# *Applications of Artificial Intelligence*

Sebastian Iwanowski
FH Wedel

**Chapter 4:**
Knowledge-Based Systems

4.4: Case-Based Reasoning (Machine learning)

# Case-Based Diagnosis

## Input to knowledge base:

- Cases with complete symptom vector and associated faults (classified unambiguously)

## a) Classical AI, with similarity measure:

- Similarity measure for incomplete symptom vectors (often weighted between different types of symptoms)

## Structure of knowledge base:

- Points in vector space
- Similarity measure

## Job of inference engine:

- For a new vector given, find the most similar symptom vector of the knowledge base.
- Assign the same fault to the new vector as associated to the reference vector in the knowledge base (possibly with a probability value).

# Case-Based Diagnosis

## Input to knowledge base:

- Cases with complete symptom vector and associated faults (classified unambiguously)

## b) with neural networks:

- Neural network with input layer (for symptom vector) and output layer (for faults) and (optionally) intermediate layer of nodes and edges, marked by variable weights.

## Structure of knowledge base:

- Points in vector space
- Neural network with clearly defined weights (dependent on trained symptom vectors and associated faults)

## Job of inference engine:

- Apply new symptom vector to the input layer of the network.
- Read the associated fault from the output layer.

# Case-Based Reasoning (Machine Learning)

## Generalisation of case-based diagnosis to arbitrary case-based reasoning strategies:

### Principle (also called *supervised* learning):

- Given cases as vectors (*complete symptom vectors*): These are “learnt” and build the knowledge base.
- Given new vectors, of which not all parameters are known (*incomplete symptom vectors*): These are to be classified.
- Assign values to the unknown parameters.

### Job of inference engine (simple variant):

- For the new vector, find the closest symptom vector learnt by the knowledge base.
- For the unknown parameters of the new vector, assign the same values as in the associated symptom vector learnt by the knowledge base.

**This variant only makes sense when the unknown values come from a discrete (better finite) domain !**

# Case-Based Reasoning (Machine Learning)

## Improvement for continuous value domains:

### Job of inference engine (better variant):

- For the unknown parameters of the new vector, assign values “in between” values of “nearby” symptom vectors learnt by the knowledge base.

### Other mathematical formulation of this method:

- Consider the unknown parameters of the new vectors as function values of the known parameters: Find a continuous function where all vectors learnt by the knowledge base are contained.
- Of this function, assign the function values of the known parameters to the unknown parameters.

**Query:** **How do we get an appropriate function for a given set of reference vectors?**

**Answer:**

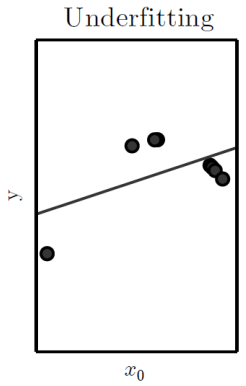
- Take a class of functions, each function differing by certain parameters.
- Determine the parameters solving an equation system obtained from the known reference vectors.

# Case-Based Reasoning (Machine Learning)

## Distinguish between training errors and test errors:

Fixing the polynomial degree for the classification function f will make it impossible that all training examples solve the function f correctly.

➔ **training error**



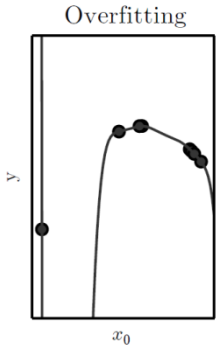

Optimisation goal: Make the training error as small as possible.

Taking a too small degree for the classification function causes **underfitting**:

Once, a classification function is chosen,
this function will classify the test examples.

➔ **test error**

Optimisation goal: Make the test error not much bigger than the training error,
i.e. minimise the expected difference between test error and training error.




Taking a too high degree for the classification function causes **overfitting**:

graphics from deeplearning book, Goodfellow et al.

# Case-Based Reasoning (Machine Learning)

## Determining parameters in function classes (regression):

### Linear regression:

- Find the weights in a linear function of the form: $f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i x_i$

### Generalisation:

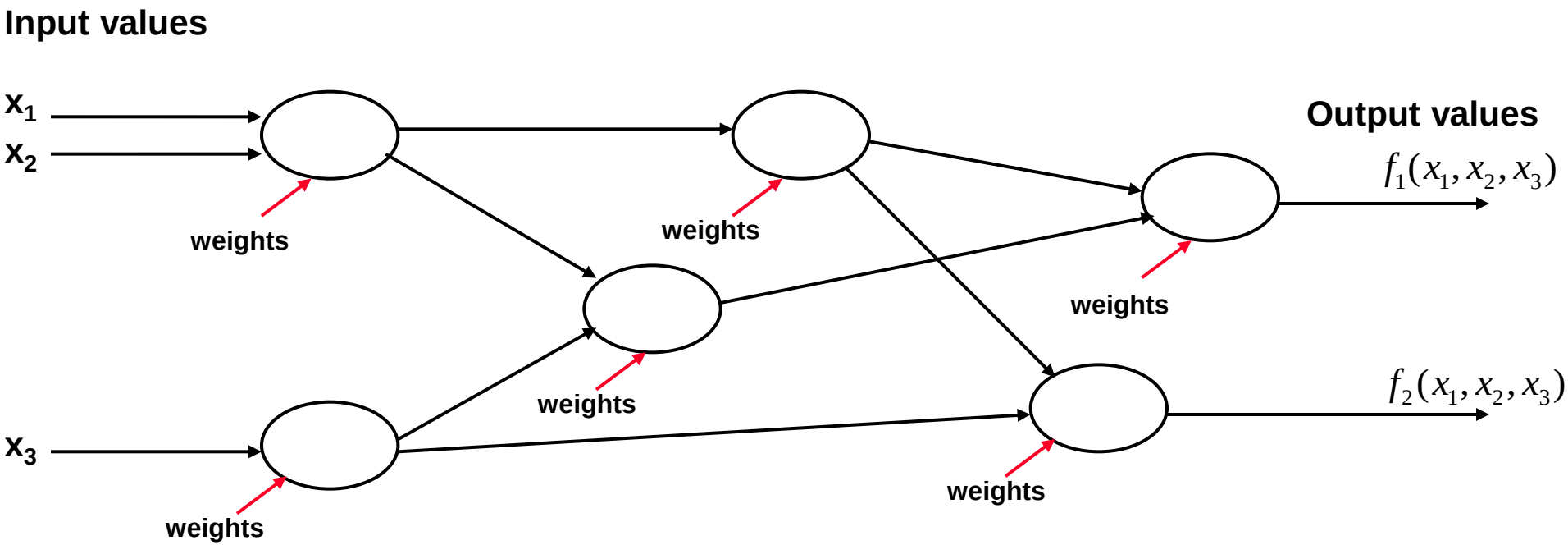
1. Find the weights in a linear equation **system**.
2. Find the weights in equation systems of higher order.
3. Find the weights in parametrised inequality systems.

- **Case-based reasoning is designed for systems which can*not* be modeled easily.**
- **This is why a higher order equation system does not make sense.**
- **It is better to work with many weakly connected equation systems and distribute the unknown knowledge.**

# Neural Networks

## Idea of neural networks:

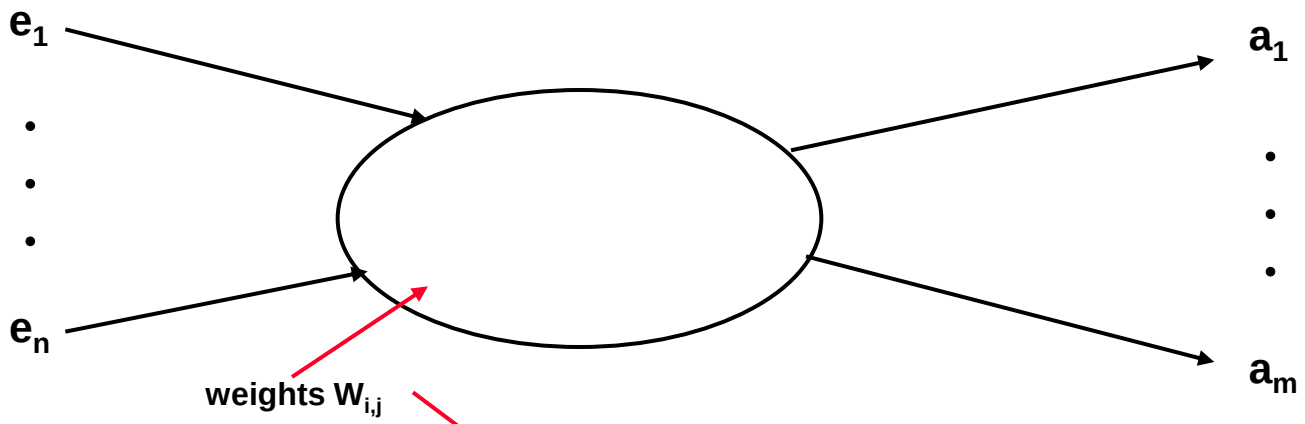
**Given a multi-valued function $f$ (notation: $f_i(x_1, x_2, \ldots, x_n)$)**




- The weights may be preset but are adapted to the examples learnt.
- Function values of new inputs are obtained applying the neural network.

# Neural Networks

## Functionality of a single neuron:

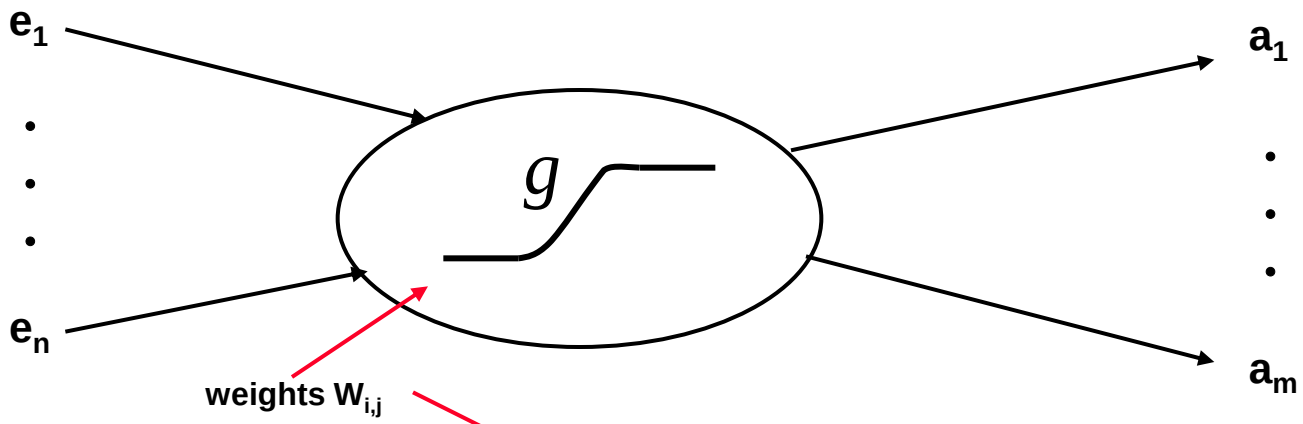



$$a_i(e_1, e_2, \ldots, e_n) = \sum_{j=1}^{n} w_{i,j} \cdot e_j$$

- A = ($a_1$, …, $a_m$) is a linear function in the e's. A is represented by an (m x n)-matrix of the $w_{ij}$'s.

# Neural Networks

## Functionality of a single neuron:




$$a_i(e_1, e_2, \ldots, e_n) = g\left(\sum_{j=1}^{n} w_{i,j} \cdot e_j\right)$$

- A = ($a_1$, …, $a_m$) is a linear function in the e's. A is represented by an (m x n)-matrix of the $w_{ij}$'s.
- g is a generalised threshold function which is the same for all outputs of the same neuron.

# Neural Networks

## Different layered neural networks:

### In standard neural networks, all neurons are placed on certain layers:

- There is an input neuron for each input variable and an output neuron for each output variable. Thus, the input neurons represent linear (1,n) functions and the output neurons linear (n,1) functions.
- Neurons on the same layer have the same distance to input and output.
- Links are only existing between neurons of adjacent layers.
- By default, all neurons of a layer are connected to all neurons of the adjacent layers (one in input, one in output direction). But this may change even dynamically during the learning process.

### Neural networks without intermediate layers:

- Neurons of the first layer accepting the inputs are connected to neurons of the second layer providing the outputs.

### Neural networks with intermediate layers (*deep learning* approach)

- Input and output layers are connected by further “hidden” intermediate layers.
- The term “deep learning” is usually only applied when there are at least two hidden layers.

### Neural networks with feedback (Recurrent Neural Networks, RNN):

- Generation of “memory”

# Neural Networks

## Basic technique for adjusting the weights (modern approach):

**Initialisation of the weights of all neurons:**

- Principally, arbitrary weights may be chosen.
- There may be specific initialisation heuristics for special types of networks.

**For each (input, output) training sample:**

- Forward propagation from input to output
- Comparison between predicted values and real values at the output
- For the error estimate (called „loss function“), different methods may be possible:
  e.g. means-squared, cross-entropy
- Backward propagation from output to input: Adjusting the weights considering this sample

**„Backpropagation algorithm“ (Rumelhart, 1986)**

Details and examples (including implementation) in:
Erik Genthe, „ Backpropagation in neural networks – explained at examples“ (in German), FH Wedel seminar presentation, SS 2020, https://intern.fh-wedel.de/fileadmin/mitarbeiter/iw/Lehrveranstaltungen/2020SS/Seminar/Ausarbeitung3BackpropagationGenthe.pdf

# Neural Networks

## Basic technique for adjusting the weights (modern approach):

### Forward and backward propagation in detail:

#### Forward propagation from input to output:

- The input values are attached to the input layer.
- The output of each neuron is computed with the so far used weights successively (in the order how far the neuron is apart from the input).
- This is repeated until the output layer has got all values.

#### Backward propagation from output to input:

- The output layer values are compared with the real output values of the sample, and the output error is computed **using the predefined loss function**.
- For each neuron directly connected to the output layer, it is computed how much this contributed to the output error. The more it contributed, the more the corresponding weight is changed.
- This is repeated towards the input layer, i.e. for each input $x_i$ of the latest layer considered (which is the output of the previous layer considered), it is computed how much each neuron of the previous layer contributed to $x_i$.
- **Note**: The error should not be corrected to zero (danger of overfitting). This is compensated by a **learning rate factor** <<1 which has to be multiplied with the optimum corrector.
- **Important:** There may be *several* ways to compute the amount of contribution of a neuron.

# Neural Networks

## Basic technique for adjusting the weights (modern approach):

### How to perform backward propagation:

#### Gradient descent method (the most popular example for backward propagation):

- The impact of a weight to an error is computed by the partial derivative w.r.t. to this weight of the error function. The derivative of the function of a neuron corresponds linearily to the input by which the weight is multiplied, but the error function may be nonlinear.
- The easiest formulae describe the impact of the weight of the output layer. Then the partial derivative is rather simple.
- The impact of the weight of earlier layers is obtained by plugging in the dependency of later values from previous weights. Then the derivative becomes more complicated and is highly nonlinear due to the chain rule involving the weights of later neurons.
- If an activation function is involved in a neuron, this must also be considered in the descent function. Then the derivative of the activation function must be considered as well.
  Note: Typically, the activation function is not linear but of higher order or even exponential.
- Notation: The derivative of a multidimensional function w.r.t. all involved variables,
  is called the **Nabla ∇ operator** consisting of all single partial derivatives.
- If the function is multi-valued, the Nabla operator is a matrix consisting of the partial derivatives.

Detailed desription with example also in:
Michel Belde, Bachelor thesis on „ Improvement of a consulting app for the sales department using image recognition“, chapter 3.1, on class website (in German)

# Neural Networks

## Basic technique for adjusting the weights (modern approach):

### Improvements for Backpropagation: Batches

- Consider a set of several input/output samples and feed them together in the network.
- The weights are then adjusted such that the average error is minimised.

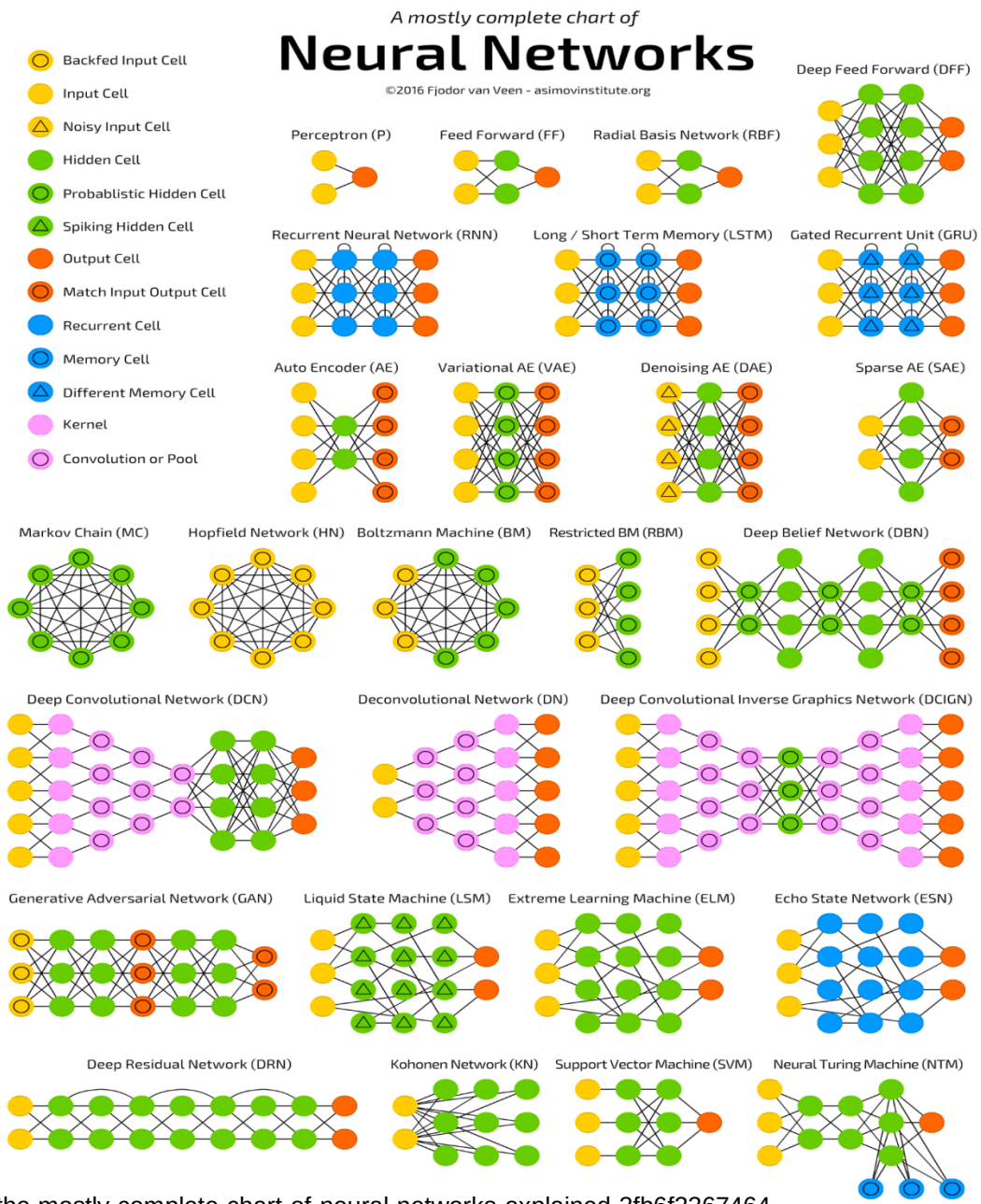
### Problems with backpropagation in deep networks:

#### Vanishing / exploding gradient problem:

- If gradients are close to zero, they accumulate in deep networks and vanish until input.
- This is mainly due to unfortunate activation functions.
- Deep networks need activation functions with a derivative not close to zero in all of the domain.
- If gradients are considerably higher than one, they accumulate in deep networks and may explode.
- Deep networks need activation functions with a derivative not much higher than one in all of the domain.

More details in:
Jacob Hansen, „ Details of the backpropagation algorithm“, FH Wedel seminar presentation, SS 2019, http://intern.fh-wedel.de/fileadmin/mitarbeiter/iw/Lehrveranstaltungen/2019SS/Seminar/Ausarbeitung1_Backpropagation_Hansen.pdf

Dennis Maas, „Specific methods for training and evaluation of deep neuronal nets”, FH Wedel seminar presentation, SS 2019 (in German),
http://intern.fh-wedel.de/fileadmin/mitarbeiter/iw/Lehrveranstaltungen/2019SS/Seminar/Ausarbeitung3_TiefeNetze_Maas.pdf

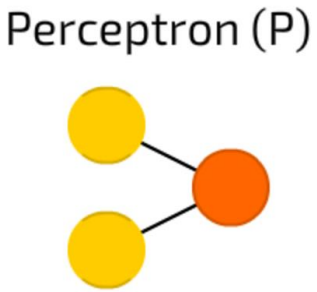



https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

# Types of Neural Networks

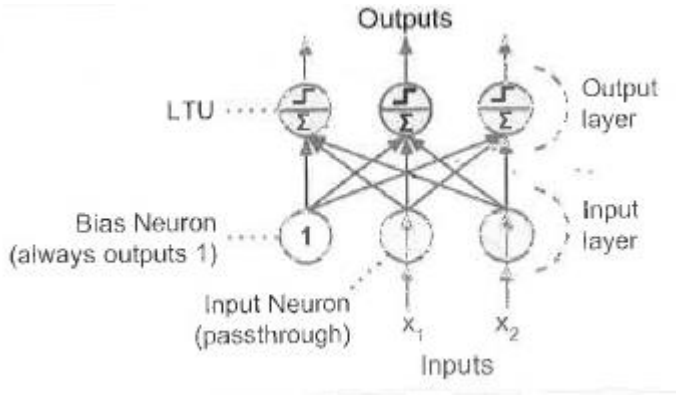
## The classical one: Perceptron (1957, Frank Rosenblatt)

- corresponds to a single regression function.
  The weights are in the individual input neurons.
- Unique output neuron has a step function only, no further weight.
- cannot compute XOR function (Minsky 1971).
- can be generalised for an n-m function having several outputs.




https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Aurelien Geron: Hands-On Machine Learning with SciKit Learn & Tensorflow, O'Reilly 2017



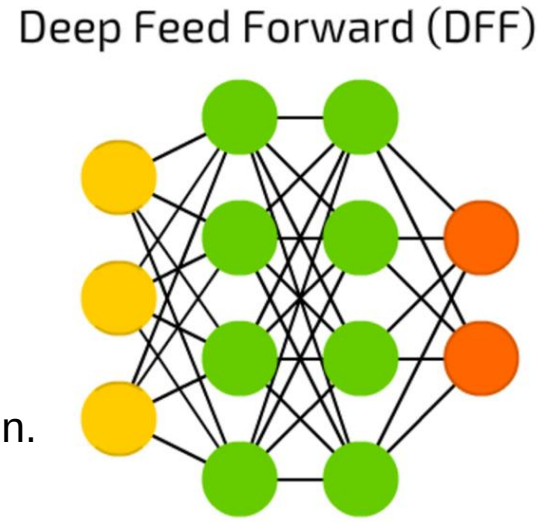

## Weight adjusting technique:

- does not apply backpropagation algorithm as described before.
- Applies Hebb's rule instead: Cells that fire together, wire together.

# Types of Neural Networks

## Deep Feed Forward Network

- uses backpropagation algorithm as described before (proposed by Rumelhart 1986).
- can compute all logical functions.
- differs from Perceptron not only by internal layers, but also by different activation functions than the step function.
- is nowadays used in a lot of standard learning settings.

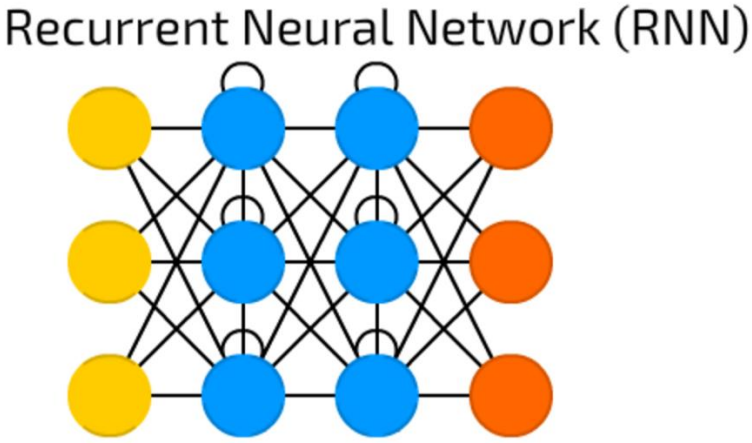



https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

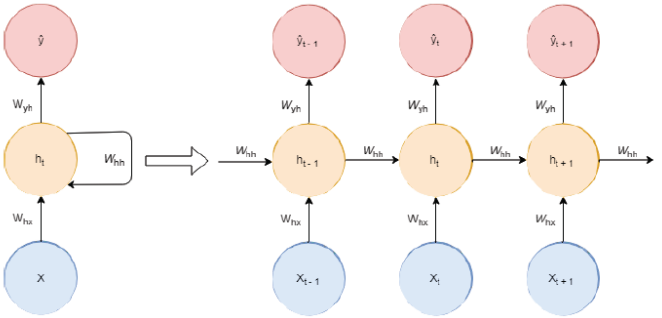
# Types of Neural Networks

## Recurrent Neural Network

- network with “memory”
- Recurrent neurons receive their own output as an input with a delay.
- suitable for context dependent input: In which order did data occur?
- With the use of simple neurons one can store the order only, by more complicated “memory cells” one can store past information for a given time.
- Rumelhart’s backpropagation algorithm can be adapted when the network is unfolded for several time stamps (only discrete time possible): *backpropagation-through-time* algorithm




https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

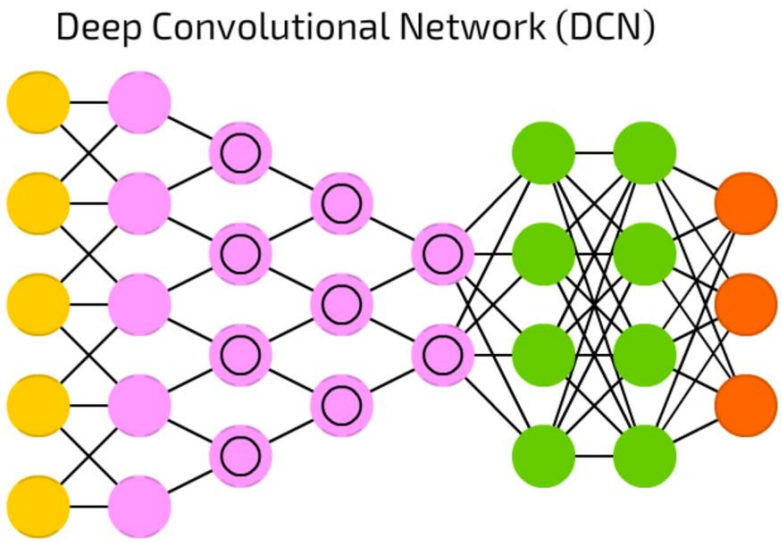


More details in:
Marcello Attila Messina, „Weight adjustment in neural networks with memory”, FH Wedel seminar presentation, SS 2019 (in German),
http://intern.fh-wedel.de/fileadmin/mitarbeiter/iw/Lehrveranstaltungen/2019SS/Seminar/Ausarbeitung5_RNN_Messina.pdf

# Types of Neural Networks

## Deep Convolutional Network

- After the (pink) “convolutional layers”, the (circled) “pooling layers” extract the important features and neglect unimportant ones.
- Unlike in other internal layers, neurons of pooling layers are not connected to all neurons of the adjacent layers.
- This is the modern standard for image recognition and classification.

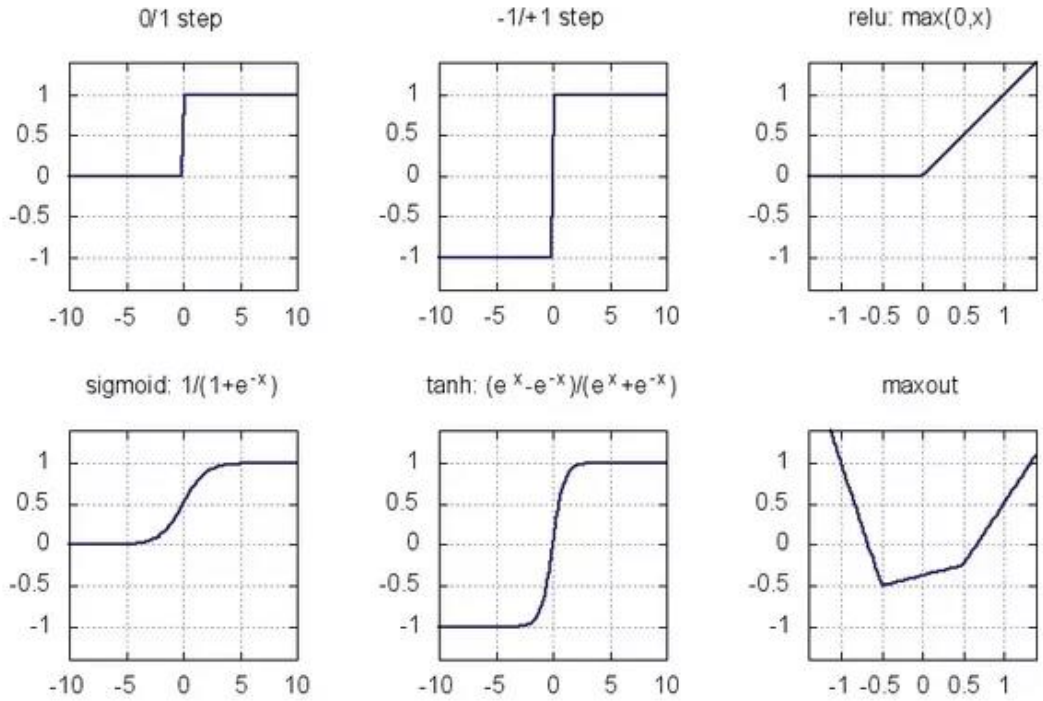



https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

# Neural Networks Techniques

## Different Activation Functions

- The activation function of a neuron gets the input to that neuron and transforms this into a new value which is really fed into that neuron (applying the weight function).
- It is customary to choose the same activation function for all neurons of a given network.



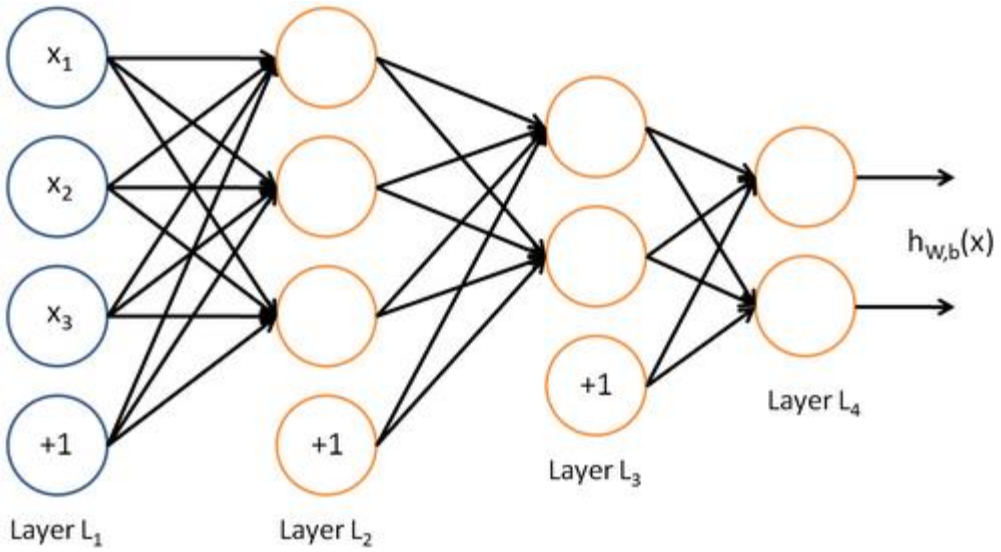

From: Thimo Tollmien, Master thesis on „Optimisation of delay predictions with deep learning“, FH Wedel 2018

# Neural Networks Techniques

## Special Improvement Techniques

### Improvement by bias nodes

- Bias nodes are additional input nodes for each layer of the network. They are not connected with the previous layer. So they feed in extra information.
- Usually they feed in an extra “1” per layer.
- This may help weight adjustment in the training period.



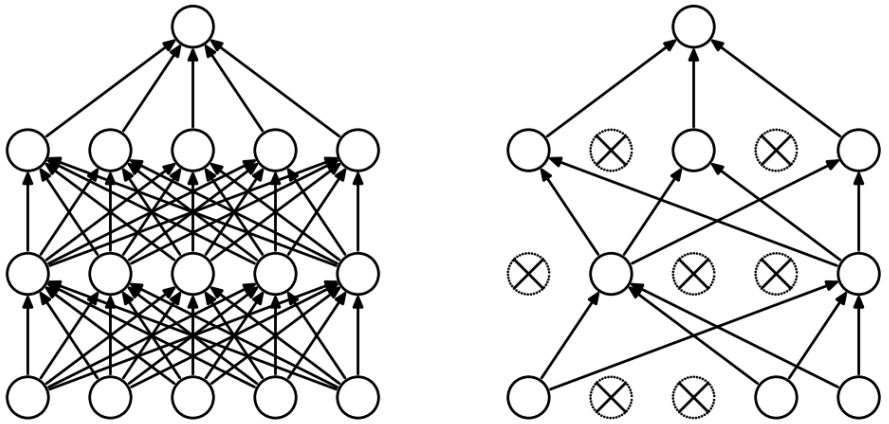

https://www.quora.com/What-is-bias-in-artificial-neural-network

# Neural Networks Techniques

## Special Improvement Techniques

### Dropout

- avoids overfitting
- Input vectors are fed into the network several times
- Each time a different set of randomly chosen connection is cut.
- If the avoidance of a connection does not increase the training error, then this connection will be ommitted in the future.



(a) Standard Neural Net (b) After applying dropout.

From: Thimo Tollmien, Master thesis on „Optimisation of delay predictions in public transportation with deep learning“, FH Wedel 2018

# Neural Networks Techniques

## Training Techniques

**The quality of a trained neural network decisively depends on the quality of input samples and features it was trained with.**

### Cross validation

- Split the training set into two (or more) parts: Train only with one part and test the result with the other. Do the same vice versa.

### Feature selection

- Add or remove certain features (dimensions) of the training set.

# Machine learning without known results

## Data mining (unsupervised learning)

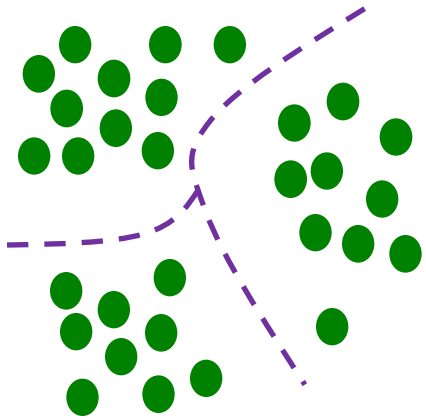
Given some data input. Is it possible to classify this input into different categories?

**Different classification methods:**

- Cluster detection
- Greatest gap detection

**Common clustering techniques:**

- nearest neighbor
- k-means (for k clusters)



From: Dirk Lützelberger, NXP, Colloquium talk on Machine Learning FH Wedel 2018

Data mining is an own field with a rich variety of classification techniques. **„Big data“**

***Learning*** in this context means: Continuous adaptation to changing data

# Machine learning without known results

## Data mining (unsupervised learning)

Given some data input. Is it possible to classify this input into different categories?

**k-means details (most primitive method):**

1) Decide for a certain number k and keep it fixed.
2) Select k arbitrary clusters and compute their means.
3) Determine the variance (sum of the square distances to the mean) for the clusters.
4) Reshuffle the k clusters, and repeat Step 3). Store the clusters with the respective minimum.
5) Repeat Steps 3) and 4) until the minimum does not “considerably” improve.

**There are a lot of refinements for this method.**

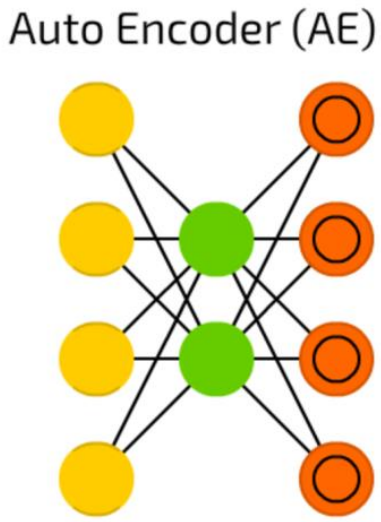
Note: Due to Step 1), this method does not help to detect the „optimum“ k.

Remark: The problem „Find the best arrangement of k clusters such that the distance to the respective means is minimised“ is NP-hard (i.e. not likely to be computable efficiently)

# Types of Neural Networks

## Autoencoder

- suitable for unsupervised learning
- Input and output layers should have the same number of neurons.
- Hidden layers should have fewer neurons than input und output layers.




https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

# Machine Learning in Practice

## Graduation theses in Machine Learning supervised by iw

Tjark Smalla (WS 2016): Implementation of a Neural Network for Order Prediction and Comparison with an Existing Logistic Regression
**Otto GmbH**

Bronislav Koch (WS 2017): Determination of a set of clients with maximum probability for project success in a multivariate model,
**Sven Mahn IT GmbH**

Lasse Karls (WS 2017): Graph-based feature extraction to improve machine learning in predicting the business affiliation of a Signal Iduna customer,
**Signal Iduna**

Thimo Tollmien (SS 2018): Optimizations of Delay Predictions in Local Public Transport Using Deep Learning,
Master thesis in cooperation with **HBT**

Michel Belde (WS 2018): Improvement of a consulting app for the sales department using image recognition,
**akquinet Engineering GmbH**

Dennis Maas (SS 2019): Transformation invariant bar code recognition using neural networks,
**Opus//G GmbH**

# Machine Learning in Practice

## Graduation theses in Machine Learning supervised by iw

Henning Brandt (WS 2019): *Implementation of a model for calculating the concentration of volatile organic compounds in a multi-capillary gas chromatograph using machine learning*,
**bentekk GmbH / Dräger AG**

Linus Stenzel (WS 2019): *Development of an artificial intelligence with human play style in the game Canasta*,
**LITE Games GmbH**

Frederik Schnoege (SS 2020): *Use of natural language processing in IT support,*
Master thesis in cooperation with **Beiersdorf Shared Services GmbH**

Vincent Grohne (WS 2020): *Comparison of different machine learning models for the detection of potentially failed securities deliveries,*
**Berenberg Gossler KG (Bank)**

Ines Kemsies (WS 2021): *Prediction of system failures by using a recurrent neural network,*
**Akquinet engineering GmbH**

This is 47% of all theses I supervised during this period!

# Applying Neural Networks in Practice

## Software Kits

**SciKit Learn (http://scikit-learn.org)**

- open-source library implemented in Python

**Tensorflow (http://tensorflow.org)**

- open-source library implemented in Python und C++
- developed at Google, used in internal software

```
1 model.add(layers.Dense(200, activation="sigmoid", input_shape=(312,)))
2 model.add(layers.Dropout(0.1, noise_shape=None, seed=None))
```

From: Thimo Tollmien, Master thesis on „Optimisation of delay predictions in public transportation with deep learning“, FH Wedel 2018

# Neural Networks

## References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville: *Deep Learning*. MIT Press, 2016.
http://www.deeplearningbook.org

Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn & TensorFlow*. O'Reilly, 2017.

# Other modern ML techniques

## Support Vector Machines

- applies high dimensional vector algebra
- tries to find separating hyperplanes between different sets of classification
- may be regarded as follow-up of the classical approach

**Literature: (available in our library in 8.5.3)**

Nello Christianini / John Shawe-Taylor:: *An Introduction to Support Vector Machines*. Cambridge Univ. Press, 2000 (2006).

Vojislav Kecman: *Learning and Soft Computing*. MIT Pr. 2001

# Case-Based Reasoning (Machine Learning)

## What is the crucial difference between neural networks and „classical“ CBR systems?

- **Neural networks distribute the knowledge about the cases learnt.**

### Theoretical advantages of distribution:

- Arbitrariness of function class chosen does not play such an important role.
- Intransparent cases are handled by an intransparent method:
  The distributed method is “self-adjusting”.

### Practice shows:

- Good neural networks need fewer training cases than classical CBR systems.
- Neural networks provide better classification results.
- **Deep networks made a great boost to AI in general**: They are applied in products already.

# Summary: Case-Based Reasoning (Machine Learning)

**Let‘s get back to our sample application diagnoses for comparing different KBR methods:**

## Advantages and Disadvantages:

- **The method is simple.**

\+ - The diagnosis of the run time component is very fast.

\+ - Knowledge acquisition can easily be automatised.

– - The knowledge base can only be generated for systems where experience is given.

– - The knowledge base consumes a lot of storage (classical approach only).

# Summary: Case-Based Reasoning (Machine Learning)

## Advantages and Disadvantages:

- **The knowledge base does not contain any other structural knowledge than the similarity measure or the NN.**

| | | |
|---|---|---|
| + | - | All application domains are equally suited. |
| + | - | The same inference engine may be applied for totally different application domains. |
| + | - | For systems *without a reasonable model*, classification results are rather good (at least for neural networks). |
| – | - | Even with a small change of the system, the knowledge base cannot be used reliably. |
| – | - | Similarity measure and neural network are arbitrary. |
| – | - | Each run time diagnosis may be wrong. |
| – | - | **The result is not justifyable (at least for neural networks): Distrust "algorithms".** |

In this context, „algorithms" denote statistical algorithms for backpropagation plus the network design.

This is not to be confused with algorithms outside AI or algorithms for symbolic (rule-based) AI.