Learning &
Adaptive Systems

Neural Networks

Zhenrong Lang[1] and Rajesh Sharma[2]

Introduction to Machine Learning

[1]Department of Information Technology and Electrical Engineering, ETH Zürich
[2]Department of Computer Science, ETH Zürich

Where we are

- Lectures
  - Week 5 Kernels
  - Week 6 **(This Week!) Neural Networks I, II**
  - Week 7 Neural Networks III, IV
- Tutorials
  - Week 5 Classification and Model Selection
  - Week 6 **(Today!) Neural Network Basics**
  - Week 7 Homework Review, CNN and other Networks

Outline

- First session (Zhenrong)
  - Motivation for neural networks
  - Multilayer perceptrons
  - Activation functions
  - Forward and backward propagation
  - Gradient Descent
- Second session (Rajesh)
  - Code a simple neural network
    - **Click here for Notebook**
    - Data Generation and Visualization
    - Setting up Weights and Biases
    - Training Loop
    - Updating the network
    - Visualizing the results
  - Keras/Tensorflow demo

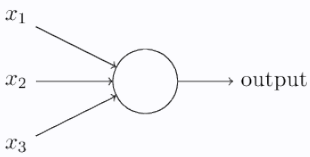
Motivation for neural networks

Motivation for neural networks

Up till now:

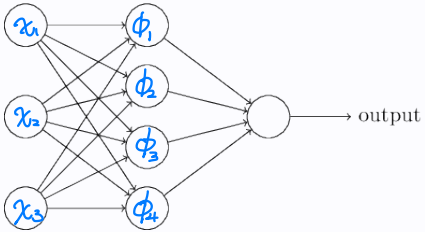
Linear:

$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$



Non-linear:

$f(\boldsymbol{x}) = g(\boldsymbol{\phi}(\boldsymbol{x}))$ with $g(\boldsymbol{z}) = \boldsymbol{w}^T \boldsymbol{z}$

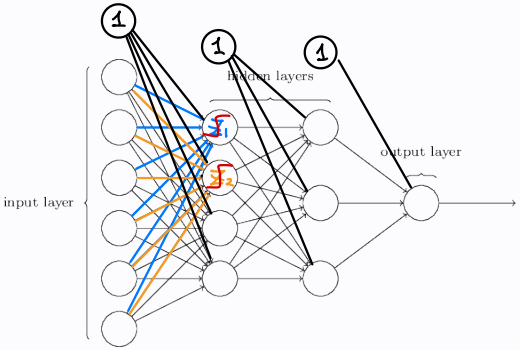


The features are hand-designed (chosen manually).

Goal: learn features from data automatically.

Multilayer Perceptrons

Multilayer Perceptrons



Node (neuron/perceptron)

pre-activation $z = \boldsymbol{w}^T \boldsymbol{v}$

apply activation: $v = \varphi(z)$

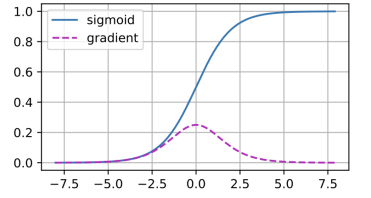
Bias-trick:

$\boldsymbol{w} = (w_0, w_1, ..., w_{d-1})^T$

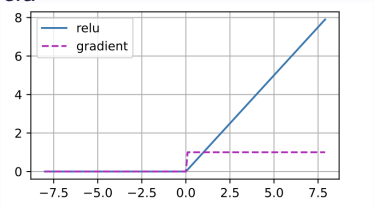
$\boldsymbol{x} = (1, x_1, ..., x_{d-1})^T$

Activation Functions

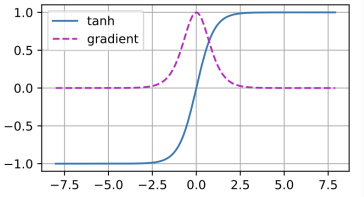
sigmoid



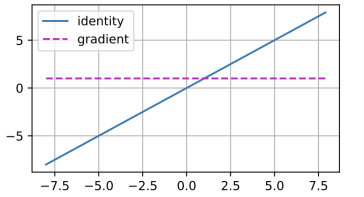
relu



tanh

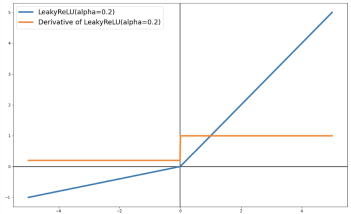


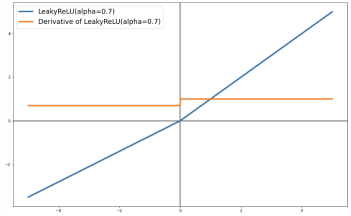
identity (is this a good activation function?)



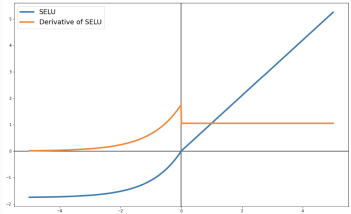
No, shouldn't use everywhere, otherwise linear

Activation Functions





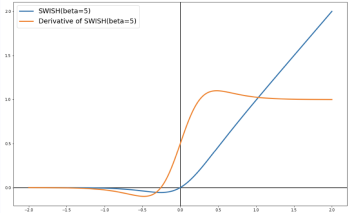
LeakyReLU $\begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$

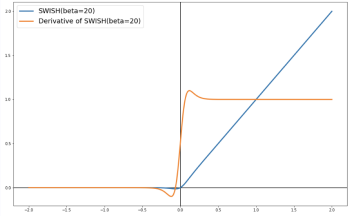


$\alpha = 1.6733;\ \lambda = 1.0507$

SeLU $\lambda \begin{cases} x & x > 0 \\ \alpha e^x - \alpha & x \leq 0 \end{cases}$

input
zero mean unit variance
stay
zero mean unit variance





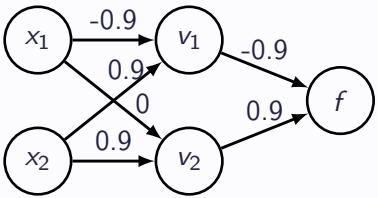
SWISH

$x \cdot \text{sigmoid}(\beta x) = \frac{x}{1 + \exp(-\beta x)}$

β trainable parameter

Forward and backward propagation

Forward propagation with a numerical example



activation function: ReLU

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Pre-activation:

$$\boldsymbol{z} = W^{(1)}\boldsymbol{x}$$

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

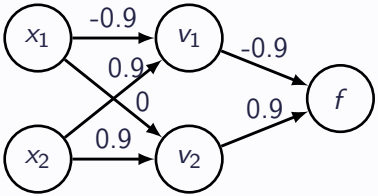
$$\begin{pmatrix} 0 \\ 0.9 \end{pmatrix} = \begin{pmatrix} -0.9 & 0.9 \\ 0 & 0.9 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Hidden Layer:

$$\boldsymbol{v} = \boldsymbol{\varphi}(\boldsymbol{z})$$

$$\begin{pmatrix} 0 \\ 0.9 \end{pmatrix} = \begin{pmatrix} \varphi(0) \\ \varphi(0.9) \end{pmatrix}$$

Forward propagation with a numerical example (cont.)



activation function: ReLU

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Output

$$f = W^{(2)}\boldsymbol{v}$$

$$f = \begin{pmatrix} W_{11}^{(2)} & W_{12}^{(2)} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

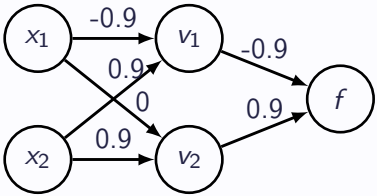
$$0.81 = \begin{pmatrix} -0.9 & 0.9 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0.9 \end{pmatrix}$$

Mean Square Error:

$$l = (y - f)^2$$

$$0.0361 = (1 - 0.81)^2$$

Backward propagation with a numerical example



activation function: ReLU

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$\frac{\partial I}{\partial W^{(2)}} = \frac{\partial I}{\partial f} \cdot \frac{\partial f}{\partial W^{(2)}}$$

$$= \frac{\partial (y-f)^2}{\partial f} \cdot \frac{\partial W^{(2)} \boldsymbol{v}}{\partial W^{(2)}}$$

$$= \begin{pmatrix} 0 \\ 0.9 \end{pmatrix} \cdot 2(0.81 - 1) = \begin{pmatrix} 0 \\ -0.342 \end{pmatrix}$$

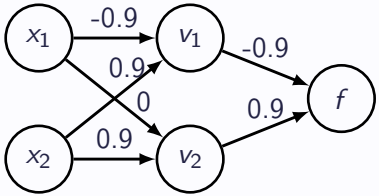
$$W^{(2)}_{new} = W^{(2)} - \eta \nabla_{W^{(2)}} I(W, x, y)$$

$$= W^{(2)} - \eta \left(\frac{\partial I}{\partial W^{(2)}}\right)^T$$

$$= \begin{pmatrix} -0.9 & 0.9 \end{pmatrix} - 0.2 \cdot \begin{pmatrix} 0 & -0.342 \end{pmatrix}$$

$$= \begin{pmatrix} -0.9 & 0.9684 \end{pmatrix}$$

Backward propagation with a numerical example (cont.)



activation function: ReLU

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$\frac{\partial l}{\partial W^{(1)}} = \frac{\partial l}{\partial f} \cdot \frac{\partial f}{\partial \boldsymbol{v}} \cdot \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial W^{(1)}}$$

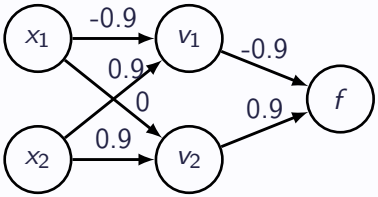
$$= \frac{\partial (y-f)^2}{\partial f} \cdot \frac{\partial W^{(2)}\boldsymbol{v}}{\partial \boldsymbol{v}} \cdot \frac{\partial \boldsymbol{\varphi}(\boldsymbol{z})}{\partial \boldsymbol{z}} \cdot \frac{\partial W^{(1)}\boldsymbol{x}}{\partial W^{(1)}}$$

$$= \boldsymbol{x} \cdot 2(f-y) \cdot W^{(2)} \cdot diag(\boldsymbol{\varphi}'(\boldsymbol{z}))$$

$$= \begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot 2(0.81-1) \cdot \begin{pmatrix} -0.9 & 0.9 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= -0.38 \cdot \begin{pmatrix} -0.9 & 0.9 \\ -0.9 & 0.9 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -0.342 \\ 0 & -0.342 \end{pmatrix}$$

Backward propagation with a numerical example (cont.)



activation function: ReLU

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$W_{new}^{(1)} = W^{(1)} - \eta \nabla_{W^{(1)}} l(W, x, y)$$

$$= W^{(1)} - \eta \left(\frac{\partial l}{\partial W^{(1)}}\right)^T$$

$$= \begin{pmatrix} -0.9 & 0.9 \\ 0 & 0.9 \end{pmatrix} - 0.2 \cdot \begin{pmatrix} 0 & 0 \\ -0.342 & -0.342 \end{pmatrix}$$

$$= \begin{pmatrix} -0.9 & 0.9 \\ 0.0684 & 0.9684 \end{pmatrix}$$

$$\begin{pmatrix} -0.9 & 0.9 \\ 0.0684 & 0.9684 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1.0368 \end{pmatrix}$$

$$\begin{pmatrix} -0.9 & 0.9684 \end{pmatrix} \cdot \begin{pmatrix} \varphi(0) \\ \varphi(1.0368) \end{pmatrix} = 1.004$$

Gradient Descent

Gradient Descent

Recall the following Gradient Descent Methods:
(Batch) Gradient Descent: uses all examples in the training data.
Stochastic Gradient Descent: uses one randomly chosen example.
Mark the following as (T)rue or (F)alse:

- Batch Gradient Descent is suited for large number of training data. F
- Batch Gradient Descent gives the global optimal solution given sufficient time. F non-convex
- Stochastic Gradient Descent tends to escape local minima. T randomness
- Stochastic Gradient Descent reaches convergence faster than Batch Gradient Descent. T fit in memory, update more regularly
- Batch Gradient Descent is faster and less computationally expensive. F

References

Neural Networks and Deep Learning by Michael Nielsen

Dive into Deep Learning by Aston Zhang

Deep Learning by Ian Goodfellow

Backup slide: numerator layout

Scalar by vector

$$\frac{\partial y}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top$$

Vector by vector

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial A\boldsymbol{x}}{\partial \boldsymbol{x}} = A$$

Scalar by matrix

$$\frac{\partial y}{\partial X} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{21}} & \cdots & \frac{\partial y}{\partial x_{p1}} \\ \frac{\partial y}{\partial x_{12}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{p2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{1q}} & \frac{\partial y}{\partial x_{2q}} & \cdots & \frac{\partial y}{\partial x_{pq}} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{a}^\top X \boldsymbol{b}}{\partial X} = \boldsymbol{b}\boldsymbol{a}^T$$