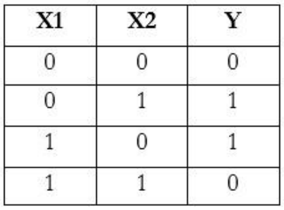
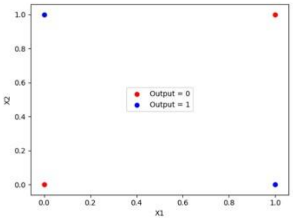
Problem 3

Manually design MLP network to perform the XOR Gate with the truth table and its plot on 2D as follows:



X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0



Start with uniform random initialization for parameters w_{ij} . Perform forward and backward pass in the following case:

1. Activation function is (a) Sigmod (b) ReLu, (c)Tanh
2. Divergence is defined as (a) L2_norm, (b) cross entropy
3. Train the network for 2 iterations (3 forward pass and 2 backward pass)

Please report the parameters and actual output from the MLP in each iteration

Solution:

Results:

1. Activation function = “Sigmoid”:

(a) Loss function = “L2_norm”:

Iteration #	Hidden_Layer _Weights	Hidden_Layer _bias	Output_Layer _Weights	Output_Layer _bias	Output
1	[[4.17e-01 7.20e-01] [2.88e-04 3.02e-01]]	[[0.186 0.186]]	[[0.1533] [0.0994]]	[[0.35657451]]	[[0.61685237] [0.61845272] [0.62391468] [0.62516723]]
2	[[4.17e-01 7.20e-01] [2.95e-04 3.02e-01]]	[[0.186 0.186]]	[[0.1535] [0.0997]]	[[0.35699465]]	[[0.621227] [0.62294524] [0.62868852] [0.63003223]]
3rd Forward Pass	[[4.17e-01 7.20e-01] [2.95e-04 3.02e-01]]	[[0.186 0.186]]	[[0.15359096] [0.09975817]]	[[0.35699465]]	[[0.62139453] [0.62311739] [0.62887147] [0.63021874]]

(b) Loss function = “cross entropy”:

Iteration #	Hidden_Layer _Weights	Hidden_Layer _bias	Output_Layer _Weights	Output_Layer _bias	Output
1	[[4.17e-01 7.20e-01] [2.94e-04 3.02e-01]]	[[0.186 0.186]]	[[0.1535] [0.0997]]	[[0.35692588]]	[[0.62139453] [0.62311739] [0.62887147] [0.63021874]]

2	[[4.17e-01 7.20e-01] [2.94e-04 3.02e-01]]	[[0.186 0.186]]	[[0.1535] [0.0997]]	[[0.35693714]]	[[0.6213671] [0.62308921] [0.62884152] [0.63018821]]
3rd Forward Pass	[[4.17e-01 7.20e-01] [2.94e-04 3.02e-01]]	[[0.186 0.186]]	[[0.15355652] [0.09971985]]	[[0.35693714]]	[[0.62137159] [0.62309382] [0.62884642] [0.6301932]]

2. Activation function = "ReLu":

(a) Loss function = "L2_norm":

Iteration #	Hidden_Layer_Weights	Hidden_Layer_bias	Output_Layer_Weights	Output_Layer_bias	Output
1	[[0.41493 0.71895] [-0.00351 0.29996]]	[[0.17284 0.17284]]	[[0.13699] [0.07115]]	[[0.3015187]]	[[0.40426918] [0.43447352] [0.54017733] [0.57038167]]
2	[[0.41221 0.71754] [-0.00598 0.29867]]	[[0.16573 0.16573]]	[[0.12291] [0.045]]	[[0.26735965]]	[[0.34208635] [0.43274728] [0.35289814] [0.64103395]]
3rd Forward Pass	[[0.41221 0.71754] [-0.00598 0.29867]]	[[0.16573 0.16573]]	[[0.12291263] [0.0455603]]	[[0.26735965]]	[[0.36520978] [0.46835011] [0.39459023] [0.66166997]]

(b) Loss function = "cross entropy":

Iteration #	Hidden_Layer_Weights	Hidden_Layer_bias	Output_Layer_Weights	Output_Layer_bias	Output
1	[[0.41956 0.72026] [0.00076 0.30118]]	[[0.18319851 0.18319851]]	[[0.16443078] [0.12209497]]	[[0.37103171]]	[[0.36520978] [0.46835011] [0.39459023] [0.66166997]]
2	[[0.39022 0.69847] [-0.02502 0.2820]]	[[0.09944918 0.09944918]]	[[0.03588006] [-0.10725715]]	[[0.07873921]]	[[0.35136858] [0.44136289] [0.35136858] [0.71886144]]
3rd Forward Pass	[[0.3902 0.69847] [-0.02502 0.28202]]	[[0.09944918 0.09944918]]	[[0.03588006] [-0.10725715]]	[[0.07873921]]	[[0.35006347] [0.65385821] [0.64978023] [1.23263521]]

3. Activation function = "Tanh":

(a) Loss function = "L2_norm":

Iteration #	Hidden_Layer_Weights	Hidden_Layer_bias	Output_Layer_Weights	Output_Layer_bias	Output
1	[[0.38676 0.70611] [-0.02833 0.28963]]	[[0.10855063 0.10855063]]	[[-0.02232229] [-0.21101983]]	[[-0.11444101]]	[[0.84237693] [0.90311116] [0.95375232] [0.96533756]]
2	[[0.38775 0.71299] [-0.02732 0.29700]]	[[0.12585137 0.12585137]]	[[-0.05040367] [-0.26152037]]	[[-0.2115684]]	[[0.8100415] [0.88303578]]

					[0.94516725] [0.95935902]]
3rd Forward Pass	[[0.38775 0.71299] [-0.02732 0.29700]]	[[0.12585137 0.12585137]]	[[-0.05040367] [-0.26152037]]	[[-0.2115684]]	[[0.76673228] [0.85577097] [0.93355129] [0.95138511]]

(b) Loss function = “cross entropy”:

Iteration #	Hidden_Layer _Weights	Hidden_Layer _bias	Output_Layer _Weights	Output_Layer _bias	Output
1	[[0.38938 0.71923] [-0.02523 0.30558]]	[[0.15097123 0.15097123]]	[[-0.07548259] [-0.30574885]]	[[-0.31034454]]	[[0.76673228] [0.85577097] [0.93355129] [0.95138511]]
2	[[0.3907 0.72331] [-0.0234 0.31137]]	[[0.1689692 0.1689692]]	[[-0.09136964] [-0.33263613]]	[[-0.36845206]]	[[0.96937507] [0.98251721] [0.98994769] [0.99208102]]
3rd Forward Pass	[[0.3907 0.72331] [-0.0234 0.31137]]	[[0.1689692 0.1689692]]	[[-0.09136964] [-0.33263613]]	[[-0.36845206]]	[[0.99731293] [0.99835904] [0.99886269] [0.99903997]]

Problem 4

Building a MLP with one hidden layer to perform classification task with the following description:

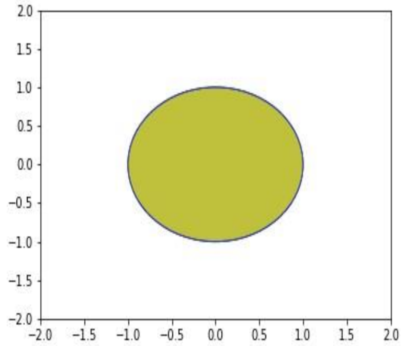
+ Training data (X, Y):

Training data contains N_1 = 10,000 points in 2-dimentional space and are followed by the uniform radius between 0 and 2 and its label is 1 is it is inside the yellow circle, otherwise it is 0 + Validation data (X, Y):

Validation data contains N_2 = 2,000 points in 2-dimentional space and are followed by the uniform radius between 0 and 2 and its label is 1 is it is inside the yellow circle, otherwise it is 0

+ Testing data (X, Y):

Testing data contains N_2 = 2,000 points in 2-dimentional space and are followed by the uniform radius between 0 and 2 and its label is 1 is it is inside the yellow circle, otherwise it is 0



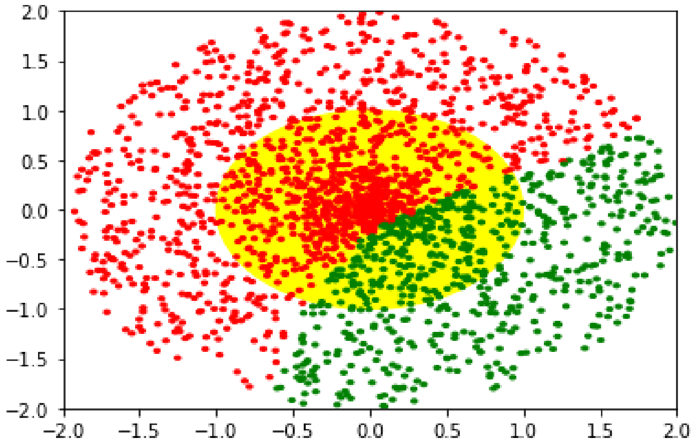
Assume that we use CrossEntropyLoss (nn. CrossEntropyLoss) and GradientDescent(torch.optim.SGD) with lr = 0.01

Report the training loss, validation loss, testing accuracy (in number and visualized by figure) in the following case
+ Train the MLP with 10 iterations
+ Train the MLP with 100 iterations
+ Train the MLP with 1000 iterations

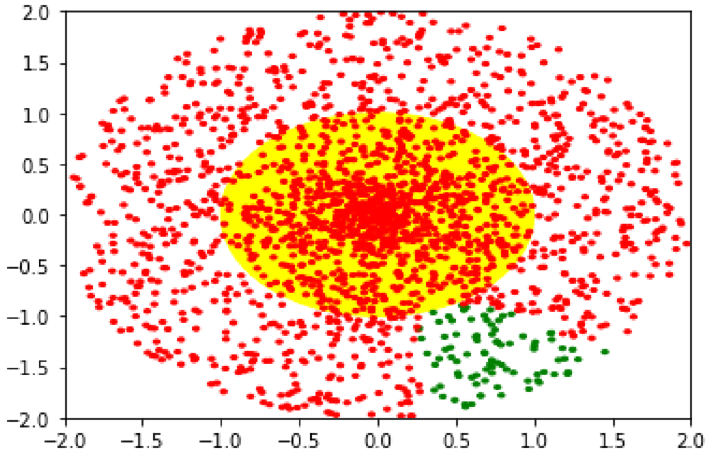
Note: use import **matplotlib.pyplot** to plot figures

Solution:

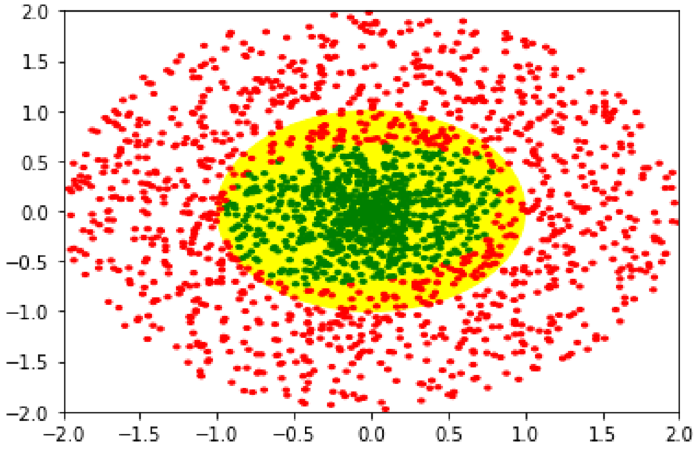
<u>After training with 10 iterations:</u>

Parameters	**Value**	**Plot**
Training loss	0.69662243	
Validation loss	0.6973139	
Testing accuracy	0.4485	

<u>After training with 100 iterations:</u>

Parameters	**Value**	**Plot**
Training loss	0.702289	
Validation loss	0.7016716	
Testing accuracy	0.4525	

<u>After training with 1000 iterations:</u>

Parameters	**Value**	**Plot**
Training loss	0.6067765	
Validation loss	.60660905	
Testing accuracy	0.8995	