Report for Assignment 1 Part 3
“Perception” part 1

1.1. The program is doing 20 iteration by default. It reaches 0 wrong rate after around 3 iterations.

It can take the parameters

python assign1-3_2_1.py image.data <your number of iterations> <your number of features><your learning rate> < f – for initial weights equal not zero but random number>< p – for printing images>< wout/rout- for printing rates or weights>

python assign1-3_1.py image.data 4 50 0.1 t f rout

Report with input parameters:

Wrong guesses: 0

Stopped changing on 2

Number of iterations - 4, number of features - 50, learning rate - 0.1, weights as zeros - True, print images False, print weights out (wout) False, print rates out (rout) True

It is printing out a wrong rate. And the number of the iteration when it stropped changing.

python assign1-3_1.py image.data 10 50 0.1 f f wout

This command will give the rates which also freeze on the same iteration.

Changing learning rate from 0.1 to 0.01 gives worse result.

It is also possible to change a number of iterations from 10 to 5 for example. It does not make difference as the number of iterations it stop increasing less than 5

It is also possible to change a number of features. If there are (3 not 50) features the average number of Iteration to receive 0 wrong will increase to 11.

The evaluating the perceptron's performance on the training data is not a good measure of effectiveness the same as for other ML methods, as the model is built based on dataset and there is a risk to be overfitted to include all the inputs of the training data. And it makes the model not generalising well previously unseen data.

(Could not generate additional images with python so added other dataset -modified iris data set -to the second part)

“Perception” part 2

1.2. The program is doing 20 iterations by default. It reaches 87.5 correct rate after 11 iterations.

Iterration 11

Weigths: [[-1.1, 0, 0, -0.6, -0.4, 0, 0, -1.2], [-1.1, 0, 0, -0.6, -0.4, 0, 0, -1.2], [-1.1, 0, 0, -0.6, -0.4, 0, 0, -1.2]]

Correct rate: 87.5%

Modification of input parameters .

Set rounds(third position) = 30, leaving learning rate as 0.1, bios as 1 and setting initial weights as random numbers between -1 and 1.

python assign1-3_2.py ass1_3_2.data 30 0.1 1 f

The result is a little bit different each time. The number of iterations to come to 87.5 will be different.

1 run:

Iterration 14

Weigths: [[-1.7, -0.19, 0.0, -0.43, -0.35, 0.45, -0.59, -0.69], [-1.05, -0.84, -0.27, -0.64, -0.31, 0.01, -0.03, -1.56], [-1.1, -0.84, 0.2, -1.13, -0.45, 0.94, 0.66, -0.6]]

Correct rate: 87.5%

2 run:

Iterration 11

Weigths: [[-0.97, -0.07, 0.06, -0.79, -0.13, -0.61, -0.44, -1.68], [-0.34, 0.02, 0.74, -0.45, -0.43, -0.96, -0.12, -1.99], [-1.02, 0.44, 0.06, -0.43, -0.59, 0.68, -0.85, -2.19]]

Correct rate: 87.5%

3 run:

Iterration 16

Weigths: [[-1.67, -0.11, 0.8, -1.12, -0.12, -0.49, -0.79, -2.02], [-2.13, -0.89, -0.33, -0.02, -0.07, 0.43, 0.23, -1.29], [-1.03, 0.47, 0.87, -1.07, -1.09, 0.83, 0.91, -1.11]]

Correct rate: 87.5%

4 run:

Iterration 6

Weigths: [[0.52, -0.68, 0.94, -0.97, -0.23, 0.16, 0.16, -0.52], [0.16, 0.4, 0.88, -0.18, -1.2, 0.22, -0.39, -1.16], [-1.08, 0.65, -0.38, 0.15, 0.23, 0.7, -0.24, -0.62]]

Correct rate: 87.5

5 run:

Iterration 6

Weigths: [[0.17, 0.51, -0.6, -0.48, -0.15, 0.56, -0.54, -1.7], [-1.51, -0.05, 0.33, -0.98, -0.6, -0.2, -0.34, 0.25], [-1.08, 0.46, -0.11, -0.97, -0.26, -0.53, 0.66, -1.62]]

Correct rate: 87.5%

Some ideas of perception function for binary data are taken from :

https://www.geeksforgeeks.org/implementation-of-perceptron-algorithm-for-or-logic-gate-with-2-bit-binary-input/

2.2.Analysis

87.5% of correct was a maximum correct rate result , considering a limited number of features – 3. And only 8 instances. So it takes more times to reach the maximum correct rate rather than in the first example with 50 features. This model is simpler then previous one and bios could be added as separate parameter and also modified. The specific of this set was binary input. However, the same model is working well with other datasets as well.

2.3. Adding another dataser

If to run the same script wit a different data set.
python assign1-3_2.py iris_test.txt 20 0.1 1 f or python assign1-3_2.py iris_test.txt 20 0.1 1

The rate of corrected guesses will be 99.01%

This result will come after around 11 iterations with random weights from the beginning.

Iteration 11

Weigths: [[-0.13, -0.41, 0.14, -0.86, -0.1, -0.25, -0.53, -0.29, -0.39, 0.2, 0.21, -0.1, -0.89, -0.69, -0.31, -0.35, -0.08, 0.1, 0.3, 0.23, -0.35, -0.47, -0.1, -0.02, -0.86, -0.33, 0.27, -0.5, -0.55, 0.28, 0.27, -0.12, -0.1, -0.3, 0.07, 0.19, -0.23, -0.22, -0.24, -0.32, -0.18, -0.16, -0.43, 0.06, -0.14, -0.44, -0.8, -0.06, -0.81, -0.33, 0.99, -0.09, -0.53, -0.85, 0.48, 0.16, 0.88, 0.67, 0.87, -0.18, 0.26, 0.99, -0.75, 0.53, 0.11, -0.07, -0.35, -0.1, 0.35, -0.12, 0.03, -0.52, -0.25, 0.29, -0.01, 0.08, 0.18, -0.43, 0.51, -0.01, 0.89, -0.52, -0.2, -0.38, 0.72, 0.56, 0.39, 0.82, 0.75, 0.16, -0.55, 0.91, -0.43, 0.58, -0.06, -0.14, -0.43, -0.14, 0.19, -0.82, 0.65], [-0.33, 0.07, -1.06, 0.83, -0.53, -0.33, -0.11, -0.1, 0.05, -1.25, -1.2, -0.39, 0.96, 0.23, -0.48, 0.26, -0.88, -0.91, -1.08, -1.35, 0.04, 0.37, -0.56, -0.64, -0.81, -0.29, -1.34, -0.94, 0.15, -1.37, -1.33, -0.91, -0.94, 0.13, -0.9, -1.67, -0.22, -0.14, -0.03, -0.05, -0.69, -0.29, -0.79, -0.73, -0.34, -0.9, -0.17, -0.6, -0.03, 0.11, 0.14, -0.05, 0.22, 0.84, 0.7, 0.12, -0.52, 0.07, 0.82, -0.33, -0.02, -0.39, 0.91, -0.24, 0.31, -0.42, 0.08, 0.14, -0.29, 0.29, 0.91, 0.95, 0.46, 0.94, -0.17, 0.8, -0.54, 0.98, 0.08, 0.01, 0.9, 0.54, 0.01, -0.15, -0.38, 0.62, 0.43, 0.81, -0.39, 0.61, 0.64, -0.67, -0.09, 0.94, -0.2, 0.17, 0.02, -0.19, 0.18, 0.92, 0.21], [-0.57, -0.17, -1.7, -0.23, -0.88, 0.35, -0.96, -1.62, -0.19, -1.39, -0.66, -0.89, 0.82, -0.06, 0.61, -0.95, -0.11, -0.69, -1.2, 0.01, 0.6, -0.24, -0.75, 0.29, -0.64, -0.09, -0.64, 0.19, 0.79, 0.39, -0.18, 0.72, 0.54, -0.5, -0.38, 0.17, -0.85, -1.28, -1.49, -1.07, 0.66, -0.96, 0.76, -0.57, -0.17, 0.27, -0.08, -0.83, 0.94, -0.59, -0.59, 0.75, 0.87, 0.49, -0.68, -0.52, -0.69, -0.99, 0.28, 0.59, 0.86, 0.71, -0.38, -0.63, -0.26, 1.15, 0.94, 0.48, -0.19, -0.27, -0.34, 0.39, 0.6, 0.89, -0.17, -0.03, 0.82, -0.13, 0.42, -0.41, 0.5, 0.09, -0.03, 0.54, -0.45, -0.77, -0.34, -0.1, 0.77, -0.16, -0.54, -0.46, 0.6, 0.94, 0.32, -0.45, 0.67, 0.79, -0.14, 0.28, 0.89]]

Correct rate: 99.01%

The data set is taken from iris.data and modified. It cut up to 100 instances.

There are only 2 classes Iris-setosa represented as 0 and Iris-versicolor represented as 1.

The first column with first the feature replaced in iris_test.txt by indexes. And the header added.