorem 0.2 (Stone-Weierstraß) *Let $\mathcal{B}$ is linear subspace of $C_{\Re^n}$ such that nonzero constant function is in $\mathcal{B}$ and for arbitrary $\vec{x}, \vec{y} \in \Re^n$, $\vec{x} \neq \vec{y}$, there exists $\widetilde{f} \in \mathcal{B}$ such that $\widetilde{f}(\vec{x}) \neq \widetilde{f}(\vec{y})$. Then $\mathcal{B}$ is dense in $C_{\Re^n}$.*

(e.g. any function in $C_{\Re^n}$ can be approximated by functions from $\mathcal{B}$ with arbitrary precission)

Proof is easy, no deep mathematics is needed.

Theorem 0.3 (***) *Let $\widetilde{\sigma}$ is sigmoidal function. Then the linear hull of the set $\left\{\widetilde{\sigma}(ax - t) \,|\, a, t \in \Re, x \in \langle 0, 1\rangle\right\}$ is dense in $C_{\langle 0,1\rangle}$.*

Problem:

- *** theorem can not be generalized to more dimensional case

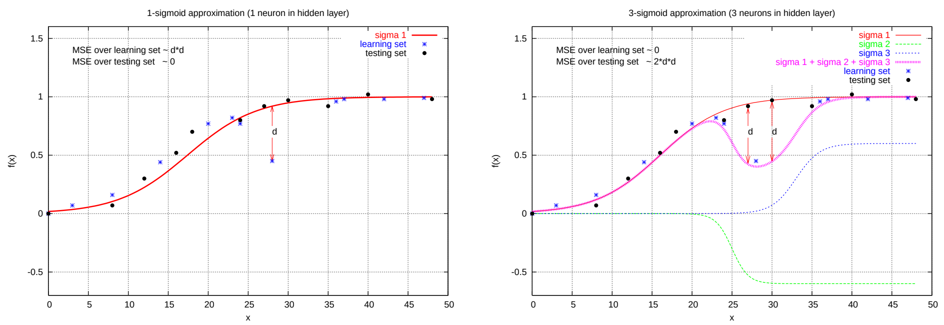
but: $\vec{x}, \vec{y} \in \Re^n, \vec{x} \neq \vec{y}$. Put $\vec{a} = \vec{x} - \vec{y}$. Then $e^{\langle \vec{a} | \vec{x} \rangle} \neq e^{\langle \vec{a} | \vec{y} \rangle}$. Also $e^{\langle \vec{o} | \vec{x} \rangle} = 1 \neq 0$. $\Rightarrow$ by Stone–Weierstraß theorem the linear hull of $\left\{ e^{\langle \vec{a} | \vec{x} \rangle} \,|\, \vec{a} \in \Re^n \right\}$ is dense in $C_{\Re^n}$.

and theorem *** implies that any $e^{\langle \vec{a} | \vec{x} \rangle}$, $\vec{a} \in \Re^n$, can be on $\langle 0, 1\rangle$ approximated by sum of sigmoids, i.e. $e^{\langle \vec{a} | \vec{x} \rangle}$ can be approximated by sum of $\widetilde{\sigma}(ax - t)$ functions.

Therefore, MLP with one hidden layer can approximate any continuous function on a given compact set in $\Re^n$.

Overlearning – principle

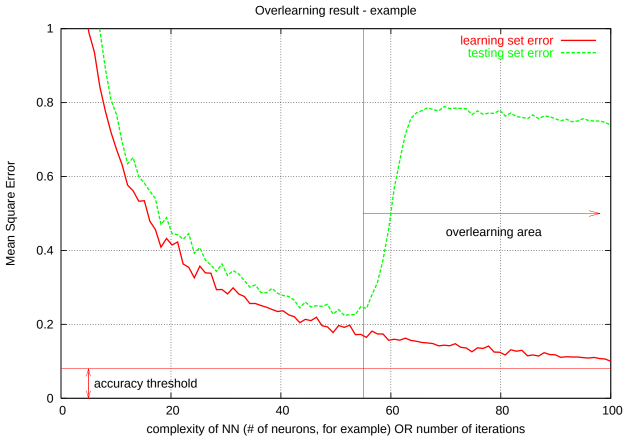
Approximation of function by multilayer perceptron with one neuron in a hidden layer (left) and with three neurons in a hidden layer (right). One neuron approximation produce better result for testing set (•) approximation, than three neuron approximation one (which fit learning set (×) almost exactly but with worse approximation of testing set, e.g. poor generalization).



Recomendation:

- use less complicated model
- and/or use larger learning & testing sets

Overlearning – exapmle



Recomendations:

- choose NN and learn and test
- enlarge NN model and learn till MSE for testing set is significantly worse than for learning set
- simplify NN and find optimal complexity of NN
- MSE for learning and testing set should be resemble

(accuracy threshold should be zero for universal approximator (like MLP) but, in fact, is nonzero, see numerical properties of delta rule)

Delta rule = back-propagation for single perceptron

patterns: $(\vec{x}_1, y_1), \ldots, (\vec{x}_t, y_t)$, main goal: minimize $\widetilde{E}(\vec{w}) \stackrel{\text{def}}{=} \sum_{p=1}^{t} (y_p - \langle \vec{w} \,|\, \vec{x}_p \rangle)^2$

- initial weights are set randomly
- cyclic adaption of weight vector by delta rule formula (η - convergence parameter)

$$\delta \vec{w} \stackrel{\text{def}}{=} \eta \left(y - \langle \vec{w} \,|\, \vec{x} \rangle\right) \vec{x}\ ,$$

This adaption leads to formula

$$\vec{w}_{k+1} \stackrel{\text{def}}{=} \vec{w}_k + \delta \vec{w}_k = \vec{w}_k + \eta \left(y - \vec{w}_k^T \vec{x}\right) \vec{x} = \left(\boldsymbol{I} - \eta \vec{x} \vec{x}^T\right) \vec{w}_k + \eta y \vec{x}$$

patterns are applied in epoch: $\vec{x}_1, \ldots, \vec{x}_t, \vec{x}_1, \ldots, \vec{x}_t, \cdots$ till some stop condition

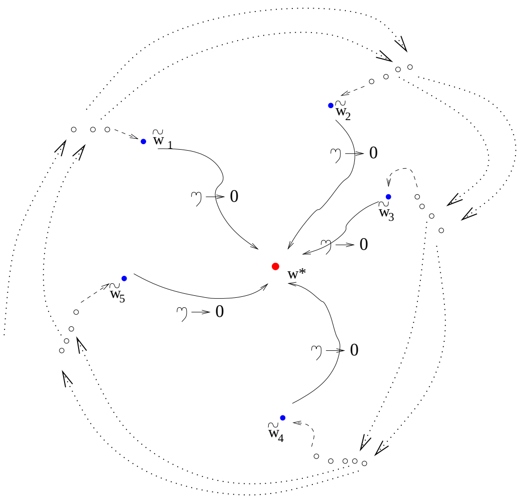
Definition 0.2 $\boldsymbol{B}_{\vec{x}_p} = \left(\boldsymbol{I} - \eta \vec{x}_p \vec{x}_p^T\right)$, $\boldsymbol{\Lambda}_\pi \stackrel{def}{=} \boldsymbol{B}_{\pi(t)} \boldsymbol{B}_{\pi(t-1)} \cdots \boldsymbol{B}_{\pi(1)}$ *(π is any permutation)*

Lemma 0.4 *Let $0 \leq \eta \leq \frac{2}{\|\vec{x}_p\|^2}$ is satisfied for all numbers $p \in \{1, \ldots, t\}$ and the system of vectors $\vec{x}_1, \ldots, \vec{x}_t$ generate whole space $\Re^n$. Than for each permutation π, $\|\boldsymbol{\Lambda}_\pi\| < 1$.*

hence if $\vec{z}_{k+1} = \boldsymbol{\Lambda}_i \vec{z}_k$, than $\{\vec{z}_j\}_1^\infty$ is convergent to unique solution of $\vec{z} = \boldsymbol{\Lambda}_i \vec{z}$.

Theorem 0.5 *Let $\vec{w}^*$ minimizes value of $\widetilde{E}(\vec{w})$. Than the sequence $\{\vec{w}_i\}_1^\infty$ converges to finite cycle $\{\vec{w}_i\}_1^t$ and each vector $\vec{w}_i$ is unique fixed point of contracted mapping $\widetilde{F}_i(\vec{w}) \stackrel{def}{=} \boldsymbol{\Lambda}_{\pi_i} \vec{w} + \eta \vec{h}_{\pi_i}$, (permutation π_i is defined as $(i+1, \cdots, t, 1, \cdots, i)$) Let $\vec{w}(\eta)$ be arbitrary term of the above cycle for a given fixed value of $\eta > 0$. Than: 1. $\|\vec{w}(\eta) - \vec{w}^*\| = o(\eta)$ 2. $\left|\widetilde{E}(\vec{w}(\eta)) - \widetilde{E}(\vec{w}^*)\right| = o\left(\eta^2\right)$.*

Delta rule – convergence properties



● w^* – optimum weight vector, ● $\tilde{w}_i$ – limit cycle, ○ – iterations

- weights are oscilating during learning process, do not use weights for stop criterion
- after some iterations ($\widetilde{E}(w^*)$ seems to be constant) take smaller parameter η for better approaching to w^*

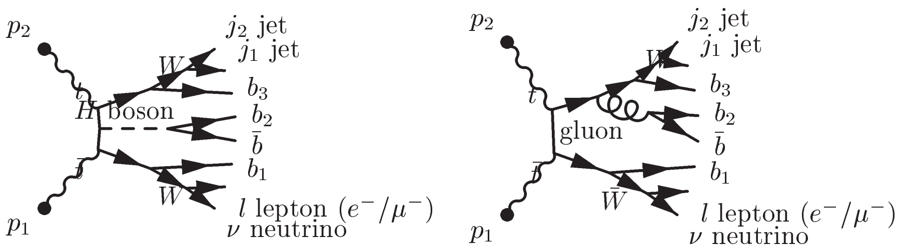
Conclusion of NN part

- **WHEN USE NEURAL NETWORKS?**
  1. try clasical numerical, statistical methods to solve your problem
  2. if 1) fails, better understand your problem
  3. if 2) fails, try neural networks
- **BUT !**
  - make correct dichotomy of testing and learning data (they schould have the same distribution of all features, number of learning samples should be comparable to number of testing samples)
  - find convenient size of neural network (larger does not imply better, find border of overlearning area)
  - stop learning proces if value of error function seems to be stable (do not expect weight convergence)
  - there exist another neural network models than MLP, try them.

Higgs boson search

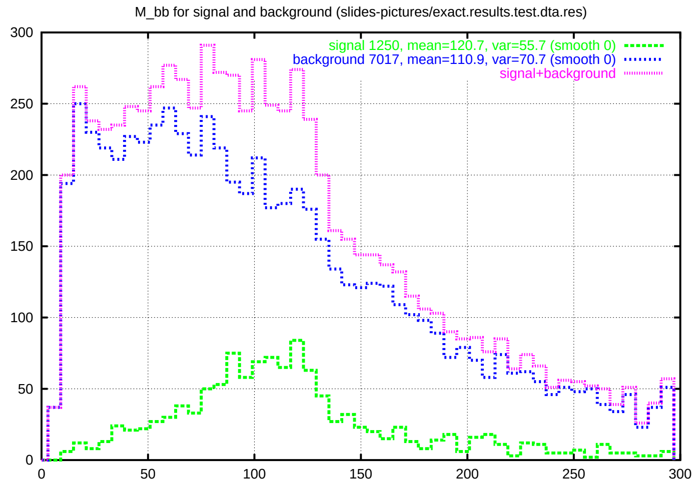
aim: separate (at least partly) signal and background and detect improvement of peak in $M_{b\bar{b}}$ distribution

Topology for signal and background events



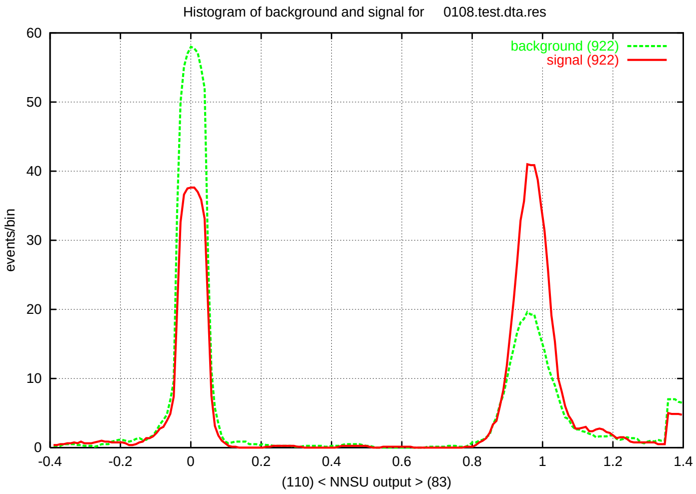
input variables: (η, ϕ, pT) of b, $\bar{b}$, b_1, b_2, j_1, j_2, l, ν (e.g. 24 real numbers)

M_{bb} distribution of original data



histogram for 1250 signals and 7017 background events (events with pseodorapidity out of $(-2.5, 2.5)$, with $P_{T(l=b)} < 20GeV$, $P_{T(l=muon)} < 6GeV$ and $P_{T(some\ jet)} < 15GeV$ are ignored)

Neural network output



signal should be mapped to 1, background should be mapped to 0 (in ideal case)

best signal window: $(0.5, 1.2)$

best background window: $(-0.2, 0.4)$

M_{bb} distribution over

best background window | best signal window