Function Classification with Neural Networks

Pascal S.P. Steger

Theory, Simulation, Programming of Neural Networks, 2010

ETH Zürich

1. Introduction
   - Neural Network: Topology
   - Backpropagation
2. Methods
   - Functions
   - Program Flux
3. Output
   - Logic Functions on Two Input Neurons (a, b)
   - Output Window
   - Applied on Training Data
   - Applied on Noisy Data
4. Errors
   - Parametrization of Sigmoid Function
   - Learning Rate, Number of Iterations
   - Number of Hidden Neurons
5. To Be Done
   - Projects

Literature

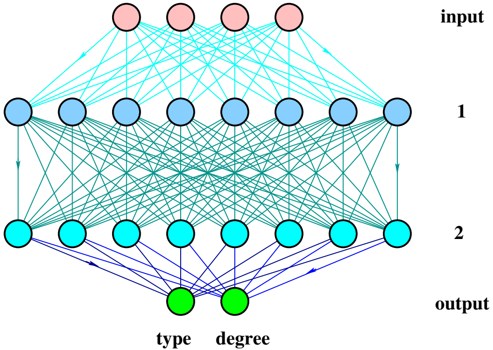
- R. D. Jones et al.
  Function approximation and time series prediction with neural networks.
  Proc. Int. Joint Conf. on Neural Networks, 1990.

- U. Lämmel and J. Cleve.
  Künstliche Intelligenz.
  Carl Hanser Verlag München Wien, 2004.

- R. Rojas.
  Neural Networks - A Systematic Introduction.
  Springer-Verlag, Berlin, New-York, 1996.

- R. Stoop.
  Theorie und Simulation Neuronaler Netze.
  lecture notes, 2010.

Neural Network: Topology

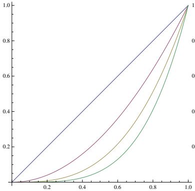


Backpropagation

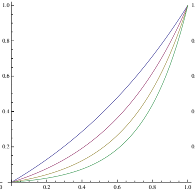
```
ohh   =   sigmoid[whh.in];
oh    =   sigmoid[wh.ohh];
out   =   sigmoid[wo.oh];

e     =   t - out;
od    =   e out (1 - out);
hd    =   oh (1 - oh) [wo'].od;
hhd   =   ohh (1 - ohh) [wh'].hd;

wo   +=   eta Outer[Times, od, oh];
wh   +=   eta Outer[Times, hd, ohh];
whh  +=   eta Outer[Times, hhd, in];
```

1. Introduction
   - Neural Network: Topology
   - Backpropagation
2. Methods
   - Functions
   - Program Flux
3. Output
   - Logic Functions on Two Input Neurons (a, b)
   - Output Window
   - Applied on Training Data
   - Applied on Noisy Data
4. Errors
   - Parametrization of Sigmoid Function
   - Learning Rate, Number of Iterations
   - Number of Hidden Neurons
5. To Be Done
   - Projects

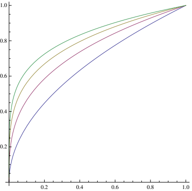
Functions



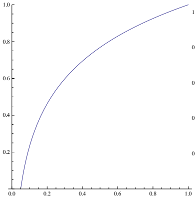
x^n



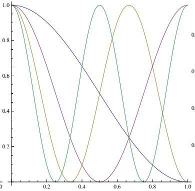
$\exp(nx)$



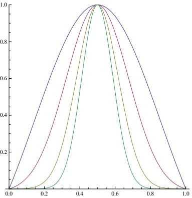
$x^{1/(1+n)}$



$\log(nx)$

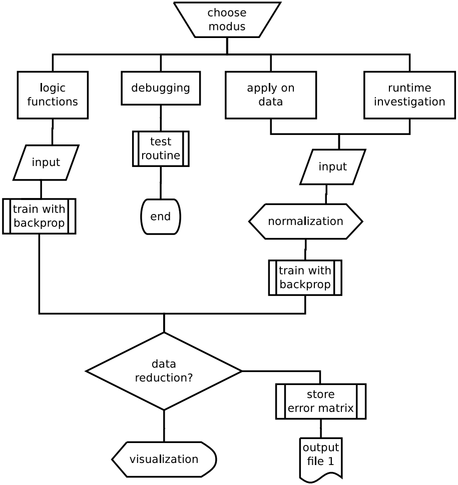


$\cos(n\pi x)$



$e^{(-(2n(x-0.5))^2)}$

Program Flux



Input

$$
\begin{aligned}
\mathbf{r}'_i &= \mathbf{r}_i - \tilde{\mathbf{r}}, \\
\mathbf{r}''_i &= \mathbf{s} \cdot \mathbf{r}'_i, \\
\mathbf{e}''_i &= \mathbf{s} \cdot \mathbf{e}'_i,
\end{aligned}
$$

original			normalized		
(6.7,49.6)	±	(1.7,2.5)	(0.0,0.4)	±	(0.02,0.02)
(46.6,69.8)	±	(2.7,2.9)	(0.4,0.6)	±	(0.03,0.03)
(93.5,20.2)	±	(1.6,3.3)	(1.0,0.1)	±	(0.01,0.03)
(58.5,96.2)	±	(4.7,0.8)	(0.5,1.0)	±	(0.05,0.00)
(23.7,10.0)	±	(2.5,2.8)	(0.1,0.0)	±	(0.02,0.03)
...	±	...	...	±	...

Training

```
void train(int nitmax, double a, double eta) {
    for (int nit = 0; nit < nitmax; ++nit) {
        int i = (ttout.size() * Math.random());
        setInput(ttin.elementAt(i));
        setTarget(ttout.elementAt(i));

        fireCascade(a);
        getErr();
        correctW(eta);
    }
}
```

findFunction

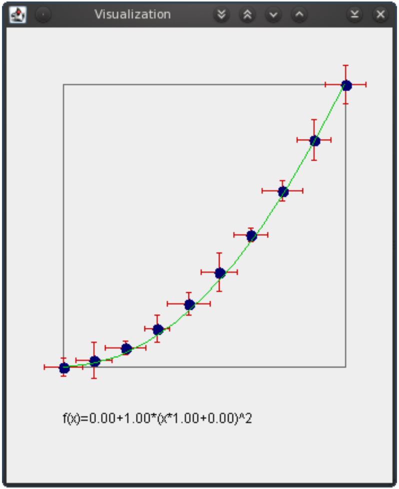
```
@Override public SFunction findFunction() {
    double a = 0.2, eta = 0.9;
    int nitmax = 10000;
    generateTraining();
    n.train(nitmax, a, eta);
    n.setInput( in );
    Matrix fin = new Matrix(n.fireCascade(a));
    double type = fin.get(0, 0) * 6.0;
    double deg = fin.get(1, 0) * 4.0;
    return new SFunction(type, deg);
}
```

1. Introduction
   - Neural Network: Topology
   - Backpropagation
2. Methods
   - Functions
   - Program Flux
3. Output
   - Logic Functions on Two Input Neurons (a, b)
   - Output Window
   - Applied on Training Data
   - Applied on Noisy Data
4. Errors
   - Parametrization of Sigmoid Function
   - Learning Rate, Number of Iterations
   - Number of Hidden Neurons
5. To Be Done
   - Projects

Logic Functions on Two Input Neurons (a, b)

$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	error e
0	0	0	0	0.0000
1	0	0	0	0.0005
...				
1	0	1	0	0.0003
0	1	1	0	**0.0021**
1	1	1	0	0.0007
0	0	0	1	0.0009
1	0	0	1	**0.0010**
0	1	0	1	0.0004
...				
0	1	1	1	0.0006
1	1	1	1	0.0000

Output Window



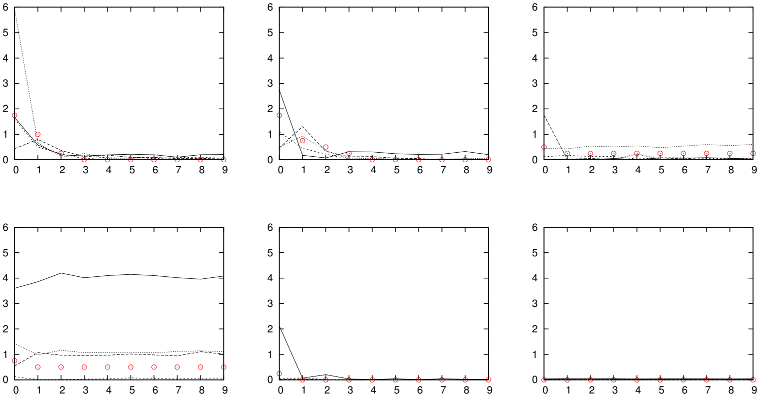
f(x)=0.00+1.00*(x*1.00+0.00)^2

Applied on Training Data

type	deg	found	
1	1	0.968	1.018
1	2	1.000	2.022
1	3	1.024	3.035
1	4	1.019	3.982
2	1	2.003	1.016
2	2	2.019	2.012
2	3	1.983	2.989
2	4	2.010	3.938
3	1	2.955	1.041
3	2	3.039	2.008
3	3	2.972	3.029
3	4	3.029	3.901

type	deg	found	
4	1 . . . 4	**4.001**	**2.587**
5	1	5.010	1.075
5	2	5.026	2.065
5	3	4.997	3.086
5	4	4.991	3.974
6	1	5.906	1.077
6	2	5.965	2.089
6	3	5.974	3.061
6	4	5.938	3.946

Applied on Noisy Data

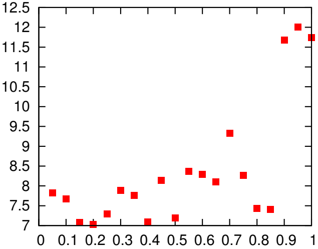


$$10(1-\delta)\%$$

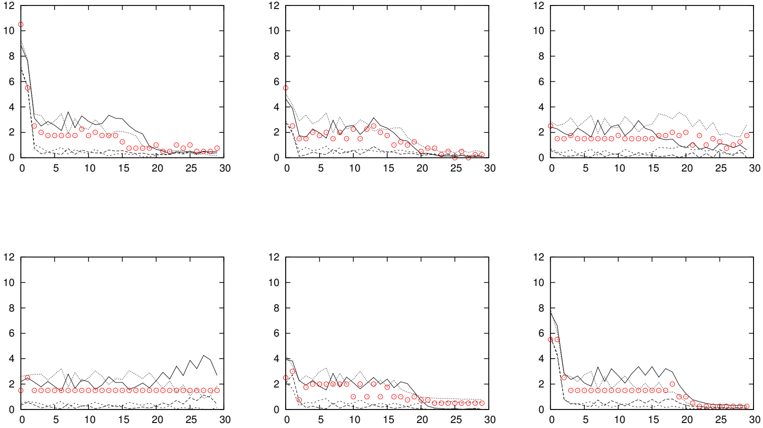
1. Introduction
   - Neural Network: Topology
   - Backpropagation
2. Methods
   - Functions
   - Program Flux
3. Output
   - Logic Functions on Two Input Neurons (a, b)
   - Output Window
   - Applied on Training Data
   - Applied on Noisy Data
4. Errors
   - Parametrization of Sigmoid Function
   - Learning Rate, Number of Iterations
   - Number of Hidden Neurons
5. To Be Done
   - Projects

Parametrization of Sigmoid Function

$$\begin{aligned}\mathrm{sig}_a(x) &\equiv \frac{1}{1+\exp(-ax)} \\ \mathrm{sig}'_a(x) &= a\cdot\mathrm{sig}_a(x)[1-\mathrm{sig}_a(x)]\end{aligned}$$



Learning Rate, Number of Iterations



$\cdot 1000$

Number of Hidden Neurons

$n_{\text{hid},(1,2)}$	error	time [s]
1	86.23	1.7
2	30.56	1.9
3	14.84	2.5
4	12.38	2.9
5	9.31	3.5
6	10.79	4.0
7	10.00	4.7
8	9.32	5.3
9	9.33	6.0
10	7.32	6.7
15	7.64	11.3
20	6.58	17.5

Projects

- more sampling points
- random input
- linear superposition
- three-dimensional data
- holographic neuron (with 2D FFT)
- ...

ETH Zürich