Uge 41

Ex 1

Exercise 1. k-Nearest Neighbors: Prediction

Suppose you are trying to predict a continuous response y to an input x and that you are given the set of training data $D = [(x_1, y_1), \dots, (x_{11}, y_{11})]$ reported and plotted in Figure 1.

$$D = \begin{bmatrix} (8, & 8.31) \\ (14, & 5.56) \\ (0, & 12.1) \\ (6, & 7.94) \\ (3, & 10.09) \\ (2, & 9.89) \\ (4, & 9.52) \\ (7, & 7.77) \\ (8, & 7.51) \\ (11, & 8.0) \\ (8, & 10.59) \end{bmatrix}$$

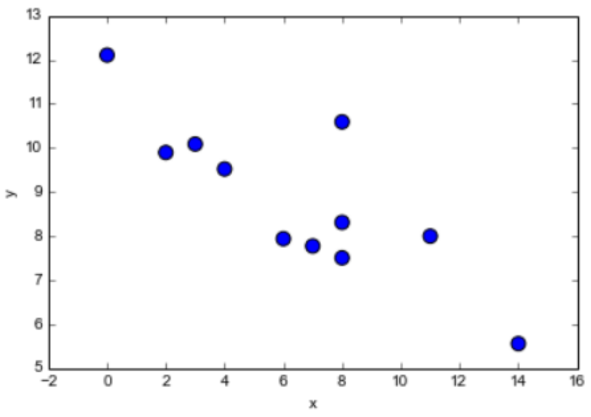


Figure 1: The data for Exercise 1.

Using 5-nearest neighbors, what would be the prediction on an new input $x = 8$?
What form of learning is this exercise about?

- Supervised learning, regression
- Supervised learning, classification
- Unsupervised learning
- Reinforcement learning

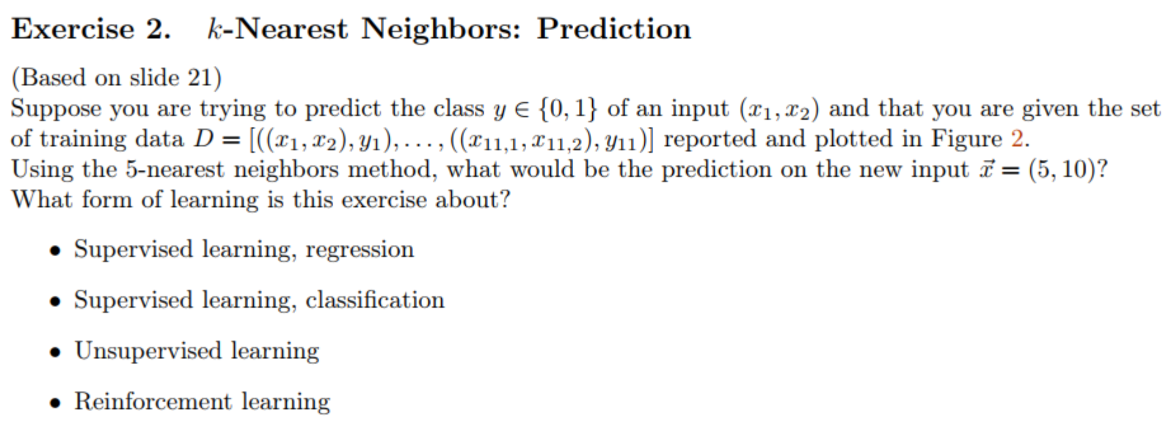
By inspection we find the 5 nearest neighbors to be:

$$(8, 8.31), (8, 7.51), (8, 10.59), (7, 7.77), (6, 7.94)$$

$$\hat{y} = \frac{1}{k} \cdot \sum_{i | x_i \in N_k(x)} y_i = \frac{1}{5} \cdot (8{,}31 + 7{,}51 + 10{,}59 + 7{,}77 + 7{,}94) = 8{,}424$$

This is supervised learning, regression

Ex 2



Exercise 2. k-Nearest Neighbors: Prediction

(Based on slide 21)
Suppose you are trying to predict the class $y \in \{0, 1\}$ of an input (x_1, x_2) and that you are given the set of training data $D = [((x_1, x_2), y_1), \ldots, ((x_{11,1}, x_{11,2}), y_{11})]$ reported and plotted in Figure 2.
Using the 5-nearest neighbors method, what would be the prediction on the new input $\vec{x} = (5, 10)$?
What form of learning is this exercise about?

- Supervised learning, regression
- Supervised learning, classification
- Unsupervised learning
- Reinforcement learning

$$D = \begin{bmatrix} ((10, & 2), & 1) \\ ((15, & 2), & 1) \\ ((6, & 11), & 1) \\ ((2, & 3), & 0) \\ ((5, & 15), & 1) \\ ((5, & 14), & 1) \\ ((10, & 1), & 0) \\ ((1, & 6), & 0) \\ ((17, & 19), & 1) \\ ((15, & 13), & 0) \\ ((19, & 9), & 0) \end{bmatrix}$$

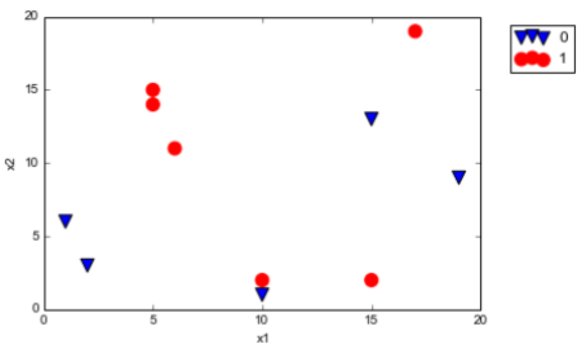


Figure 2: The data for Exercise 2.

We calculate the Euclidean distance for each point:

$$d(\vec{x_i}, \vec{x}) = \sqrt{\sum_j (x_{ij} - x_j)^2}$$

Like this for all points:

$$d(\vec{x}, \vec{x_1})\sqrt{(5-10)^2 + (10-2)^2} \approx 9{,}4339811321$$

x1	**x2**	**y**	**distance**
10	2	1	9,433981132
15	2	1	12,80624847
6	11	1	1,414213562
2	3	0	7,615773106
5	15	1	5
5	14	1	4
10	1	0	10,29563014
1	6	0	5,656854249
17	19	1	15
15	13	0	10,44030651
19	9	0	14,03566885

We find that the 5 nearest points are:

x1	**x2**	**y**	**distance**
6	11	1	1,414213562
5	14	1	4
5	15	1	5
1	6	0	5,656854249
2	3	0	7,615773106

We find that there are more 1's than 0's, meaning this is a 1.

This is a supervised learning, classification problem.

Ex 3

Exercise 3. Linear Regression: Prediction

(Based on slides 24-26)
As in Exercise 1. you are trying to predict a response y to an input x and you are given the same set of training data $D = [(x_1, y_1), \ldots, (x_{11}, y_{11})]$, also reported and plotted in Figure 3. However, now you want to use a linear regression model to make your prediction. After training, your model looks as follows:

$$g(x) = -0.37x + 11.22$$

The corresponding function is depicted in red in Figure 3. What is your prediction $\hat{y}$ for the new input $x = 8$?

$$D = \begin{bmatrix} (8, & 8.31) \\ (14, & 5.56) \\ (0, & 12.1) \\ (6, & 7.94) \\ (3, & 10.09) \\ (2, & 9.89) \\ (4, & 9.52) \\ (7, & 7.77) \\ (8, & 7.51) \\ (11, & 8.0) \\ (8, & 10.59) \end{bmatrix}$$

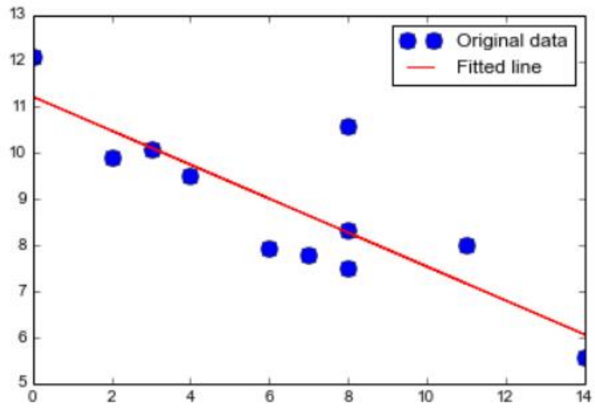
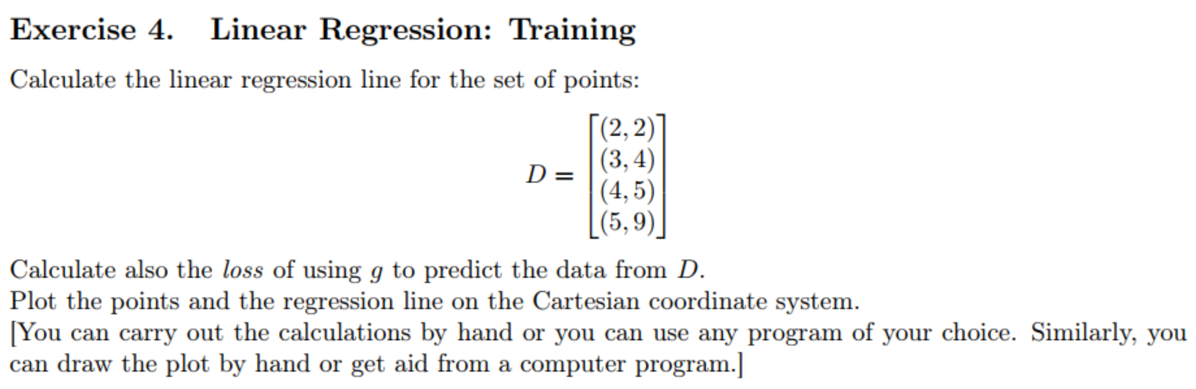


Figure 3: The data for Exercise 3.

$$g(x) \coloneqq -0{,}37 \cdot x + 11{,}22$$

$$g(8) \approx 8{,}26$$

Ex 4



Exercise 4. Linear Regression: Training

Calculate the linear regression line for the set of points:

$$D = \begin{bmatrix}(2,2)\\(3,4)\\(4,5)\\(5,9)\end{bmatrix}$$

Calculate also the *loss* of using g to predict the data from D.
Plot the points and the regression line on the Cartesian coordinate system.
[You can carry out the calculations by hand or you can use any program of your choice. Similarly, you can draw the plot by hand or get aid from a computer program.]

From slides we get:

$$a = \frac{\Sigma\left((x_j - \bar{x})(y_j - \bar{y})\right)}{\Sigma\left((x_j - \bar{x})^2\right)}$$

$$b = \bar{y} - a\bar{x}$$

$$\bar{x} = \frac{1}{m}\Sigma(x_j)$$

$$\bar{y} = \frac{1}{m}\Sigma(y_j)$$

We calc:

$$\bar{x} = \frac{1}{4}\cdot(2+3+4+5) = 3{,}5$$

$$\bar{y} = \frac{1}{4}\cdot(2+4+5+9) = 5$$

Slet definitioner:

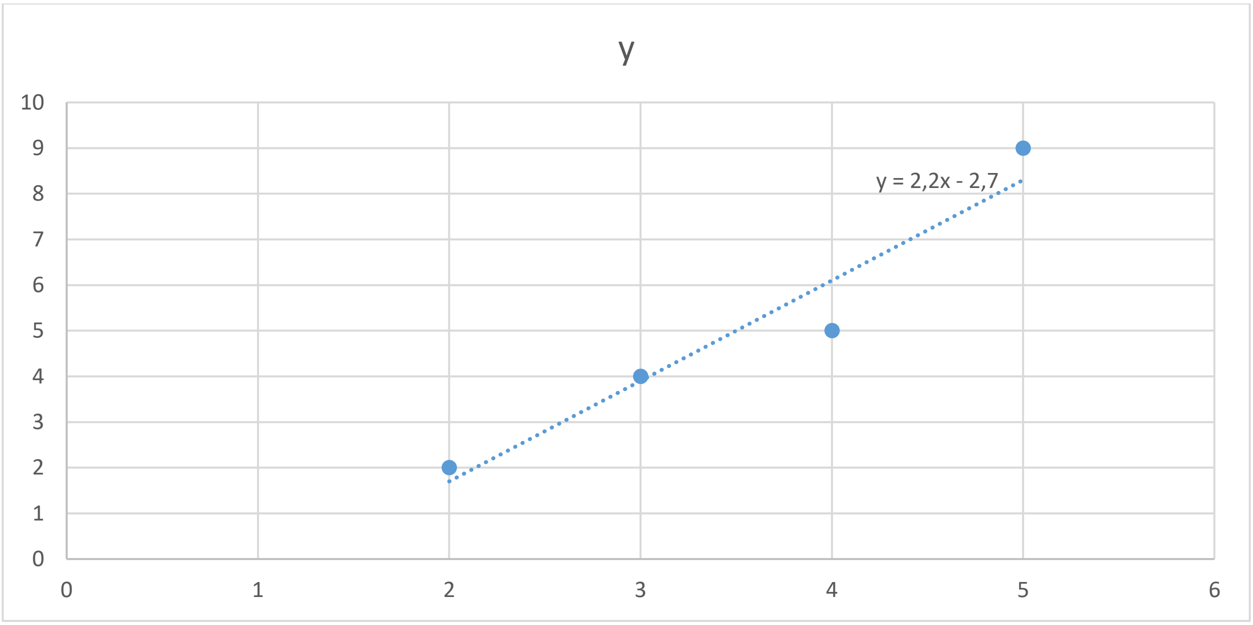
$$\Sigma\left((x_j - \bar{x})(y_j - \bar{y})\right) = (2-3{,}5)\cdot(2-5) + (3-3{,}5)\cdot(4-5) + (4-3{,}5)\cdot(5-5) + (5-3{,}5)\cdot(9-5) = 11$$

$$\Sigma\left((x_j - \bar{x})^2\right) = (2-3{,}5)^2 + (3-3{,}5)^2 + (4-3{,}5)^2 + (5-3{,}5)^2 = 5$$

$$a = \frac{\Sigma\left((x_j - \bar{x})(y_j - \bar{y})\right)}{\Sigma\left((x_j - \bar{x})^2\right)} = \frac{11}{5} \approx 2{,}2$$

$$b = \bar{y} - a\bar{x} = 5 - \frac{11}{5}\cdot 3{,}5 \approx -2{,}7$$

$$\hat{L} = \left(2 - \frac{11}{5} \cdot 2 - (-2{,}7)\right)^2 + \left(4 - \frac{11}{5} \cdot 3 - (-2{,}7)\right)^2 + \left(5 - \frac{11}{5} \cdot 4 - (-2{,}7)\right)^2 + \left(9 - \frac{11}{5} \cdot 5 - (-2{,}7)\right)^2 \approx 1{,}8$$



Can also use python to do this see Marcos solution.

Ex 5

Exercise 5. Logical Functions and Perceptrons

Perceptrons can be used to compute the elementary logical functions that we usually think of as underlying computation. Examples of these functions are AND, OR and NOT.

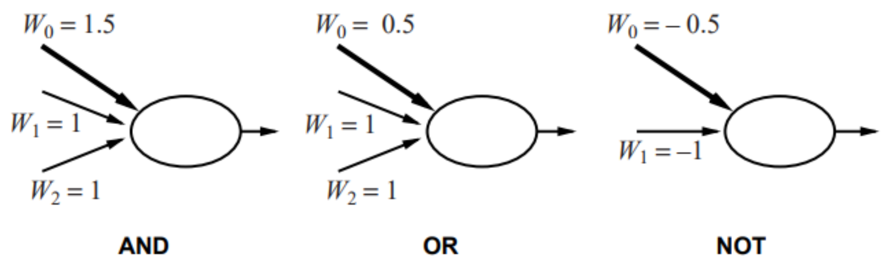


Figure 4: Logical functions and perceptrons. Exercise Exercise 5..

In class, we carried out the verification that the left most perceptron in Figure 4 is a correct representation of the AND operator.

- Verify that the perceptrons given for the OR and NOT cases in Figure 4 are also correct representations of the corresponding logical functions.
- Design a perceptron that implements the logical function NAND.

Later in this Assignment Sheet we will see that there are also Boolean functions that cannot be represented by a single perceptron alone.

$$\sum_{j=0}^{p} w_{ji}x_j > 0 \quad \text{or} \quad \vec{w} \cdot \vec{x} > 0$$

AND:

$a_0 \cdot 1.5$	$a_1 \cdot 1$	$a_2 \cdot 1$	O
-1	0	0	$-1{,}5 + 0 + 0 \rightarrow 0$
-1	0	1	$-1{,}5 + 0 + 1 \rightarrow 0$
-1	1	0	$-1{,}5 + 1 + 0 \rightarrow 0$
-1	1	1	$-1{,}5 + 1 + 1 \rightarrow 1$

OR:

$a_0 \cdot 0.5$	$a_1 \cdot 1$	$a_2 \cdot 1$	O
-1	0	0	$-0{,}5 + 0 + 0 \rightarrow 0$
-1	0	1	$-0{,}5 + 0 + 1 \rightarrow 1$
-1	1	0	$-0{,}5 + 1 + 0 \rightarrow 1$
-1	1	1	$-0{,}5 + 1 + 1 \rightarrow 1$

Not:

$a_0 \cdot -0.5$	$a_1 \cdot -1$	O
-1	0	$0{,}5 + 0 \rightarrow 1$
-1	1	$0{,}5 - 1 \rightarrow 0$

NAND:

$a_0 \cdot -2$	$a_1 \cdot -1$	$a_2 \cdot -1$	O
-1	0	0	$2 + 0 + 0 \rightarrow 1$
-1	0	1	$2 + 0 - 1 \rightarrow 1$
-1	1	0	$2 - 1 + 0 \rightarrow 1$
-1	1	1	$2 - 1 - 1 \rightarrow 0$

or like Marco -3, -2, -2

Ex 6

Exercise 6. Multilayer Perceptrons

Determine the truth table of the Boolean function represented by the perceptron in Figure 5:

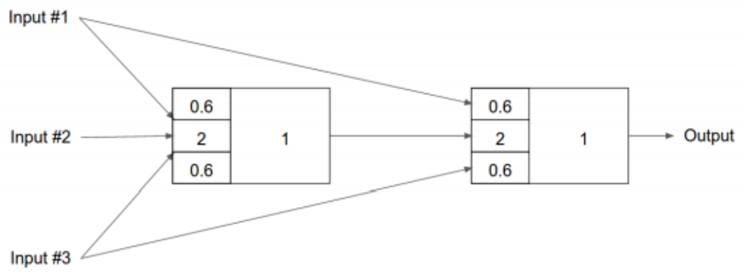


Figure 5: . The multilayer perceptron of Exercise 6.

x	y	z	O1	O2
0	0	0	$0{,}6 \cdot 0 + 2 \cdot 0 + 0{,}6 \cdot 0 = 0{,}0$	$0{,}6 \cdot 0 + 2 \cdot 0 + 0{,}6 \cdot 0 = 0$
0	0	1	$0{,}6 \cdot 0 + 2 \cdot 0 + 0{,}6 \cdot 1 = 0{,}6$	$0{,}6 \cdot 0 + 2 \cdot 0 + 0{,}6 \cdot 0 = 0$
0	1	0	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2{,}0$	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2$
0	1	1	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 1 = 2{,}6$	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2$
1	0	0	$0{,}6 \cdot 1 + 2 \cdot 0 + 0{,}6 \cdot 0 = 0{,}6$	$0{,}6 \cdot 0 + 2 \cdot 0 + 0{,}6 \cdot 0 = 0$
1	0	1	$0{,}6 \cdot 1 + 2 \cdot 0 + 0{,}6 \cdot 1 = 1{,}2$	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2$
1	1	0	$0{,}6 \cdot 1 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2{,}6$	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2$
1	1	1	$0{,}6 \cdot 1 + 2 \cdot 1 + 0{,}6 \cdot 1 = 3{,}2$	$0{,}6 \cdot 0 + 2 \cdot 1 + 0{,}6 \cdot 0 = 2$

Ex 7

Exercise 7. Feed-Forward Neural Networks: Single Layer Perceptron

Determine the parameters of a single perceptron (that is, a neuron with step function) that implements the majority function: for n binary inputs the function outputs a 1 only if more than half of its inputs are 1.

$$W_0 = \frac{n}{2}$$

$$W_n = 1$$

Eg. $n = 3$

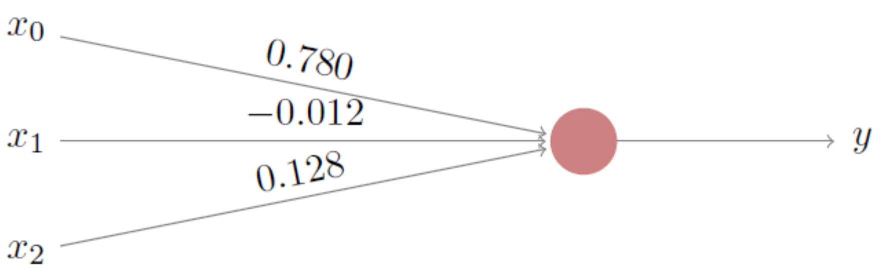
$$-1 \cdot W_0 = -1{,}5$$

Kun når mere end 2 er 1, vil vi komme over 0.

Ex 8

Exercise 8. Single Layer Neural Networks: Prediction

In Exercise 2. we predicted the class $y \in \{0, 1\}$ of an input (x_1, x_2) with the 5-nearest neighbors method using the data from set D. We used those data to train a single layer neural network for the same task. The result is depicted in Figure 6.



- Calculate the prediction of the neural network for the new input $\vec{x} = (5, 10)$. Assume a step function as activation function in the unit (which is therefore a perceptron).
- Calculate the prediction of the neural network for the new input $\vec{x} = (5, 10)$. Assume a sigmoid function as activation function in the unit (which is therefore a sigmoid neuron).
- Compare the results at the previous two points against the result in Exercise 2. Are they all consistent? Is this expected to be always the case? Which one is right?
- In binary classification, the loss can be defined as the number of mispredicted cases. Calculate the loss for the network under the two different activation functions. Which one performs better according to the loss?
- Derive and draw in the plot of Exercise 2. the decision boundaries between 0s and 1s that is implied by the perceptron and the sigmoid neuron. [See Section 2.1.3 of the Lecture Notes.] Are the points linearly separable?

- perceptron

$$-0{,}012 \cdot 5 + 0{,}128 \cdot 10 - 0{,}780 \approx 0{,}44$$

since larger than 0 we get 1.

- sigmoid:

$$\frac{1}{1 + e^{-(-0{,}012 \cdot 5 + 0{,}128 \cdot 10 - 0{,}780)}} \approx 0{,}608$$

since larger than 0.5 we get 1.

$$e^2 \approx 7{,}39$$

- comparison

Solution

The three methods return the same result in this case but they could return different results, in particular the k-nearest neighbors can be different from the single neuron cases. It is impossible to say who is right, because we do not know the actual response to x.

- loss, and comparison:

The training error is defined as the number of wrong predictions. We do the calculations above for all points and compare them to the actual classes:

x1	x2	class	perception	sigmoid	class-p	class-s
10	2	1	-0,644	0,344343	0	0
15	2	1	-0,704	0,330926	0	0
6	11	1	0,556	0,635527	1	1
2	3	0	-0,42	0,396517	0	0
5	15	1	1,08	0,746494	1	1
5	14	1	0,952	0,721517	1	1
10	1	0	-0,772	0,316047	0	0
1	6	0	-0,024	0,494	0	0
17	19	1	1,448	0,80969	1	1
15	13	0	0,704	0,669074	1	1
19	9	0	0,144	0,535938	1	1

We observe that both two types of neurons perform in the same way: they predict 4 cases wrong and 7 right. The training error defined as the number of wrong predictions is 4.

- derive and draw

perceptron:

$$-0{,}012 \cdot x_1 + 0{,}128 \cdot x_2 - 0{,}780 = 0$$

$\Updownarrow$ *Ligningen løses for x_2 vha. CAS-værktøjet WordMat.*

$$x_2 = 0{,}0938 \cdot x_1 + 6{,}09$$

sigmoid:

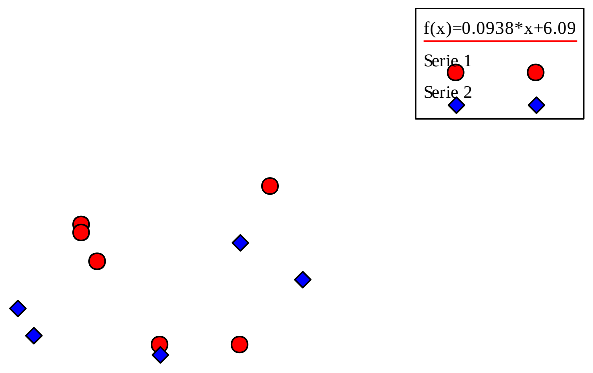
$$\frac{1}{1 + e^{-(-0{,}012 \cdot x_1 + 0{,}128 \cdot x_2 - 0{,}780)}} = 0{,}5$$

$\Updownarrow$ *Ligningen løses for x_2 vha. CAS-værktøjet WordMat.*

$$x_2 = 0{,}0938 \cdot x_1 + 6{,}09$$

The two neurons lead to the same separator function

$$f(x) = 0{,}0938 \cdot x + 6{,}09$$



Finally, we can observe that the data points are not linealry separable and that hence a training error equal to zero on this training set is not possible with the single layer neurons analysed.

Ex 9

Exercise 9. Single Layer Perceptrons

Can you represent the two layer perceptron of Figure 7 as a single perceptron that implements the same function? If yes, then draw the perceptron.

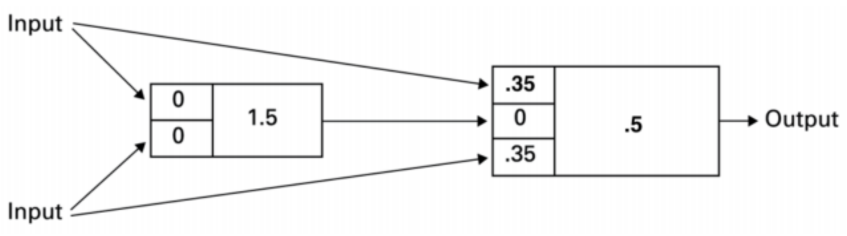


Figure 7: A two layer neural network

Da vægtene er 0 for node 1, vil vi altid få 0 som andet input i node 2. Derfor har den ingen contribution. Vi kan derfor udelade denne og få:

.35	.5
.35	

Vi kan også se at output fra O1 er ganget med 0, dvs. ingen indflydelse.

Ex 10

Exercise 10. Expressivness of Single Layer Perceptrons

Is there a Boolean (logical) function in two inputs that cannot be implemented by a single perceptron? Does the answer change for a single sigmoid neuron?

Solution

Yes, there is a Boolean (logical) function in two inputs that cannot be implemented by a single perceptron. We saw, for example, in the lecture notes that the algebraic expression of a perceptron is:

$$\text{output} := \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Then the decision boundary is

$$\sum_j w_j x_j = \text{threshold}$$

In the case of two inputs, x_1 and x_2, this becomes: $w_1 x_2 + w_2 x_2 = \text{threshold}$, which corresponds to the equation of a line in the Cartesian plane:

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{1}{w_2}\text{threshold}$$

(you might have seen this with y in place of x_2 and x in place of x_1.) Figure 9 taken from the slides gives an example of a non separable case:

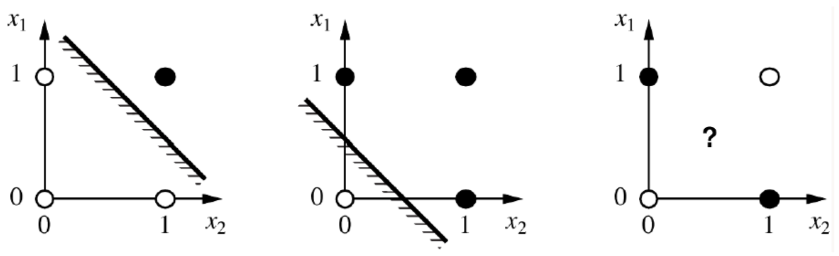


Figure 9: The figure is part of the solution only.

A sigmoid neuron would have the same problem. Indeed, if we use the value 0.5 as the discriminant on the output of a sigmoid neuron to answer 0 or 1 then the decision boundary corresponds to:

$$\frac{1}{1 + \exp(-\sum_j w_j x_j - b)} = 0.5$$

Solving in $\vec{x}$ we obtain an equation of the form:

$$\sum_j w_j x_j - b = \text{constant}$$

which therefore is also a line.

$$\frac{1}{1+e^{\sum_j w_j x_j - b}} = threshold$$

$$1 = threshold \cdot \left(1 + e^{\sum_j w_j x_j - b}\right)$$

$$1 = threshold + threshold \cdot e^{\sum_j w_j x_j - b}$$

$$1 - threshold = threshold \cdot e^{\sum_j w_j x_j - b}$$

$$\frac{1 - threshold}{threshold} = e^{\sum_j w_j x_j - b}$$

$$\log_e \frac{1 - threshold}{threshold} = \log_\mathrm{e} e^{\sum_j w_j x_j - b}$$

$$\log_e \frac{1 - threshold}{threshold} = \sum_j w_j x_j - b$$

$$\sum_j w_j x_j - b = constant$$

Ex 11

Exercise 11. Logical Functions and Neural Networks

The NAND gate is universal for computation, that is, we can build any computation up out of NAND gates. We saw in Exercise 5. that a single perceptron can model a NAND gate. From here, it follows that using networks of perceptrons we can compute any logical function.
For example, we can use NAND gates to build a circuit which adds two bits, x_1 and x_2. This requires computing the bitwise sum, x_1 XOR x_2, as well as a carry bit which is set to 1 when both x_1 and x_2 are 1, i.e., the carry bit is just the bitwise product x_1x_2. The circuit is depicted in Figure 8.

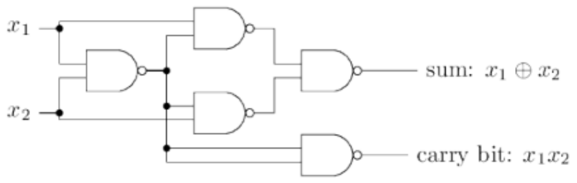
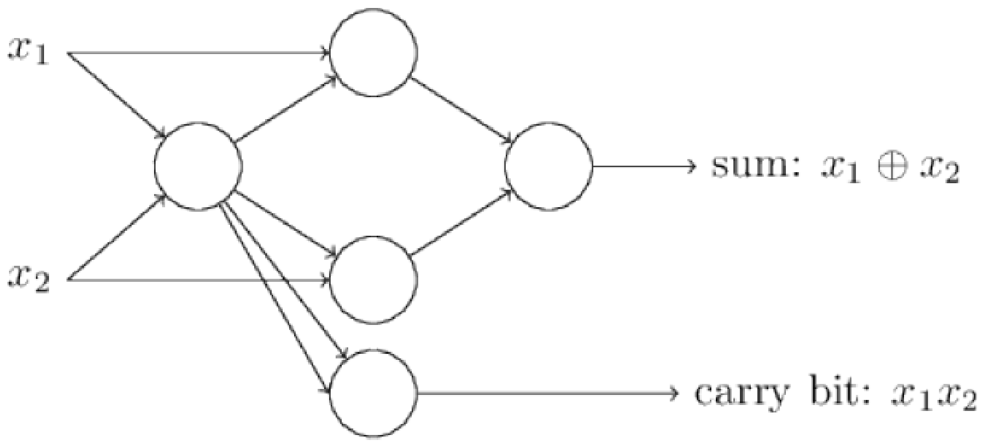


Figure 8: The adder circuit of Exercise 11.. All gates are NAND gates.

Draw a neural network of NAND perceptrons that would simulate the adder circuit from the figure. [You do not need to decide the weights. You have already discovered which weights for a single perceptron would implement a NAND function in Exercise 5.]

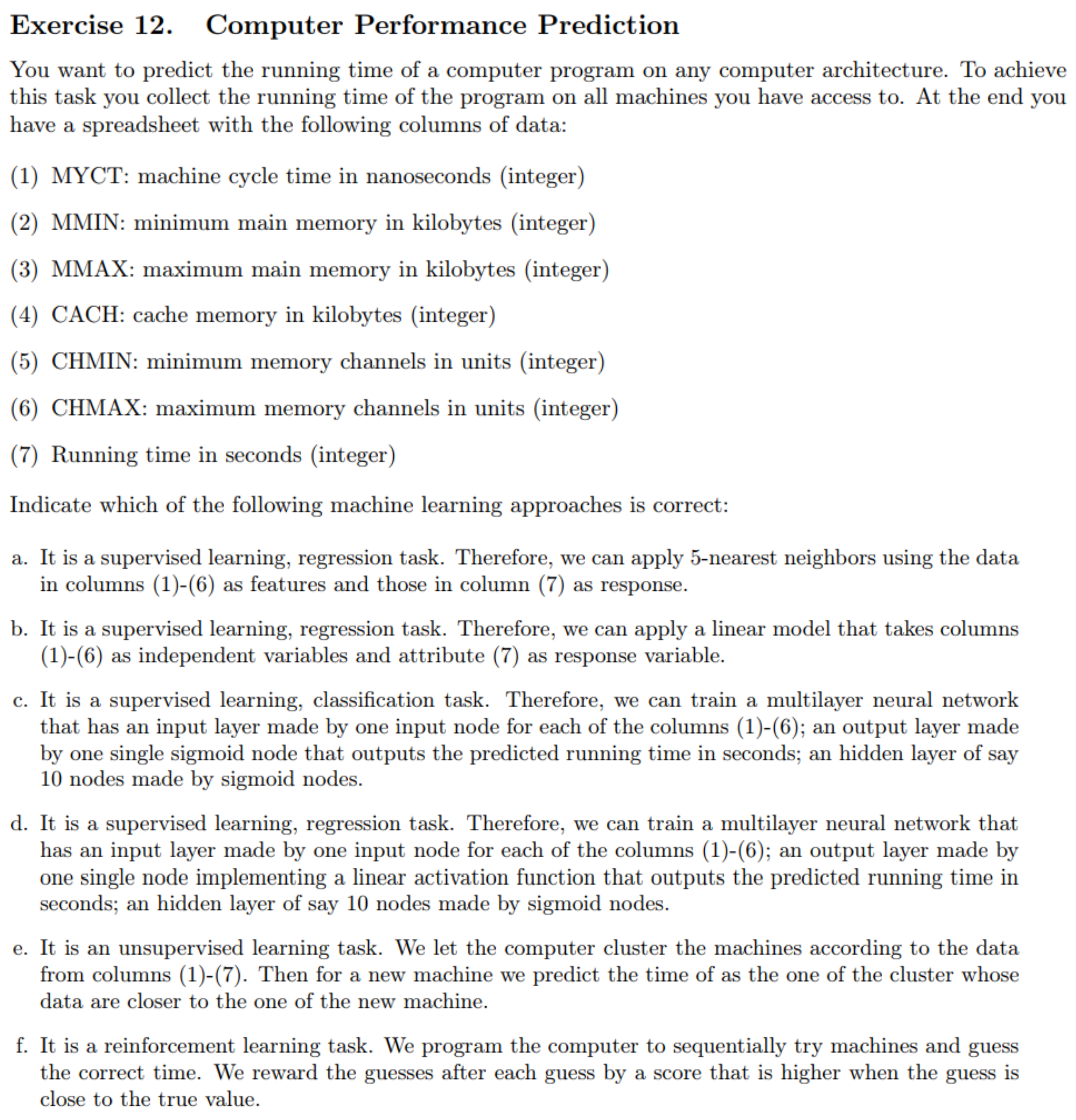
What is the advantage of neural networks over logical circuits when representing Boolean functions?

Dette er bare en direkte oversættelse, for hver NAND gate indsæt en perceptron.



It turns out that we can devise learning algorithms which can automatically tune the weights and biases of a network of artificial neurons. This tuning happens in response to external stimuli, without direct intervention by a programmer. These learning algorithms enable us to use artificial neurons in a way which is radically different to conventional logic gates. Instead of explicitly laying out a circuit of NAND and other gates, our neural networks can simply learn to solve problems, sometimes problems where it would be extremely difficult to directly design a conventional circuit

Ex 12



Exercise 12. Computer Performance Prediction

You want to predict the running time of a computer program on any computer architecture. To achieve this task you collect the running time of the program on all machines you have access to. At the end you have a spreadsheet with the following columns of data:

(1) MYCT: machine cycle time in nanoseconds (integer)

(2) MMIN: minimum main memory in kilobytes (integer)

(3) MMAX: maximum main memory in kilobytes (integer)

(4) CACH: cache memory in kilobytes (integer)

(5) CHMIN: minimum memory channels in units (integer)

(6) CHMAX: maximum memory channels in units (integer)

(7) Running time in seconds (integer)

Indicate which of the following machine learning approaches is correct:

a. It is a supervised learning, regression task. Therefore, we can apply 5-nearest neighbors using the data in columns (1)-(6) as features and those in column (7) as response.

b. It is a supervised learning, regression task. Therefore, we can apply a linear model that takes columns (1)-(6) as independent variables and attribute (7) as response variable.

c. It is a supervised learning, classification task. Therefore, we can train a multilayer neural network that has an input layer made by one input node for each of the columns (1)-(6); an output layer made by one single sigmoid node that outputs the predicted running time in seconds; an hidden layer of say 10 nodes made by sigmoid nodes.

d. It is a supervised learning, regression task. Therefore, we can train a multilayer neural network that has an input layer made by one input node for each of the columns (1)-(6); an output layer made by one single node implementing a linear activation function that outputs the predicted running time in seconds; an hidden layer of say 10 nodes made by sigmoid nodes.

e. It is an unsupervised learning task. We let the computer cluster the machines according to the data from columns (1)-(7). Then for a new machine we predict the time of as the one of the cluster whose data are closer to the one of the new machine.

f. It is a reinforcement learning task. We program the computer to sequentially try machines and guess the correct time. We reward the guesses after each guess by a score that is higher when the guess is close to the true value.

a. true
b. true
c. false, output always 0,1, because classification
d. true
e. false, we would not have (7) for the new maschine.
f. In reinforced learning we do not have supervision at every decision, but only at the end.