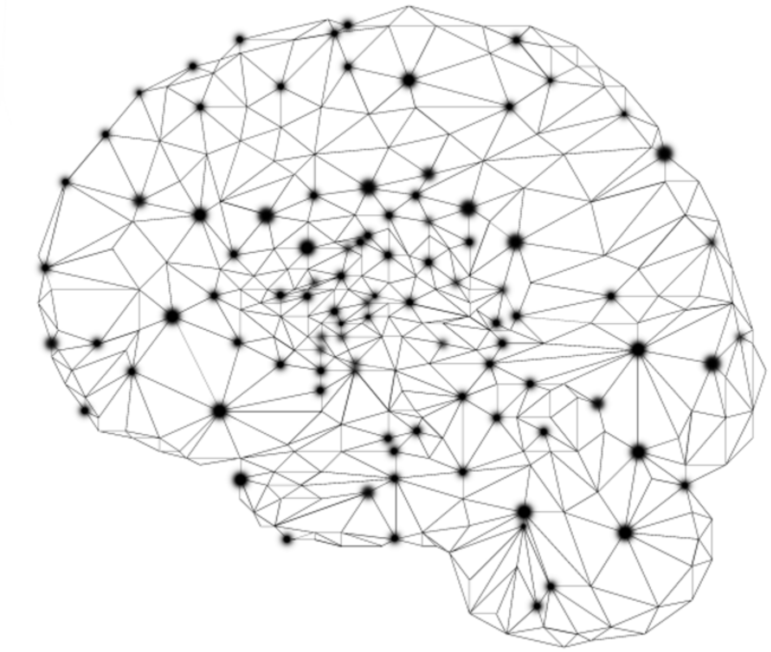
NEURAL NETWORKS

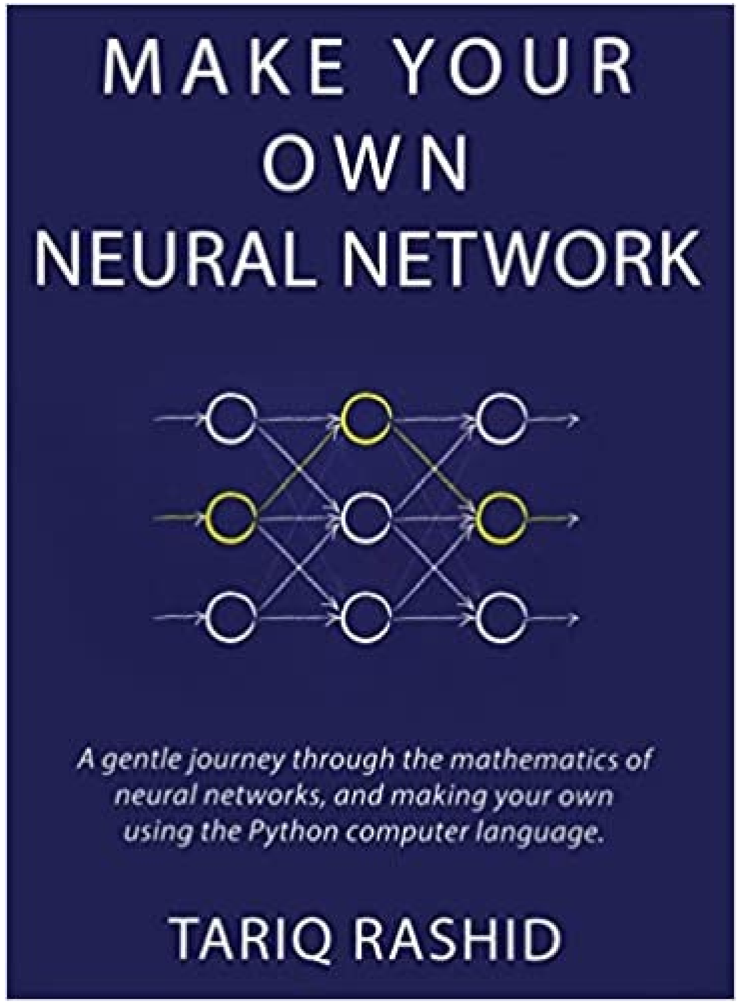
Lecture 7: Training and Backpropagation



Marcel Völschow
Hochschule für Angewandte Wissenschaften Hamburg
22.11.2023

HAW HAMBURG

IF YOU WANNA LEARN MORE …

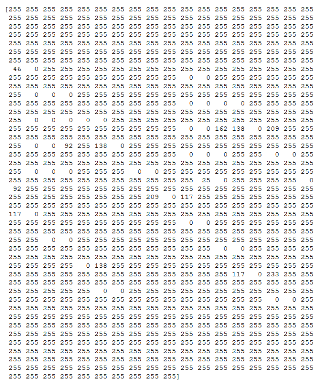


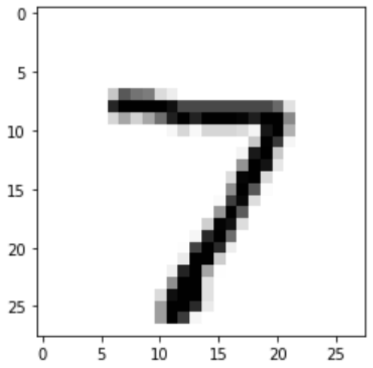
WHAT IS A NN?

A NN is a chain of linear and non-linear transformations applied to an input vector

SIGNAL FEED-FORWARD

I. Input layer (784 nodes)





Dot product: wih * input

Apply sigmoid function

II. Hidden layer (100 nodes)

```
[2.41792527e-03 1.10299319e-12 9.58926222e-05 2.41395445e-11
 7.78628311e-05 4.44162273e-20 9.99909568e-01 2.67383949e-12
 1.24303873e-01 5.91268930e-05 1.23644027e-01 6.15685354e-11
 6.33027791e-07 1.22016810e-07 2.98387247e-07 9.45567933e-05
 8.68749020e-05 9.99730834e-01 3.28582482e-11 5.89070143e-05
 5.13603744e-06 3.99916687e-08 3.21972056e-06 1.00000000e+00
 4.15461563e-03 2.69911130e-04 1.62828675e-09 4.39410565e-07
 3.94994661e-04 9.99022659e-01 2.96670603e-06 3.06577608e-06
 2.82313871e-09 9.99533298e-01 2.19969831e-07 1.18015670e-08
 3.58296737e-14 4.35269474e-13 1.47678576e-07 6.95528880e-09
 1.23659513e-12 1.44085848e-04 5.48810614e-06 2.04256443e-10
 1.16948194e-06 2.31804989e-07 2.48182532e-07 2.24920806e-13
 4.84913342e-06 4.30006189e-06 4.13803950e-07 1.11251392e-05
 3.31332897e-08 1.93942672e-01 1.65626312e-09 1.78564511e-08
 3.30632034e-08 3.91382983e-06 2.05142528e-11 2.14213678e-04
 5.44112858e-08 1.20251474e-07 8.05326951e-04 3.84056809e-10
 9.72551232e-01 2.97104772e-11 4.09035111e-13 7.18237324e-11
 6.22399765e-08 2.13573010e-08 6.16334160e-05 6.53225739e-13
 1.45819673e-08 9.99204220e-02 1.38881809e-04 7.86723727e-09
 9.99999834e-01 1.89808632e-15 1.42010176e-02 1.08437044e-04
 8.00671644e-10 1.10047033e-05 3.00049192e-17 3.12103408e-06
 4.93474020e-09 8.88113435e-02 1.02655128e-09 2.73271118e-06
 9.99707382e-01 7.41835912e-14 1.20566212e-06 1.05372275e-07
 2.89354798e-06 9.70455447e-01 2.12703667e-18 3.64019170e-15
 3.38688270e-05 5.79111423e-06 8.22853229e-09 9.99999991e-01]
```

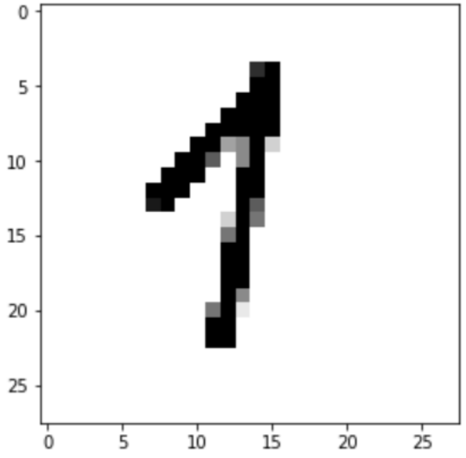
Dot product: who * hidden

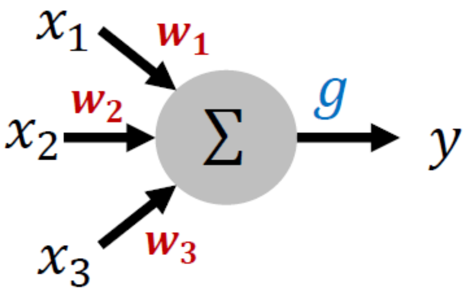
Apply sigmoid function

III. Output layer (10 nodes)

0.0114
0.0066
0.0010
0.0050
0.0007
0.0012
0.0004
0.9941
0.0070
0.0002

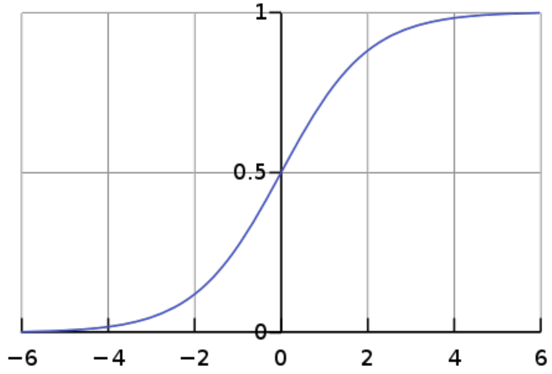
ACTIVATION FUNCTIONS



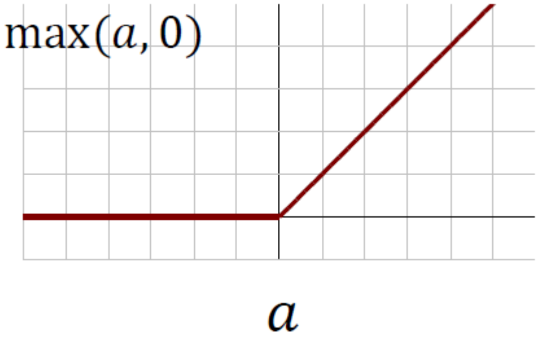


Activation functions g model the response of a neuron

Sigmoid function



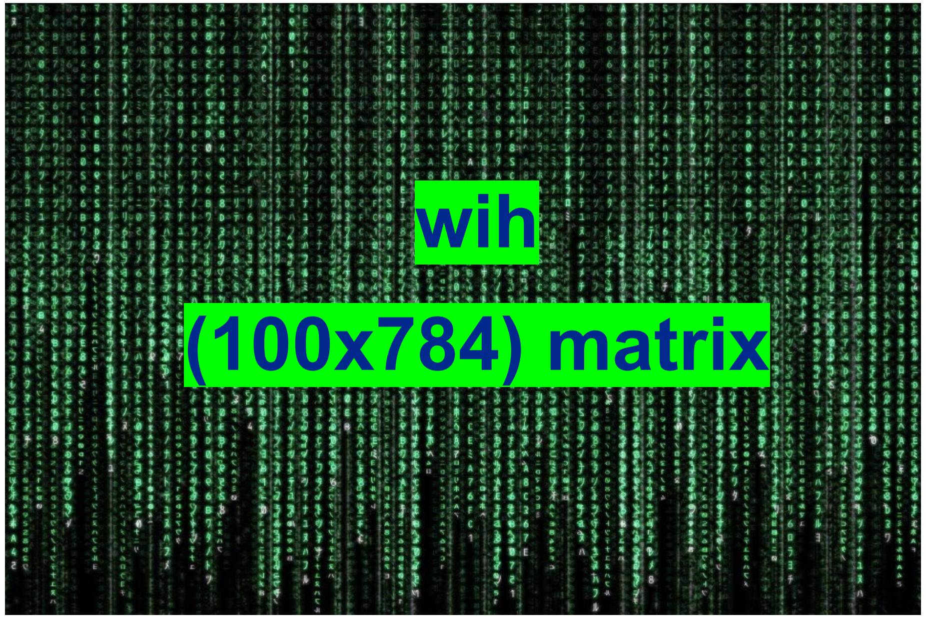
Rectified linear (ReLu)

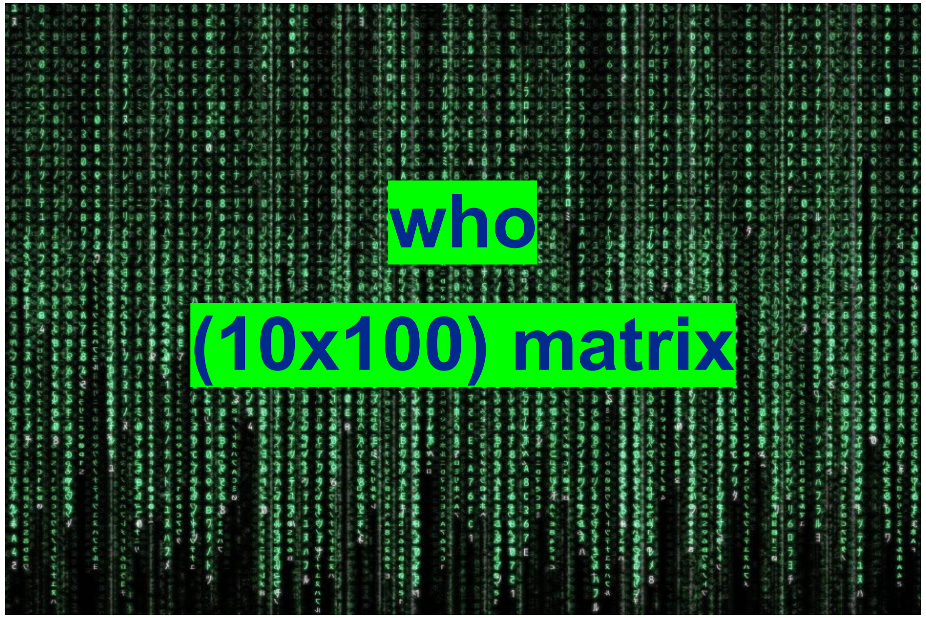


HAW HAMBURG

WHAT'S THE CATCH?

We need to find all entries of the weight matrices which represent the CNNs memory:





78400 + 1000 = 79400 unknown parameters …

=> Backpropagation

HAW HAMBURG

BACK TO 1986

Letter | Published: 09 October 1986

Learning representations by back-propagating errors

David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams

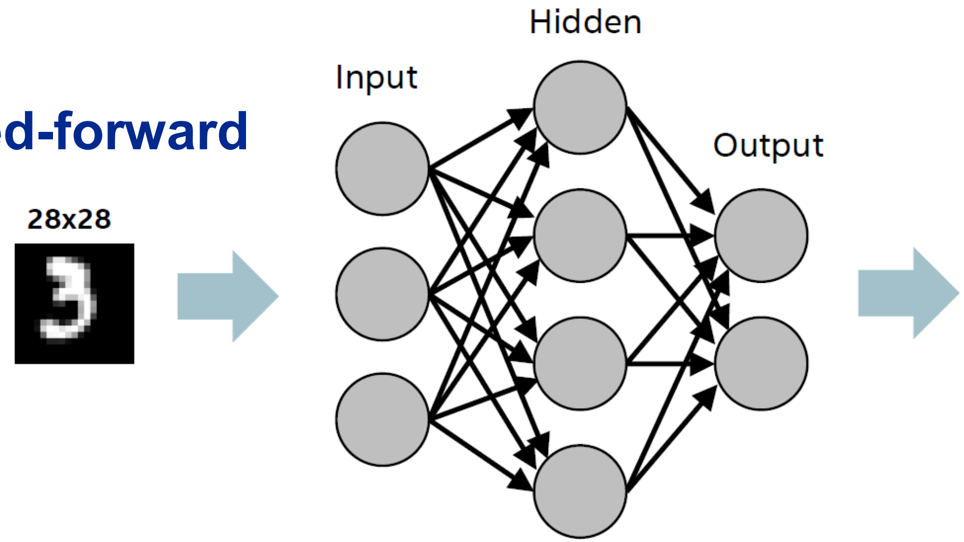
Nature **323**, 533–536 (1986) | Cite this article

Abstract

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

TRAINING

Feed-forward



Output O

0.0	0
0.1	0
0.0	0
0.3	**1**
0.1	0
0.1	0
0.0	0
0.0	0
0.4	0
0.0	0

Target T

Loss function

Difference between current and target output

LOSS FUNCTION DEFINITIONS

Sum of …

1. **… differences**
2. **… absolute differences**
3. **… squared differences**

$$\left|\left(\begin{array}{c} \text{Output O} \\ \begin{bmatrix} 0.0 \\ 0.1 \\ 0.0 \\ 0.3 \\ 0.1 \\ 0.1 \\ 0.0 \\ 0.0 \\ \mathbf{0.4} \\ 0.0 \end{bmatrix} \end{array} - \begin{array}{c} \text{Target T} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}\right)^2\right|$$

Compare calculation of the standard deviation …

OUR LOSS FUNCTION

Output layer components

$$L = \sum_{n} (O_n - T_n)^2$$

Sum over all output layer components

Target components

Which set of network weights minimizes the loss function?

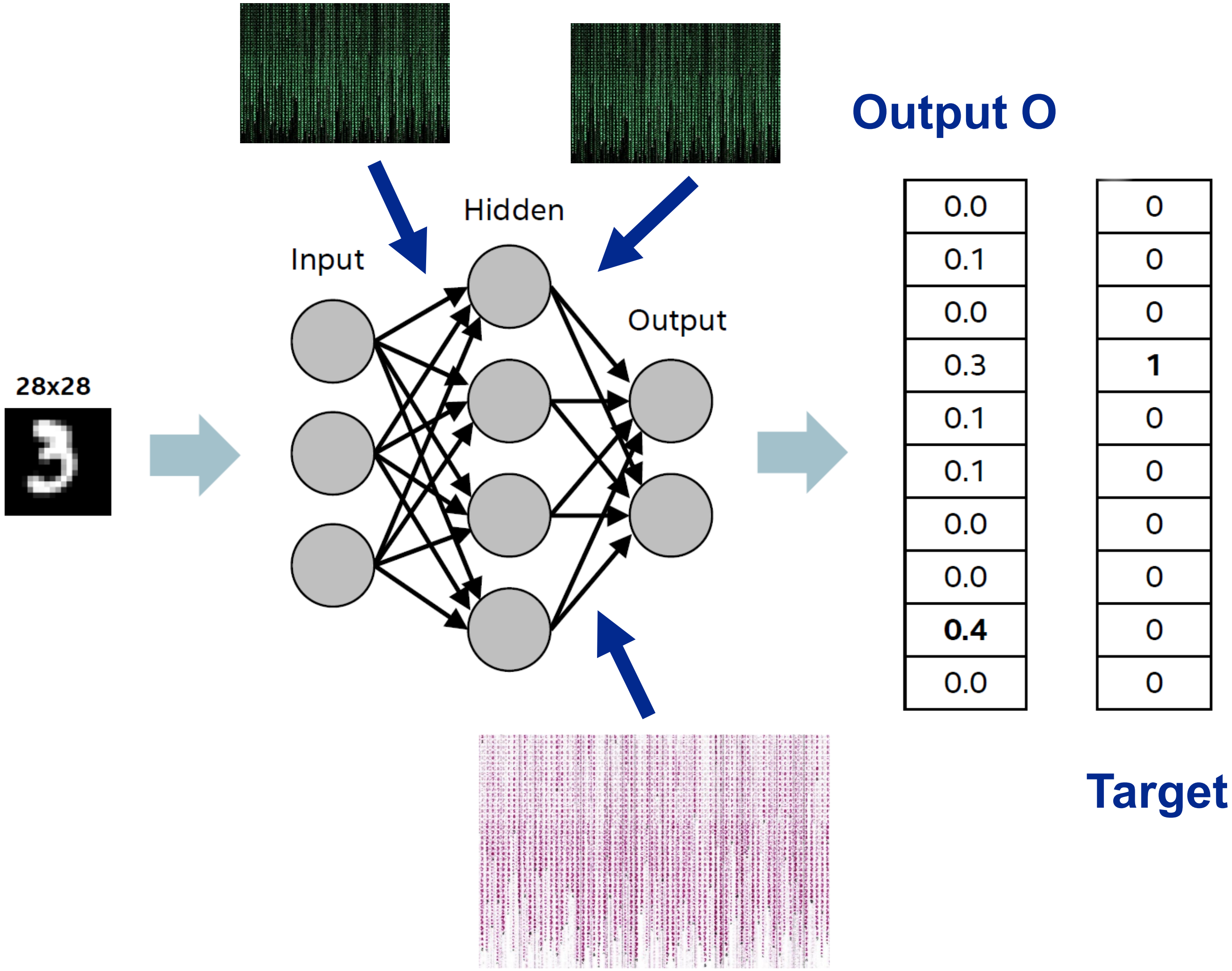
HAW HAMBURG

BACKPROPAGATION

Matrices feed the signal **FORWARD** through the network, we get the loss (O-T)

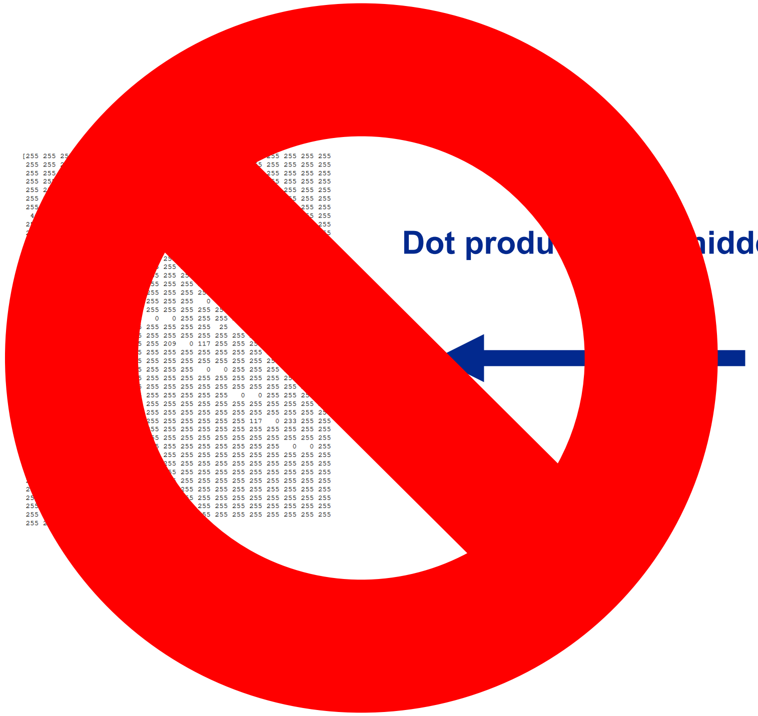
We would like to distribute the loss according to the weights that feed the signal forward

Feed the loss **BACKWARD** through the network using **inverse matrices**



LOSS BACKPROPAGATION

III. Input loss (784 nodes)



II. Hidden loss (100 nodes)

```
[2.41792527e-03 1.10299319e-12 9.58926222e-05 2.41395445e-11
 7.78628311e-05 4.44162273e-20 9.99909568e-01 2.67383949e-12
 1.24303873e-01 5.91268930e-05 1.23644027e-01 6.15685354e-11
 6.33027791e-07 1.22016810e-07 2.98387247e-07 9.45567933e-05
 8.68749020e-05 9.99730834e-01 3.28582482e-11 5.89070143e-05
 5.13603744e-06 3.99916687e-08 3.21972056e-06 1.00000000e+00
 4.15461563e-03 2.69911130e-04 1.62828675e-09 4.39410565e-07
 3.94994661e-04 9.99022659e-01 2.96670603e-06 3.06577608e-06
 2.82313871e-09 9.99533298e-01 2.19969831e-07 1.18015670e-08
 3.58296737e-14 4.35269474e-13 1.47678576e-07 6.95528880e-09
 1.23659513e-12 1.44085848e-04 5.48810614e-06 2.04256443e-10
 1.16948194e-06 2.31804989e-07 2.48182532e-07 2.24920806e-13
 4.84913342e-06 4.30006189e-06 4.13803950e-07 1.11251392e-05
 3.31332897e-08 1.93942672e-01 1.65626312e-09 1.78564511e-08
 3.30632034e-08 3.91382983e-06 2.05142528e-11 2.14213678e-04
 5.44112858e-08 1.20251474e-07 8.05326951e-04 3.84056809e-10
 9.72551232e-01 2.97104772e-11 4.09035111e-13 7.18237324e-11
 6.22399765e-08 2.13573010e-08 6.16334160e-05 6.53225739e-13
 1.45819673e-08 9.99204220e-02 1.38881809e-04 7.86723727e-09
 9.99999834e-01 1.89808632e-15 1.42010176e-02 1.08437044e-04
 8.00671644e-10 1.10047033e-05 3.00049192e-17 3.12103408e-06
 4.93474020e-09 8.88113435e-02 1.02655128e-09 2.73271118e-06
 9.99707382e-01 7.41835912e-14 1.20566212e-06 1.05372275e-07
 2.89354798e-06 9.70455447e-01 2.12703667e-18 3.64019170e-15
 3.38688270e-05 5.79111423e-06 8.22853229e-09 9.99999991e-01]
```

Dot product: who^{-1} * output

I. Output loss (10 nodes)

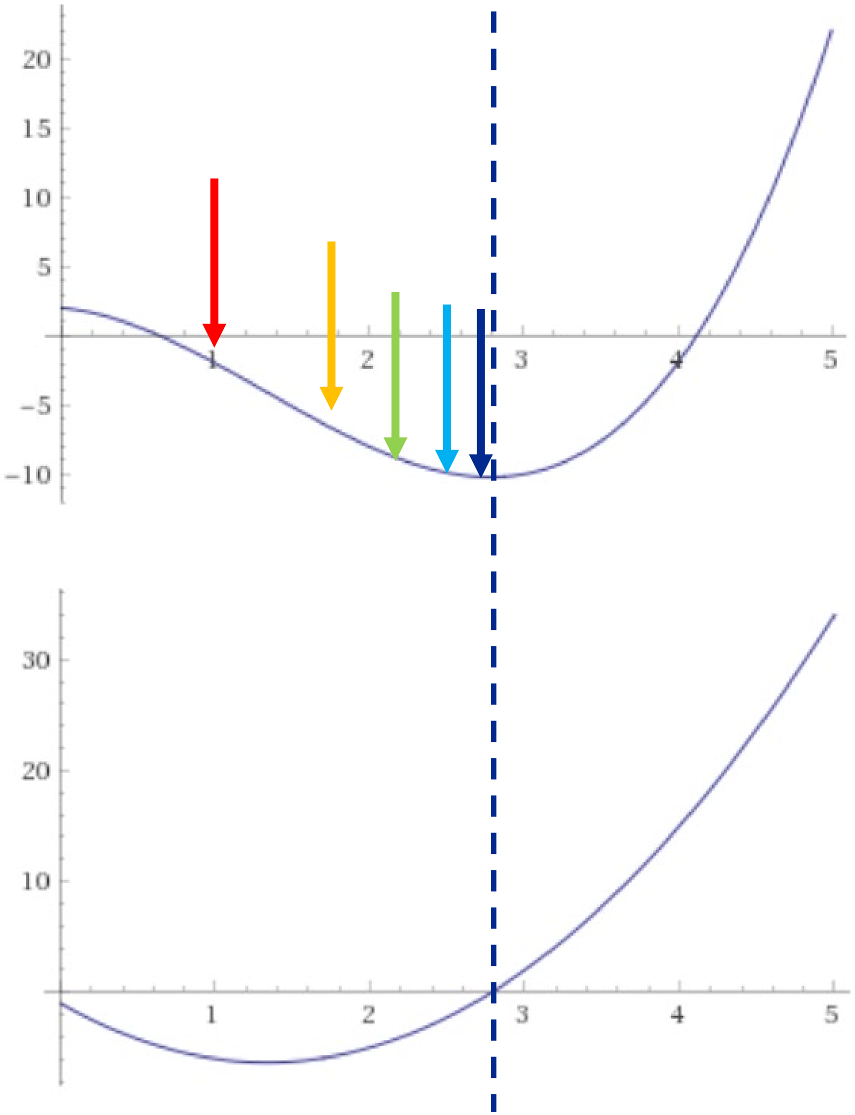
```
0.7943
0.0251
0.8141
0.9993
0.9961
0.8981
0.3300
0.5485
0.9814
0.4532
```


HAW HAMBURG

1D OPTIMIZATION

f(x)

f'(x)



Start at $x_0 = 1$

Derivative tells us how to get to the minimum

f'(x)<0 => Minimum to the right
f'(x)>0 => Minimum to the left

Go one step closer to the minimum:

$$\Delta x = -\alpha \frac{df}{dx}(1)$$

New position: $x_1 = x_0 + \Delta x$

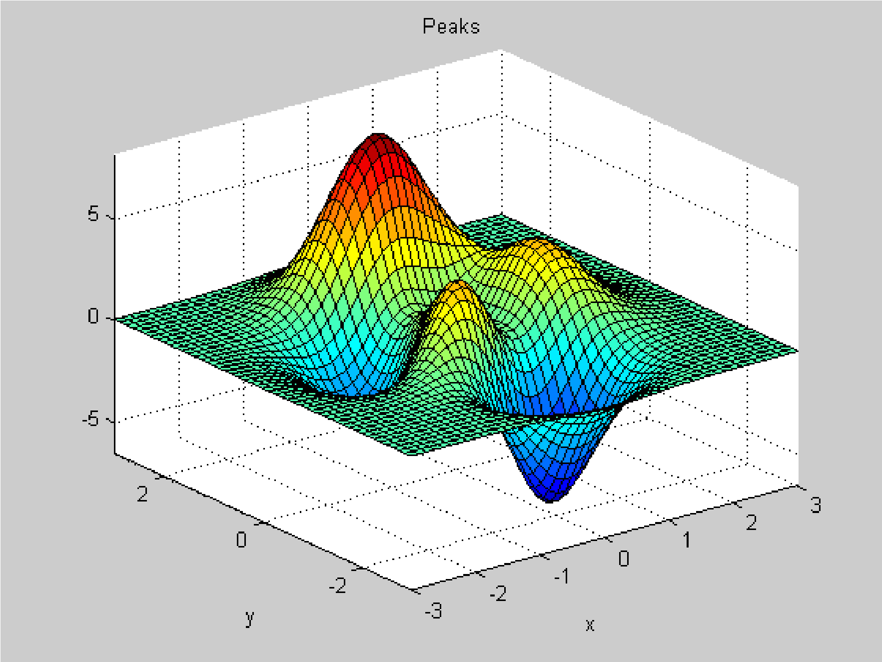
Evaluate derivative, next step, …

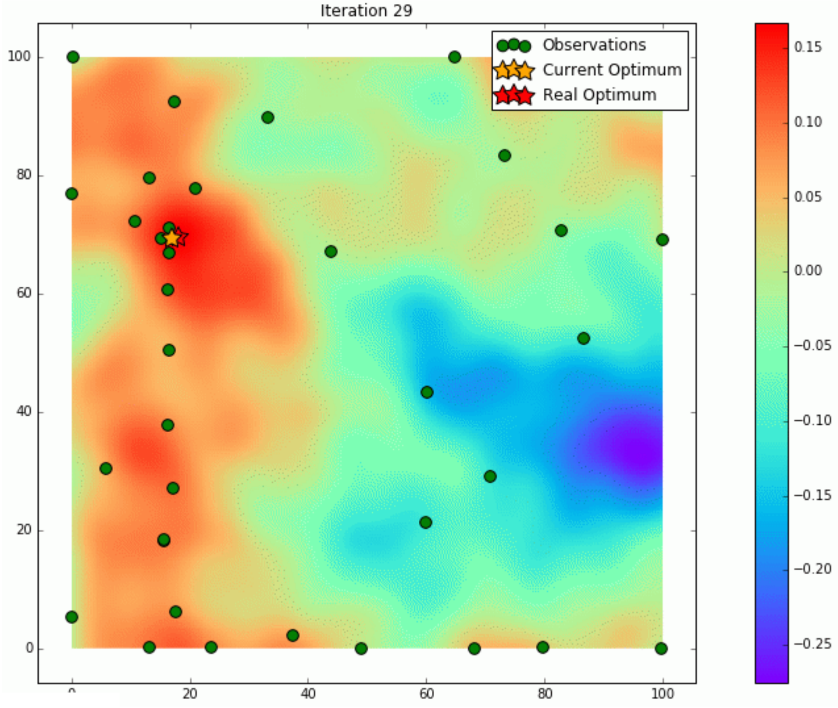
HAW HAMBURG

2D OPTIMIZATION

Compare geography vs. topographical map …

$$f(x, y)$$





Use the **gradient** to descent towards the minimum:

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

APPLICATION TO OUR NETWORK 1

Optimize the weights of who (= wij) to get the output closest to the training image:

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_n (O_n - T_n)^2$$

One line of who produces one entry of the loss function:

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (O_j - T_j)^2$$

Apply the chain rule:

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial O_j} \frac{\partial O_j}{\partial w_{ij}} = -2\,(T_j - O_j)\,\frac{\partial O_j}{\partial w_{ij}}$$

APPLICATION TO OUR NETWORK 2

Sigmoid activation function and derivative:

$$f(x) = \frac{1}{1 + \exp(-x)}$$

$$f'(x) = f(x)\ (1 - f(x))$$

Output layer is given by f (matrix who times hidden layer):

$$O_j = f\left(\sum_i w_{ij} H_j\right)$$

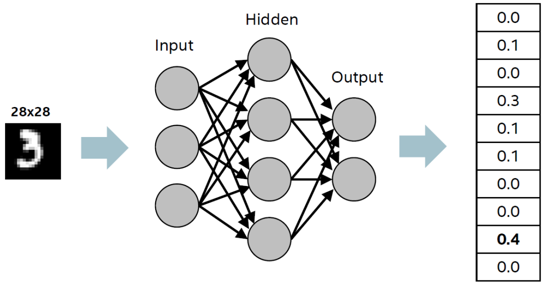
Use that, the derivative of f and the chain rule again:

$$\frac{\partial L}{\partial w_{ij}} = -2\,(T_j - O_j)\ f\left(\sum_i w_{ij} H_i\right)\ \left(1 - f\left(\sum_i w_{ij} H_i\right)\right)\ H_i$$

HAW HAMBURG

TRAINING STRATEGY SUMMARY

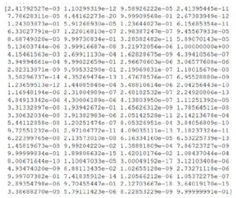
1. Feed-forward the training image



2. Calculate the loss, i.e. the difference between output and target

$$\left(\begin{bmatrix} 0.0 \\ 0.1 \\ 0.0 \\ 0.3 \\ 0.1 \\ 0.1 \\ 0.0 \\ 0.0 \\ \mathbf{0.4} \\ 0.0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

3. Use the inverse of who to propagate the loss back to the hidden layer



Dot product: who^{-1} * output

```
0.7943
0.0251
0.8141
0.9993
0.9961
0.8981
0.3300
0.5485
0.9814
0.4532
```

4. Apply an optimization algorithm to modify all elements of wih and who

$$\frac{\partial L}{\partial w_{ij}} = -2\,(T_j - O_j)\; f\left(\sum_i w_{ij} H_i\right)\left(1 - f\left(\sum_i w_{ij} H_i\right)\right) H_i$$