✗ COMP307 – Assignment 2
✗ Duong Tran
✗ 300589631

Part 1: Classifying Pingu with a Neural Network (50 Marks)

a)

```
First instance has label Adelie, which is [0] as an integer, and [1. 0. 0.] as a list of outputs.

Predicted label for the first instance is: ['Chinstrap']
```

b)

```
Weights after performing BP for first instance only:
Hidden layer weights:
 [[-0.28056275 -0.21970523]
 [ 0.07826717  0.20090767]
 [-0.30124328  0.32065123]
 [ 0.09932855  0.01035171]]
Output layer weights:
 [[-0.27633516  0.01659626  0.1982984 ]
 [ 0.09427539  0.11599737 -0.37222444]]
```

c)

```
After training:
Hidden layer weights:
 [[ 0.93328452 -9.81120147]
 [-7.28786875  5.20357946]
 [ 2.38840536 -1.40663748]
 [ 2.4705405   1.42934008]]
Output layer weights:
 [[ -9.67523157  -2.44440275   3.24170455]
 [  4.90940805  -2.87316161 -11.65020389]]
Accuracy in test set: =  0.8153846153846154
```

- With train-set accuracy after 100^{th} epoch is 0.8283582089552238, the test-set accuracy is roughly what I would expect. The difference between train-set accuracy and test-set accuracy is roughly equivalent so it proves my programs are not either overfit or underfit.

d)

- With a small dataset like in our case study, the program converges very quickly, it doesn't take too much time to run and predict the output.
- As I mentioned above, the accuracy between both sets is roughly equivalent so it proves that my programs are neither overfit nor underfit. A statistical model, or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data. Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have less data to build an accurate model and also when we try to build a linear model with less non-linear data. In this case study, the programs perform good on test data, it shows that we give enough data to the program and underfit did not occur. A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore, they can really build unrealistic models. If we see a trend that accuracy in train set is continuously increasing but in test set, the accuracy does not increase and in some cases, it even decreases, the chance that we have overfitted our program. In this program, this didn't happen when both test and train increase after each epoch. So, the program is surely not overfit.

1.1 Further Improving Your Network (15 marks)

-

```
After training:
Hidden layer weights:
 [[ -2.26420111 -10.88648616]
 [ -8.14815943   5.65280866]
 [  3.13224798  -1.29228614]
 [  3.9457565    1.95532896]]
Output layer weights:
 [[ -6.17249613  -3.97649849   4.29907339]
 [  4.73362019  -3.63421374 -11.27601845]]
Accuracy in test set: =  1.0
```

- As we can see from above, with biases, the accuracy in test set has been increased from roughly 81% to 100%. It shows that the biases definitely help improve the programs.
- That's the reason why we need bias neurons in neural networks. Without these spare bias weights, our model has quite limited "movement" while searching through solution space. To gain more flexibility we need to get back to the original model with bias. It will equip us with weight w_0, not tied to any input. This weight allows the model to move up and down if it's needed to fit the data.

1.2 Sensitivity Testing (10 bonus marks)

- I will choose learning rate and number of epochs to do sensitivity testing on, and I will do sensitivity testing on a program without biases
- ***Learning rate:***
  - Learning rate, generally represented by the symbol 'α', is a hyper-parameter used to control the rate at which an algorithm updates the parameter estimates or learns the values of the parameters.
  - Learning rate is used to scale the magnitude of parameter updates during gradient descent. The choice of the value for learning rate can impact two things: 1) how fast the algorithm learns and 2) whether the cost function is minimized or not.
  - Large learning rate may cause oscillating behaviors and small learning rate may cause slow convergence.
  - Generally, 0.2 is a good starting point in practice so I chose to start with learning rate = 0.2, and it performs well. The accuracy is decent, the value of learning is not too big that it creates unnecessary oscillating behaviors but also not too small. With a massive learning rate like 10, the effect we are instantly visible is the accuracy decreases and same goes to too small learning rate like 0.0001, the accuracy decreases enormously to 20%. It proves that the value of learning rate

is quite sensitive, either too big or too small will cause unwanted effects to our program.

```
test_learning_rate(10)
test_learning_rate(0.2)
test_learning_rate(0.001)
test_learning_rate(0.0001)
test_learning_rate(0.0000001)
```

```
Accuracy in test set: =  0.6461538461538462
Accuracy in test set: =  0.8
Accuracy in test set: =  0.4461538461538462
Accuracy in test set: =  0.2
Accuracy in test set: =  0.2
```

- ***Epoch:***
    - Epochs are defined as the total number of iterations for training the machine learning model with all the training data in one cycle. In the Epoch, all training data was used exactly once. Further, in other words, Epoch can also be understood as the total number of passes an algorithm has completed around the training dataset. A forward and a backward pass together counted as one pass in training.
    - The number of iterations for each epoch is different due to the fact it is related to batch size, the bigger the batch size, the smaller number of iterations will be for each epoch.
    - Batch size is defined as the total number of training examples that exist in a single batch. You can understand batch with the above-mentioned example also, where we have divided the entire training dataset/examples into different batches or sets or parts. For example, batch size = 100, it means 100 training examples that exist in a single batch.
    - With a smaller number of epochs, we are likely to see an underfitting happening in our data, especially in large datasets. Because the training examples have not been trained enough in order to provide an accurate prediction. In contrast, if the number of epochs is too big, it may cause overfit because we overtrain the training set.
    - We can see below that with one epoch only, the accuracy is significantly low, and it slowly increases with more epochs for have. In this case study, a big number of epochs didn't cause overfit, but it is not a suggestion that we should keep a big number of epochs.
    - It proves that the value of epoch is quite sensitive, either too big or too small will cause unwanted effects to our program.

```
test_epochs(1)
test_epochs(10)
test_epochs(100)
test_epochs(500)
test_epochs(10000)
```

```
Accuracy in test set: =  0.35384615384615387
Accuracy in test set: =  0.8
Accuracy in test set: =  0.8153846153846154
Accuracy in test set: =  0.8923076923076924
Accuracy in test set: =  0.9076923076923077
```

Part 2: Genetic Programming for Symbolic Regression (35 marks)

a)

- Terminal set will be the value of X in the regression.txt

b)

- For function set:

```
pset = gp.PrimitiveSet("MAIN", 1)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(math.cos, 1)
pset.addPrimitive(math.sin, 1)
pset.addEphemeralConstant("rand101", lambda: random.randint(-1,1))
pset.renameArguments(ARG0='x')
```

c)

- Firstly, I transform the tree expression into a function, I will need that function later on when I want to calculate y value. I set a new variable called SSD and loop through the terminal set. Inside the loop for each value of terminal set, I provide a prediction of y value and then find MSE of prediction and actual value. For each difference between prediction and actual value, I sum up to SSD. Finally, I divide SSD by the total number of data points in the terminal set.

```python
def evalSymbReg(individual):
    # Transform the tree expression in a callable function
    func = toolbox.compile(expr=individual)
    # Evaluate the mean squared error between the expression
    SSD = 0
    for i in range(len(x)):
        SSD += (y[i] - func(x[i]))**2
    SSD = SSD/len(X)
    return SSD,
```

d)

- Random seed: 100
- Population: 300
- Min depth: 1
- Max Depth: 10
- Stopping criteria would be the number of generations: in this case study I set it to 100

e)

- First solution:
    - Random seed: 128
    - Generation: 100
    - Fitness value: 0.0099691

88	184	147.362	88	36469.3	0.0107541	184	2112.19	315.55	88	394	173	184	22.8592
89	163	13.9856	89	1059.21	0.0107541	163	74.3692	312.95	89	380	116	163	27.2904
90	165	15.7246	90	2097.88	0.0107541	165	136.844	314.16	90	370	109	165	25.7773
91	162	33.7855	91	5508.21	0.0106209	162	331.563	316.857	91	402	113	162	21.8905
92	148	19.6323	92	2805.39	0.0106209	148	193.885	316.487	92	401	207	148	20.3684
93	158	84.3481	93	15742.8	0.0106209	158	946.694	317.377	93	387	176	158	19.6347
94	162	50.8609	94	10912.6	0.0106209	162	639.846	316.797	94	399	2	162	31.0646
95	142	45.4504	95	4662.76	0.0106209	142	372.097	319.79	95	398	3	142	26.5417
96	155	9.78808	96	1217.46	0.00985749	155	82.1695	320.8	96	391	179	155	20.8194
97	170	122.134	97	34003.4	0.00985749	170	1960.16	319.95	97	391	177	170	26.8467
98	168	39.3694	98	6893.74	0.0099691	168	406.158	321.68	98	389	178	168	25.1711
99	176	11.5453	99	2229.26	0.00975804	176	130.238	324.003	99	440	110	176	30.0346
100	173	20.7536	100	1917.59	0.0099691	173	155.16	328.787	100	414	84	173	32.2062

- Second solution:
    - Random seed: 222
    - Generation: 100
    - Fitness value: 0.0292067

88	180	19.5092	88	3202.14	0.0341892	180	193.614	829.25	88	914	735	180	17.4207
89	164	14.571	89	2584.1	0.0327044	164	151.067	826.433	89	914	650	164	24.7195
90	164	7.89994	90	842.813	0.0327044	164	55.4654	828.04	90	914	694	164	18.2833
91	148	4.48809	91	216.233	0.0327044	148	21.9198	828.56	91	995	660	148	25.1938
92	157	3.74631	92	189.474	0.0327044	157	18.4554	826.743	92	925	412	157	36.1578
93	197	12.7092	93	1308.81	0.0314723	197	83.7826	827.377	93	904	409	197	39.8325
94	183	8.65595	94	947.111	0.0314723	183	60.0072	834.16	94	943	738	183	21.0917
95	161	2.30614	95	102.022	0.0314723	161	10.0901	839.143	95	926	720	161	22.7677
96	160	5.20109	96	160.751	0.0314723	160	21.4425	841.657	96	922	501	160	28.1029
97	154	18.7896	97	3390.06	0.0314723	154	205.107	843.787	97	922	613	154	24.8041
98	163	6.93312	98	576.381	0.0292067	163	39.0091	845.513	98	952	678	163	24.9103
99	139	3.68168	99	224.606	0.0292067	139	19.0011	845.407	99	952	456	139	31.2793
100	140	3.23504	100	198.076	0.0292067	140	17.1111	846.917	100	925	734	140	18.308

- Third solution:
    - Random seed: 122
    - Generation: 100
    - Fitness value: 0.0682912

90	156	16.0553	90	669.389	0.0877858	156	65.4081	164.047	90	226	70	156	13.536
91	156	7.54843	91	186.607	0.0877858	156	24.3096	163.78	91	215	31	156	15.2547
92	165	21.6259	92	3594.75	0.0877858	165	212.182	164.753	92	222	55	165	14.6105
93	169	7.7815	93	305.636	0.0686247	169	29.1707	164.637	93	248	37	169	15.3499
94	164	7.41172	94	186.607	0.0686247	164	25.1378	164.347	94	246	42	164	18.1903
95	166	11.8371	95	1001.52	0.0686247	166	63.799	164.437	95	247	51	166	16.2737
96	170	22.9246	96	4704.37	0.0686247	170	272.052	165.17	96	247	77	170	15.7975
97	162	9.14464	97	277.599	0.0686247	162	36.9988	164.61	97	223	69	162	15.6867
98	170	22.7179	98	3788.4	0.0686247	170	221.073	166.763	98	245	36	170	15.3911
99	174	18.2847	99	1505.46	0.0686247	174	113.072	168.423	99	263	13	174	20.466
100	156	6.30652	100	138.4	0.0682912	156	21.6696	171.007	100	245	41	156	17.6463

- There are differences among all three different results which proves setting different seeds will have some effects on our final results.