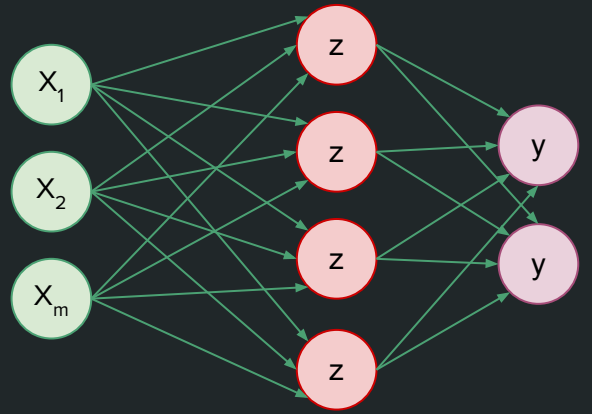
Shallow Introduction to Deep Learning

Pau Badia



Context

Artificial Intelligence

Any technique that enables computers to mimic human behaviour

Machine Learning

Ability to learn without explicitly being programed

Deep Learning

Extract patterns from data using neural networks

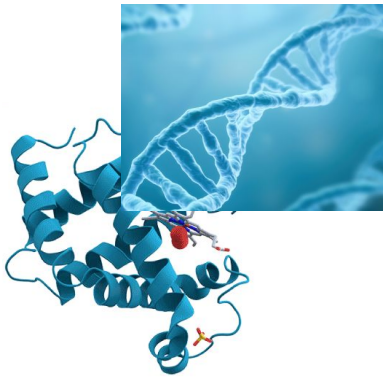
Why deep learning

- Features describe your data
- Deep learning learns underlying features directly from data

High level features

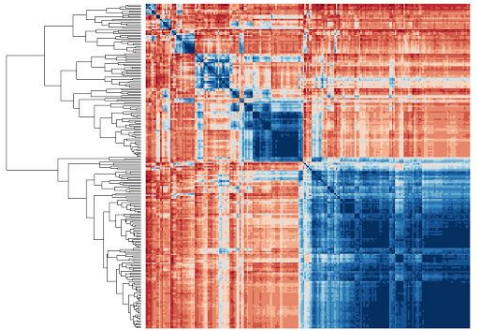
Patient's health status

Mid level features



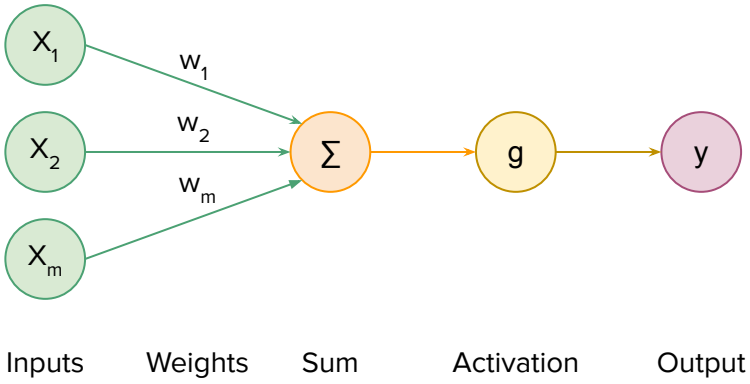
Genome
Proteome

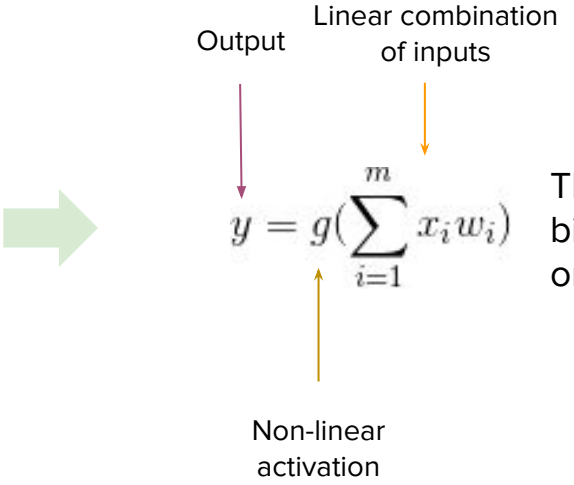
Low level features



Specific mutation profile
Metabolic quantification profile

Perceptron (neuron)



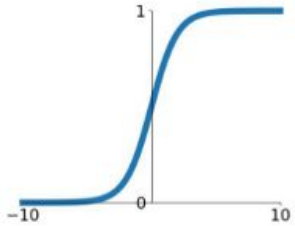


There can also be biases (b) but we will omit them.

Activation functions

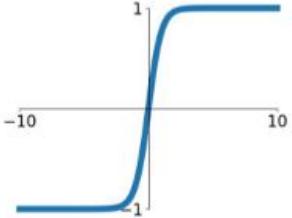
Sigmoid

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



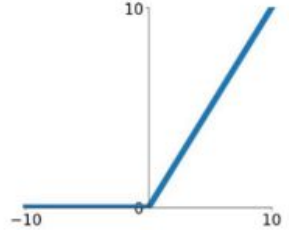
tanh

$$\tanh(x)$$



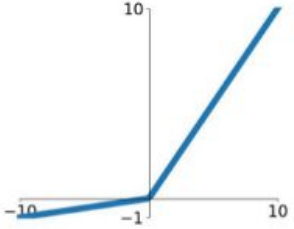
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

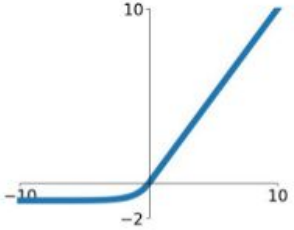


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

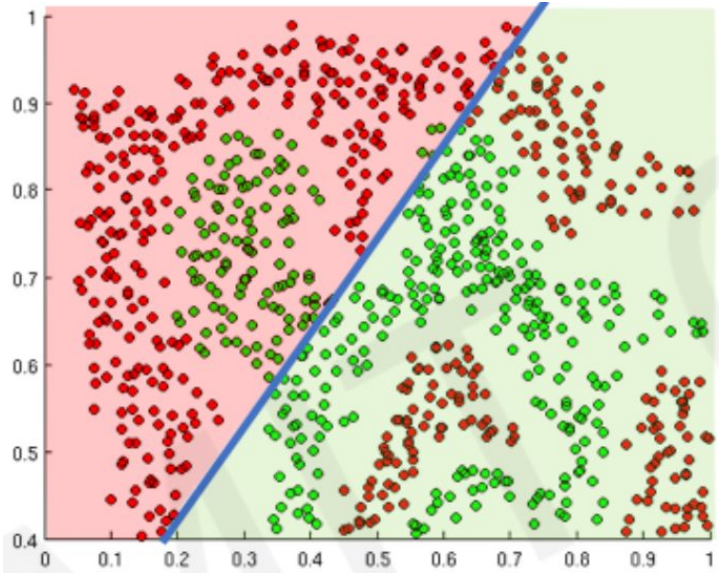
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

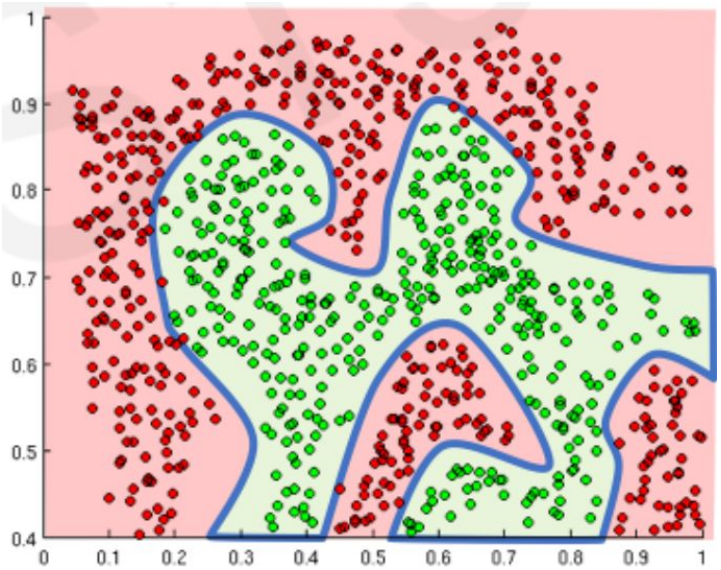


Activation functions

- Activation functions introduce non-linearities into the network

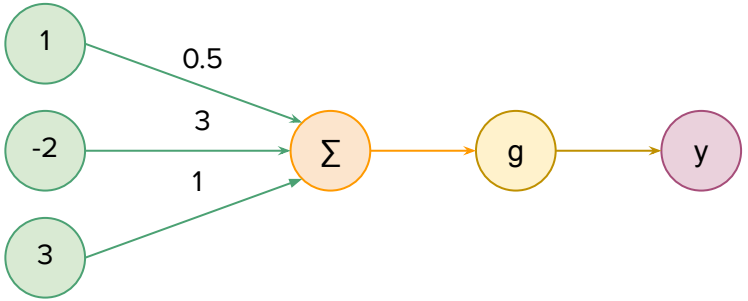


Linear decision



Arbitrarily complex functions

Example



Inputs Weights Sum Activation Output

$$y = g(\sum_{i=1}^{m} x_i w_i)$$

y = g((1 x 0.5) + (-2 x 3) + (3 x 1))

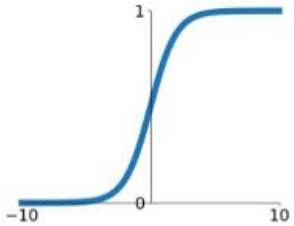
y = g(0.5 - 6 + 3)

y = g(-2.5)

y = 0.075

Sigmoid

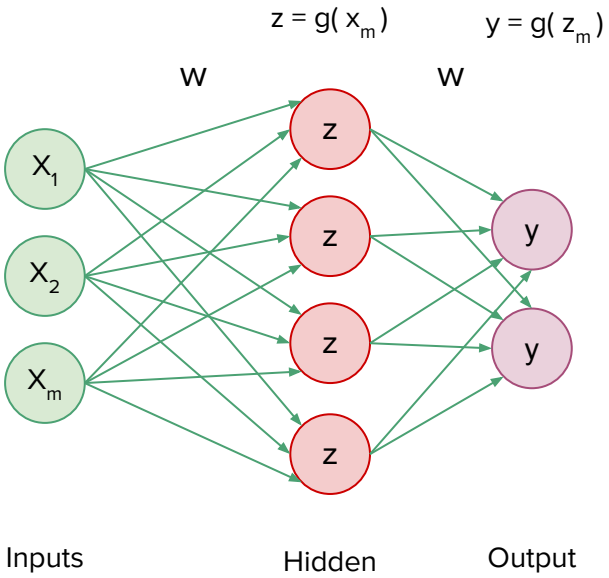
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Values from 0 to 1

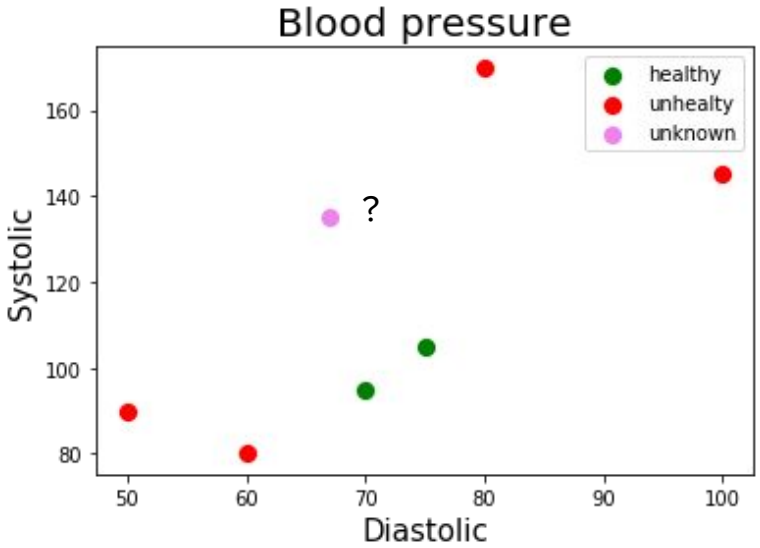
Multi Output Perceptron

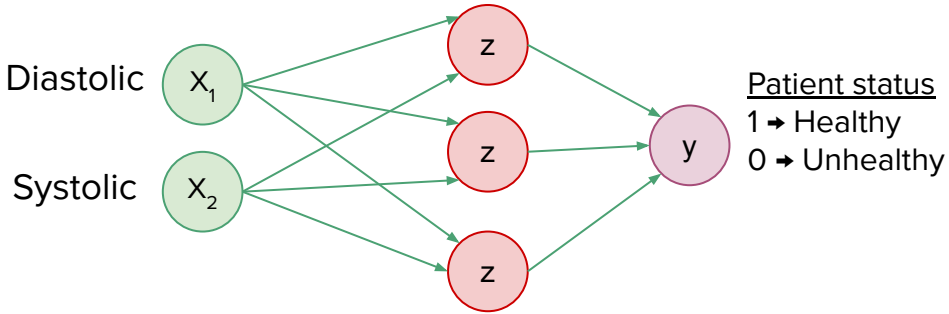
- Inputs densely connected to all outputs ➔ Dense layers



Example problem

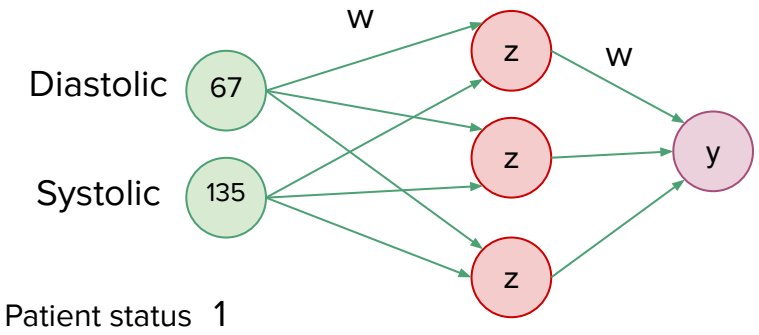
- Is the patient healthy? x = [67, 135]





Loss function

- The loss (L) measures the cost incurred from incorrect predictions



Predicted: 0.1
Actual: 1

W need to change so
L is minimal

$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

Predicted Actual

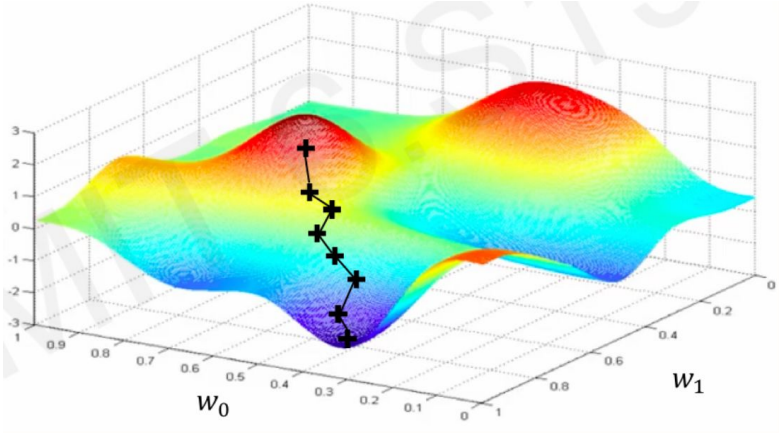
$$f(x) \quad y$$

$$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

Training NN: Loss optimization

We want to change W to achieve lowest loss

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$



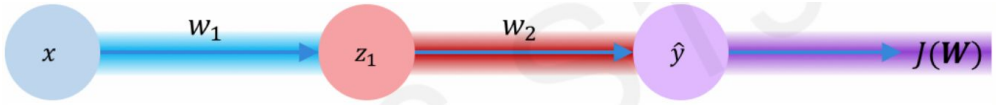
1. Randomly pick initial W
2. Compute gradient
3. Small step opposite of gradient
4. Repeat until convergence

Algorithm

1. Initialize weights randomly

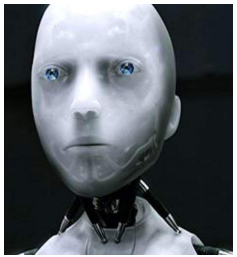
2. Loop until convergence:
   a. Compute gradient
   b. Update weights

3. Return weights



$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

NN are just a function optimization, no AI uprising is possible!



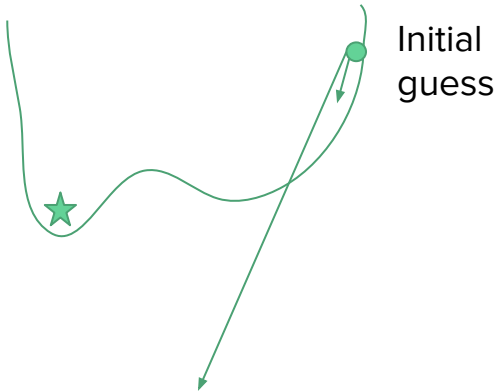
Learning rate

Factor that controls how quickly the model converges

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$$

Most important parameter, controls training:

- Low lr: model doesn't converge
- HIgh lr: model converges too quickly to a local minima



<u>Adaptive learning rates:</u> Lr tuned according to:

- Gradient size
- Size weights
- etc.

Important terminology

Epoch:

One Epoch is when the ENTIRE dataset is passed forward and backward through the neural network. An epoch is comprised of one or more batches.

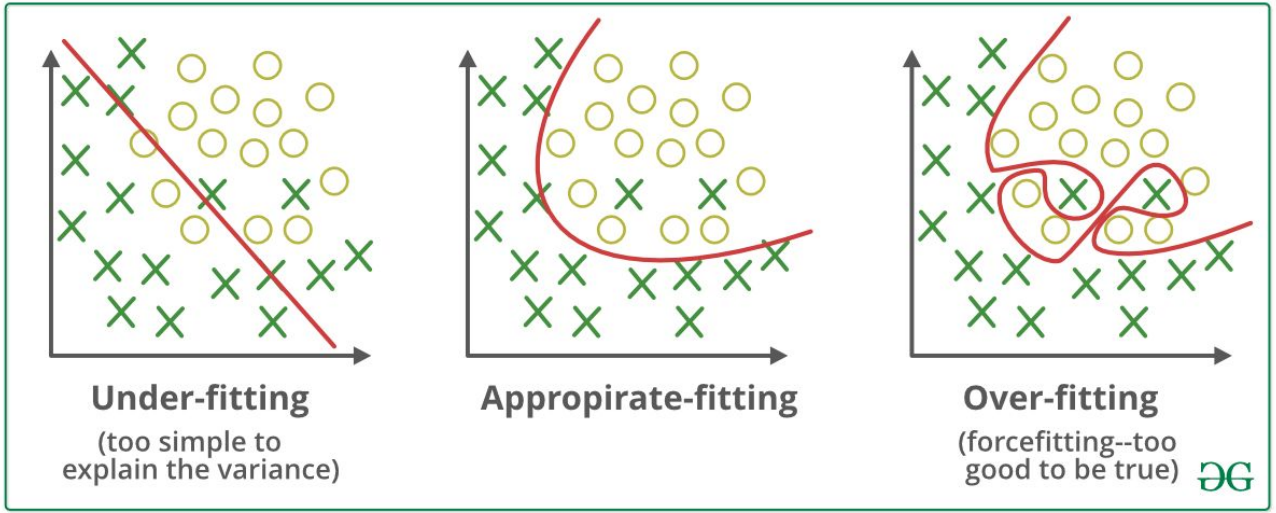
Batch:

Part of the dataset that is passed to the network

Batch size:

Total number of training examples present in a single batch

Overfitting problem

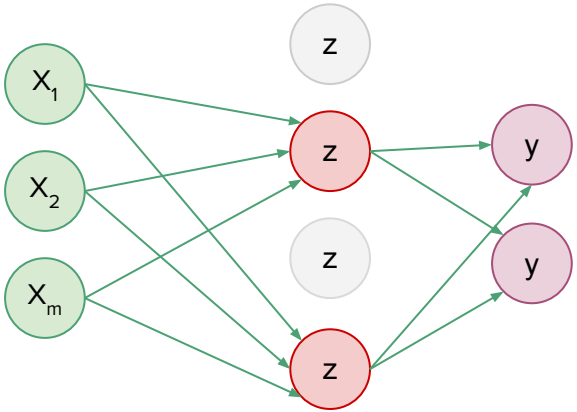


We want our model to be generalistic

New data would confuse the model

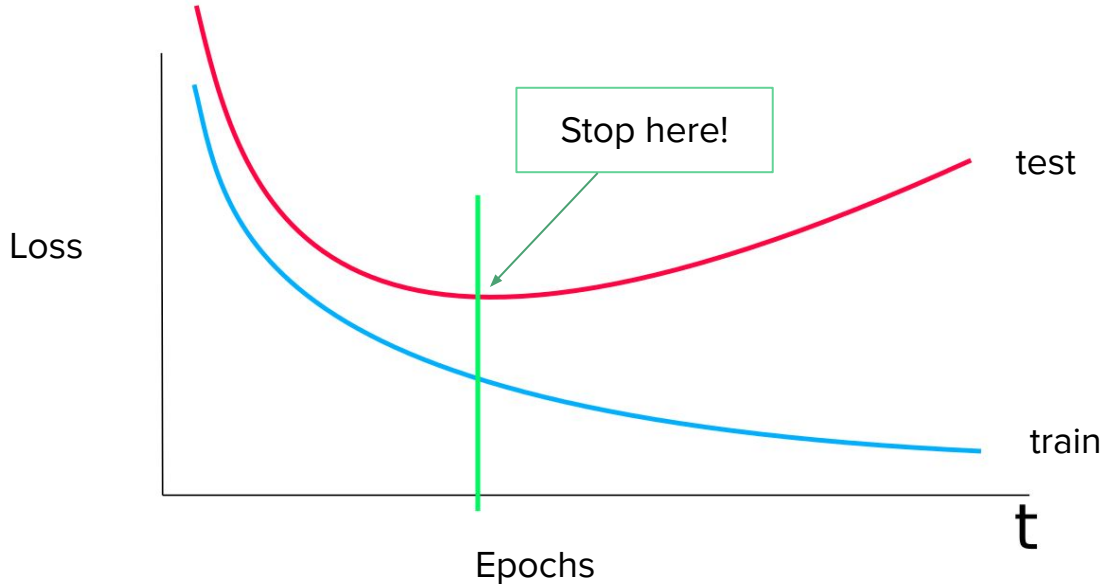
Solution 1: Dropout

- During training, randomly set some activations to 0
- Forces network to not rely on any single node ➔ generalization



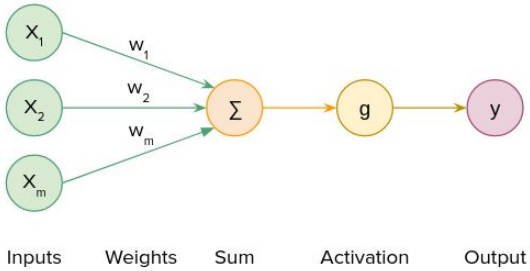
Solution 2: Early Stopping

- Split data into training and test datasets ➜ Test will never update weights
- Stop training before we overfit



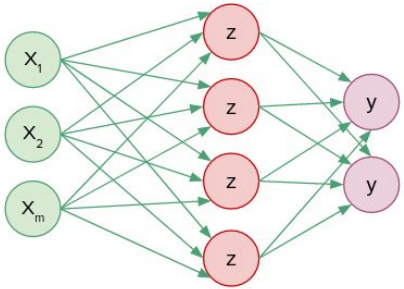
Take home messages

Perceptron



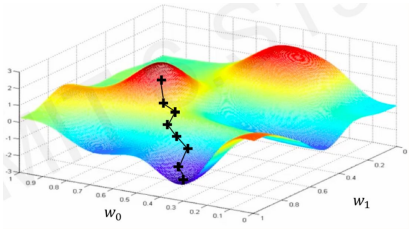
- Building block NN
- Nonlinear activation functions

Neural Networks



- Stack of perceptrons
- Optimize by backpropagation

Training



- Learning Rate
- Batching
- Overfitting