CIVIL-557

Decision Aid Methodologies In Transportation

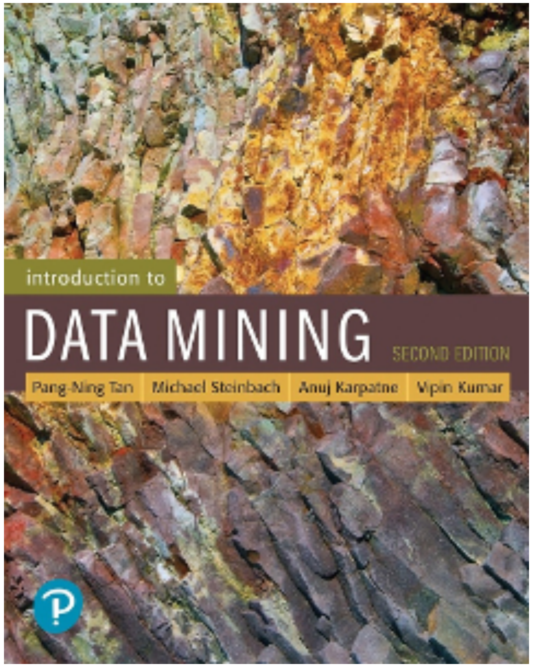
Lecture 12: Data Mining in Transport – Artificial Neural Networks

Nikola Obrenovic

Transport and Mobility Laboratory TRANSP-OR
École Polytechnique Fédérale de Lausanne EPFL

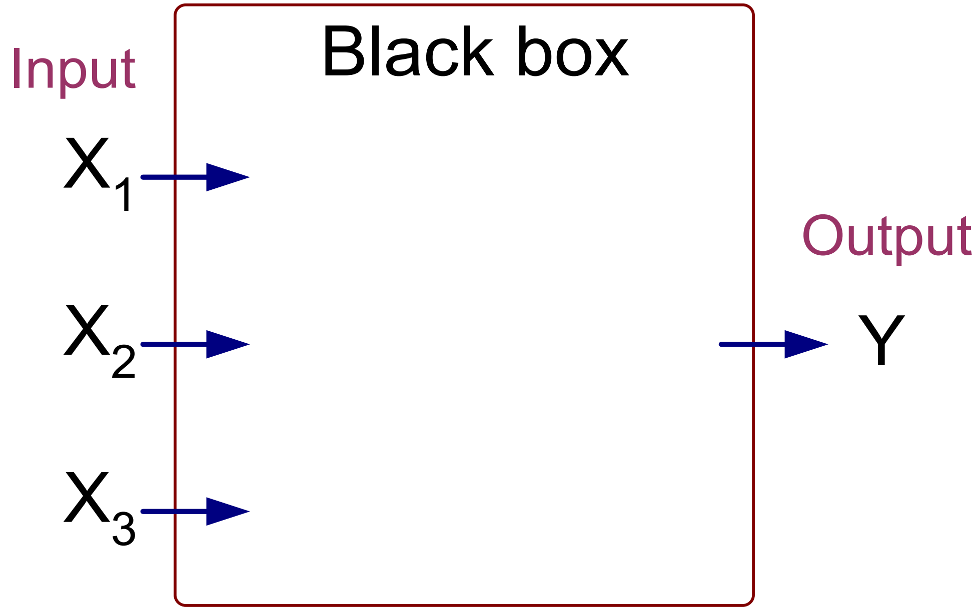
Acknowledgement

- The content of these slides has been partially taken over from the official slides accompanying the book: P.-N. Tan, M. Steinbach, A. Karpatne, V. Kumar: Introduction to Data Mining (2nd Edition)
- https://www-users.cs.umn.edu/~kumar001/dmbook/index.php



Artificial Neural Networks (ANN)

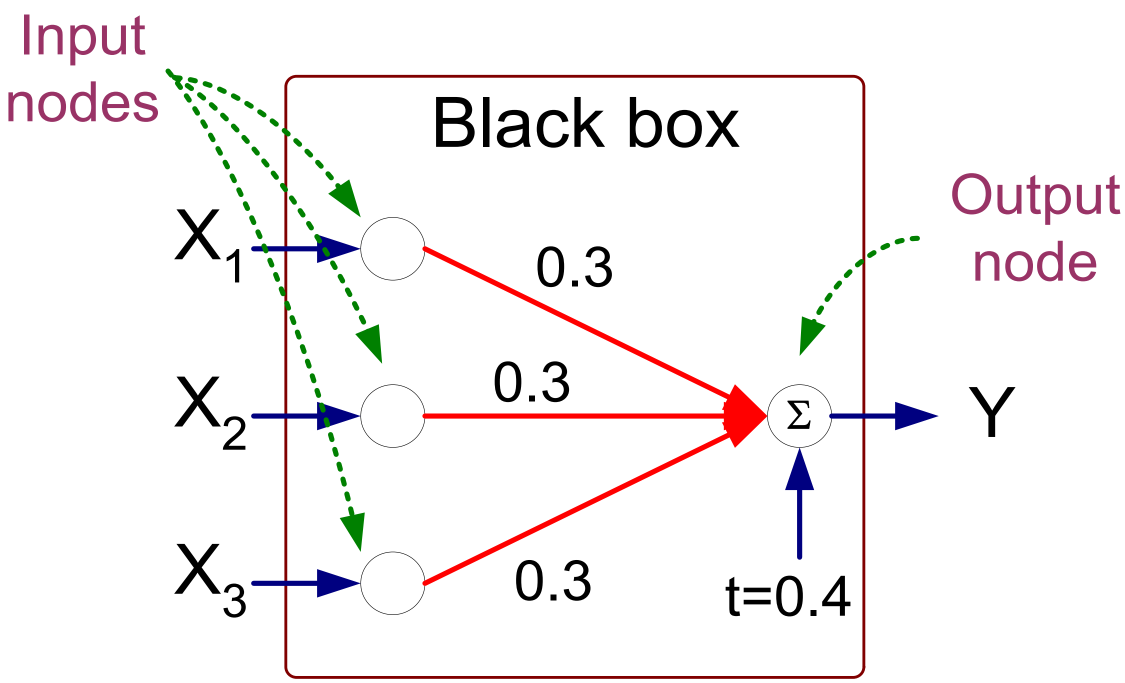
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

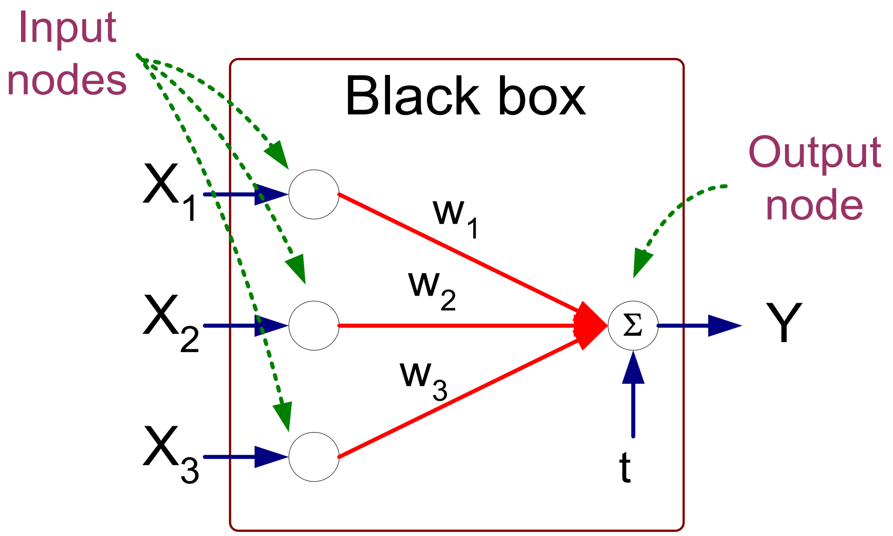


$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

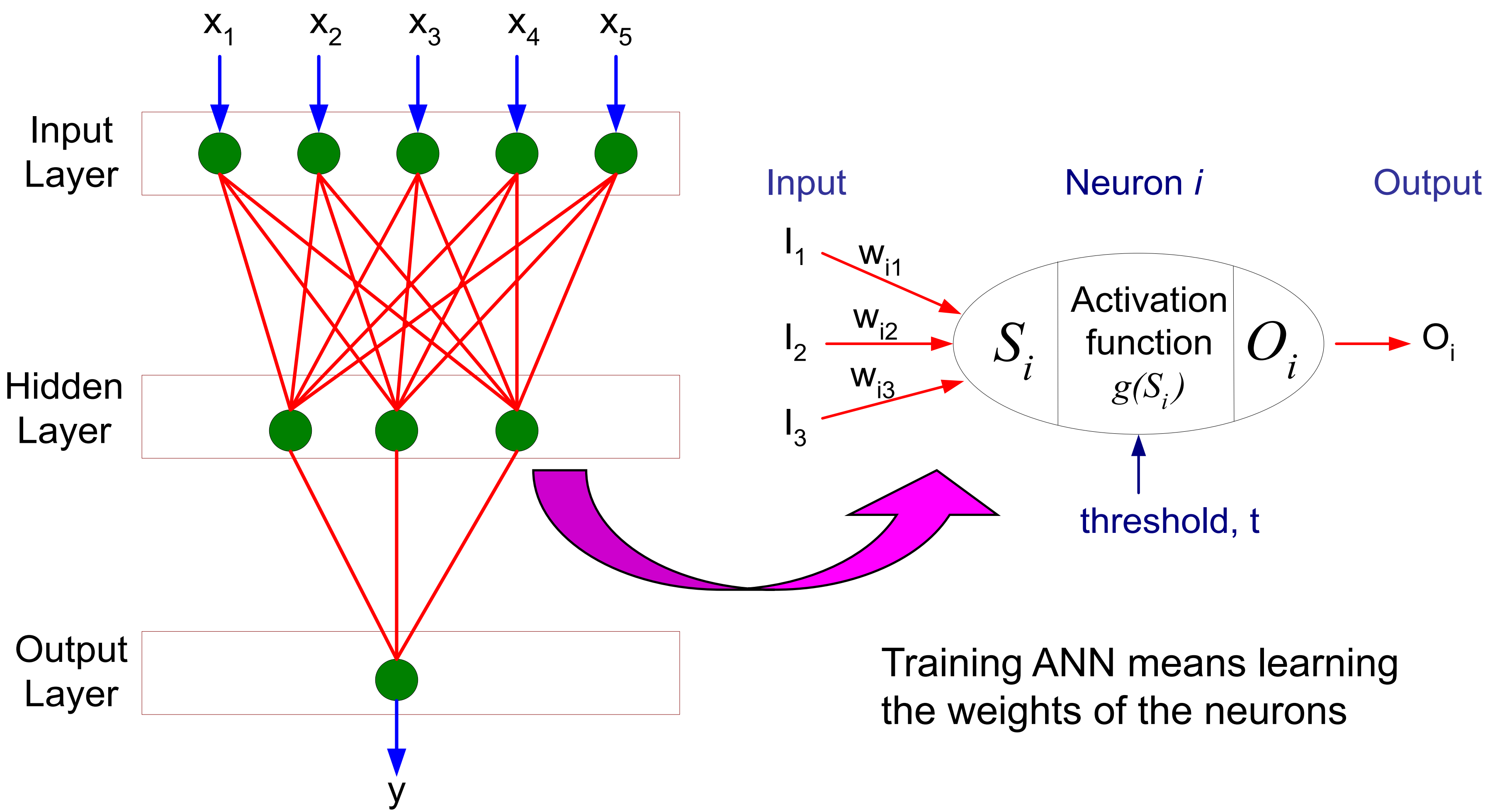


Perceptron Model

$$Y = sign(\sum_{i=1}^{d} w_i X_i - t)$$

$$= sign(\sum_{i=0}^{d} w_i X_i)$$

General Structure of ANN



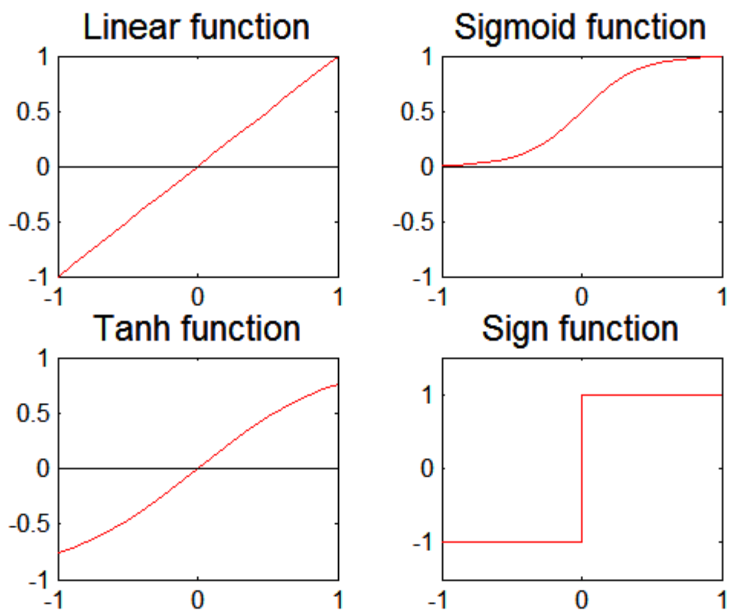
Training ANN means learning the weights of the neurons

Artificial Neural Networks (ANN)

- Various types of neural network topology
  - single-layered network (perceptron) versus multi-layered network
  - Feed-forward versus recurrent network

- Various types of activation functions (f)

$$Y = f(\sum_i w_i X_i)$$



Perceptron

- Single layer network
  - Contains only input and output nodes

- Activation function: f = sign(w•x)

- Applying model is straightforward

$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

  - $X_1 = 1$, $X_2 = 0$, $X_3 = 1$ => y = sign(0.2) = 1

Perceptron Learning Rule

- Initialize the weights (w_0, w_1, …, w_d)
- Repeat
  - For each training example (x_i, y_i)
    - Compute f(w, x_i)
    - Update the weights:

$$w^{(k+1)} = w^{(k)} + \lambda\left[y_i - f(w^{(k)}, x_i)\right]x_i$$

- Until stopping condition is met

Perceptron Learning Rule

- Weight update formula:

$$w^{(k+1)} = w^{(k)} + \lambda\left[y_i - f(w^{(k)}, x_i)\right]x_i \;\; ; \;\; \lambda : \text{learning rate}$$

- Intuition:
  - Update weight based on error: $e = \left[y_i - f(w^{(k)}, x_i)\right]$
  - If y=f(x,w), e=0: no update needed
  - If y>f(x,w), e>0: weight must be increased so that f(x,w) will increase
  - If y<f(x,w), e<0: weight must be decreased so that f(x,w) will decrease

Example of Perceptron Learning

$$w^{(k+1)} = w^{(k)} + \lambda\left[y_i - f(w^{(k)}, x_i)\right]x_i$$

$$Y = sign(\sum_{i=0}^{d} w_i X_i)$$

$$\lambda = 0.1$$

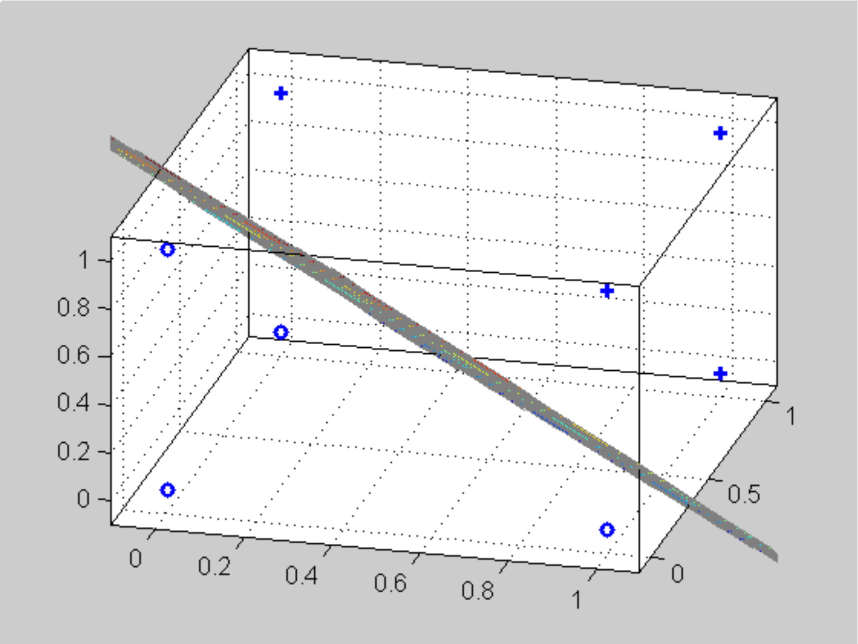
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

Epoch	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

Perceptron Learning Rule

- Since f(w,x) is a linear combination of input variables, decision boundary is linear

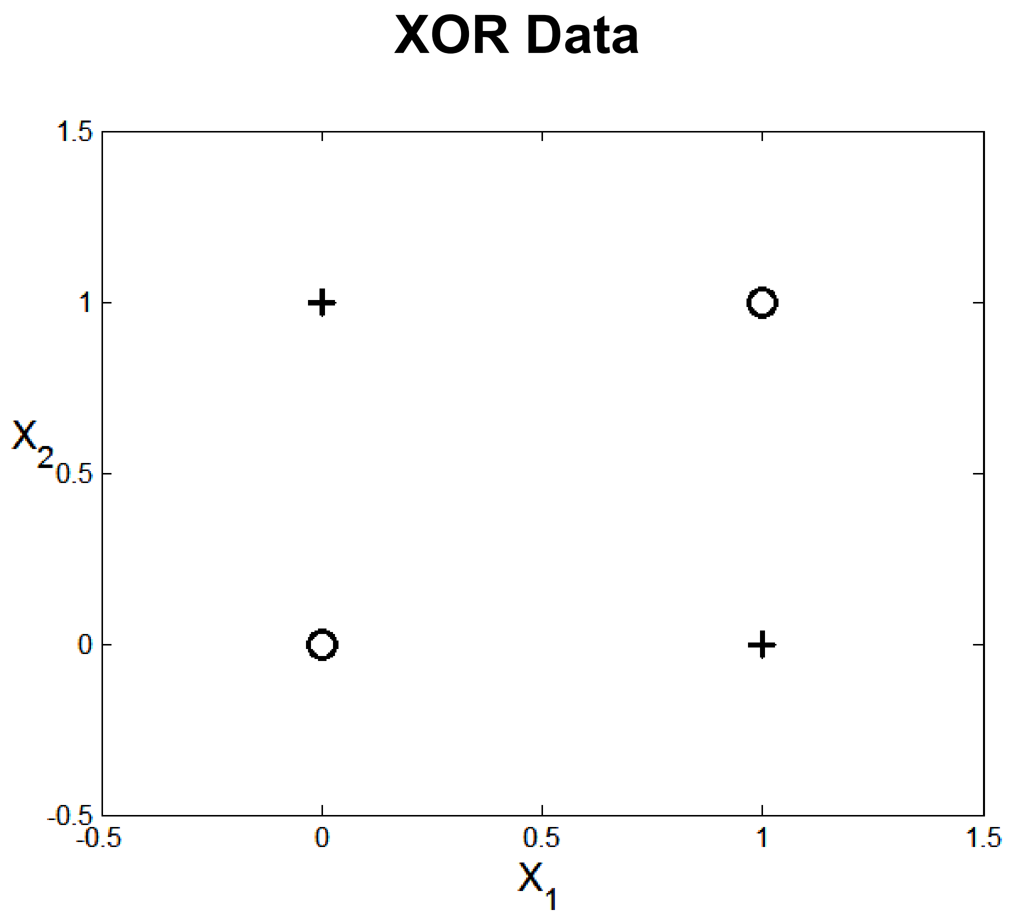


- For nonlinearly separable problems, perceptron learning algorithm will fail because no linear hyperplane can separate the data perfectly

Nonlinearly Separable Data

$$y = x_1 \oplus x_2$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

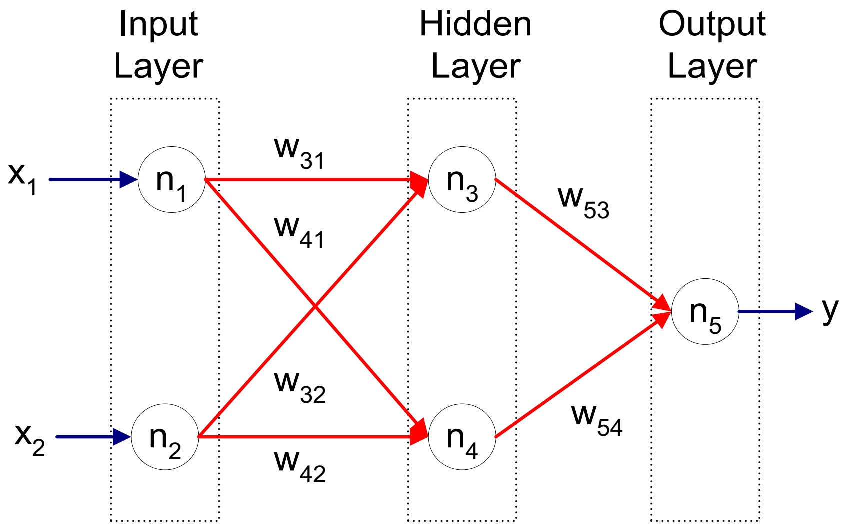


Multilayer Neural Network

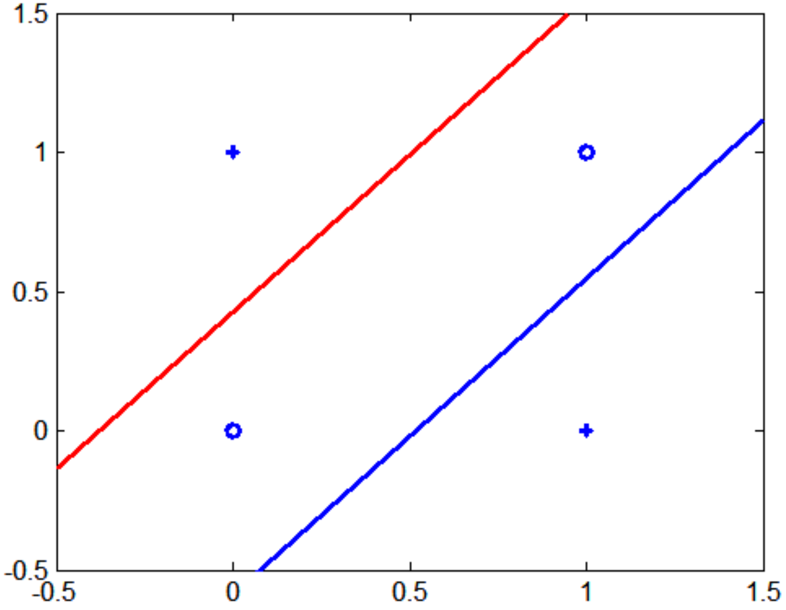
- Hidden layers
  - intermediary layers between input & output layers
- More general activation functions (sigmoid, linear, etc)

Multi-layer Neural Network

- Multi-layer neural network can solve any type of classification task involving nonlinear decision surfaces



XOR Data



Learning Multi-layer Neural Network

- Can we apply perceptron learning rule to each node, including hidden nodes?
  - Perceptron learning rule computes error term e = y-f(w,x) and updates weights accordingly
    - Problem: how to determine the true value of y for hidden nodes?
  - Approximate error in hidden nodes by error in the output nodes
    - Problem:
      - Not clear how adjustment in the hidden nodes affect overall error
      - No guarantee of convergence to optimal solution

Gradient Descent for Perceptron

- Weight update: $$w_j^{(k+1)} = w_j^{(k)} - \lambda \frac{\partial E}{\partial w_j}$$
- Error function: $$E = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - f(\sum_j w_j x_{ij})\right)^2$$
- Activation function f must be differentiable
- For sigmoid function:

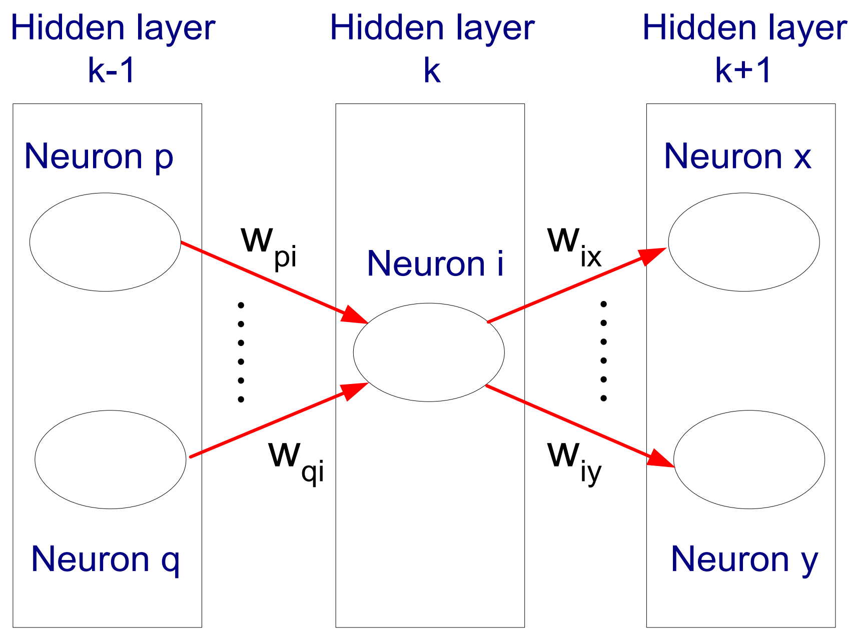
$$w_j^{(k+1)} = w_j^{(k)} + \lambda \sum_i (t_i - o_i) o_i (1 - o_i) x_{ij}$$

- Stochastic gradient descent (update the weight immediately)

Gradient Descent for MultiLayer NN

- For output neurons, weight update formula is the same as before (gradient descent for perceptron)

- For hidden neurons:



$$w_{pi}^{(k+1)} = w_{pi}^{(k)} + \lambda o_i (1 - o_i) \sum_{j \in \Phi_i} \delta_j w_{ij} x_{pi}$$

$$\text{Output neurons}: \delta_j = o_j (1 - o_j)(t_j - o_j)$$

$$\text{Hidden neurons}: \delta_j = o_j (1 - o_j) \sum_{k \in \Phi_j} \delta_k w_{jk}$$

Design Issues in ANN

- Number of nodes in input layer
  - One input node per binary/continuous attribute
  - k or $\log_2 k$ nodes for each categorical attribute with k values
- Number of nodes in output layer
  - One output for binary class problem
  - k or $\log_2 k$ nodes for k-class problem
- Number of nodes in hidden layer
- Initial weights and biases

Characteristics of ANN

- Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large
- Gradient descent may converge to local minimum
- Model building can be very time consuming, but testing can be very fast
- Can handle redundant attributes because weights are automatically learnt
- Sensitive to noise in training data
- Difficult to handle missing attributes

Recent Noteworthy Developments in ANN

- Google Brain project
  - Learned the concept of a ‘cat’ by looking at unlabeled pictures from YouTube
  - One billion connection network

Case study

- M. S. Dougherty, M. R. Cobbett: Short-term inter-urban traffic forecasts using neural networks, https://doi.org/10.1016/S0169-2070(96)00697-8
  - Short-term forecast of traffic flow , speed and occupancy
    - Also used as input data
  - Challenges:
    - Selection of input features
    - Neural network size impractical for real-time use
  - Input features selection – elasticity analysis
- What improved algorithm results to a great extent?
- How is the *curse of dimensionality* defined in this paper?

Confusion Matrix

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Class Imbalance Problem

- Lots of classification problems where the classes are skewed (more records from one class than another)
  - Credit card fraud
  - Intrusion detection
  - Defective products in manufacturing assembly line

Challenges

- Evaluation measures such as accuracy is not well-suited for imbalanced class
- Detecting the rare class is like finding needle in a haystack

Problem with Accuracy

- Consider a 2-class problem
  - Number of Class NO examples = 990
  - Number of Class YES examples = 10

- If a model predicts everything to be class NO, accuracy is 990/1000 = 99 %
  - This is misleading because the model does not detect any class YES example
  - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

Alternative Measures

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F-measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

Alternative Measures

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	10	0
Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2*1*0.5}{1+0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

Alternative Measures

	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS Class=Yes	10	0
ACTUAL CLASS Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2*1*0.5}{1+0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS Class=Yes	1	9
ACTUAL CLASS Class=No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F - measure (F)} = \frac{2*0.1*1}{1+0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

Alternative Measures

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	40	10
Class=No	10	40

$$\text{Precision (p)} = 0.8$$

$$\text{Recall (r)} = 0.8$$

$$\text{F - measure (F)} = 0.8$$

$$\text{Accuracy} = 0.8$$

Alternative Measures

	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS: Class=Yes	40	10
ACTUAL CLASS: Class=No	10	40

$$\text{Precision (p)} = 0.8$$
$$\text{Recall (r)} = 0.8$$
$$\text{F - measure (F)} = 0.8$$
$$\text{Accuracy} = 0.8$$

	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS: Class=Yes	40	10
ACTUAL CLASS: Class=No	1000	4000

$$\text{Precision (p)} =\sim 0.04$$
$$\text{Recall (r)} = 0.8$$
$$\text{F - measure (F)} =\sim 0.08$$
$$\text{Accuracy} =\sim 0.8$$

Measures of Classification Performance

	PREDICTED CLASS		
		Yes	No
ACTUAL CLASS	Yes	TP	FN
	No	FP	TN

α is the probability that we reject the null hypothesis when it is true. This is a Type I error or a false positive (FP).

β is the probability that we accept the null hypothesis when it is false. This is a Type II error or a false negative (FN).

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

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

Alternative Measures

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	40	10
Class=No	10	40

$$\text{Precision (p)} = 0.8$$
$$\text{TPR} = \text{Recall (r)} = 0.8$$
$$\text{FPR} = 0.2$$
$$\text{F - measure (F)} = 0.8$$
$$\text{Accuracy} = 0.8$$

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	40	10
Class=No	1000	4000

$$\text{Precision (p)} =\sim 0.04$$
$$\text{TPR} = \text{Recall (r)} = 0.8$$
$$\text{FPR} = 0.2$$
$$\text{F - measure (F)} =\sim 0.08$$
$$\text{Accuracy} =\sim 0.8$$

Main references

- P.-N. Tan, M. Steinbach, A. Karpatne, V. Kumar: Introduction to Data Mining, 2nd Edition, 2006, Pearson Education Inc.