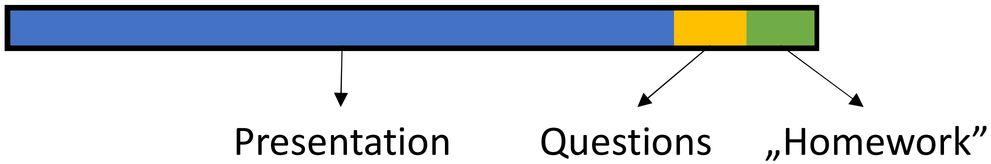
Introduction to Deep Learning

Bazyli Polednia 2020

Plan for today

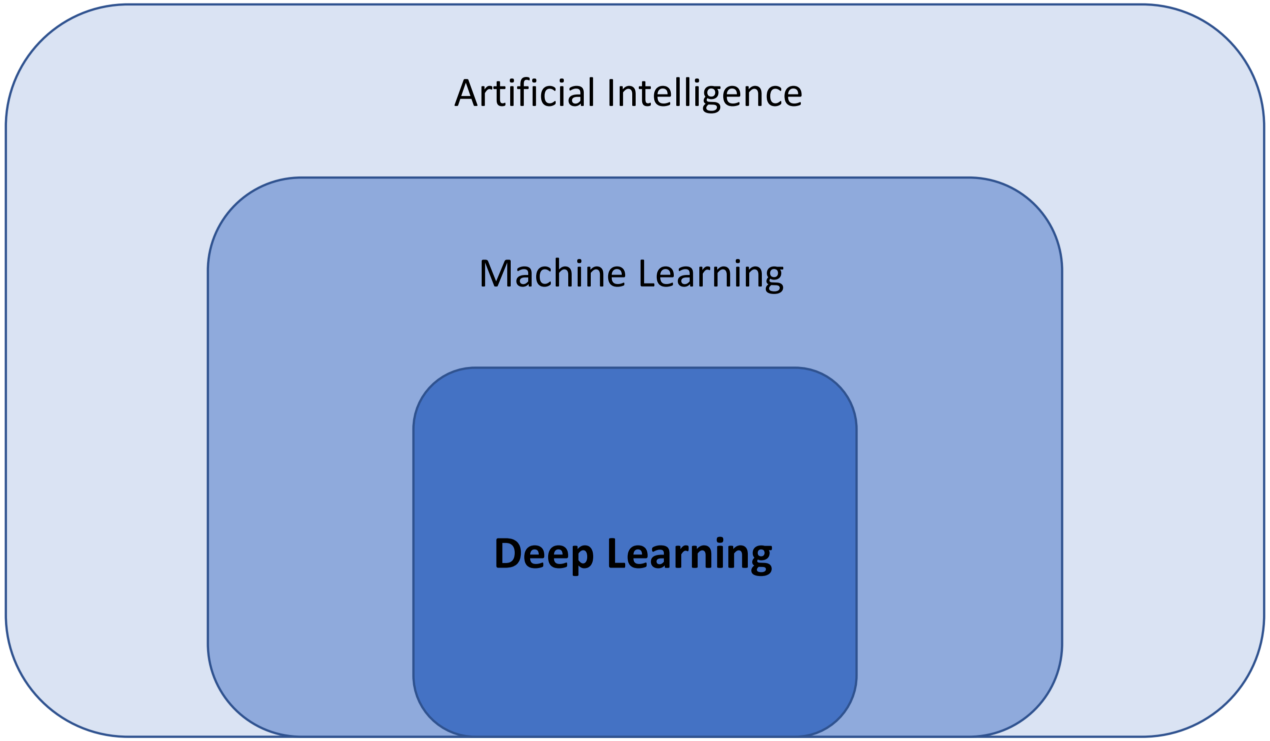
Total: 1h



Resources

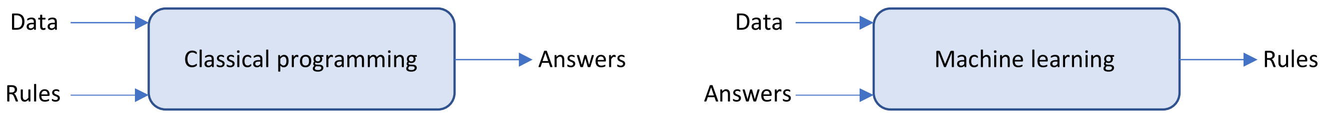
- MIT 6.S191 – Introduction to Deep Learning
- „Deep Learning with Python” – Francois Chollet
- Stanford CS230 – Deep Learning
- Tensorflow Tutorials
- Kaggle Competitions

Deep Learning – what it actually is?

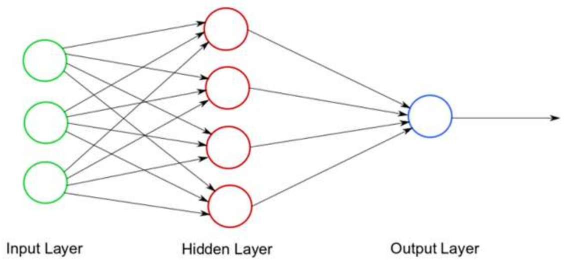


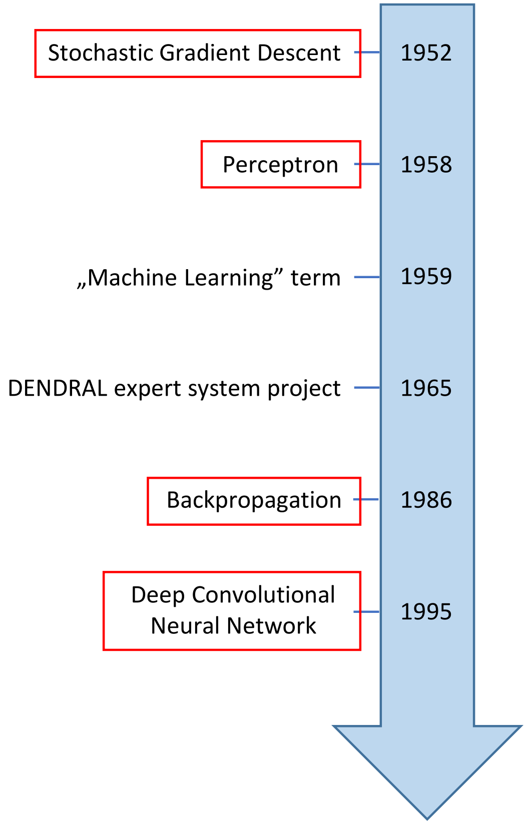
Artificial Intelligence – uses various algorithms to mimic human behaviour. Usually consists of handcrafted rules – such paradigm is called symbolic AI

Machine Learning – instead of following explicit rules provided by programmer, programs learn them by analysing already exisiting answers to given data



Deep Learning – branch of machine learning using neural networks to recognise patterns in data





Deep Learning Boom in recent years

1. **Hardware and software**
    - Introduction of TPUs and more powerful GPUs
    - Parallelization of computations
    - Programming interfaces e.g. Nvidia CUDA
    - Deep Learning libraries and toolkit e.g. Tensorflow, PyTorch, Keras

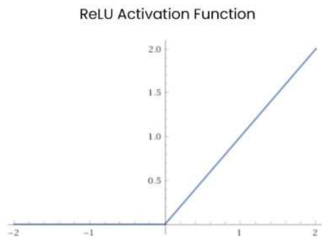
TensorFlow

2. **Datasets and benchmarks**
    - „Big Data”
    - Exponential growth of hardware storage
    - Public datasets e.g. ImageNet, Kaggle competitions

IMAGENET

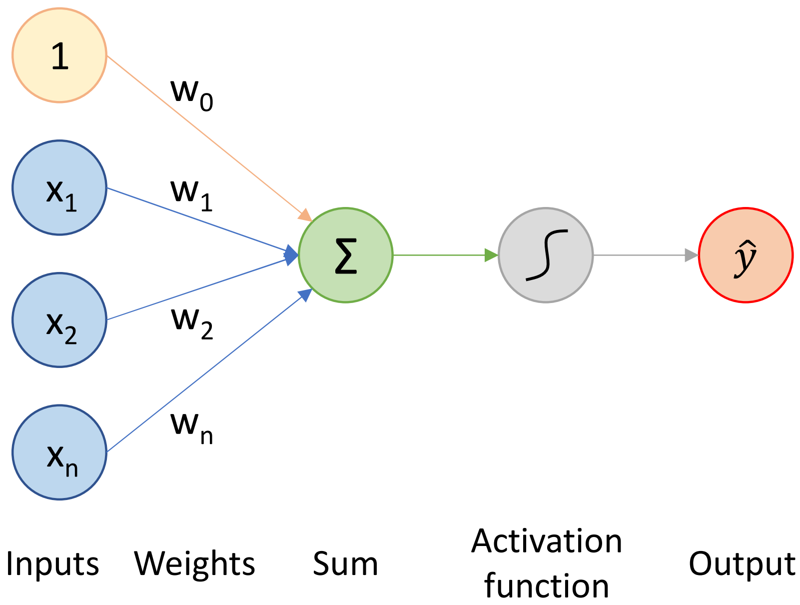
kaggle

3. **Algorithmic advances**
    - New activation functions
    - New optimizers
    - Batch optimalization
    - …



Perceptron

How is a perceptron built?



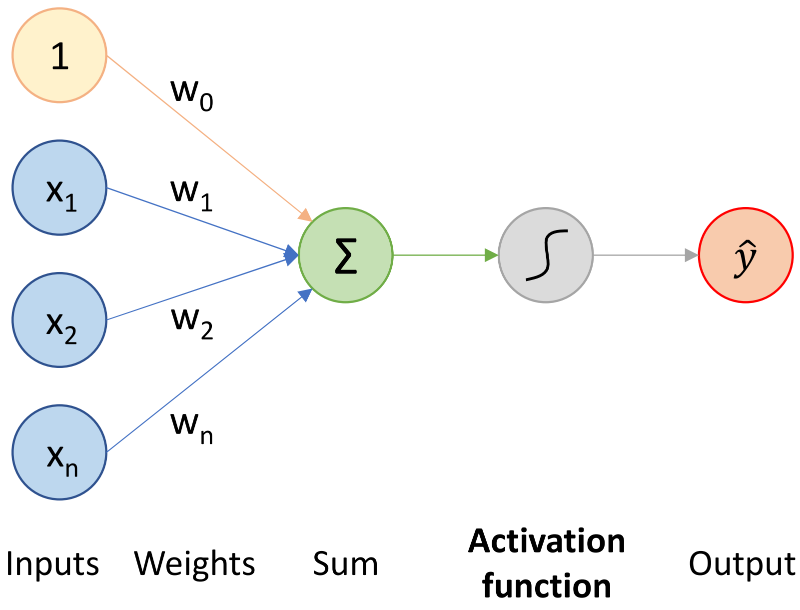
Output · Activation function · Bias · Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{j=1}^{n} x_j\, w_j\right)$$

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \mathrm{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$$\hat{y} = g(w_0 + X^T\, W)$$

How is a perceptron built – activation function



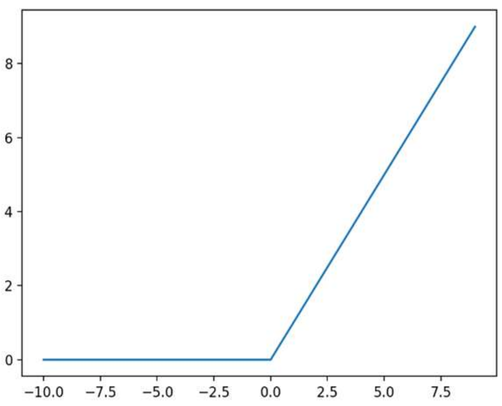
Output | **Activation function** | Bias | Linear combination of inputs

$$\hat{y} = g(w_0 + X^T W)$$

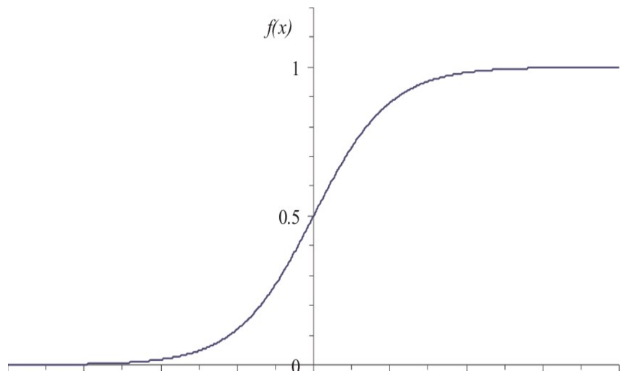
Activation function – what is it?

- Non-linear function
- Transforms linear operations (dot product and addition) of weighted inputs and bias
- Introduces non-linearity to the output

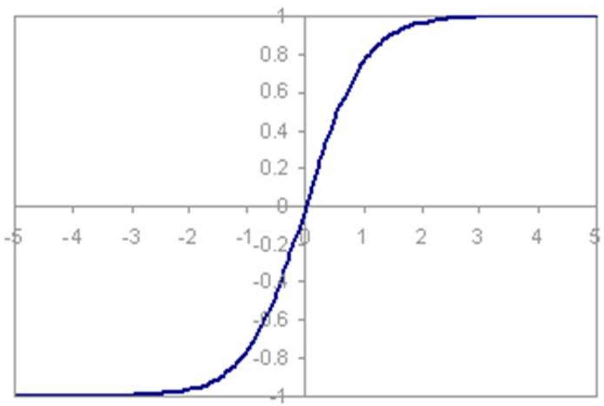
Common examples



ReLU (Rectified Linear Unit)

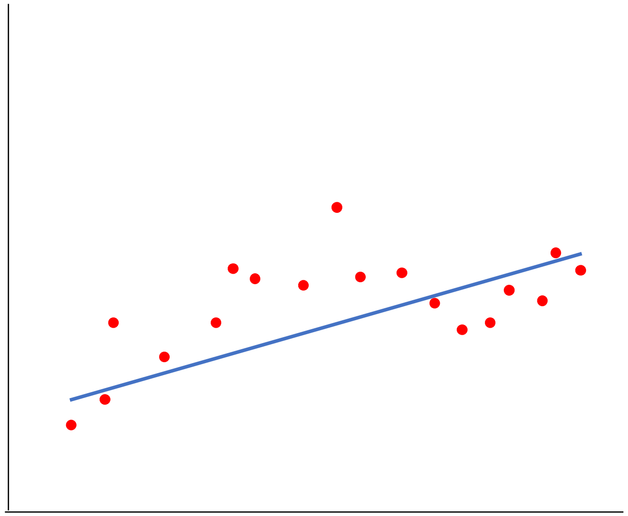


Sigmoid

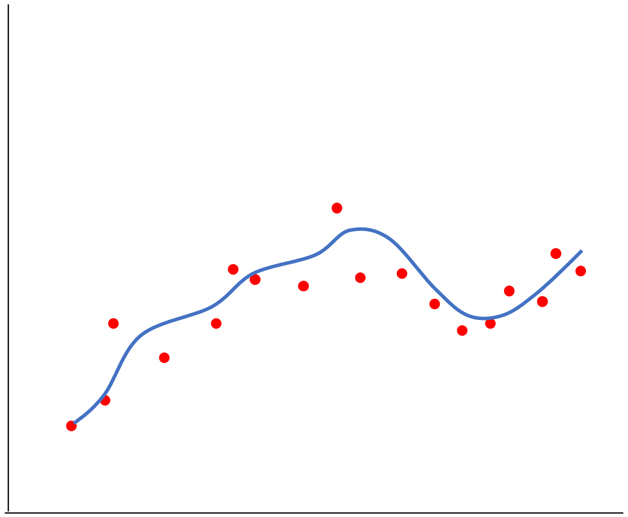


Hyperbolic tangent

Activation function – why is it used?



Linearity

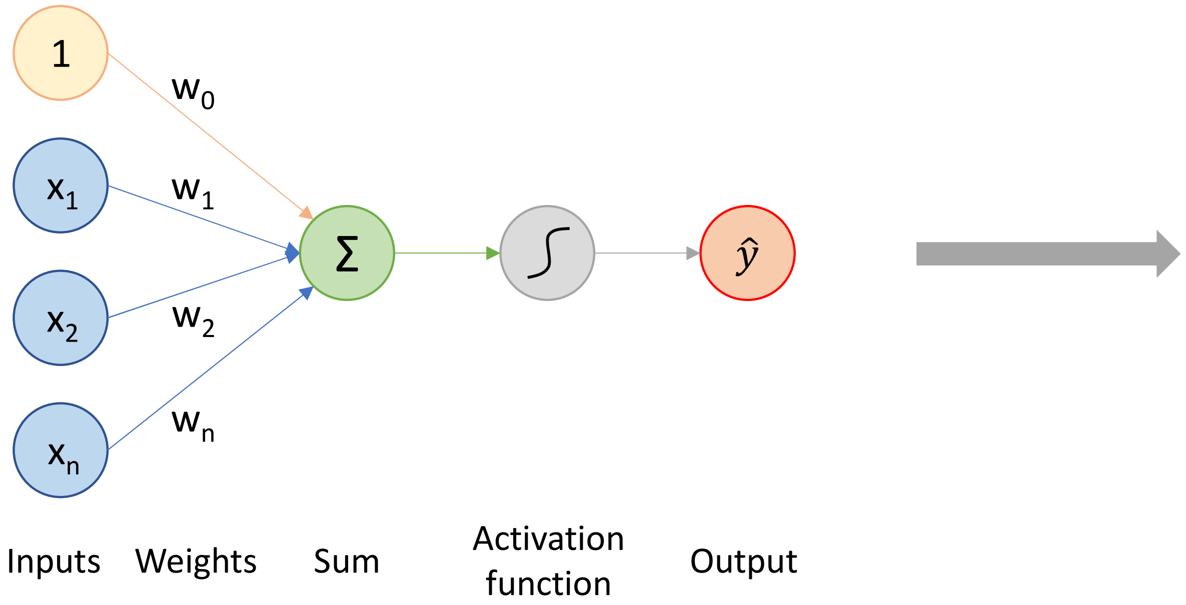


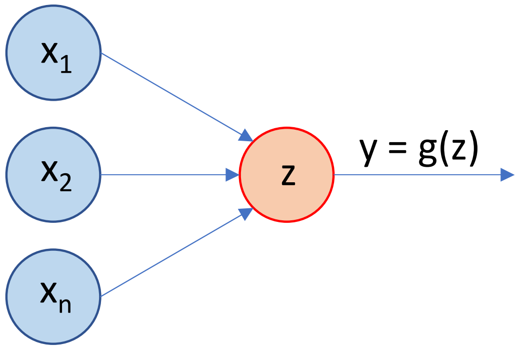
Non-linearity

Without non-linear activation function, the perceptron could only learn linear transformations and would only produce linear outputs – which almost never exist in real life

Neural network

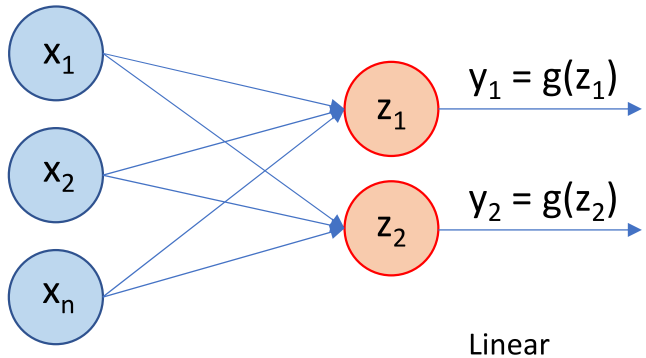
Simplified perceptron notation





$$z = w_0 + \sum_{j=1}^{n} x_j \, w_j$$

Let’s connect two perceptrons to the same inputs



Dense layer

Linear combination of inputs

Bias

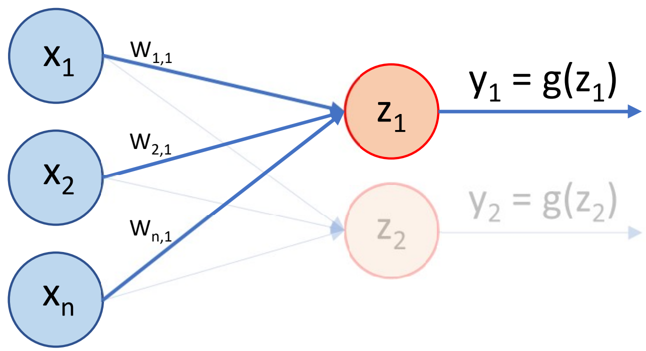
$$z_i = w_{0,i} + \sum_{j=1}^{n} x_j \, w_{j,i}$$

$$\hat{y}_i = g(z_i)$$

Output

Activation function

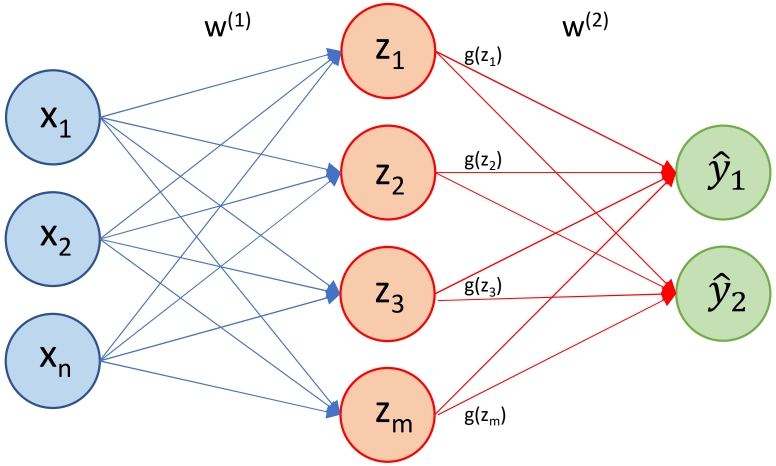
Let's connect two perceptrons to the same inputs



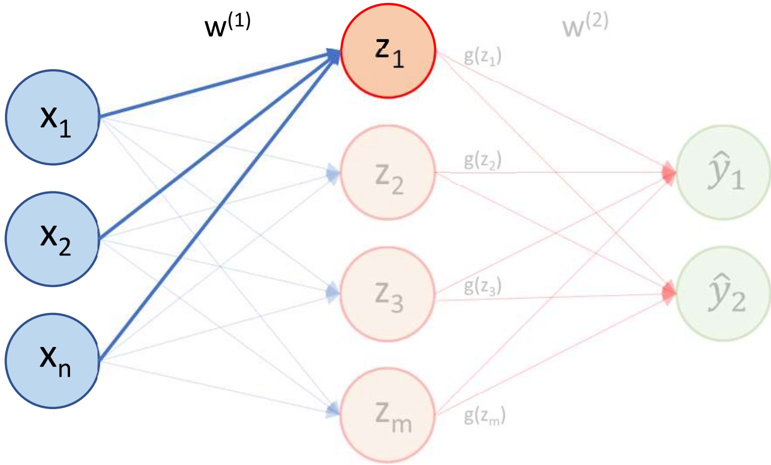
$$z_1 = w_{0,1} + \sum_{j=1}^{n} x_j \, w_{j,1}$$

$$\hat{y}_1 = g(z_1)$$

Single Layer Neural Network

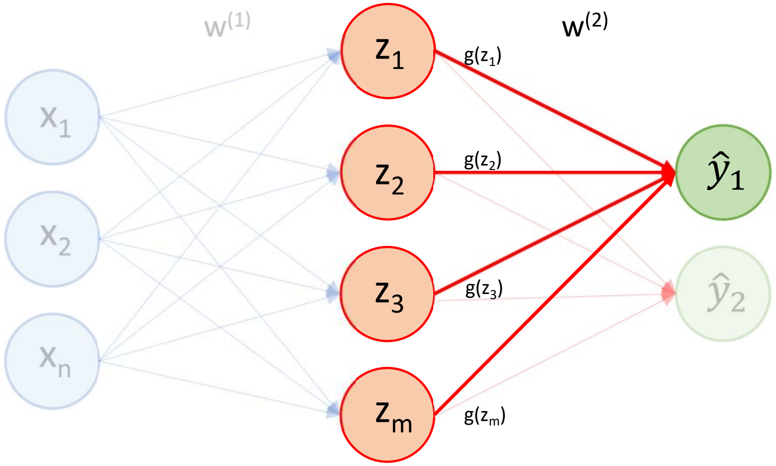


Single Layer Neural Network



$$z_1 = w_{0,1}^{(1)} + \sum_{j=1}^{n} x_j \, w_{j,1}^{(1)}$$

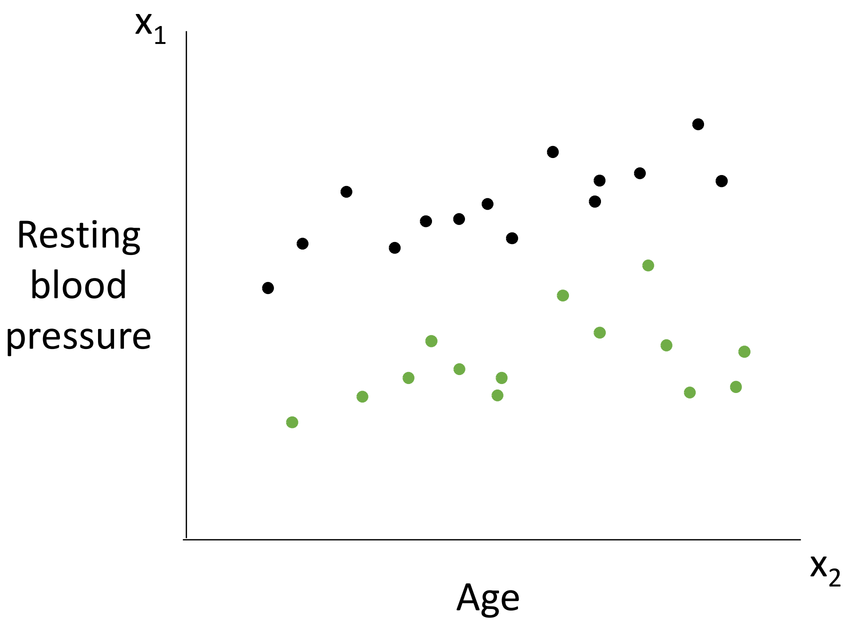
Single Layer Neural Network

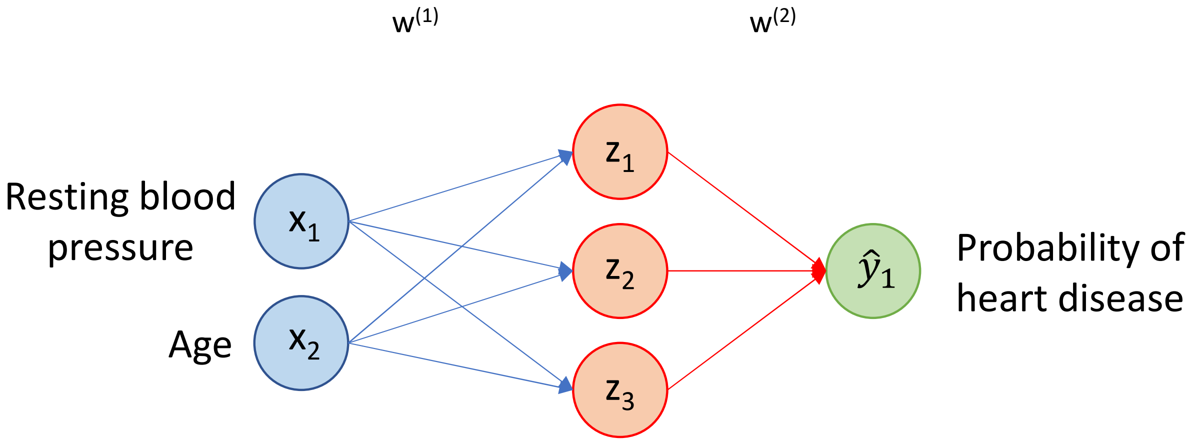


$$\hat{y}_1 = g\left(w_{0,1}^{(2)} + \sum_{j=1}^{m} x_j \, w_{j,1}^{(2)}\right)$$

Example problem

Does the patient have a heart disease?

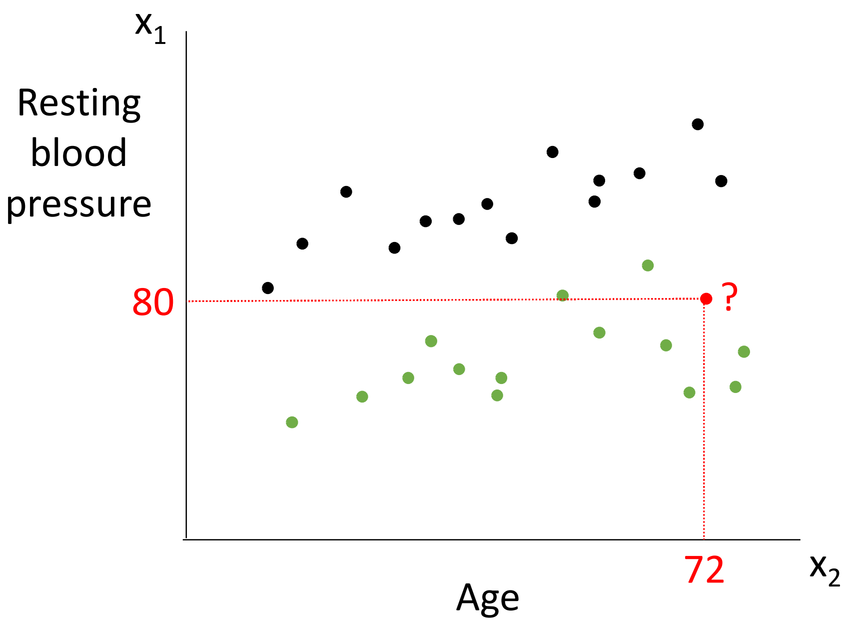


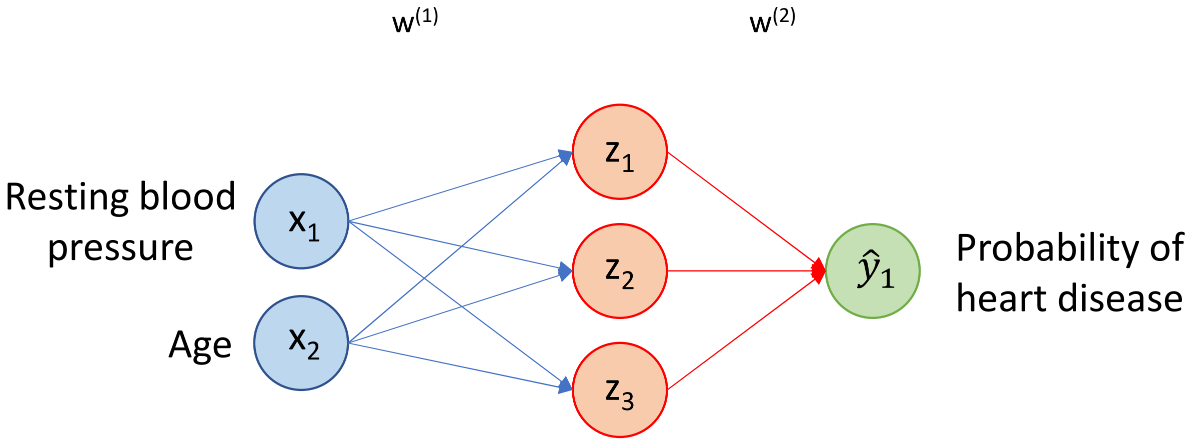


Gradient PG

Example problem

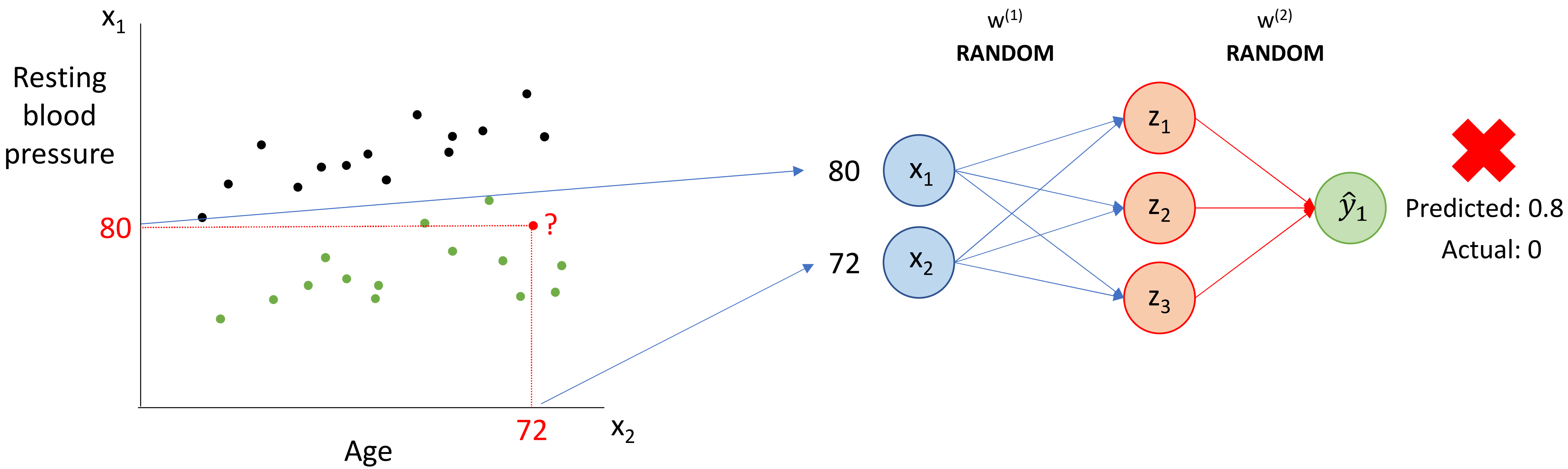
Does the patient have a heart disease?





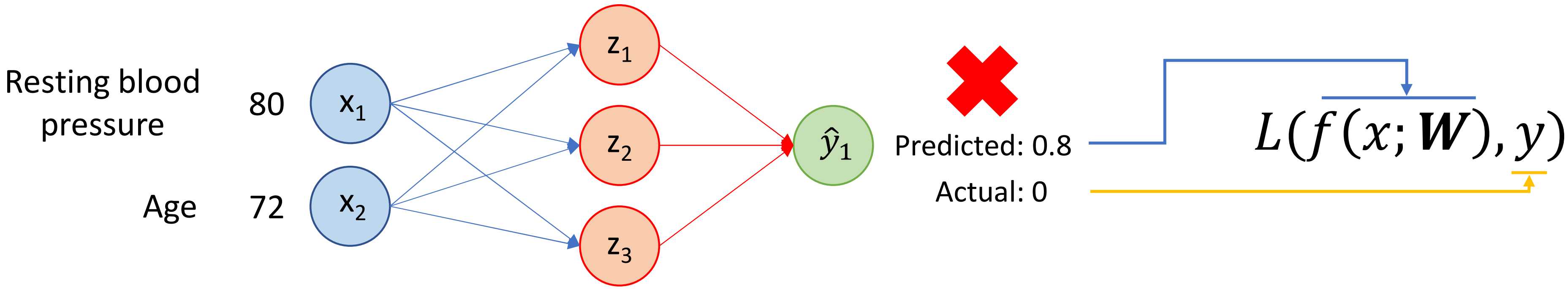
Example problem

Does the patient have a heart disease?



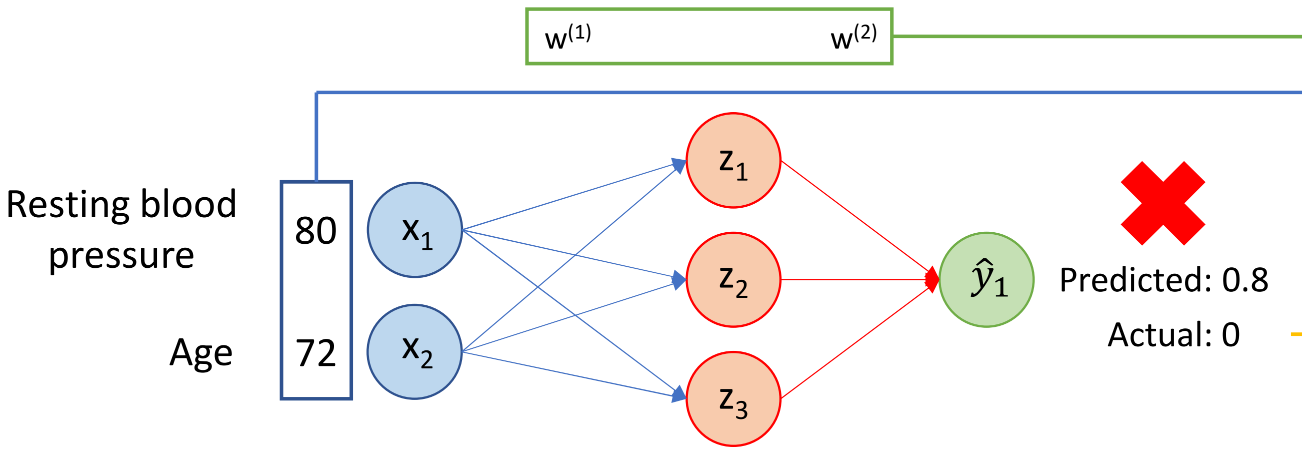
Training the network

Loss function



Loss function measures how much our predictions differ from actual results

Loss function

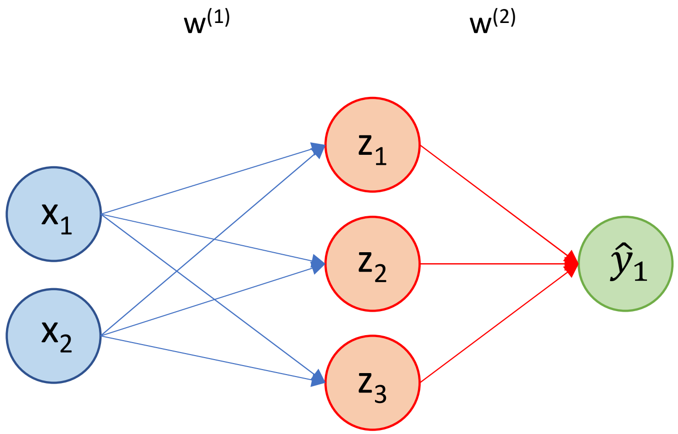


$$L(f(\boxed{x};\boxed{\boldsymbol{W}}),\underline{y})$$

Loss function measures how much our predictions differ from actual results

Empirical loss

$$\boldsymbol{X} = \begin{bmatrix} 75, 40 \\ 85, 80 \\ \vdots \\ 60, 20 \end{bmatrix}$$



$$\begin{array}{cc} \text{Predicted} & \text{Actual} \\ f(x^i; \boldsymbol{W}) & y^i \\ \begin{bmatrix} 0.1 \\ 0.3 \\ \vdots \\ 0.8 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \end{array}$$

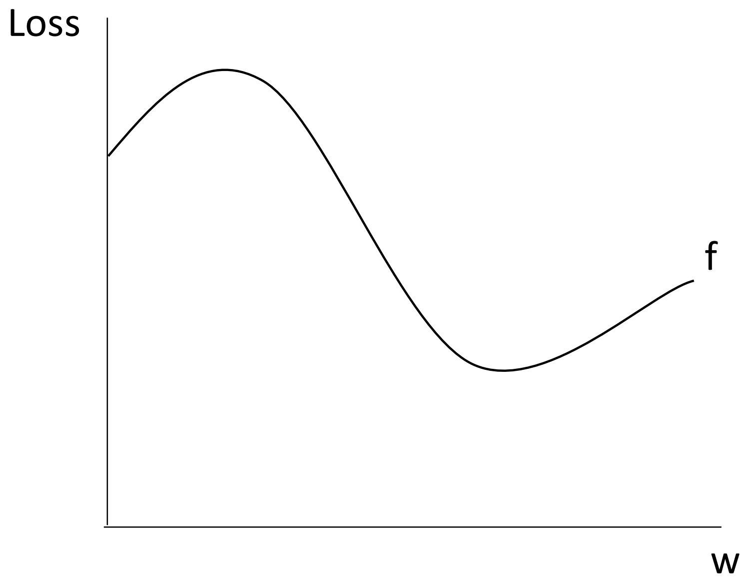
$$J(\boldsymbol{W}) = \frac{1}{n} \sum_{i=1}^{n} L\left(f\left(x^i; \boldsymbol{W}\right), y^i\right)$$

Empirical loss measures loss over the whole dataset, calculating the average of losses for each input

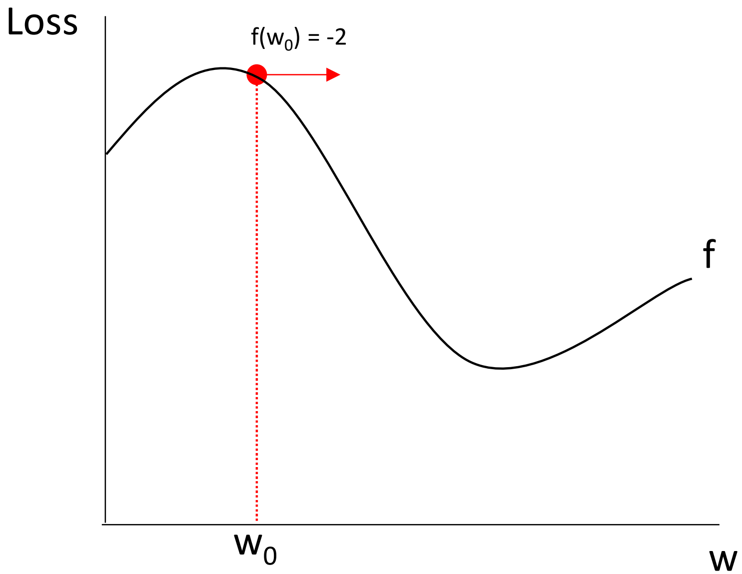
Gradient PG

Gradient-based optimization

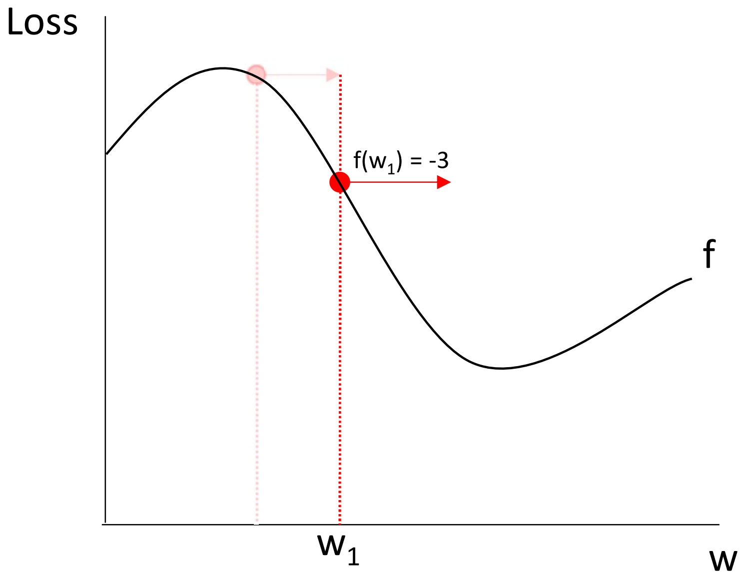
Problem: minimize loss over weight w



Gradient-based optimization

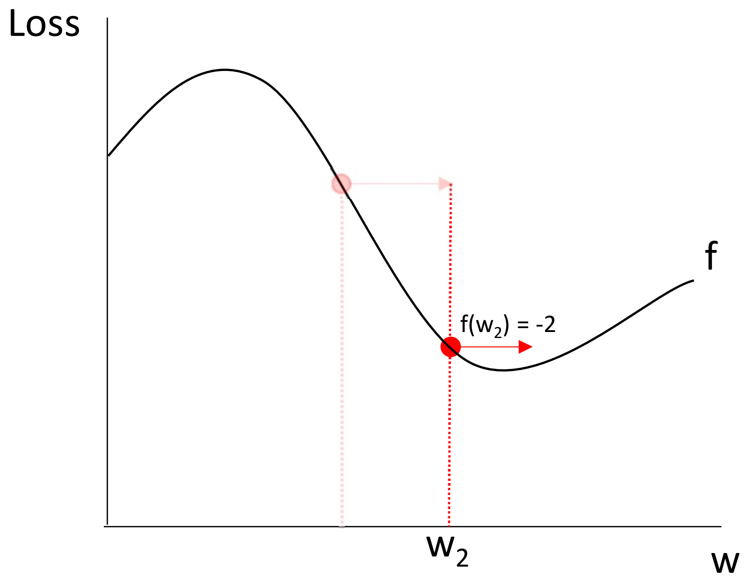


Gradient-based optimization

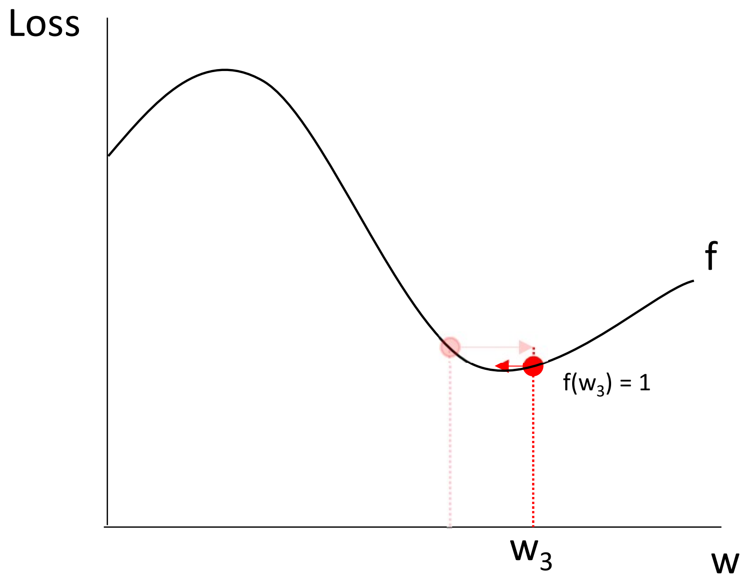


Gradient PG

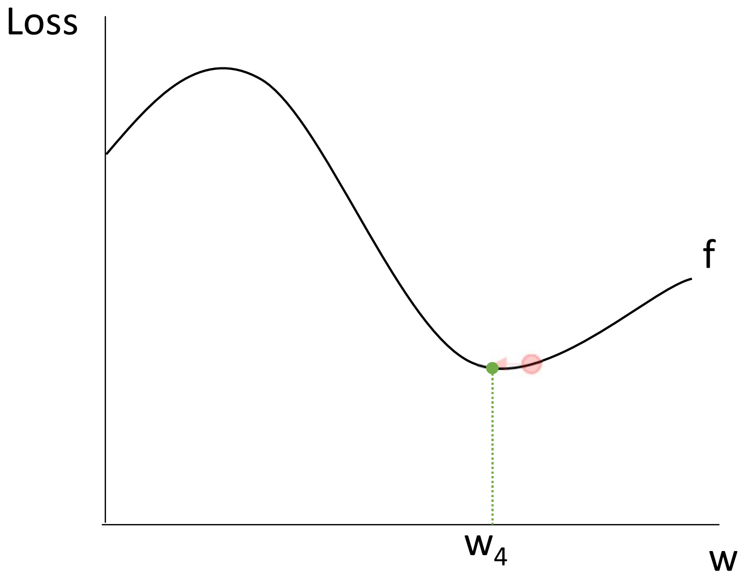
Gradient-based optimization



Gradient-based optimization



Gradient-based optimization



Gradient Descent

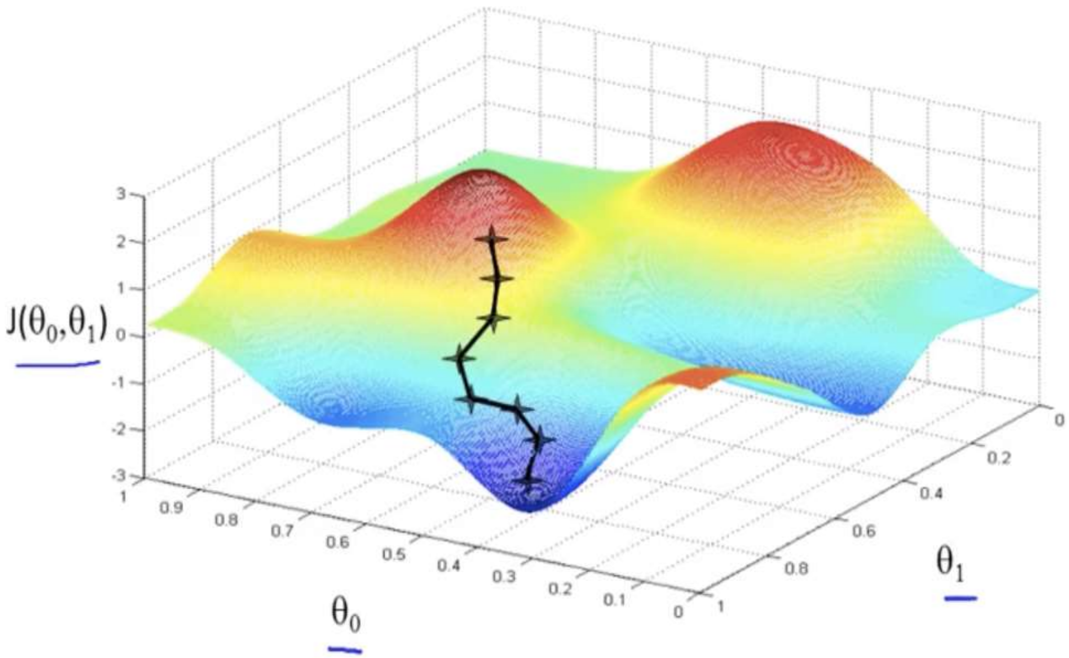


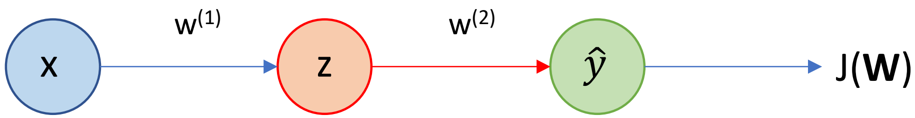
Image source: „On Why Gradient Descent is Even Needed” Daniel Burkhardt Derigo

Gradient Descent

1. Initialize random weights
2. Loop until convergence:
    - Compute loss function gradient: $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
    - Update weights based on gradient: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$ η – learning rate

Backpropagation

Problem: compute loss function gradient $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$

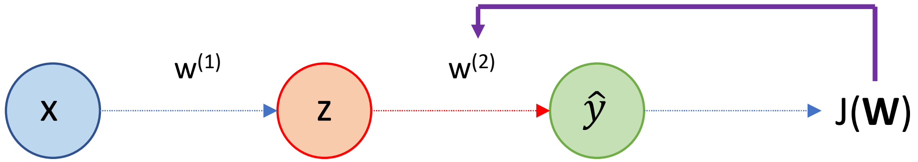


$$\frac{\partial J(\boldsymbol{W})}{\partial w^{(1)}} \quad ? \quad \frac{\partial J(\boldsymbol{W})}{\partial w^{(2)}}$$

Chain rule

Backpropagation

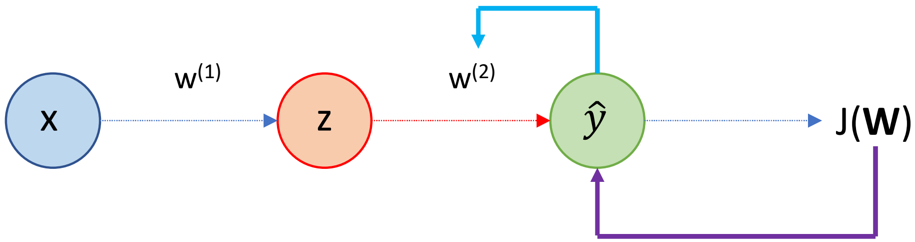
Problem: compute loss function gradient $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$



$$\frac{\partial J(\boldsymbol{W})}{\partial w^{(2)}} = ?$$

Backpropagation

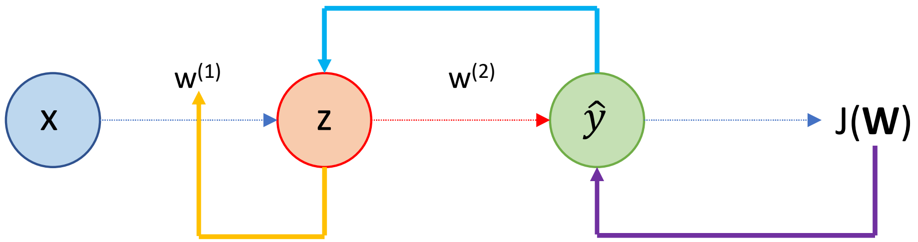
Problem: compute loss function gradient $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$



$$\frac{\partial J(\boldsymbol{W})}{\partial w^{(2)}} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w^{(2)}}$$

Backpropagation

Problem: compute loss function gradient $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$

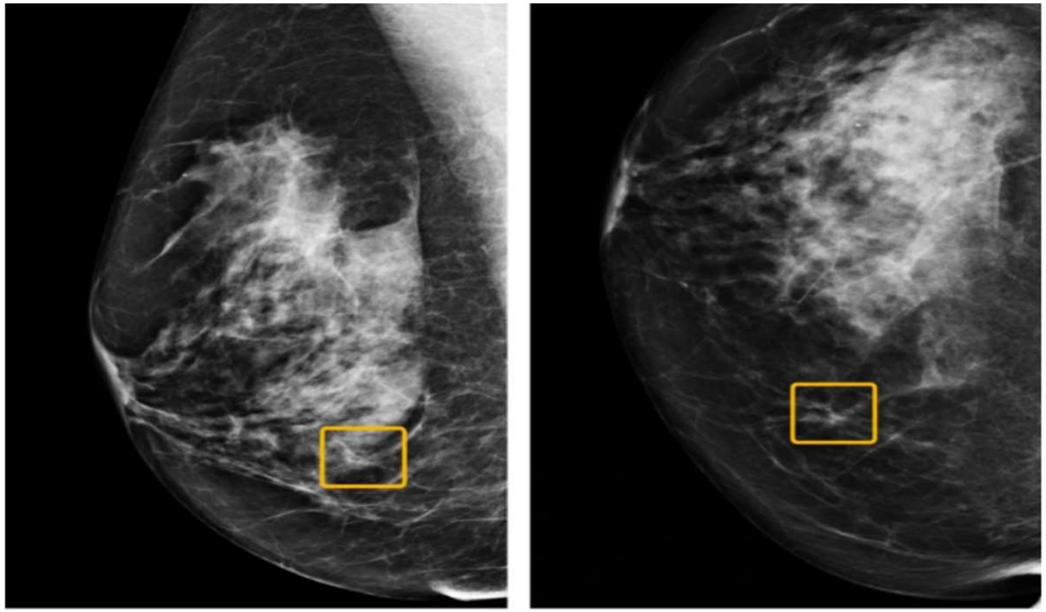


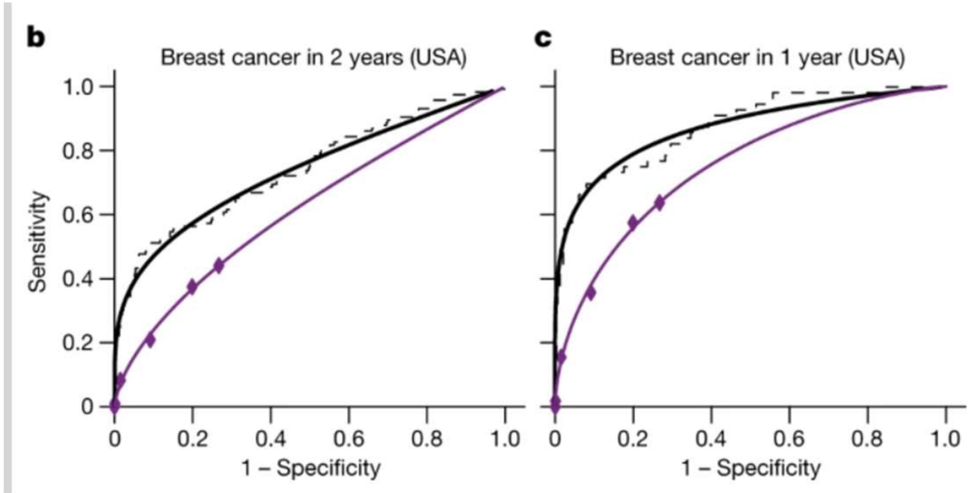
$$\frac{\partial J(\boldsymbol{W})}{\partial w^{(2)}} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w^{(2)}}$$

$$\frac{\partial J(\boldsymbol{W})}{\partial w^{(1)}} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial w^{(1)}}$$

What can we use Deep Learning for?

Clinical diagnosis





„International evaluation of an AI system for breast cancer screening" S. M. McKinney, M. Sieniek, V. Godbole, J. Godwin 2020

Real-time object detection

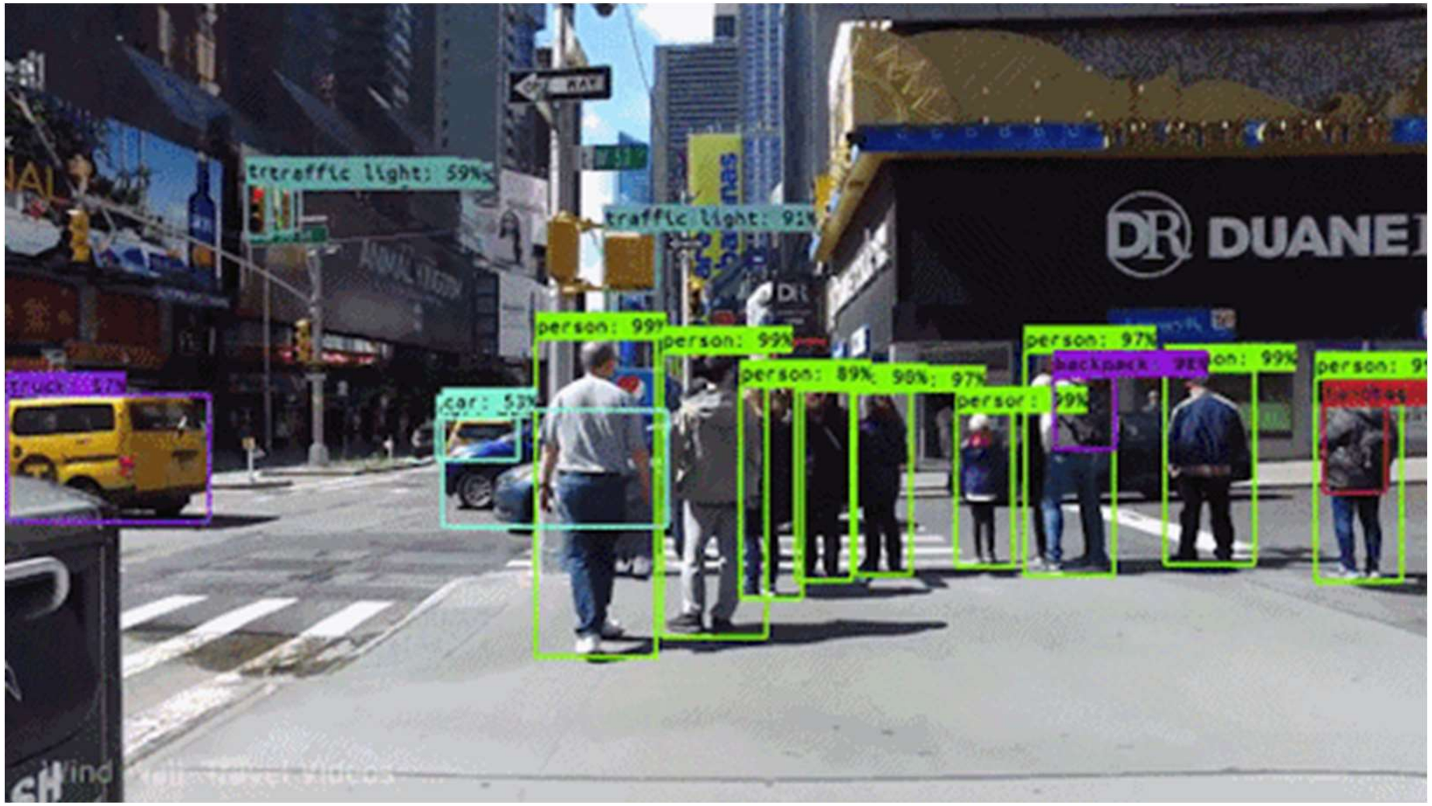


Image source: Detect-Me

Image alteration

Image alteration

And much more…

Questions?

Thank you for attention