Introduction to Neural Networks and Uses in EDM

http://gdac.uqam.ca/Tutorial@EDM23/

Agathe Merceron

Berlin: about 12 flight hours from Bengaluru!

- A nice place to live!

http://www.iheartberlin.de/berlin-is-on-ice-impressions-of-a-frozen-city/

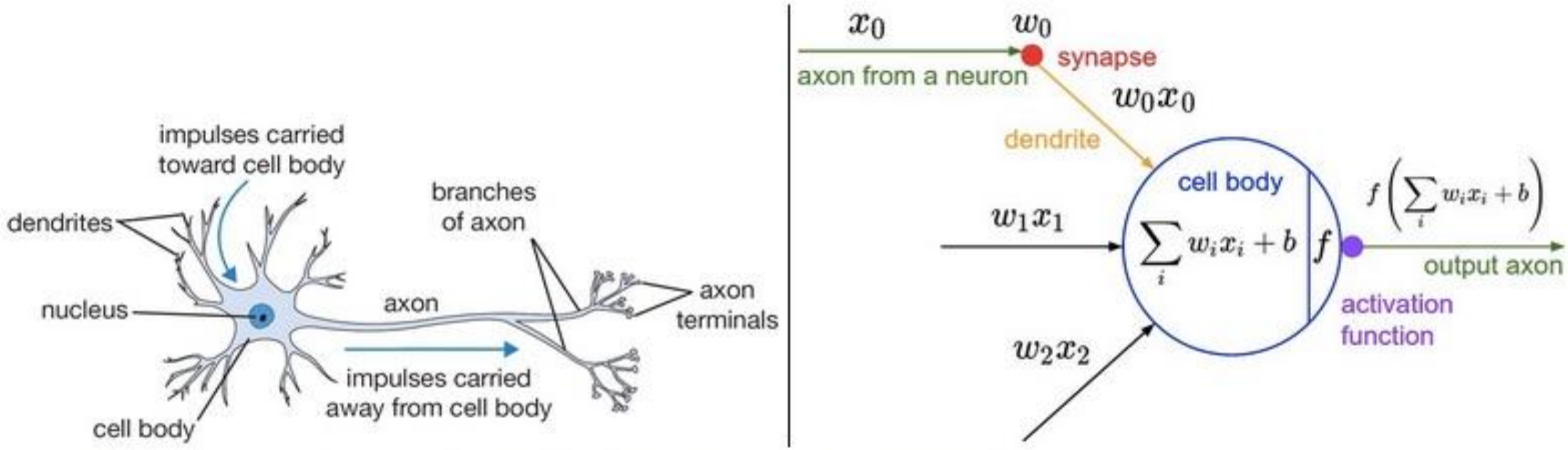
http://awesomeberlin.net/wp-content/uploads/2017/06/wann1.jpg

Outline

- Artificial neuron
- Activation function
- Feedforward neural networks
- Forward calculation
- Loss function
- Backpropagation
- Use in EDM
  - Application to predict dropout

Neuron

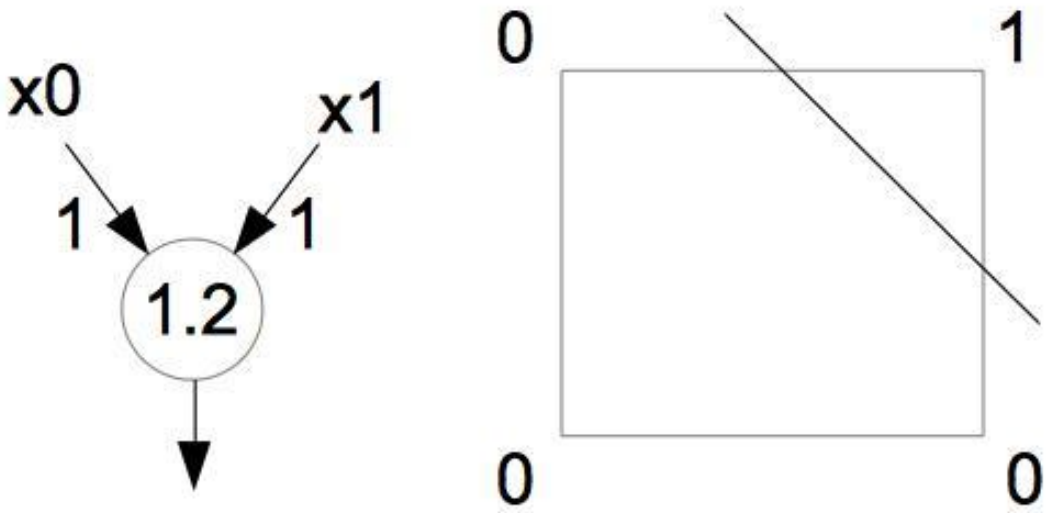
- http://cs231n.github.io/neural-networks-1/



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

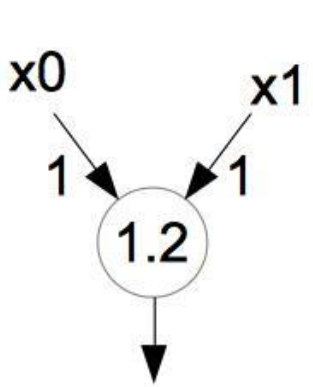
Neural networks and Boolean operators

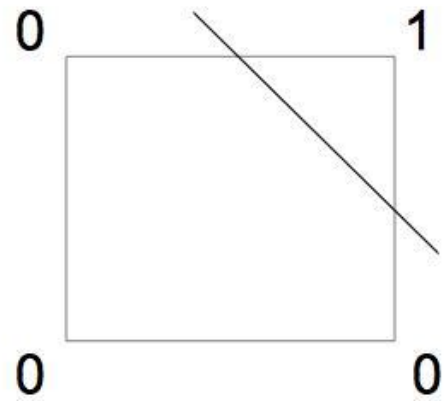
- The operator AND can be represented by a single neuron
- Activation function: Heaviside function: 0 if the weighted sum is smaller than the number in the neuron (chosen in advance, here 1.2), 1 otherwise



Neural networks and Boolean operators

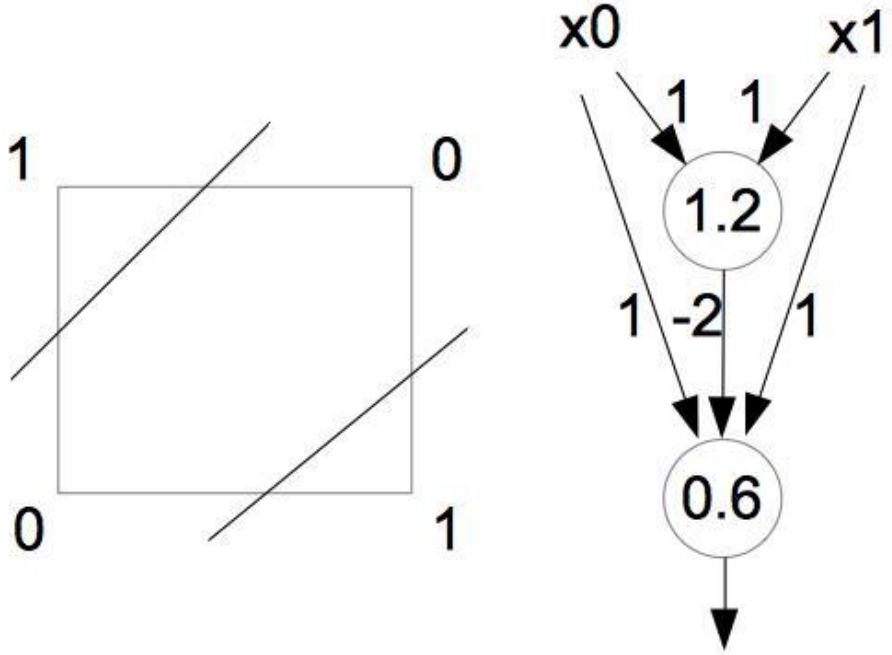
x0	x1	AND	Output
0	0	1*0+1*0 < 1.2	0
0	1	1*0+1*1 < 1.2	0
1	0	1*1+1*0 < 1.2	0
1	1	1*1+1*1 ≥ 1.2	1





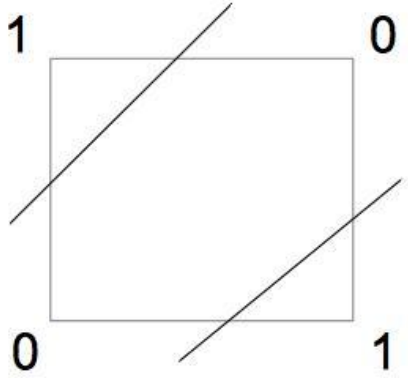
Neural networks and Boolean operators

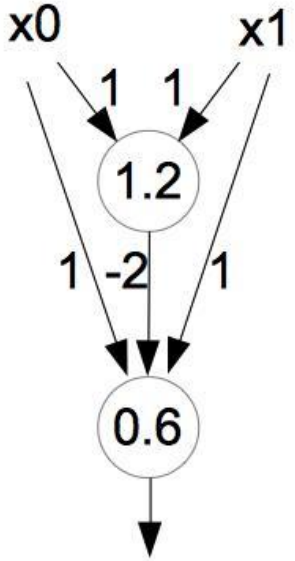
- The operator XOR cannot be represented by a single neuron: A second neuron is needed
- Activation function: Heaviside function: 0 if the weighted sum is smaller as the number in the neu



Neural networks and Boolean operators

x0	x1	AND			Output
0	0	1*0+1*0 < 1.2	0	1*0+-2*0+1*0 < 0.6	0
0	1	1*0+1*1 < 1.2	0	1*0+-2*0+1*1 ≥ 0.6	1
1	0	1*1+1*0 < 1.2	0	1*1+-2*0+1*0 ≥ 0.6	1
1	1	1*1+1*1 ≥ 1.2	1	1*1+-2*1+1*1 < 0.6	0



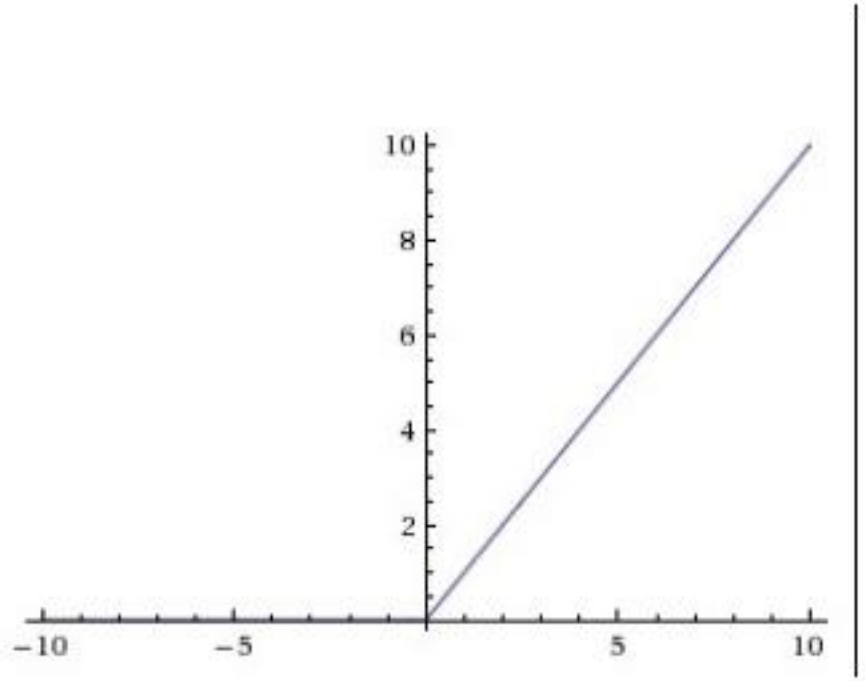


Activation functions

- Heaviside function: 1 if weighted sum of the inputs bigger than the threshold in the neuron, 0 otherwise.
- Rectified Linear Units (ReLU): $f(x) = \begin{Bmatrix} 0, if\ x \leq 0 \\ x, if\ x > 0 \end{Bmatrix}$
- Logistic sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$
- tanh: $f(x) = \frac{e^{x} - e^{-x}}{e^{x}+e^{-x}}$

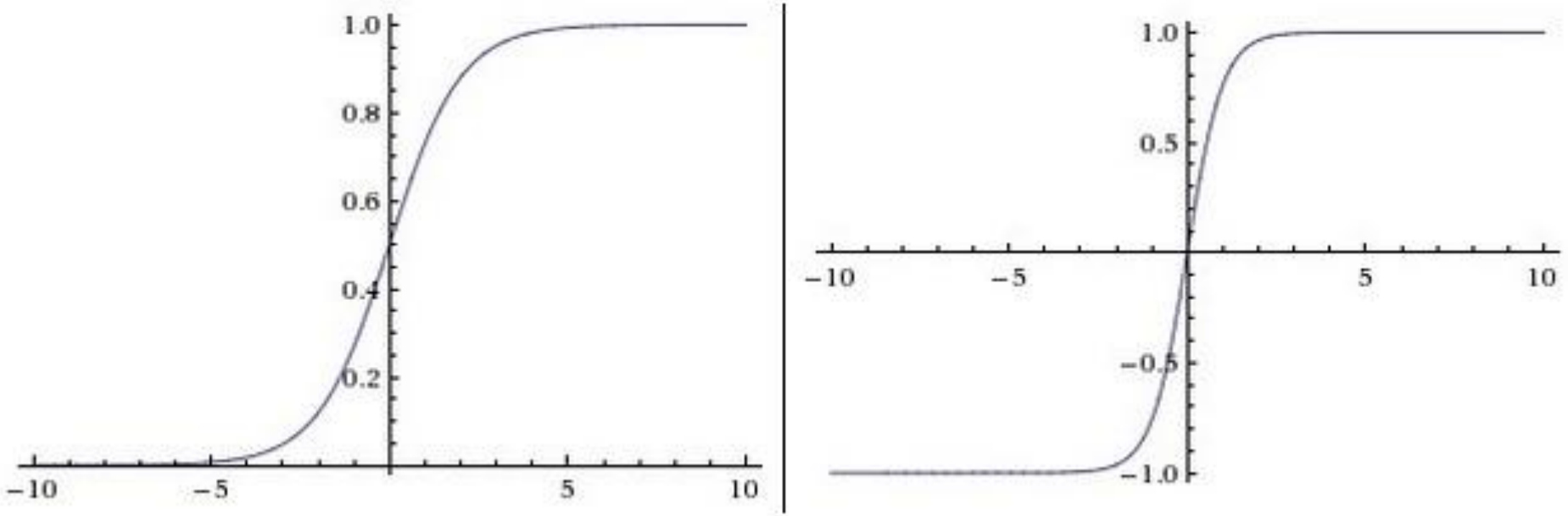
Activation functions

- Rectified Linear Units (ReLu):



https://cs231n.github.io/neural-networks-1/#classifier

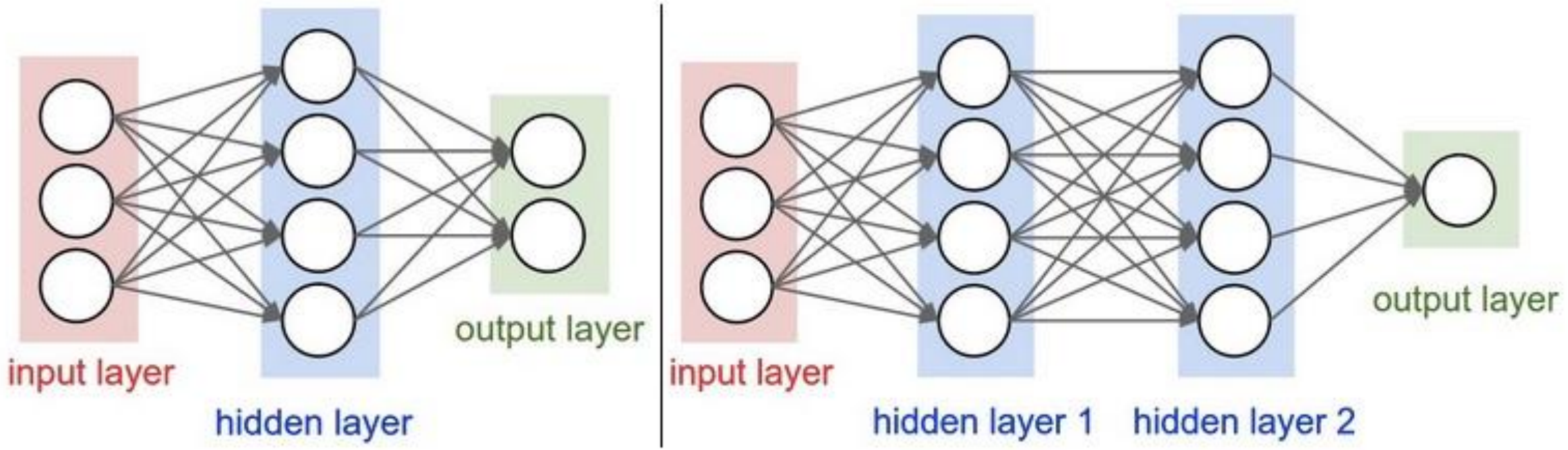
Activation functions: squashing functions



Left: Sigmoid non-linearity squashes real numbers to range between [0,1] **Right:** The tanh non-linearity squashes real numbers to range between [-1,1].

https://cs231n.github.io/neural-networks-1/#classifier

Feedforward neural networks

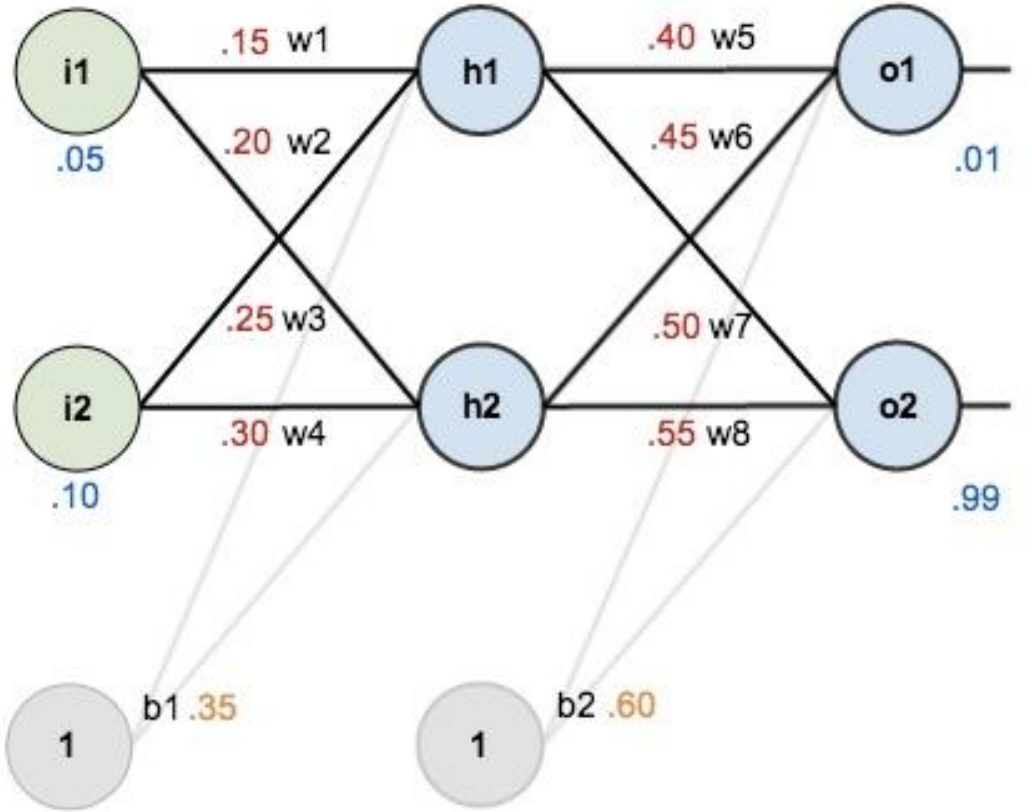


Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs.
Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

https://cs231n.github.io/neural-networks-1/#classifier

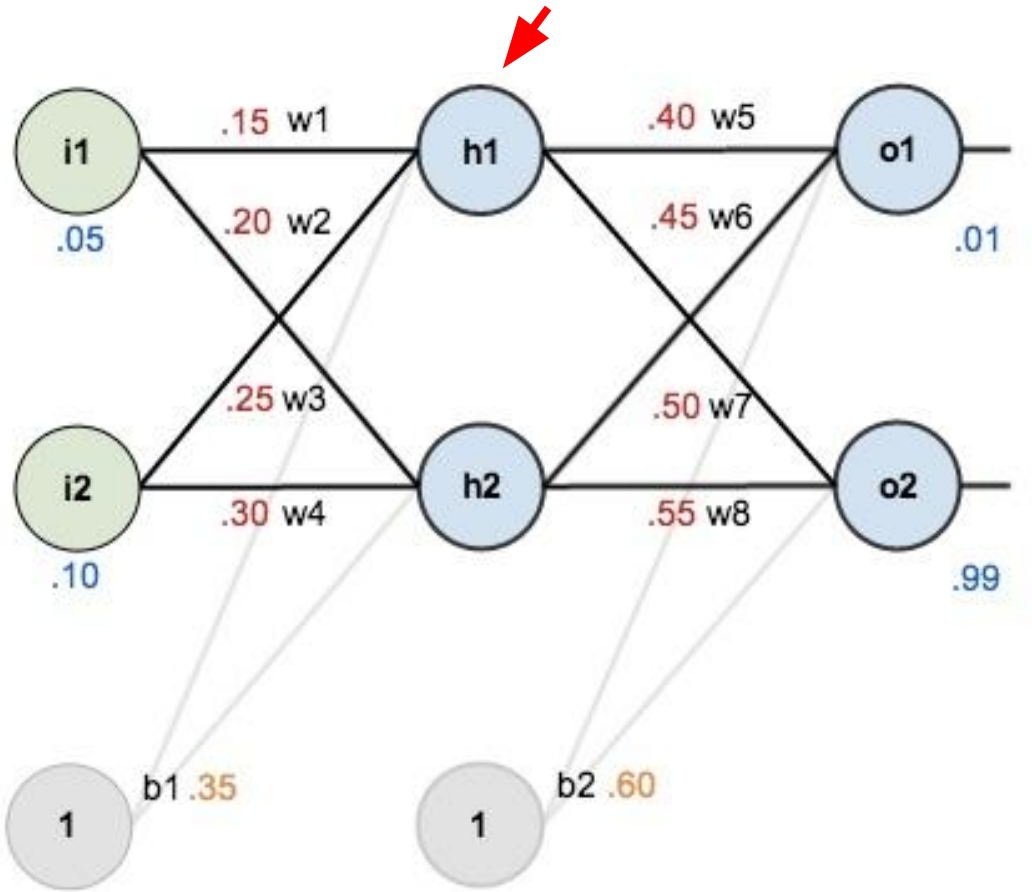
Hands-on: forward calculation

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/ note the bias (similar to the intercept b in the linear function ax+b)



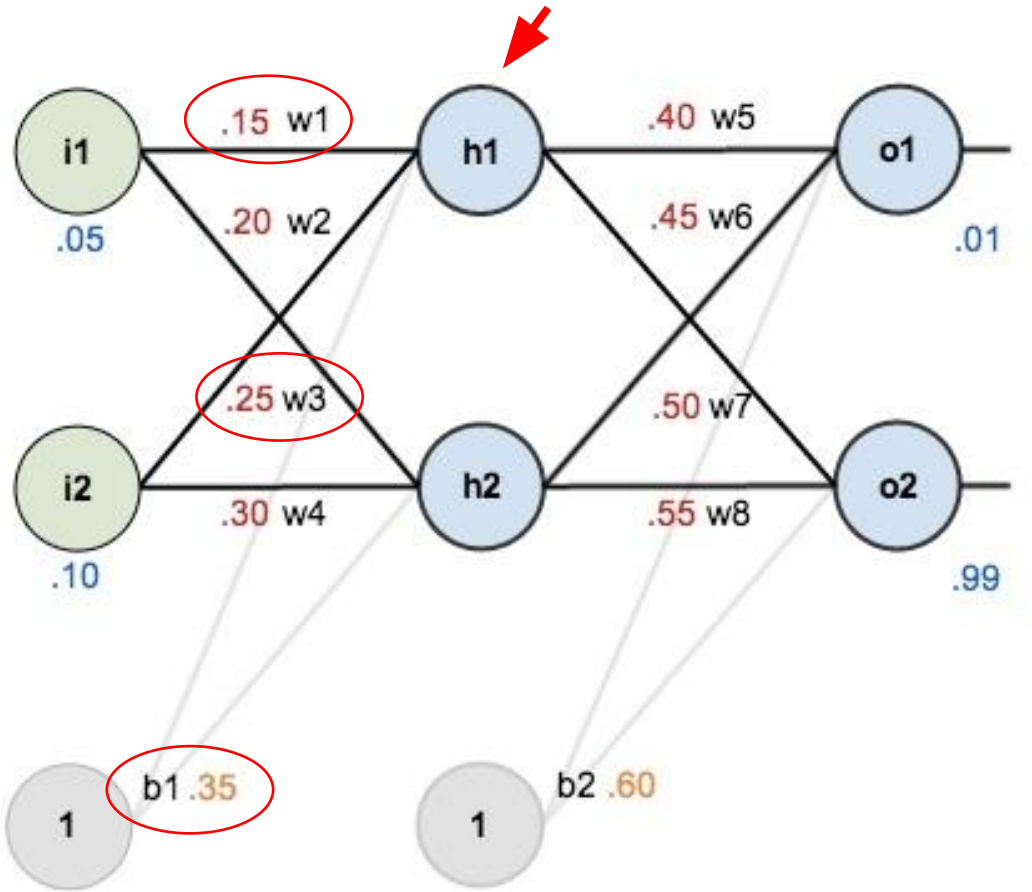
Hands-on: forward calculation 1

- Calculate the output of neuron h1 for the inputs (0.05, 0.1) and the sigmoid function f(x) $= \frac{1}{1+e^{-x}}$



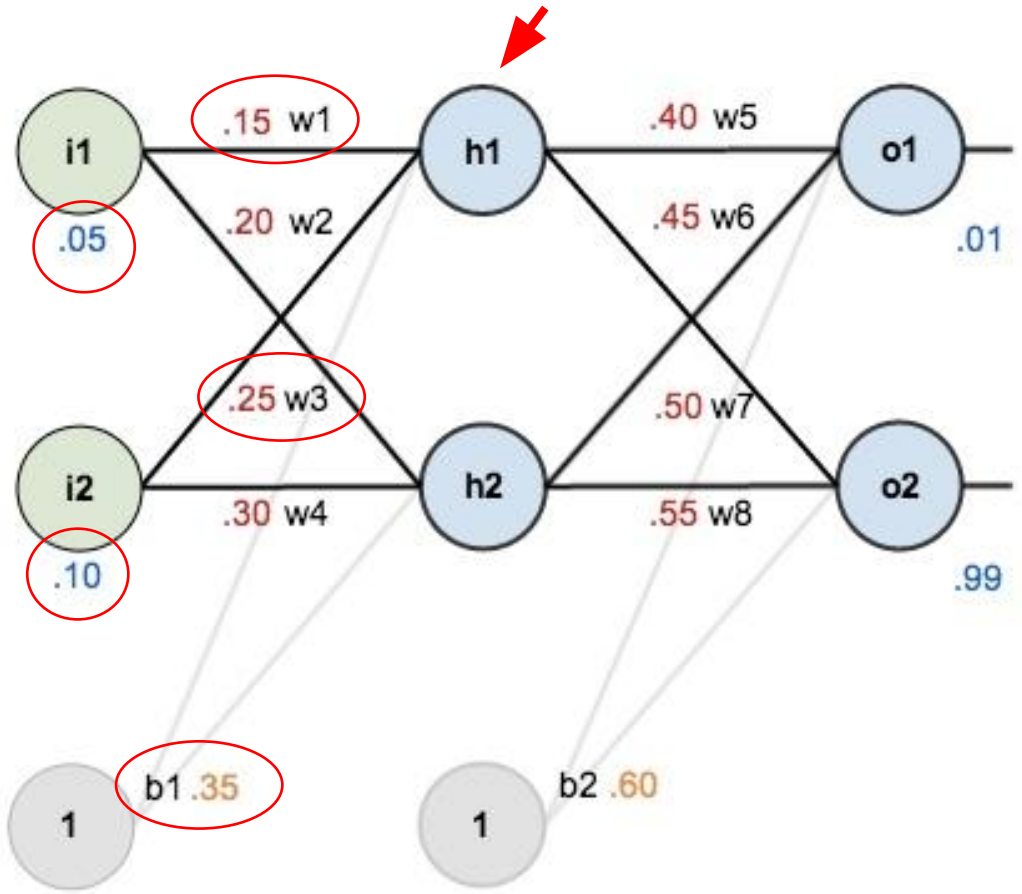
Hands-on: forward calculation 1

- Calculate the output of neuron h1 for the inputs (0.05, 0.1) and the sigmoid function f(x) $= \frac{1}{1+e^{-x}}$



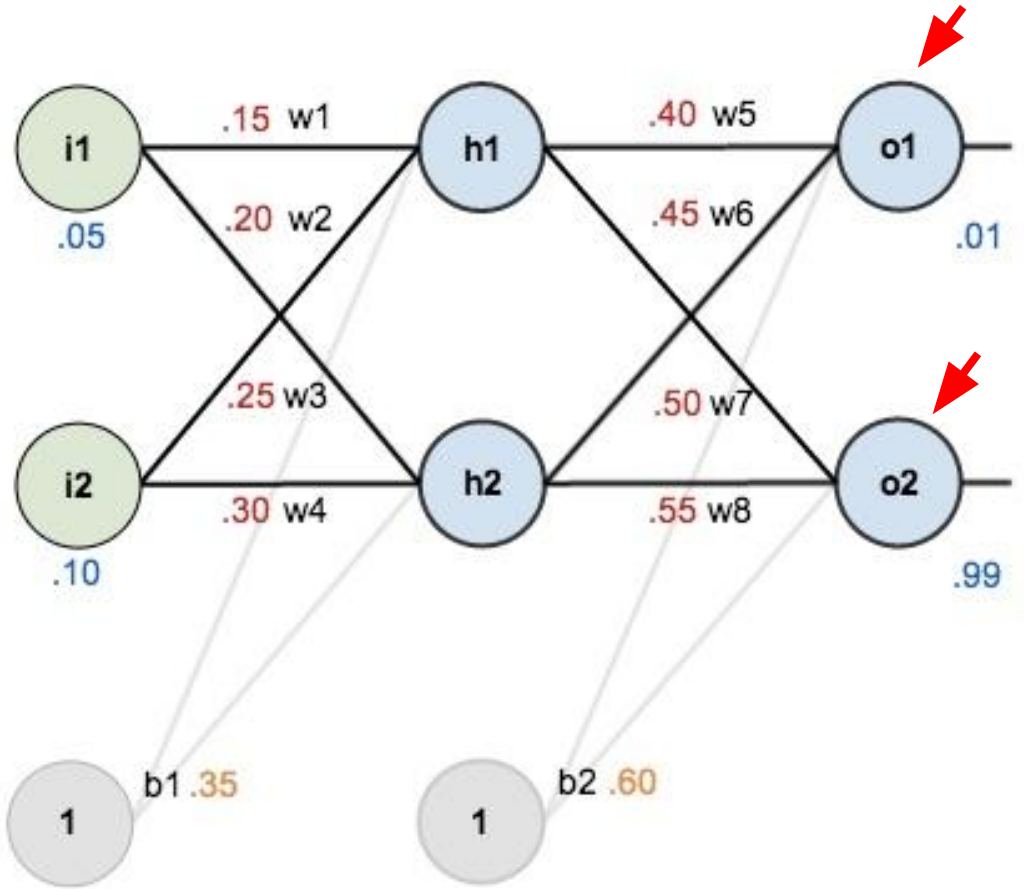
Hands-on: forward calculation 1

- Input h1 = 0.05*0.15 + 0.10*0.25 + 0.35 = 0.3775
- $f(x) = \frac{1}{1+e^{-0.3775}} = 0.5932$



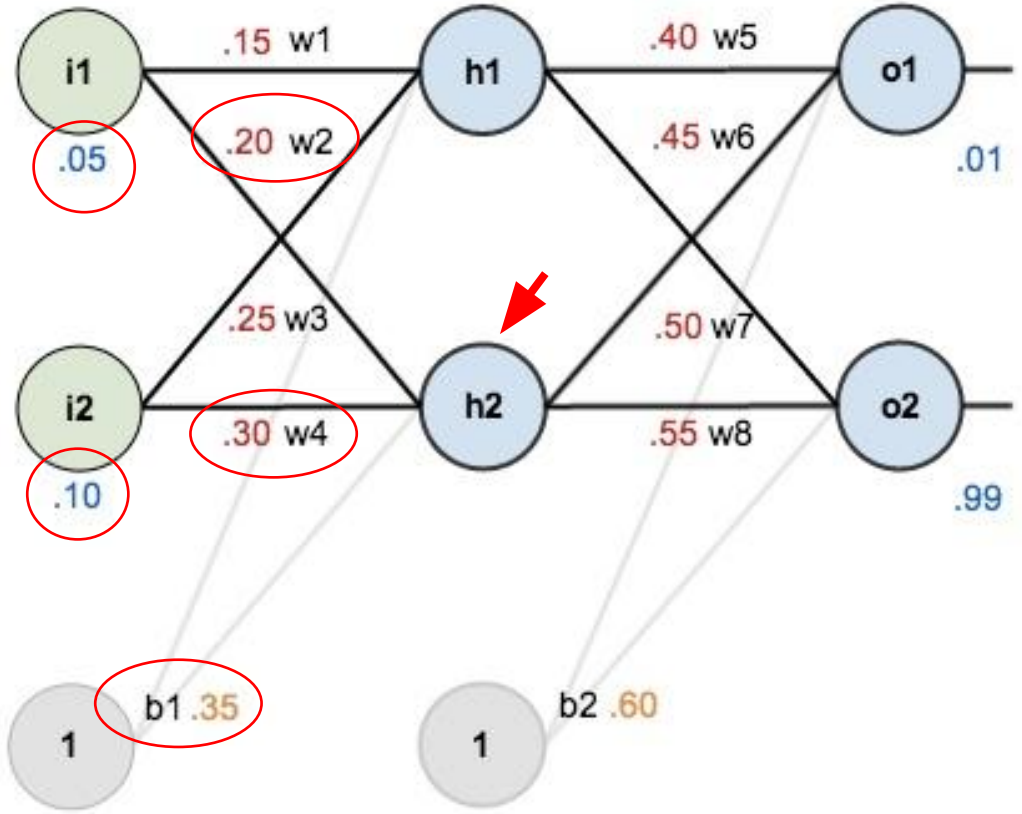
Hands-on: forward calculation 2

- Calculate the output of neurons o1 and o2 for the inputs (0.05, 0.1) and the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$



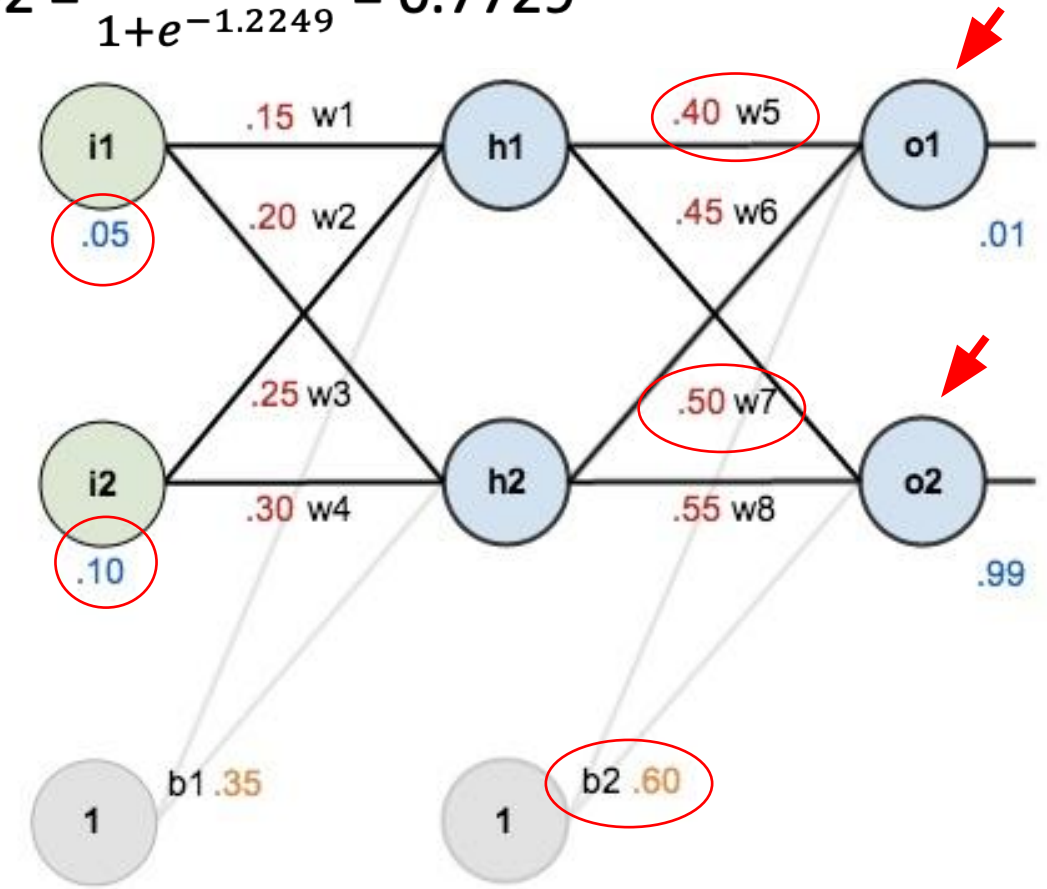
Hands-on: forward calculation 2

- Input h2 = 0.05*0.20 + 0.10*0.30 + 0.35 = 0.3925
- $f(x) = \frac{1}{1+e^{-0.3925}} = 0.5968$



Hands-on: forward calculation 2

- Input o1 = 0.5932*0.40 + 0.5968*0.50 + 0.60 = 1.1059
- Out o1 = $\frac{1}{1+e^{-1.1059}}$ = 0.7514, Out o2 = $\frac{1}{1+e^{-1.2249}}$ = 0.7729



Universal approximation theorem

“a feedforward network with a linear output layer and at least one hidden layer with any “squashing” activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units.... A neural network may also approximate any function mapping from any finite dimensional discrete space to another.“

Deep Learning; Ian Goodfellow, Yoshua Bengio, Aaaron Courville; MIT Press; 2016. P. 198

Feedforward neural network

- Structure must be chosen:
  - Number of inputs, of hidden layers, of neurons per hidden layers, activation function, output function, loss function etc. : the hyperparameters
  - Training costly (also in energy)

- In the training, the weights and biases are learned (stochastic gradient descent, backpropagation algorithm)

Training loop [Cholet p. 49]

- Draw a batch of training samples x with class T
- Run the network on x to obtain output O
- Compute the loss of the network, i.e. mismatch between O and T
- Compute the gradient of the loss
- Update the weights and biases
- Repeat till termination condition: the errors do not change or the loss is small enough

Hands-on: compute the loss (Mean Squared Error)

$$\text{Loss} = \sum_{i=1}^{2} \frac{1}{2}(T_i - O_i)^2$$

$$T_1 = 0.01,\ T_2 = 0.99,\ O_1 = 0.7514,\ O_2 = 0.7729$$

$$\text{Loss} = \frac{1}{2}(0.01 - 0.7514)^2 + \frac{1}{2}(0.99 - 0.7729)^2$$

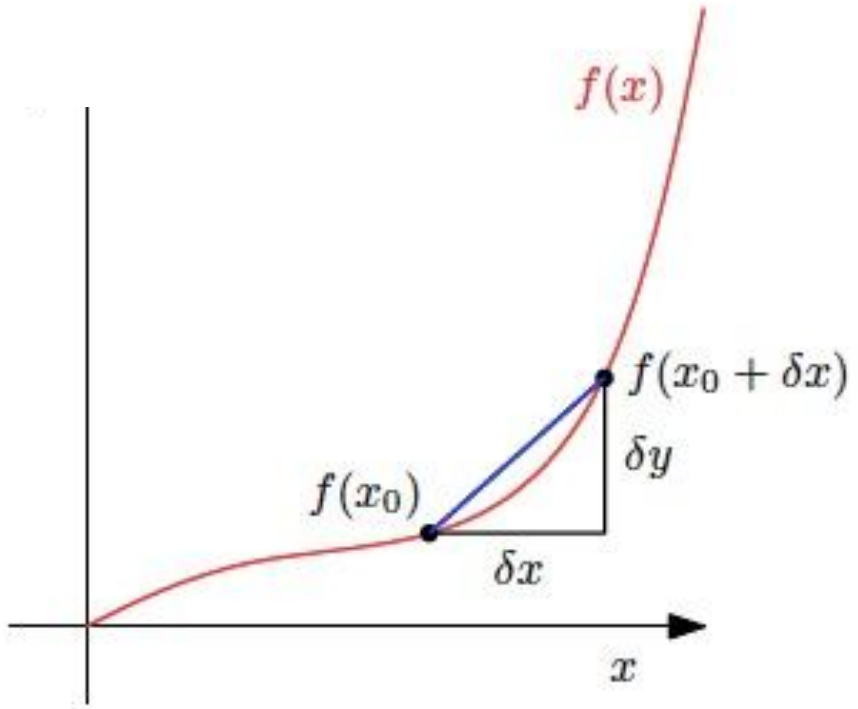
$$= 0.2984$$

Gradient of the loss: why?

- If the loss is not 0, how do we know whether we should increase or decrease a weight?
- We need to know whether our overall function is ascending (weight should be decreased) or descending (weight should be increased)
- For a simple function f: $R \rightarrow R$, the derivative gives this information
- For a complex function f: $R^n \rightarrow R^m$, the gradient gives this information

Gradient of the loss: why?

- The derivative below shows that f increases at x_0: $\frac{df}{dx} = \lim_{\partial x \to 0} \frac{f(x_0 + \partial x)}{\partial x}$



Mathematics of Machine Learning p.1

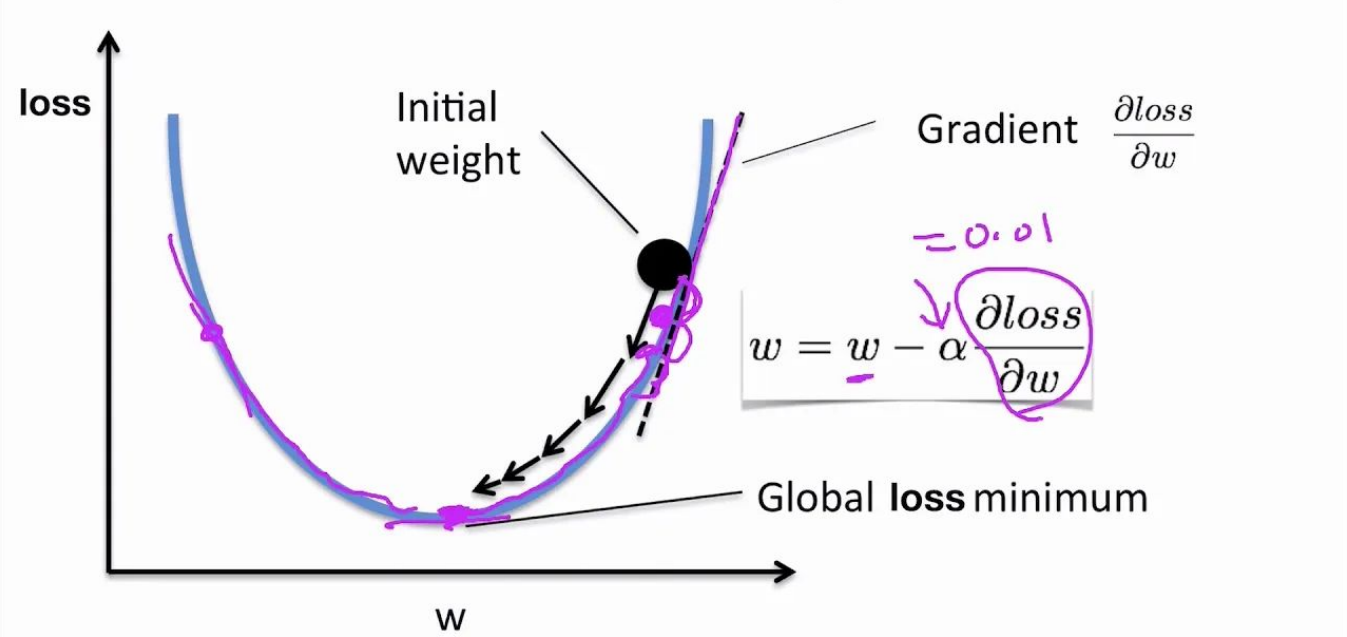
Gradient of the loss: why?

- For a complex function f: $R^n \rightarrow R^m$, the gradient of the loss $\nabla\, loss$ gives this information.

- New weights: $w' = w\ - l\, \nabla Loss$, where l is the learning rate.

Backpropagation

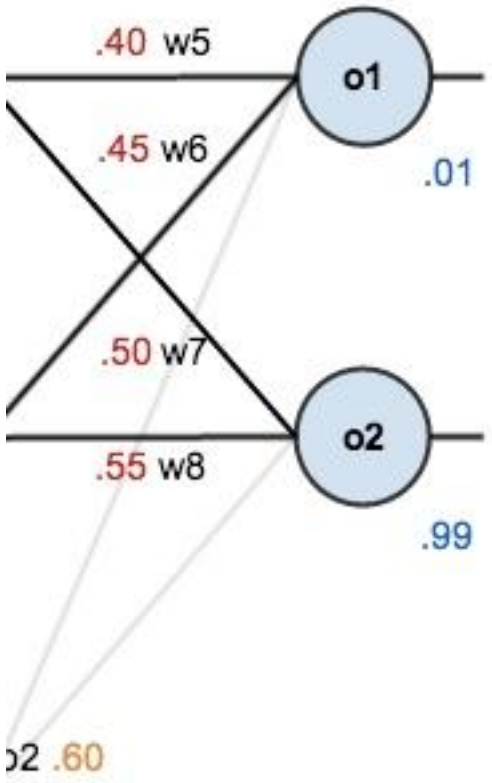
- Uses partial derivatives and the chain rule to calculate the change for each weight and bias efficiently
- Starts with the derivative of the loss function and propagates the calculations backward



https://medium.com/nerd-for-tech/linear-regression-from-scratch-pt2-the-gradient-descent-algorithm-130d421ea40c

Hands-On - Backpropagation

- How much a change in w_5 affects the loss? To know it, calculate the partial derivative of the loss with respect to w5. There is a chain of three functions:
- Loss = $\sum_{i=1}^{2} \frac{1}{2}(T_i - O_i)^2$
- $O_i = \frac{1}{1+e^{-Input\_i}}$
- $\text{Input\_i} = \sum_{k=1}^{2} wk * ik + \text{b2}$



Hands-On: Backpropagation

- Partial derivatives with respect to $w5$:
- $\text{Loss} = \frac{1}{2}(T1 - O1)^2 + \frac{1}{2}(T2 - O2)^2$
- $O1 = \frac{1}{1+e^{-Input\_1}}$
- $Input\_1 = w5 * out\ h1 + w6 * out\ h2 + b2$

- $\frac{\partial Loss}{\partial w5} = \frac{\partial Loss}{\partial O1} * \frac{\partial O1}{\partial Input\_1} * \frac{\partial Input\_1}{\partial w5}$

Hands-On: Backpropagation

- Loss $= \frac{1}{2}(T1 - O1)^2 + \frac{1}{2}(T2 - O2)^2$
- $\frac{\partial Loss}{\partial O1} = \frac{1}{2} * 2(T1 - O1) * -1 = -(T1 - O1) = 0.7414$

- T1 : 0.01 and $O1$: 0.7514

Hands-On: Backpropagation

- $O1 = \frac{1}{1+e^{-Input\_1}}$
- $\frac{\partial O1}{\partial Input\_1} = O1(1 - O1) = 0.7514\ (1 - 0.7514) = 0.1868$

Hands-On: Backpropagation

- $Input\_1 = w5 * out\ h1 + w6 * out\ h2 + b2$

- $\frac{\partial Input\_1}{\partial w5} = out\ h1 = 0.5932$

Hands-On: Backpropagation

- $\frac{\partial Loss}{\partial w5} = \frac{\partial Loss}{\partial O1} * \frac{\partial O1}{\partial Input_1} * \frac{\partial Input\_1}{\partial w5}$
- $\frac{\partial Loss}{\partial w5} = 0.7414 * 0.1816 * 0.5932 = 0.0821$
- $w5' = w5 - l * 0.0821 = 0.4 - 0.5 * 0.0821 = 0.3589$
- With 0.5 as learning rate.

Hands-On: Backpropagation

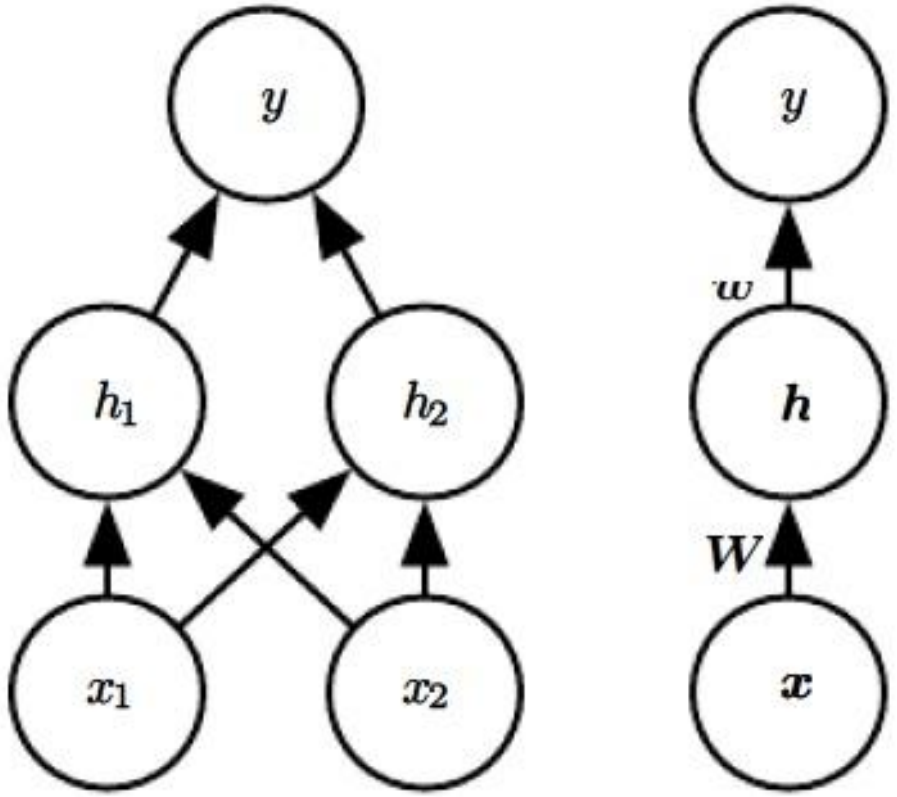
- $W'_5 = 0.3582$ (before 0.4)
- $O'_1 = 0.7467$ (before 0.7514)
- $Loss'_{o1} = 1/2(0.01-0.4467)^2 = 0.2714$
  - $Loss_{o1\ before}$: 0.275 > 0.2714

Training and Backpropagation

- Possible problem: weight decay

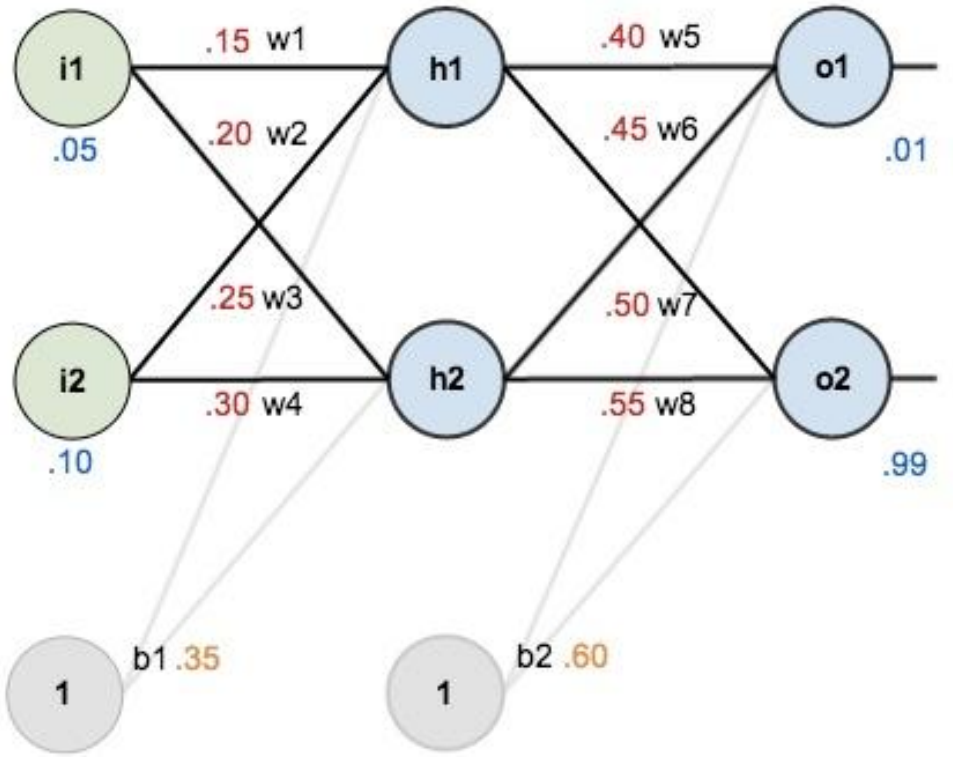
Feed Forward Neural Networks

- Compact graphical representation: W is the weights-matrix. Deep Learning; Ian Goodfellow, Yoshua Bengio, Aaaron Courville; MIT Press; 2016. P. 170



Feedforward Neural Networks

- Compact graphical representation: W is the weights-matrix. *It can include the bias.*
- *h = f(W x)* h: neurons in the hidden layer, x : input (1 for the bias), f: activation function.



$$W = \begin{matrix} 0.15 & 0.20 \\ 0.25 & 0.30 \\ 0.35 & 0.35 \end{matrix} \qquad x = \begin{matrix} 0.5 & 0.1 & 1 \end{matrix}$$

Another Loss Function: Binary Cross Entropy

- Value of the output 1 or 0 – binary classification
- Activation function of the last layer gives the **probability for the value 1**
- $L = -\frac{1}{N}\sum_{i=1}^{N} y_i \, Log\big(p(y_i)\big) + (1 - y_i) Log\big(1 - p(y_i)\big)$, N: Number of input objects.

0.751	1	-0.124	-0.604	1*-0.124 + (1-1)*-0.604	-0.124
0.773	0	-0.113	-0.644	0*-0.113 + (1-0)*-0.644	-0.644

$$L = -\frac{1}{2}(-0.124 + -0.644) = 0.384$$

Neural Networks and Deep Learning

- Well-known types of neural networks:
  - **Convolutional Neural Networks (CNN):** reduce full connectedness through the use of a convolutional operator.
  - **Long Short Term Memory (LSTM)**: recurrent topology

- Deep Learning: Hidden layers extract abstract features from the data.

Feed Forward NN in EDM

- Common application: Predicting dropout
  - Next hands-on.
- “Model the influence of co-taken courses on student’s grade on a course” *Z. Ren, X. Ning, A. Lan, and H. Rangwala, EDM 2019*
- “To automatically assess the design of a program and provide personalized feedback to guide students” *J. Walker Orr and Nathaniel Russell, EDM 2021*

Feed Forward NN to Predict Dropout in a Degree Program

- Wagner et al. *Which Approach Best Predicts Dropouts in Higher Education?* 2023
- Data from a medium size German university
- Three degree programs Architecture (AR), Computer Science and Media (CM), Print and Media Technology (PT); all six-semesters bachelor
- Data from 2012 till 2019:
  - Enrollment date in the degree program
  - Courses enrolled with marks and semester
  - Graduation date or exmatriculation date
  - Gender

Feed Forward NN to Predict Dropout in a Degree Program

- Data from 2012 till 2019:
  - Enrollment date in the degree program,
  - Courses enrolled with marks and semester,
  - Graduation date or exmatriculation date,
- LF - Local features: marks in courses; features are specific to a degree program:
  - One model per study program.
- **GF - Global features: features must be engineered** – mean mark, number of failed courses etc; features are common across degree programs:
  - **One global model for all study programs may be built**.

Feed Forward NN to Predict Dropout in a Degree Program

- Many calculated features. Feature selection to predict dropout at the end of the 1st semester:
  - P_Postponed: Proportion of the number of postponed courses
  - P_Not_Enrolled: Proportion of the number of courses without enrolment
  - Mean_O_Gr: Mean grade of all courses
  - MAD_O_Gr: Mean absolute deviation of all courses
- Data standardized

Feed Forward NN to Predict Dropout in a Degree Program

- Global feature set: Is the global model fair w.r.t. study programs?
  - Slicing Analysis, Gardner et al. 2019
    - Training set: 80% data from all three programs
    - Test separately for each study program
    - Test set: 20% of the newest data

Feed Forward NN to Predict Dropout in a Degree Program

- Results with five classifiers: AdaBoost (AB), Decision Trees (DT), K-Nearest Neighbors (KNN), Logistic Regression (LR), Random Forests (RF):

		BACC						REC						ACC					
	P	AR		CM		PT		AR		CM		PT		AR		CM		PT	
S	A	GF	LF	GF	LF	GF	LF	GF	LF	GF	LF	GF	LF	GF	LF	GF	LF	GF	LF
1	AB	0.804	0.879	0.938	0.935	0.846	0.815	0.844	0.838	0.877	0.870	0.859	0.797	0.828	0.854	0.884	0.878	0.855	0.803
	DT	0.879	0.889	0.932	0.910	0.846	0.831	0.838	0.857	0.864	0.821	0.859	0.828	0.854	0.870	0.872	0.831	0.855	0.829
	KNN	0.883	0.843	0.944	0.914	0.763	0.789	0.844	0.818	0.889	0.827	0.859	0.828	0.859	0.828	0.895	0.837	0.829	0.816
	LR	0.886	0.886	0.941	0.923	0.828	0.786	0.877	0.851	0.883	0.846	0.906	0.906	0.880	0.865	0.890	0.855	0.882	0.868
	RF	0.899	0.847	0.944	0.941	0.828	0.831	0.851	0.799	0.889	0.883	0.906	0.828	0.870	0.818	0.895	0.890	0.882	0.829

- What with Neural Networks? Let us try it out!

Feed Forward NN to Predict Dropout in a Degree Program

- See: https://colab.research.google.com/drive/1RrsN3ojUv_byjgmq-Zfwn3txltyTiZP8?usp=sharing

Feed Forward NN to Predict Dropout in a Degree Program

- NN also good at predicting dropout, differences between study programs
  - need some work to check better
- What do we do with this knowledge?
  - Recommender system specifically to support students at risk of dropping out in their enrollment: Wagner et al. *Can the Paths of Successful Students Help Other Students With Their Course Enrollments? EDM 2024.*

References

- F. Chollet. *Deep Learning with Python*. Manning, 2018
- M. P. Deisenroth, A. A. Faisal, C. S. Ong. *The Mathematics of Machine Learning*. https://mml-book.github.io/
- I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press; 2016
- J. Gardner, C. Brooks, and R. Baker. *Evaluating the fairness of predictive student models through slicing analysis*. In Proceedings of the 9th International Conference on Learning Analytics & Knowledge, pages 225–234, 2019.
- J. W. Orr and N. Russell. *Automatic assessment of the design quality of python programs with personalized feedback*. In I.-H. S. Hsiao, S. S. Sahebi, F. B. chet, and J.-J. Vie, editors, Proceedings of the 14th International Conference on Educational Data Mining (EDM 2021), pages 495–501. International Educational Data Mining Society, 2021.

References

- Z. Ren, X. Ning, A. Lan, and H. Rangwala. *Grade prediction based on cumulative knowledge and co-taken courses*. In M. Desmarais, C. F. Lynch, A. Merceron, and R. Nkambou, editors, Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019), pages 158–167. International Educational Data Mining Society, 2019.
- Wagner, K., Volkening, H. Basyigit, S., Merceron, A., Sauer, P., Pinkwart, N.: *Which Approach Best Predicts Dropouts in Higher Education?* In Proceedings of the 15th International Conference on Computer Supported Education (CSEDU), 2023. DOI: 10.5220/0011838100003470

References

- François Chollet. Deep Learning with Python. Manning 2018.
- Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. The Mathematics of Machine Learning. https://mml-book.github.io/
- Ian Goodfellow, Yoshua Bengio, Aaaron Courville. Deep Learning. MIT Press; 2016.

Thank you !

Observations ?

Questions ?

BHT

Berliner Hochschule für Technik

Studiere Zukunft