Part 1: Classifying Penguins with a Neural Network

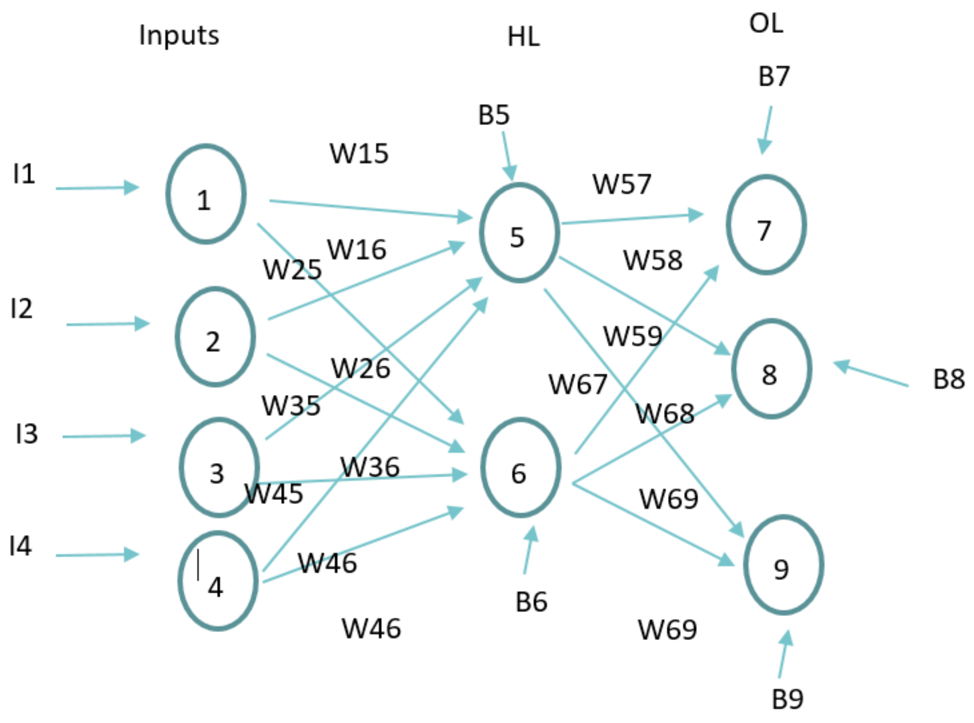

Fig.1 – NN with biases and weights numbers

Initial hidden layer weights

	1	2	3	4
5	-0.28	0.08	-0.3	0.1
6	-0.22	0.2	0.32	0.01

Initial output layer weights

	5	6
7	-0.29	0.08
8	0.03	0.13
9	0.21	-0.36

Biases

	5	6	7	8	9
	-0.33	0.26	0.06	-0.02	-0.2

1. ***Report the output and predicted class of the first instance in the dataset using the provided weights.***

The first output is [0.47627171854273365, 0.5212308085921598, 0.47617240962812674] The highest number is 0.52123 so it could be encoded as [0,1,0] or [1], which is the second instance according to integer and one hot encoded labels and this is a Chinstrap.

2. ***Report the updated weights of the network after applying a single back-propagation based on only of the first instance.***

Results :

```
Hidden layer weights:
 [[-0.28052064 -0.21971835]
 [ 0.07839682  0.20086728]
 [-0.30115025  0.32062226]
 [ 0.09937879  0.01033606]]
Output layer weights:
 [[-0.27752534  0.01757923  0.19865828]
 [ 0.0941994   0.11586195 -0.37290981]]
```

3. ***Report the final weights and accuracy on the test set after 100 epochs. Analyse the test accuracy and discuss your thoughts.***

ACCURACY: 82,84%

```
After training:
Hidden layer weights:
 [[ 0.93300053 -9.81238895]
 [-7.29027973  5.20341616]
 [ 2.38938873 -1.40616717]
 [ 2.47148091  1.43004753]]
Output layer weights:
 [[ -9.67263879  -2.44486417   3.24212769]
 [  4.90797584  -2.87370739 -11.64832746]]
```

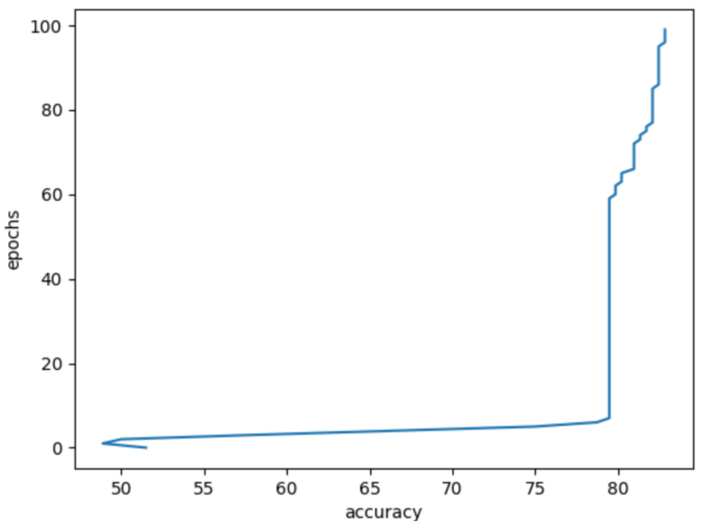

Fig.2 Accuracy for test set 100 epochs.

The NN demonstrates very quick convergence. The accuracy started to increase sharply from 5-th epoch and then, after 60-th shows not stable behaviour what might tell about overfitting.

4. Discuss how your network performed compared to what you expected. Did it converge quickly? Do you think it is overfitted?

We could check it using the test data set.
Accuracy starts from 0 end then in 5-th epoch ACCURACY for Test dataset: 58.46 %
If to look at the Fig3. we can see that around 12-th epoch accuracy for test set is maximum 63.08 %   and then starts decreasing after 72 epochs.

For this NN minimum number of epochs to converge should be 12 and it is overfitted after 72 epoch.

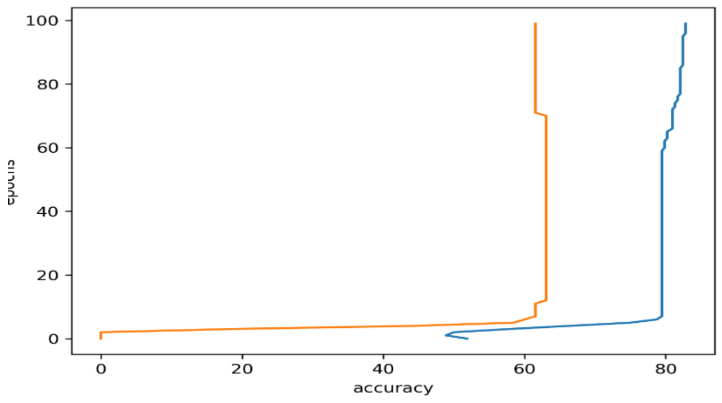

Fig. 3 Accuracy for training set (blue) and test set (orange)

5. ***Add bias nodes into the above network (with the below initial weights). Train it using the same parameters as before, and report your test accuracy. Compare the accuracy achieved to that of the original network, and discuss possible reasons for any performance differences.***

There are 2 additional files a2_Part1_with_biases.python and euralNetwork_with_biases.py

Initial biases taken from initial table , added to forward path and updated together with weights.

ACCURACY: 97.39 % on 28-th epoch for training set, however test set accuracy is decreasing after it-s maximum 47.69 % on 13-th epoch

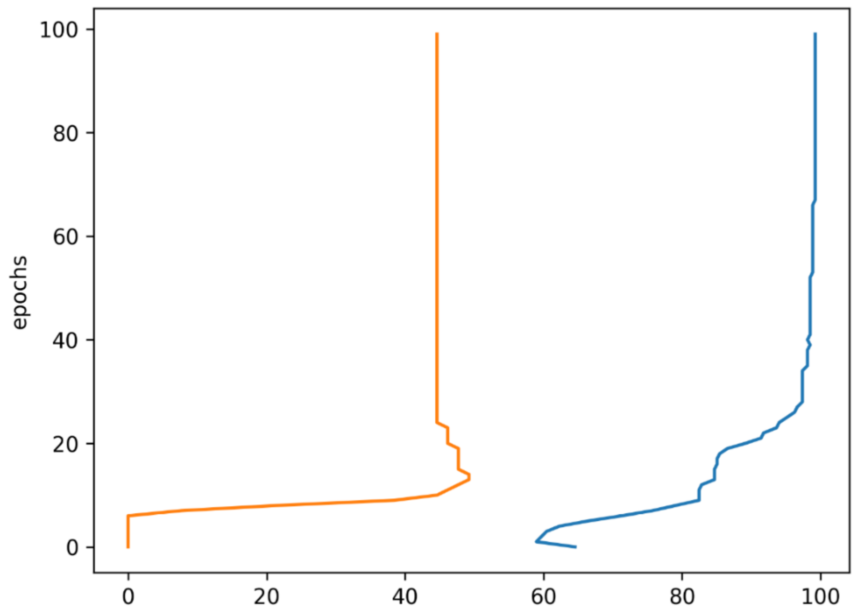

Fig.4. Accuracy vs Epochs with biases calculated for Training set(blue) and Test set (orange)

As bias transforms activation function and allows to move the line to fit the predictions of the training set, the overfitting problem could not be solved by adding the biases, moreover it is even getting bigger, as Fig3 and Fig4. Are indicating.

About the scripts: I have removed the train function from the NeuralNetwork.py file and implemented the training with epochs, all the instances , testing and plotting it in “main” function of a2Part1.py script.

Part 2: Genetic Programming for Symbolic Regression

1. ***Justify the terminal set you used for this task***

Terminal sets for the Symbolic Regression task are x values which are going as points to evalSymbReg function. And also random constant which is represented as

```
pset.addEphemeralConstant("rand101", lambda: random.randint(-1,1))
```

in the main programme.

2. ***Justify the function set you used for this task.***

Function set for symbolic regression mostly the same as for as GP. {+,-,x,%(safe division)). In the programme they are set here:

```
pset.addPrimitive(operator.add, 2)

pset.addPrimitive(operator.sub, 2)

pset.addPrimitive(operator.mul, 2)

pset.addPrimitive(protectedDiv, 2)

pset.addPrimitive(operator.neg, 1)
```

3. ***Formulate the fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate, e.g. good pseudo-code).***

Fitness class for the programme going to be x and corresponding y in the regression.txt dataset. Fitness measure in SymbolicRegression could be a sum of absolute errors.

$f=\sum|y_i-y\hat{}_i|$

or sum of mean squared errors

$f=\sum((y_i-y\hat{}_i)**2)/n$

i – number of current instance; n-lengths of set

For this purpose MSE is used in the assignment.

In Python this formula could be written as following.

math.fsum((func(x_list[ind]) - y_list[ind])2 for ind in range(len(x_list))) / len(x_list)**

x_list -list of x values, y-list – list of y values

4. *Justify the relevant parameter values and the stopping criteria you used.*

Parameters which are used in `eaSimple` function

Population - 300

cxpb (crossover probability) – 0.5

mutpb (mutation probability) – 0.1

ngen (number of genreations) -200

Stopping criteria for this programme is number of generations.

5. *Different GP runs will produce a different best solution. List three different best solutions evolved by GP and their fitness values (you will need to run your GP system several times with different random seeds).*

5.1. *For random.seed(318)*

sub(mul(add(-1, add(x, x)), add(x, x)), add(add(-1, add(x, mul(protectedDiv(sub(0, x), neg(protectedDiv(sub(protectedDiv(add(-1, add(x, x)), x), protectedDiv(x, -1)), -1))), protectedDiv(sub(protectedDiv(add(-1, add(x, x)), x), protectedDiv(protectedDiv(x, -1), -1)), -1)))), neg(protectedDiv(sub(protectedDiv(add(sub(add(x, x), sub(x, x)), x), x), protectedDiv(-1, add(add(-1, add(x, mul(protectedDiv(sub(add(x, x), protectedDiv(sub(protectedDiv(add(-1, add(x, x)), x), protectedDiv(protectedDiv(x, -1), -1)), -1)), neg(protectedDiv(sub(protectedDiv(add(add(x, -1), add(x, x)), add(x, x)), protectedDiv(protectedDiv(-1, 0), -1)), -1))), add(-1, x)))), neg(protectedDiv(sub(protectedDiv(add(-1, add(x, x)), x), protectedDiv(protectedDiv(x, -1), -1)), -1))))), -1))))

(2.7749014841789874,)

5.2. *For random.seed(200)*

mul(sub(x, 1), add(x, add(x, sub(add(x, x), protectedDiv(add(sub(add(add(protectedDiv(add(1, x), neg(-1)), x), protectedDiv(sub(x, 1), protectedDiv(protectedDiv(add(sub(x, 1), x), sub(-1, x)), sub(x, protectedDiv(x, sub(-1, x)))))), protectedDiv(protectedDiv(x, sub(-1, x)), sub(add(sub(add(x, protectedDiv(sub(x, 1), sub(-1, protectedDiv(x, sub(-1, x))))), protectedDiv(protectedDiv(x, sub(-1, x)), sub(x, protectedDiv(x, sub(-1, x))))), x), protectedDiv(x, sub(-1, x))))), protectedDiv(x, sub(-1, x))), sub(add(x, x), protectedDiv(x, sub(-1, x))))))))

(2.3049533406824945,)

5.3. *For random.seed(1001)*

mul(mul(add(x, -1), mul(x, x)), sub(x, 1))

(1.0000037500070316,)

6. ***optional, bonus, 5 marks) Analyse one of the best programs from above and explain how different parts the tree “work together” to solve the task.***

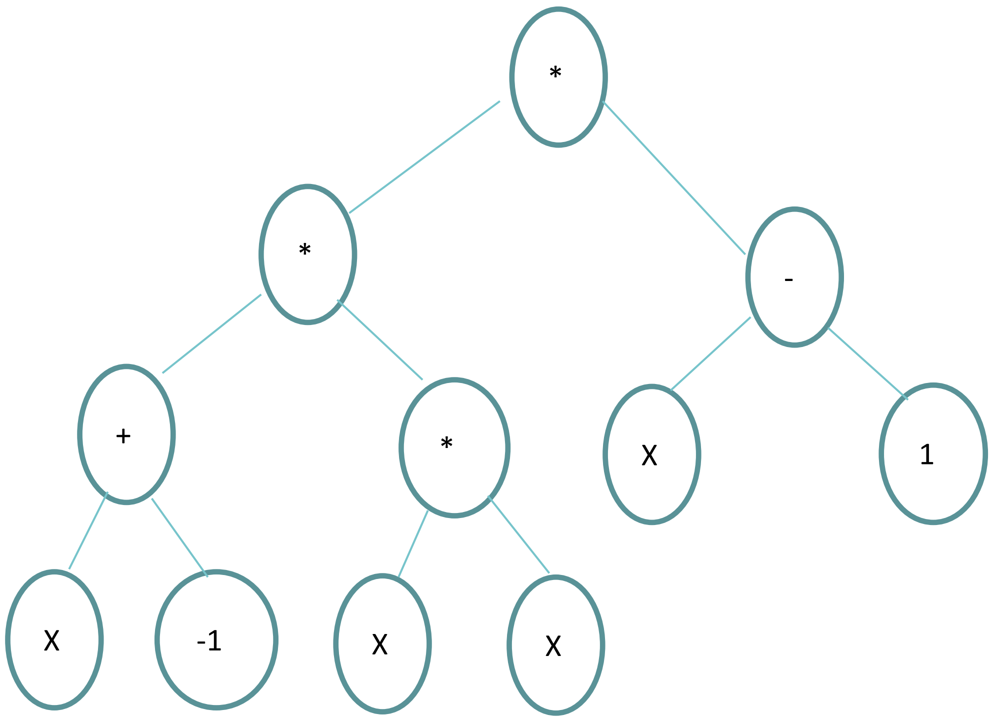

Fig 5. The best equation in tree

((x-1)*x*x)*(x-1) =(x-1)*(x-1)*x*x = (x-1)2 *x**2==(x**2-2*1*x+1**2)X**2 = x**4-2*x**3+x**2**

If transform this to normal equation view, we will see the following formula.

$y = x^4 - 2x^3 + x^2$

About the script:

The example was taken from
https://github.com/DEAP/deap/blob/master/examples/gp/symbreg.py

evalSymbReg function, toolbox were modified to fit the task