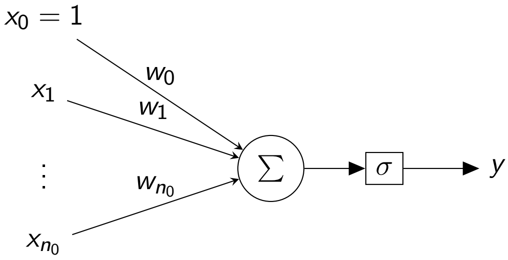
5 Neural Networks

Neural networks - one perceptron



- $z = \sum_{i=0}^{n_0} w_i x_i$
- $y = f(z)$
- f non-linear activation function, often sigmoid σ

Non-linear activation function

- enables non-linearity in NNs

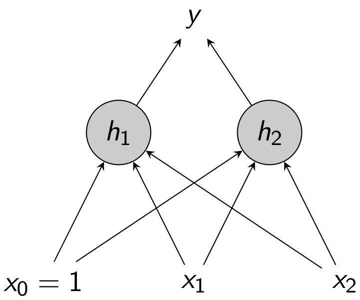
- sigmoid $\sigma(z) = 1/(1 + \exp -z)$
- tanh $= (\exp(z) - \exp -z)/(\exp(z) + \exp -z)$
- rectified linear unit $\mathrm{ReLU}(z) = \max\{0, z\}$
- $\ldots$

- $\mathrm{softmax}(z_i) = \exp(z_i)/\sum_{j=1}^{k} \exp(z_j), i = 1, \ldots, k$

XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function?

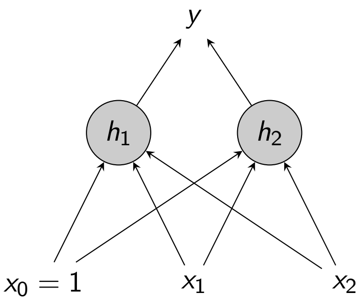
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

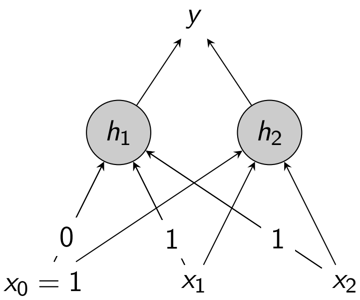
x_1	x_2	y	h_1	h_2
0	0	0	0	$-1 \to 0$
0	1	1	1	0
1	0	1	1	0
1	1	0	2	1



XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

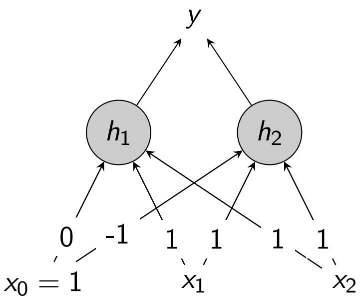
x_1	x_2	y	h_1	h_2
0	0	0	0	$-1 \to 0$
0	1	1	1	0
1	0	1	1	0
1	1	0	2	1



XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

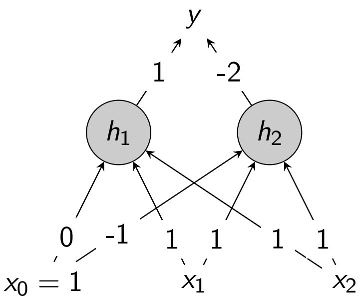
x_1	x_2	y	h_1	h_2
0	0	0	0	$-1 \rightarrow 0$
0	1	1	1	0
1	0	1	1	0
1	1	0	2	1



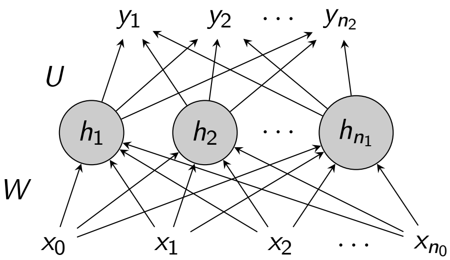
XOR problem

- why more than one layer networks?
- XOR = exclusive OR
- cannot be calculated by only one perceptron
- which activation function? ReLU

x_1	x_2	y	h_1	h_2
0	0	0	0	$-1 \rightarrow 0$
0	1	1	1	0
1	0	1	1	0
1	1	0	2	1

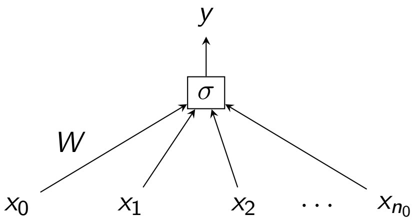


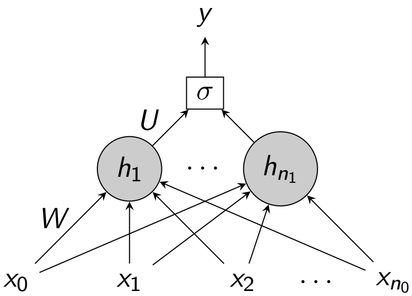
Feedforward (FF) neural networks



- for historic reasons also known as multi-layer perceptrons (MLP)
- diy: write CBOW, skip-gram, PV-DM and PV-DBOW as FF NN

Logistic regression as FF NN

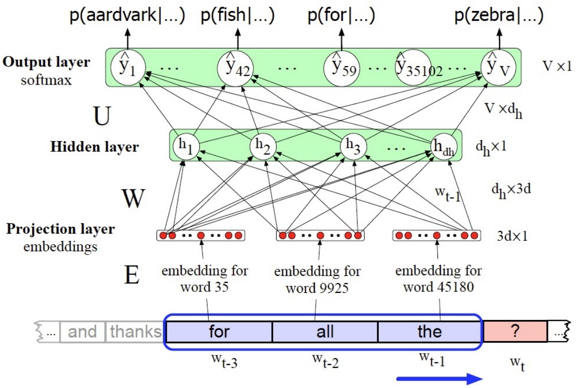




- logistic regression is a NN without hidden layer (left figure)
- adding a hidden layer (right figure) maybe improve performance
- hidden layer allows for non-linear interactions between features

- power of deep learning, e.g., NN, is to learn feature interaction
- no need of complicated human-engineered features

Neural language model (NLM)



https://web.stanford.edu/~jurafsky/slp3/

- predict next word using a sliding window (fixed length)

NLM vs. n-gram LM

I have to make sure that the cat gets fed.
(not seen: dog gets fed)

- task: I forgot to make sure that the dog gets ...
- n-gram LM cannot predict "fed" (not seen)
- NLM finds similarity of embeddings of "cat" and "dog"
- NLM is able to generalize and predict "fed" after "dog" (not seen)

Training NNs

- for all training pairs (x, y)
  - forward step to find $\hat{y}$
  - compute loss $L(\hat{y}, y)$
  - update/optimize weights of output layer
  - assess the influence of (all) hidden nodes and update weights

Training NNs

- for all training pairs (x, y)
  - forward step to find $\hat{y}$
  - compute loss $L(\hat{y}, y)$
  - update/optimize weights of output layer
  - assess the influence of (all) hidden nodes and update weights
    this is the "critical" part
    (let us have a short excursion to backpropagation; for more details, cf., deep learning lecture)