Advanced Artificial Intelligence

Machine learning

Machine Learning vs Data Mining

- Machine learning
  - The algorithms (that can be used for data mining)
  - Development, refinement, adaptation of ML algorithms
- Data mining
  - The application of a machine learning algorithms with a specific purpose
  - Data mining...is the process of discovering valuable and hidden knowledge in multidimensional data sets that you never knew existed
  - is a largely automated process
    - Due to machine learning algorithms
  - Encompasses knowledge discovery (description) and prediction

Mitchell’s definition of learning (1997)

“A computer program is said to ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”*

Hand et al.’s definition of data mining (2001)

“… the analysis of … observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.”

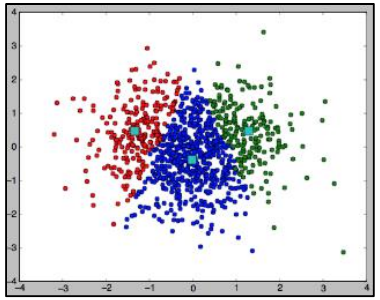
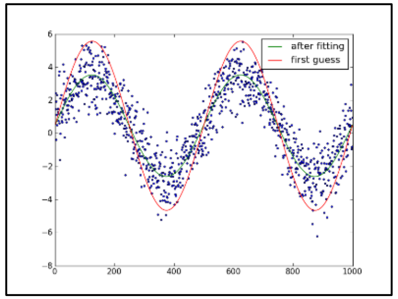
Machine Learning

- In a conventional programmed an algorithm gives a step by step description of the what has to be done
  - All possible alternatives need to be considered (e.g. if-then-else, case)

- Machine Learning
  - Algorithms that figures out the sequence of steps to take by itself



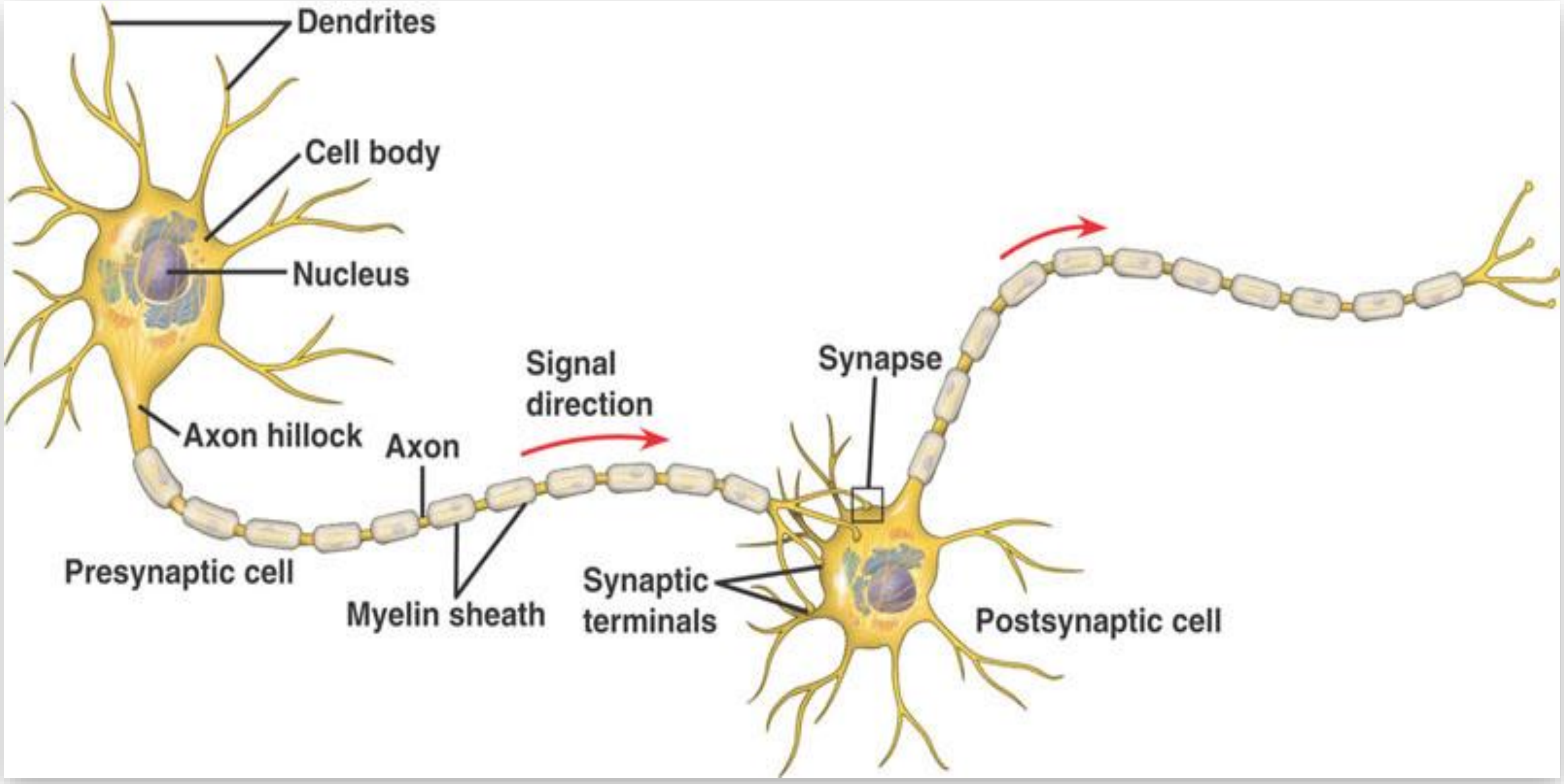
  - Find patterns, trends in data based on statistics

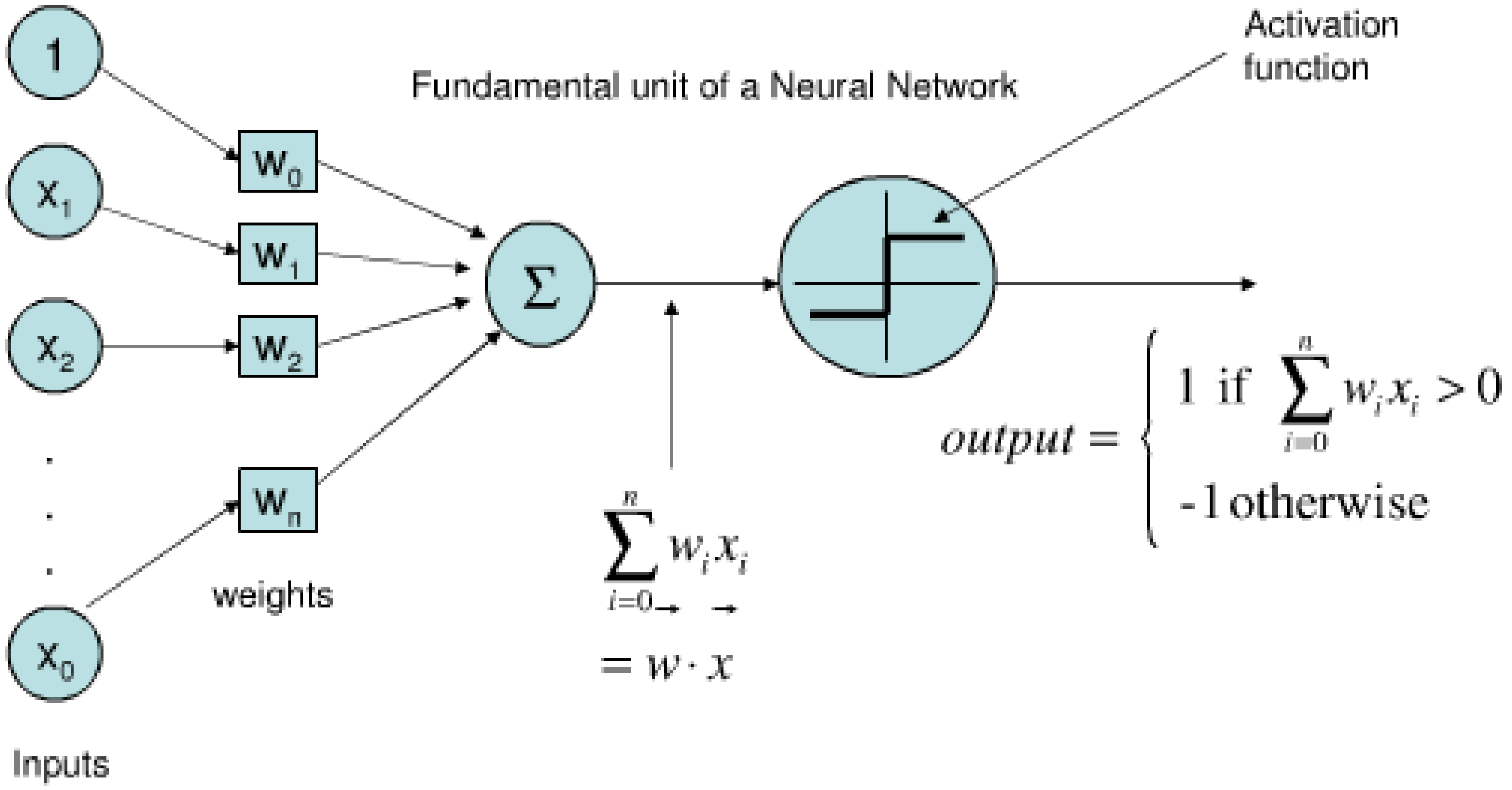
Types of Learning

- **Supervised learning**
  - Learning from examples
    - E.g. learn to distinguish between cats and dogs by looking at many cat and dog pictures that are labeled until you can identify cats and dogs on unseen pictures good enough
- **Unsupervised learning**
  - Figuring out the differences themselves
    - E.g. knowing there are two groups to divide the pictures into, comparing the pictures and find the most important combination of variables that are different and hence divide the pictures into two groups.
      - You don't know what the agent learns.
      - It could be cats vs dogs. It could be sunny pictures vs rainy pictures regardless of the animal in the pictures.
- **Reinforcement learning**
  - Figuring out the right action (sequence of actions) to do in order to achieve the goal of getting the highest reward
  - The agent choses the action(s) to do it by itself
  - It needs an evaluation function to decide if the action(s) lead to something good or not so good and presenting it with the reward

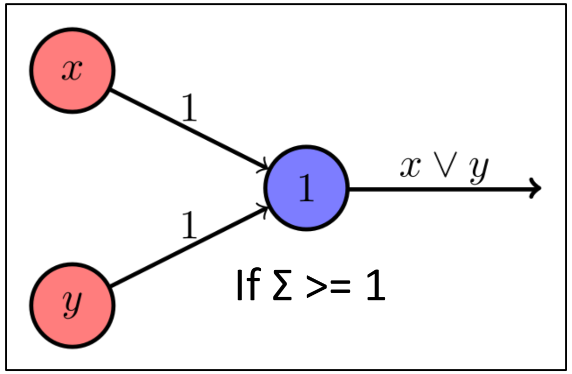
Neural network

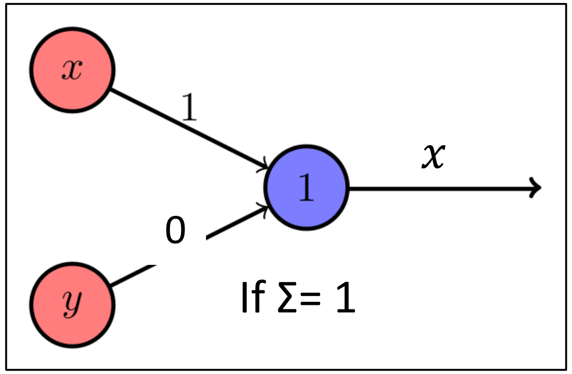


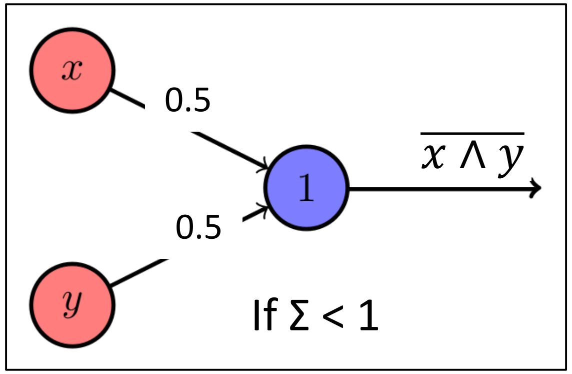
Perceptron (artificial neuron)

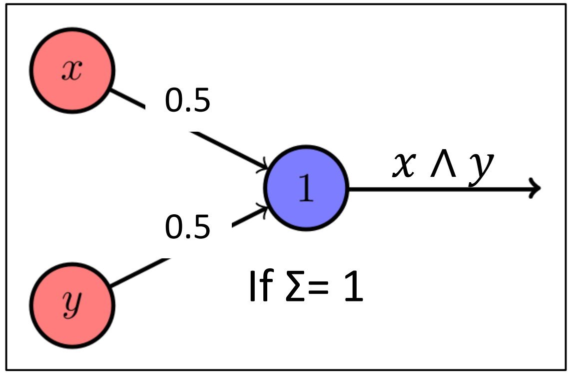


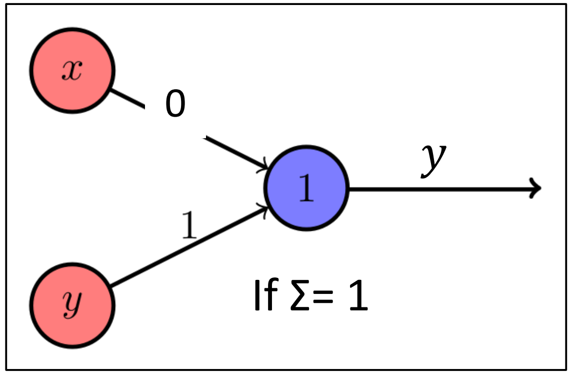
Perceptron -> logical gates



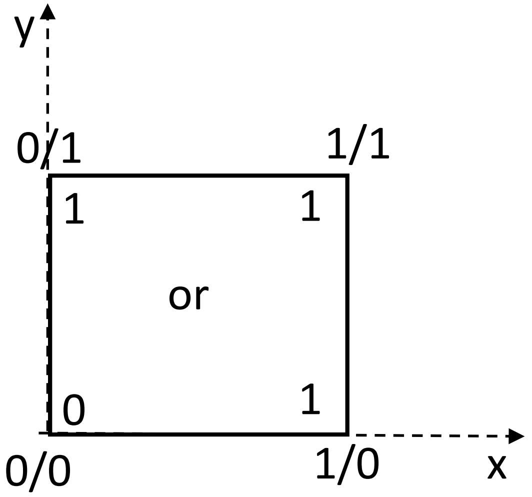


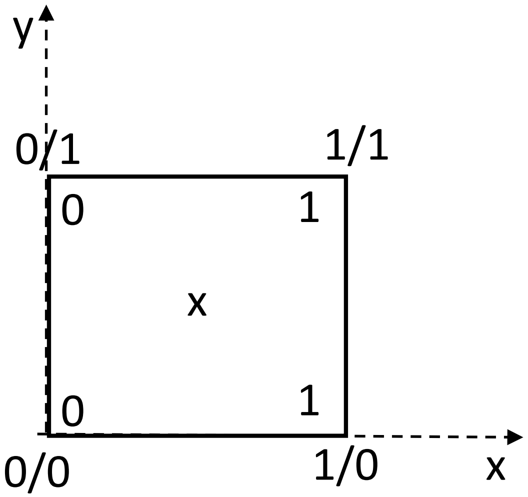


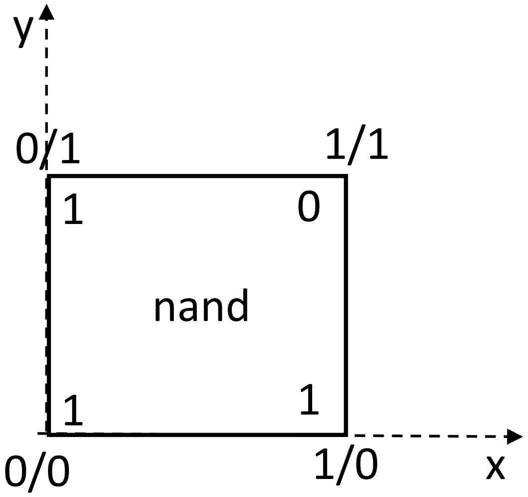


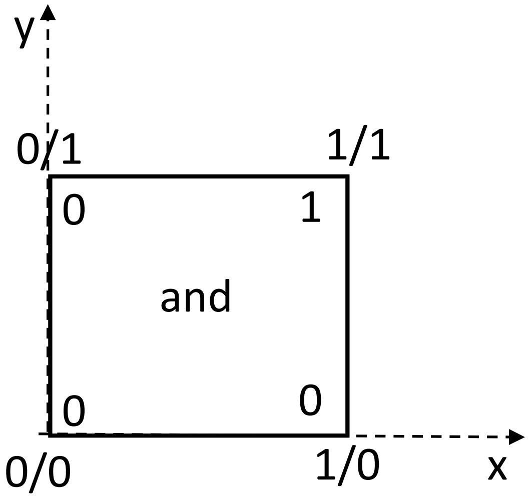


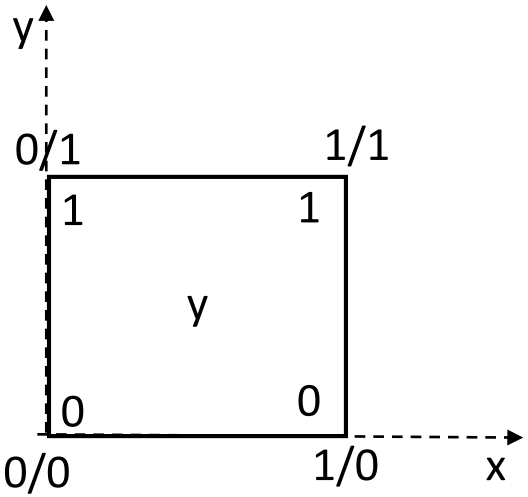
Perceptron -> logical gates



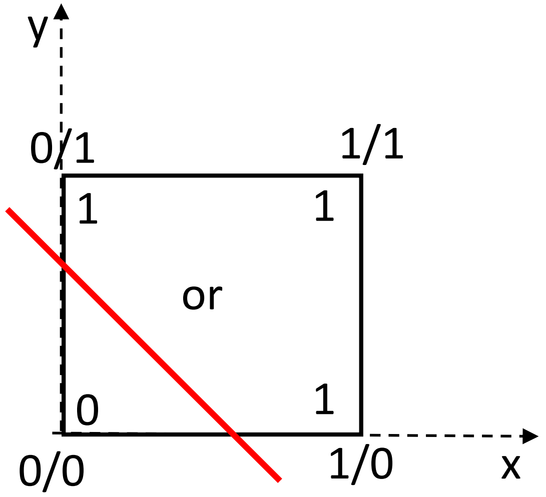


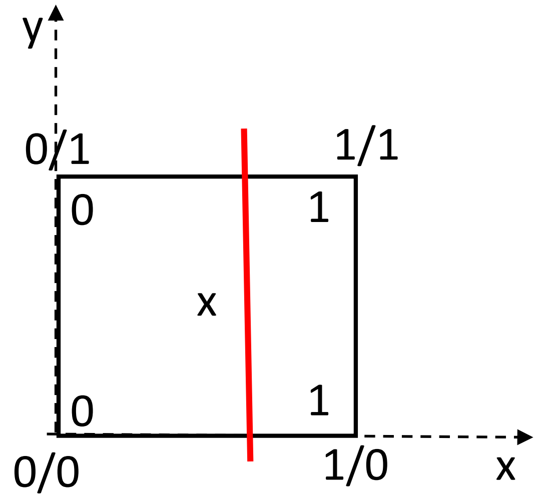


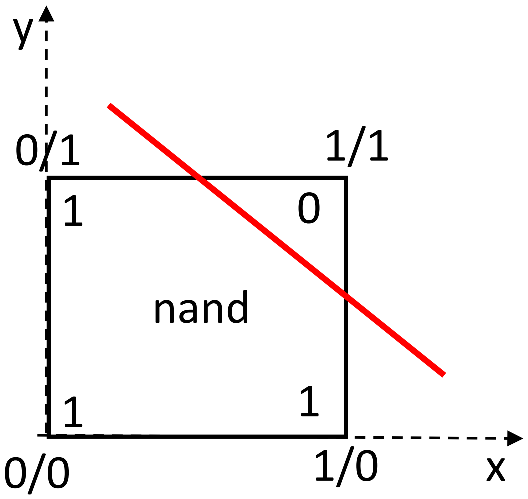


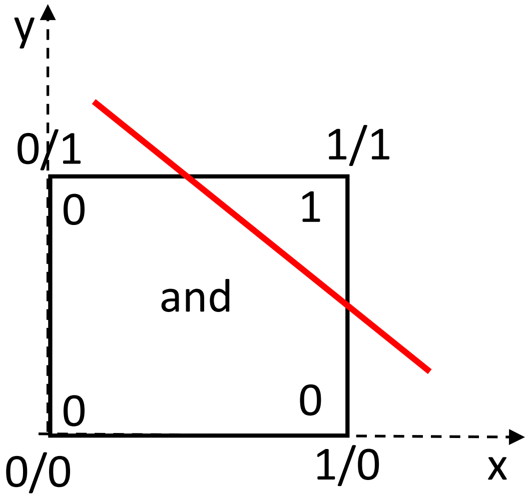


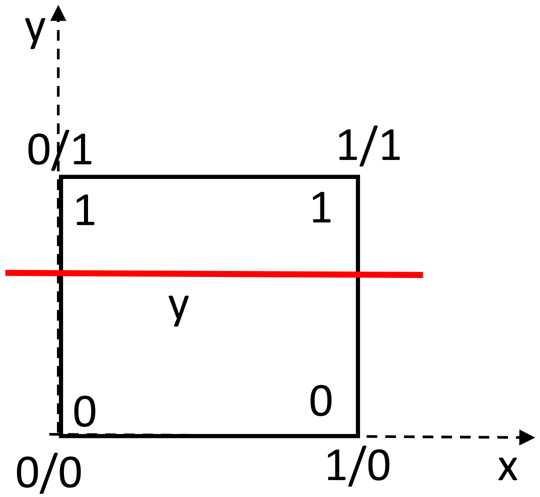
Perceptron -> logical gates









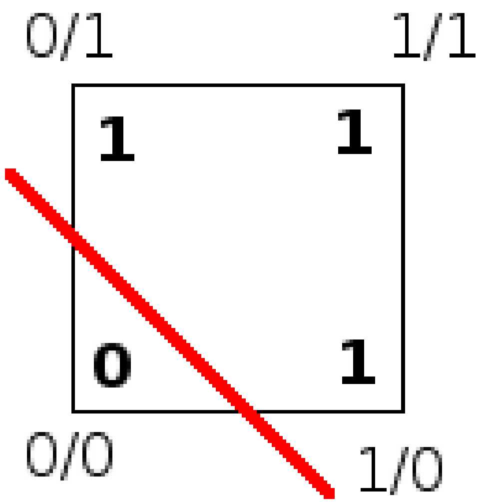


Formal for the straight line:
$y = f(x) = mx + b$

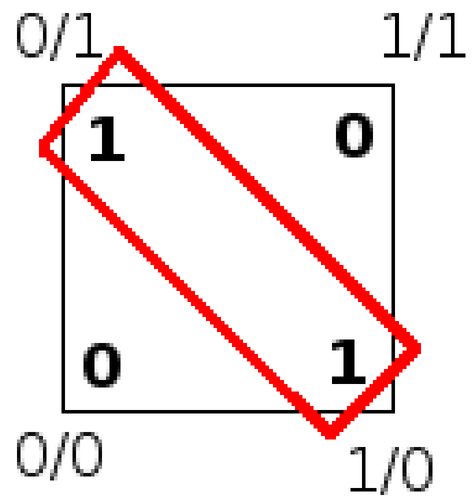
where:

- input x
- weight m
- bias b

XOR-Problem

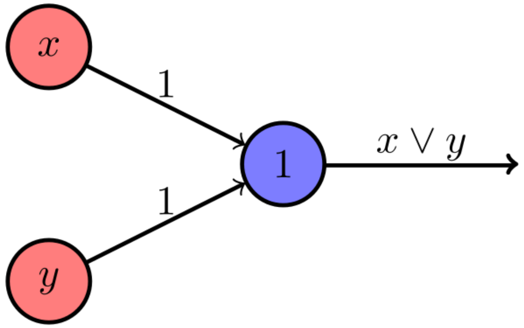


OR-Funktion:
Linear separierbar

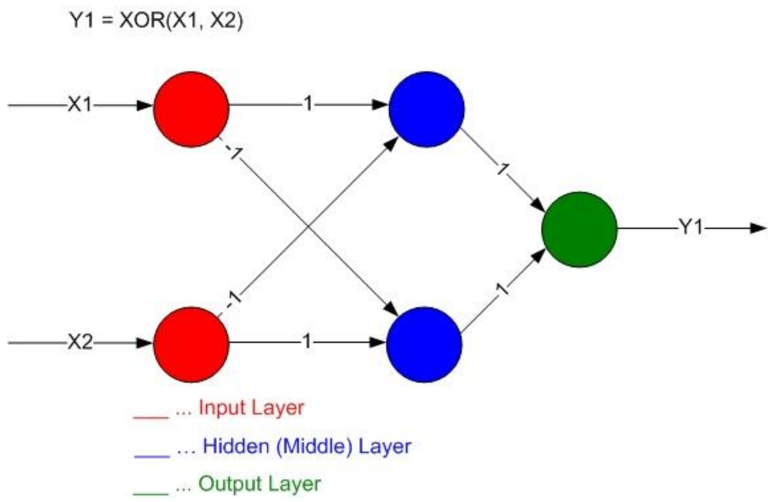


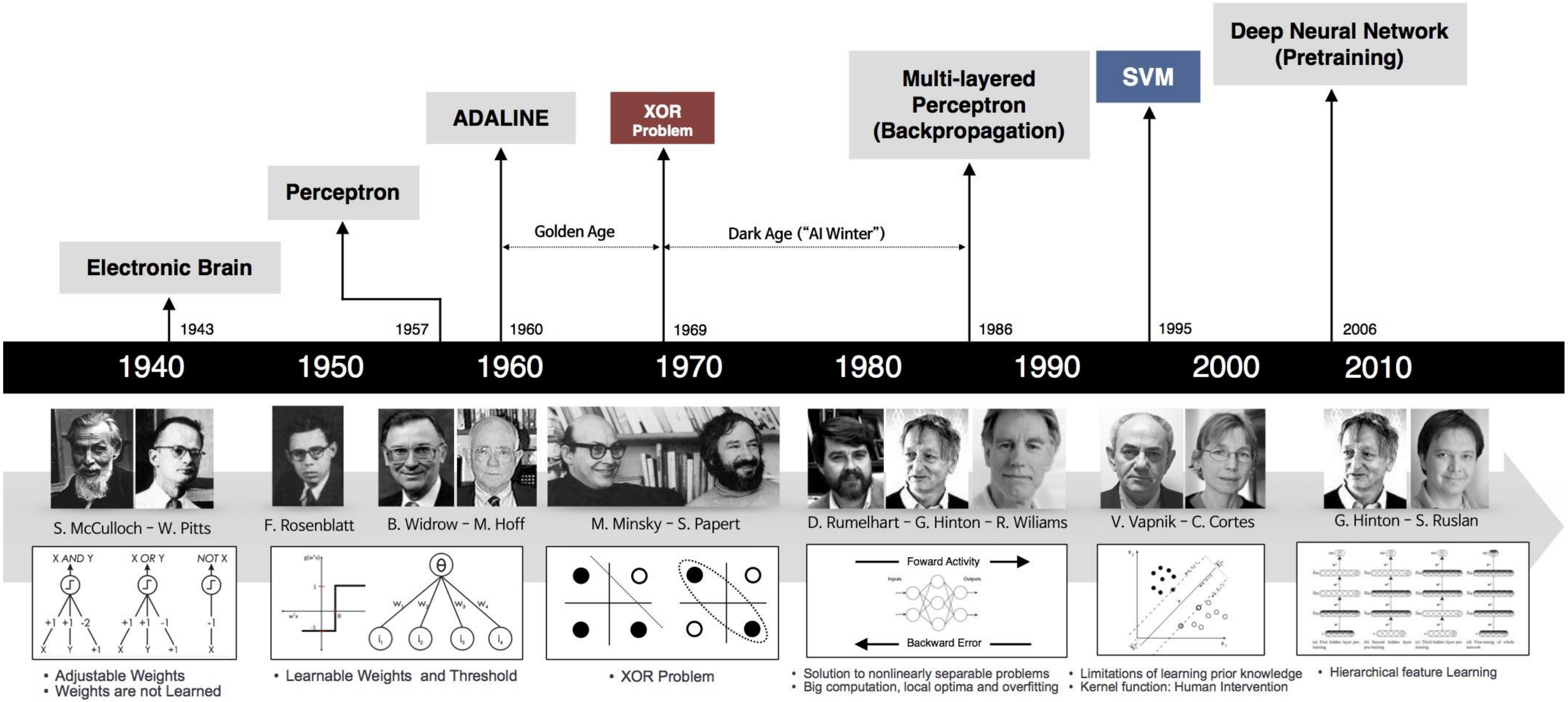
XOR-Funktion:
Nicht linear separierbar

Perceptron



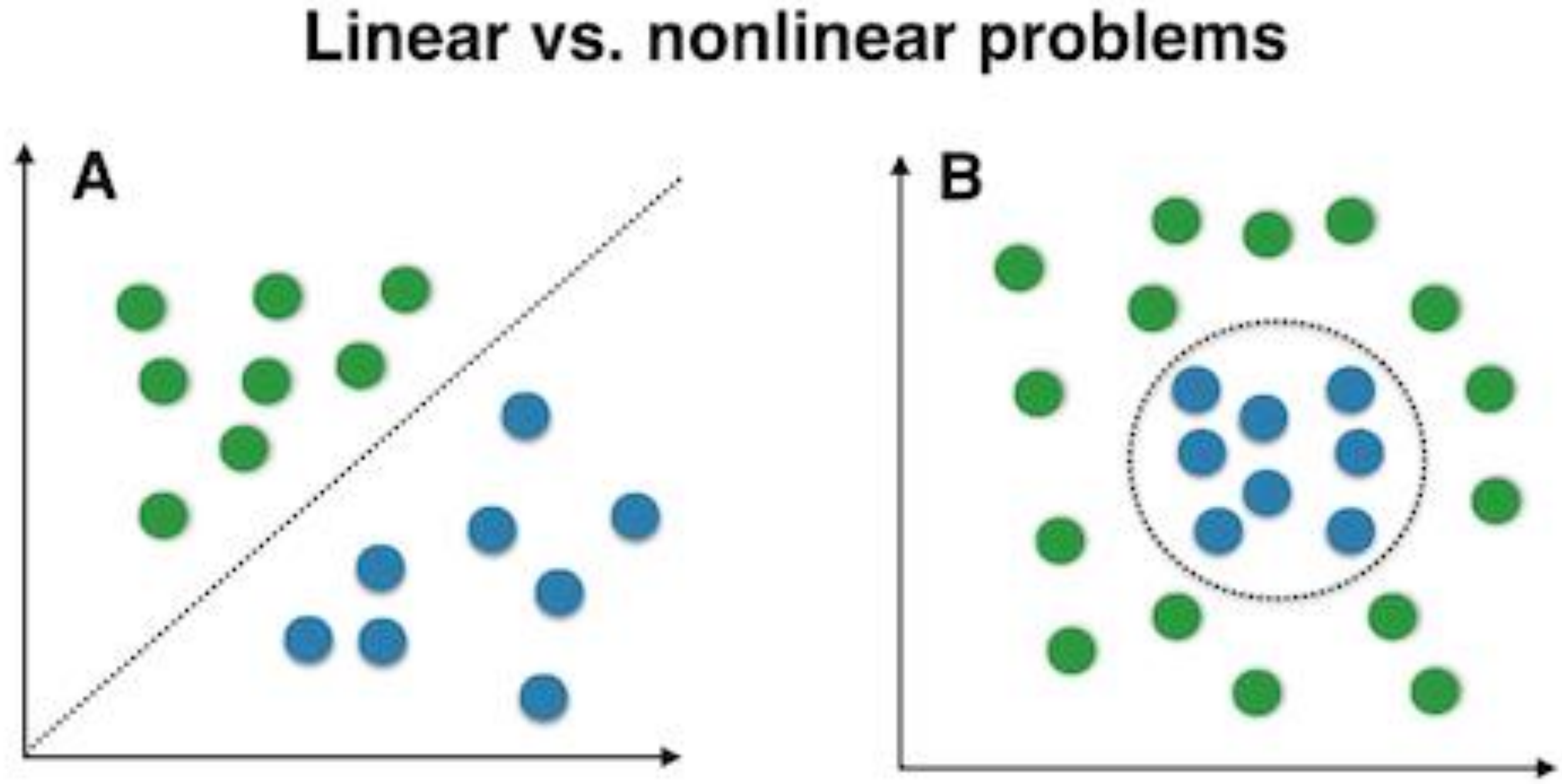
Multi-layer Perceptron



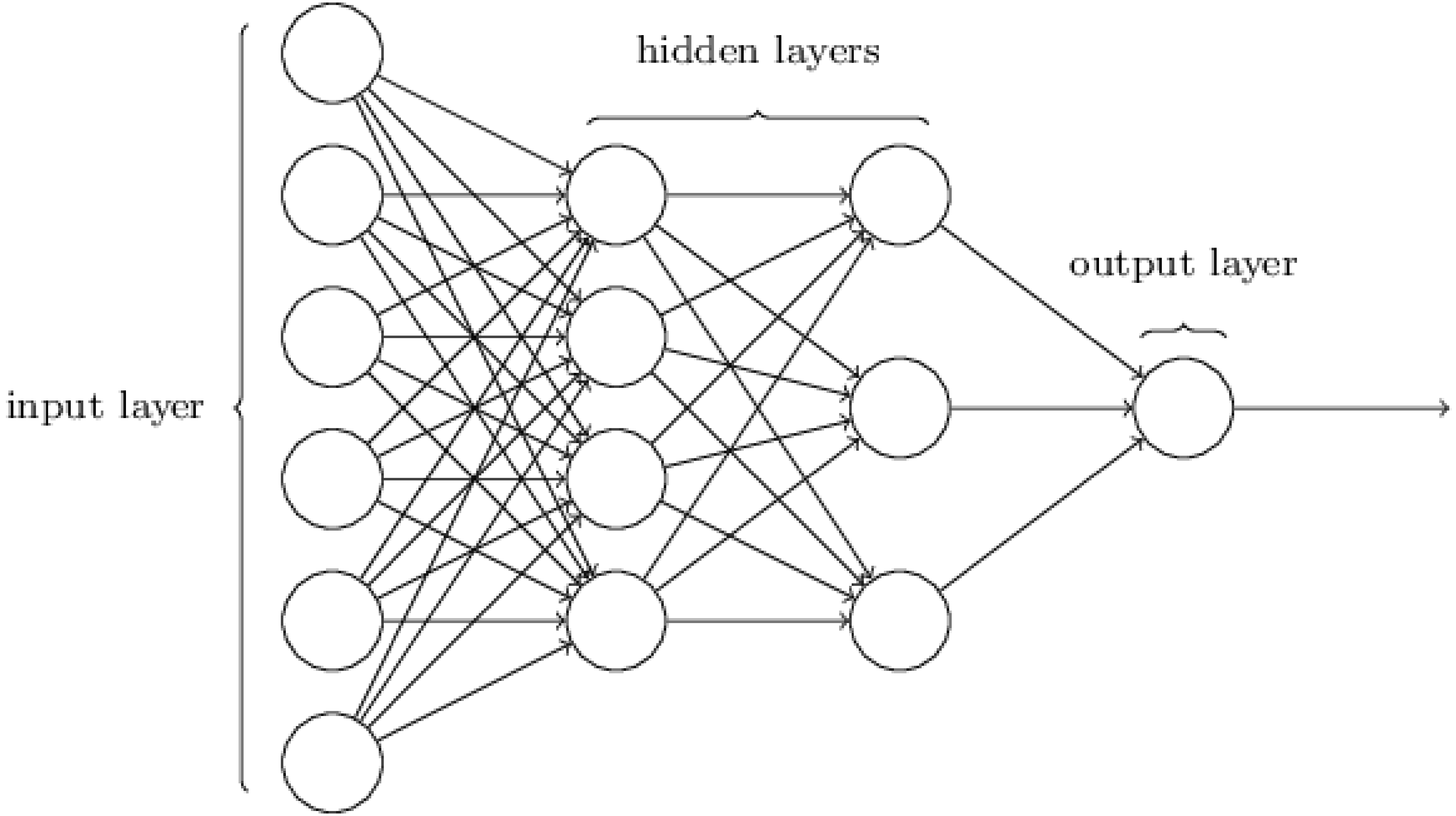


Electronic Brain
Perceptron
ADALINE
XOR Problem
Multi-layered Perceptron (Backpropagation)
SVM
Deep Neural Network (Pretraining)
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

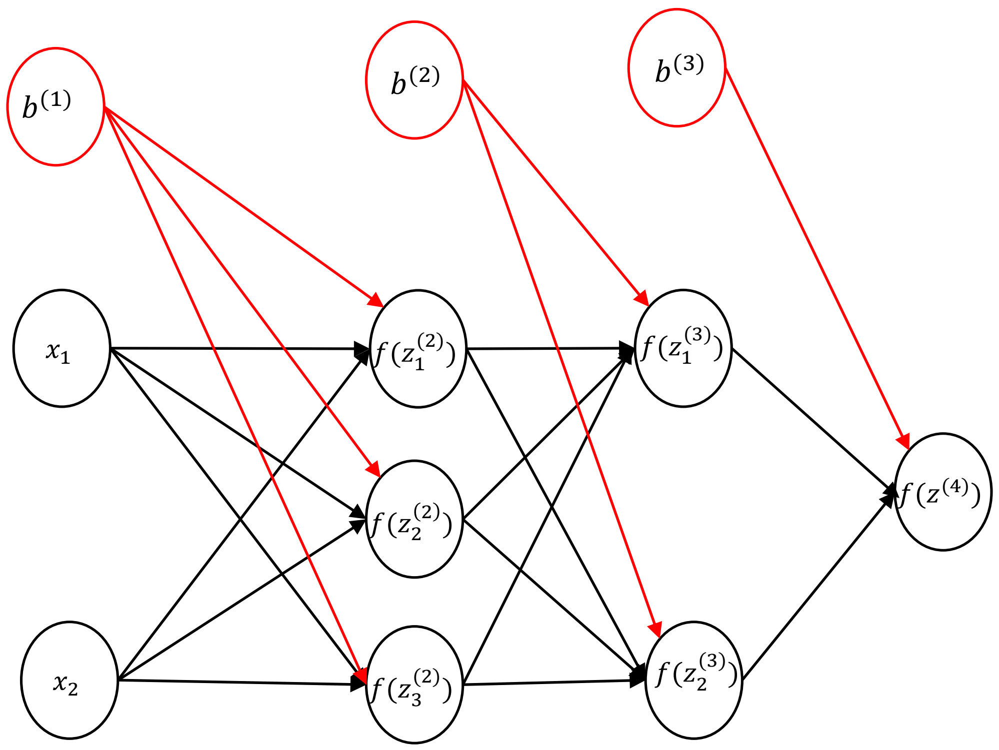
Classification



Multi Layer Perceptron (MLP)



Bias

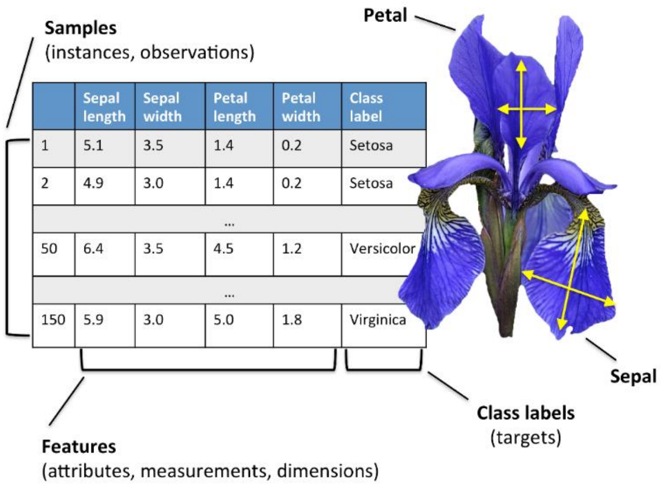


$$z_1^{(2)} = w_{11}^{(1)} * x_1 + w_{21}^{(1)} * x_2 + w_{b1}^{(1)} * b^{(1)}$$

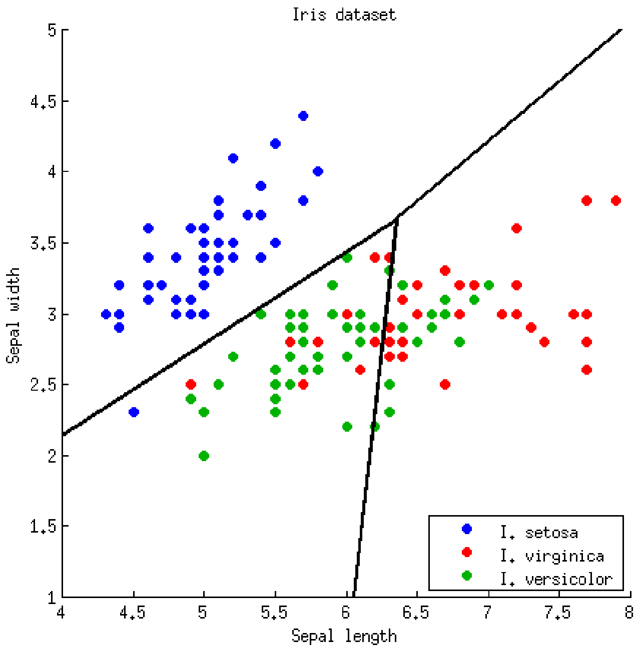
$$z_1^{(3)} = w_{11}^{(2)} * a_1^{(2)} + w_{21}^{(2)} * a_2^{(2)} + w_{31}^{(2)} * a_3^{(2)} + w_{b1}^{(2)} * b^{(2)}$$

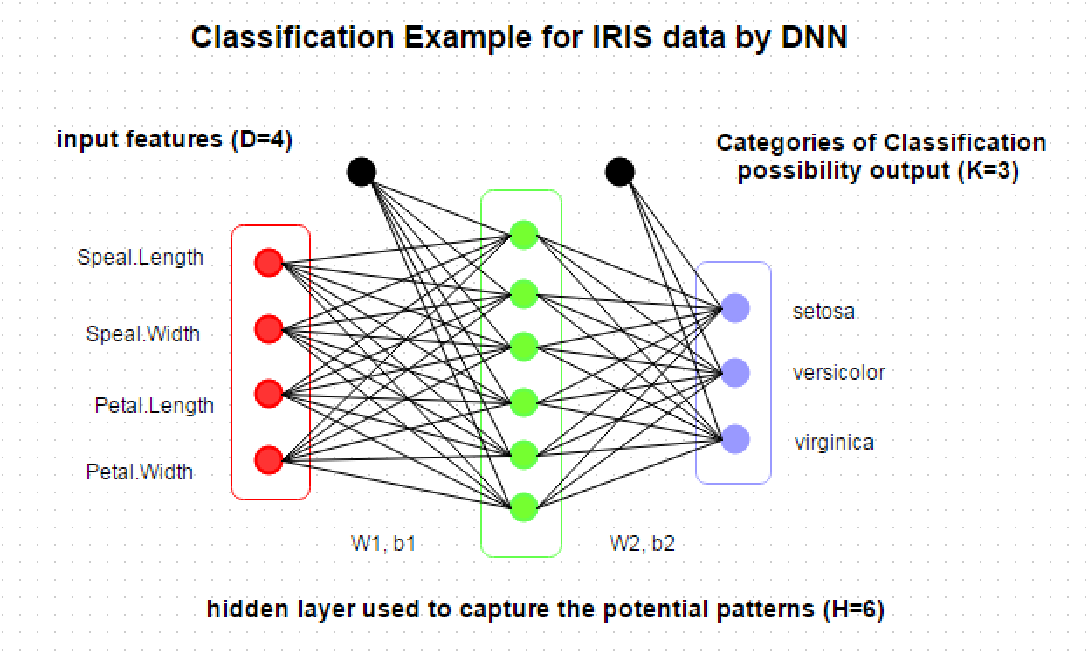
Multiclass Classification

Iris Versicolor **Iris Setosa** **Iris Virginica**

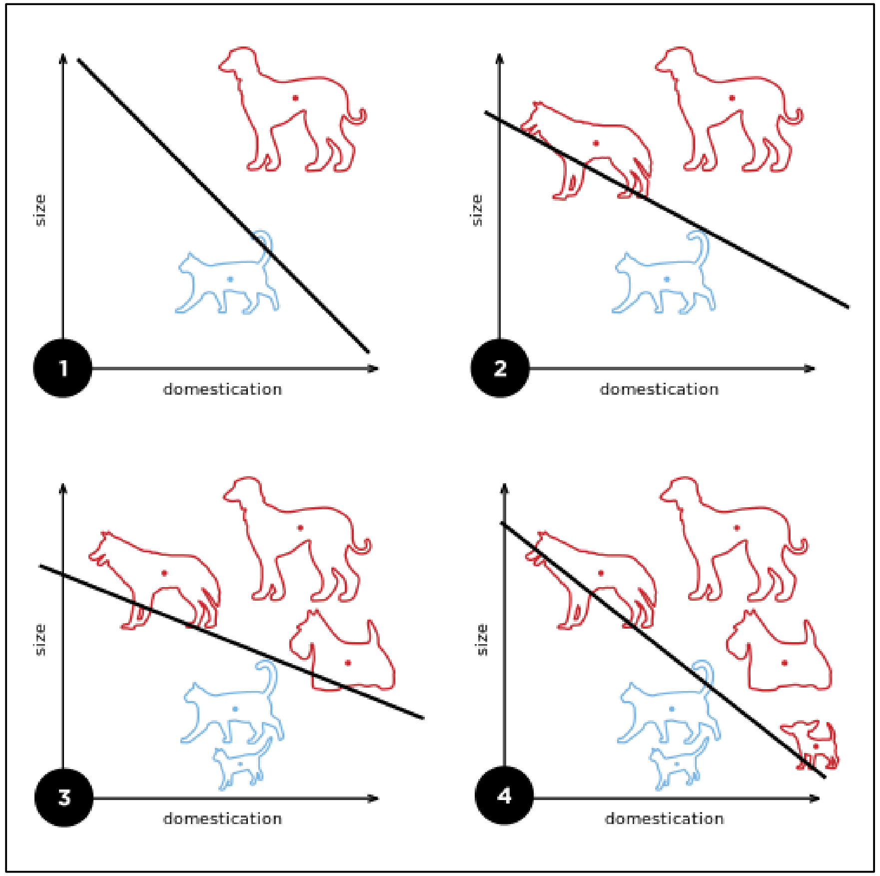


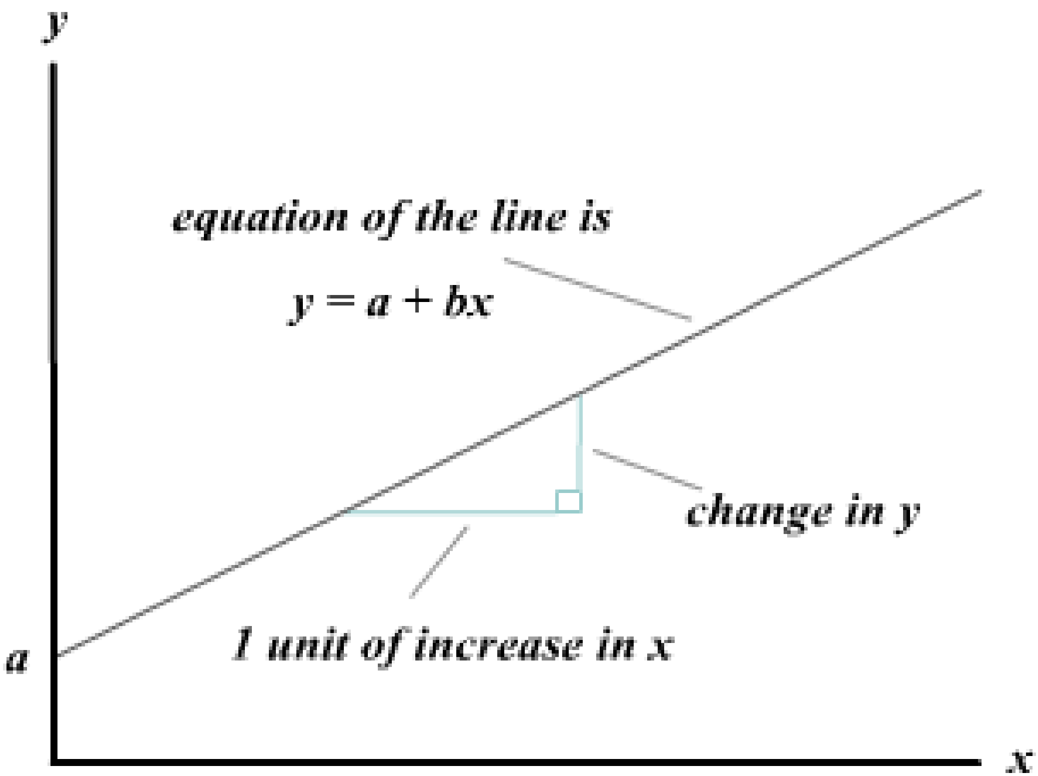
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica



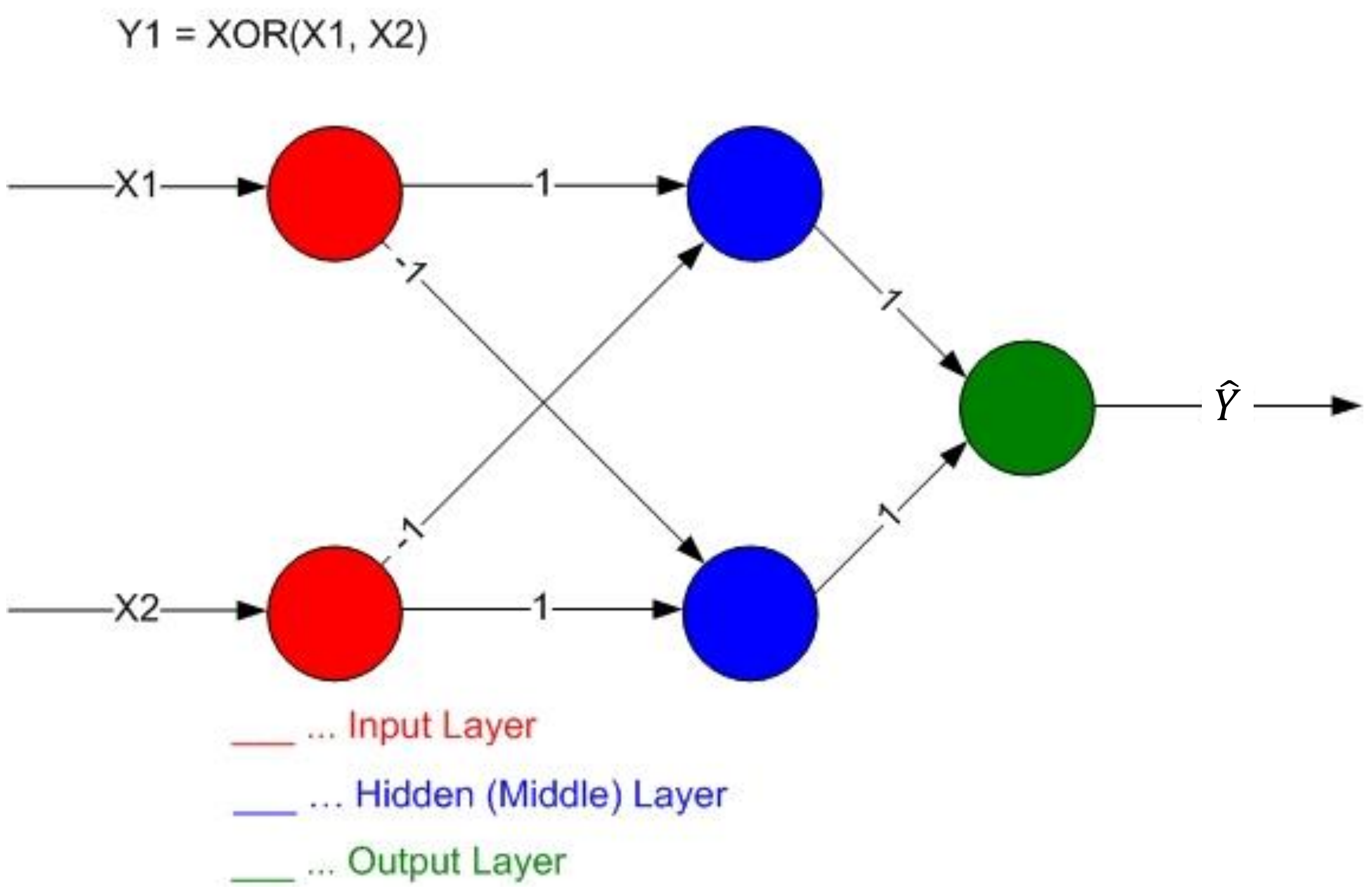


Classification

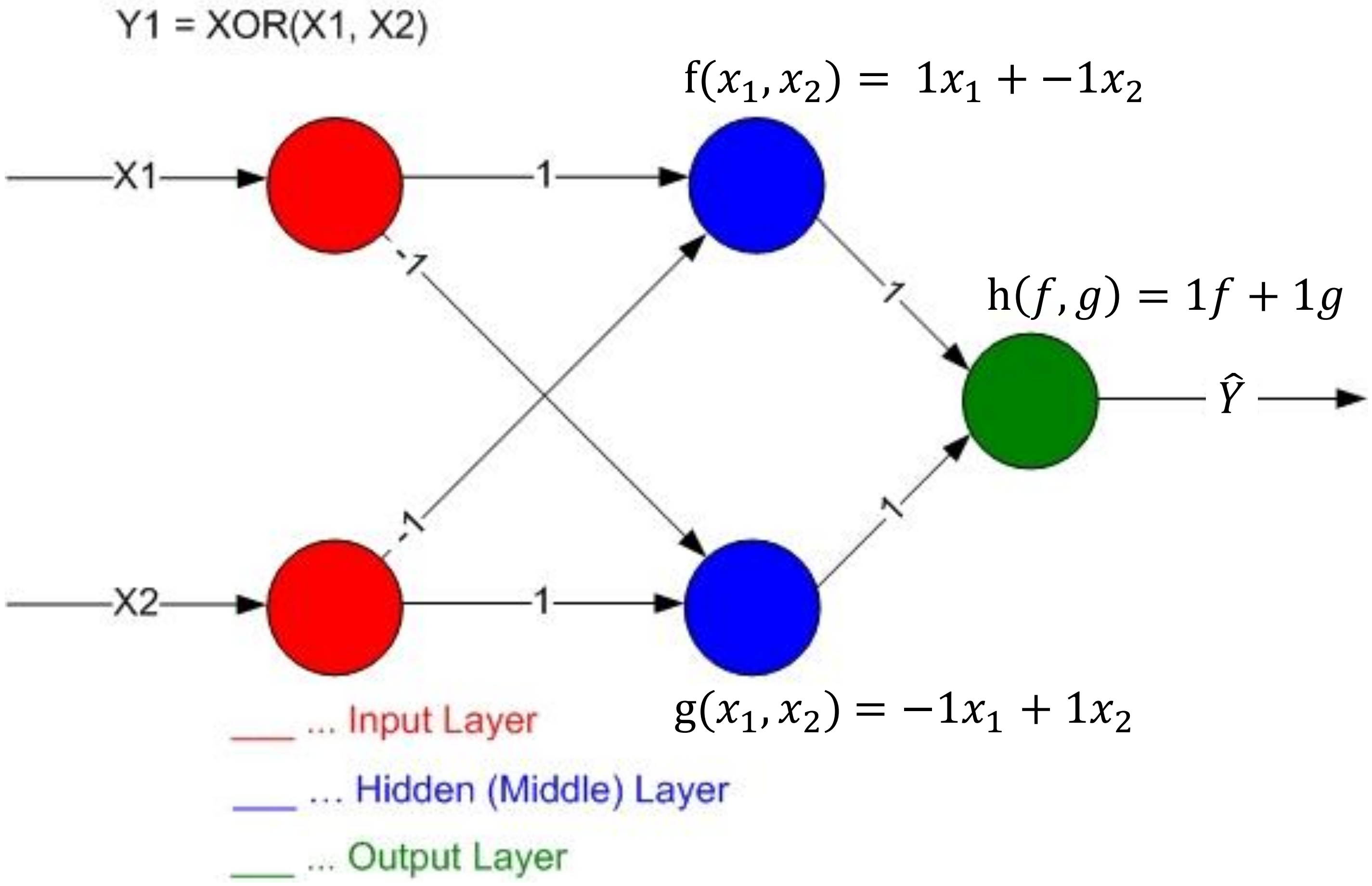




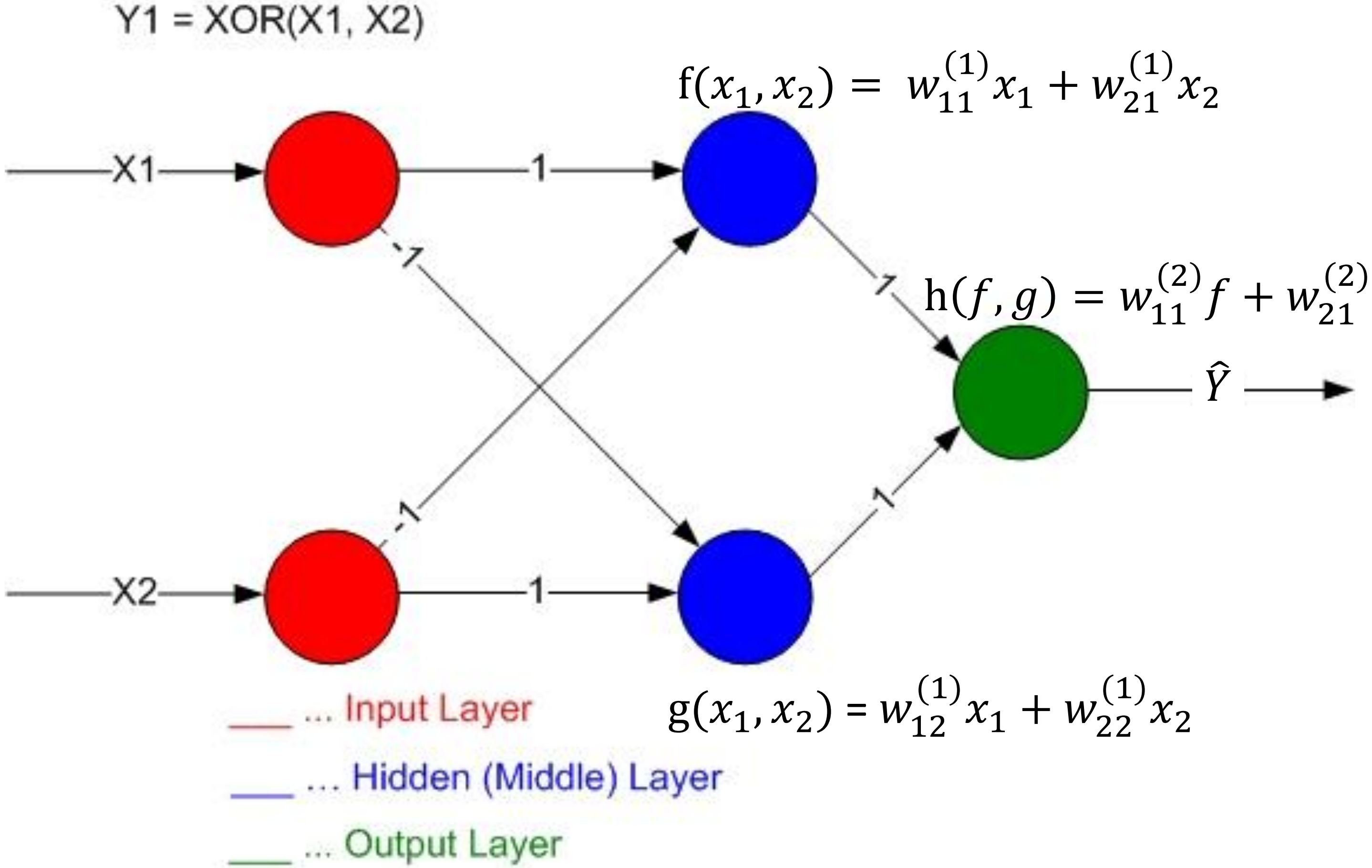
Multilayer Perceptron



Multilayer Perceptron



Multilayer Perceptron



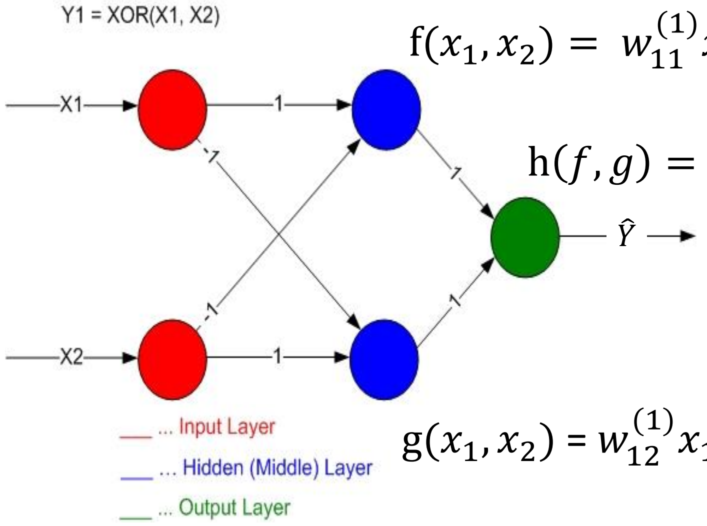
$\mathrm{f}(x_1, x_2) = \; w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2$

$\mathrm{h}(f, g) = w_{11}^{(2)} f + w_{21}^{(2)} \; g = w_{11}^{(2)}(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2) + w_{21}^{(2)}(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2)$

$\hat{Y}$

$\mathrm{g}(x_1, x_2) = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2$

Multilayer Perceptron



$$f(x_1, x_2) = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2$$

$$h(f, g) = w_{11}^{(2)} f + w_{21}^{(2)} g = w_{11}^{(2)}(w_{11}^{1} x_1 + w_{21}^{(1)} x_2) + w_{21}^{(2)}(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2) = \hat{Y}$$

$$g(x_1, x_2) = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2$$

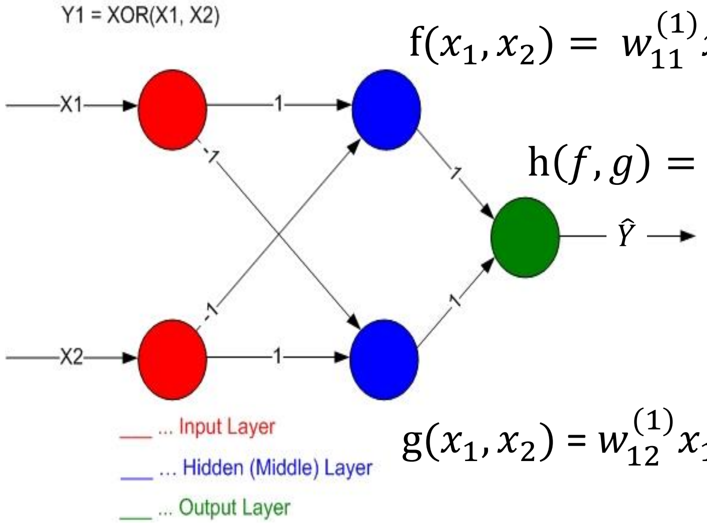
Hidden layer:

$$[x_1 \quad x_2] * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = [x_1 * w_{11}^{(1)} + x_2 * w_{21}^{(1)} \quad x_1 * w_{12}^{(1)} + x_2 * w_{22}^{(1)}] = [z_1^{(2)} \quad z_2^{(2)}]$$

Output layer:

$$[z_1^{(2)} \quad z_2^{(2)}] * \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \end{bmatrix} = [z_1^{(2)} * w_{11}^{(2)} + z_2^{(2)} * w_{21}^{(2)}] = [z_1^{(3)}] = \hat{Y}$$

Multilayer Perceptron



$$f(x_1, x_2) = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2$$

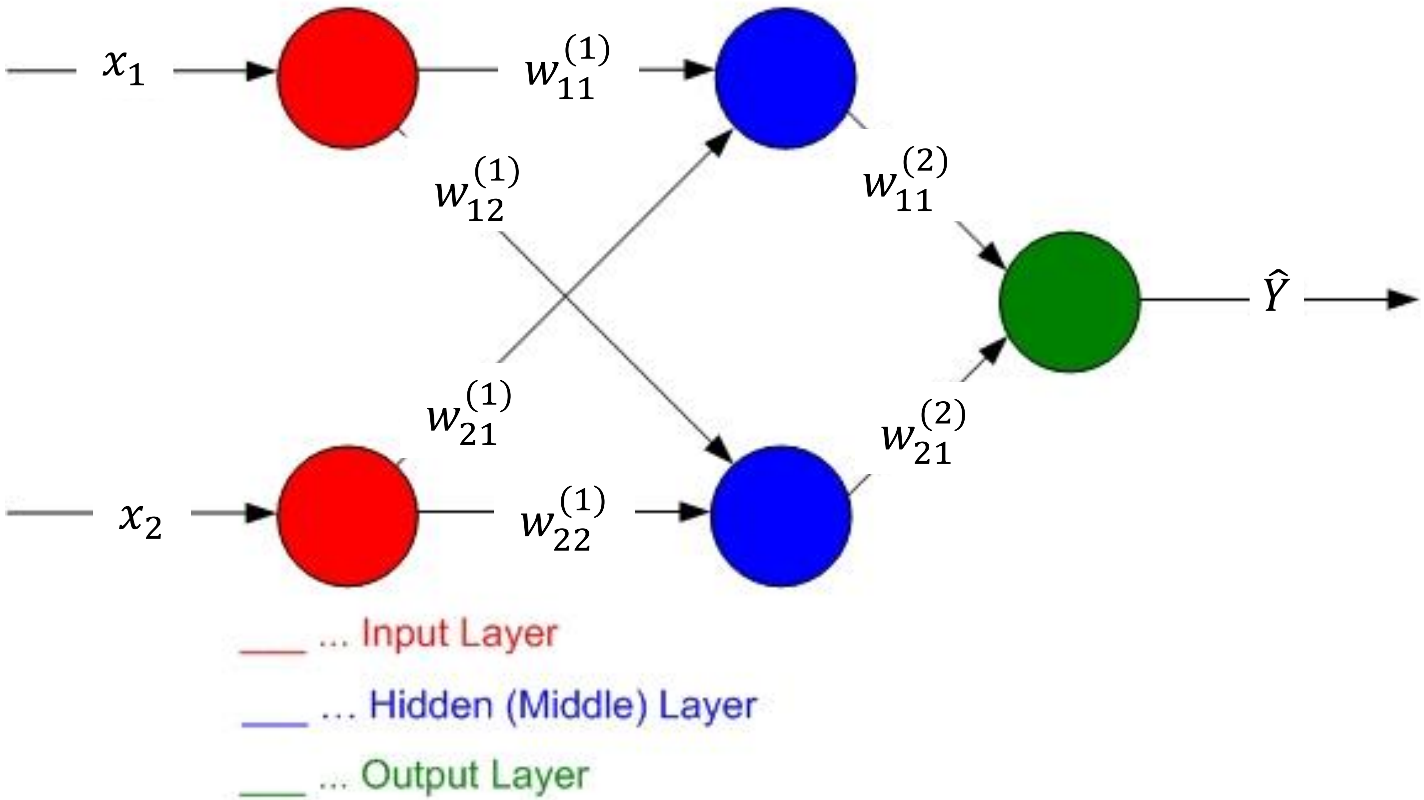
$$h(f, g) = w_{11}^{(2)} f + w_{21}^{(2)} g = w_{11}^{(2)}(w_{11}^{1} x_1 + w_{21}^{(1)} x_2) + w_{21}^{(2)}(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2) = \hat{Y}$$

$$g(x_1, x_2) = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2$$

Calculation for the hidden layer for multiple inputs:

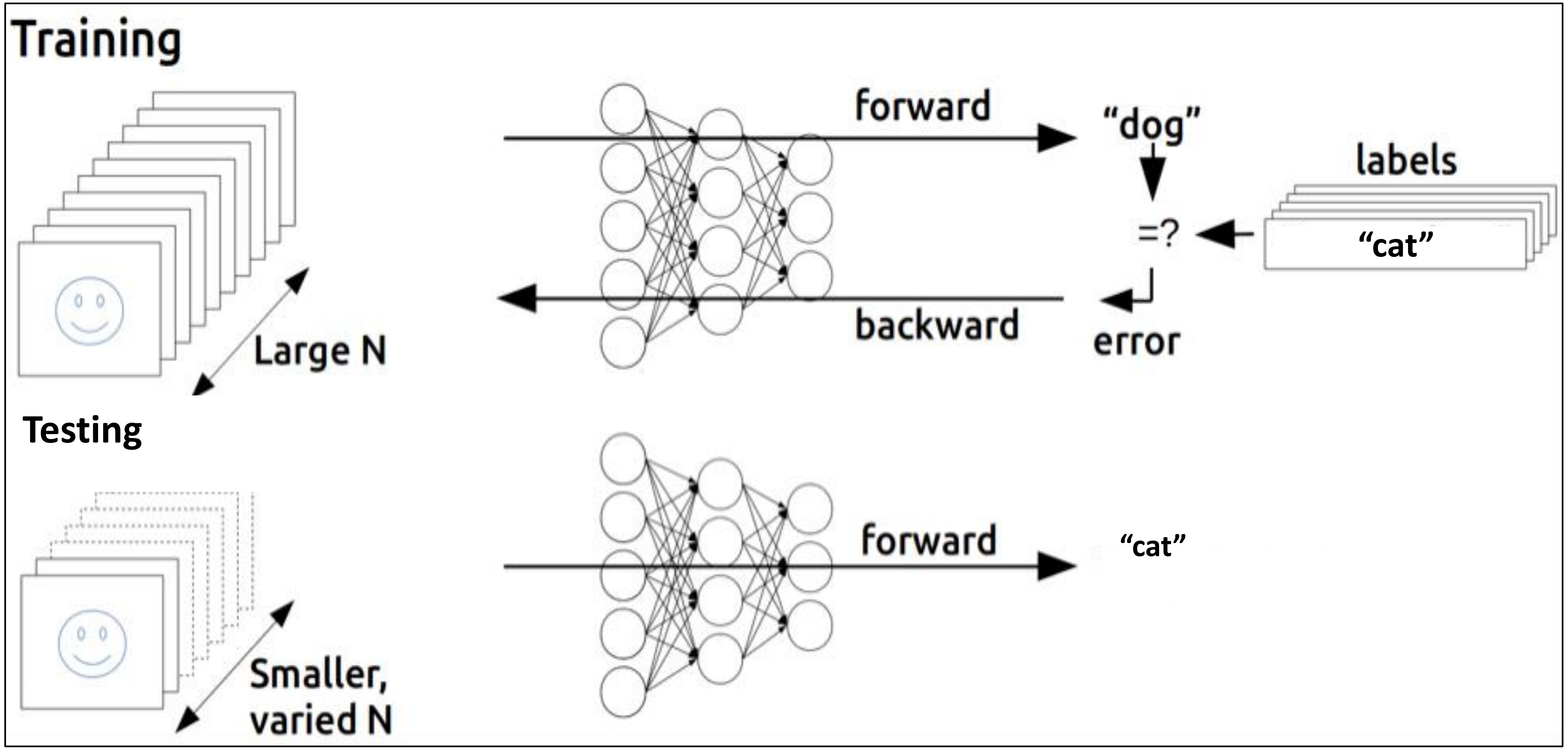
$$\begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} z_{11}^{(2)} & z_{12}^{(2)} \\ z_{12}^{(2)} & z_{22}^{(2)} \\ z_{13}^{(2)} & z_{23}^{(2)} \end{bmatrix}$$

Multilayer Perceptron

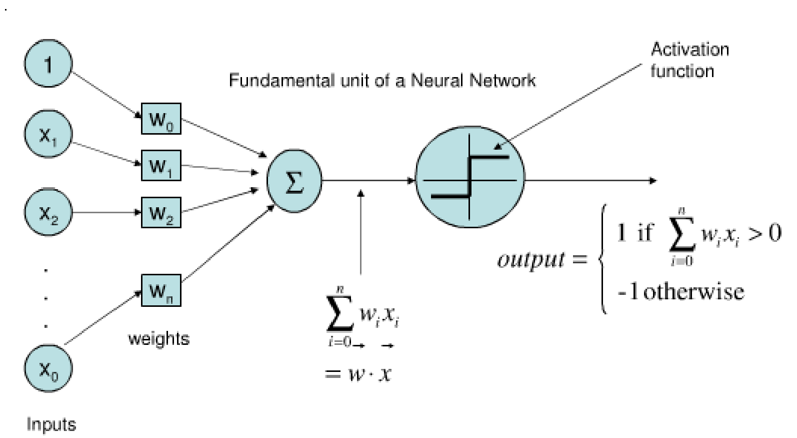


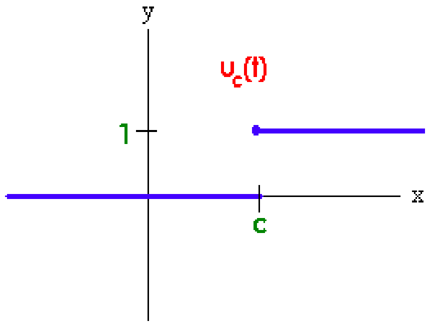
- Weights (often randomized initially) are assigned to the edges, and used in the calculation of network output
- During learning, records in the training set are fed into the network and output is calculated
- The difference between the known output and the actual output is calculated and used to update the weights (supervised learning)
- The training set is presented a number of times (epochs), typically until there is little or no difference between the actual output and known output

Multilayer perceptron

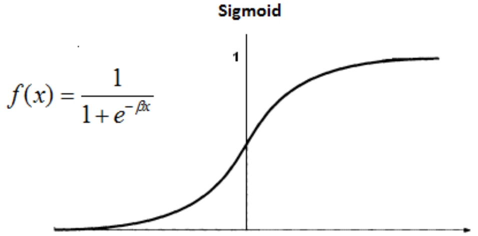


Activation functions (examples)

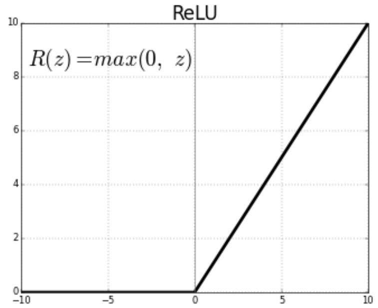




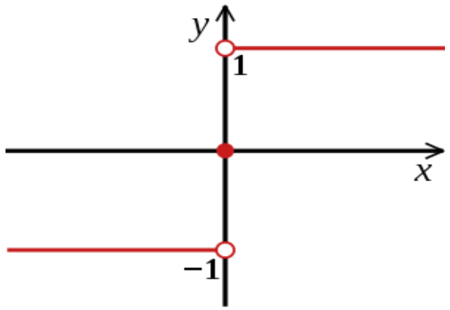
Step function



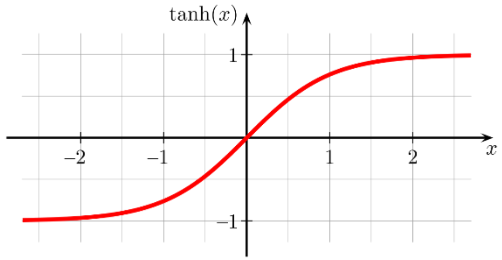
Sigmoid function



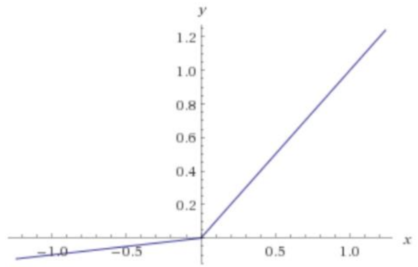
ReLU function



Sign function

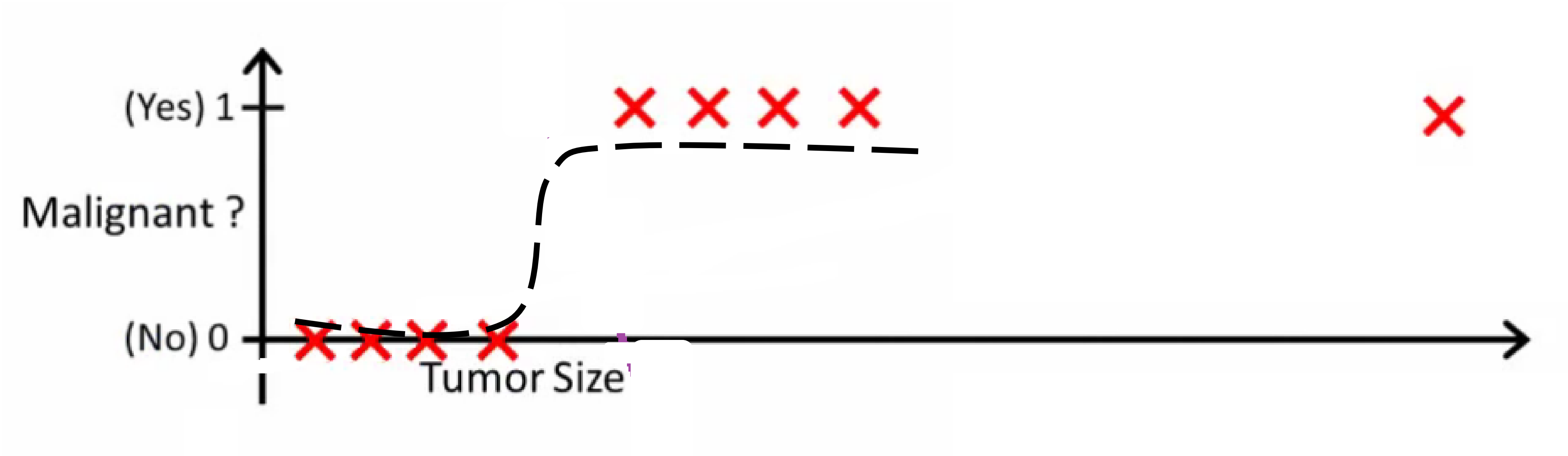


Tanh function

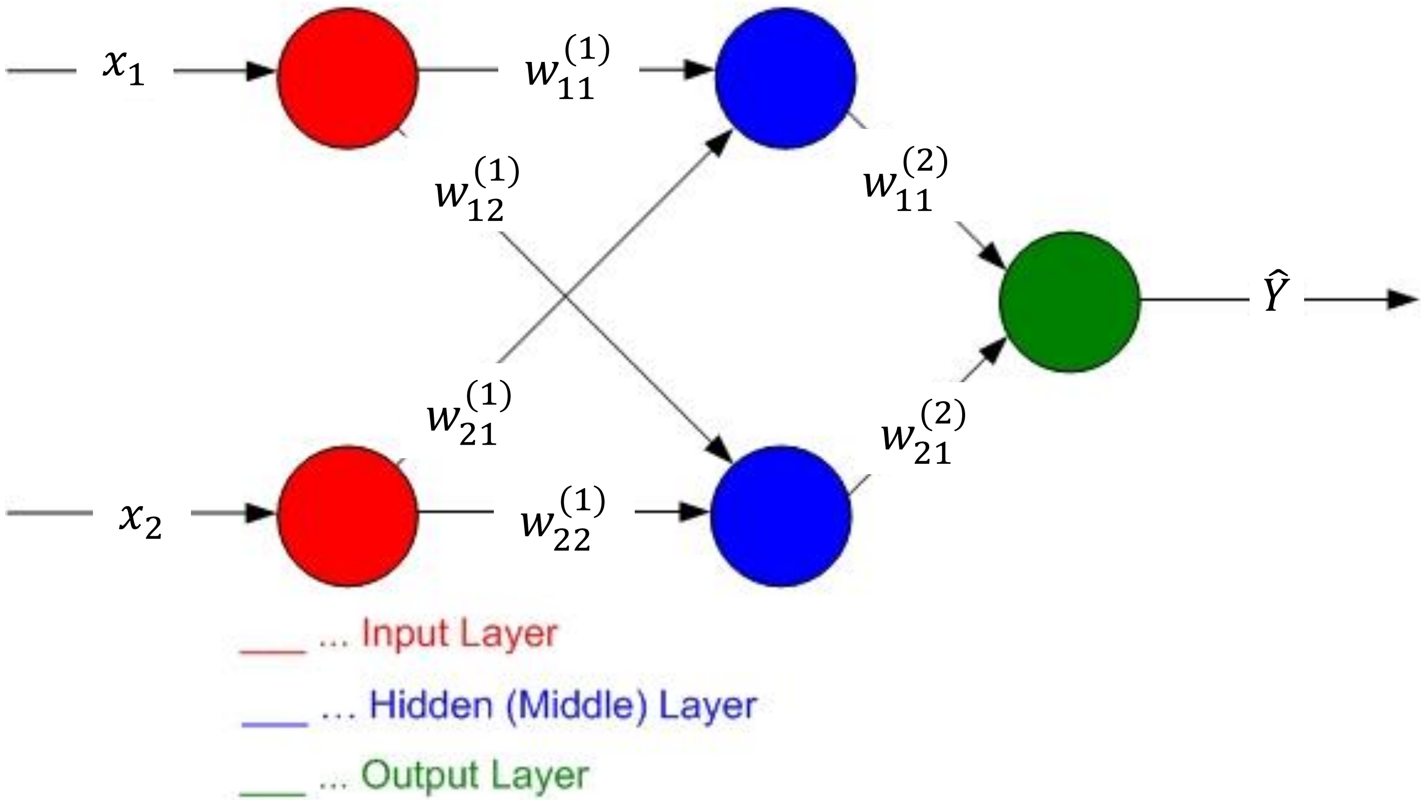


Leaky ReLU function

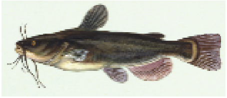
Classification with logistic regression



Multilayer Perceptron



Supervised Learning

	Training set x		Estimated result	True result (Cat = 1 Not cat = 0)
	whiskers x_1	pointy ears x_2	$\hat{y}$	y
	**1**	**1**	**0.25**	**1**
	**0**	**0**	**0.63**	**0**
	**1**	**1**	**0.51**	**1**
	**1**	**0**	**0.23**	**0**

Cost function / Loss function

- How good is the output?

- Classification problem where y ∈ $\{0,1\}$
  - y is the real class of the input
  - $\hat{y}$ is the hypothesis of the outcome

- We want the hypothesis $\hat{y}$ at least to be very close to 1 or very close to 0 for positive and negative examples, respectively

- Cost function that tells us how far of we are from the wanted result

error

$$cost(y, \hat{y}) = \hat{y} - \mathrm{y}$$

squared error

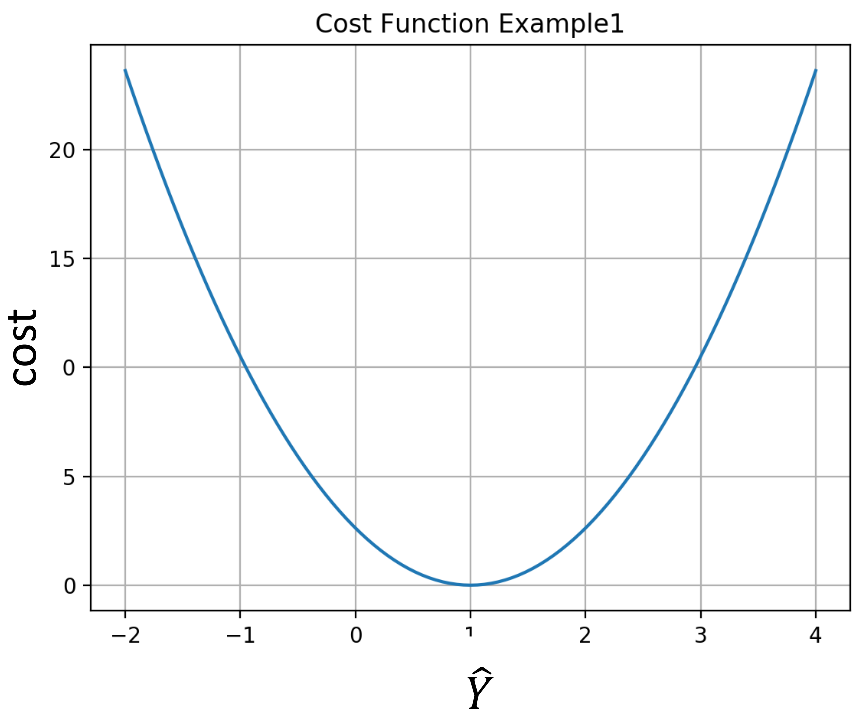
$$cost(y, \hat{y}) = (\hat{y} - \mathrm{y})^{2}$$

mean squared error (for multiple input

$$cost(Y, \hat{Y}) = \frac{1}{2m}\sum_{i=0}^{m}(\hat{y}_i - y_i)^2$$

Often used for logistic regression

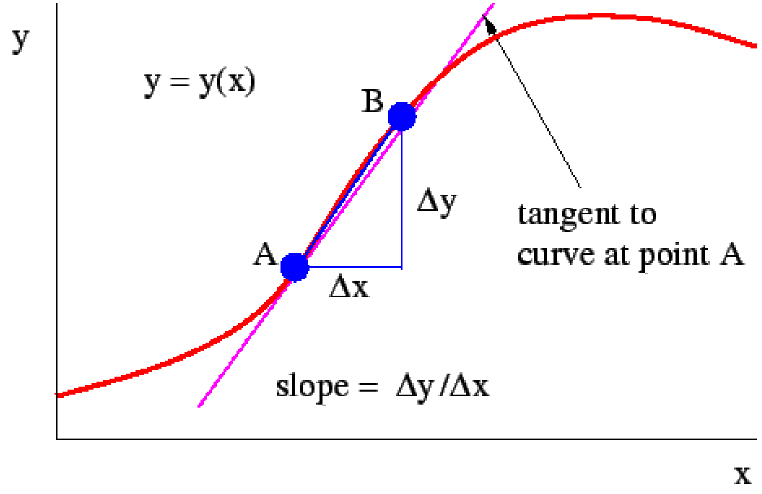
$$cost(y, \hat{y}) = \begin{cases} -\log(\hat{y}) \;\; if \;\; y = 1 \\ -\log(1 - \hat{y}) \;\; if \;\; y = 0 \end{cases}$$



Partial derivatives and gradient

- How much influence has each weight on the result
- *Partial derivative* of the cost function with regard to the chosen weight

$$\frac{\partial cost}{\partial w_{ij}^{(k)}}$$



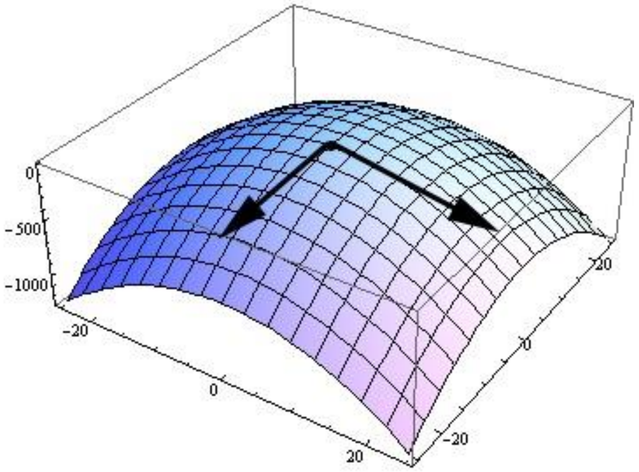
- The *gradient* is the vector of all partial derivatives

$$W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \end{bmatrix}$$

Weight matrix $W^{(1)}$

$$\nabla W^{(1)} = \begin{bmatrix} \frac{\partial cost}{\partial w_{11}^{(1)}} & \frac{\partial cost}{\partial w_{21}^{(1)}} \\ \frac{\partial cost}{\partial w_{12}^{(1)}} & \frac{\partial cost}{\partial w_{22}^{(1)}} \end{bmatrix}$$

Gradient for weight matrix $W^{(1)}$



Updating the weights

Gradient for weight matrix $W^{(1)}$

Learning rate

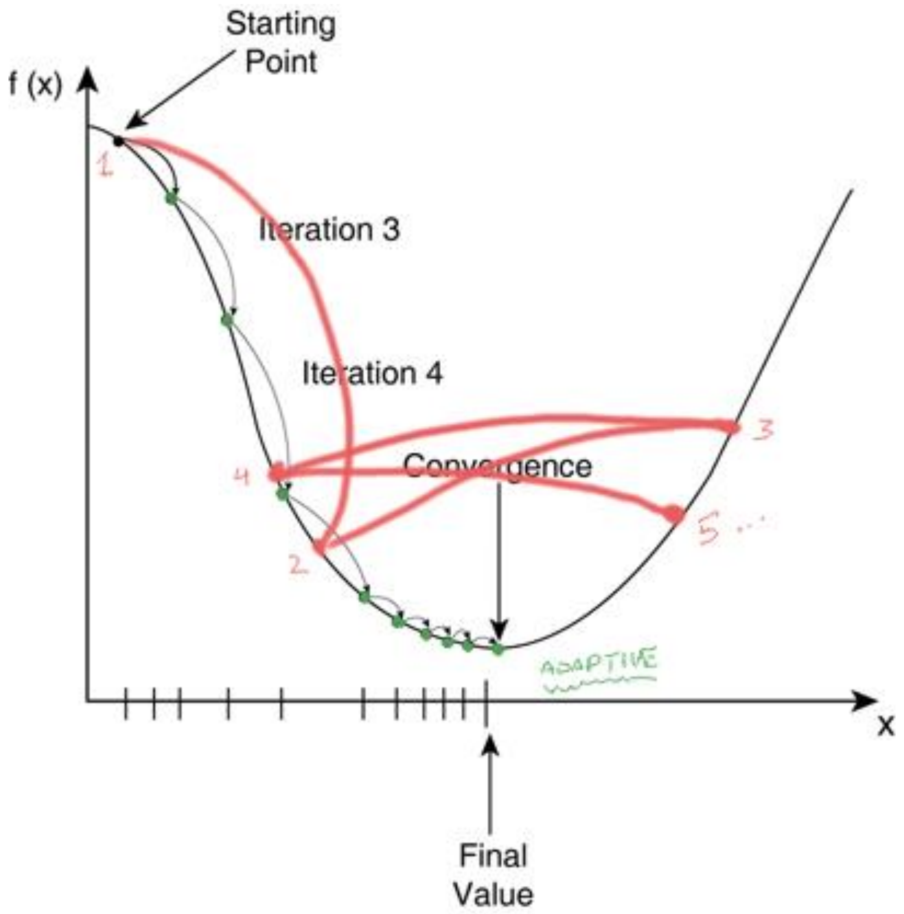
$$newW^{(1)} = W^{(1)} - (\nabla W^{(1)} * \alpha)$$

$$\nabla W^{(1)} * \alpha = \begin{bmatrix} \frac{\partial cost}{\partial w_{11}^{(1)}} & \frac{\partial cost}{\partial w_{21}^{(1)}} \\ \frac{\partial cost}{\partial w_{12}^{(1)}} & \frac{\partial cost}{\partial w_{22}^{(1)}} \\ \frac{\partial cost}{\partial w_{13}^{(1)}} & \frac{\partial cost}{\partial 23} \end{bmatrix} * \alpha = \begin{bmatrix} \alpha \frac{\partial cost}{\partial w_{11}^{(1)}} & \alpha \frac{\partial cost}{\partial w_{21}^{(1)}} \\ \alpha \frac{\partial cost}{\partial w_{12}^{(1)}} & \alpha \frac{\partial cost}{\partial w_{22}^{(1)}} \\ \alpha \frac{\partial cost}{\partial w_{13}^{(1)}} & \alpha \frac{\partial cost}{\partial 23} \end{bmatrix}$$

Learning rate

With constant learning rate

With Learning rate decay



- Learning rate typically between 0.01 and 0.0001
- With a small learning rate it takes long time to find the minimum of the cost function
- With a big learning rate, it takes less time to go downhill
  but the step size might be too big to hit the minimum
- With learning rate decay, you start with a big learning rate
- to go quickly downhill
  You in decrease learning rate in each step to make smaller steps in the end
  in order to be able to hit the minimum of the cost function

Decaying learning rate

(some common examples)

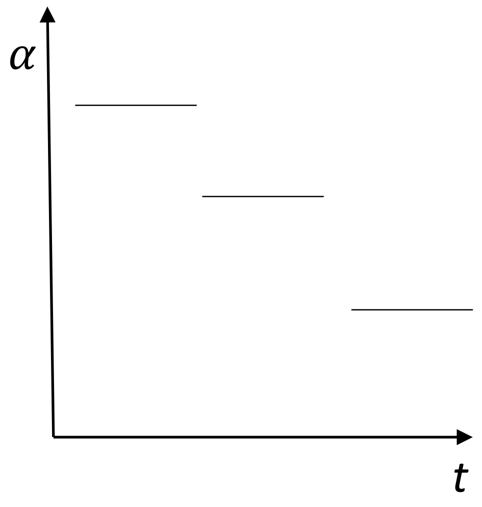
$$\alpha = \frac{1}{1 + decay\_rate \ * epoch\_number} * \alpha_0$$

$$\alpha = 0.95^{epoch\_number} * \alpha_0$$

$$\alpha = \frac{k}{\sqrt{epoch\_number}} * \alpha_0$$

$$\alpha = \frac{k}{\sqrt{t}} * \alpha_0$$

Learning rate as a step function



Manual decay (only recommended for small datasets)

Train, validate, test

- **Training data set**
  - The data that the model is trained on
  - Big risk of overfitting

- **Validation data set**
  - The data that the model build during training is validated on
  - If the validation is not good enough you go back to train the model after which you validate again on the validation set
  - Overfitting likely

- **Test data set**
  - Only used once when the final model is ready.
  - The algorithm has never seen this data before

Training data set	~60%
Validation data set	~20%
Test data set	~20%

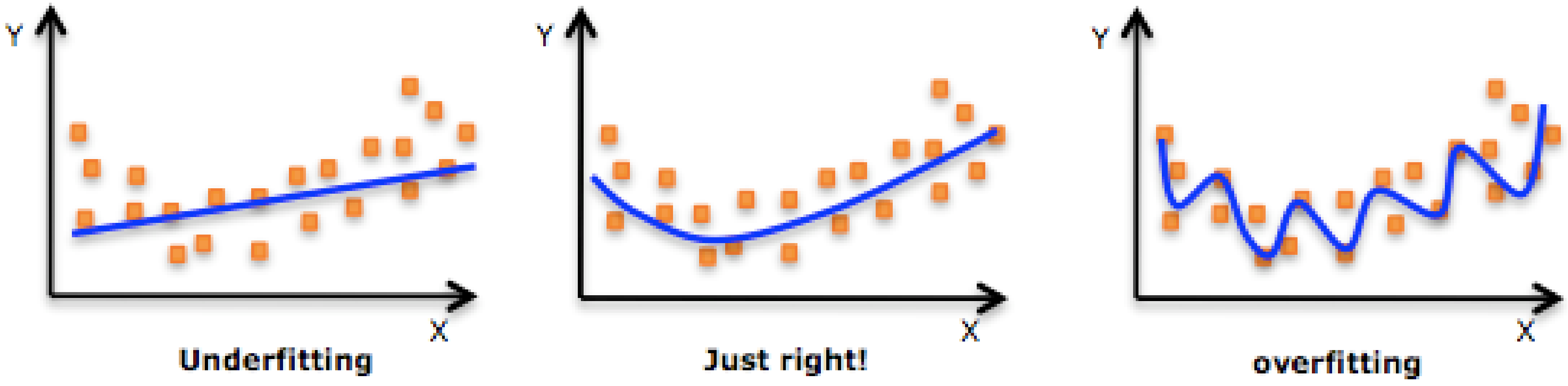
Overfitting and Underfitting

Overfitting

- The algorithm trains too long
- The line fits the training data too perfectly
- The line will not be good for predicting new data.

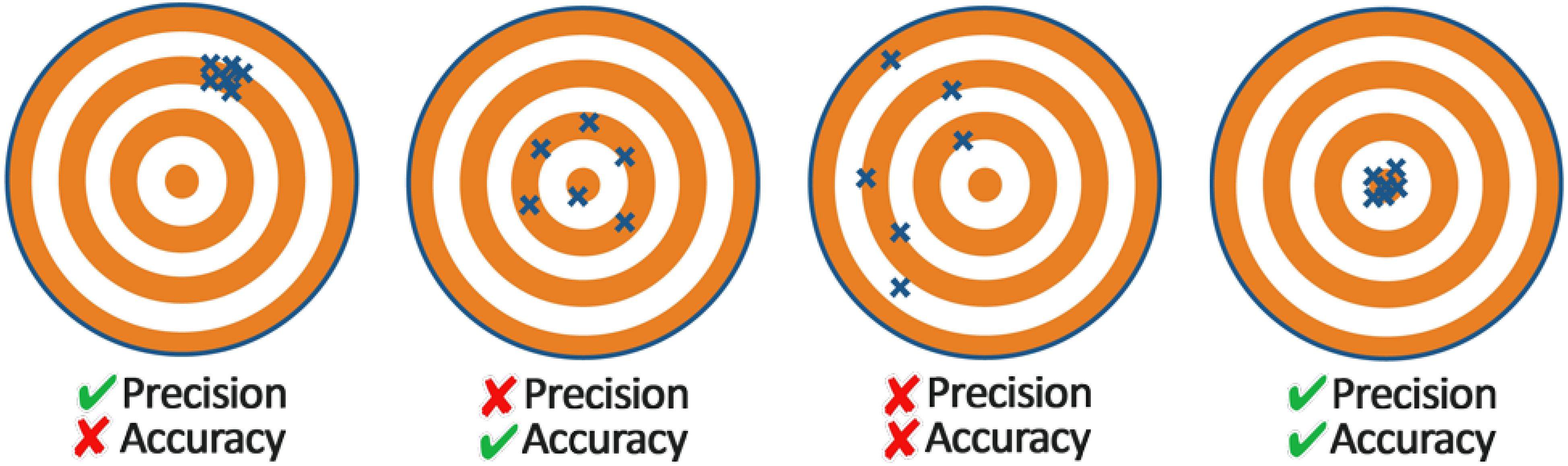
Underfitting

- The algorithm trains not long enough
- The curve established fits the data poorly
- The curve will not be good for predicting new data

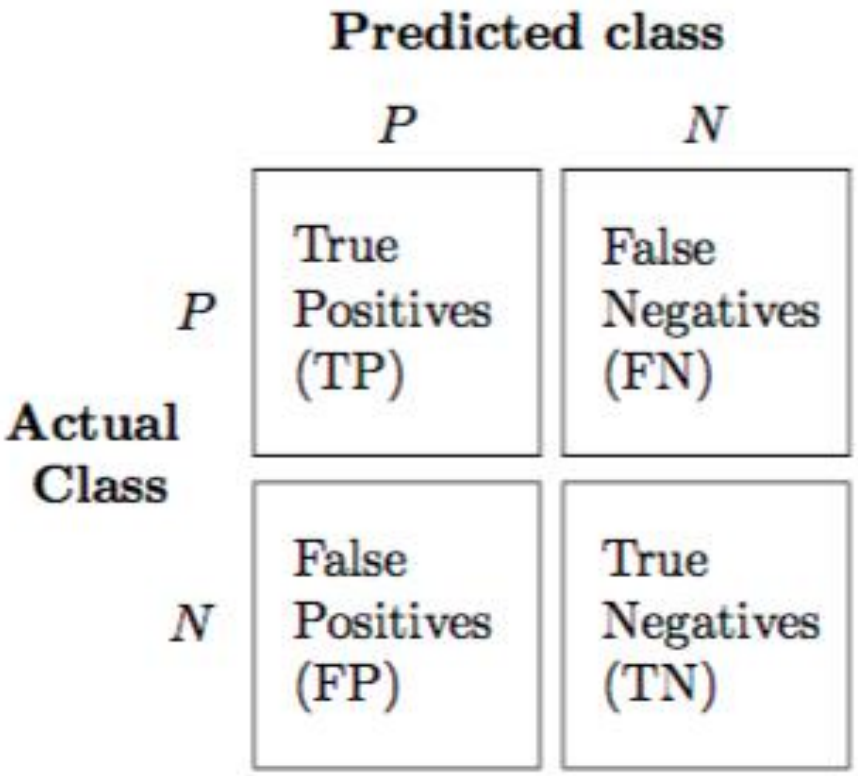


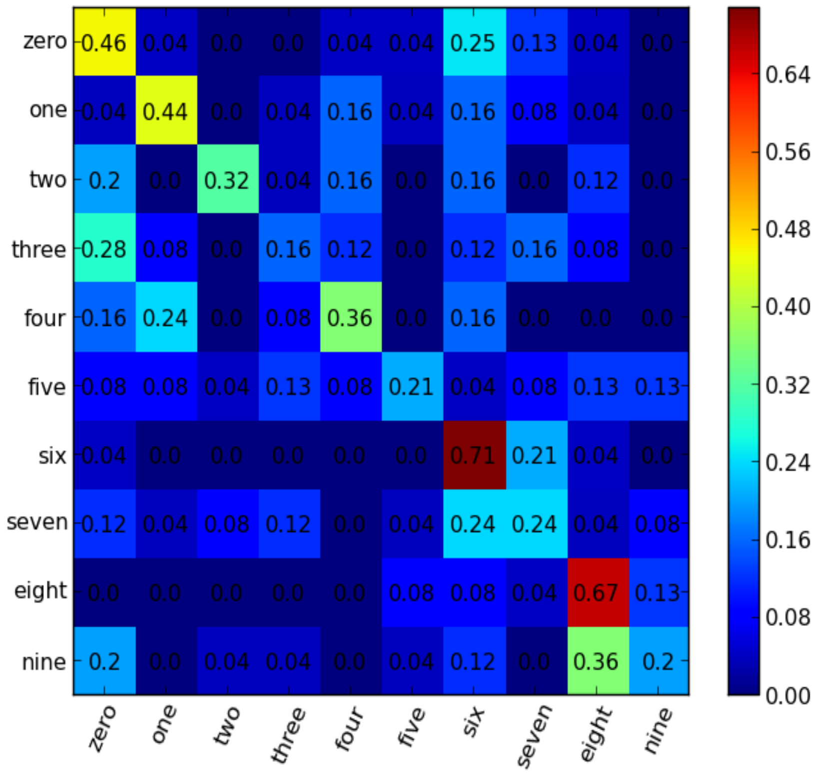
Presenting Results

Precision vs Accuracy



Confusion matrix





Accuracy, Precision, Recall

Accuracy is the ratio of correctly predicted observation to the total observations

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

$$\text{Recall} = \frac{TP}{TP+FN}$$

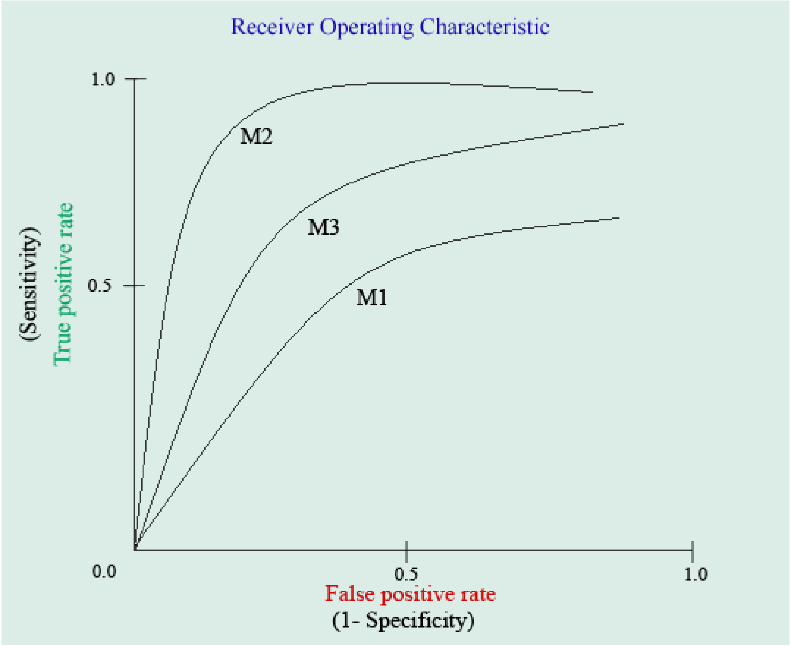
F1 score, RoC, AUC

F1 Score is the weighted average of Precision and Recall

$$F1 = \frac{2*(Recall*Precision)}{Recall+Precision}$$

The ROC plots the true positives against the false positives for binary classification problems with various different threshold settings

AUC is the area under the ROC curve. It can be used to compare different classifiers.
The bigger the AUC the better the classifier.



Supervised Learning

Labeled data

Predictive modeling

Unsupervised Learning

Unlabeled data

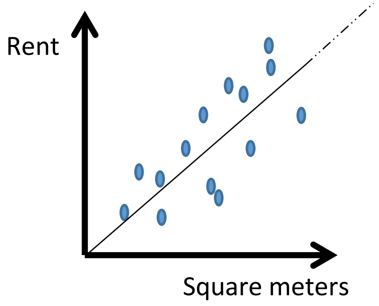
Descriptive modeling

Reinforcement learning

Unlabeled data

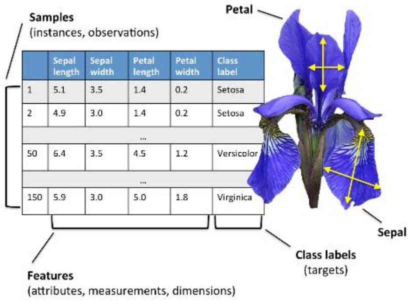
Observation data

Reward



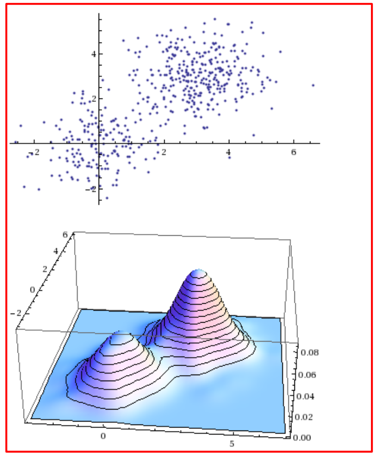
Regression

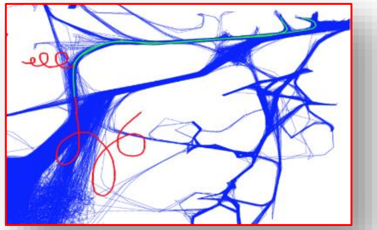
Classification



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Cluster detection





Anomaly detection

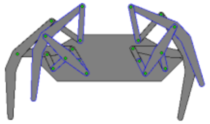
Learning strategies

Learning complex tasks

Driving

Cycling

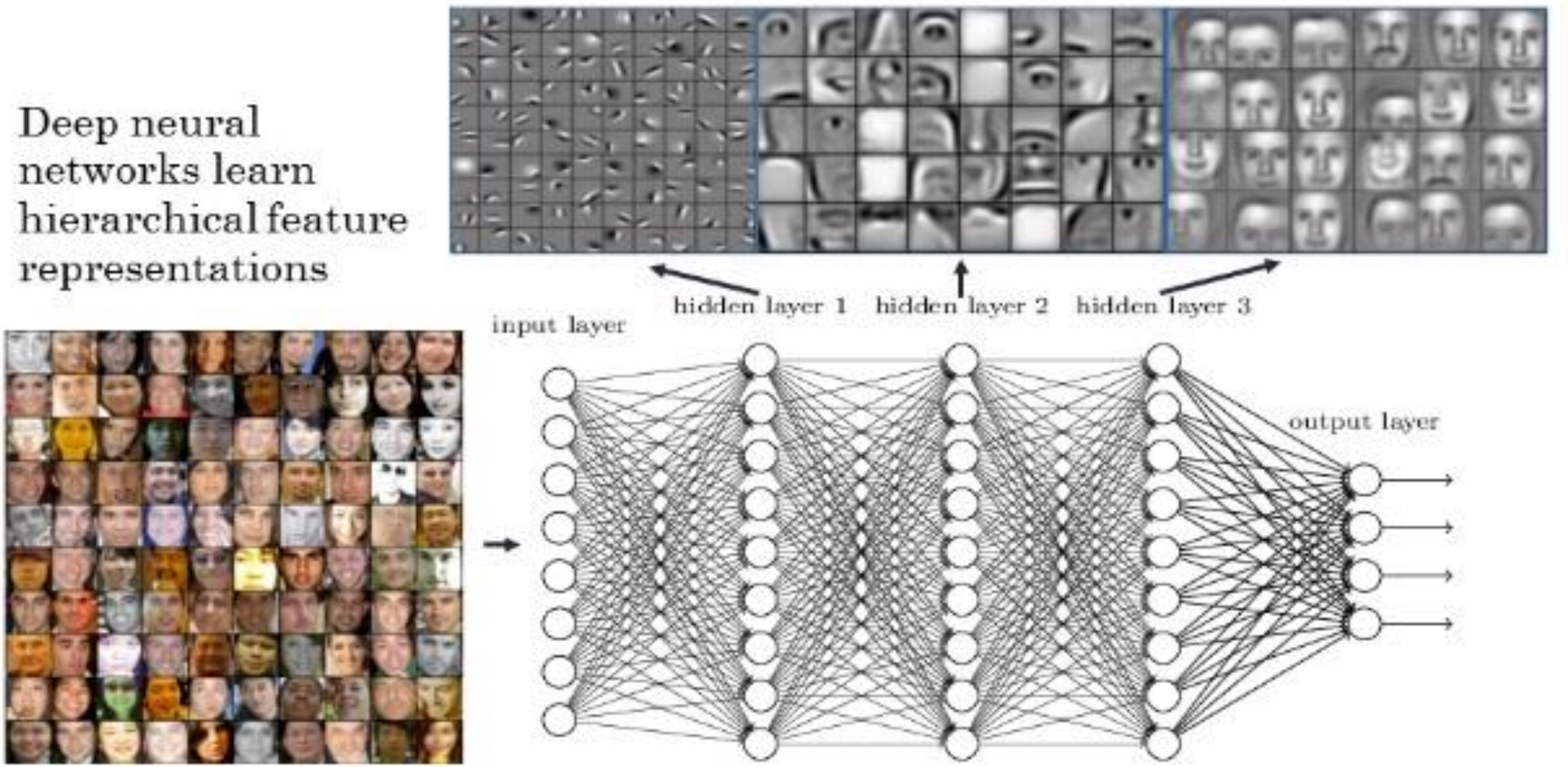
walking

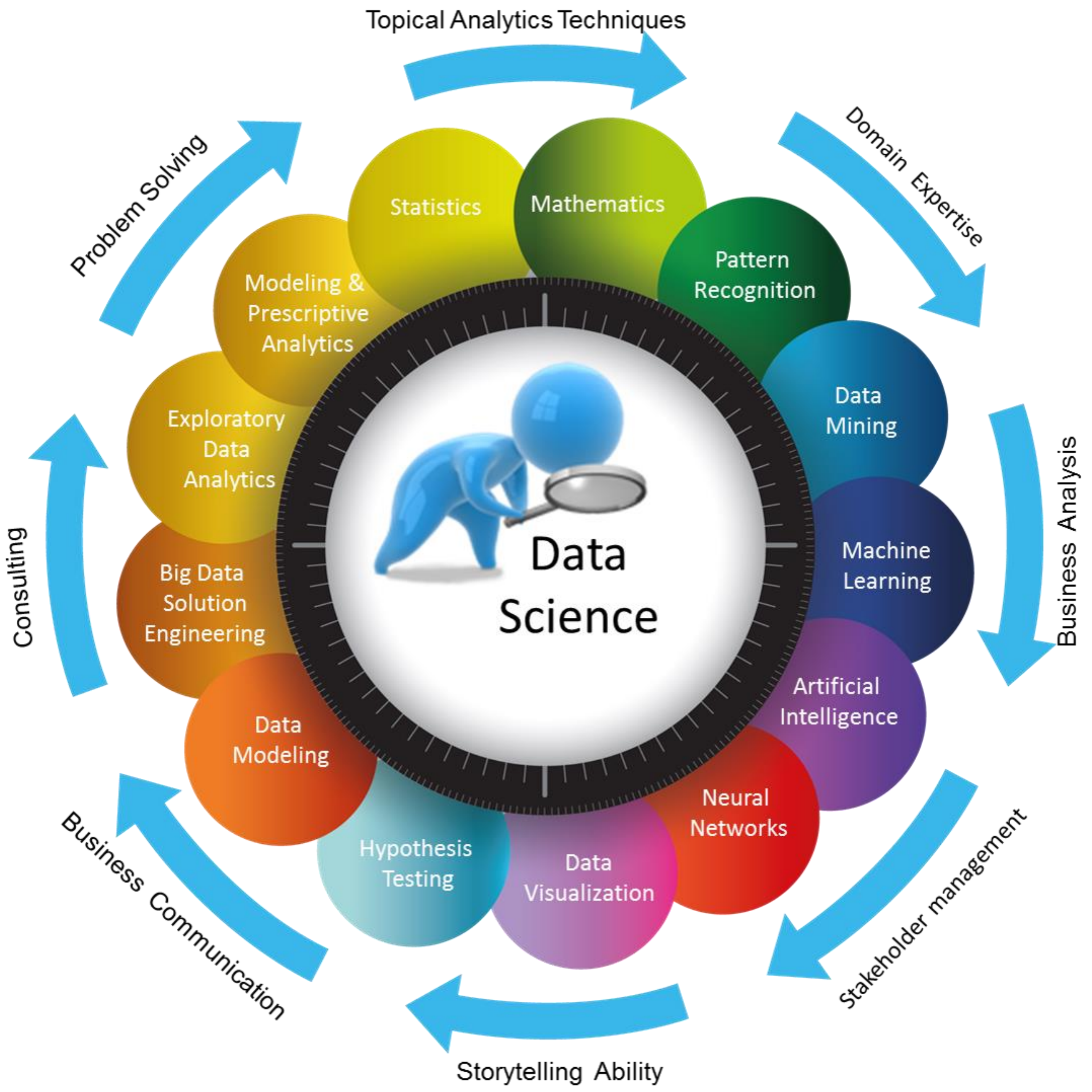


Some interesting types of neural networks

- Perceptron
- Multi layer perceptron
  - Feed Forward (forward propagation) neural networks (calculating results)
  - Back propagation (learning)
- Convolutional neural networks
  - Image processing
- Recurrent neural networks
  - Sequential data
- Relational Neural Networks
  - Relationships in data

Deep learning




Topical Analytics Techniques
Domain Expertise
Business Analysis
Stakeholder management
Storytelling Ability
Business Communication
Consulting
Problem Solving
Statistics
Mathematics
Pattern Recognition
Data Mining
Machine Learning
Artificial Intelligence
Neural Networks
Data Visualization
Hypothesis Testing
Data Modeling
Big Data Solution Engineering
Exploratory Data Analytics
Modeling & Prescriptive Analytics
Data Science

Thank you for your attention

Good Luck for the exam!