Machine Learning and
Computer Vision Group

IST AUSTRIA
Institute of Science and Technology

Deep Learning with TensorFlow

http://cvml.ist.ac.at/courses/DLWT_W18

Lecture 3:
Artificial Neural Networks
(Multilayer Perceptrons)

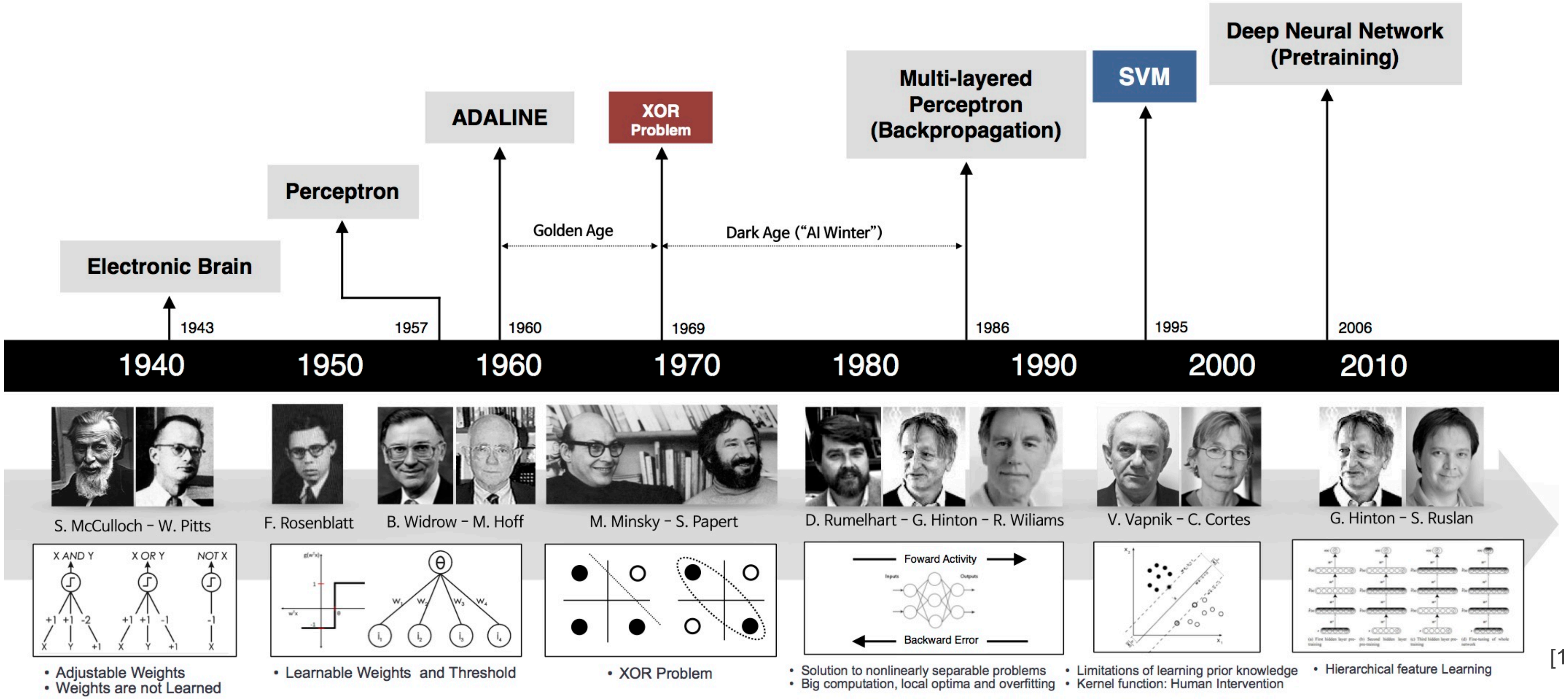
Artificial Neural Networks

Multi-layer Perceptrons

Lars Bollmann

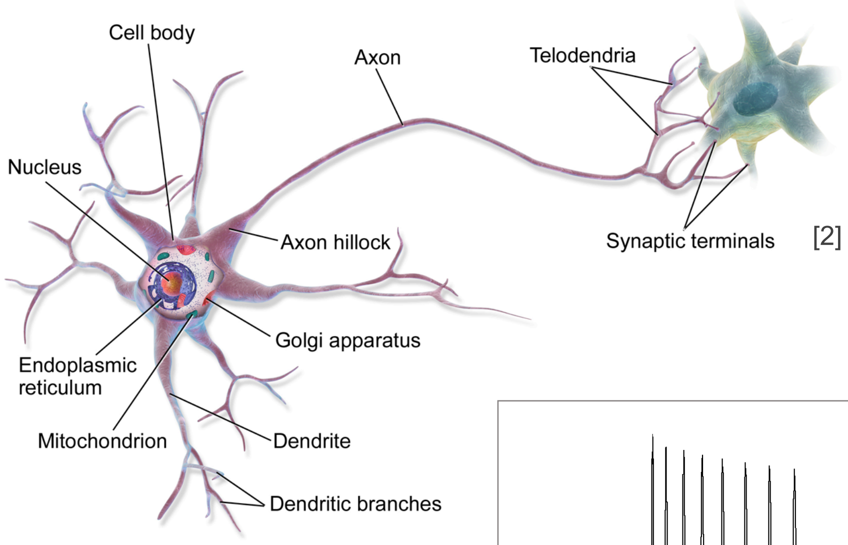
Contents

Brief history

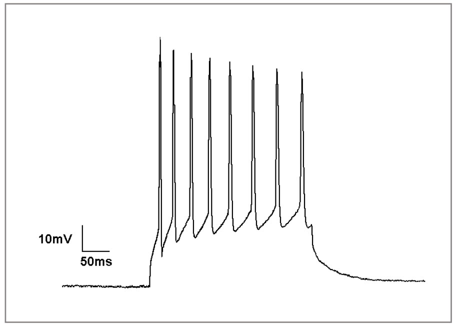


[1]

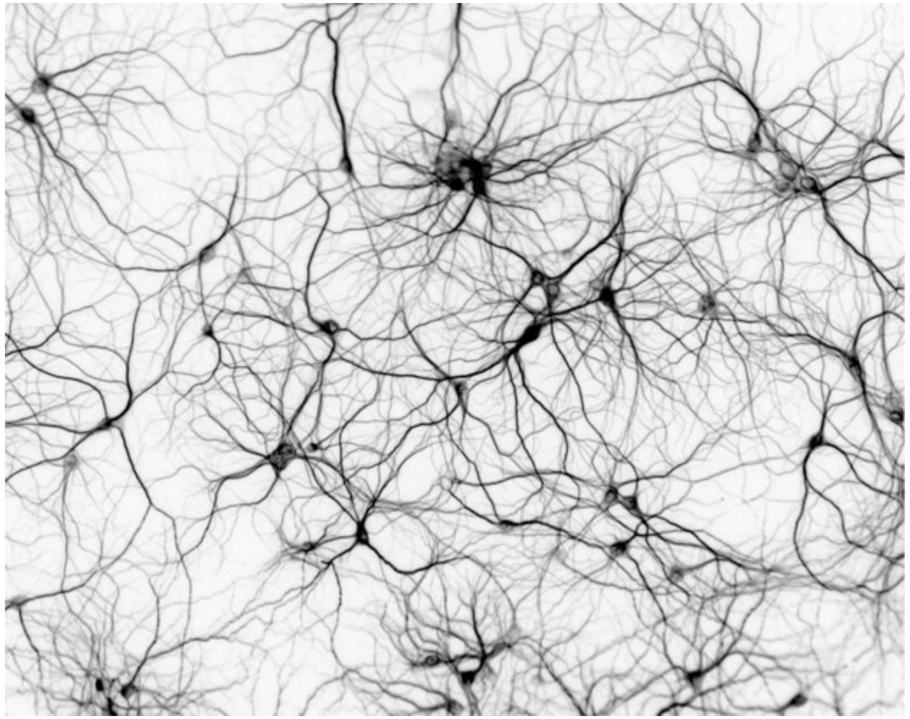
Biological neuron



[2]



[3]

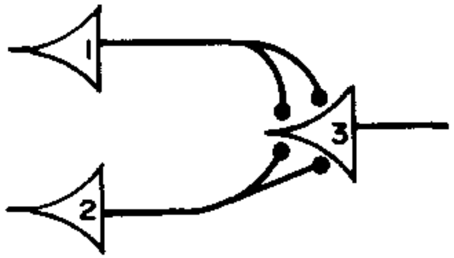


[4]

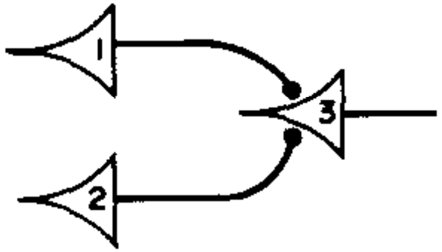
Artificial neural network (ANN)

McCulloch & Pitts (1943)

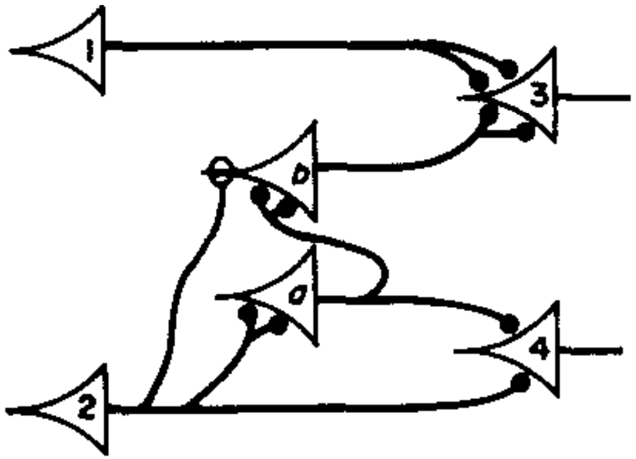
Logical OR



Logical AND



Heat illusion

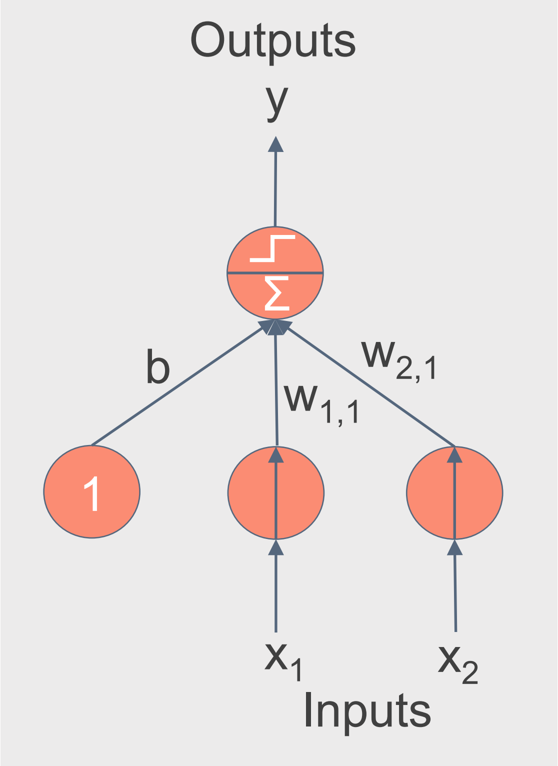


“learning”

All images from [5]

The Perceptron

Rosenblatt (1957)



Features

- Linear threshold unit:

$$y = \begin{cases} 1 & if \ \sum_{i=1}^{m} w_{i,1} \cdot x_i + b > 0 \\ 0 & otherwise \end{cases}$$

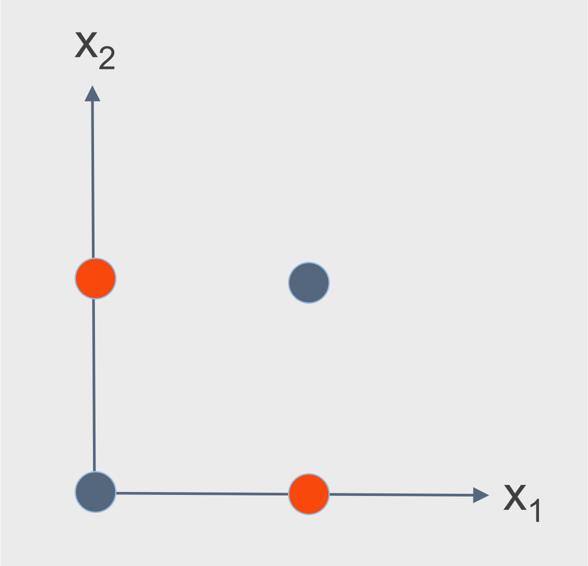
- Update rule for weights:

$$w_{i,j}^{(next\_step)} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

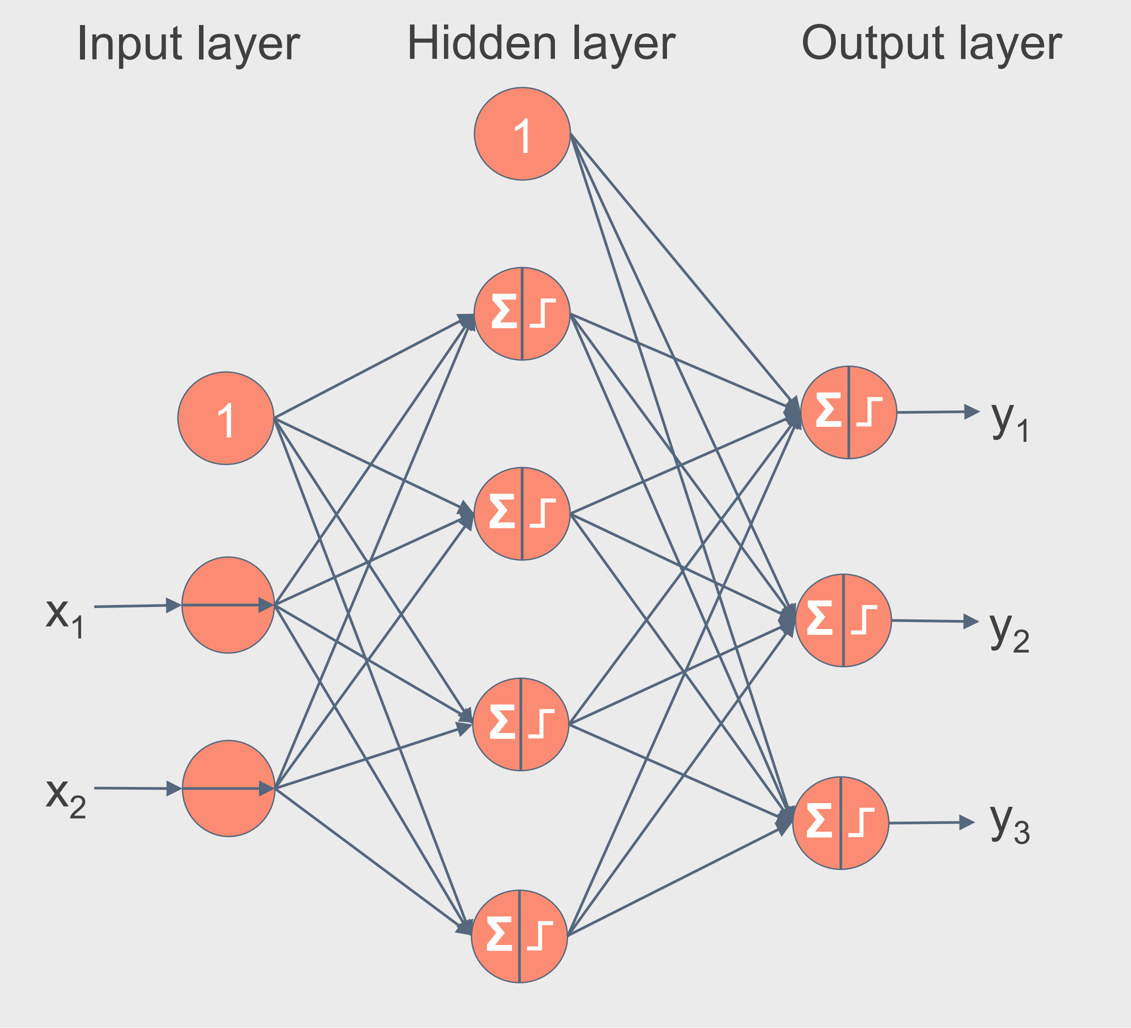
- Converges if training instances are linearly separable

Drawbacks

- Still a linear classifier



Multi-layer Perceptron (MLP)



- Activation of one neuron:

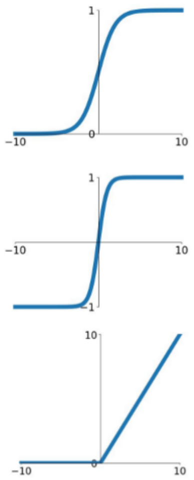
$$a_i^{(L)} = f_{activation}\left(\sum_j w_{(i,j)} a_j^{(L-1)} + b\right)$$

- Alternative activation functions:

Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$

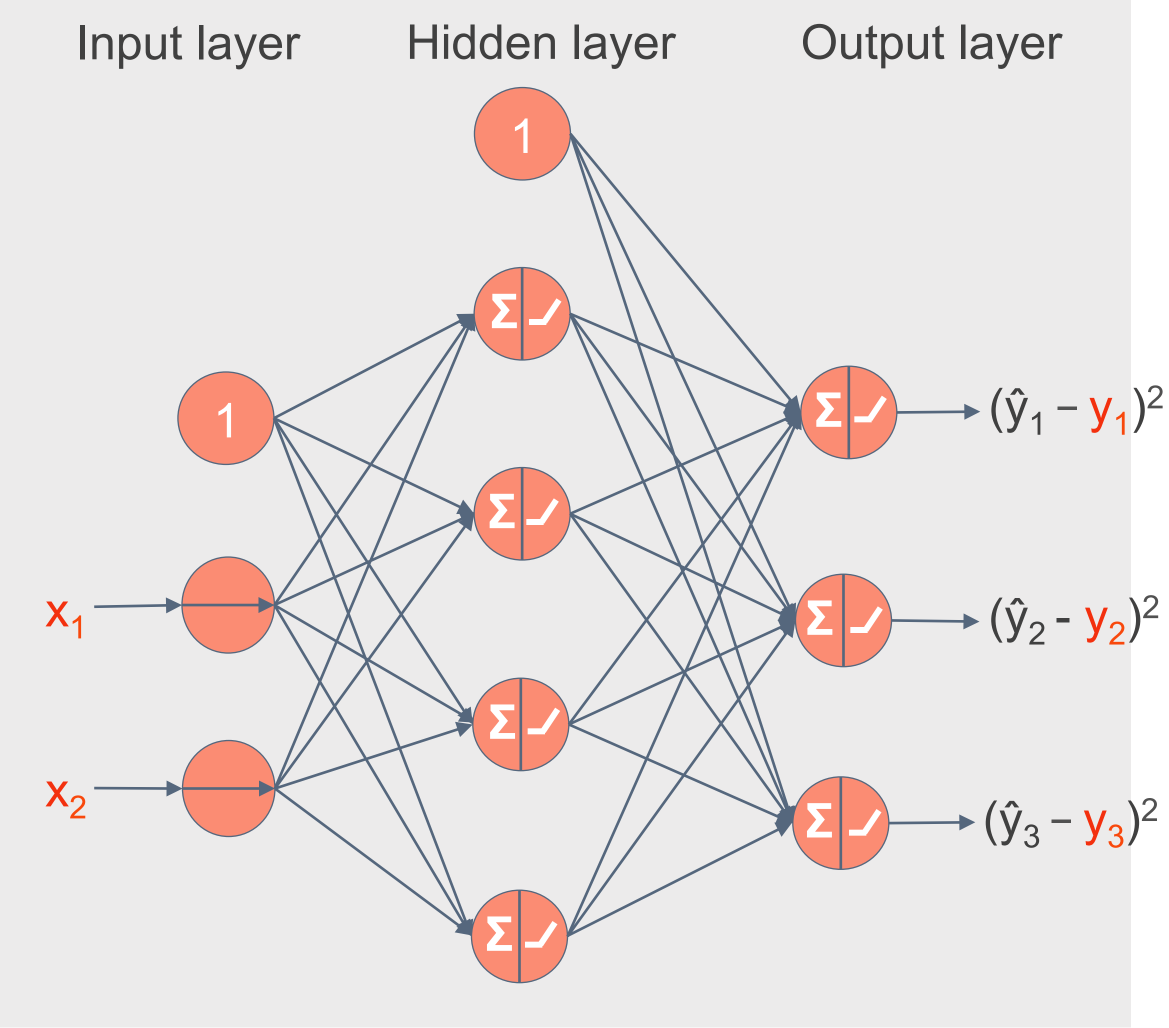
tanh
$\tanh(x)$

ReLU
$\max(0, x)$



[6]

Training

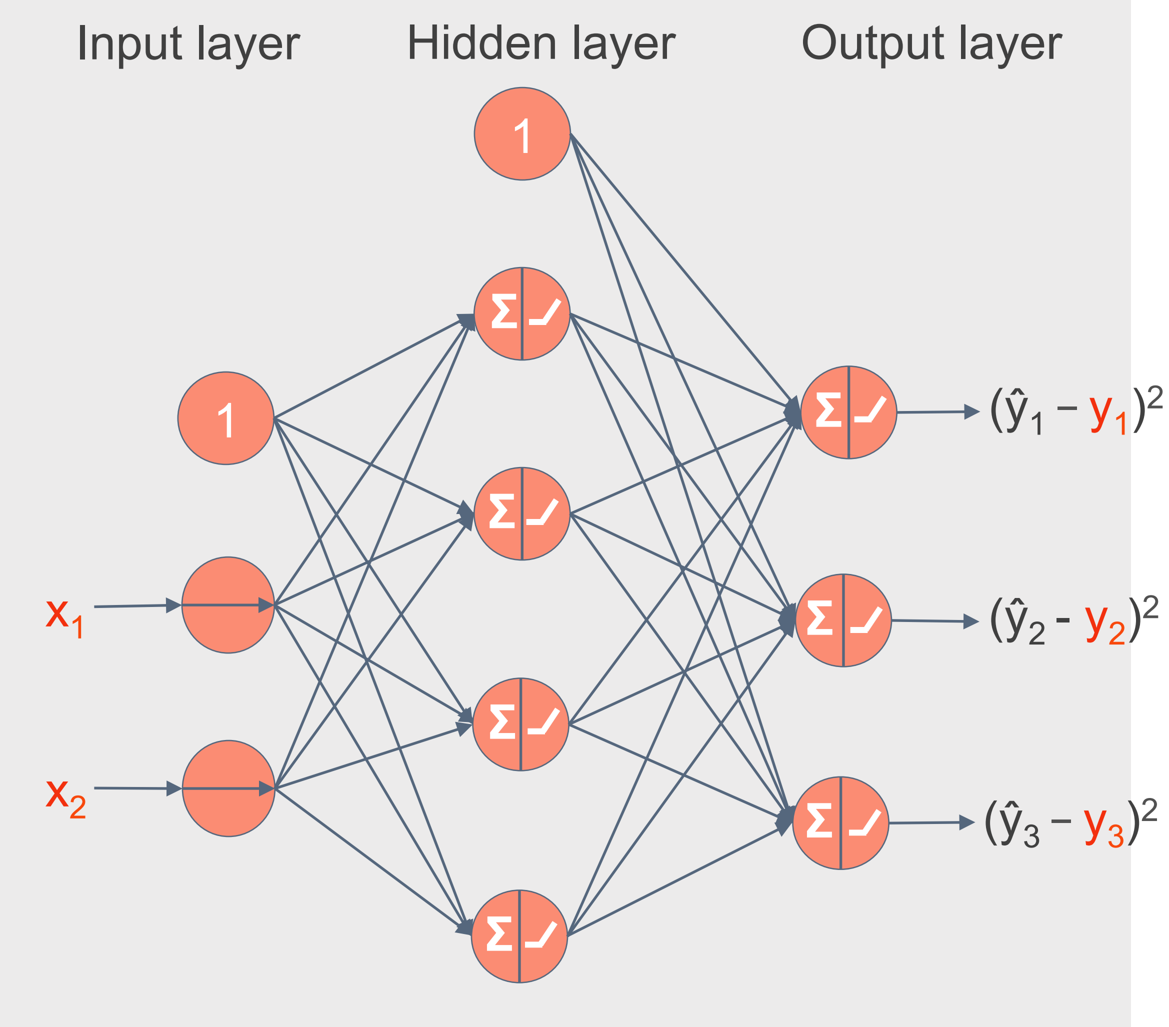


- Activation of one neuron:

$$a_i^{(L)} = f_{activation}\left(\sum_j w_{(i,j)} a_j^{(L-1)} + b\right)$$

- Find optimal weights & bias terms to reduce cost function (e.g. MSE, cross-entropy)
- Weights: influence of neurons in previous layer
- Bias terms: nudge towards active/inactive

Training



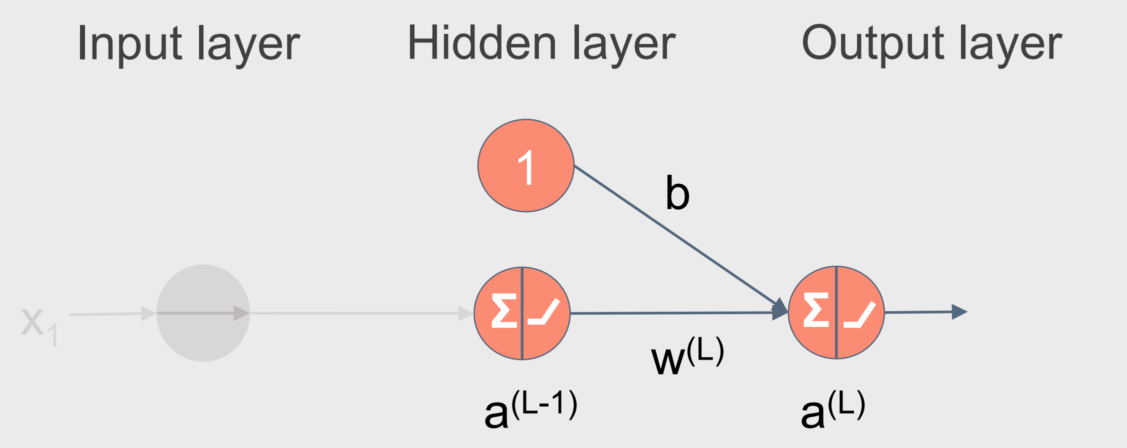
Minimize cost function using GD

- Initialize weights & bias terms with random values
- Average cost over all instances:

$$C = \frac{1}{m}\sum_{i=1}^{m} C_i$$

- Mini-batches to compute gradient with respect to weights and biases
- Gradient descent step

Training



- Cost: $C_0 = \left(a^{(L)} - y\right)^2$
- Activation: $a^{(L)} = f_{act}(z^{(L)})$ with $z^{(L)} = w^{(L)} a^{(L-1)} + b$
- Derivative of C_0 with respect to $w^{(L)}$:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}}$$

Backpropagation

- Forward pass: compute output of all neurons for training instance
- Backward pass: compute partial derivatives for each weight and bias
- Average over mini-batch:

$$\frac{\partial C_{mb}}{\partial w^{(L)}} = \frac{1}{m_{mb}} \sum_{i=1}^{m_{mb}} \frac{\partial C_i}{\partial w^{(L)}}$$

- Combine partial derivatives
  → gradient vector

Tensorflow

High-Level API

- TF.Learn
- DNNClassifier
- Parameters:
  - # layers
  - # neurons per layer
  - batch size
  - # iterations
  - activation function

Plain Tensorflow

1. Construction phase
2. Training phase
3. Using the trained network

→ Demo in Jupyter

Hyperparameters

hidden layers

- More layers:
    - exponentially fewer neurons for complex functions
    - Converge faster
    - Generalize better
    - Reuse of layers
- Start with few layers and increase number

neurons / layer

- Funnel or constant size
- “black art”
- Too many neurons cause overfitting

activation

- ReLU is a good choice in general
- Softmax for output when classes are mutually exclusive

Summary

From Artifical Neurons to Deep Neural networks

Initial concepts in the 1940s, dark age and recent boom

Multi-layer Perceptron

Layers, neurons, bias, weights, activation function

Training of ANN

Gradient descent, backpropagation & mini-batches

Implementation in Tensorflow

High-Level API, Plain TensorFlow, hyperparameter tuning

References

[1] https://www.slideshare.net/deview/251-deep-learning-using-cu-dnn/4

[2] https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

[3] https://en.wikipedia.org/wiki/Neural_oscillation

[4] http://www.lesicalab.com/research/

[5] McCulloch and Pitts, A logical calculus of the ideas immanent in nervous activity

[6] https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba09