cs577 Assignment 1: Report

Yuanxing Cheng, A20453410, CS577-f22
Department of Mathematics
Illinois Institute of Technology

September 21, 2022

Question 1

Problem Statement

Doing several bi-class classification porblem.

data types

- A: a ring of label 1, and label 2 inside
- B: first and third quadrant of label 1, and label 2 otherwise
- C: 2 non-overlapping circle
- D: 2 spiral curve

Proposed solution

Use densely connected neural networks to classify the classes.

Implementation details

- the first three problem is quite easy, and it just take some time to figure out what features to use
- the last one is a bit of complex, so I will have to increase the number of hidden layer, and units within, also decrease the learning rate to ensure the convergence.
- To use the program, apply the result in the table of next subsection into the webpage.

Results and discussion

neural network architecture are represented in a vector: with length of the vector equal to the number of hidden layer and elements being the hidden units in that corresponding layer. For example, 1 hidden layer with 1 neuron is represented as $(1,)$; 2 hidden layers with 1 neuron in the first layer and 2 neurons in the second layer is represented as $(1,2,)$. The results are summarized in the following table.

Also, all experiments are run under the following condition

- ratio of training to test data: 0.7
- noise: 0
- batch size: 30

Data	features	learning rate	activation	regularization	reg-rate
A	x_1^2, x_2^2	0.3	Tanh	None	N/A
B	$x_1 x_2$	1	Tanh	None	N/A
C	$x_1, x_2, x_1 x_2$	1	Tanh	None	N/A
D	$x_1, x_2, x_1^2, x_2^2, \sin(x_1), \sin(x_2)$	0.03	Tanh	L2	0.001

Data	epochs	network architecture	trainingtraining loss	test loss
A	72	(1,)	0.000	0.000
B	46	(1,)	0.000	0.000
C	22	(1,)	0.000	0.000
D	800	(8,7)	0.014	0.017

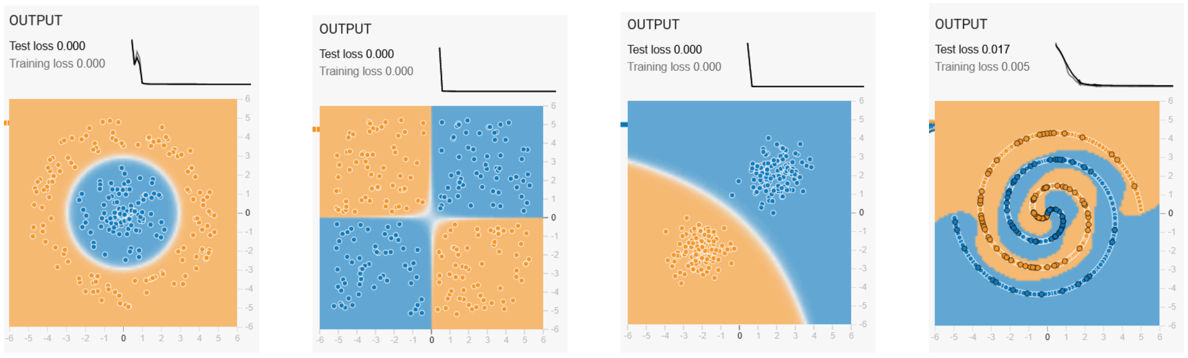

Figure 1: *4 classification results in tensorflow playground*

Question 2

Problem statement

image tri-class classification problem

Proposed solution

as asked, use densely connected neural networks to classify the classes.

Implementation details

I will use colab to do all the work. The dataset are load from tensorflow.datasets so no extra link. The classes I pick are the first three.

The training image is vectorized and converted to value between 0 and 1, and their labels are already in integers so no further process on it. The tuning process is summarized in the following tables

NN architecture	activation	optimizer	epoch	batch size
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	20	512
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	100	1024
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	200	1024
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	75	1024
(64,64,3)	(Tanh,Tanh,softmax)	rmsprop, lr=0.001	75	1024
(64,64,64,64,3)	(relu,...,relu,softmax)	rmsprop, lr=0.001	75	1024
(99,99,99,99,3)	(relu,...,relu,softmax)	rmsprop, lr=0.001	75	1024
$(64_{\times 7},3)$	(relu,...,relu,softmax)	rmsprop, lr=0.001	75	1024

loss	acc	val loss	val acc	comment on next tuning
0.7066	0.7066	0.6712	0.7153	haven't cvg yet and need bigger batch size
0.5245	0.7851	0.5729	0.7767	haven't cvg yet, but batch size enough
0.3873	0.8447	0.5927	0.7787	actually overfit after 75 epoch
0.5873	0.7577	0.5944	0.7547	tune activation
0.5590	0.7662	0.6739	0.7167	not good, tune NN
0.5394	0.7780	0.5991	0.7620	better, try more units then
0.5830	0.7582	0.6156	0.7413	not good, try more layers then
0.5597	0.7648	0.6177	0.7467	no big improvement, so save the 6th model

The loss vs epoch and accuracy vs epoch figures are below; row 1 and row 3 are loss vs epoch and row 2 and row 4 are accuracy vs epoch.

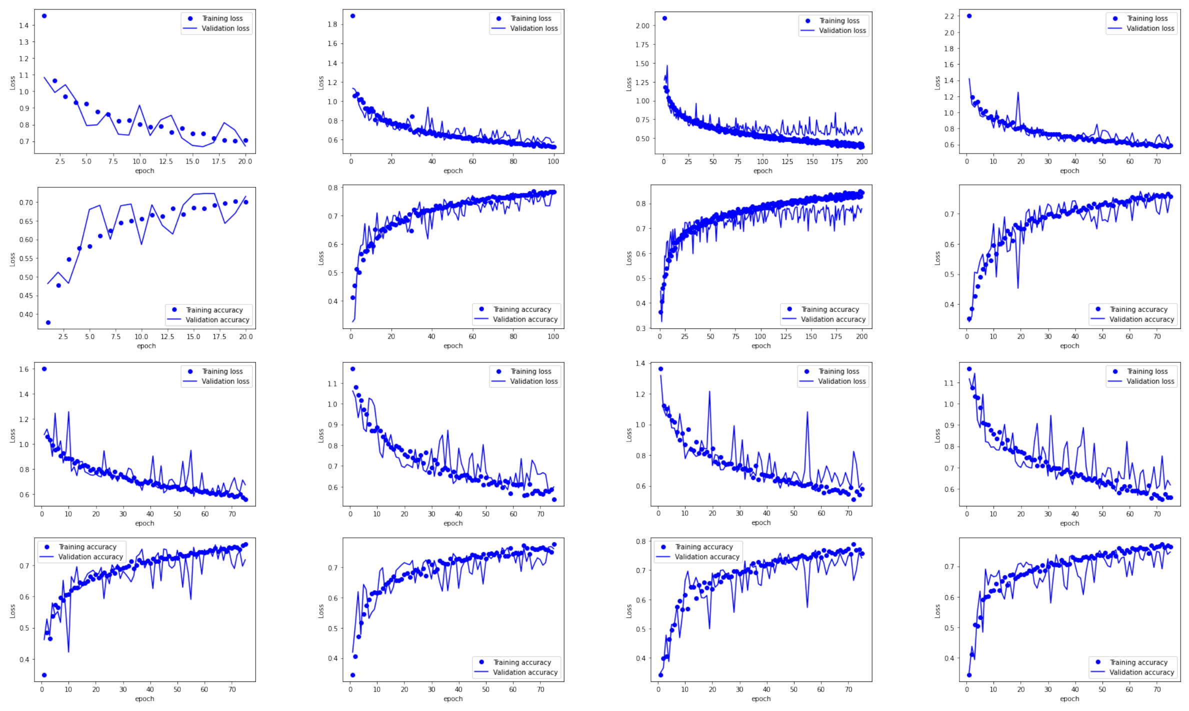

Figure 2: *subset cifar 10 classification tuning checkpoints*

Results and discussion

- train data size: 15000-1500
- validation data size: 1500
- test data size: 3000
- test acc: 0.7436

I know there'll could be better result, but considering the limited time, I am satisfied for the result.

The saved model is uploaded to a github repo, and the link is contained in both the notebook file and models.txt.

Question 3

Problem statement

text bi-class classification problem, non-spam or spam.

Proposed solution

same as last one

Implementation details

same as last one and here're the result tables and figures

NN architecture	activation	optimizer	epoch	batch size
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	20	512
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.0005	100	512
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.0005	30	512

loss	acc	val loss	val acc	comment on next tuning
0.1602	0.9390	0.1911	0.9325	not cvg and cvg too fast; need smaller lr, more epoch to see if overfit
0.0849	0.9734	0.1581	0.9425	better, now maybe stop at 30 epoch
0.1725	0.9379	0.2057	0.9300	i could stop here

The loss vs epoch and accuracy vs epoch figures are below; row 1 are loss vs epoch and row 2 are accuracy vs epoch.

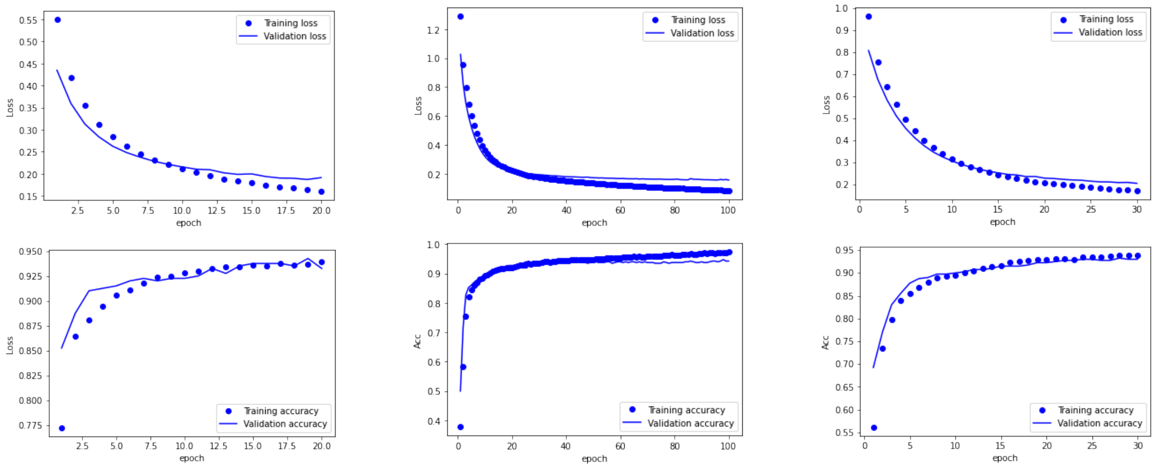

Figure 3: *spambase data bi-class classification tuning checkpoints*

Results and discussion

- train data size: 3220-400
- validation data size: 400
- test data size: 1381
- test acc: 0.9399

I'm pretty amazed on how easy this task is. Though clearly there could be a better result, but I am satisfied enough to carry on to the last question.

Question 4

Problem statement

a regression problem,

Proposed solution

as asked, use densely connected neural networks to do regression task.

Implementation details

About the dataset, it contains various missing values, spreading all the place, and no full rows. In this case, I will discard several columns of data for whose missing value percentage is about 60 percent or higher.

In all, the discarded columns represents

- LemasSwornFT: number of sworn full time police officers
- LemasSwFTPerPop: sworn full time police officers per 100K population
- LemasSwFTFieldOps: number of sworn full time police officers in field operations
- LemasSwFTFieldPerPop: sworn full time police officers in field operations
- LemasTotalReq: total requests for police
- LemasTotReqPerPop: total requests for police per 100K popuation
- PolicReqPerOffic: total requests for police per police officer
- PolicPerPop: police officers per 100K population
- RacialMatchCommPol: a measure of the racial match between the community and the police force. High values indicate proportions in community and police force are similar
- PctPolicWhite: percent of police that are caucasian
- PctPolicBlack: percent of police that are african american
- PctPolicHisp: percent of police that are hispanic
- PctPolicAsian: percent of police that are asian
- PctPolicMinor: percent of police that are minority of any kind
- OfficAssgnDrugUnits: number of officers assigned to special drug units
- NumKindsDrugsSeiz: number of different kinds of drugs seized
- PolicAveOTWorked: police average overtime worked
- PolicCars: number of police cars
- PolicOperBudg: police operating budget
- LemasPctPolicOnPatr: percent of sworn full time police officers on patrol
- LemasGangUnitDeploy: gang unit deployed
- PolicBudgPerPop: police operating budget per population

Next found that the first 5 column of data are non-predictive, I will also discard them.

Last step, delete rows containing the rest missing values, and transform the strings form to numerical form.

Then after normalizing using training dataset, we start to train the NN. Here's the table for tuning results. Note across them the following hyperparameters are maintained

- NN architecture: (64,64,3)
- activation: (relu,relu,softmax)
- optimizer: rmsprop, lr=0.001

epoch	batch size	mean validate mae	comment on next tune
20	1	0.1083	not cvg yet, need more epoch to see if overfit
100	1	0.1107	still
200	1	0.1163	I could stop here

The loss vs epoch and accuracy vs epoch figures are below; row 1 are loss vs epoch and row 2 are accuracy vs epoch.

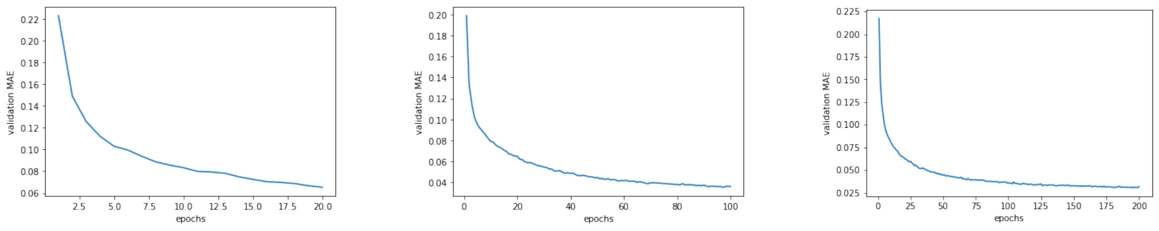

Figure 4: *crime Rate regression tuning checkpoints*

Results and discussion

- train data size: 1047
- validation data size: 348
- test data size: 598
- test acc: 0.09977

More smooth than I expected.