3 Neural Networks

PVK 2019: MAD

bacdavid@student.ethz.ch

gitlab.ethz.ch/bacdavid

Schedule

1. Theory
    1. Setup
    2. Neurons
    3. Neural Networks
    4. Operations on Neural Networks
    5. Gradient Descent
    6. Backpropagation
2. Exercises
3. Homework

Theory

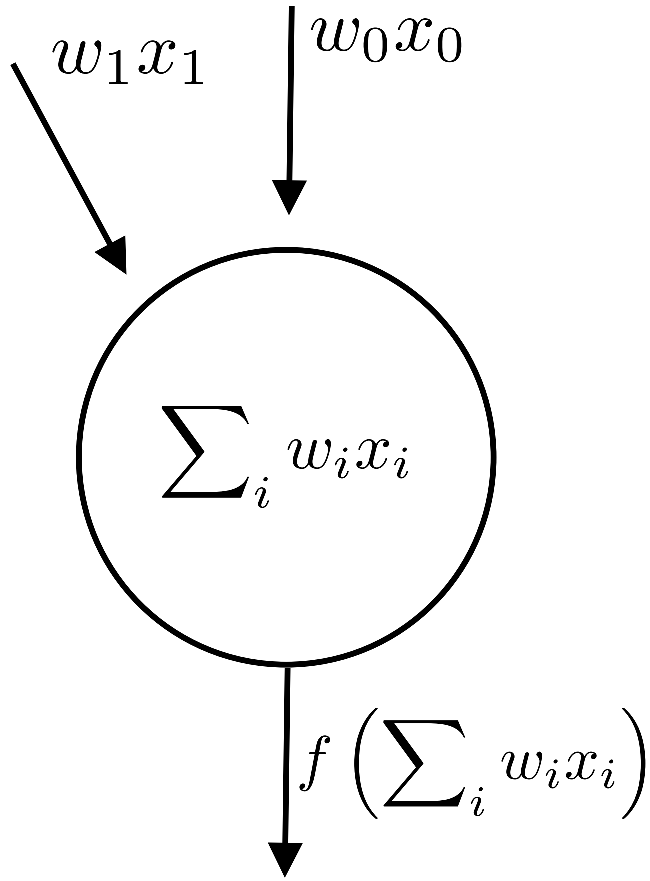
Setup

- Dataset $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
- Inputvector $\vec{x} = [x_1, \dots, x_n]^T$
- Weightvector $\vec{w} = [w_1, \dots, w_n]^T$
- Weightmatrix $W = [\vec{w}_1, \dots, \vec{w}_k]^T$
- Biasvector $\vec{b} = [b_1, \dots, b_k]^T$
- Intermediate output $\vec{o} = [o_1, \dots, o_k]^T$
- Final output y
- Activation function $f(\cdot)$
- Elementwise activation function $\vec{F}(\vec{v}) = [f(v_1), \dots]$

Neurons

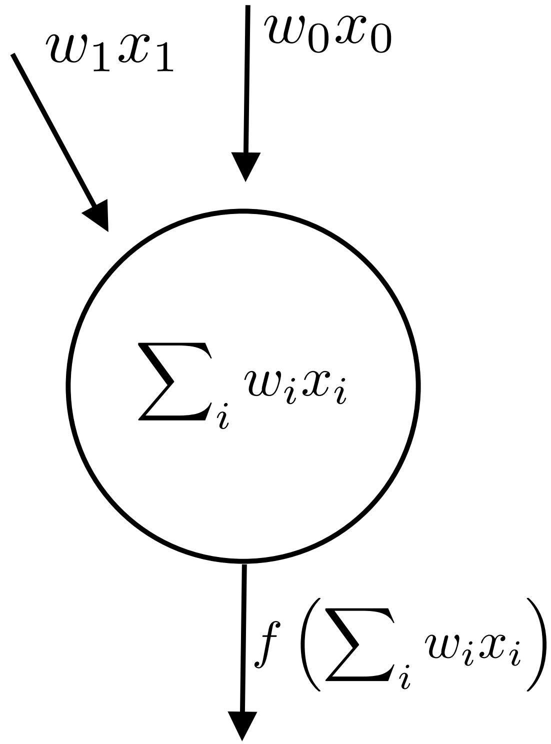
- Operation of each neuron:

1. Each input x_i is weighted by w_i
2. Then summed over
3. Then an activation function f is applied



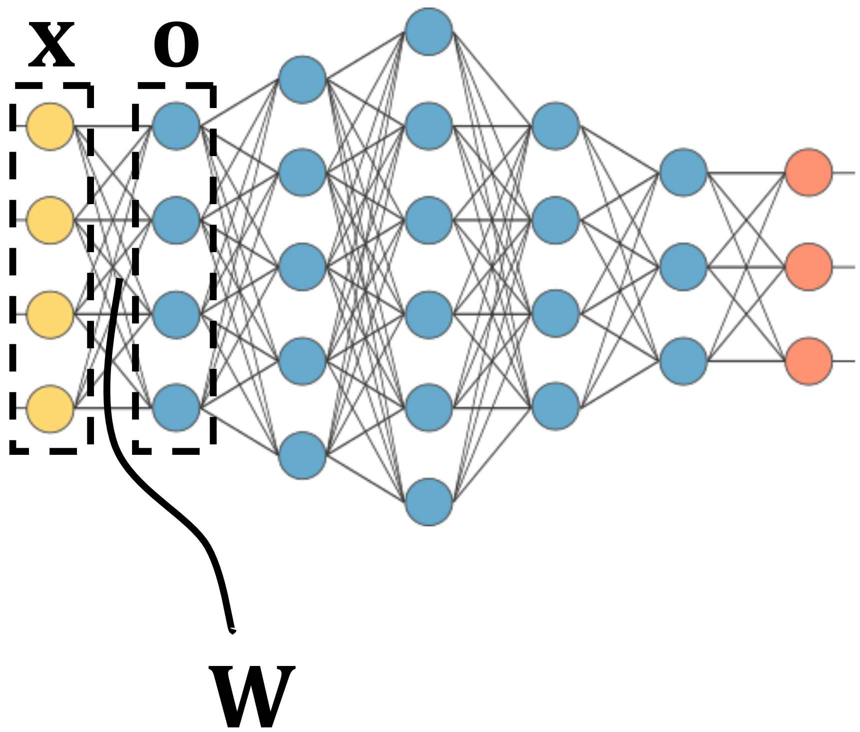
Neurons cont. Example

- Write the output of the neuron using $\vec{x}, \vec{w}$



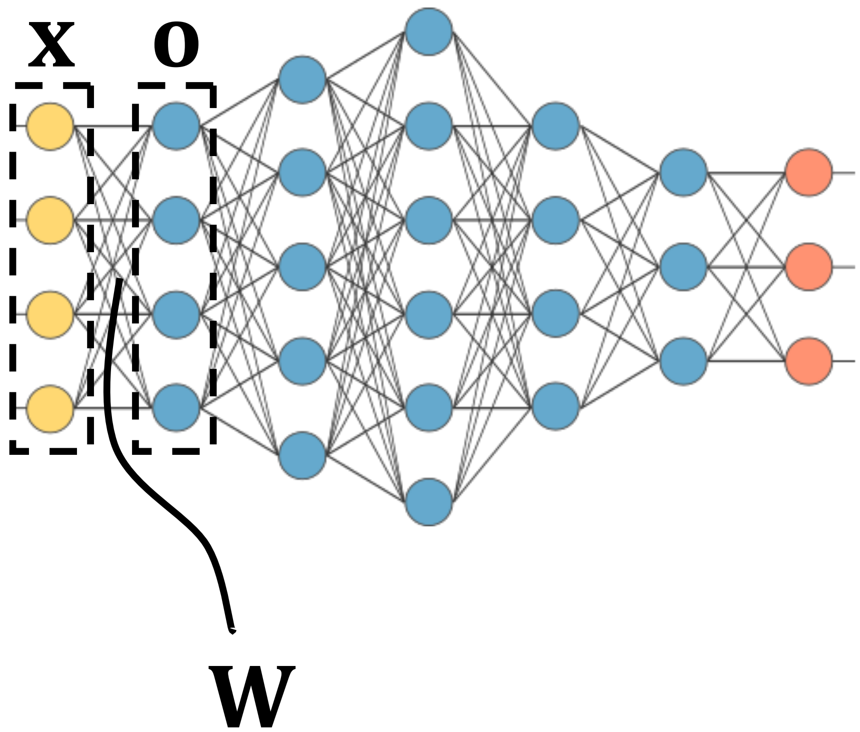
Neural Network

- Layers of interconnected neurons
- Universal function approximator
- Can approximate any function



Neural Networks cont. Example

- Describe W and $\vec{F}(\cdot)$
- Write the output of one layer of a neural network using $W, \vec{x}$



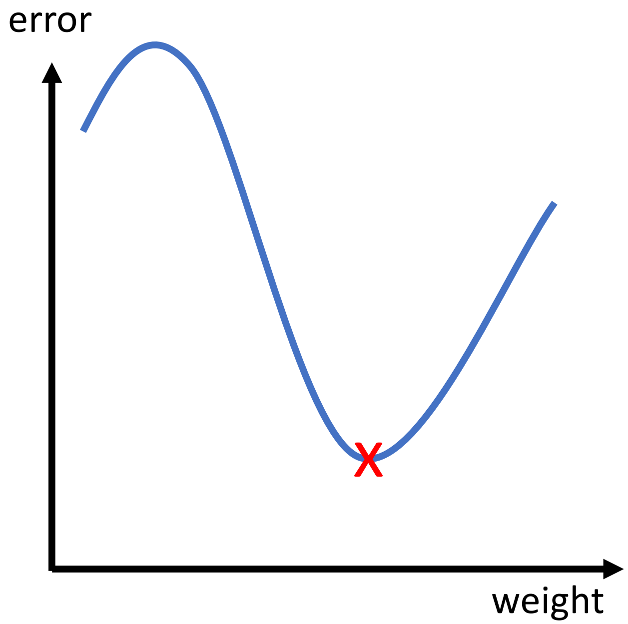
Gradient Descent

- Minimize the error:

$$e(w) = \sum_{i=1}^{N} \left(y_i - y(w)\right)^2$$

- Use gradient descent for each weight:

$$w_j \leftarrow w_j - \eta \cdot \frac{d\, e(w_j)}{dw_j}$$

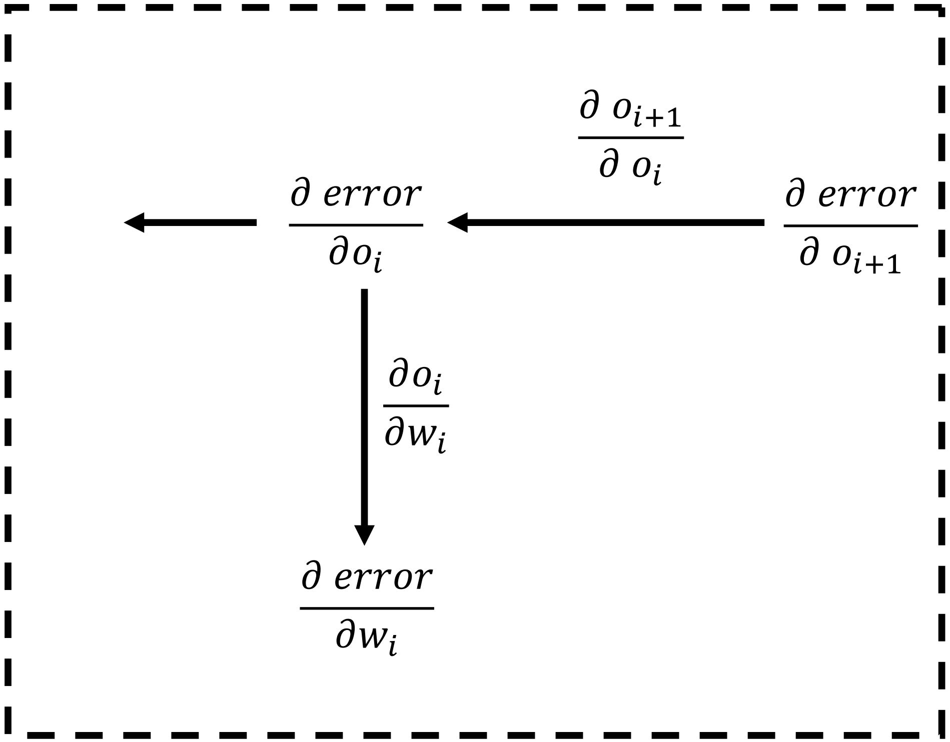


Back propagation

$$\frac{\partial\, error}{\partial w_i} = \underbrace{\frac{\partial\, error}{\partial o_i}}_{\frac{\partial\, o_{i+1}}{\partial\, o_i} \cdot \frac{\partial\, error}{\partial\, o_{i+1}}} \cdot \frac{\partial o_i}{\partial w_i}$$

- Method to use gradient descent in neural network efficiently
- Use chain rule of the gradient
- How it is done:

1. Do a forward pass: Check what output is being produced
2. Compare the output with the target (check the error)
3. Check how every weight is responsible for the produced error: Gradient
4. Use the chain rule to propagate the error back through the network and adjust the weights accordingly (learning rate η)



Back propagation cont. Example

- Given:
  - 2D input, 1 neuron on hidden layer, RELU activation; 1 neuron on readout, Linear activation
- Task:
  - Draw the system
  - Compute the gradients
  - Compute one weight update starting with $w_1 = w_2 = w_3 = 1$ and $x_1 = x_2 = 2$ and $\eta = 0.1$ and $y_{target} = 2$ of $w_1 \leftarrow w_1 \; - \eta \cdot \frac{d\, e(w_1)}{dw_1}$.

Exercises

Exercise 1

- Task:
  - What operations does the neural network perform on a forward pass?
  - What operation can be performed on a backward pass?
  - Does a backward pass require a forward pass for optimization?

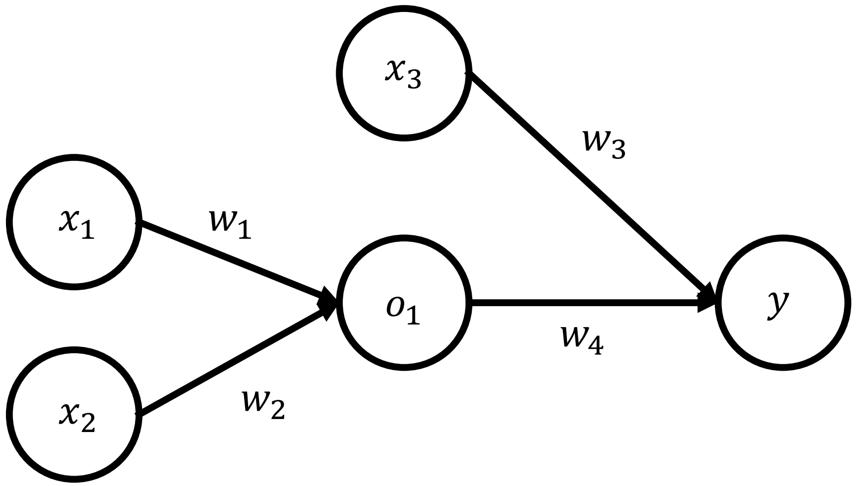
Exercise 2

- Task:
  - Will we find a global minimum with Gradient descent?
  - What does the derivative intuitively mean?
  - What’s the derivative in higher dimensions?

Exercise 3

- o_1 uses a $f(x) = \max(x, 0)$ (called ReLU) activation function, y a linear one

- The neural network on the right can be written as layers: $o_1 = f(\vec{w}_1^T \, \vec{x})$and $y = \vec{w}_2^T \vec{o}$. What are the entries in $\vec{w}_1, \vec{w}_2, \vec{x}$, and $\vec{o}$?
- Define the error as $(y - \tilde{y})^2$. Compute $\frac{d\ e}{dw_1}$.
- Assume $w_1 = w_2 = w_3 = w_4 = 1$ and $x_1 = x_2 = x_3 = 1$ and $\eta = 0.1$ and $y_{target} = 2$. Compute one gradient update step as $w_1 \leftarrow w_1 \ - \eta \cdot \frac{d\ error(w_1)}{dw_1}$.
- Compute the gradient of the error ∇e and all relevant terms for the back propagation.



Exercise 4

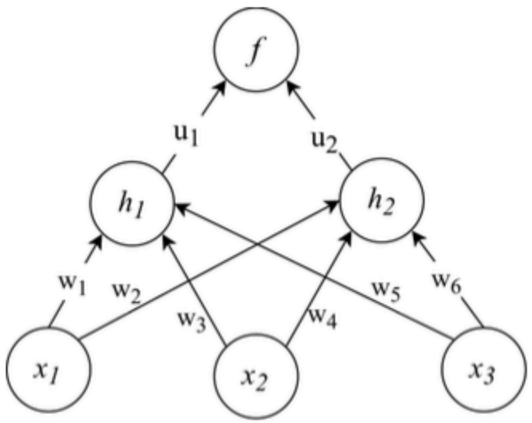
- Task:
  - Draw the following network:
    1. Input with 3 entries
    2. Hidden layer with 4 neurons and ReLU activation
    3. Second hidden layer with 2 neurons and ReLU activation
    4. Readout layer with 2 outputs and linear activation
  - Write down the matrices
  - Write down the gradient of the error $\nabla error$, (don't decompose the derivatives)

Homework

HW 1

Consider the following neural network with two logistic hidden units h_1, h_2, and three inputs x_1, x_2, x_3. The output neuron f is a linear unit, and we are using the squared error cost function $E = (y - f)^2$. The logistic function is defined as $\rho(x) = 1/(1 + e^{-x})$.

[*Note: You can solve part (c) without using the solution for part (b).*]



(a) Consider a single training example $\boldsymbol{x} = [x_1, x_2, x_3]$ with target output (label) y. Write down the sequence of calculations required to compute the squared error cost (called forward propagation).

(b) A way to reduce the number of parameters to avoid overfitting is to tie certain weights together, so that they share a parameter. Suppose we decide to tie the weights w_1 and w_4, so that $w_1 = w_4 = w_{\text{tied}}$. What is the derivative of the error E with respect to w_{tied}, i.e. $\nabla_{w_{\text{tied}}} E$?