Exam IN3050 Spring 2021 with solutions

1 Search

1 Search

Name 2 differences between *running 20 different hillclimbing algorithms with different starting points* and *an Evolutionary Algorithm with a population of 20*. Assume we are working on a binary optimization problem (trying to find the best combination of 1's and 0's in a string of length N), and the EA applies:

-One-point crossover

-A mutation rate of 0.1

-Fitness-proportionate selection for parent selection

-Tournament selection for survivor selection

These EA settings may not all be of relevance to your answer, but you may mention it if any of them are relevant to the difference.

The two algorithms have the same starting and termination conditions. Explain any other assumptions you have to make about the two algorithms. Explain/justify how each of the two differences affect the search.

Maximum marks: 6

Students get 3 points per reasonable difference (max 6 points). Could include things like (the top 2 are the ones that are perhaps most obvious):

-In the EA, parts of solutions can be combined during crossover, whereas in hillclimbing, each solution is isolated from the others. This can make the EA explore more as it can make large jumps in the fitness landscape.

-The EA can move its focus from some kinds of solutions to others, since all compete for survival. However, the hillclimber always keeps all the same "individuals", simply climbing all their respective peaks in the fitness landscape. In other words, EA can shift computing resources over to a more promising part of the fitness landscape.

-The EA may be searching for "neighbor" solutions a bit more randomly: It always just flips at random around 10% of the bits. The hillclimber may (depending on the assumptions made) test out neighbors in a more organized way – and thus "exploit" more while the EA "explores" more.

Other well-reasoned differences may be accepted, but these are probably the most likely ones. Partial points may be given for partially correct answers.

2 Evolutionary Algorithms

2

Individual number	Genotype	Fitness
1	0 1 1 1 1	1
2	0 0 0 0 1	2
3	1 0 0 1 1	3

(a) Selection in Evolutionary Algorithms - Fitness Proportional

Consider the individuals listed in the table on the left, from an evolutionary algorithm. Assume this is the full population, and you are about to do parent selection from it. Show your calculations and reasoning for the problems that follow.

What is the selection probability of each individual given *Fitness Proportional selection*? Note that the probability you calculate *is for a single selection*, not the probability that the individual gets selected into the pool of parents after N different selections.

Maximum marks: 2

In the problems below, if the students have correctly found the numbers and showed reasonable steps of their calculations, they get a full score. Only numbers with no reasoning/calculation gives half score. Incomplete calculations can give intermediate scores.

a) In fitness proportionate selection, the selection probability is equal to the individual's part of the total fitness value. The total is here 6, so individual 1 has a selection probability of 1/6, individual 2 2/6 (or 1/3) and individual 3 3/6 (or ½).

(b) Selection in Evolutionary Algorithms - Tournament Selection

What is the selection probability of each individual given Tournament Selection? Assume we perform deterministic tournaments of size 2 selecting individuals without replacement.

Maximum marks: 4

b) In tournament selection, k individuals compete against each other for survival. The probability of selecting any individual into the tournament is the same for all individuals: 2/3. Since we select individuals without replacement, we can never select the same individual twice for a single tournament. Since the tournament is deterministic, the strongest individual always wins.

The easiest way to calculate probabilities here is to do it one individual at the time. Individual 1 has the lowest fitness, and since tournaments are deterministic and without replacement, it can never win the tournament. Its selection probability is therefore 0.

Individual 2 is more fit than individual 1, and less fit than individual 3. There are 3 different tournaments that can occur here: 1vs2, 1vs3 and 2vs3. All are equally likely. Only the first would be won by individual 2, making its selection probability 1/3.

Following a similar reasoning, the selection probability for individual 3 is 2/3.

(c) Selection in Evolutionary Algorithms - Fitness Sharing

Assume we want to do fitness sharing to reduce the potential for premature convergence. We will use a simple distance measure that simply counts the number of genes in two genotypes that are in the same location but have different values:

$$d(g1, g2) = \sum_{i=1}^{5} (abs(g1[i] - g2[i])$$

where g1 and g2 are the two genotypes being compared, g1[i] denotes the i'th gene of g1, and abs() indicates the absolute difference between the two elements. We use a niche size σ_{share} of 3, and set α=1. What is the modified fitness value of each individual f'(i) assuming the original fitness value was the one you calculated with Fitness Proportional selection? (6p)

Maximum marks: 6

Note: There were two small errors in the task description here – the following correction was posted on the course website:

1) The value **alpha (=1) does not belong in the formula**. You can ignore that value

2) The problem refers to the fitness value you calculated in a. That is incorrect, what is meant **is the fitness value that is listed in the table on the left**. In a, you actually calculated selection probabilities, not fitness values.

Suggested solution:

Due to the errors mentioned above, if a student has used the alpha-value instead of sigma below, they should not get point reduction. Also, if students have used the selection probability from a instead of the fitness value in the formula, points should not be deducted.

The students here need to use the formula for fitness sharing from the book or slides:

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh(d(i,j))} \qquad sh(d) = \begin{cases} 1 - d/\sigma & d \le \sigma \\ 0 & otherwise \end{cases}$$

The d in the formula is the distance measured as given in the problem statement. The distance between individuals is calculated as follows (it is simply a count of all genotype positions that are different):

Dist(1,2) = 3

Dist(2,3) = 2

Dist(1,3) = 3

Finally, the distance from any individual to itself is 0.

The sum over sh(d(i,j)) sums up how much fitness individual i should share with each other individual j, according to the formulas on the left. We can calculate the sharing-component of each pair as follows (all are closer than or equal to than the niche size σshare):

Sh(d(1,2)) = 1-d(1,2)/ σshare = 1-3/3 = 0

Sh(d(2,3)) = 1-(2/3) = 1/3

Sh(d(1,3)) = 1-(3/3) = 0

Finally, the sharing component each individual has to itself is

Sh(d(X,X)) = 1-0/3 = 1

We can now calculate the adjusted fitness value as follows:

f'(1) = f(1) / (sh(d(1,1))+sh(d(1,2))+sh(d(1,3)))= 1/(1+0+0) = 1

f'(2) = f(2) / (sh(d(2,1))+sh(d(2,2))+sh(d(2,3)))= 2/(0+1+(1/3)) = 2/(4/3) = 1.5

f'(3) = f(3) / (sh(d3,1))+sh(d(3,2))+sh(d(3,3)))= 3/(0+(1/3)+1) = 3/(4/3) = 2.25

If the students forgot that each individual shares fitness with itself, but were otherwise correct, they can be awarded 4 points. They will then have a problem calculating the final values, as there will be a zero under the divider in the final formula. Mistakes caused by wrong initial f(i) values from problem a) should not give a reduced score here.

3 Features

3

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0	no	text	none
4	no	2,454	61	0	no	text	small
5	no	41,623	1088	9	no	html	small
...							

The table shows the first entries from a data set of e-mails. The data set will be used for recognizing spam mails. In addition to the label, 'spam'/'no spam', 6 attributes are extracted from each e-mail. The attribute 'format' has two possible values: 'text' and 'html', the attribute 'numbers' has three possible values, 'small', 'big', and 'none', while ' "winner" occurs' can take the values 'yes' or 'no'.

(a) Categorical and numerical features

Explain in 2-4 sentences what is meant by categorical features and numerical features in machine learning. You may use the data set as example.

Maximum marks: 3

a) In machine learning an item is described by a number of features. A numerical feature takes a numerical value, in the example: chars, line breaks, " 'dollar' occurs numbers". For each categorical feature, there is a finite set of categories, and the feature picks one of these values. In the example, "'winner' occurs", format, and number are categorical features.

(b) Classifiers and features

We have considered several kinds of classifiers including

- Decision Trees
- *k*NN
- Perceptron
- Logistic regression
- Feed-forward neural networks (Multi-layer perceptrons)

Some of them use categorical features while others use numerical features. Which of them can handle categorical features without further preprocessing the data, and which of them can handle numerical features without preprocessing? You do not have to justify your answers.

Maximum marks: 3

b) Decision trees is based on categorical features. The others are based on numerical features.

(c) Preprocessing categorical attributes

Suppose you will apply a classifier which expects numerical features to a data set where some attributes are categorical. How can this data set be preprocessed to fit the classifier? You may use the spam data set as example.

Maximum marks: 3

c) "'Winner' occurs" is a binary feature taking the values "no" and "yes". It can be replaced with a numerical feature taking the values 0 and 1. Similarly, for "format" which is also a binary feature.

"Number" takes three possible values. It can be replaced by three features such that

n1= 1, n2=0, n3=0 replaces number=small

n1= 0, n2=1, n3=0 replaces number=big

n1= 0, n2=0, n3=1 replaces number=none

In general, a categorical feature taking n possible values for $n>2$ can be replaced by n many different binary numerical features.

(d) Further preprocessing

Suppose you are to prepare the spam data set for a classifier taking numerical features. Are there other preprocessing steps you would take? Explain shortly which steps you would take and why.

Maximum marks: 3

d) We see that there is a large variation in magnitude between the numerical features. The replaced categorical features take absolute values 0 or 1, while chars can be as large as 41,623. We should scale all the features.

(e) Preprocessing numerical data

Suppose instead you will apply a classifier which expects categorical features to a data set where some attributes are numerical. How can this data set be preprocessed to fit the classifier? You may use the spam data set as example.

Maximum marks: 3

e) To turn the numerical data into categorical data, we may put them into bins. We could, e.g., replace 'chars' with a categorical feature with three values

chars-many = few for chars<100

chars-many = medium for 1000≤chars<10,000

chars-many = many for chars≥10,000

We could alternatively make a more coarse-grained or fine-grained division.

(Not asked for: With decision trees, we can do the binning dynamically during training choosing the bin boundaries that are optimal)

4 Perceptron

4(a) Classifying

We will use a perceptron for classification. There are two classes, 0 and 1. The data points have two features, e.g., the point $\mathrm{x}_1 = (\mathrm{x}_{1,1}, \mathrm{x}_{1,2}) = (1, 1)$. We add a bias term $\mathrm{x}_{j,0} = -1$ to each data point x_j. Assume the current weights are $\mathrm{w} = (\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2) = (0.1, 0.1, 0.2)$. How will the perceptron classify x_1?

Maximum marks: 4

a) After adding bias, we get $x_1' = (-1, 1, 1)$

Then $z = \boldsymbol{w} \cdot \boldsymbol{x}_1' = (0.1, 0.1, 0.2) \cdot (-1, 1, 1) = 0.2 > 0$ and the predicted class is $y = 1$

4(b) Training

Assume that the correct class for $\mathrm{x}_1 = (1, 1)$ is $\mathrm{t}_1 = 0$. We are training the perceptron sequentially. How will the weights $\mathrm{w} = (\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2) = (0.1, 0.1, 0.2)$ be updated from observing $(\mathrm{x}_1, \mathrm{t}_1)$? Assume a learning rate of 0.2.

Maximum marks: 4

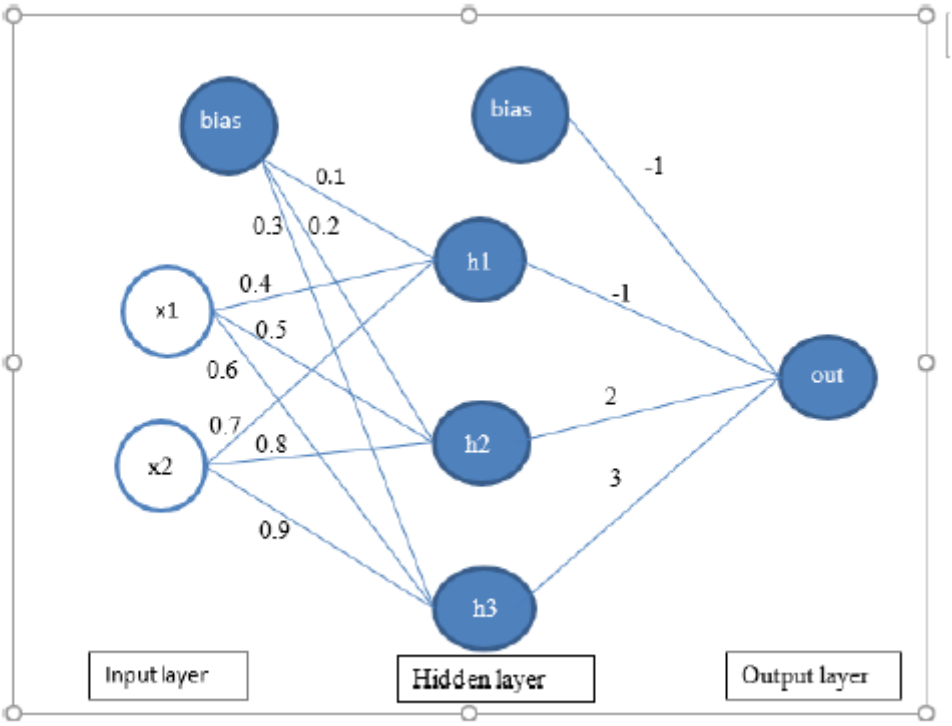
b) Assume a learning rate $\eta = 0.2$

Then $\boldsymbol{w} = \boldsymbol{w} - \eta(y - t)\boldsymbol{x}_1 =$

$$(0.1, 0.1, 0.2) - 0.2(1 - 0)(-1,1,1) = (0.1, 0.1, 0.2) - (-0.2, 0.2, 0.2) = (0.3, -0.1, 0)$$

5 Neural Networks and Backpropagation

5



The neural network.

(a) Value 1

Consider the network in the figure. Assume that the bias terms are set to -1 for both layers. Moreover, we assume that the logistic function (sigmoid) is used as activation in the hidden layer, and that the learning rate is 0.1. When the input is $(x_1, x_2) = (2, -1)$, what is the output from the hidden node h3?

Maximum marks: 4

a) $\boldsymbol{x}_1 = (-1, 2, -1)$

Weighted sum into h3: $h_3 = (0.3, 0.6, 0.9) \cdot (-1, 2, -1) = 0$

Output from h3: $a_3 = \frac{1}{1+\exp(-h3)} = \frac{1}{1+\exp(0)} = 0.5$

(b) Value 2

Assume that the network is used for regression with no activation function in the output layer. What is the final output when the input is $(x_1, x_2) = (2, -1)$?

Maximum marks: 4

b) We see similarly that $h_1 = h_2 = 0$ and $a_1 = a_2 = 0.5$

and the final output is $y = z = (-1, -1, 2, 3) \cdot (-1, 0.5, 0.5, 0.5) = 3$

(c) Learning

Assume that the correct prediction for $(x_1, x_2) = (2, -1)$ is $t = 5$. We are training the network sequentially. What will the updated weight from the hidden node h3 to the out-node be after backpropagation?

Maximum marks: 5

c) We assume 0.5 Squared Error as loss function

Then: $w_{3,out} = w_{3,out} - \eta(y - t)(a_3) = 3 - 0.1(3 - 5)(0.5) = 3.1$

(d) Backpropagation

What will the updated weight from the x2-node to the hidden node h3 be after backpropagation?

Maximum marks: 5

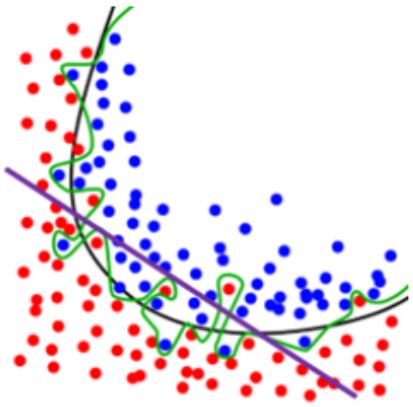
d) We must calculate the delta-term at the node h3. Use the w from the figure before the update

Delta-term at h3: $(y - t)(w_{3,1})a_3(1 - a_3) = (3 - 5)3 * 0.5(1 - 0.5) = -1.5$

Hence $v_{2,3} = v_{2,3} - \eta\, \delta(h3)x_2 = 0.9 - 0.1(-1.5)(-1) = 0.9 - 0.15 = 0.75$

6 Overfitting and Bias

6



Figure

(a) Overfitting

What do we mean by overfitting in machine learning and why can it be a problem? You may refer to the figure for an example.

Maximum marks: 3

a) When training a ML classifier or regressor we use a training set. This set is a subset of a more general population. The goal of the learned system is to perform well on the population as large. Even if we manage to draw the training set randomly from the population the distribution of feature

values will not reflect the distribution of feature values in the population. Overfitting means that the system had learned these peculiarities of the training set too well. An overfitted system may perform much better on the training set than on the rest of the population.

(b) kNN

Suppose your classifier is using a *k*NN algorithm, and that is has problems with overfitting. Which adjustments would you propose to the classifier to overcome some of the effect?

Maximum marks: 3

b) For the kNN algorithm, a smaller *k* value increases the chance of overfitting, while a large *k* will in general result in a smoother decision boundary and reduce the chance of overfitting. In the figure, assuming that this is the training set, the black (?) curve might be the best decision boundary for the population, while the green line is the (overfitted) decision boundary of kNN for *k*=1.

(c) Inductive bias

The percetron algorithm and logistic regression have a similar inductive bias. What is it? Use it to describe what is meant by inductive bias.

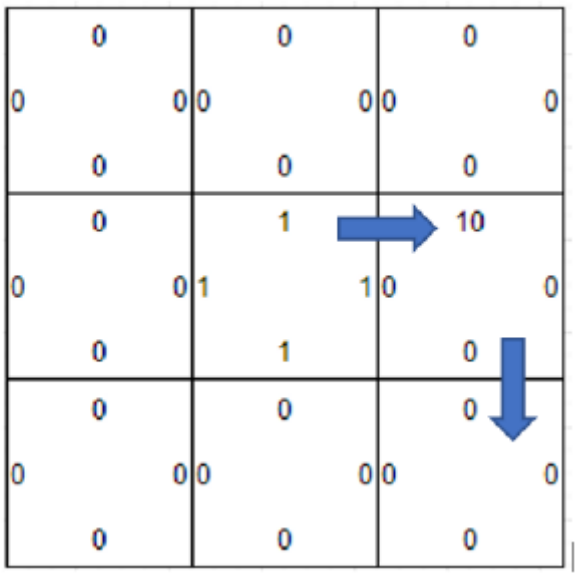
Maximum marks: 3

c) Both the percpetron algorithm and logistic regression classifier are linear classifiers. They will yield linear decision boundaries, like the purple line in the figure with two features, a plane with three features, and in general a hyper-plane. They are not able to find decision boundaries like the black curve or the green line in the figure.

In general, a learning algorithm in machine learning will carry certain assumptions regarding what the trained model should look like. Without assumptions, an algorithm would not be able to generalize from a finite set of observations.

7 Reinforcement Learning

7



(a) Reinforcement Learning - Q learning

Consider the Reinforcement Learning grid world on the left. Each square is a state, and numbers represent Q-values associated with moving to the neighbor state in the given direction. As normal, the Q-values are estimates, that we want to update by exploring the environment.

We initialize an agent in the center state, and let it explore according to its policy. In this world, **every move from one state to another yields a reward of 1**. We want to train the agent with a **learning rate μ=0.1** and **discount factor** γ**=0.1**.

Using its policy, the agent performs first the action "right", then the action "down", as shown in the figure.

Calculate the updated Q-value (using the formula from the lecture slides or text book) for the center state, that is Q(center state, right), as indicated by the top arrow. You should calculate the updated Q-value according to the (off-policy) Q-learning algorithm.

Maximum marks: 4

a) Q-learning works according to the following formula:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

The old value of the given state is Q(s,a) = 1. The learning rate and discount factor are 0.1. The reward of the action is 1. The estimate of optimal future value in q-learning is equal to the maximum achievable q-value currently available in the new state. That is 10 (for the action of going upwards). The new q-value is thus:

Q = 1 + 0.1*(1+0.1*10-1) = 1.1

Partially correct answers may give partial scores, the full calculation and correct answer gives 4 points.

(b) RL - SARSA

Assuming we again start from the same initial state and Q-values, as in the previous task, and perform the same actions, calculate the updated Q-value for state-action pair (center, right) according to the (on-policy) SARSA algorithm.

Like before, every move from one state to another yields a reward of 1. We again want to train the agent with a **learning rate μ=0.1** and **discount factor** γ**=0.1**.

Maximum marks: 4

b) SARSA follows the same formula as Q-learning with one important exception: Rather than assuming we will make the optimal action in the next state (state t+1) we assume we take the action that we actually did take. In the new state, the optimal action is "up" with Q-value 10. The action we did take was "down" with Q-value 0. The q-value in this case is thus:

Q= 1+ 0.1*(1+0.1*0-1) =1

Partially correct answers may give partial scores, the full calculation and correct answer gives 4 points.

(c) Reinforcement Learning Q vs SARSA

With reference to the two previous tasks, explain why the updated Q-values from Q-learning and SARSA are different: Why does on-policy and off-policy learning give different results here?

Maximum marks: 4

c) **A full score should contain more or less the following reasoning (repeated from the previous question):** SARSA follows the same formula as Q-learning with one important exception: Rather than assuming we will make the optimal action in the next state (state t+1) we assume we take the action that we actually did take. In the new state, the optimal action is "up" with Q-value 10. The action we did take was "down" with Q-value 0.

Another way to state this, which would also give a full score: On-policy learning learns the Q-value associated with acting according to our own policy, which is different than what (off-policy) Q-learning learns, which is the Q-value associated with always following a greedy policy.

(d) Reinforcement Learning - Policy

Consider again the grid world on the left. Assume the agent is now placed in the rightmost state in the center row (the state where the downwards-pointing arrow begins).

With the Q-values currently filled in (not considering the updates you made in the previous tasks), what is the likelihood of choosing the action "down" in the state on the right, given you are using the following policies: 1) A greedy policy? 2) An epsilon-greedy policy with epsilon equal to 0.1? 3) A soft-max policy with a temperature of 1? Show your calculations and/or justify your answers.

Maximum marks: 5

d) **Full scores require correct answers and justifications:**

1) Zero, since a greedy policy always chooses the action associated with the highest q-value **(1p)**

2) With epsilon-greedy and epsilon equal to 0.1, we will have a 10% chance to choose the action at random rather than greedily. Since there are 4 available actions, the final probability for choosing down is thus (0.1*0.25) = 0.025. **The answer 0.1 here can give half a point.(2p)**

3) The softmax function is:

$$P\left(Q_{s,t}(a)\right) = \frac{e^{(Q_{s,t}(a)/\tau)}}{\sum_b e^{(Q_{s,t}(b)/\tau)}}$$

With all q-values but Q(up) being 0, we get:

e^{10} = 22026.4657948. e^0 = 1.

1/(1+1+1+22026.4657948) = 0.000045

Small calculation errors are accepted if the use of the formula is correct. (2p)

(Not needed for full score, but perhaps interesting to know: The probability is very close to 0, because softmax (unlike epsilon greedy) is affected by the fact that one action has a much higher q-value than the others.)

8 Unsupervised Learning

8(a) Unsupervised Learning - Making Data

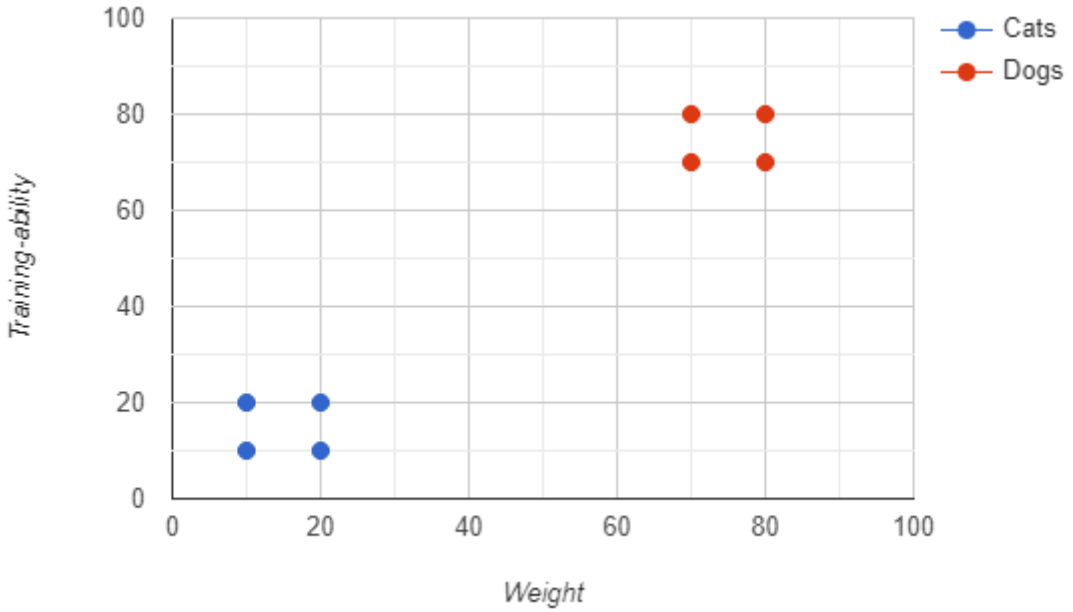
Assume you have a collection of data on two different animals, cats and dogs, measuring 1) their weight (between 0 and 100 kg), and 2) their train-ability (how easy it is to train the animals – ranging from 0, impossible to train to 100, very easy to train). You want to try to use a K-means classifier to discriminate between cats and dogs based on weight and trainability.

You will here make up some data on the cats and dogs, and in the following question use this data to explain how K-means clustering can proceed to divide your data into two parts.
The data you make up should be submitted as drawings (plots with x- and y-axes). It is not necessary to make up the exact numbers/coordinates for your datapoints - hand-drawn plots with approximate data positions are enough. You should include at least 4 datapoints representing cats and 4 datapoints representing dogs.

a) Make data (including values for both features and the correct label) that k-means can perfectly divide into 2 clusters, separating cats and dogs. (2p)
b) Make a new set of data that k-means *can not* perfectly divide into two clusters, separating cats and dogs. (2p)

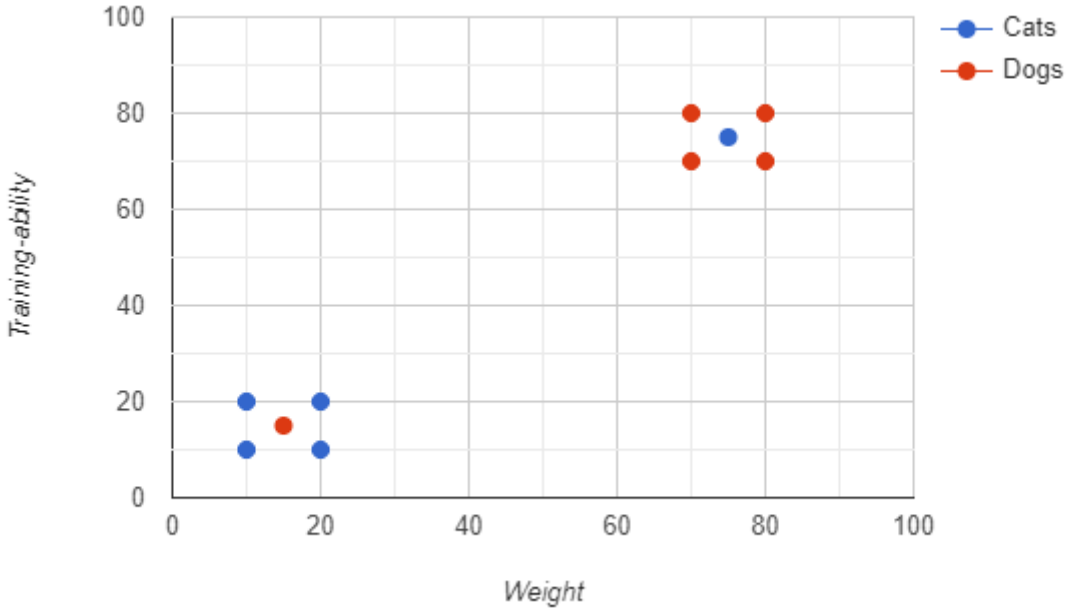
Maximum marks: 4

a) **Students are awarded full scores if they have 1) made a dataset with cats and dogs that can be separated into two clusters (2p) and 2) a dataset that k-means clustering will not cluster correctly (2p).** For 1) it is important that there is no cat-data mixed in with the dog-data or vice versa, as that would not give a perfect clustering. In other words, no cat data should be closer to the center of the dog cluster than the dog data that is furthest from that center (and vice versa). It is also important that the student has provided at least 4 datapoints of each type and shows the class of each datapoint. An example is shown below: Note the cat data and dog data are clearly separated, so clustering will be easy.



For 2), the most likely way to generate data that k-means clustering will not cluster correctly, is to make some cat data that finds itself separated from other cat data by some dog data (or vice versa). That way, k-means clustering cannot correctly divide the data into two clusters. **Other ways that make k-means fail may be possible and should be awarded points if correct.**

An example is shown below: The cat data mixed in with dog data, and opposite, will make it impossible for k-means clustering to perfectly cluster this dataset.



8(b) Unsupervised Learning - Forming clusters

Referring to the cat and dog data you made up (the data you think *can* be separated), describe how the k-means algorithm forms clusters. You do not have to show all calculations, but explain how (and why) the cluster centers move around, and how (and why) the clustering changes as you iterate.

Maximum marks: 2

b) **Students are awarded full scores if they have explained 1) how cluster centers move in k-means clustering (1 p) and 2) explained how the clustering (assignment) is updated in k-means clustering (1p).**

Minimal answer for part 1: Cluster centers are updated in each iteration by placing them in the mean (or center) of all points currently assigned to this cluster.

Minimal answer for part 2: Clustering/assignment of points to a cluster is updated in each iteration by assigning each data point to the cluster center that is currently closest to it.

8(c) Unsupervised Learning - Why

Referring to the two datasets you made up about cats and dogs, why did k-means succeed on one dataset, but not the other?

Maximum marks: 3

c) **The answer should correspond to the type of problem the student illustrated in a.**

The most likely problem is that some cat data that finds itself separated from other cat data by some dog data (or vice versa). That way, k-means clustering cannot correctly divide the data into two clusters.

If the problem is the one suggested above, the minimal answer should describe that k-means clustering depends on distances from cluster centers, and if a cat finds itself closer to the “dog center” than a dog does, then that cat will be clustered together with the dogs.

If the student has correctly illustrated another problem with clustering the cats and dogs, and correctly described here why it happened, they should receive a full score.

9 L-systems

9 L-systems

Bracketed L-Systems are useful when visualizing the resulting string of an L-System. Consider the following string resulting from a bracketed L-System: F[+F][-F].

Assume 'F' is interpreted as a draw command. The symbols '+' and '-' are interpreted as 'turn' being either a positive or negative turn of 45 degrees respectively. The brackets '[' and ']' are used for storing and retrieving the position and orientation (the state) when drawing the L-System. Here the state is effectively pushed and popped on a stack.
Can you draw the resulting tree from this string? Indicate the position where you started drawing by adding an S to the starting point. (A positive turn command assumes rotation in a counter-clockwise direction)

You should deliver three separate drawings below:

a) The resulting tree after the first draw command (2p)

b) The resulting tree after the first and second draw command (2p)

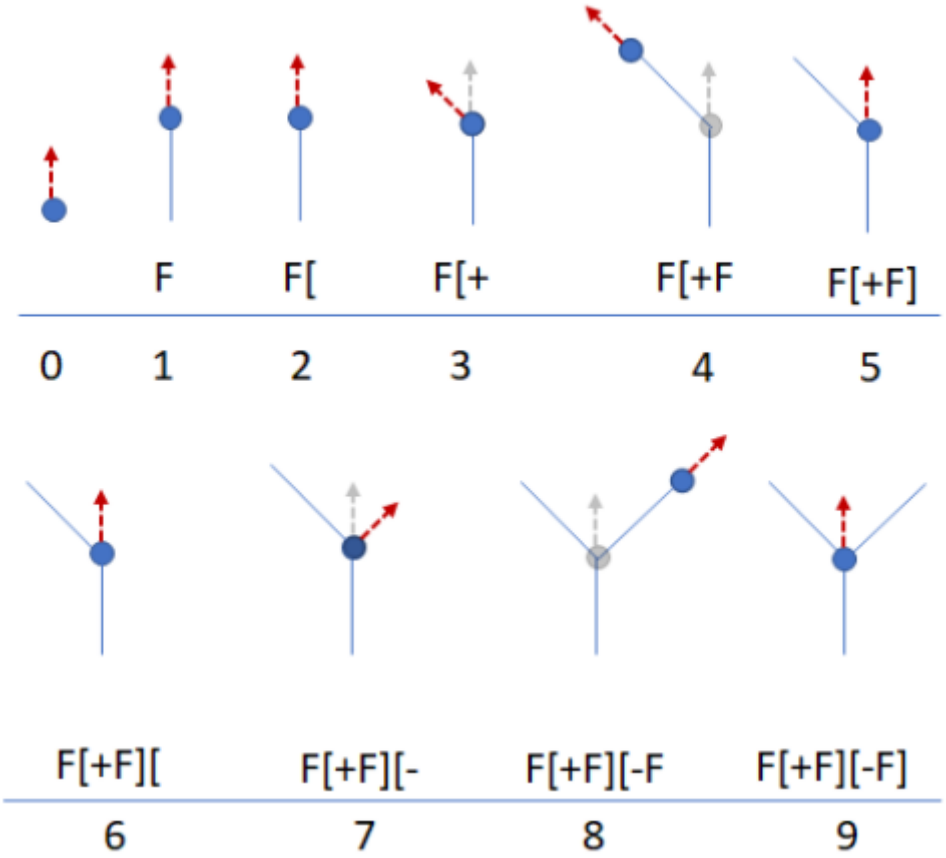
c) The final drawing (2p)

Maximum marks: 6

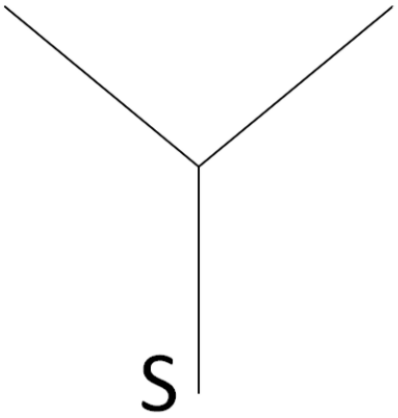
Following the rules for visualizing bracketed L-systems, the commands following the string F[+F][-F] will be:

1. draw
2. push
3. turn positive
4. draw
5. pop
6. push
7. turn negative
8. draw
9. pop

The steps are illustrated below.



Students should illustrate the progress after each draw command (each F above). That means, answer a) should reflect the tree drawn after step 1, b) should reflect the tree after step 4 and c) the tree after step 9, as seen here with an S marking the starting position:



Note that students are not supposed to draw the circle and dotted arrow above, those are just for visualizing the drawing progress here.

Students should not get deducted points due to somewhat imprecise drawings, as their drawing tools may be a bit hard to use. If they have understood the rules and made a more or less correct tree, they should get a full score.

Students who have mixed up the meaning of + and – (and thus drawn the right part of the tree before the left) can also be awarded a full score.