Machine learning

Neural Networks

Wouter
Gevaert

Content

- Introduction
- The artificial neural network
- Getting started with Keras and Tensorflow

Introduction

What is a neural network?

Definition

According to Simon Haykin, Neural Networks - a comprehensive Foundation, Prentice Hall, New Jersey, 2nd edition.

A neural network (NN) is a massively parallel distributed processor that has a natural propensity for malfunctioning experimental knowledge and making it available for use.

It resembles the brain in two aspects:

- Knowledge is acquired through the network through a **learning process**.
- Interneuron connection strengths known as **synaptic weights** are used to store the knowledge.

What is a neural network?

Inspired by the human brain

The human brain excels in tasks such as **pattern recognition**, **speech recognition**, **perception**, and so on.

- As the brain learns more, it learns faster.
- The brain is very strong in parallel processing, a classic computer in serial processing.

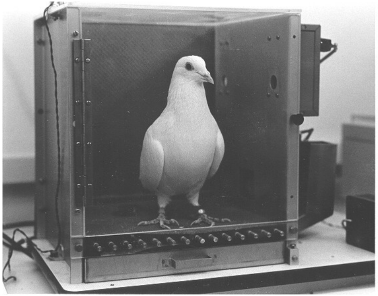
Human brain vs computer

	Brain	Computer
Processing Elements	10^{10} neurons	10^{8} transistors
Element Size	10^{-6} m	10^{-6} m
Energy Use	30 W	30 W (CPU)
Processing Speed	10^{2} Hz	10^{12} Hz
Style Of Computation	Parallel, Distributed	Serial, Centralized
Energetic Efficiency	10^{-16} joules/opn/sec	10^{-6} joules/opn/sec
Fault Tolerant	Yes	No
Learns	Yes	A little

Biological model

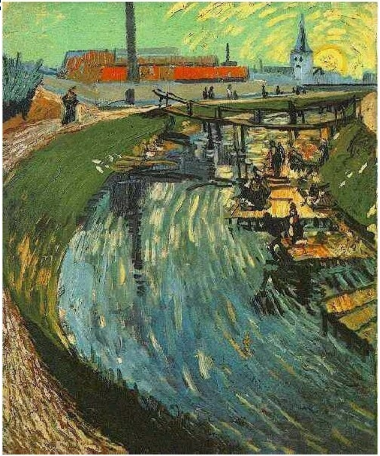
Pigeons as art experts (Watanabe et al. 1995)

- Pigeon in Skinner box.
- Show paintings by two painters (Chagall and Van Gogh).
- Give pigeon a reward for paintings by a certain painter.



Biological model

Pigeons as art experts (Watanabe et al. 1995)



Biological model

Pigeons as art experts (Watanabe et al. 1995)

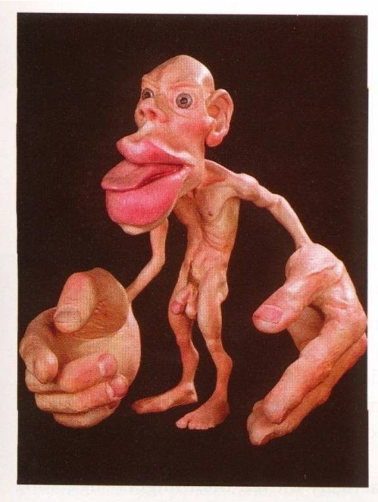
- The pigeons were able to correctly classify the training set with 95%.
- On the test set with unseen paintings they achieved an accuracy score of 85%.
- Pigeons not only memorize images, they recognize patterns.
- They generalize already seen information to base predictions on.

Biological model

See

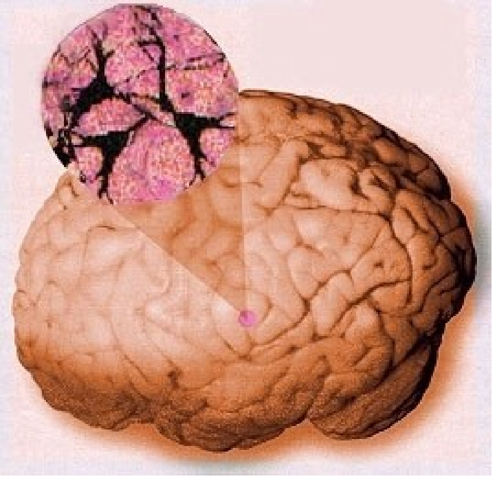
Biological model

Human body scaled to allocated volume of brain



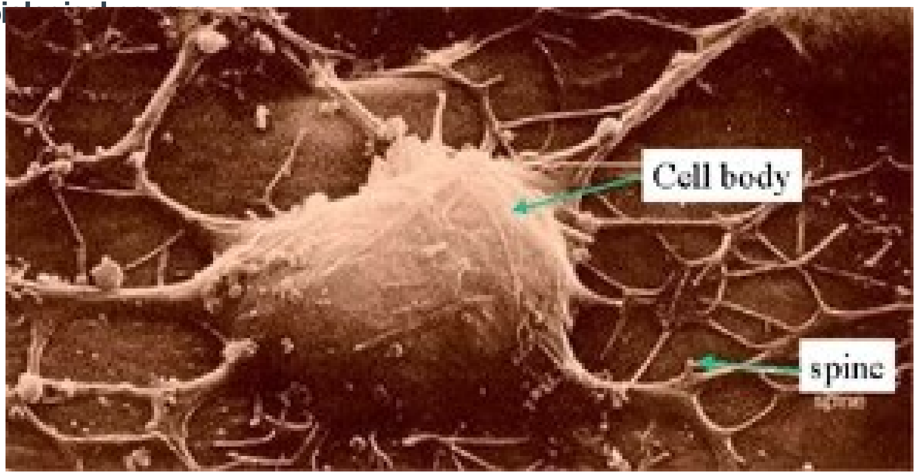
Biological model

The human brain



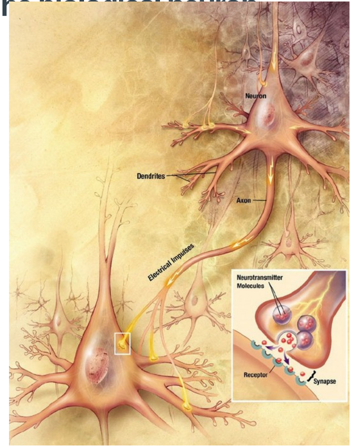
Biological model

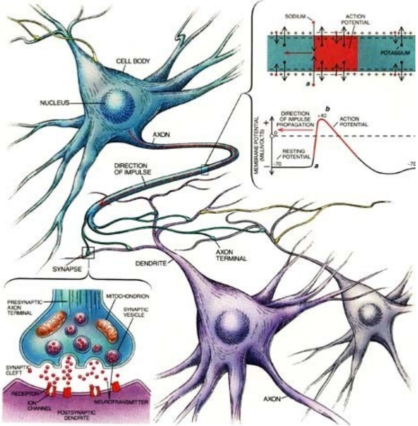
The bi



Biological model

The biological neuron

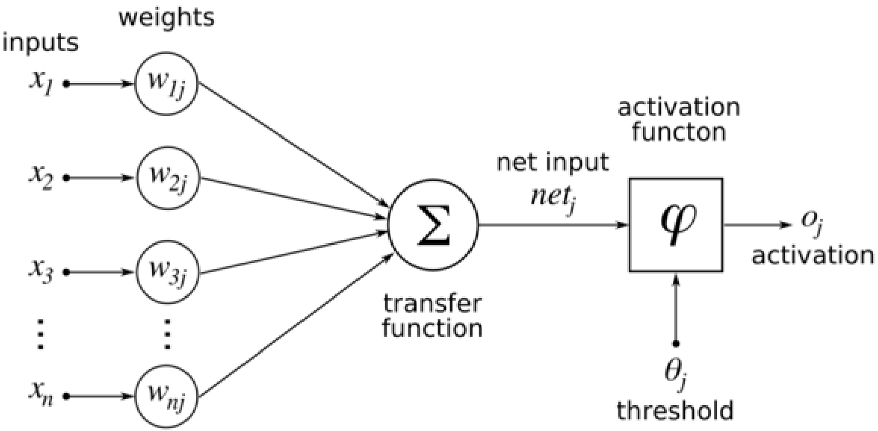




The artificial neural network

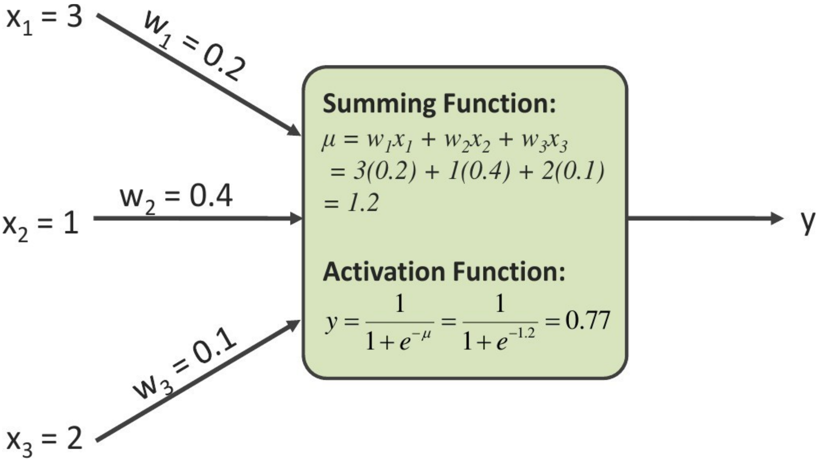
The artificial neural network

Perceptron



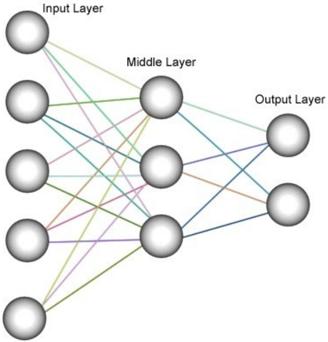
The artificial neural network

Perceptron



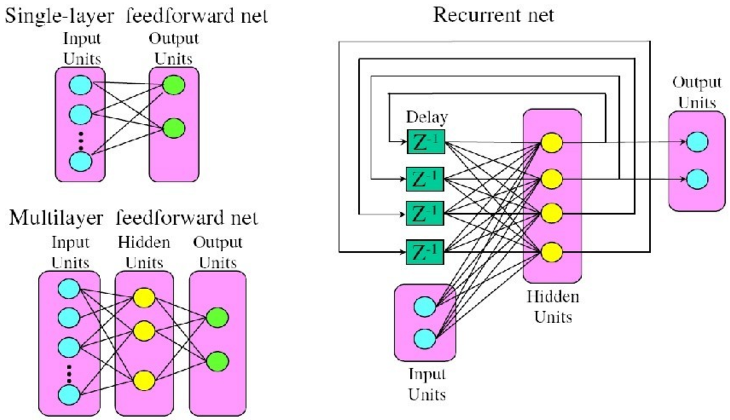
Characteristics of a neural network

- Network architecture
- Learning algorithm
- Activation functions



Network architecture

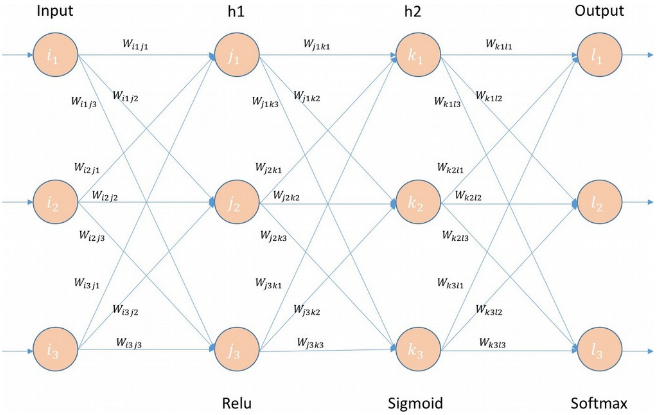
Overview



Meer info: http://www.asimovinstitute.org/neural-network-zoo/

Feedforward neural network

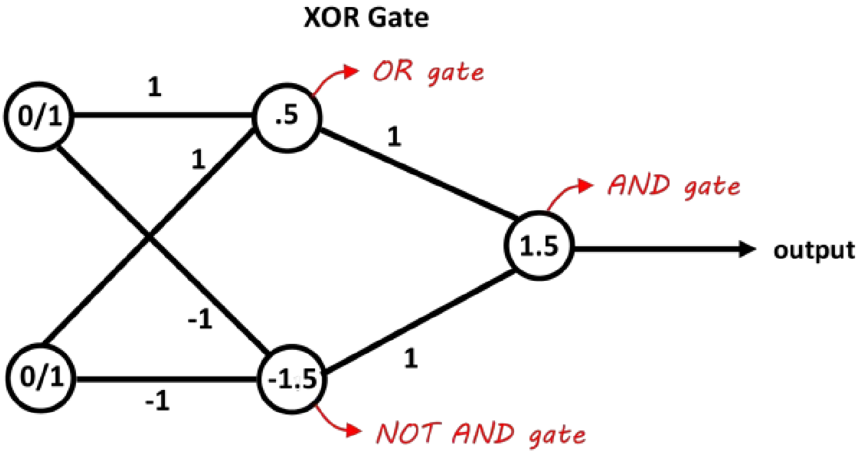
Structure



The information flows one way from input to output.

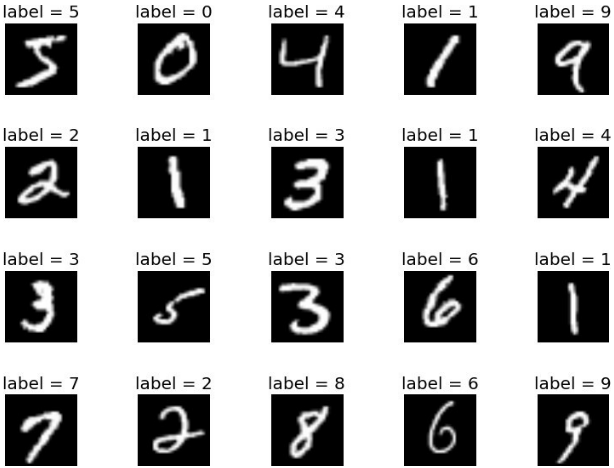
Feedforward neural network

Example: XOR function



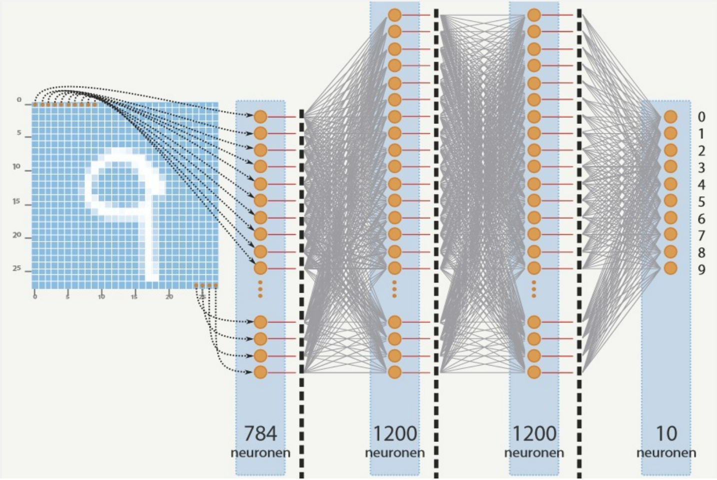
Feedforward neural network

Example MNIST



Feedforward neural network

Example MNIST

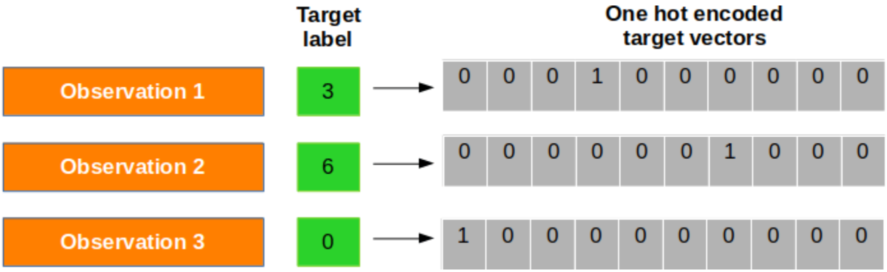


Feedforward neural network

One-hot encoding

One-hot encoding is a way to transform categorical features / targets to a more suitable format.

The index of 1 in a column vector (with only zeros) corresponds to the category of the feature / target.



Feedforward neural network

One-hot encoding

```
# one-hot encoding class labels

from keras.utils import np_utils

y_train[:10]
```

```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

```
y_train_OneHotEncoding = np_utils.to_categorical(y_train)
y_train_OneHotEncoding[:10]
```

```
array([[ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.],
       [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.]])
```

Feedforward neural network

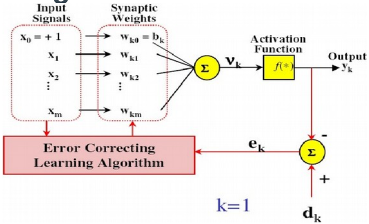
One-hot encoding

```python
from sklearn import preprocessing
print(y_train)

lb = preprocessing.LabelBinarizer()
lb.fit(y_train)
y_train = lb.transform(y_train)

print(y_train)
```

```
[5, 0, 4, 1, 9, 2, 1, 3, 1, 4]
[[0 0 0 0 0 1 0]
 [1 0 0 0 0 0 0]
 [0 0 0 0 1 0 0]
 [0 1 0 0 0 0 0]
 [0 0 0 0 0 0 1]
 [0 0 1 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 0 0 1 0 0 0]
 [0 1 0 0 0 0 0]
 [0 0 0 0 1 0 0]]
```

Feedforward neural network

Backpropagation learning



η is the learning rate

$0 < m\eta < 1$ with m number of inputs

- Error: $e_k(n) = d_k(n) - y_k(n)$
- Minimize the cost based on the current error $e_k(n)$

$$\varepsilon(n) = \frac{1}{2}\sum_k e_k^2(n)$$

- $\Delta W_{kj}(n) = \eta e_k(n) x_j(n)$ with η the learning rate
- Adjust the weight according to :

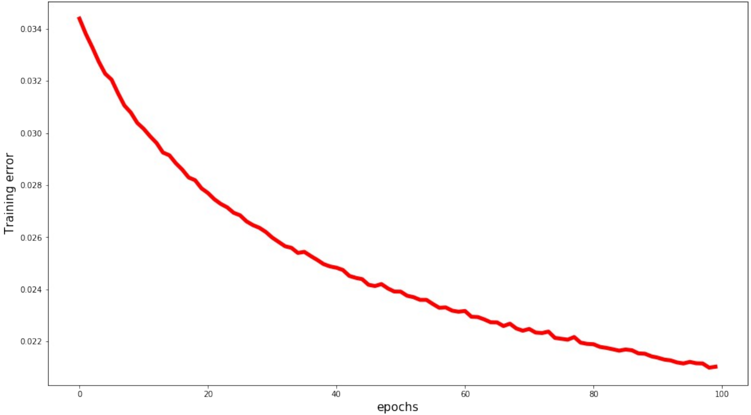
$$\Delta W_{kj}(n+1) = W_{kj}(n) + \Delta W_{kj}(n)$$

Meer info: https://becominghuman.ai/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

Feedforward neural network

Backpropagation learning

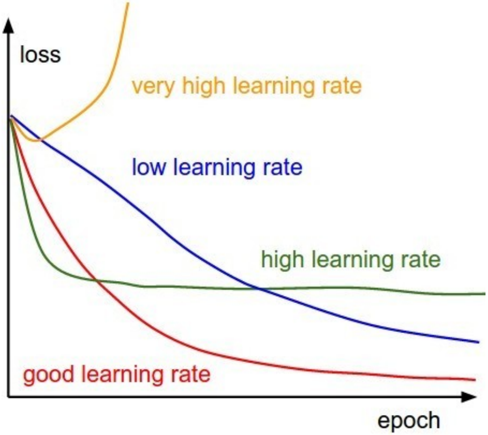
Error function (= loss) in function of number of training epochs



Feedforward neural network

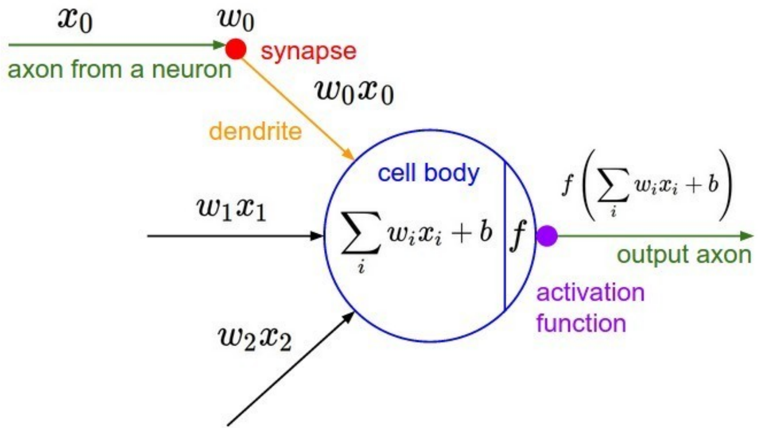
Backpropagation learning

Influence/Impact of the learning rate on the training



Feedforward neural network

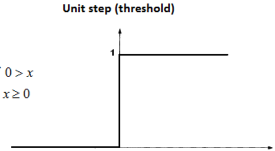
Activation function



Activation functions

Step function

$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



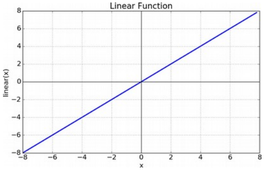
- Output is 1 when the value > 0
- Output is 0 when the value < 0

Disadvantage:

- Can only say yes or no (100% or 0%)
- Problems with multiple classes.
- What if multiple classes are on 1?
- Backpropagation does not work. The derivative =0

Activation functions

Linear function (Adaline)



Output is proportional to the input

Disadvantages:

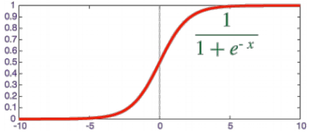
- No matter how many layers you use, the final activation will remain linear.
- The derivative is constant and has no relation with the entrance.

Use:

- Input layer
- Output layer in regression

Activation functions

Sigmoid



Non-linear

- The output is always between 0 and +1

Disadvantages

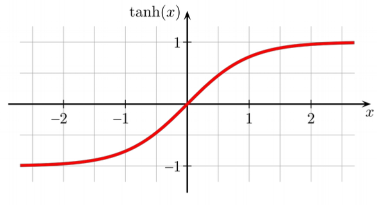
- Vanishing gradient problem.
- Problematic with neural networks with many hidden layers.

Use:

- Not often used anymore.
- Sometimes for output layer at classification problems.

Activation functions

Hyperbolic tangent (tanh)



Non-linear

- The output is always between -1 and +1
- Centered around 0

Disadvantages

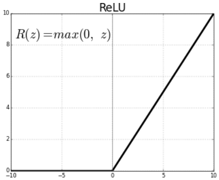
- Vanishing gradient problem remains.

Use:

- Not often used anymore.

Activation functions

Relu = Rectified Linear Unit



- ReLu is non-linear and each function can be approached by combining relay functions.
- Very efficient to computing power.
- Sparse activation. Many activations become 0.

Disadvantages:

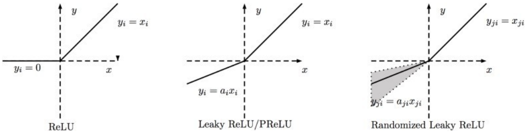
- Dead neurons can no longer be activated.

Use:

- For hidden layers

Activation functions

Leaky Relu



- Variant on ReLu.
- Do not die.

Disadvantages:

- More parameters to train.

Use:

- For hidden layers.

Activation functions

Conclusions

Hidden layers:

- First use ReLu.
- Try Leaky ReLu.
- Do not use Sigmoid or Tanh.

Output layer:

- Linear in regression.
- Softmax / Sigmoid at classification.

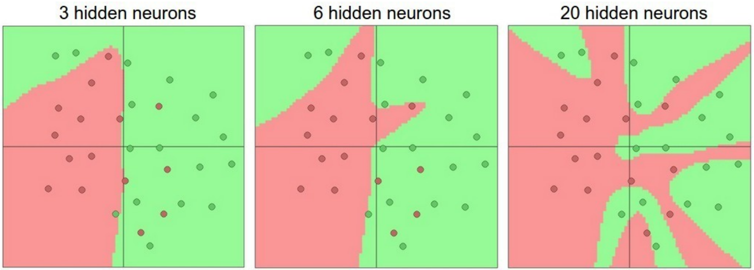
Sofmax is a generalization of the Sigmoid :

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}.$$

Works with mult-class classification.

Underfitting and overfitting

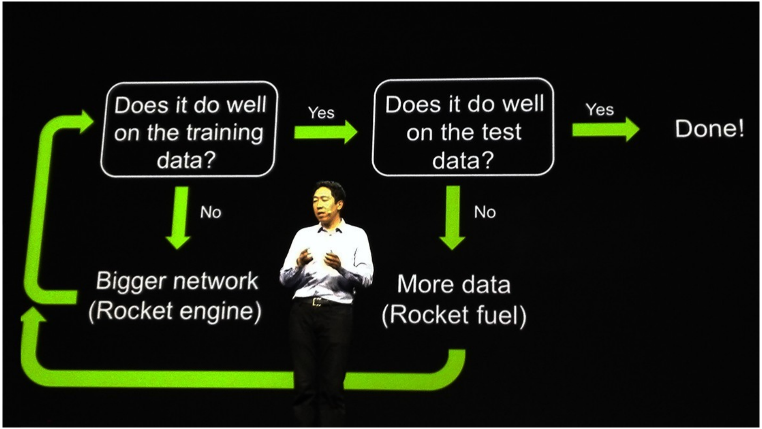
Number and size of the layers



- Too large neural network: overfitting.
- Too small neural network: underfitting.

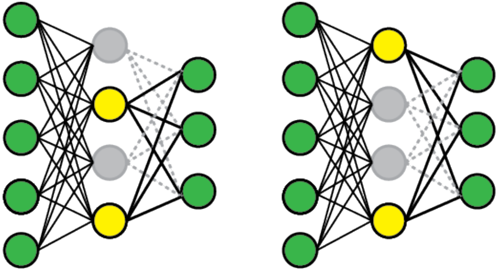
Underfitting and overfitting

Regularization of a neural network



Underfitting and overfitting

Dropou
t



- Technique to prevent overfitting.
- Switch off a certain percentage of the neurons in a layer.
- Other neurons have to indent for neurons that are off.

Getting started with Keras and Tensorflow

Getting started with Keras and Tensorflow

What is Keras?

https://keras.io/

- Neural Network library written in Python.
- Built on top of Tensorflow and Theano.
- Easy to use.Modular and expandable.
- Allows to build complex (deep learning) neural networks.

Getting started with Keras and Tensorflow

Install Keras and Tensorflow

- pip install tensorflow
- pip install keras

Getting started with Keras and Tensorflow

The sequential model

- Allows to stack different layers of a neural network
- https://keras.io/#getting-started-30-seconds-to-keras

Construction of the sequential model:

```
from keras.models import Sequential
model = Sequential()
```

```
from keras.layers import Dense, Activation
model.add(Dense(units=30, input_dim=
10)) model.add(Activation('relu'))
model.add(Dense(units=5))
model.add(Activation('softmax'))
```

10 inputs | 30 hidden layer ReLu units | 5 softmax uitgangen

Getting started with Keras and Tensorflow

The sequential model

Compile and train the sequential model + predictions

```
model.compile(loss='categorical_crossentropy',
optimizer='sgd',
metrics=['accuracy'])

model.compile(loss=keras.losses.categorical_cr
ossentropy,optimizer=keras.optimizers.SGD(lr=0.
01, momentum=0.9,
nesterov=True))

model.fit(x_train,y_train, epochs=5, batch_size=

32)classes=model.predict(x_test,batch_size=

128)
```

Parameters of the sequential model

Activation functions

https://keras.io/activations/

Available activation functions:

- softmax
- relu
- sigmoid
- tanh
- linear
- https://keras.io/layers/advanced-activations/

Parameters of the sequential model

Learning rate optimizers

https://keras.io/optimizers/

- **SGD + Nesterov** (Stochastic Gradient Descent)
- RMSProp: more suitable for recurrent networks
- Adagrad
- **Adam**
- Adamax

Parameters of the sequential model

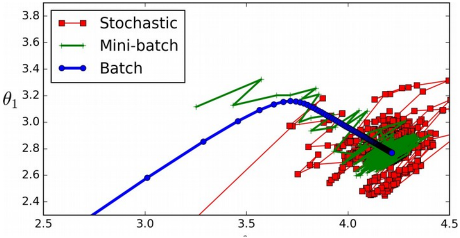
Epochs - batch size – iterations

- Epochs: the number of times the neural network sees the complete training set.
- Iterations: The number of times the weights are adjusted. Is equal to the number of epochs times the number of batches.
- Batch size: The number of samples that the neural network will see before it updates the weights. Updating of the weights is based on the average error of a batch.
- Batch mode: The batch size is equal to the number of training samples.
- Mini-batch mode: The batches are larger than 1 and smaller than the number of training samples.
- Stochastic mode: The batch size = 1. After each training sample there is an update of the weights.

Parameters of the sequential model

Batch size

- Advantages and disadvantages of a small batch size
- Small batches take up less memory.
- The network usually trains faster in small batches.
- Smaller batches give quicker feedback.
- Disadvantage of small batches: less accurate estimation of the gradient. Network stabilizes on the basis of the last training samples.



Parameters of the sequential model

Loss function

https://keras.io/losses/#available-loss-functions

The cost function to be minimized during training.

Error losses: usually used in regression mean_squared_error(y_true, y_pred)

- mean_absolute_error(y_true, y_pred)
- mean_absolute_percentage_error(y_true, y_pred)
- mean_squared_logarithmic_error(y_true, y_pred)

Parameters of the sequential model

Loss function

https://keras.io/losses/#available-loss-functions

Hinge losses: usually used with Support Vector Machines.

- squared_hinge(y_true, y_pred)
- hinge(y_true, y_pred)
- categorical_hinge(y_true, y_pred)

Parameters of the sequential model

Loss function

https://keras.io/losses/#available-loss-functions

Class losses: used in classification.

binary_crossentropy (y_true, y_pred)
For binary classification or multilabel classification: problems where multiple labels can be assigned to a training sample (with sigmoid output layer).
Example a photo contains both a dog and a cat.

categorical_crossentropy (y_true, y_pred)
With multiclass classification with softmax where you want to predict one particular class with high certainty.
Example with the MNIST classification.

Parameters of the sequential model

Metrics

Used to assess the performance of the model.

Regression metrics:

- Mean Squared Error: mean_squared_error of mse
- Mean Absolute Error: mean_absolute_error, mae
- Mean Absolute Percentage Error: mean_absolute_percentage_error, mape
- Cosine Proximity: cosine

Parameters of the sequential model

Metrics

Classification metrics:

- Binary Accuracy: binary_accuracy, acc
- Categorical Accuracy: categorical_accuracy, acc
- Top k Categorical Accuracy: top_k_categorical_accuracy
- Cosine Proximity

Parameters of the sequential model

Other parameters

- **validation_split:** fraction of training data used for validation.
- **sample_weight_mode**: for unbalanced data: some classes are more common in the training set than others.