ETH zürich

Exercise 2: Neural Networks

MAD
bacdavid@student.ethz.ch

Outline

1. Information
2. Theory
3. Example
4. Exercises
5. Coding Example
6. Questions

Information

Submissions & Questions

- Check LAB slides!
- Use google & think first!

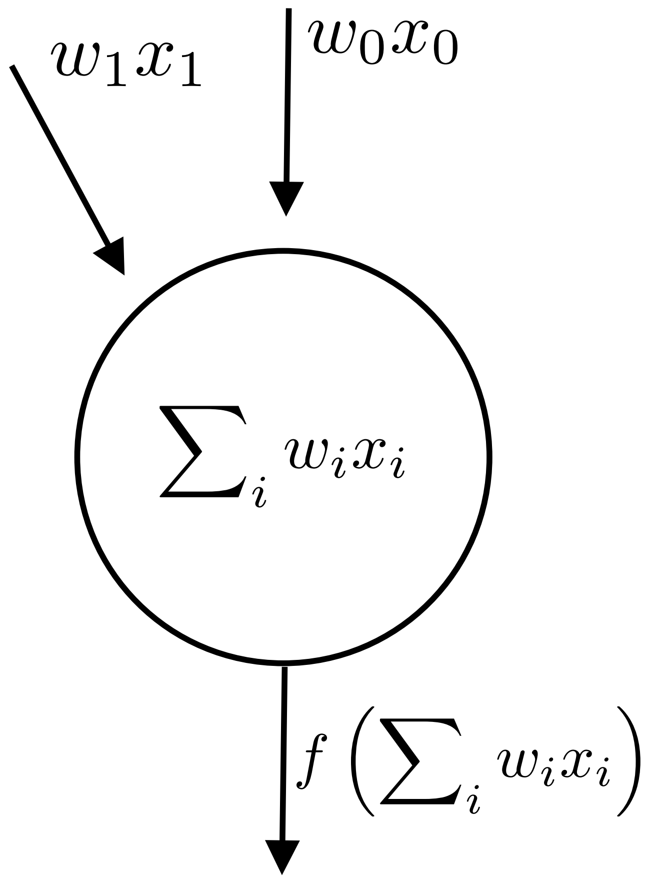
Theory

Neurons

1. Each input x_i is weighted by w_i
2. Then summed
3. Then an activation function f is applied
4. $f(\sum w_i x_i)$ constitutes the input for the next layer

Note

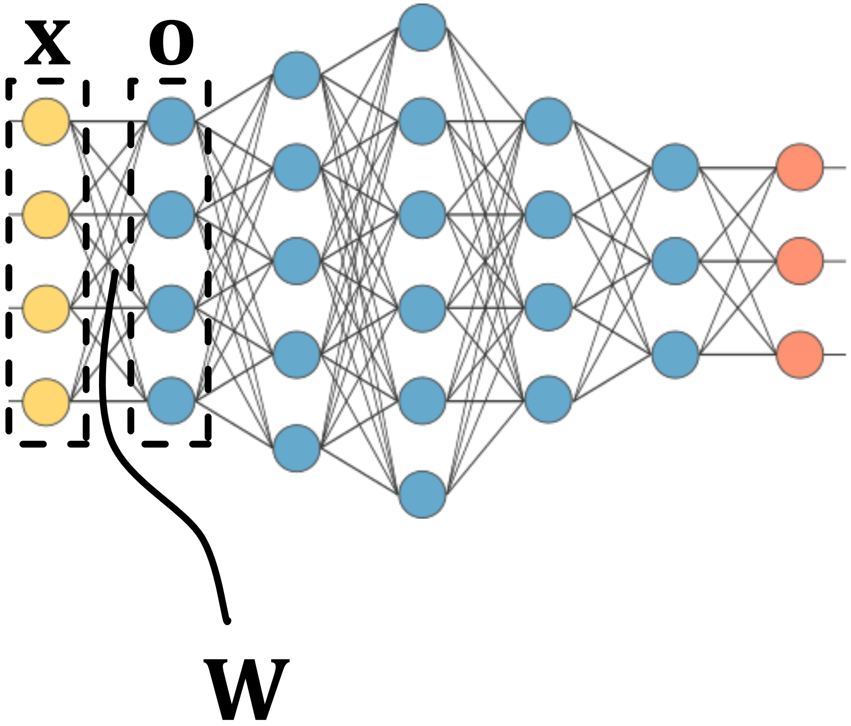
- $\sum w_i x_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$ “dot product”
- Usually the is a bias “offset” term b_i: $\sum w_i x_i + b_i$
- Notation is different in the exercises



Neural Networks

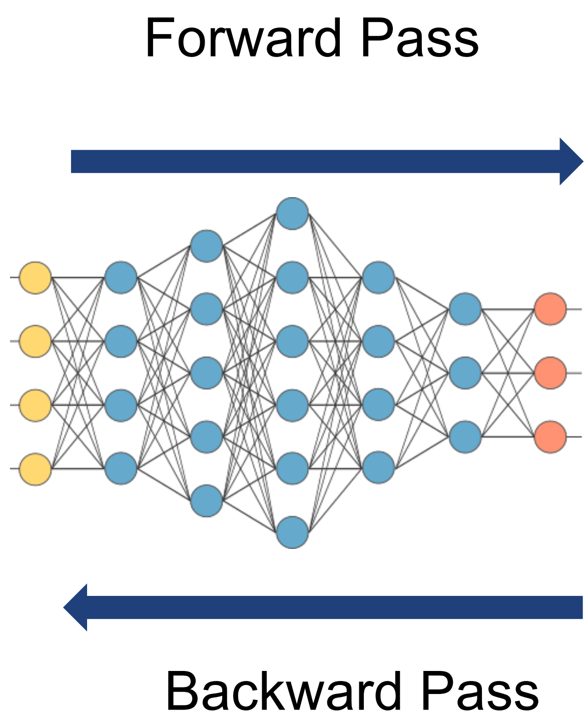
- Universal function approximator
- Can approximate any function
- Layers of interconnected neurons

- Using lin. alg. notation:
  1. $f\left(\mathbf{w}_1^{\mathrm{T}}\mathbf{x}\right) = \mathrm{o}_1$ & $f\left(\mathbf{w}_2^{\mathrm{T}}\mathbf{x}\right) = \mathrm{o}_2$ etc.
  2. Stack: $\hat{f}(\mathbf{W}\mathbf{x}) = \mathbf{o}$



Operations on Neural Network

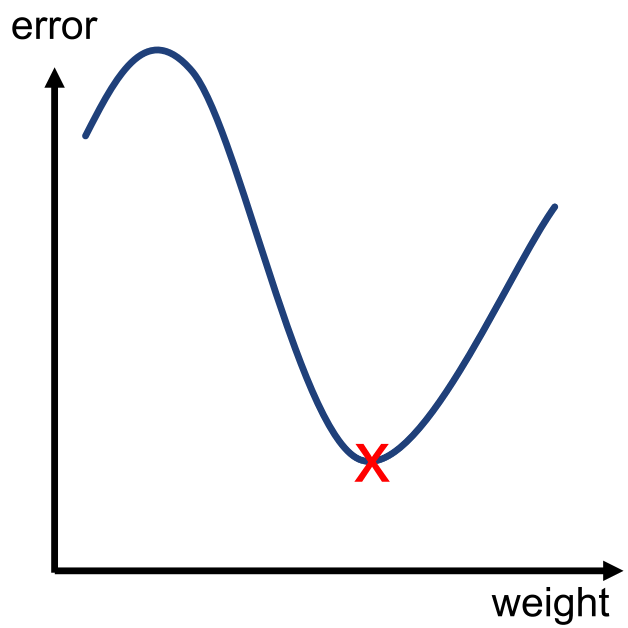
- Forward pass: read out $\mathbf{y} = \mathbf{F}^{\mathrm{NN}}(\mathbf{x})$
- Backward pass: gradient descent



Gradient Descent

- Gradient Descent is used to adjust the weights w_i as such that they produce the output we want
- What do we want? Minimize $error = \left(y_{target} - y(w)\right)^2$
- Update:

$$w \leftarrow w - \eta \cdot \frac{d\ error(w)}{dw}$$



Questions

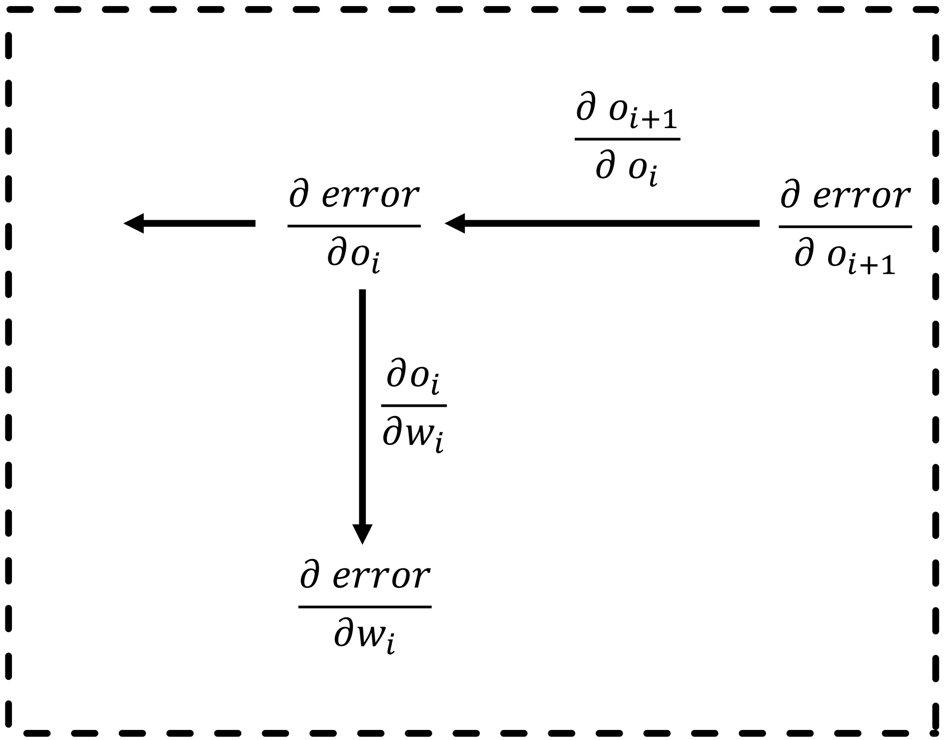
- Will we find a global minimum?
- What does the derivative intuitively mean?
- What's the derivative in higher dimensions?

Back Propagation

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

- Very efficient way to calculate the gradient for G.D.
- How does it work?

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)



Example

Exercises

Exercise 1

- Different notation used:

  x_i: $Input$

  w_{ij}: $Weight\ connecting\ neuron\ i\ and\ j$

  z_j^k: $Output\ of\ neuron\ j\ in\ layer\ k$

  $o_j^k = \sigma(z_j^k)$: $Activated\ output\ of\ neuron\ j\ in\ layer\ k$

  $\sigma(.)$: $Activation\ function$

  h_{k-1}: $Number\ of\ neurons\ in\ the\ layer\ k-1$

  b_j^k: $Bias\ of\ neuron\ j\ in\ layer\ k\ (can\ be\ neglected\ for\ simplicity)$
- Derive some of the aspects of back propagation: plug in & check what drops out

  ie. $\frac{\partial z_j^k}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}^k}\left(\sum_{i'} w_{i'j}^k o_{i'}^{k-1}\right) = o_i^{k-1}$ (ex. 1a)

Exercise 2

- Simple introduction to tensorflow
- Tensorflow is different than what you are used to terms programming:
    1. Set up a graph with all the functions that you will use afterwards
    2. Feed values into the graph; Values «travel» through graph and are returned

- Programming example follows

Coding Example

ETH zürich

Questions?