# UNIVERSITY OF OSLO

## Faculty of Mathematics and Natural Sciences

**IN3050/4050 — Introduction to Artificial Intelligence and Machine Learning**
**Trial Exam Spring 2020 – Suggested Solutions**
**Exam content: Around 2 working days**
**Deadline: June 2 at 2:30 PM**
**Permitted materials: All**

## 1) Simulated Annealing (3p)

a) What is the role of the temperature parameter in simulated annealing? (1p)
b) What would happen if we start with a very low temperature (keeping low through search)? Which search algorithm would this be similar to? (1p)
c) What would happen if we start with a very high temperature and never decrease? (1p)

## Suggested solution

a) The temperature *balances the degree of exploitation/exploration* in simulated annealing by tuning the *randomness of the algorithm in selecting the next solution*. High temperatures indicate a lot of randomness and exploration and low temperatures indicate less randomness and more frequently exploiting, selecting the best individual. By gradually decreasing the temperature, simulated annealing initially explores the search landscape, before focusing on the best solutions towards the end.
b) We would have a method doing almost only exploitation. This would be similar to hill climbing / local search, but with a bit more randomness, depending on how low we set the temperature.
*c)* We would have a lot of randomness and never settle on the areas in the fitness landscape with the best solution. This would be similar to random search/exhaustive search. *We would essentially move through random neighbors in the search landscape.*

## 2) Evolutionary Algorithm: Minimizing f(x) (10p)

In lecture 3, slide 32, we saw an example of an EA done by hand for a full generation (also found in Eiben & Smith, page 35). The simple problem was to optimize the value of the function f(x)=$x^2$ for integers in the range 0-31. Now, you are going to do the same, but for a minimization problem. That is, minimize the function f(x) = $x^2$ in the range 0-31, using the same genotype, phenotype, mutation and crossover as the example from the lecture.

a) Suggest a way to change the fitness function to make this a minimization problem. The original example used the following selection function:

$$p_i = f(i) / \sum_{j \in P} f(j)$$

Will you need to change this, or are the changes to the fitness function sufficient? Briefly explain why/why not? (2p)

Perform a full round of the EA on this minimization problem by filling in the tables below (don't fill in cells marked with a "-"). Note that the individuals in the population are different from those in the example from the lecture. Calculating the x-value for an individual is done with straightforward binary-to-decimal decoding. That is, each digit in the genotype is multiplied by $2^i$, where i indicates the position in the genotype from right to left, starting with 0: $[g_4, g_3, g_2, g_1, g_0]$. We have filled in two phenotype values to help you get started.

b) Parent selection:

When calculating the actual count, instead of randomly sampling, you can just *round the expected count to the nearest integer*. (If that gives you too few or too many individuals, try to select fairly according to the expected count). (2p)

| *String no.* | *Initial population* | *x value* | *Fitness f(x) you defined above* | *Prob$_i$* | *Expected Count* | *Actual count* |
|---|---|---|---|---|---|---|
| *1* | 0 1 1 0 1 | 13 | | | | |
| *2* | 0 0 1 1 1 | 7 | | | | |
| *3* | 1 1 1 0 0 | | | | | |
| *4* | 1 0 0 1 1 | | | | | |
| *Sum* | - | - | | | | |
| *Average* | - | - | | | | |
| *Best* | - | - | | | | |

c) Crossover:

Insert individuals into the mating pool in the same order they are listed above, according to the actual count. Use the indicated crossover points to cross over individual pairs (1,2) and (3,4). (1p)

| *String no.* | *Mating pool* | *Crossover point* | *Offspring after crossover* | *x Value* | *Fitness f(x)* |
|---|---|---|---|---|---|
| *1* | | 3 | | | |
| *2* | | 3 | | | |
| *3* | | 1 | | | |
| *4* | | 1 | | | |
| *Sum* | - | - | - | - | |
| *Average* | - | - | - | - | |
| *Best* | - | - | - | - | |

d) Mutation:

Now, mutate the offspring. Normally, we would do that randomly, but instead we will tell you how to mutate this time. Perform the normal binary mutation, on the following genes:

-Gene 1 in individual 1
-Gene 4 in individual 2
-Gene 2 in individual 4

(1p)

| *String no.* | *Offspring after crossover* | *Offspring after mutation* | *x Value* | *Fitness f(x)* |
|---|---|---|---|---|
| *1* | | | | |
| *2* | | | | |
| *3* | | | | |
| *4* | | | | |
| *Sum* | - | - | - | |
| *Average* | - | - | - | |
| *Best* | - | - | - | |

e) What is the best resulting x value and associated fitness value? Has the EA resulted in an increase to the average and best fitness? Why/why not? (2p)

f) Assume we want to hybridize our EA with a local search. The local search searches through all solutions that are 1 bit-flip away from our current solutions (in other words, the search tests 5 “neighbours” of the current individual), and selects the best one found. We insert the local search right after mutation, as the final part of the evolutionary loop before the next generation begins. Assume we have the following individual after mutation: [1 0 0 1 0]. With your fitness function above, what is the fitness of this individual?
What is the genotype and fitness after the local search, assuming Lamarckian Evolution?
What is the genotype and fitness after the local search assuming Baldwinian Evolution? (2p)

## Suggested Solution

a) Several fitness functions may work, the simplest is probably $f(x) = (31-x)^2$. In this case, no need to change the selection function, because we still want the fitness to be proportionate to the distance from the optimal number (now 0).

*Importantly, the chosen fitness function needs to either 1) Give higher values to lower numbers, and be paired with the normal selection function above, or 2) Give lower values to lower numbers, and be paired with a selection operator that gives higher probability of choosing individuals with low fitness values. You should also be careful with fitness functions that give sub-zero fitness values as that may give some unexpected results when doing fitness-proportionate selection.*

b)

| *String no.* | *Initial population* | *x value* | *Fitness f(x) = $(31-x)^2$* | *$Prob_i$* | *Expected Count* | *Actual count* |
|---|---|---|---|---|---|---|
| *1* | 0 1 1 0 1 | 13 | 324 | 0.31 | 1.23 | 1 |
| *2* | 0 0 1 1 1 | 7 | 576 | 0.55 | 2.19 | 2 |
| *3* | 1 1 1 0 0 | 28 | 9 | 0.0 | 0 | 0 |
| *4* | 1 0 0 1 1 | 19 | 144 | 0.14 | 0.55 | 1 |
| *Sum* | (not applicable) | (not applicable) | 1053 | 1.0 | 4 | 4 |
| *Average* | (not applicable) | (not applicable) | 263.25 | 0.25 | 1 | 1 |
| *Best* | (not applicable) | (not applicable) | 576 | 0.51 | 2.02 | 2 |

0.5p for correctly using the fitness function you defined. 1p for selection probabilities that are proportionate to the fitness values, and that sum to 1. 1p if the number of individuals in the next generation corresponds to the selection probability and the correct new individuals are inserted below (0.5p).

c)

| *String no.* | *Mating pool* | *Crossover point* | *Offspring after crossover* | *x Value* | *Fitness f(x)* |
|---|---|---|---|---|---|
| *1* | 0 1 1 0 1 | 3 | 0 1 1 1 1 | 15 | 256 |
| *2* | 0 0 1 1 1 | 3 | 0 0 1 0 1 | 5 | 676 |
| *3* | 0 0 1 1 1 | 1 | 0 0 0 1 1 | 3 | 784 |
| *4* | 1 0 0 1 1 | 1 | 1 0 1 1 1 | 23 | 64 |
| *Sum* | (not applicable) | (not applicable) | | | 1780 |
| *Average* | (not applicable) | (not applicable) | | | 445 |
| *Best* | (not applicable) | (not applicable) | | | 784 |

0.5 p for doing crossover correctly and 0.5p for calculating new fitness values correctly

d)

| *String no.* | *Offspring after crossover* | *Offspring after mutation* | *x Value* | *Fitness f(x)* |
|---|---|---|---|---|
| *1* | 0 1 1 1 1 | 1 1 1 1 1 | 31 | 0 |
| *2* | 0 0 1 0 1 | 0 0 1 1 1 | 7 | 576 |
| *3* | 0 0 0 1 1 | 0 0 0 1 1 | 3 | 784 |
| *4* | 1 0 1 1 1 | 1 1 1 1 1 | 31 | 0 |
| *Sum* | - | - | - | 1360 |
| *Average* | - | - | - | 340 |
| *Best* | - | - | - | 784 |

0.5 points for mutating correctly and 0.5p for calculating new fitness values correctly

e) 3, with fitness 784. The selection helped push fitness up, while the random mutation unfortunately reduced it a bit here. (2p)

f) The fitness of [1 0 0 1 0] depends on the f(x) selected above, but assuming the one we suggested, the value is $(31 - 18)^2 = 169$.
The genotype after local search is [0 0 0 1 0] (the best solution 1 bit-flip away). Its fitness is $(31 - 2)^2 = 841$ assuming Lamarckian evolution.
In Baldwinian evolution, the fitness value resulting from local search is kept, but not the genotype, so the genotype would be the original [1 0 0 1 0] but the fitness would be the optimized value of 841.

## 3) Evolutionary Algorithms – Variation Operators (7p)

Assume the following two genotypes in a permutation-style EA problem:

g1 - [1 3 5 4 2 6 7]
g2 - [7 4 3 5 2 1 6]

Show the steps in your calculations below, and not just the final answer.

a) Why is it important to use specialized variation operators when genotypes are permutations? (1p)

b) What is the result of performing:
   a. Inversion Mutation on each genotype, from the second to the fourth gene (0.5p)
   b. Swap mutation on each genotype, with the second and fifth gene as swap points (0.5p)

c) Partially Mapped Crossover between g1 and g2, assuming the initial segment is taken from the first parent, from gene position 4 to 6 (the segment [4 2 6]). You should only create the first child. (2p)

d) Edge Crossover between g1 and g2. Whenever you have to make a random choice, choose instead the lowest number. Start with 1 as the initial element. Show your calculated edge table. (3p)

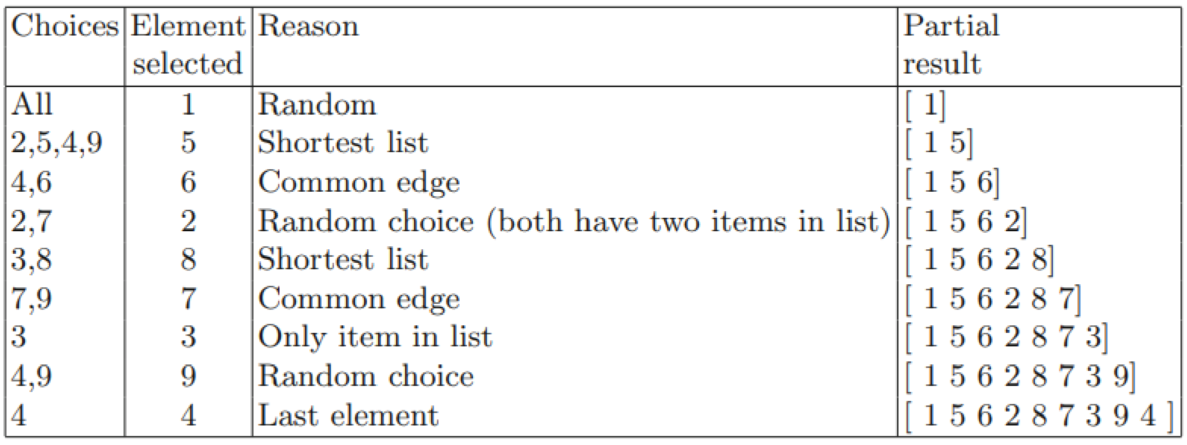
Edge table:

| Element | Edges |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |

Show how you construct the new individual by setting up a table similar to the one in the book. We have filled in the first element for you:

| Choices | Element Selected | Reason | Partial Result |
|---|---|---|---|
| All | 1 | Random choice | [1] |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

The example table from the book is given below for convenience – but note that your edges and calculations will be different.



| Choices | Element selected | Reason | Partial result |
|---|---|---|---|
| All | 1 | Random | [ 1] |
| 2,5,4,9 | 5 | Shortest list | [ 1 5] |
| 4,6 | 6 | Common edge | [ 1 5 6] |
| 2,7 | 2 | Random choice (both have two items in list) | [ 1 5 6 2] |
| 3,8 | 8 | Shortest list | [ 1 5 6 2 8] |
| 7,9 | 7 | Common edge | [ 1 5 6 2 8 7] |
| 3 | 3 | Only item in list | [ 1 5 6 2 8 7 3] |
| 4,9 | 9 | Random choice | [ 1 5 6 2 8 7 3 9] |
| 4 | 4 | Last element | [ 1 5 6 2 8 7 3 9 4 ] |

**Table 4.3.** Edge crossover: example of permutation construction

## Suggested Solution

a) Because simple crossover/mutation (e.g. the methods for integral genotypes) will destroy the permutation by inserting the same element several times and losing elements.

b) a) [1 4 5 3 2 6 7] and [7 5 3 4 2 1 6] b) [1 2 5 4 3 6 7] and [7 2 3 5 4 1 6]

c) Method from Eiben & Smith:

1. Choose two crossover points at random, and copy the segment between them from the first parent (P1) into the first offspring.
2. Starting from the first crossover point look for elements in that segment of the second parent (P2) that have not been copied.
3. For each of these (say $i$), look in the offspring to see what element (say $j$) has been copied in its place from P1.
4. Place $i$ into the position occupied by $j$ in P2, since we know that we will not be putting $j$ there (as we already have it in our string).
5. If the place occupied by $j$ in P2 has already been filled in the offspring by an element $k$, put $i$ in the position occupied by $k$ in P2.
6. Having dealt with the elements from the crossover segment, the remaining positions in this offspring can be filled from P2, and the second child is created analogously with the parental roles reversed.

Minimal detail answer for full score:
[- - - 4 2 6 -]
Inserting the element from g2 from the spot corresponding to the first crossover point. That is 5. We insert it into the spot occupied by 4 in g2 since we know that spot will not be used for other things (we already have a 4).
[- 5 – 4 2 6 -]
With the same reasoning, we insert the 1.
[- 5 – 4 2 6 1]
With all items corresponding to the initially copied substring inserted, we insert the rest of the elements from the appropriate positions in g2.
[7 5 3 4 2 6 1]

d) Method from Eiben & Smith:

In order to achieve this, an edge table (also known as an adjacency list) is constructed, which for each element lists the other elements that are linked to it in the two parents. A '+' in the table indicates that the edge is present in both parents. The operator works as follows:

1. Construct the edge table
2. Pick an initial element at random and put it in the offspring
3. Set the variable $current\_element = entry$
4. Remove all references to $current\_element$ from the table
5. Examine list for $current\_element$
   - If there is a common edge, pick that to be the next element
   - Otherwise pick the entry in the list which itself has the shortest list
   - Ties are split at random
6. In the case of reaching an empty list, the other end of the offspring is examined for extension; otherwise a new element is chosen at random

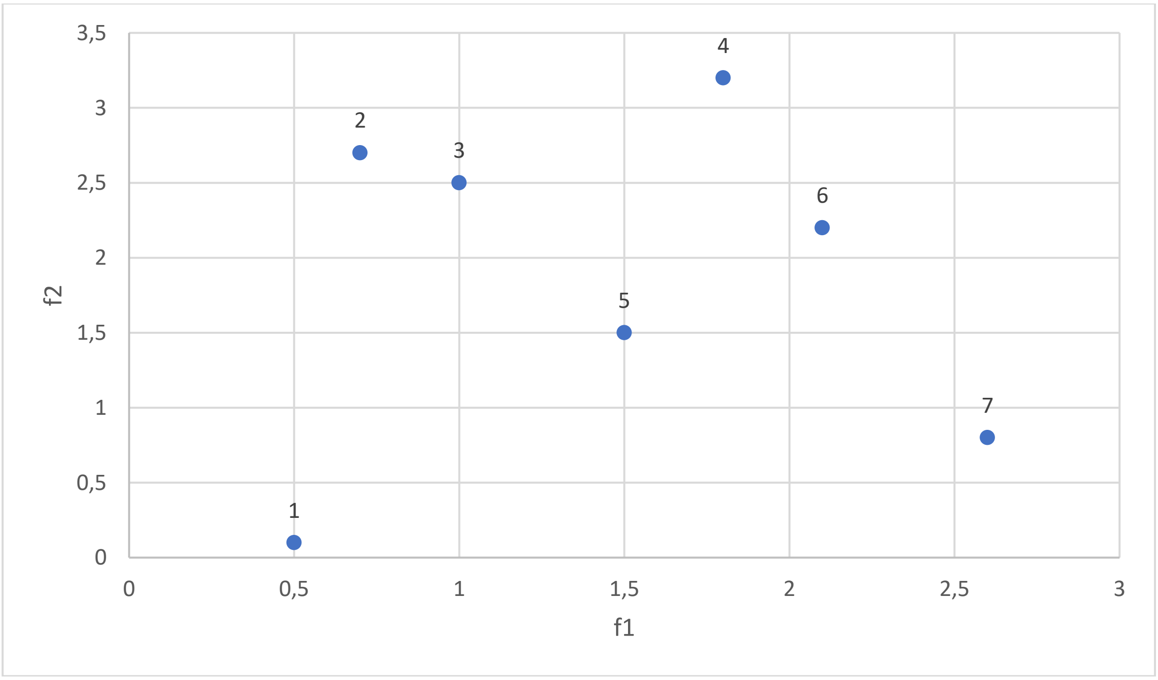
Edge table:

| Element | Edges |
|---|---|
| 1 | 3,7,2,6 |
| 2 | 4,6,5,1 |
| 3 | 1,5+,4 |
| 4 | 5,2,7,3 |
| 5 | 3+,4,2 |
| 6 | 2,1,7+ |
| 7 | 6+,1,4 |

| Choices | Element Selected | Reason | Partial Result |
|---|---|---|---|
| All | 1 | Random | [1] |
| 3,7,2,6 | 3 | Random among those with the shortest list (3,6,7) | [1,3] |
| 5,4 | 5 | Common edge | [1,3,5] |
| 4,2 | 2 | Random choice (both have 2 available edges in the list) | [1,3,5,2] |
| 4,6 | 4 | Shortest list | [1,3,5,2,4] |
| 7 | 7 | Only element left | [1,3,5,2,4, 7] |
| 6 | 6 | Only element left | [1,3,5,2,4,7, 6] |

## 4) Pareto Optimality (3p)

For an optimization problem we wish to optimize solutions according to two different objectives, f1 and f2. The fitness values according to the two objectives for 7 different solutions are plotted in the figure below.



a) What requirements do the solutions in a Pareto optimal set need to fulfill? (1p)

Find the Pareto optimal set of solutions when

b) Maximizing f1 and f2 (1p)

c) Maximizing f1 but minimizing f2 (1p)

## Suggested Solution

- Solutions on the Pareto optimal set have to be non-dominated. That is, no other solutions should be better according to both objectives (or even better on some but equally good on others).
- 4, 6 and 7
- 1, 7

## 5) Classification (10 p)

Given the following data

| Item | X1 | X2 | Class |
|---|---|---|---|
| A | -0.3 | 0.3 | no |
| B | 0.3 | 0.6 | yes |
| C | 0.6 | 0.6 | yes |
| D | 0.6 | 0.3 | no |

a) The goal is to train a perceptron classifier on these data. Set the bias to -1 and the learning rate to 0.1. Also assume that each of the weights is initially set to 0.1. Run the perceptron algorithm sequentially through the data in the order given from A to D. Show how the points get classified and update the weights. (3p)

b) Repeat two more rounds. Has the algorithm converged? (2p)

c) Suppose you had instead used a batch strategy for training. Run one round A-D with the same initial weights and show how the weights are updated. (2p)

d) Assume that we use a linear regression classifier instead. Run one round batch update with the linear regression classifier and show how the weights are updated. (3p)

## Suggested Solution

We have received several questions to this solution on Piazza. This is probably because the original posted solution mainly contained the calculations with little explanations. There seems to be some confusion. We will therefore repeat some of the theory to make sure the solution is well understood.

### Learning algorithm

You have to be aware of which learning algorithm you are using and its properties, whether it is the perceptron, linear regression, logistic regression or something else.

The goal is a classification task where the target, $t$, is 0 or 1.
All the three algorithms calculate the sum $z = \boldsymbol{w} \cdot \boldsymbol{x}$, but they deviate with respect to how they make the prediction $y$ from this

a. The perceptron uses the step function and sets $y = 1$ if $z > 0$, and $y = 0$, otherwise
b. The linear regression classifier sets $y = z$, hence $y$ can be any real number
c. The logistic regression also produces a real number, but it squeezes it to be between 0 or 1 by applying the logistic (sigmoid) function, $y = \textbf{logistic}(z)$

## Sequential vs. batch processing

With sequential processing, you consider one data point at a time. You make the prediction using the current weights, compare to the target value, see whether the weights have to be updated and update the weights. Then you do the same to the next data point.

With batch processing, you consider all the data points using the same weights. For each data point, you calculate the prediction, compare to the target value, see whether the weights have to be updated and calculate which correction this data point should make to the weights called 'Delta_w_i' in the solution below). Then for each weight, you sum all the corrections across all the data points and use this to update the weight. Marsland does this is in the code example in ch. 3, where everything happens in one line thanks to Numpy (page 52).

## Sum of squares or Mean sum of squares error (MSE)

The loss function for linear regression can either be written as

- Sum of squares $\sum_{i=1}^{N}(y_i - t_i)^2$, or
- Mean sum of squares $\frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2$

where N is the number of data points. It does not matter for the learning algorithm whether you use one or the other. You might choose different learning rates, however, depending on which of the two you are using. Marsland uses sums of squares in chapter 3 (page 65). We used the MSE in lecture 6 (see slide 45, also lecture 8, slide 12). One advantage of using MSE is that we can more easily compare the goodness of fit across various data sets, and it has an immediate interpretation; the square of the mean distance between predicted and target values.

Using this we get the following in point (d)

$$w_k = w_k - \eta\frac{\partial}{\partial w_k}f = w_k - \eta\frac{2}{N}\sum_{j=1}^{N}\left(t_j - y_j\right)\left(-x_{jk}\right) = w_k + \frac{2}{N}\sum_{i=1}^{N}(-\eta(y_j - t_j)x_{jk})$$

(Footnote: You might also see the loss function in some books

- $\frac{1}{2}\sum_{i=1}^{N}(y_i - t_i)^2$

because it yields a slightly simpler derivative, cf. Marsland page 102.)

## a) Perceptron, sequential

For each data point, make a prediction, compare to target and update the weights.

Initialize weights: [0.1, 0.1, 0.1]
Input: [-1, -0.3, 0.3], target: t=0, predicted: y=0, error: y-t =0
No change of weights

Input: [-1, 0.3, 0.6], target: t=1, predicted: y=0, error: y-t =-1
Weight w_0 = w_0 - eta * error * x_0 = 0.10 - 0.1 * -1 * -1 = 0.00
Weight w_1 = w_1 - eta * error * x_1 = 0.10 - 0.1 * -1 * 0.3 = 0.13
Weight w_2 = w_2 - eta * error * x_2 = 0.10 - 0.1 * -1 * 0.6 = 0.16

Input: [-1, 0.6, 0.6], target: t=1, predicted: y=1, error: y-t =0
No change of weights

Input: [-1, 0.6, 0.3], target: t=0, predicted: y=1, error: y-t =1
Weight w_0 = w_0 - eta * error * x_0 = 0.00 - 0.1 * 1 * -1 = 0.10
Weight w_1 = w_1 - eta * error * x_1 = 0.13 - 0.1 * 1 * 0.6 = 0.07
Weight w_2 = w_2 - eta * error * x_2 = 0.16 - 0.1 * 1 * 0.3 = 0.13

**Updated weights:[0.1, 0.07, 0.13] after 1 many rounds**

### b) Two more rounds

Input: [-1, -0.3, 0.3], target: t=0, predicted: y=0, error: y-t =0
No change of weights

Input: [-1, 0.3, 0.6], target: t=1, predicted: y=0, error: y-t =-1
Weight w_0 = w_0 - eta * error * x_0 = 0.10 - 0.1 * -1 * -1 = 0.00
Weight w_1 = w_1 - eta * error * x_1 = 0.07 - 0.1 * -1 * 0.3 = 0.10
Weight w_2 = w_2 - eta * error * x_2 = 0.13 - 0.1 * -1 * 0.6 = 0.19

Input: [-1, 0.6, 0.6], target: t=1, predicted: y=1, error: y-t =0
No change of weights

Input: [-1, 0.6, 0.3], target: t=0, predicted: y=1, error: y-t =1
Weight w_0 = w_0 - eta * error * x_0 = 0.00 - 0.1 * 1 * -1 = 0.10
Weight w_1 = w_1 - eta * error * x_1 = 0.10 - 0.1 * 1 * 0.6 = 0.04
Weight w_2 = w_2 - eta * error * x_2 = 0.19 - 0.1 * 1 * 0.3 = 0.16

**Updated weights:[0.1, 0.04, 0.16] after 2 many rounds**

Input: [-1, -0.3, 0.3], target: t=0, predicted: y=0, error: y-t =0
Input: [-1, 0.3, 0.6], target: t=1, predicted: y=1, error: y-t =0
Input: [-1, 0.6, 0.6], target: t=1, predicted: y=1, error: y-t =0
Input: [-1, 0.6, 0.3], target: t=0, predicted: y=0, error: y-t =0
No change of weights

**Converged after 2 rounds**

### c) Perceptron batch

1. For each data point, make a prediction, compare to target, calculate the delta term, but do not update the weights.
Initialize weights: [0.1, 0.1, 0.1]
Input: [-1, -0.3, 0.3], target: t=0, predicted: y=0, error: y-t =0
No addition to delta-wi-s

Input: [-1, 0.3, 0.6], target: t=1, predicted: y=0, error: y-t =-1
Delta_w_0 = - eta * error * x_0 = - 0.1 * -1 * -1 = -0.10
Delta_w_1 = - eta * error * x_1 = - 0.1 * -1 * 0.3 = 0.03
Delta_w_2 = - eta * error * x_2 = - 0.1 * -1 * 0.6 = 0.06

Input: [-1, 0.6, 0.6], target: t=1, predicted: y=1, error: y-t =0
No addition to delta-wi-s

Input: [-1, 0.6, 0.3], target: t=0, predicted: y=0, error: y-t =0
No addition to delta-wi-s

2. Sum the deltas for the data points. Here the deltas =/= 0 for only one data point.
3. Update the weights with the sum of the deltas, e.g.
w_0 = w_0 +  deltas_w_0 = 0.1 - 0.1  = 0.0

**Updated weights:[0.0, 0.13, 0.16] after 1 many rounds**

### d) Linear regression batch

Initialize weights: [0.1, 0.1, 0.1]
Input: [-1, -0.3, 0.3], target: t=0, predicted: y=-0.1000, error: y-t =-0.1000
Delta_w_0 = - eta * error * x_0 = - 0.1 * -0.1000 * -1.0000 = -0.0100
Delta_w_1 = - eta * error * x_1 = - 0.1 * -0.1000 * -0.3000 = -0.0030
Delta_w_2 = - eta * error * x_2 = - 0.1 * -0.1000 * 0.3000 = 0.0030

Input: [-1, 0.3, 0.6], target: t=1, predicted: y=-0.0100, error: y-t =-1.0100
Delta_w_0 = - eta * error * x_0 = - 0.1 * -1.0100 * -1.0000 = -0.1010
Delta_w_1 = - eta * error * x_1 = - 0.1 * -1.0100 * 0.3000 = 0.0303
Delta_w_2 = - eta * error * x_2 = - 0.1 * -1.0100 * 0.6000 = 0.0606

Input: [-1, 0.6, 0.6], target: t=1, predicted: y=0.0200, error: y-t =-0.9800
Delta_w_0 = - eta * error * x_0 = - 0.1 * -0.9800 * -1.0000 = -0.0980
Delta_w_1 = - eta * error * x_1 = - 0.1 * -0.9800 * 0.6000 = 0.0588
Delta_w_2 = - eta * error * x_2 = - 0.1 * -0.9800 * 0.6000 = 0.0588

Input: [-1, 0.6, 0.3], target: t=0, predicted: y=-0.0100, error: y-t =-0.0100
Delta_w_0 = - eta * error * x_0 = - 0.1 * -0.0100 * -1.0000 = -0.0010
Delta_w_1 = - eta * error * x_1 = - 0.1 * -0.0100 * 0.6000 = 0.0006
Delta_w_2 = - eta * error * x_2 = - 0.1 * -0.0100 * 0.3000 = 0.0003

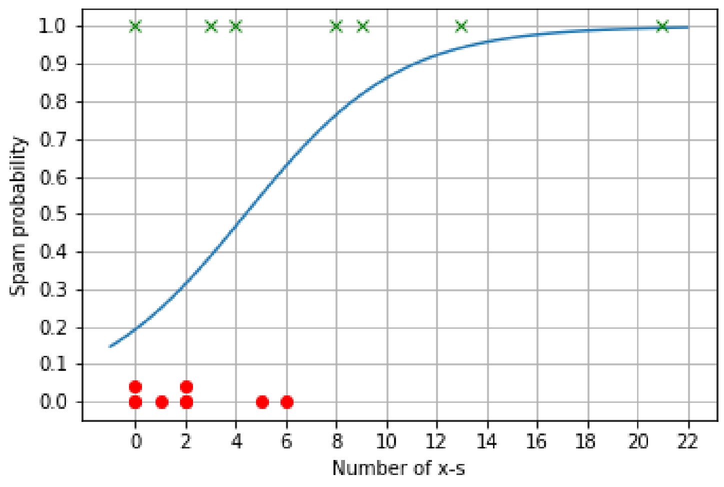
**Updated weights:[-0.005, 0.14335, 0.16135] after 1 many rounds using MSE**

## 6) Logistic Regression (6 p)

Kim is building a spam filter. She has the hypothesis that counting the occurrences of the letter 'x' in the e-mails will be a good indicator of spam or no-spam. She collects 7 spam messages and 7 no-spam messages and counts the number of x-s in each. Here is what she finds.

- Number of 'x'-s in each spam: [0, 3, 4, 8, 9, 13, 21]
- Number of 'x'-s in each no-spam: [0, 0, 1, 2, 2, 5, 6]

She trains a logistic regression classifier on the data and plots the classifier against the data.



Assume the logistic regression model and answer the following questions:

a) Consider an e-mail with no 'x'-s. According to the model, what is roughly the probability of this message being a spam message and what is the probability of it not being a spam. (1p)

b) How many x-s must an e-mail contain to guarantee it is a spam mail? (1p)

c) How is a logistic regression model normally turned into a binary classifier? If you turn the model into a classifier in this way, what is the accuracy of the classifier on the training data? (2p)

d) It is most important that no no-spams are classified as spams. How can this goal be described in terms of precision and recall? How can the logistic regression classifier be modified to try to achieve this goal? (2p)

## Suggested Solution

a) We see from the graph that P(1 | x=0) = 0.2 and P(0 | x=0) = 0.8.

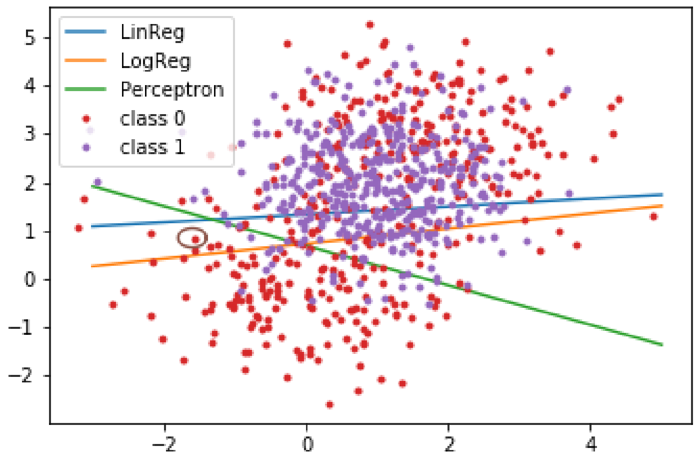
b) You can never be 100% sure with a logistic regression model,

c) This is normally done by choosing the class 1 if P(1 | x) > 0.5.
We see from the graph that this classify 4 spams correctly and 3 spams incorrectly and 5 no-spams correctly and 2 incorrectly. Altogether 9 out of 14 are classified correctly, yielding and accuracy of 9/14.

d) We could either say that the goal is to get a good precision for spam, or a good recall for no-spam. We achieve this by raising the threshold from 0.5. For the training data, a threshold of 0.7 would suffice. If we want to be prepared for more variation in the test data, we could set the threshold even higher.

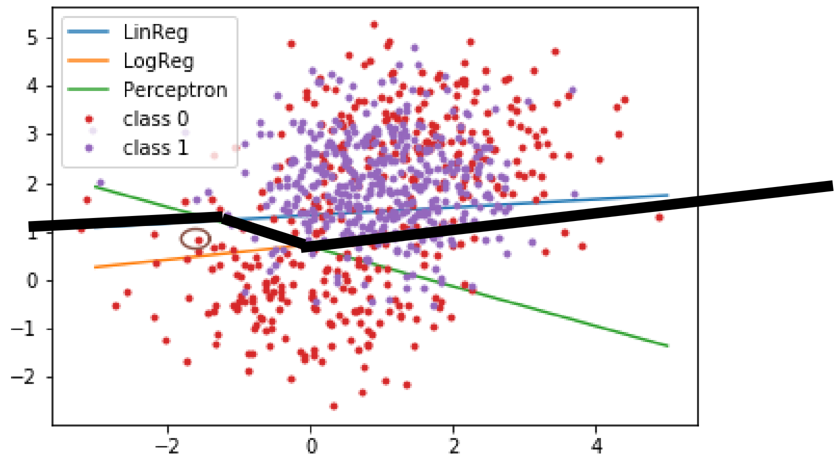
## 7) Majority Voting Classifier (4 p)

We have trained three different classifiers on the same training data (from mandatory assignment 2); a linear regression classifier, a logistic regression classifier, and a perceptron classifier. We have plotted the decision boundaries for all three classifiers on the training data in the figure. They all classify the points above their boundaries as class 1 (purple) and the points below the boundary as class 0 (red). By referring to the figure and the circled point, explain how a majority voting classifier works. In particular, describe the decision boundary for the majority vote classifier.



## Suggested Solution

A voting classifier consists of several different classifiers trained on the same problem and (possible subsets of) the same data. To make a prediction, the voting classifier applies all the classifiers on the datapoint and let them each predict a class; collect the predictions and choose the majority class. In the figure, the circled red point will be classified as red, since it is classified as red by two of the three classifiers (the perceptron and the linear regression) even though it is classified as purple by the logistic regression classifier. In this case, the decision boundary will look as the tick, black lines.



## 8) Backpropagation (10p)

(From INF3490/4490 — Biologically Inspired Computing, exam 2014)
SiO, the student welfare organization, would like to have a system for sorting utensils after washing. You are going to help them design a camera-based classifier system for sorting knives, forks, spoons and teaspoons into separate bins. You have a machine vision library available that lets you identify where there is a utensil in the camera images, and it extracts a large number of features for each identified object that we can use as inputs.

a) What class of learning algorithm would be best to use in this case: supervised, unsupervised or reinforcement learning? Justify your answer. (1p)

b) We would like to make a system for distinguishing the utensils using a multi-layer perceptron network. How many output neurons should the network have, and what would each of them represent? (1p)

c) Sketch the steps in the forward and backward phase of the multi-layer perceptron algorithm (backpropagation). Use words and not equations. (3p)

d) An error term is used for updating the weights of the output layer:
Explain the different parts (including what they represent) of the equations (you don´t need to use indices).

$$\delta_o(\kappa) = (y_\kappa - t_\kappa)\, y_\kappa(1 - y_\kappa)$$

How is the term above used for updating the weights in the hidden layer? (2p)

e) What are the different approaches to how often weights are updated during training? (1p)

f) How would you find out when to stop the training? (2p)

## Suggested Solution

a) Supervised learning

b) 4. One for each class

c) We assume that there is one hidden layer. To simplify the description let

- m be the number of features in an input observation (the number of input nodes)
- n the number of output nodes
- k the number of nodes in the hidden layer
- v_ij for 1<= i <= m, 1<= j <= k, the weight on the connection from input node i to hidden node j
- v_0j for 1<= j <= k, the weight on the connection from input bias node to hidden node j
- w_ij for 1<= i <= k, 1<= j <= n, the weight on the connection from hidden node i to output node j
- w_0j for 1<= j <= k, the weight on the connection from hidden bias node to output node j

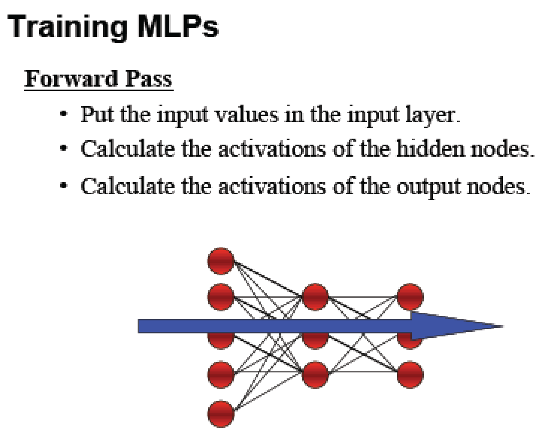
**Forward**

1. An input vector is put into the input nodes 1 to m and an extra feature (-1) is put into node 0
2. At each hidden j node
    - the value at each input node, including the bias node 0, is multiplied with the weight v_ij corresponding to node j, and these products are summed
    - the resulting sum is sent though an activation function, e.g. the logistic function
3. At each output node j
    - the output value at each hidden node together with the value of the hidden bias node (-1), is multiplied with the weight W_ij corresponding to node j, and these products are summed
    - the resulting sum is sent though an activation function, e.g. the logistic function (it could also be the identity function, i.e. no additional activation)

**Backward**

1. A loss function (error function, cost function), e.g. the sum of squares, is used to calculate the difference between the output and the target value(s).
2. Applying gradient descent, one calculates the error term at the output nodes.
3. The error term is back-propagated to the hidden layer to construct the error term at the hidden layer.
4. The second layer weight and the first layer weights are updated by multiplying the error term with the learning rate and the incoming value to the weight from the forward phase.

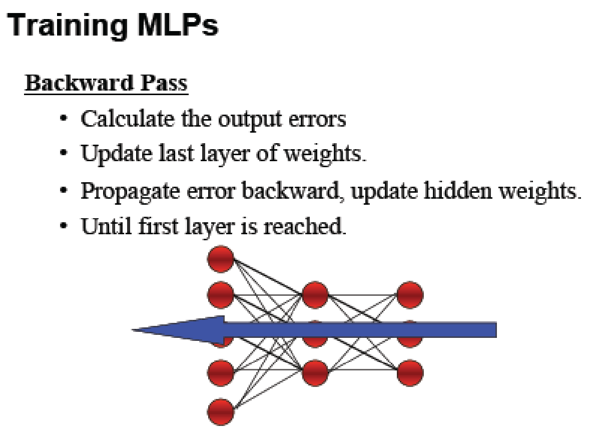
This process is repeated for each training vector. This is the sequential version of the backpropagation algorithm (other variants called batch and epoch training accumulate the error for a number of training vectors before the weights are updated).



## Training MLPs

**Forward Pass**

- Put the input values in the input layer.
- Calculate the activations of the hidden nodes.
- Calculate the activations of the output nodes.



## Training MLPs

**Backward Pass**

- Calculate the output errors
- Update last layer of weights.
- Propagate error backward, update hidden weights.
- Until first layer is reached.

d) y-t is the output error and y(1-y) is the derivate of the logistic activation function

The delta term is multiplied with the corresponding weights in the output layer. This sum-of-products is then multiplied with the derivate of the hidden layer activation function to compute delta terms for the hidden layer nodes. The computed error values impact how much each weight in the hidden layer is updated.

$$\delta_h(\zeta) = a_\zeta(1 - a_\zeta) \sum_{k=1}^{N} w_\zeta \delta_o(k)$$

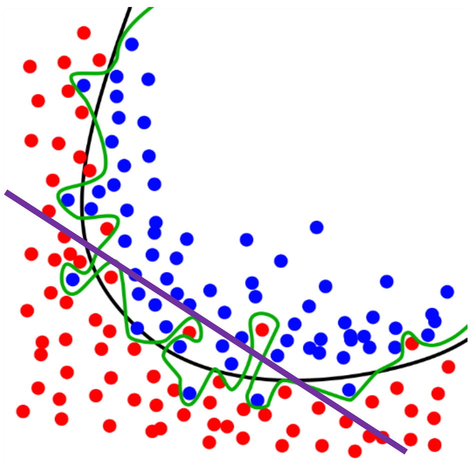
e)
**Batch:**
Run the forward step and calculate the delta terms for each datapoint without updating the weights.
At each node sum the corresponding delta terms over all datapoints.
Update each weight using the sum of delta terms.

**Mini-batch**
Split the training data into smaller sets.
Perform the same algorithm for each set.

**Stochastic gradient descent**
Pick one datapoint at random.
Calculate the delta terms and update the weights from this datapoint.
Repeat.

f) Make a validation set separate from the training set. Decide on a fixed number, *n*, e.g. 10, 50, 100 or something else. After each *n* numbers of epochs, calculate the loss on the validation set. When the loss stops decreasing and starts to increase, stop the training.

## 9) Overfitting and Bias (10 points)



a) What do we mean by 'overfitting' in machine learning, and why can it become a problem? You may refer to the figure in the discussion or make examples. (2p)

b) The term 'bias' is used in various ways in machine learning. We are here considering the use as in "the bias-variance trade-off" and "inductive bias". What do we mean by 'bias' in these contexts? Again, you may refer to the figure in the discussion or make examples. (2p)

c) You are facing a regression problem. You first try a linear regression model. What are the possible problems you may encounter with 'overfitting' and 'bias'? (1p)

d) You want to address the bias problem. You still want to use linear regression as learning algorithm. Which other settings can you change to get a model which faces less problems with the bias, and how will you change them? Which new problems may these improvements cause? (2p)

e) You are training a logistic regression classifier. Which problems can arise with respect to overfitting and bias? (1p)

f) We will concentrate on overfitting. Which methods are there for avoiding overfitting when using gradient descent for logistic regression? Explain the main parts of each method. (2p)

### Suggested Solution

a) (from lecture 9 slide 12) "Whenever we choose a random sample of individuals from a larger population, there might be attributes that have a different distribution in the sample than in the population at large." If the learning algorithm focus too much on these data, it may not fit other samples from the population equally well. The wavy line in the figure could be an example of this.

b) Bias: By choosing a learner, we can only train models that are within the class of the learner, e.g. a linear regressor can only produce a linear classifier. The decision border of any linear classifier will be a straight line, like the purple line in the figure. We see that this is not a good classifier for this problem.

c) A linear regression model can, as indicated by the name, only result in a linear regression model corresponding to a straight line. This is a strong bias. Many problems are, however, non-linear.

A linear regression model can also experience problems with overfitting, in particular if there are many features.

d) You can engineer your features, e.g. adding polynomial forms of them. The danger is that if you are too clever in the engineering, you may overfit the data. For example, if you have more features than data, you may train a perfect model on the training data, which does not generalize very well.

e) The logistic regression classifier can also only predict linear decision boundaries and run into similar problems as the linear regressor with respect to bias and overfitting.

f) To avoid overfitting, one possibility is to apply early stopping. Another possibility is to use regularization. L2-regularization punishes large weights, Thereby it avoids putting too much emphasis on some features which may have a distribution in the training data very different from its distribution in the population at large.

## 10) Decision Tree Classifier (10 points)

(The following example is of course simplified.) Kim is training an entailment classifier on 25 training items. Each item consists of a premise, P, and a hypothesis, H. The test items belong to one of two classes: Entailment or Non-entailment. Kim has decided to use two features only, whether the premise contains the word *not* and whether the hypothesis contains *not*. The 25 observations are summarized in the following table.

| P contains "not" | H contains "not" | class | Number of obs. |
|---|---|---|---|
| yes | yes | entailment | 4 |
| yes | no | non-entail | 6 |
| no | yes | non-entail | 3 |
| no | no | entailment | 12 |
| all other combinations | | | 0 |

a) Construct a decision tree classifier from the training data. (3p)

b) What is the accuracy of the classifier on the training data? (1p)

c) What is the precision and recall of the classifier for the entailment class? (1p)

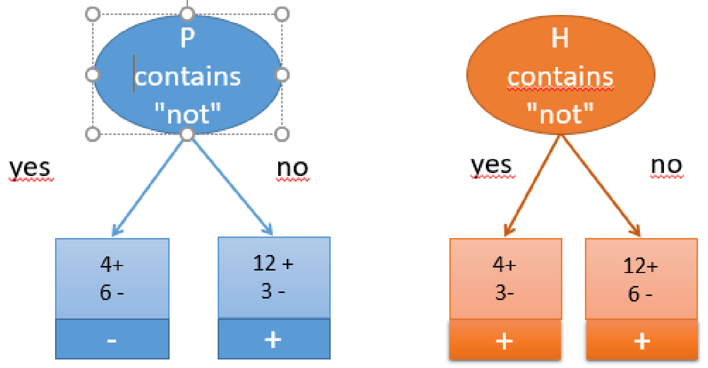
d) Suppose you prune the tree after the first stump. What is now the accuracy for the classifier and the precision and recall for the entailment class? (1p)

e) Make a plot showing the points and the decision boundary/boundaries for the decision tree classifier. (2p)

f) What is the best accuracy a logistic regression classifier can achieve on this data set? Sketch a decision boundary for such a classifier in the same figure. (2p)

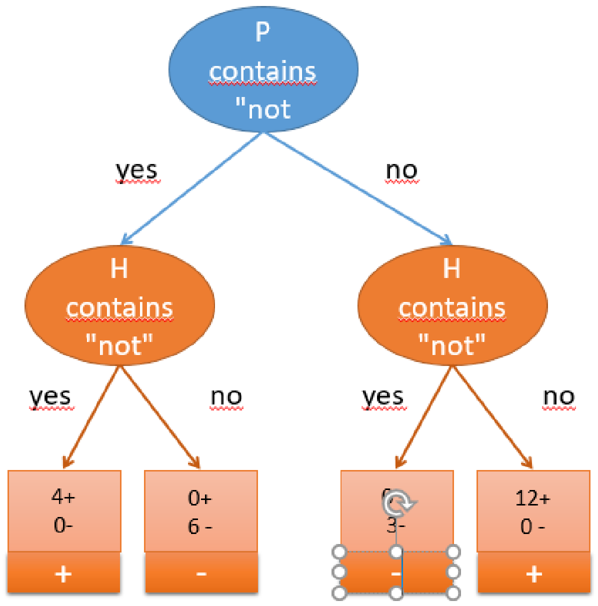
## Suggested Solution

a) There are two stumps



We consider the majority class and count how many are correct
<u>P contains not</u>: 6+12=18/25    <u>H contains not</u>: 4+12=16/25
Choose the attribute which is most informative, here: <u>P contains not</u>

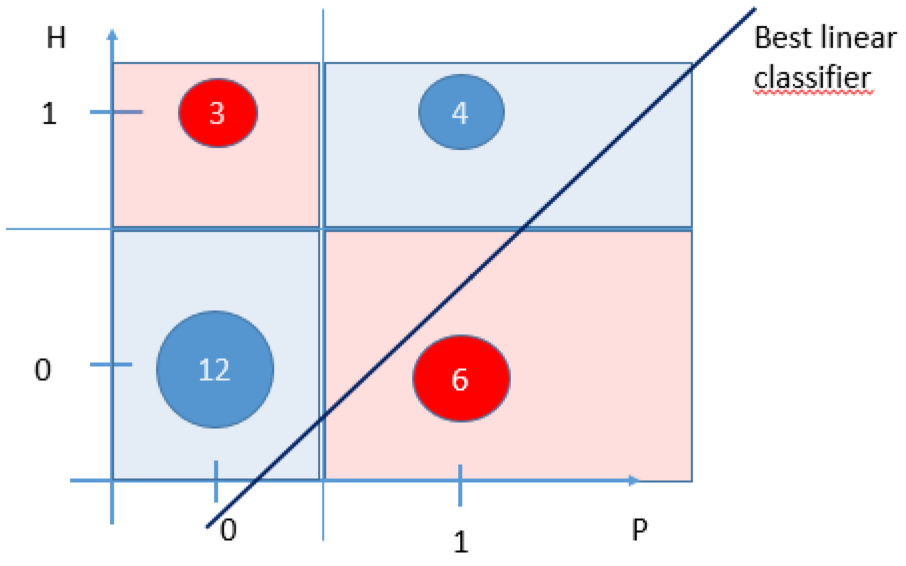
<u>Final tree:</u>



b+c) Vi see that the full tree classifies all the training data correctly. Hence accuracy, precision and recall are all 1.

d) We saw that the top stump corresponding to the feature ‘P contains “not” ′ has accuracy 18/25. For the class ‘entailment’ we see that
Of the 16 entailment cases, it classifies 12 as entailment, hence recall is 12/16=3/4=0.75
It classifies 15 items as entailment, hence precision is 12/15= 0.8

e)



The decision boundary for the decision tree consists of the horizontal and vertical lines crossing in the middle as indicated by the colored background rectangles. The best linear classifier and best logistic regression classifier is indicated by the diagonal line. It classifies the points to the left above it as blue and entailment. Hence it makes 3 misclassifications and has an accuracy of 22/25. For the entailment class, the Recall is 1 while the Precision is (12+4)/(12+4+3) = 16/19.

## 11) Markov Property (4p)

Explain the Markov property with reference to the following formulas:
Formula 1:

$$Pr(r_t = r', s_{t+1} = s' | s_t, a_t).$$

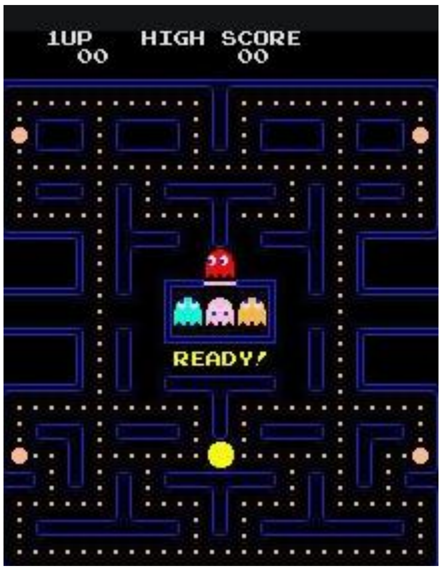
Formula 2:

$$Pr(r_t = r', s_{t+1} = s' | s_t, a_t, r_{t-1}, s_{t-1}, a_{t-1}, \ldots r_1, s_1, a_1, r_0, s_0, a_0)$$

(1p)

Do the following problems fulfill the Markov Property? Briefly explain why. If not, specify some of the information we lose by modelling them as Markov Decision Processes. Specify any additional assumptions you make. (1p for each)

a) Playing chess, assuming the state is a description of the positions of all chess pieces on the board.
b) Driving a car assuming the state is an image of what is currently in front of the car and actions are accelerate, break, turn left or turn right.
c) Playing the computer game Pac-Man assuming the state is an image of the current situation in the game.



## Suggested Solution:

An MDP assumes states have enough information about the environment to be able to calculate the probabilities of future states and rewards without looking at past states.

The first formula shows that the probability distribution for receiving a given reward and ending up in a given next state only depends on the current state and action. The second shows that the same probability distribution depends on all past states and actions. If we assume we are working with an MDP, we assume probabilities are determined by the much simpler first formula rather than the second.

a) Yes. Because knowing the state of the chess board, we know exactly the result of a given action.
b) No. We lose a lot of information, including all info on any dangers behind us or slightly out of view, and on the speed of the car. Some of this information would be useful to estimate future states and rewards, and could have been available given the full state history of the car.
c) Could accept both yes and no, given good arguments. Yes. Because knowing the state of the pacman game from the image, we can model exactly the outcome of an action. Although it may be sensible to argue for no, given that the current state does not give info on the direction ghosts are moving in, which does impact the resulting next state and potentially the reward.

## 12) Unsupervised Learning (20p)

Alice and Bob, your colleagues from the astrophysics department, have given you a collection of astronomical data[1] describing exoplanets in different star systems. Each exoplanet is described by the distance from its orbiting star in AU (astronomical units), its mass (as multiples of the Earth), and the degree of light reflection (as an albedo integer). See table below.

[1] Exoplanet names are real. All the other details are made up.

**a) Visualization (2p)**

| | AU from star | Mass | Albedo |
|---|---|---|---|
| **HD 209458 b** | 2 | 3 | 7 |
| **HD 189733 b** | 5 | 3 | 3 |
| **51 Pegasi b** | 7 | 2 | 5 |
| **PSR B1257+12 B** | 3 | 5 | 6 |
| **PSR B1257+12 C** | 5 | 4 | 5 |
| **OGLE-TR-56 b** | 7 | 4 | 3 |
| **Fomalhaut b** | 3 | 3 | 8 |
| **2M1207 b** | 4 | 3 | 7 |

Some of these data (about half) have been collected using the *transit detection method* and others (about half) using an *infrared detection method*. Alice and Bob know that these two methods are sensitive to exoplanets with different features, but they do not know which sample has been collected with which method.

Alice argues that looking at *AU from the star* and *albedo* may help them infer which observations were performed with which techniques; Bob holds that looking at *AU from the star* and *mass* may provide a better perspective to group the exoplanets by their discovery method.

**Plot the data first according to Alice's hypothesis and then Bob's hypothesis. Which hypothesis seems more likely?**

**b) k-means (5p)**

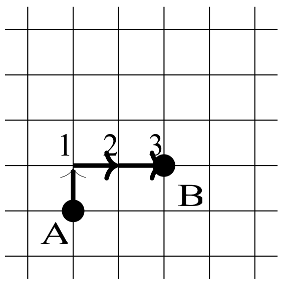
To give more grounding to your conclusions, you decide to run the *k-means algorithm* on your data using two clusters, one for each detection method. Let us call one cluster the blue cluster, and the other one the red cluster.

**Run three iterations of k-means (assignment, recomputation of the centroids) on Alice's and Bob's data.**

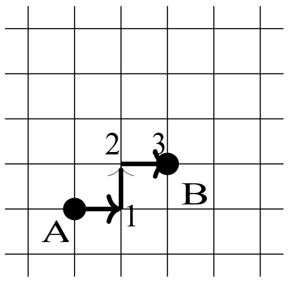
Initialize the blue cluster at (3,2), and the red cluster at (8,4).

In the assignment phase, use as a distance function the Manhattan distance $D_{Man}[x_i,x_j]$, that gives you the number of straight segments necessary to get from one point to the other, for example:

If a point is the same distance from the center of both clusters, assign it to the blue cluster.



(a) Manhattan distance



(b) Notice that the distance is independent from the path.

In the recomputation of the centroids, round the values of the nearest integer:

$$5.3 \rightarrow 5$$
$$2.5 \rightarrow 3$$
$$3.8 \rightarrow 4$$

**How do your results agree with your conclusions from the visualization exercise?**

**c) Quantitative evaluation (5p)**

Bob and Alice look very interested in your results: it seems that clustering based on a given pair of features is better than clustering on another set of features. However, they are uneasy accepting a solution based on an intuitive visualization. They ask if your results may be given a quantitative explanation.

You think that an easy way would be to compute the *separation* between the clusters, that is computing the distance between the blue point that is closest to the red cluster and the red point that is closest to the blue cluster. This measure would quantify the gap between the two clusters.
**Compute the separation for the clustering of Alice's data and Bob's data.**

This measure would quantify the gap between the two clusters.
**What would you conclude from the computation of cluster separation?**

Yet, you feel this measure is not very robust.
**What problem could you imagine having when using cluster separation?**

You ask around your colleagues, and Yoshua explains to you that there are two important measures to evaluate clustering: the *inter-cluster distance*, measuring how separate two clusters are, and the *intra-cluster distance*, measuring how compact a cluster is.
**Compute the inter-cluster distance for the clustering of Alice's data and Bob's data (do not round to integers).**

To compute inter-cluster distance simply compute the distance between the centroid of the red and blue cluster: $D_{inter}[c_{blue}, c_{red}] = D_{Man}[t_{blue}, t_{red}]$, where $c$ is a cluster and $t$ is a centroid.
**Compute the intra-cluster distance for the clustering of Alice's data and Bob's data (do not round to integers).**

Differently from the inter-cluster distance, the intra-cluster must be computed for each cluster individually. For each cluster, red or blue, compute the average distance of all the cluster points from the cluster center. For the blue cluster:
$D_{intra}[c_{blue}] = \frac{1}{N_{blue}} \sum_{x \in c_{blue}} D[x, t_{blue}]$; similarly for the red cluster. Average then the intra-cluster distance of the blue and red cluster to get the overall intra-cluster distance for Alice and Bob: $\frac{1}{2}(D_{intra}[c_{blue}] + D_{intra}[c_{red}])$.

A good cluster is a cluster that clumps its point tightly close to each other, and that is far removed from other cluster. It is natural to assess the goodness of your clustering as the ratio between inter-cluster distance (which you want to be big) and intra-cluster distance (which you want to be small).
**Compute the ratio of inter-cluster distance and intra-cluster distance for the clustering of Alice's data and Bob's data (do not round to integers). How does this confirm/reject your previous conclusions?**

**d) Processing new data (3p)**

Alice and Bob are happy with your solution and decide to adopt the clustering you argued being the best one. From now on, we will use only the clustering that you proved being the best. Now new data has come in:

| | AU from star | Mass | Albedo |
|---|---|---|---|
| **Beta Pictoris c** | 9 | 3 | 6 |
| **K2-282c** | 6 | 5 | 7 |
| **Kepler-1658b** | 2 | 2 | 8 |

**Start from the chosen clustering, plot the new data points and assign them to the correct cluster.**

### e) Outliers (2p)

There is a further recording, coming from another institution, that Alice and Bob would like to process:

| | AU from star | Mass | Albedo |
|---|---|---|---|
| **Luyten 98-59 d** | 22 | 3 | 3 |

Alice and Bob are not certain about the quality of this recording and ask your opinion.

**Use the chosen clustering, plot the new data point. What do you think about this observation?**

For your own interest, you decide to analyze how this new data point will affect the clustering process.

**Restart from the original data set of eight data points; place the centroids of the two clusters as you computed them at the end of Section 1.2; add the new data on Luyten 98-59 d and run two iterations of the k-means algorithm. What happens to the clusters?**

### f) Rescaling (3p)

After discussing with other colleagues at a conference, Alice and Bob became suspicious that the recordings of albedo may be wrong. Following the suggestion of Eve, they are thinking about reducing by half all the recorded values of albedo. They present this possible change to you, and ask your opinion.

**How would you interpret the change that they have proposed?**

In particular, they are concerned whether this change would affect your results.

**Apply the transformation to the original data. Then run two iterations of k-means with the same initialization used in Section 1.2 on Alice's data. What do you observe?**

When halving the observed values of albedo, always round down to the closest integer:

$$3.5 \rightarrow 3$$

**Is k-means insensitive to the suggested transformation? If not, how would you tackle this inconsistency?**

## Suggested Solution: See “task 12 solution” PDF

## 13) Particle Swarm Optimization (PSO) and Developmental Systems (3p – 1 per question)

a) Describe what happens when the position of all particles in PSO are set to the same value of a local optimum (Think about fitness landscapes). How could you adjust PSO to get out of the local optimum to potentially find the global optimum without resetting the particles to random positions initially? Describe your approach in up to 100 words.

b) An L-System is a parallel rewrite method. Describe how from an alphabet {h,j}, the axiom h will be rewritten when using the rewrite rules: h->jjh and j->h. Write down three iterations/recursions.

c) When visualizing a string of an L-System, it is useful to implement a bracketed L-System. Describe what ‘+’, ‘-’, ‘[‘ and ‘]’ in such L-Systems are used for.

## Suggested Solutions

a) This question is very subjective and there are many possible valid solutions. You can for example set the initial velocity of every particle to a very high number. You can also add a repulsion function that makes particles repel one another. Thereby particles are gathered around a ‘center of mass’ but cannot squeeze on the same position. Related to how ‘boids’ flock which was mentioned in the lecture.

b) *h := jjh := hhjjh := jjhjjhhhjjh*

c) ‘+’ and ‘-’ refer to turning left and right with a predefined angle. The brackets ‘[‘ and ‘]’ push and pop the state when drawing an L-System. Pushing meaning that when the bracket is being read, the position and angle are being stored. The pop returns the previously stored position and angle. Bracketed L-Systems therefore enable branching. The string of an expanded L-System can be read as a set of instructions. Classical example is by having F draw a straight line. F+F[FF]+F would read as: (1)draw (2) turn (3) draw (4) push (5) draw (6) draw (7) pop (8) turn (9) draw.

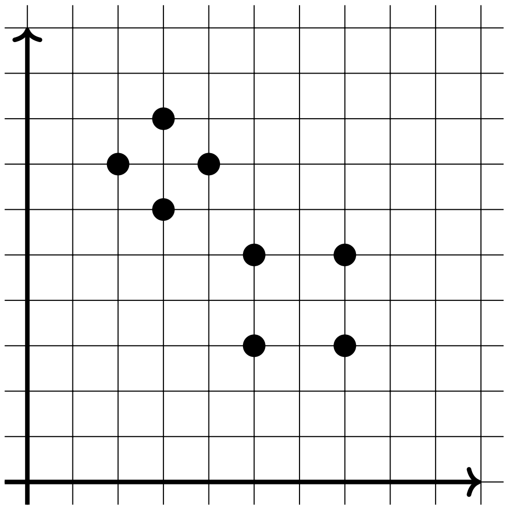
# Unsupervised Learning Problems

## 1 Unsupervised Learning Problem

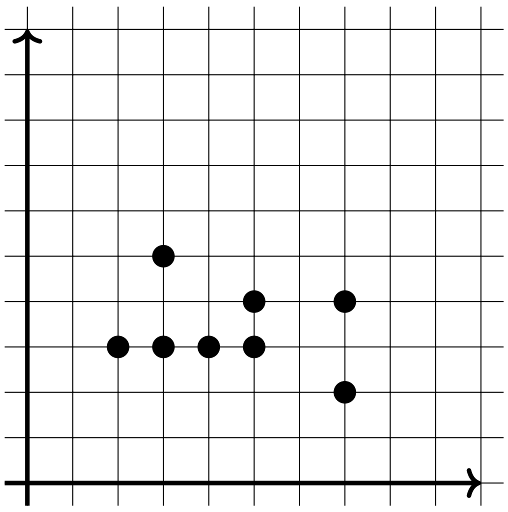
### 1.1 Visualization

**Plot the data first according to Alice's hypothesis and then Bob's hypothesis. Which hypothesis seems more likely?**

Alice's hypothesis:



Bob's hypothesis:

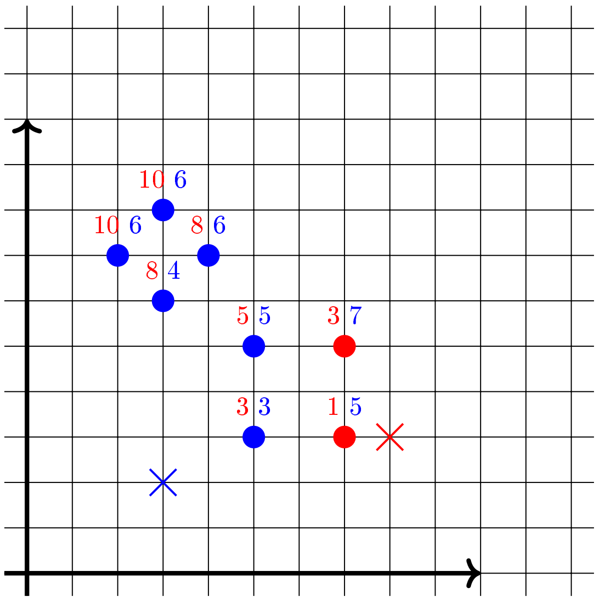


Alice's hypothesis seems to highlight more structure in the data.

## 1.2 K-Means

**Run three iterations of k-means (assignment, recomputation of the centroids) on Alice's and Bob's data.**

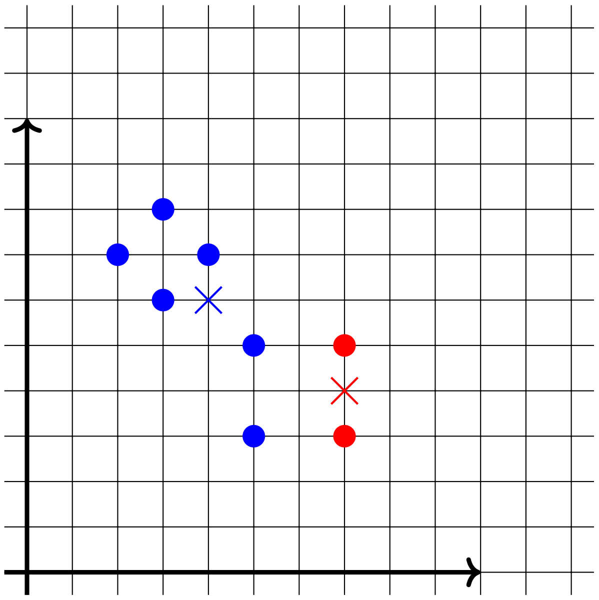
Alice's data - Iteration 1 - Assignment:



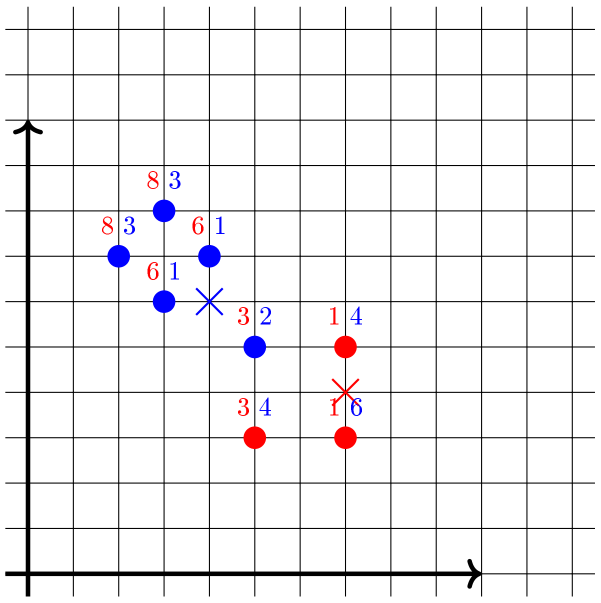
Alice's data - Iteration 1 - Recomputation:

$x = 2 + 3 + 3 + 4 + 5 + 5 = 22/6 = 4$; $y = 3 + 5 + 6 + 7 + 7 + 8 = 36/6 = 6$

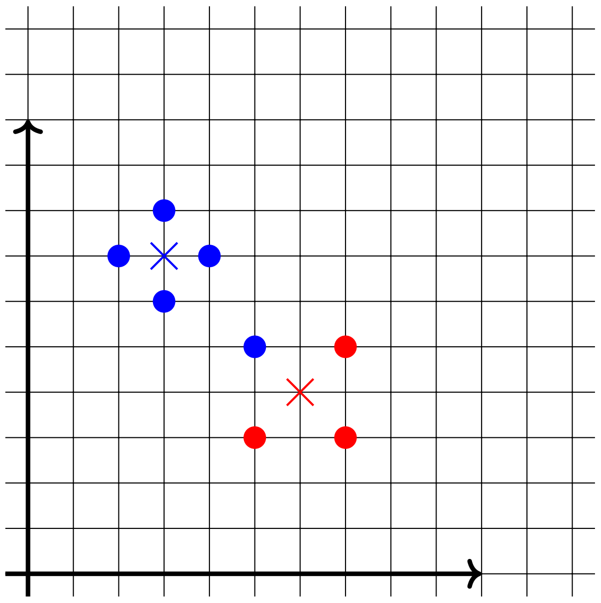
$x = 7 + 7 = 14/2 = 7$; $y = 3 + 5 = 8/2 = 4$



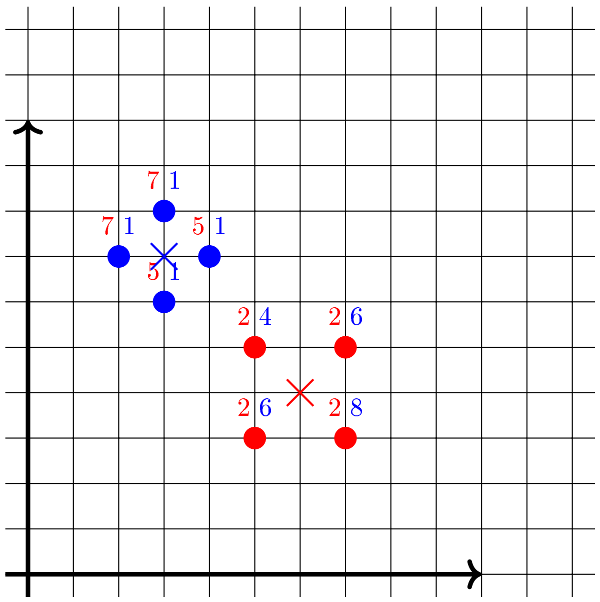
Alice's data - Iteration 2 - Assignment:



Alice's data - Iteration 2 - Recomputation:
$x = 2 + 3 + 3 + 4 + 4 + 5 = 17/6 = 3$; $y = 5 + 6 + 7 + 7 + 8 = 33/5 = 7$
$x = 5 + 7 + 7 = 19/3 = 6$; $y = 3 + 3 + 5 = 11/3 = 4$



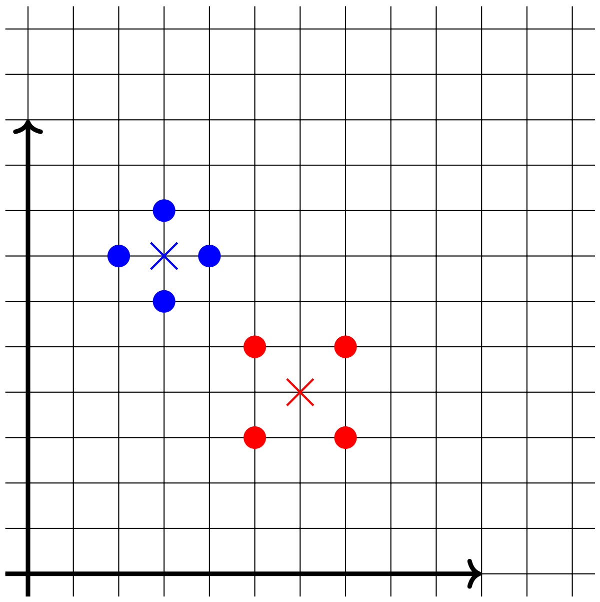
Alice's data - Iteration 3 - Assignment:



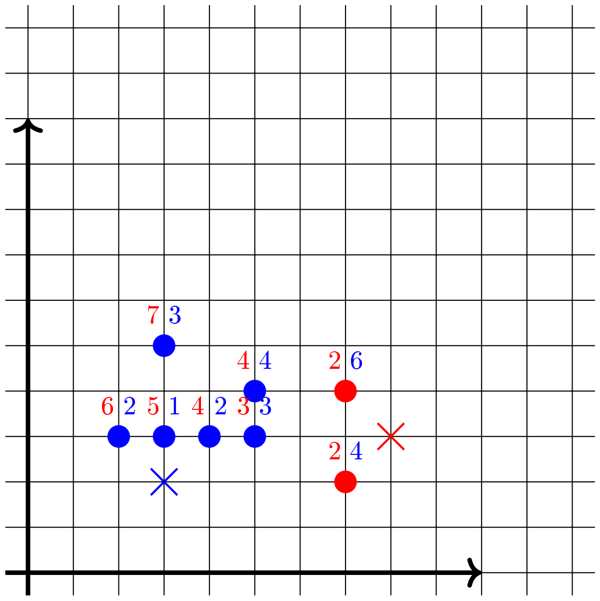
Alice's data - Iteration 3 - Recomputation:

$x = 2 + 3 + 3 + 4 = 12/4 = 3$; $y = 6 + 7 + 7 + 8 = 28/4 = 7$

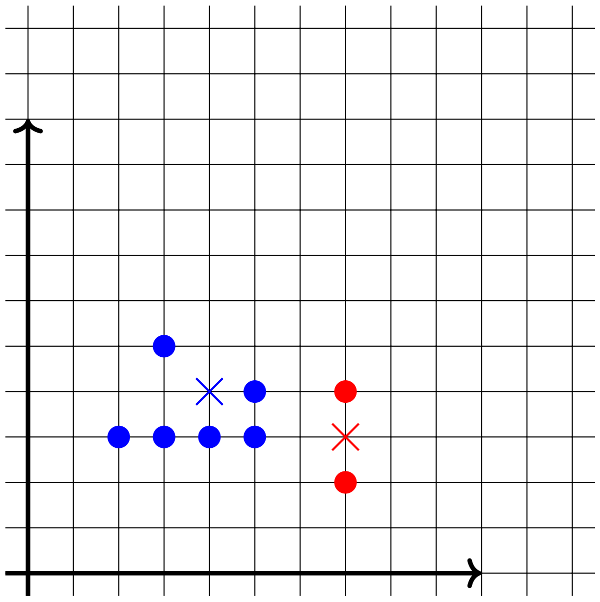
$x = 5 + 5 + 7 + 7 = 24/4 = 6$; $y = 3 + 3 + 5 + 5 = 16/4 = 4$



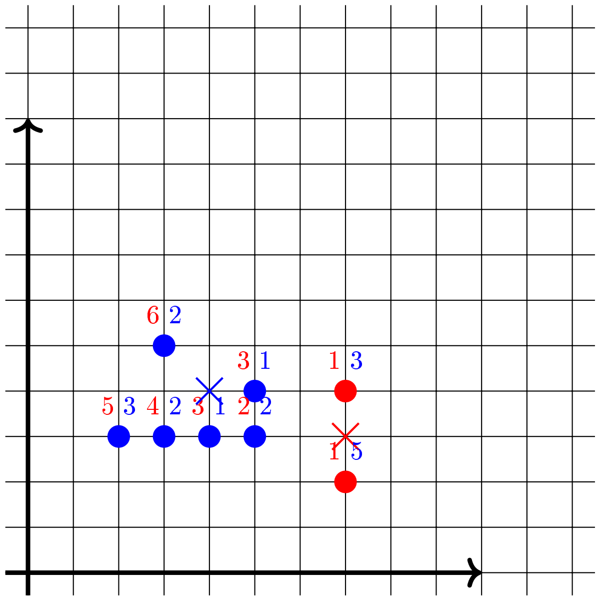
*Solution:*
Bob's data - Iteration 1 - Assignment:



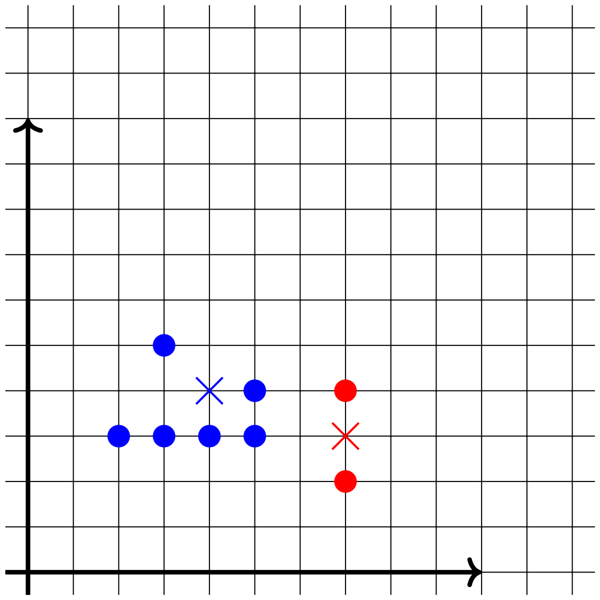
Bob's data - Iteration 1 - Recomputation:
$x = 2 + 3 + 3 + 4 + 5 + 5 = 22/6 = 4$; $y = 3 + 3 + 3 + 3 + 4 + 5 = 21/6 = 4$
$x = 7 + 7 = 14/2 = 7$; $y = 2 + 4 = 6/2 = 3$



*Solution:*
Bob's data - Iteration 2 - Assignment:



Bob's data - Iteration 2 - Recomputation:
$x = 2 + 3 + 3 + 4 + 5 + 5 = 22/6 = 4$; $y = 3 + 3 + 3 + 3 + 4 + 5 = 21/6 = 4$
$x = 7 + 7 = 14/2 = 7$; $y = 2 + 4 = 6/2 = 3$



Iteration 3 is the same. Nothing changed.

## 1.3 Quantitative Evaluation

**Compute the separation for the clustering of Alice's data and Bob's data.**

Alice: $D_{inter} = 3$ Bob: $D_{inter} = 2$

**What would you conclude from the computation of cluster separation?**

Alice's clustering provides a better separation.

**What problem could you imagine having when using cluster separation?**

Outliers have very large gaps.

**Compute the inter-cluster distance for the clustering of Alice's data and Bob's data (do not round to integers).**

Alice: $D_{inter} = 6$ Bob: $D_{inter} = 4$

**Compute the intra-cluster distance for the clustering of Alice's data and Bob's data (do not round to integers).**

Alice: $D_{intra}\left[c_b lue\right] = 1$, $D_{intra}\left[c_r ed\right] = 2$, $D_{intra} = 1.5$ Bob: $D_{intra}\left[c_b lue\right] = (1+1+2+2+2+3)/6 = \frac{11}{6}$, $D_{intra}\left[c_r ed\right] = 1$, $D_{intra} = \frac{17}{12}$

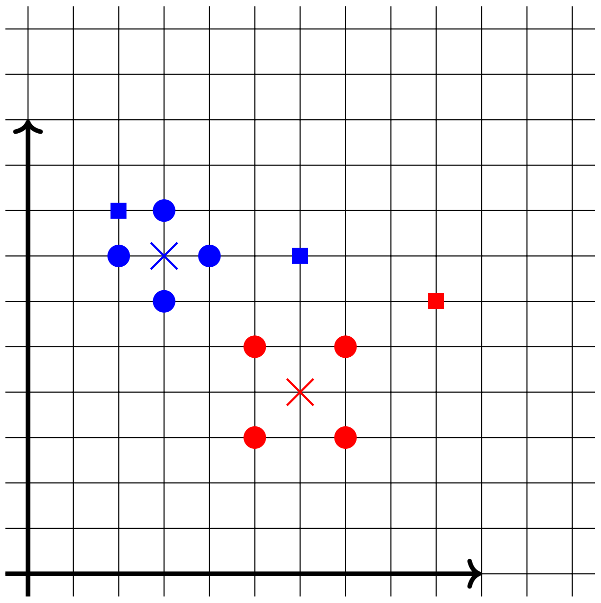
**Compute the ratio of inter-cluster distance and intra-cluster distance for the clustering of Alice's data and Bob's data (do not round to integers). How does this confirm/reject your previous conclusions?**

Alice: $6/1.5 = 4$ Bob: $4/(\frac{17}{12}) = 2.82$ Consistently with before, Alice's clustering is better.

## 1.4 Clustering of new data

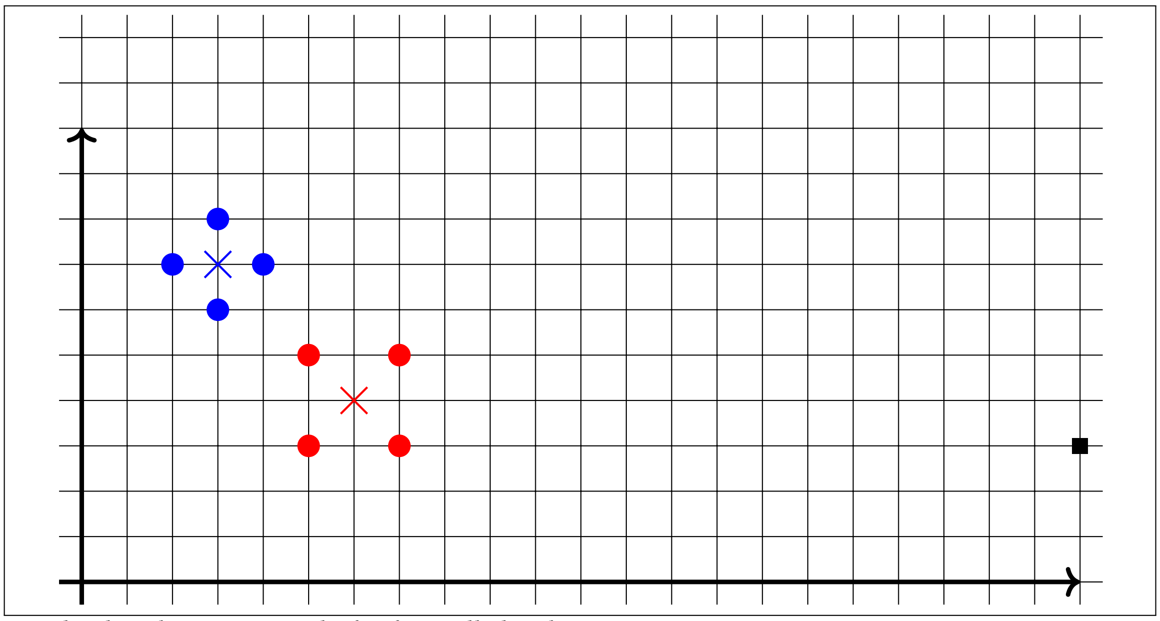
**Start from the chosen clustering, plot the new data points and assign them to the correct cluster.**

Using Alice's clustering: Beta Pictoris c goes to red cluster; K2-282c goes to blue cluster (this point is evenly far from the two centers though); Kepler-1658b goes to blue cluster.



## 1.5 Outliers

**Use the chosen clustering, plot the new data point. What do you think about this observation?**

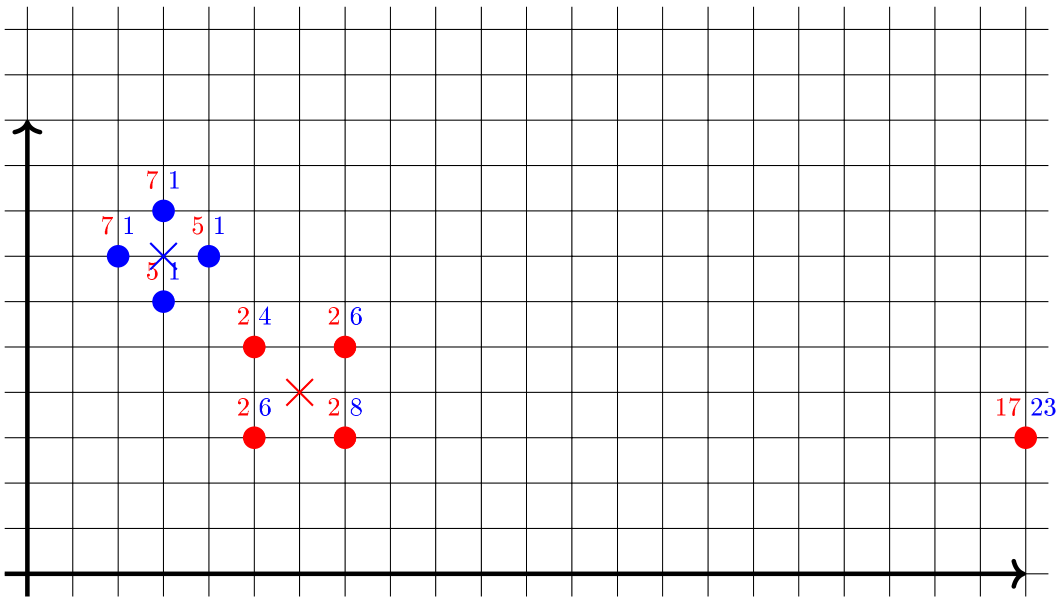


The data lies suspiciously far from all the clusters.

**Restart from the original data set of eight data points; place the centroids of the two clusters as you computed them at then end of Section 1.2; add the new data on Luyten 98-59 d and run two iterations of the k-mean algorithm.**

The clustering is derailed by the outlier

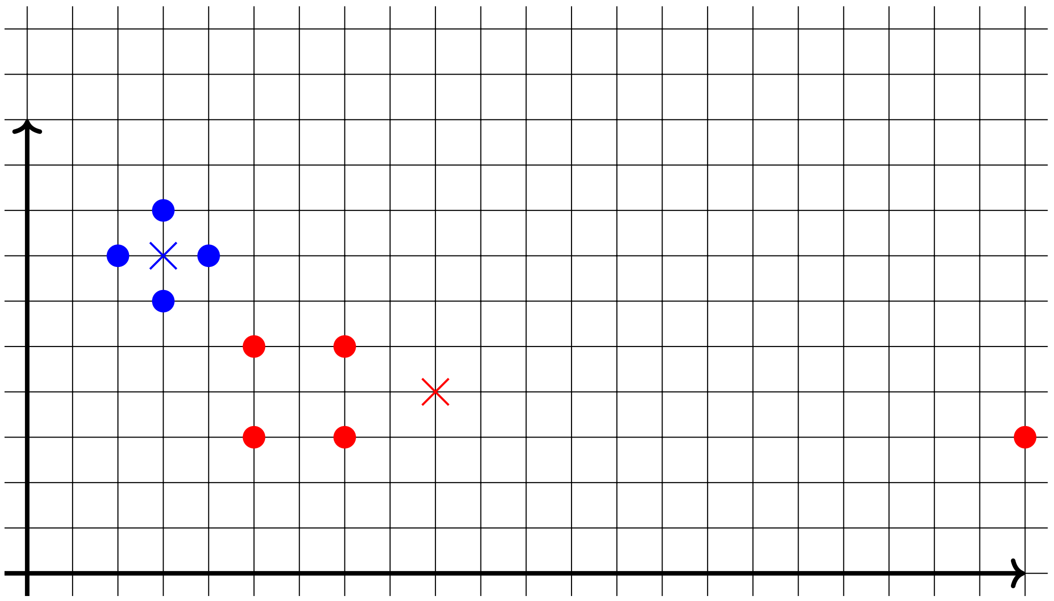
Alice's data - Iteration 1 - Assignment:



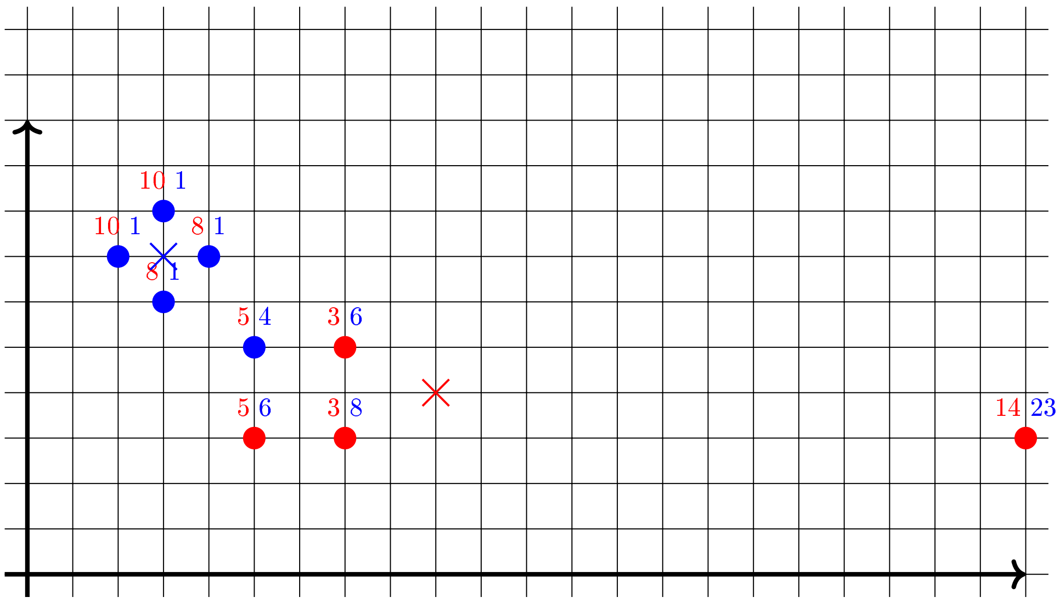
Alice's data - Iteration 1 - Recomputation:

$x = 2 + 3 + 3 + 4 = 12/4 = 3$; $y = 6 + 7 + 7 + 8 = 28/4 = 7$

$x = 5 + 5 + 7 + 7 + 22 = 46/5 = 9$; $y = 3 + 3 + 5 + 5 + 3 = 19/5 = 4$



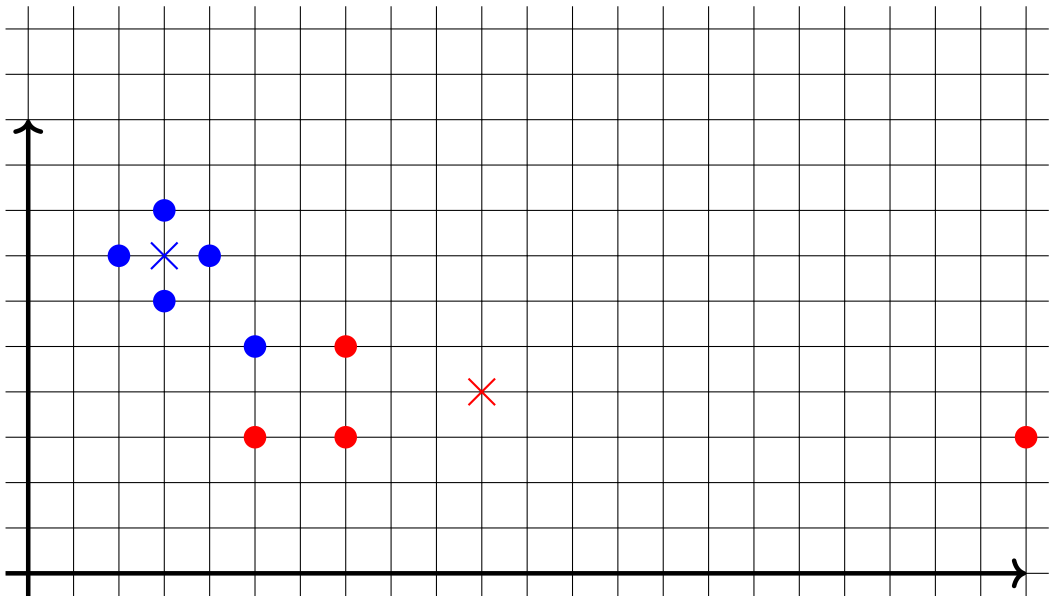
Alice's data - Iteration 2 - Assignment:



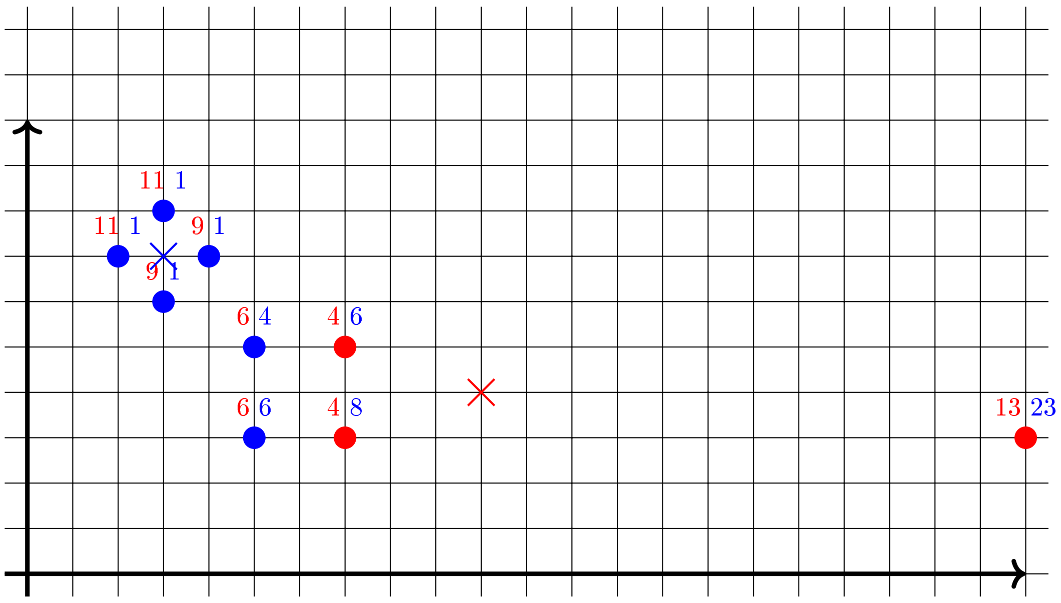
Alice's data - Iteration 2 - Recomputation:

$x = 2 + 3 + 3 + 4 + 5 = 17/5 = 3$; $y = 5 + 6 + 7 + 7 + 8 = 33/5 = 7$

$x = 5 + 7 + 7 + 22 = 41/4 = 10$; $y = 3 + 3 + 5 + 3 = 14/4 = 4$



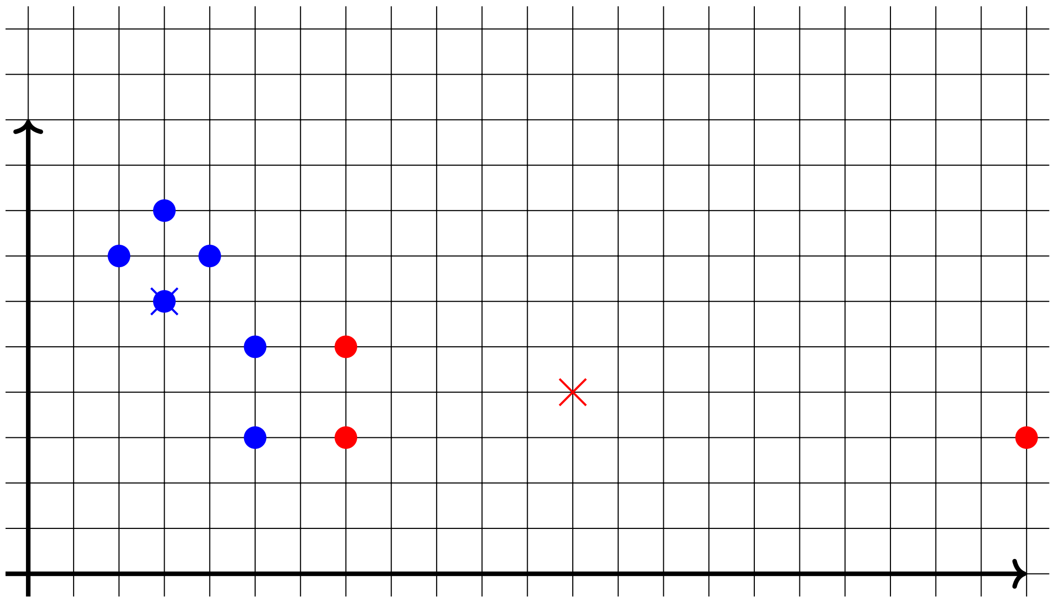
Alice's data - Iteration 3 - Assignment:



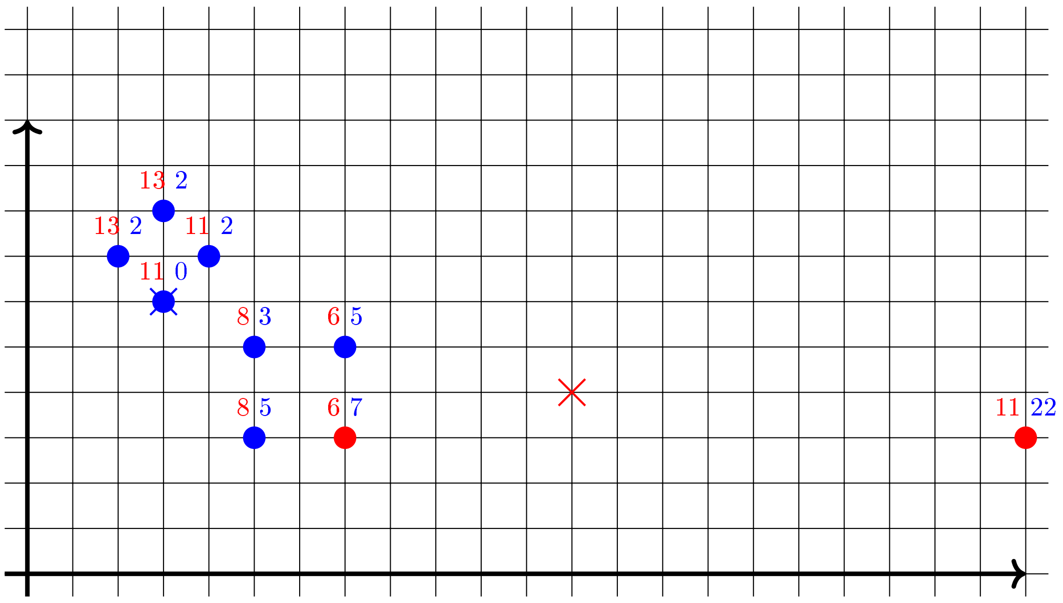
Alice's data - Iteration 3 - Recomputation:

$x = 2 + 3 + 3 + 3 + 4 + 5 = 20/6 = 3$; $y = 3 + 5 + 6 + 7 + 7 + 8 = 36/6 = 6$

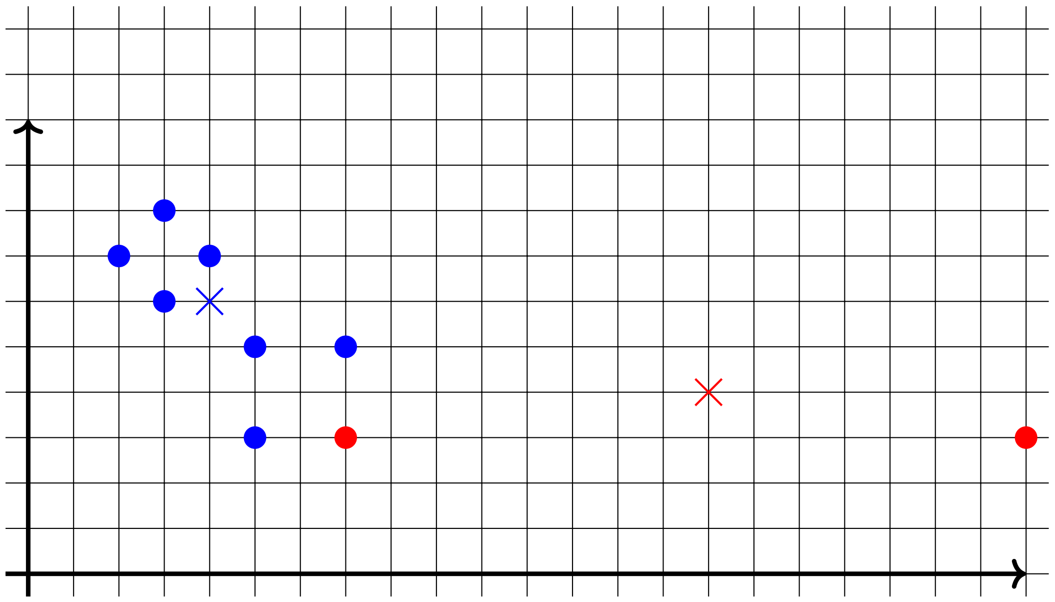
$x = 7 + 7 + 22 = 36/3 = 12$; $y = 3 + 5 + 3 = 11/3 = 4$



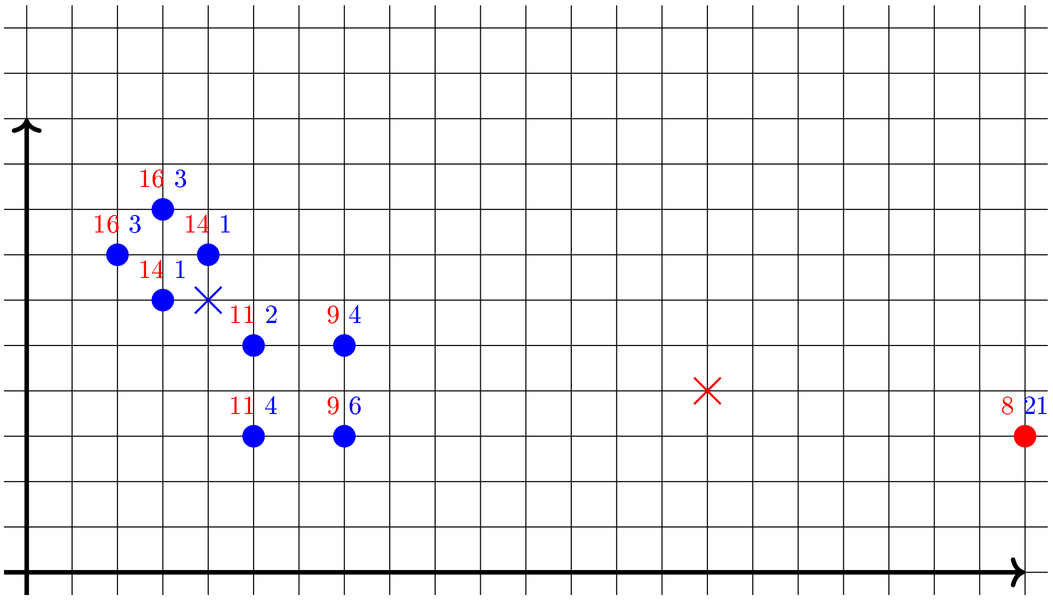
Alice's data - Iteration 4 - Assignment:



Alice's data - Iteration 4 - Recomputation:
$x = 2 + 3 + 3 + 3 + 4 + 5 + 7 = 27/7 = 4$; $y = 3 + 5 + 5 + 6 + 7 + 7 + 8 = 41/7 = 6$
$x = 7 + 22 = 29/2 = 15$; $y = 3 + 3 = 6/2 = 3$



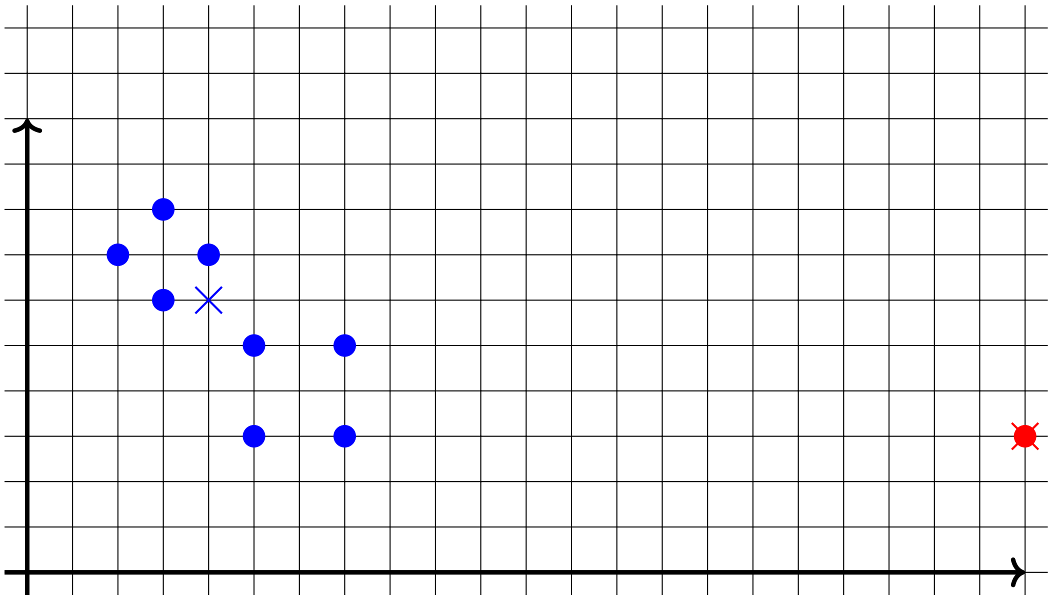
Alice's data - Iteration 5 - Assignment:



Alice's data - Iteration 5 - Recomputation:

$x = 2+3+3+3+4+5+7+7 = 34/8 = 4$; $y = 3+3+5+5+6+7+7+8 = 44/8 = 6$

$x = 22$; $y = 3$



## 1.6 Rescaling

**How would you interpret the change that they have proposed?**

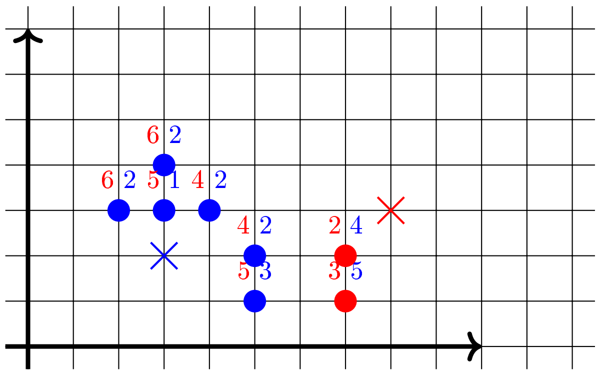
New representation or feature processing or feature scaling...

**Apply the transformation to the original data. Then run two iterations of k-means with the same initialization used in Section 1.2 on Alice's data. What do you observe?**

Rescaled data:

| | AU from star | Mass | Albedo |
|---|---|---|---|
| HD 209458 b | 2 | 3 | 3 |
| HD 189733 b | 5 | 5 | 1 |
| 51 Pegasi b | 7 | 2 | 2 |
| PSR B1257+12 B | 3 | 5 | 3 |
| PSR B1257+12 C | 5 | 4 | 2 |
| OGLE-TR-56 b | 7 | 4 | 1 |
| Fomalhaut b | 3 | 3 | 4 |
| 2M1207 b | 4 | 3 | 3 |

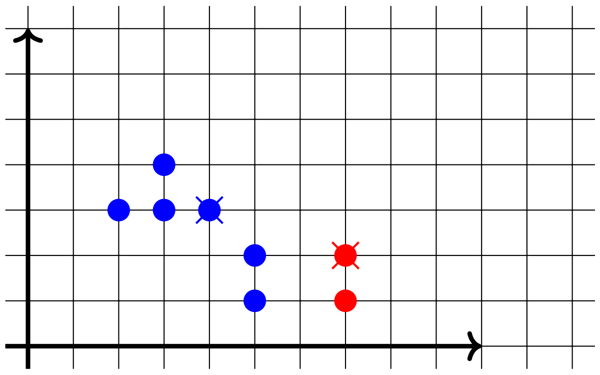
Alice's data - Iteration 1 - Assignment:



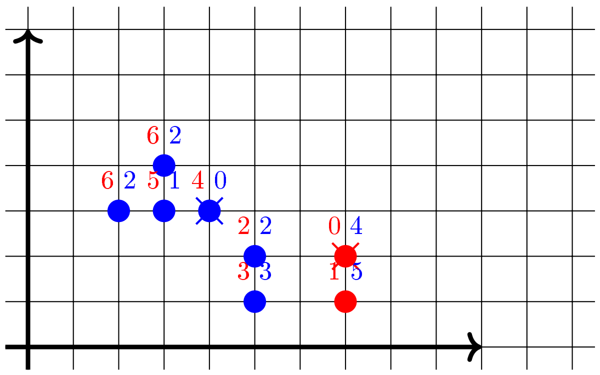
Alice's data - Iteration 1 - Recomputation:

$x = 2 + 3 + 3 + 4 + 5 + 5 = 22/6 = 4$; $y = 1 + 2 + 3 + 3 + 3 + 4 = 16/6 = 3$

$x = 7 + 7 = 14/2 = 7$; $y = 1 + 2 = 3/2 = 2$



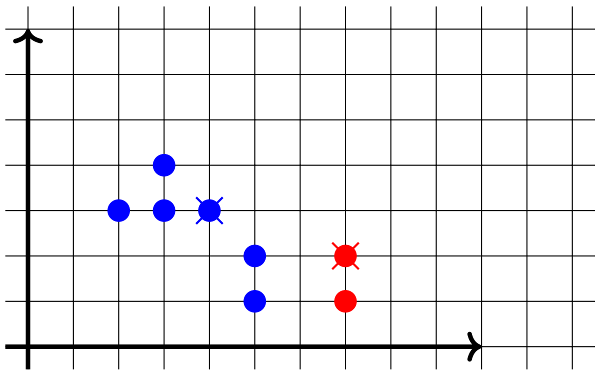
Alice's data - Iteration 2 - Assignment:



Alice's data - Iteration 2 - Recomputation:

$x = 2 + 3 + 3 + 4 + 5 + 5 = 22/6 = 4$; $y = 1 + 2 + 3 + 3 + 3 + 4 = 16/6 = 3$

$x = 7 + 7 = 14/2 = 7$; $y = 1 + 2 = 3/2 = 2$



Rescaling the data affects the results of clustering.

**Is k-means insensitive to the suggested transformation? If not, how would you tackle this inconsistency?**

No, it is not. You could re-run the algorithm with different random initializations.