2020-2021 Intro to ML Term 1 Exam Solution

This is NOT an official solution but revised by a lot of people together, credit to them as well! If you spot any mistakes or better solution, please contact me :)

1a. Below is the answer to the questions

i) Compute the initial entropy for the whole dataset.

According to the information entropy formula $H(x) = -\sum_k^K P(x_k)\log_2(P(x_k))$, the initial entropy for the whole dataset is thus $H = -0.5 \cdot \log_2 0.5 - 0.5 \cdot \log_2 0.5 = 1$

ii) Compute the information gains for selecting the colour attribute and for selecting the softness attribute, each with respect to the initial entropy of the dataset.

We first calculate the entropy of selecting the colour and softness attribute. But even before that, we first calculate the number of sweet and sour fruit in each subset

	Sour	Sweet	
Green	70	0	70
Red	20	60	80
Yellow	10	40	50
	100	100	200

	Sour	Sweet	
Hard	100	25	125
Soft	0	75	75
	100	100	200

For the colour attribute, there are three subsets, whose weighted entropy is

$$\begin{aligned} H_{colour} &= \frac{80}{200}H_{red} + \frac{70}{200}H_{green} + \frac{50}{200}H_{yellow} \\ &= \frac{80}{200}\cdot(-\frac{60}{80}\cdot\log_2\frac{60}{80} - \frac{20}{80}\cdot\log_2\frac{20}{80}) + \frac{70}{200}\cdot 0 + \frac{50}{200}\cdot(-\frac{40}{50}\cdot\log_2\frac{40}{50} - \frac{10}{50}\cdot\log_2\frac{10}{50}) \\ &= -0.4\cdot(0.75\cdot\log_2 0.75 + 0.25\log_2 0.25) - 0.25\cdot(0.8\cdot\log_2 0.8 + 0.2\cdot\log_2 0.2) \\ &= 0.504993 \end{aligned}$$

For the softness attribute, there are two subsets, whose weighted entropy is

$$\begin{aligned} H_{softness} &= \frac{125}{200}H_{soft} + \frac{75}{200}H_{hard} \\ &= 0.625\cdot(-\frac{100}{125}\log_2\frac{100}{125} - \frac{25}{125}\log_2\frac{25}{125}) + 0.375\cdot 0 \\ &= 0.451205 \end{aligned}$$

The information gain for selecting colour is thus
$IG = H_{dataset} - H_{colour} = 1 - 0.504993 = 0.495007$

The information gain for selecting softness is thus
$IG = H_{dataset} - H_{softness} = 1 - 0.451205 = 0.548795$

iii) Based on your calculations in ii), which attribute should be selected as the root node of a decision tree classifier?

We select the rule that maximise the information gain; thus in this case softness should be selected as the root node of the decision tree

1b. Below is the answer to the questions

i) given a dataset X with five points {2, 4, 6, 8, 9}, which of these two GMMs fits this dataset better?

For Gaussian Mixture Models, we use the negative log likelihood as the metric to measure the model performance. The negative log likelihood formula is $L = -\sum_i^N \ln p(x_i|\theta)$.

We first calculate the pdf of those two models.

$$p_1(x|\theta) = 0.6 \cdot \mathcal{N}(x|3,1) + 0.4 \cdot \mathcal{N}(x|8,0.5)$$
$$p_2(x|\theta) = 0.4 \cdot \mathcal{N}(x|2,0.5) + 0.3 \cdot \mathcal{N}(x|5,1) + 0.3 \cdot \mathcal{N}(x|8,1)$$

Then we plug in the five values while looking up the values of $\mathcal{N}(x|\mu,\sigma)$ in the table

$$\begin{aligned} L_1 &= -\sum_i^5 \ln p_1(x|\theta) \\ &= -\ln(0.6 \cdot 0.242 + 0.4 \cdot 0) - \ln(0.6 \cdot 0.242 + 0.4 \cdot 0) - \ln(0.6 \cdot 0.004 + 0.4 \cdot 0) \\ &\quad - \ln(0.6 \cdot 0 + 0.4 \cdot 0.798) - \ln(0.6 \cdot 0 + 0.4 \cdot 0.108) \\ &= 14.1754 \end{aligned}$$

$$\begin{aligned} L_2 &= -\sum_i^5 \ln p_2(x|\theta) \\ &= -\ln(0.4 \cdot 0.798 + 0.3 \cdot 0.004 + 0.3 \cdot 0) - \ln(0.4 \cdot 0 + 0.3 \cdot 0.242 + 0.3 \cdot 0) \\ &\quad - \ln(0.4 \cdot 0 + 0.3 \cdot 0.242 + 0.3 \cdot 0.054) - \ln(0.4 \cdot 0 + 0.3 \cdot 0.004 + 0.3 \cdot 0.399) \\ &\quad - \ln(0.4 \cdot 0 + 0.3 \cdot 0 + 0.3 \cdot 0.242) \\ &= 10.9179 \end{aligned}$$

From the calculation above, model 2 fits better to the data since it has a lower negative log likelihood value, which indicates a higher likelihood that the five data points could be observed (i.e. likelihood means the chance of observing the given data using the model with the given parameters)

ii) Increasing the number of K of a GMM will increase the model complexity. It will first decrease the value of Bayesian Information Criterion and then at some point increase it, which we want to minimize. If K is very large, i.e. K = # of data points, it will overfit the training dataset (likelihood on training dataset is 1) and not generalize well for test data.

2a. Below is the answer to the questions

i) Using the given network, calculate the predicted output probability that the word “exciting” has positive sentiment

Using the given vector and weights in the network, we calculate the final output

$$
\begin{aligned}
\begin{bmatrix}\hat{y}_1\\ \hat{y}_2\end{bmatrix} &= \text{softmax}(\begin{bmatrix}0.5 & -1\\ -0.9 & 0.5\end{bmatrix}\tanh(\begin{bmatrix}0.6 & 1.0\\ -0.9 & -0.4\end{bmatrix}\begin{bmatrix}-1\\ 2\end{bmatrix}))\\
&= \text{softmax}(\begin{bmatrix}0.5 & -1\\ -0.9 & 0.5\end{bmatrix}\tanh(\begin{bmatrix}1.4\\ 0.1\end{bmatrix}))\\
&= \text{softmax}(\begin{bmatrix}0.5 & -1\\ -0.9 & 0.5\end{bmatrix}\begin{bmatrix}0.88535\\ 0.09966\end{bmatrix})\\
&= \text{softmax}(\begin{bmatrix}0.34301\\ -0.74698\end{bmatrix})\\
&= \begin{bmatrix}0.74838\\ 0.25162\end{bmatrix}
\end{aligned}
$$

The probability that the word “exciting” has positive sentiment is roughly 25.16%

ii) Perform one gradient descent step to update the vector representation for the word “exciting”, using the given network. The correct target sentiment of the word is positive. Use learning rate 0.5.

In order to perform one gradient descent step, we need to calculate $\frac{\partial Loss}{\partial X}$ and then use the learning rate of 0.5 to update the input as $X = X - 0.5 \cdot \frac{\partial Loss}{\partial X}$

According to the chain rule, the partial derivative $\frac{\partial Loss}{\partial X}$ is calculated as follows

$$
\begin{aligned}
\frac{\partial Loss}{\partial X} &= \frac{\partial Loss}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial X}\\
&= \frac{1}{N}(\hat{y} - y) \cdot \frac{\partial Z_2}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial X}\\
&= (\frac{1}{N}(\hat{y} - y) \cdot W_2^T) \circ (1 - (\tanh(Z_1))^2)^T \cdot W_1^T\\
&= ((\begin{bmatrix}0.74838\\ 0.25162\end{bmatrix} - \begin{bmatrix}0\\ 1\end{bmatrix})^T \cdot \begin{bmatrix}0.5 & -1\\ -0.9 & 0.5\end{bmatrix}) \circ (1 - (\tanh(\begin{bmatrix}1.4\\ 0.1\end{bmatrix}))^2)^T \cdot \begin{bmatrix}0.6 & 1.0\\ -0.9 & -0.4\end{bmatrix}\\
&= \begin{bmatrix}1.04773\\ -1.12257\end{bmatrix} \circ (1 - (\tanh(\begin{bmatrix}1.4\\ 0.1\end{bmatrix}))^2)^T \cdot \begin{bmatrix}0.6 & 1.0\\ -0.9 & -0.4\end{bmatrix}\\
&= \begin{bmatrix}1.04773 \cdot (1 - (\tanh 1.4)^2) & -1.12257 \cdot (1 - (\tanh 0.1)^2)\end{bmatrix} \cdot \begin{bmatrix}0.6 & 1.0\\ -0.9 & -0.4\end{bmatrix}\\
&= \begin{bmatrix}1.04773 \cdot 0.21615 & -1.12257 \cdot 0.99007\end{bmatrix} \begin{bmatrix}0.6 & 1.0\\ -0.9 & -0.4\end{bmatrix}\\
&= \begin{bmatrix}0.22647 & -1.11142\end{bmatrix} \begin{bmatrix}0.6 & 1.0\\ -0.9 & -0.4\end{bmatrix}\\
&= \begin{bmatrix}1.13616 & 0.67104\end{bmatrix}
\end{aligned}
$$

Notice that in the calculation above, because the correct target sentiment of the word is positive (e.g. the golden truth is positive, which translates to $y = \begin{bmatrix}0\\ 1\end{bmatrix}$), we just used $y = \begin{bmatrix}0\\ 1\end{bmatrix}$

Then we update the input

$$
X_{new} = X - 0.5\frac{\partial Loss}{\partial X} = \begin{bmatrix}-1\\ 2\end{bmatrix} - 0.5 \cdot \begin{bmatrix}1.13616\\ 0.67104\end{bmatrix} = \begin{bmatrix}-1.56808\\ 1.66448\end{bmatrix}
$$

2b. Below is the answer to the questions

Describe the neural network models and experiments you would design for the given scenarios. Specify the following details: 1) the number of input neurons, 2) the number of output neurons, 3) output activation function, 4) loss function for optimisation, 5) main evaluation metric.

i) We need to automatically determine whether two photos depict the same building. The system needs to work with any buildings, not only one specific building. You are able to use a pre-trained image encoder which provides 256-dimensional vector representations of each photo.

Since we have access to a pre-trained image encoder which provides 2 * 256-dimensional vector representations of each image, the input layer simply has 256 neurons. We treat this problem as a classification problem where the output layer has two layers: one denotes yes and another denotes no. The output activation function can simply be softmax and the loss function is binary cross-entropy. For final evaluation, we can calculate prevision, recall, and then F1 score of the model to evaluate its performance.

Alternatively, the output layer can have only one single neuron and then we use tanh/sigmoid as the activation function of the output layer.

ii) We need to predict the number of cars and the number of bicycles that cross a particular intersection in a day. For each day, we are able to use the corresponding information from the previous 7 days to make the prediction.

Since we have the information from the previous 7 days, the input layer will have 2 * 7 neurons. Since this problem is clearly a regression task and relates to two output variables (car and bicycle), the output neuron will have 2 neurons representing car and bicycle count crossing that intersection. The output activation funcion simply be a linear function since the number of cars and bicycles are unbounded. For the loss function, we can use Mean Squared Error function to optimize/train the model. For the evaluation metric, we can use the Rooted Mean Squared Error on both outputs.

The expriment process of both tasks involves in the following steps: first randomly shuffle the dataset so that its internal ordering does not correlate with any attribute, and then divide the dataset into training, validation, and test set by the ratio of 8:1:1. Then we perform data preprocessing and normalization. Then we randomly initialize the weights in the network. Then train the neuron networks using the training set which is divided into mini-batches. We then perform cross-validation (if needed) with different neural network structures (e.g. number of layer, number of neurons in the hidden layers, optimzer, learning rate, etc.) and use the evaluation step to find out which hyper-parameter/model performs better. We then select the optimal model and use all the data (train + validation + test) to train the optimal again. (Final step is optional)

3. Below is the answer to the questions

a) Explain in one or two sentences why this algorithm is appropriate in this use case.

MAP-Elites is simple to implement yet is capable of finding high-performing and diverse solutions. In this case, we are trying to find a collection of drone configurations of maximum payload and maximum parcel size, while each producing the shorter delivery time, which fits MAP-Elites definition

b) Define the fitness function and the behavioural descriptors (including their discretisation) that you would use in this case. Use one or two sentences for each of them.

For the fitness function, we can use $\frac{1}{\text{Avg.DeliveryTime}}$ as the value. We can also define it as $-Avg.DeliveryTime$. I think turn it into negative value is generally better because it doesn't have the problem of dividing by zero.

We want configurations that produce shorter average delivery time to have a higher fitness score, thus we use 1 dvided by the average delivery time. When average delivery time becomes -1, we also need to set fitness value to a negative value to indicate it's not viable. For the behavioural descriptors, we need to discretize both two aspects (max payload, max parcel size, and avg. delivery time). For maximum payload, we can discretize by each 0.5kg; for maximum parcel size, we can discretize by each 2cm. Thus, the descriptor space has the size of $18 \cdot 20 = 360$. This design is to use as much search space as possible (400 max) of the behavioural descriptor while

(Of couse if we try to discretise the search space more carefully it will produce 400 possible categories. The above is just an illustration)

c) Give a suitable genotype, phenotype, function used to develop a genotype into a phenotype, and mutation operator, that you would use to solve the problem described above. Explain your answer in a few sentences (less than 15).

For the genotype, we can have a list of real numbers to represent the five attributes in the configuration. (e.g. one real number represents the width of the main frame, one real number for the length, and one for the number of motors, etc.) For phenotype, we can take any values within the valid range defined for each attribute. With the genotype described above, we can retain the real numbers (floating point values) for both the width and the length of the main frame, and then round the rest for all other integer attributes. We then develop a mutation operator: it will add Gaussian noise on the width and length of the main frame, while having a certain chance (say 10%) of adding/subtracting 1 for those integer traits. Notice that the process described above does not take care of values out of the defined range: if the real number is out of the defined range, then the average delivery time would be -1 and fitness function should return a big negative number to indicate it’s not feasible.

Other types of representations are possible as well.

d) Which algorithm seen in this course can you use to automatically construct this discretisation into 400 regions of similar size. Explain in a few sentence how this algorithm works.

Use K-means algorithm with K = 400, i.e. clustering those data points into 400 clusters/groups. Or alternatively, use GMM with 400 models in it, but it seems to be less efficient than K-means.

Here is the explaination for the K-means algorithm: randomly initialize the 400 centroids in the 5-dimensional feature space, then allocate each point according to the closest centroid to it. Next, update the position of the centroids based on the data points allocated to it. (e.g. calculate the mean value in all 5 dimensions) Repeat the two steps above and then stop when the centroid positions don’t move anymore (test for convergence).

December 2020 Exam Feedback – Introduction to Machine Learning

Q1

(a) (i) Most students scored full marks here as long as the answer is correct and they either showed their workings or explained their answer intuitively.

(ii) This question required an understanding of entropy/information gain as a measure of the "purity" of a dataset. While many students were able to answer this question well, many also struggled. The calculations are relatively straightforward as in the lectures once the student figures out the numbers to plug in. A lack of workings may cost the student one or two marks. A few students stopped at computing entropies; these would usually be valid, but unfortunately the question specifically asked for information gain, so some marks may be lost if only entropies were calculated.

(iii) Marks were awarded based on whether students chose the attribute with the maximum information gain from their answer in (ii), rather than whether the attribute itself is correct. Minimum entropy was also accepted.

(b) (i) The ideal answer would require students to select the model which minimizes the model's BIC or negative log likelihood (or maximizes the (log) likelihood). BIC would have been better in practice as the two models have a different number of components, but the examiner accepted both. The ideal base for log would have been log_e, but log_10 and log_2 were also accepted. Common mistakes:

1. Computing \sum_k^K log(pi_k * \sum_i^M N(x_i | theta_k)), rather than \sum_i^M log(\sum_k^K pi_k * N(x_i | theta_k)). Note where the summation across the 5 instances occur. You should perform the weighted sum across components for *one* instance, and then sum across all M instances.
2. Summing probabilities rather than multiplying them (you should either sum the LOG probabilities, or multiply probabilities)
3. Computing component responsibilities/using EM: these do not demonstrate model quality

Some students explained with diagrams - although none were convincing enough to score full marks for the question. Some students lost some marks for calculation errors - how much is deducted depends on the severity of the errors and how much workings were provided: many students who showed their workings were largely saved.

(ii) The examiner was lenient and accepted most reasonable answers. A model answer would include a discussion on the model's expressiveness/complexity and on overfitting. A discussion about bias-variance trade-off was also a common answer and was accepted assuming the student explained it correctly (does increasing K increase or decrease the bias? Some students got this wrong). It would have been ideal to be concrete about what it means by low/high bias/variance in the context of a GMM rather than in general, but the examiner accepted these for this exam. A discussion about the effects on the NLL/BIC was also generally accepted.

Q2

(a) Calculation of the forward pass was mostly performed correctly. Some students picked the value from the wrong output neuron when asked about the output probability. Also the columns and rows were sometimes mixed up when it came to arranging the weights into a matrix so that it can be used for calculations, which resulted in incorrect values.

The backpropagation part was more of a challenge. In order to complete it in reasonable time, it was important to perform the calculations in an efficient way. For example, using the joint derivative of cross-entropy and softmax given in the lectures, instead of starting to differentiate them by hand. Also, reusing the values from the first part whenever possible, such as for calculating the derivative of tanh. There were some common errors regarding not following the question - calculating updates for the weights instead of the input, or using MSE as the loss function instead of cross-entropy as was specified in the question. I awarded as many points as

possible for any partially completed answers. Any errors or omissions resulted in deductions but they did not affect points given for downstream calculations.

(b) Mostly solved correctly with a few common errors. When designing a binary classifier then multiple different output types were allowed. However, one combination that cannot work is a single output neuron with softmax activation - if you do the calculation, you'll see that this neuron would always give 1 as output.
Also, there seemed to be some confusion about input layer sizes. An input layer of size x means the network can take x real-valued inputs. If you want to have two 256-dimensional vectors as input, you need to have an input layer of size 512.

Q3

(a) Most students scored full marks here, as long as the answer contained the main reasons for using a Quality-Diversity algorithm.

(b) The vast majority of students scored full marks here as well. However, a common mistake was to search for a complex answer that was not using the information provided by the simulator. Another common mistake was to mix elements of the behavioural descriptor (the size/weight of the parcels) in the fitness function. MAP-Elites will naturally only compare solutions from the same type of solution (i.e., same parcel types). Therefore, including this in the fitness is not necessary.
Finally, two points needed to be addressed and were missing in some answers: 1) what to do when the delivery time was "-1" and 2) how to ensure that the collection can only contain up to 400 solutions.

(c) The challenge of this question was that we had to consider both discrete and continuous parameters in the genotype. While the ideal answer should have considered a genotype composed of floats, I also accepted solutions discretising the width and length parameters, or a mixture of both floats and bitstrings, as long as the overall answer was consistent.
The most common errors came from either missing elements in the answer (e.g., the function to develop the genotype into the phenotype) and inappropriate mutation operators or configurations. For instance, a gaussian mutation operator cannot be used on a bit-string. Also, when using a genotype with floats, which is then discretised for the discrete values in the phenotype, it is important to ensure that the mutation is always effective. This is not the case with a standard gaussian mutation, which if the sampled value is too small might not be enough to make a parameter move to the next range. To avoid this, a more advanced mutation operator leveraging the specificities of the problem could have been proposed.

(d) The difficult aspect of this question was to recognise that this was an unsupervised learning problem: Clustering a dataset into 400 clusters. Most of the students who recognise this also used the appropriate algorithm to be used in this case.