Exercise #5

Introduction to Knowledge Management, WS2016

Camilo Thorne

(based on slides by T. Szyler and A. Melo)

Mannheim, 11.04.2016

Exercise 05 - Question 1a

Neural Networks

The preliminary plan was to fit each tank with a digital camera hooked up to a computer. The computer would continually scan the environment outside for possible threats (such as an enemy tank hiding behind a tree), and alert the tank crew to anything suspicious. Computers are really good at doing repetitive tasks without taking a break, but they are generally bad at interpreting images. The only possible way to solve the problem was to employ a neural network.

The Implementation

The research team went out and took 100 photographs of tanks hiding behind trees, and then took 100 photographs of trees - with no tanks. They took half the photos from each group and put them in a vault for safe-keeping, then scanned the other half into their mainframe computer. The huge neural network was fed each photo one at a time and asked if there was a tank hiding behind the trees. Of course at the beginning its answers were completely random since the network didn't know what was going on or what it was supposed to do. But each time it was fed a photo and it generated an answer, the scientists told it if it was right or wrong. If it was wrong it would randomly change the weightings in its network until it gave the correct answer. Over time it got better and better until eventually it was getting each photo correct. It could correctly determine if there was a tank hiding behind the trees in any one of the photos.

Verification

But the scientists were worried: had it actually found a way to recognize if there was a tank in the photo, or had it merely memorized which photos had tanks and which did not? This is a big problem with neural networks, after they have been trained you have no idea how they arrive at their answers, they just do. The question was did it understand the concept of tanks vs. no tanks, or had it merely memorized the answers? So the scientists took out the photos they had been keeping in the vault and fed them through the computer. The computer had never seen these photos before -- this would be the big test. To their immense relief the neural net correctly identified each photo as either having a tank or not having one.

Independent testing

The Pentagon was very pleased with this, but a little bit suspicious. They commissioned another set of photos (half with tanks and half without) and scanned them into the computer and through the neural network. The results were completely random. For a long time nobody could figure out why. After all nobody understood how the neural had trained itself.

Grey skies for the US military

Eventually someone noticed that in the original set of 200 photos, all the images with tanks had been taken on a cloudy day while all the images without tanks had been taken on a sunny day. The neural network had been asked to separate the two groups of photos and it had chosen the most obvious way to do it - not by looking for a camouflaged tank hiding behind a tree, but merely by looking at the colour of the sky. The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not.

Exercise 05 - Question 1a

Neural Networks

The preliminary plan was to fit each tank with a digital camera hooked up to a computer. The computer would continually scan the environment outside for possible threats (such as an enemy tank hiding behind a tree), and alert the tank crew to anything suspicious. Computers are really good at doing repetitive tasks without taking a break, but they are generally bad at interpreting images. The only possible way to solve the problem was to employ a neural network.

The Implementation

The research team went out and took 100 photographs of tanks hiding behind trees, and then took 100
phot
safe
phot
netv ... d it
if it v ... and
bett ... s.

Classification

Why did the neural network fail? Would another classifier have had better results?

Verification

But the scientists were worried: had it actually found a way to recognize if there was a tank in the photo, or had it merely memorized which photos had tanks and which did not? This is a big problem with neural networks, after they have been trained you have no idea how they arrive at their answers, they just do. The question was did it understand the concept of tanks vs. no tanks, or had it merely memorized the answers? So the scientists took out the photos they had been keeping in the vault and fed them through the computer. The computer had never seen these photos before -- this would be the big test. To their immense relief the neural net correctly identified each photo as either having a tank or not having one.

Independent testing

The Pentagon was very pleased with this, but a little bit suspicious. They commissioned another set of photos (half with tanks and half without) and scanned them into the computer and through the neural network. The results were completely random. For a long time nobody could figure out why. After all nobody understood how the neural had trained itself.

Grey skies for the US military

Eventually someone noticed that in the original set of 200 photos, all the images with tanks had been taken on a cloudy day while all the images without tanks had been taken on a sunny day. The neural network had been asked to separate the two groups of photos and it had chosen the most obvious way to do it - not by looking for a camouflaged tank hiding behind a tree, but merely by looking at the colour of the sky. The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not.

Exercise 05 - Question 1a

Neural Networks

The preliminary plan was to fit each tank with a digital camera hooked up to a computer. The computer would continually scan the environment outside for possible threats (such as an enemy tank hiding behind a tree), and alert the tank crew to anything suspicious. Computers are really good at doing repetitive tasks without taking a break, but they are generally bad at interpreting images. The only possible way to solve the problem was to employ a neural network.

The Implementation

The research team went out and took 100 photographs of tanks hiding behind trees, and then took 100
phot
safe
phot
netv
if it v
bett

d it
and
os.

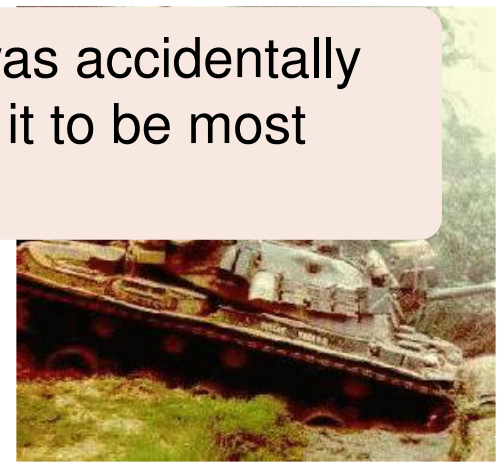
Classification

Why did the neural network fail? Would another classifier have had better results?

Verification

But t
mer
have
und
the p
thes
phot

The training data included a very distinctive attribute that was accidentally related to the task. Any other classifier would have chosen it to be most expressive, too.



Independent testing

The Pentagon was very pleased with this, but a little bit suspicious. They commissioned another set of photos (half with tanks and half without) and scanned them into the computer and through the neural network. The results were completely random. For a long time nobody could figure out why. After all nobody understood how the neural had trained itself.

Grey skies for the US military

Eventually someone noticed that in the original set of 200 photos, all the images with tanks had been taken on a cloudy day while all the images without tanks had been taken on a sunny day. The neural network had been asked to separate the two groups of photos and it had chosen the most obvious way to do it - not by looking for a camouflaged tank hiding behind a tree, but merely by looking at the colour of the sky. The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not.

Exercise 05 - Question 1b

Neural Networks

The preliminary plan was to fit each tank with a digital camera hooked up to a computer. The computer would continually scan the environment outside for possible threats (such as an enemy tank hiding behind a tree), and alert the tank crew to anything suspicious. Computers are really good at doing repetitive tasks without taking a break, but they are generally bad at interpreting images. The only possible way to solve the problem was to employ a neural network.

The Implementation

The research team went out and took 100 photographs of tanks hiding behind trees, and then took 100 phot
safe
phot
netv
if it
bett
Ver

Classification

If the scientists had trained a decision tree to classify the pictures they would have known the reason for the misclassification a lot faster. Why?

d it
and
os.

But the scientists were worried: had it actually found a way to recognize if there was a tank in the photo, or had it merely memorized which photos had tanks and which did not? This is a big problem with neural networks, after they have been trained you have no idea how they arrive at their answers, they just do. The question was did it understand the concept of tanks vs. no tanks, or had it merely memorized the answers? So the scientists took out the photos they had been keeping in the vault and fed them through the computer. The computer had never seen these photos before -- this would be the big test. To their immense relief the neural net correctly identified each photo as either having a tank or not having one.

Independent testing

The Pentagon was very pleased with this, but a little bit suspicious. They commissioned another set of photos (half with tanks and half without) and scanned them into the computer and through the neural network. The results were completely random. For a long time nobody could figure out why. After all nobody understood how the neural had trained itself.

Grey skies for the US military

Eventually someone noticed that in the original set of 200 photos, all the images with tanks had been taken on a cloudy day while all the images without tanks had been taken on a sunny day. The neural network had been asked to separate the two groups of photos and it had chosen the most obvious way to do it - not by looking for a camouflaged tank hiding behind a tree, but merely by looking at the colour of the sky. The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not.

Exercise 05 - Question 1b

Neural Networks

The preliminary plan was to fit each tank with a digital camera hooked up to a computer. The computer would continually scan the environment outside for possible threats (such as an enemy tank hiding behind a tree), and alert the tank crew to anything suspicious. Computers are really good at doing repetitive tasks without taking a break, but they are generally bad at interpreting images. The only possible way to solve the problem was to employ a neural network.

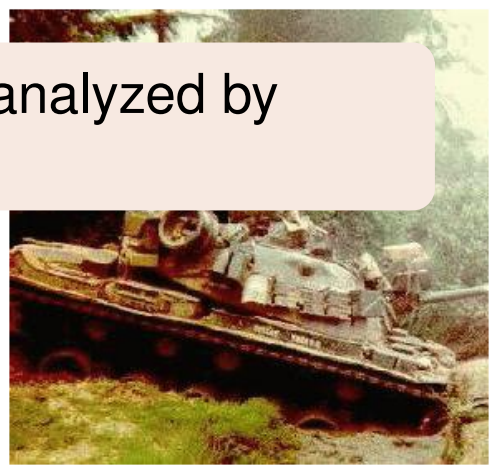
The Implementation

The research team went out and took 100 photographs of tanks hiding behind trees, and then took 100 phot
safe
phot
netv
if it v
bett

d it
and
s.

Ver

But the scientists were worried: had it actually found a way to recognize if there was a tank in the photo, or had it mer
have
und
the
thes
photo as either having a tank or not having one.



Independent testing

The Pentagon was very pleased with this, but a little bit suspicious. They commissioned another set of photos (half with tanks and half without) and scanned them into the computer and through the neural network. The results were completely random. For a long time nobody could figure out why. After all nobody understood how the neural had trained itself.

Grey skies for the US military

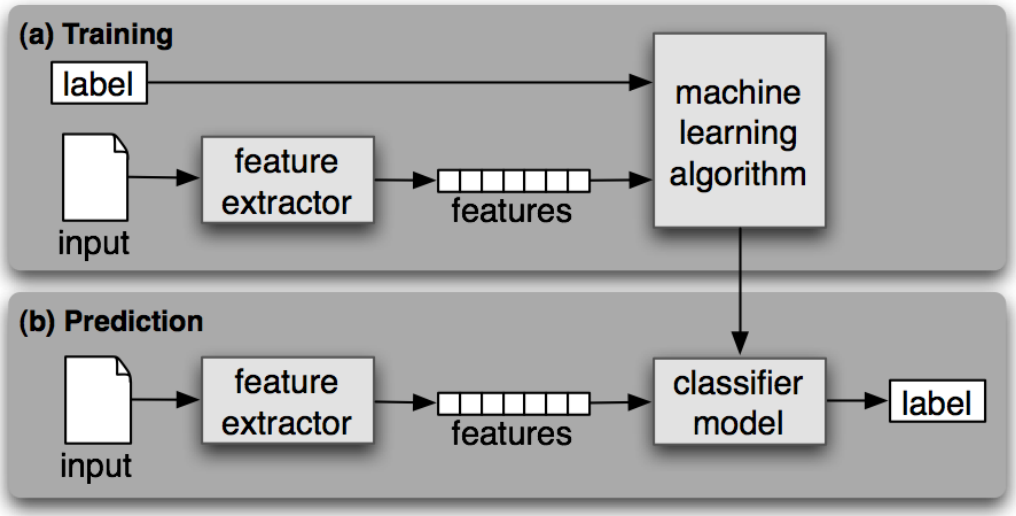
Eventually someone noticed that in the original set of 200 photos, all the images with tanks had been taken on a cloudy day while all the images without tanks had been taken on a sunny day. The neural network had been asked to separate the two groups of photos and it had chosen the most obvious way to do it - not by looking for a camouflaged tank hiding behind a tree, but merely by looking at the colour of the sky. The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not.

Classification

If the scientists had trained a decision tree to classify the pictures they would have known the reason for the misclassification a lot faster. Why?

The rules of a decision tree can be made explicit and thus analyzed by experts making it easy to identify nonsense rules.

(Supervised) Classification - Reminder



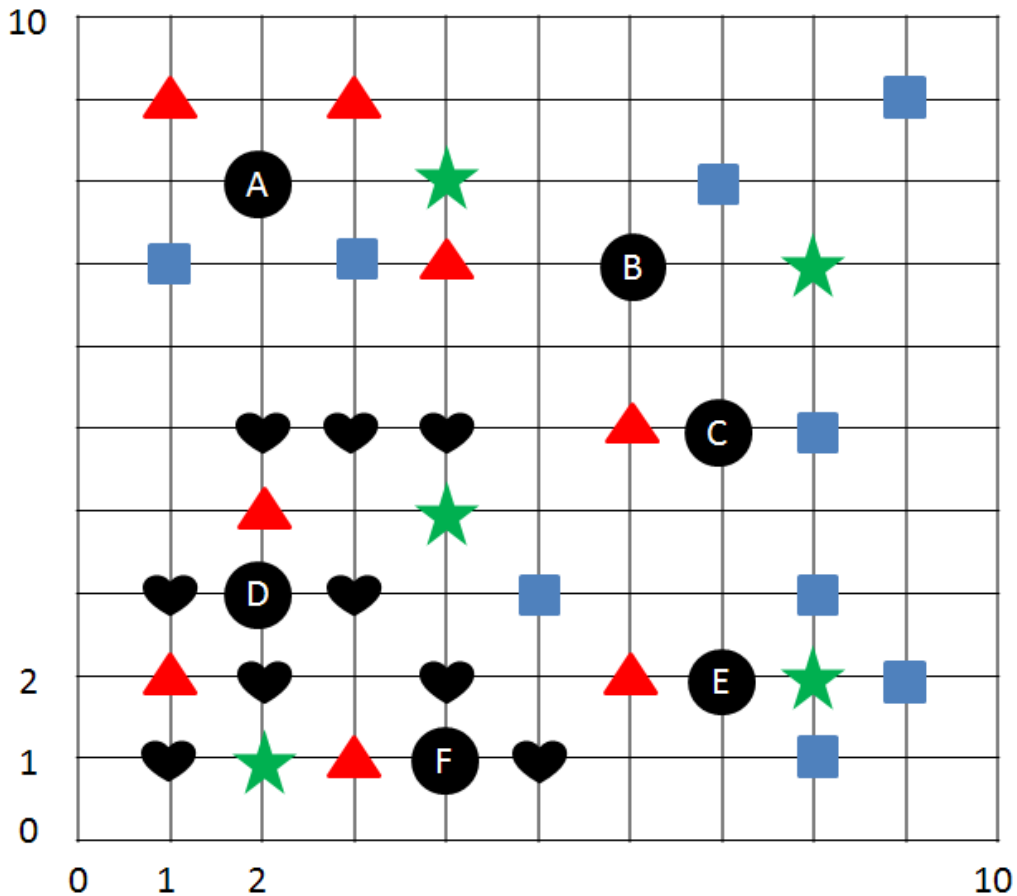
1. classes C $c_1, \ldots, c_{|C|}$
2. features F $X_1, \ldots, X_{|F|}$
3. training set S $(x_1^{(1)}, \ldots, x_{|F|}^{(1)}, c^{(1)})^T, \ldots, (x_1^{(|S|)}, \ldots, x_{|F|}^{(|S|)}, c^{(|S|)})^T$
4. unlabeled data D $(x_1^{(1)}, \ldots, x_{|F|}^{(1)})^T, \ldots, (x_1^{(|D|)}, \ldots, x_{|F|}^{(|D|)})^T$

Task:

- **learn** function $h\colon X_1 \times \cdots \times X_{|F|} \to C$ (classifier) from training set S s.t.
- for each unlabeled data point $(x_1^{(j)}, \ldots, x_{|F|}^{(j)})^T$ in D
- h assigns to it a class c_i in C, viz., $h((x_1^{(j)}, \ldots, x_{|F|}^{(j)})^T) = c_i$

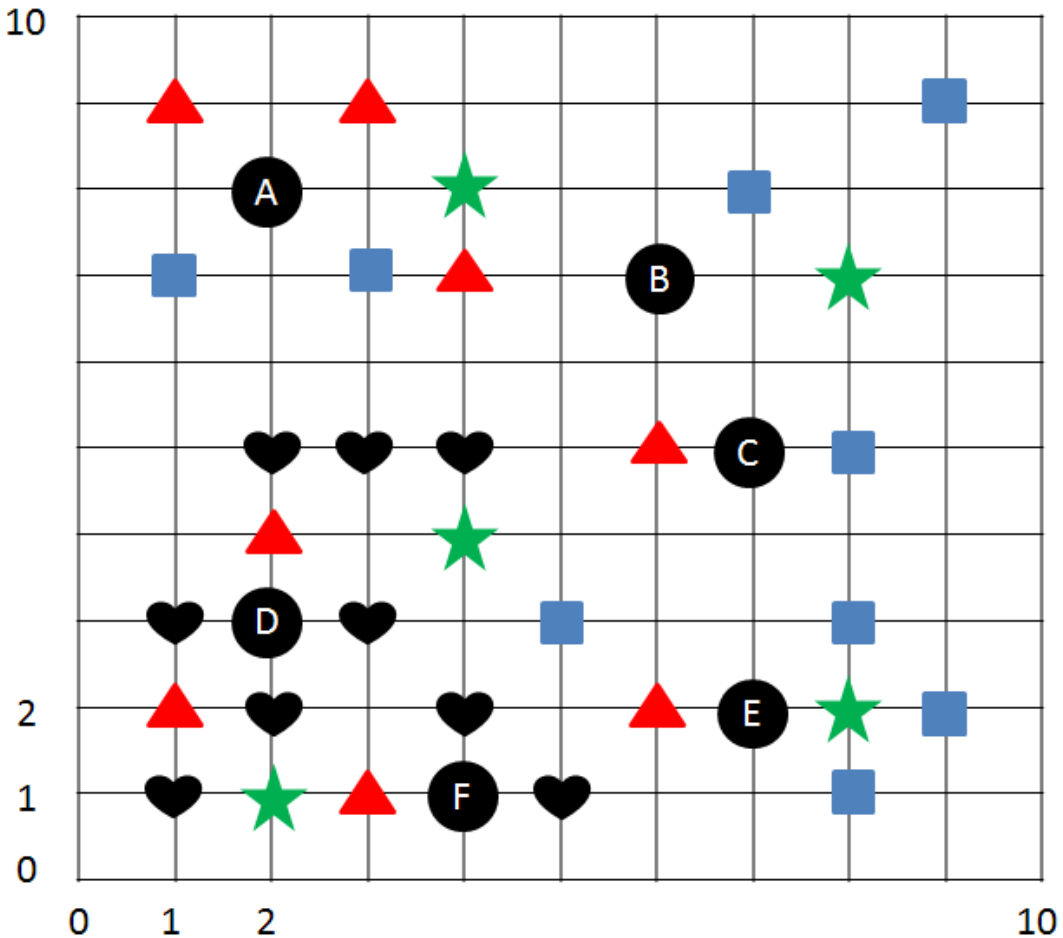
Exercise 05 - Question 2

K-Nearest Neighbors



Exercise 05 - Question 2.1

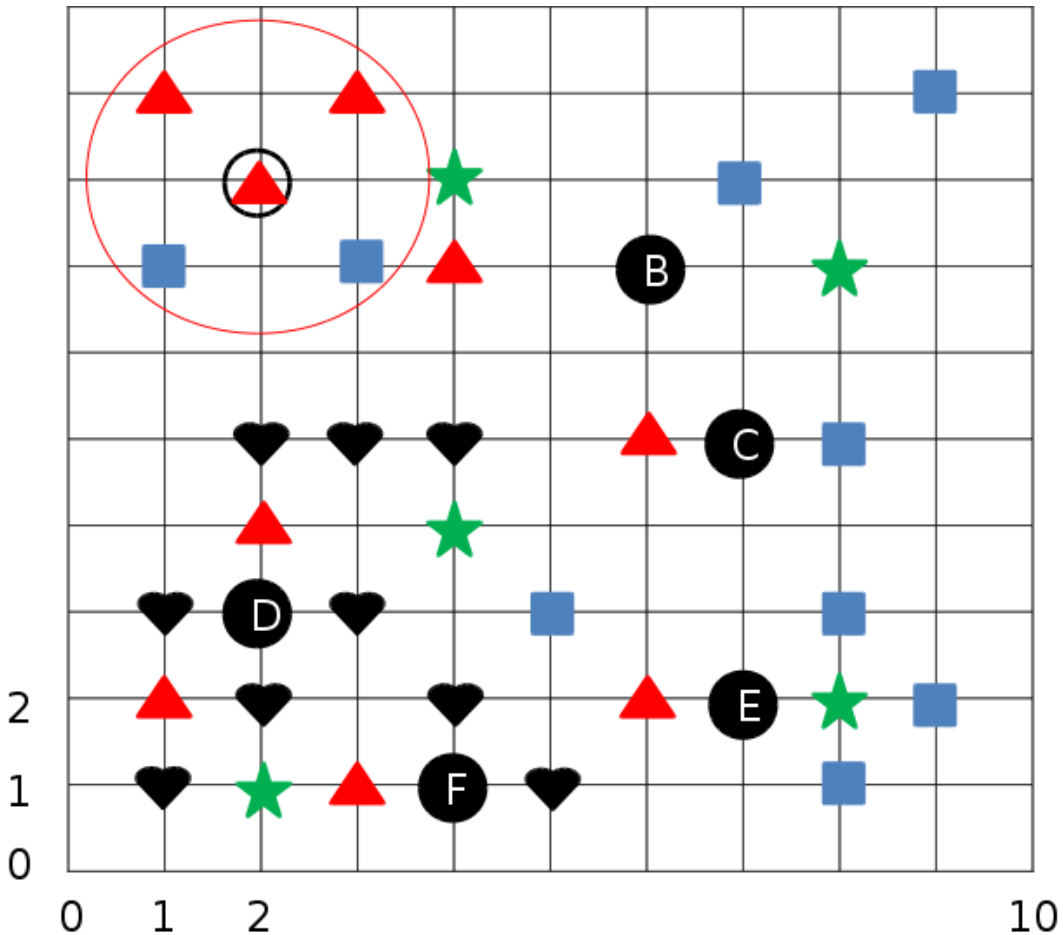
K-Nearest Neighbors



A = ?
B = ?
C = ?
D = ?
E = ?
F = ?

Exercise 05 - Question 2.1

K-Nearest Neighbors



A = *red triangle/blue rectangle*
B = *?*
C = *?*
D = *?*
E = *?*
F = *?*

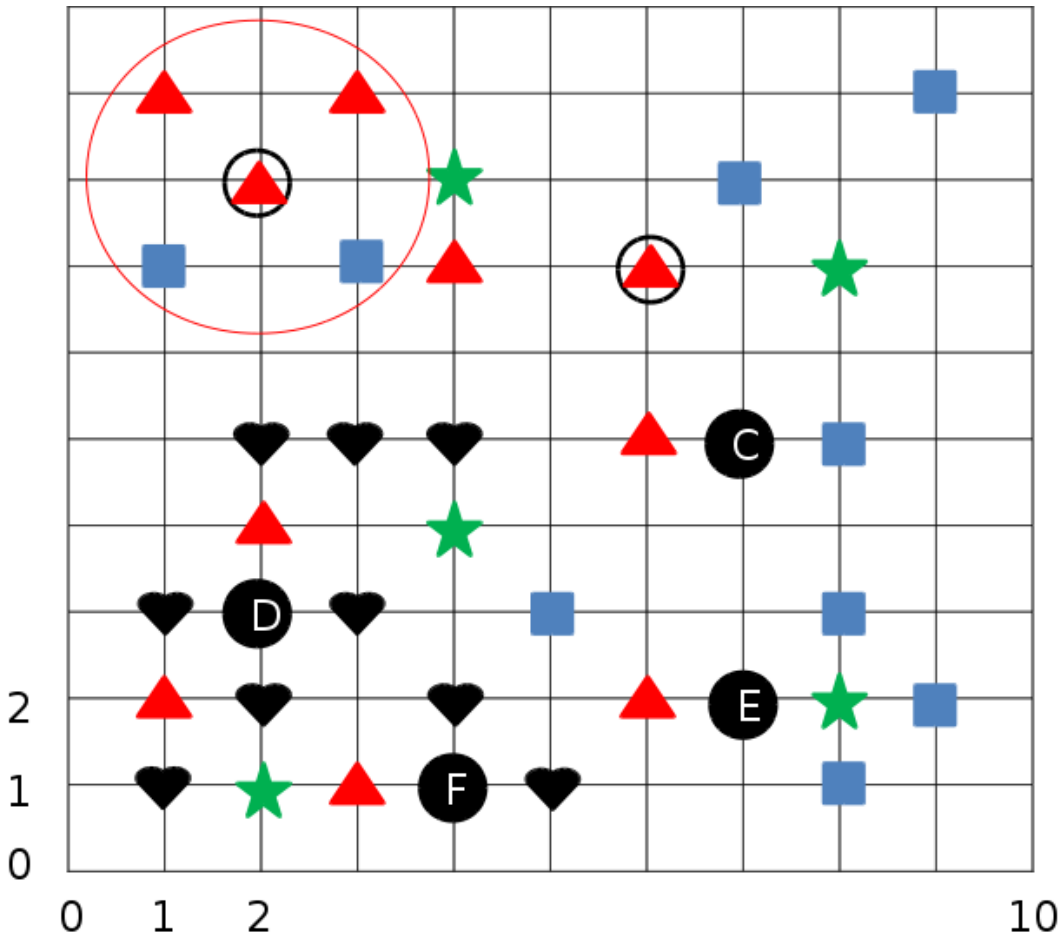
Exercise 05 - Question 2.1

K-Nearest Neighbors - Special case: a tie

If the class label count of the KNN results in a tie, a viable strategy is using K-1 nearest neighbours. If this still results in a tie, run K-2 NN and so on until K=1 which makes sure the point is assigned to a class. For equidistant points: flip a coin. If you have data where this is often the case, you should think about another algorithm.

Exercise 05 - Question 2.1

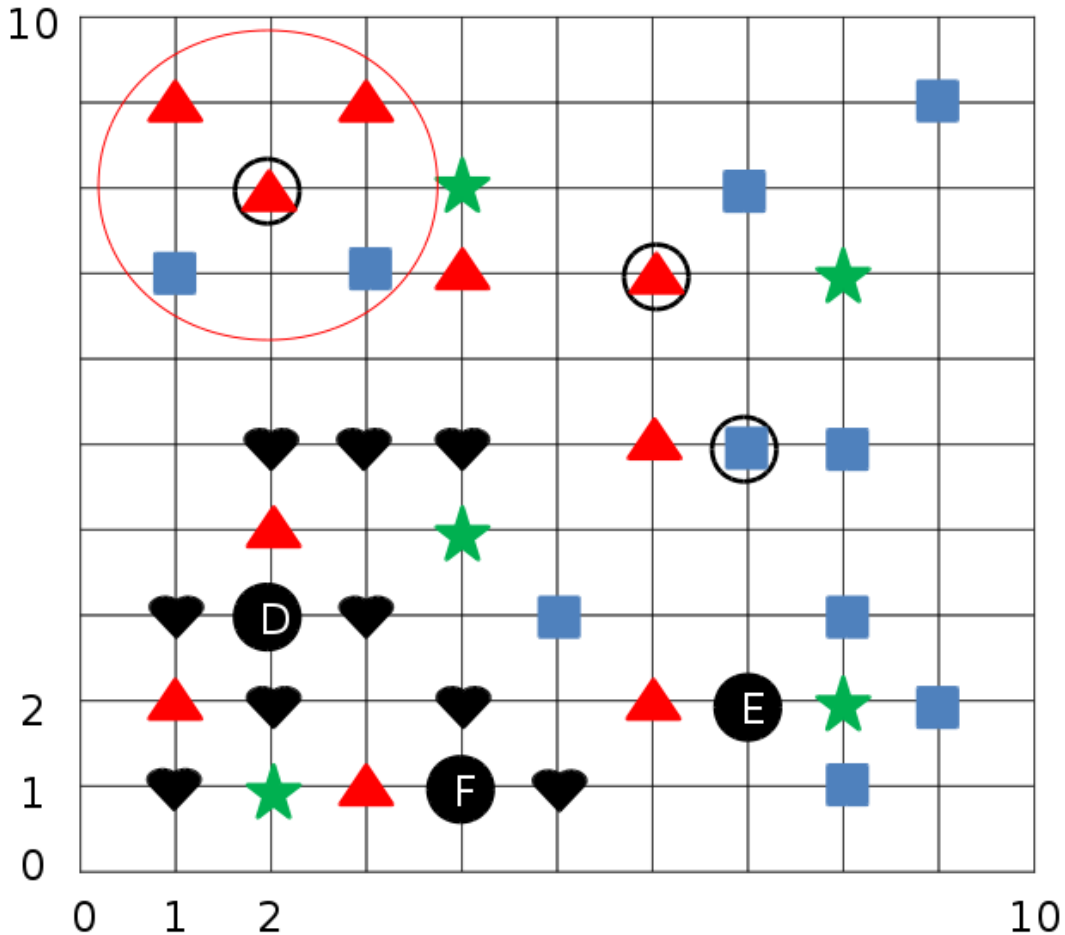
K-Nearest Neighbors



A = *red triangle/blue rectangle*
B = *red triangle*
C = *?*
D = *?*
E = *?*
F = *?*

Exercise 05 - Question 2.1

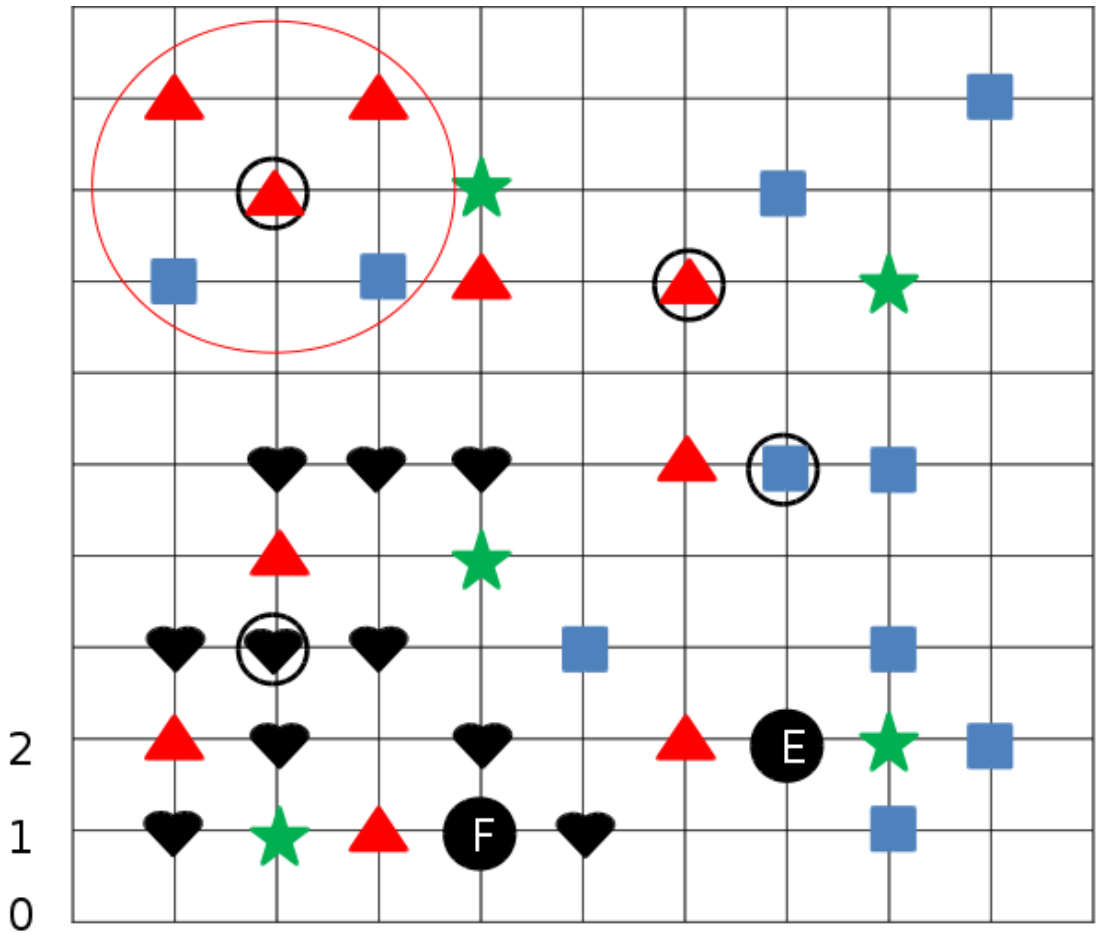
K-Nearest Neighbors



A = *red triangle/blue rectangle*
B = *red triangle*
C = *blue rectangle*
D = *?*
E = *?*
F = *?*

Exercise 05 - Question 2.1

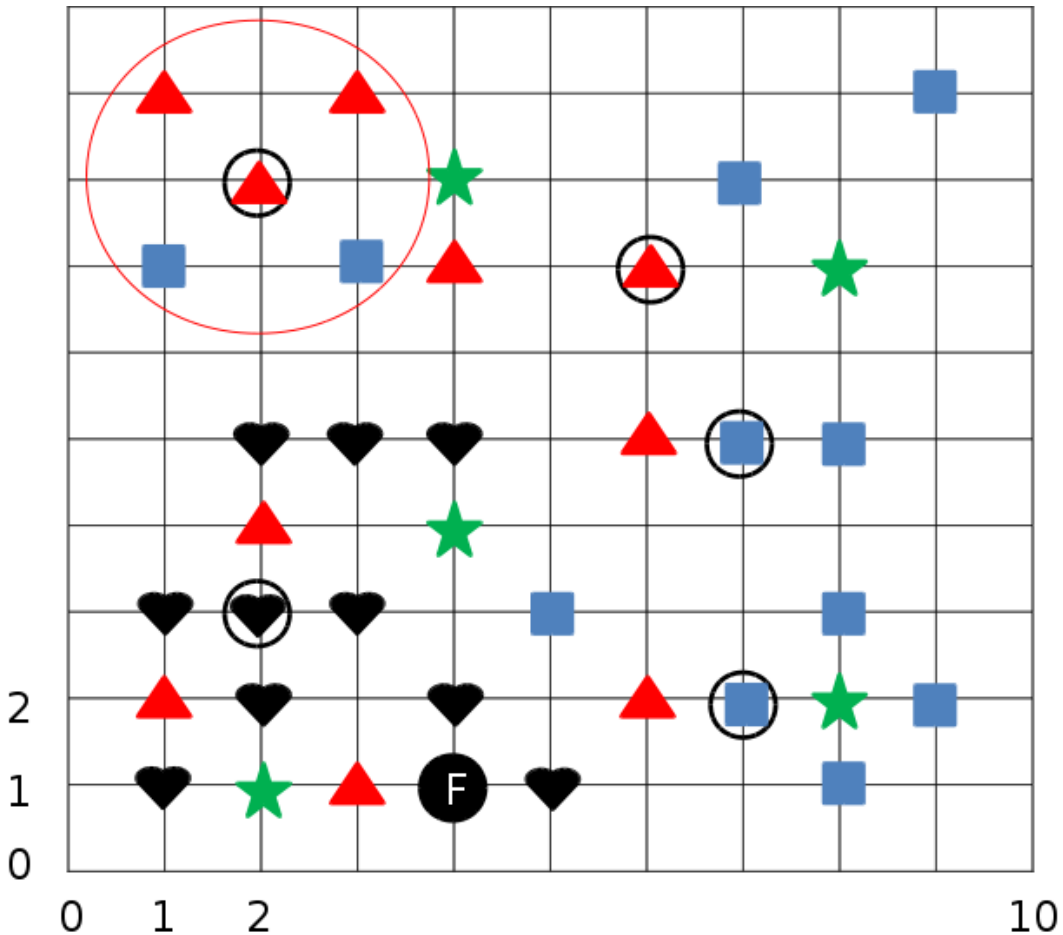
K-Nearest Neighbors



A = *red triangle/blue rectangle*
B = *red triangle*
C = *blue rectangle*
D = *black heart*
E = *?*
F = *?*

Exercise 05 - Question 2.1

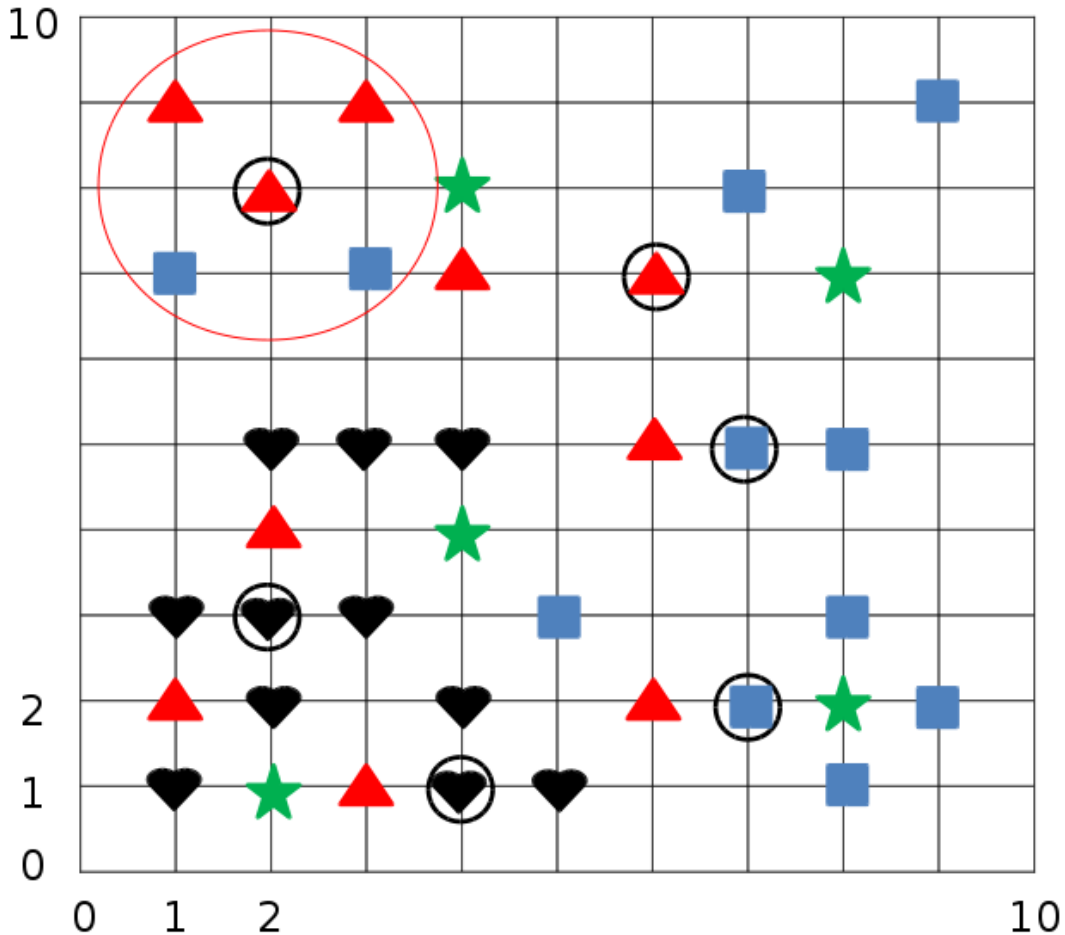
K-Nearest Neighbors



A = *red triangle/blue rectangle*
B = *red triangle*
C = *blue rectangle*
D = *black heart*
E = *blue rectangle*
F = *?*

Exercise 05 - Question 2.1

K-Nearest Neighbors



A = *red triangle/blue rectangle*
B = *red triangle*
C = *blue rectangle*
D = *black heart*
E = *blue rectangle*
F = *black heart*

Exercise 05 - Question 2.2a

K-Nearest Neighbors

The Green Star

Can you find a point in the data that would be classified as belonging to the green star class? Why?

Exercise 05 - Question 2.2a

K-Nearest Neighbors

The Green Star

Can you find a point in the data that would be classified as belonging to the green star class? Why?

Point $(6, 8)$ is can be classified unambiguously as a green star using K=4. Other points on the grid cannot be assigned to the green star class unambiguously, it is spread too far over the matrix.

Exercise 05 - Question 2.2b

K-Nearest Neighbors

The Green star

How would your answer in a) change for different values of k?

Exercise 05 - Question 2.2b

K-Nearest Neighbors

The Green star

How would your answer in a) change for different values of k?

To unambiguously classify more points as belonging to the green star class k must be 1 which adds a lot of noise to the clustering. Even with changing the values of k the problem does not disappear. KNN cannot deal with wide spread classes.

Exercise 05 - Question 3.1

Decision Trees

A Health Status Classifier

Build a decision tree using **Information Gain**

#	Heart Rate	Blood Pressure	Class
1	irregular	normal	Ill
2	regular	normal	Healthy
3	irregular	abnormal	Ill
4	irregular	normal	Ill
5	regular	normal	Healthy
6	regular	abnormal	Ill
7	regular	normal	Healthy
8	regular	normal	Healthy

Table : Medical Training data

Exercise 05 - Question 3.1

Decision Trees (Information Gain)

Rank features by **information gain**:

$$InfGain(S, X) = Entr(S) - \underbrace{\left[\sum_{x \in Values(X)} \left(\frac{|S_x|}{|S|} \times Entr(S_x)\right)\right]}_{= Entr(S|X)}$$

where:

- $S_x = \{\vec{x} \in S \mid X(\vec{x}) = x\}$: examples $\vec{x}$ where feature X has value x
- $Entr(S) = -\sum\{p(c_i) \times \log_2 p(c_i) \mid 1 \leq i \leq |C|\}$

Note: Information gain measures the **loss** in information (in terms of *Shannon entropy*) in the sample S obtained when **removing** or observing feature X: the bigger the loss, the better a predictor the feature is!

Exercise 05 - Question 3.1

Decision Trees

Now: $C = \{\text{Ill}, \text{Healthy}\}$, $F = \{\text{HeartRate}, \text{BloodPressure}\}$

$$\Rightarrow Entr(S) = -\left[\left(\frac{4}{8} \times \log_2 \frac{4}{8}\right) + \left(\frac{4}{8} \times \log_2 \frac{4}{8}\right)\right] = 1$$

$$\Rightarrow InfGain(S, X) = 1 - \left[\sum_{x \in Values(X)} \frac{|S_x|}{|S|} \times Entr(S_x)\right], \quad \text{for all } X \in F$$

Exercise 05 - Question 3.1

Decision Trees

#	Heart Rate	Blood Pressure	Class
1	irregular	normal	Ill
2	regular	normal	Healthy
3	irregular	abnormal	Ill
4	irregular	normal	Ill
5	regular	normal	Healthy
6	regular	abnormal	Ill
7	regular	normal	Healthy
8	regular	normal	Healthy

Heart Rate

$$Entr(h_{regular}) = \left(\frac{1}{5} \times \log_2 \frac{1}{5}\right) + \left(\frac{4}{5} \times \log_2 \frac{4}{5}\right) = 0.722$$

$$Entr(h_{irregular}) = \left(\frac{3}{3} \times \log_2 \frac{3}{3}\right) + 0 = 0 + 0 = 0$$

$$InfGain(h) = Entr(S) - \left[\left(\frac{5}{8} \times Entr(h_{regular}\right) + \left(\frac{3}{8} \times Entr(h_{irregular})\right)\right] =$$

$$1 - \left[\left(\frac{5}{8} \times 0.722\right) + \left(\frac{3}{8} \times 0\right)\right] = 1 - 0.45 = 0.55$$

Exercise 05 - Question 3.1

Decision Trees

#	Heart Rate	Blood Pressure	Class
1	irregular	normal	Ill
2	regular	normal	Healthy
3	irregular	abnormal	Ill
4	irregular	normal	Ill
5	regular	normal	Healthy
6	regular	abnormal	Ill
7	regular	normal	Healthy
8	regular	normal	Healthy

Blood Pressure

$$Entr(bp_{normal}) = \left(\frac{2}{6} \times \log_2 \frac{2}{6}\right) + \left(\frac{4}{6} \times \log_2 \frac{4}{6}\right) = 0.918$$

$$Entr(bp_{abnormal}) = \left(\frac{2}{2} \times \log_2 \frac{2}{2}\right) + 0 = 0$$

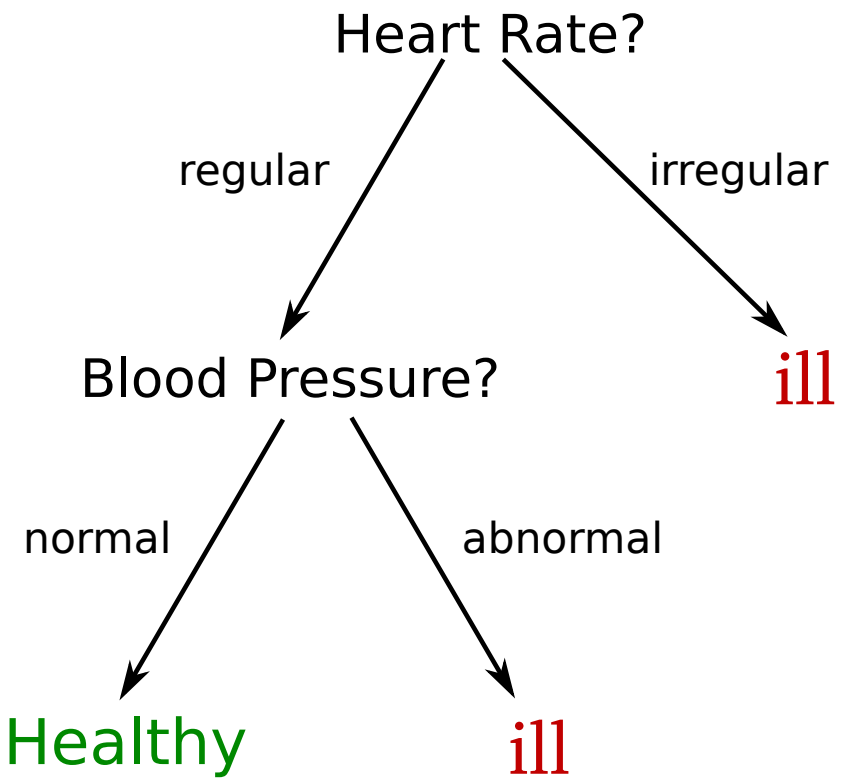
$$InfGain(bp) = Entr(S) - \left[\left(\frac{6}{8} \times Entr(bp_{normal})\right) + \left(\frac{2}{8} \times Entr(bp_{abnormal})\right)\right] =$$

$$1 - 0.69 = 0.31$$

Exercise 05 - Question 3.1

Decision Trees

$InfGain(h) = 0.55$
$InfGain(bp) = 0.31$



Exercise 05 - Question 3.2

Decision Trees

Deep vs. Broad Trees

Should a decision tree be rather deep or broad? Explain why!

Exercise 05 - Question 3.2

Decision Trees

Deep vs. Broad Trees

Should a decision tree be rather deep or broad? Explain why!

Usually classification should be fast (e.g. mark emails as spam). The deeper a tree is the more decisions have to be made to come to a conclusion. So broad trees usually result in faster search.
Deeper trees may have a better precision, so if this is more important than speed a decision tree should be deep.

Exercise 05 - Question 3.3

Decision Trees

Attribute Value Range

Imagine that the blood pressure would be stored in absolute values. How would the decision tree look like if we use the same strategy to build it as before? An informal description is sufficient.

Exercise 05 - Question 3.3

Decision Trees

Attribute Value Range

Imagine that the blood pressure would be stored in absolute values. How would the decision tree look like if we use the same strategy to build it as before? An informal description is sufficient.

Every numeric entry would be considered as a feature on its own because there is no abstraction over the values which results in very deep decision trees.
However there are decision tree algorithms that are able to deal with numeric attributes. They test the values for $< or >$.

Exercise 05 - Question 4.1 (Rapidminer - Bonus)

Learning a decision tree using ID3

Find the ID3 operator and apply it to the data in order to learn a decision tree. Set the criterion to information gain. Report the result in text format (click on Text View in the resulting tree view).

Exercise 05 - Question 4.1 (Rapidminer - Bonus)

Learning a decision tree using ID3

Find the ID3 operator and apply it to the data in order to learn a decision tree. Set the criterion to information gain. Report the result in text format (click on Text View in the resulting tree view).

```
hair = blonde
|   lotion = no: sunburned {sunburned=2, none=0}
|   lotion = yes: none {sunburned=0, none=2}
hair = brown: none {sunburned=0, none=3}
hair = red: sunburned {sunburned=1, none=0}
```

Exercise 05 - Question 4.2a (Rapidminer - Bonus)

Cross Validation

Explain how n-fold cross validation works and why it is needed to evaluate the performance of a model.

Exercise 05 - Question 4.2a (Rapidminer - Bonus)

Cross Validation

Explain how n-fold cross validation works and why it is needed to evaluate the performance of a model.

k-fold cross-validation splits the data set D into k mutually exclusive subsets $D_1, D_2, ..., D_k$.

- Train and test the learning algorithm k times
- Each time it is trained on $D \setminus D_i$ and tested on D_i

Benefits: Can be applied multiple times on one dataset.

Exercise 05 - Question 4.2b (Rapidminer - Bonus)

Cross Validation

Per default RapidMiner's cross validation uses *stratified sampling* rather than linear sampling. Explain what stratified sampling is and why it is generally safer to use it than linear sampling.

Exercise 05 - Question 4.2b (Rapidminer - Bonus)

Cross Validation

Per default RapidMiner's cross validation uses *stratified sampling* rather than linear sampling. Explain what stratified sampling is and why it is generally safer to use it than linear sampling.

Sampling fraction in each of the strata $D_1, D_2, ..., D_k$ (subsets of training data) is proportional to the one of the whole data set D

- Within each stratum, all examples have the same chance of being included in the sample.
- Across the strata, examples generally do not have the same probability of being included in the sample

Exercise 05 - Question 4.2c (Rapidminer - Bonus)

Cross Validation

RapidMiner comes with example data sets, one of them is the Iris data set containing descriptions of flowers with length and width of the sepals and petals in centimetres.
Apply k nearest neighbor classification (*k*-NN in Lazy Modeling) on the Iris data set and perform 10-fold cross validation (*X-validation*). Report the accuracy of the results and show the confusion matrix.

Exercise 05 - Question 4.2c (Rapidminer - Bonus)

Cross Validation

RapidMiner comes with example data sets, one of them is the Iris data set containing descriptions of flowers with length and width of the sepals and petals in centimetres.
Apply k nearest neighbor classification (*k*-NN in Lazy Modeling) on the Iris data set and perform 10-fold cross validation (*X-validation*). Report the accuracy of the results and show the confusion matrix.

Accuracy: 96.00% +/- 3.27% (mikro: 96.00%)

	Iris-setosa:	Iris-versicolor	Iris-virginica
Iris-setosa:	50	0	0
Iris-versicolor	0	47	3
Iris-virginica	0	3	47

Thank You!