Assignement 1

Development environment

This assignment has been done using python on Visual Studio Code. I would have been wiser to use google collab because the training is still a bit slow on my personal computer due to the absence of GPU. All the training was done by cpu (intel core -i5 8th gen).

Boston Housing

Data Preprocessing

For the first dataset since the data were already an array of number, I only use the train_test_split function from sklearn to get two distinct dataset(aprox 80%,20%). Then I moved the last row of the table out to get the target array and transformed each array into tensor.

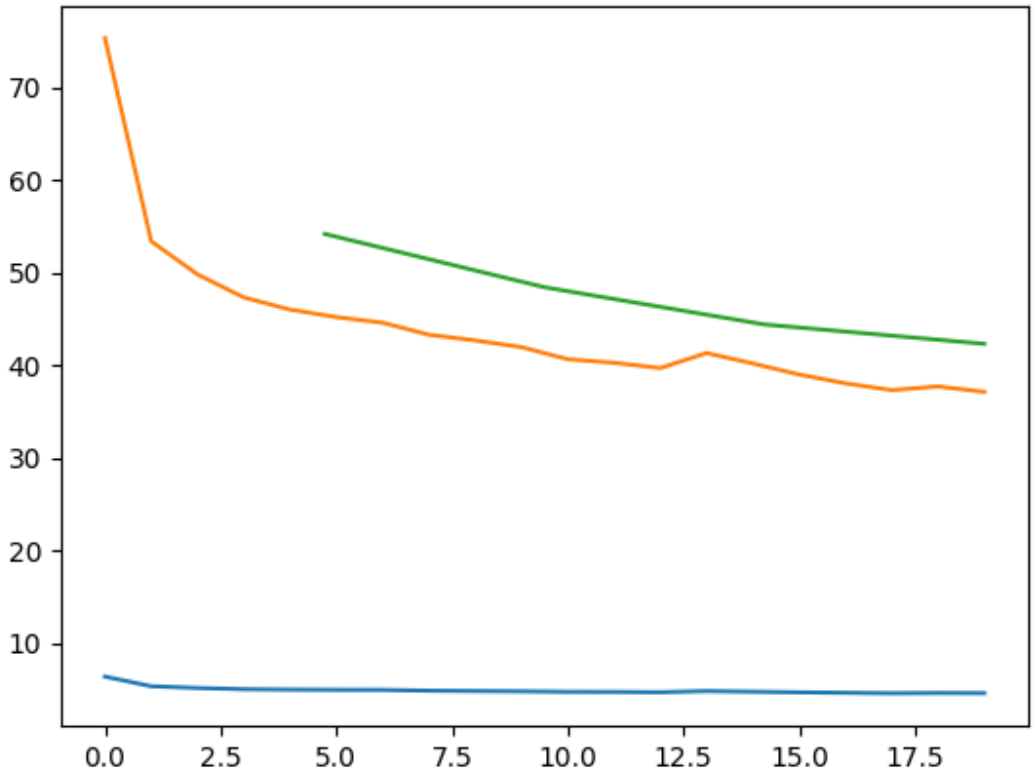
Model Architecture

For this problem we only need to do a regression, so I choose to use two linear gate going from 13 to 6 nodes and then from 6 to 1 node. I added a Relu activation layer in between to add some non-linearity to the model.

Choice of Hyperparameters

For all the model I left the epoch number on twenty because making more epoch won’t make a big improvement but may consume a lot of time, for the learning this one is at 0.01 because if I set it lower than that the curve are not nearly flat at the end of the training.

Learning Curves



with:

- Orange: Average loss for each epoch
- Blue: Average difference between the prediction and the correct answer
- Green: The average loss for the Validation epochs

Summary

Breast Cancer

Data Preprocessing

As for the previous exercise, I loaded the data and separate it into train and test data then into item and targets. Once this was done, I loaded everything into a

TensorDataset and then this dataset into a Dataloader to speed up training.

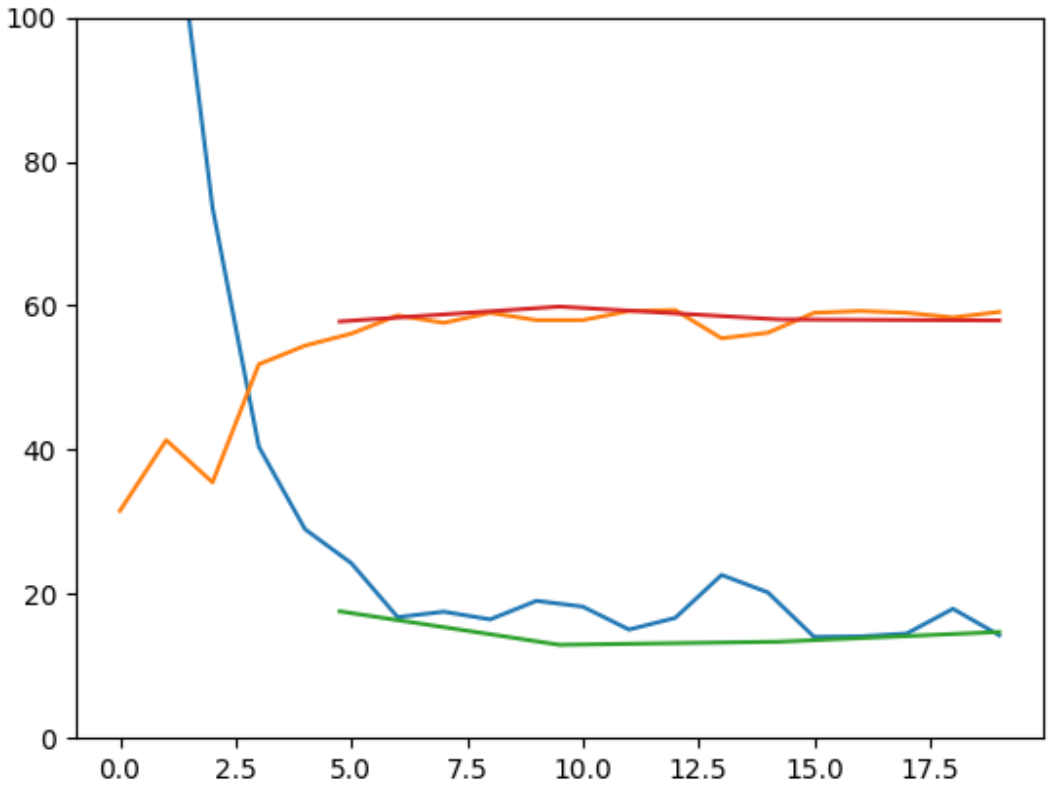
Model Architecture

Since this model is a binary classification problem, I made a neural network composed of two linear layers going from 30 to 15 and then from 15 to 1 with a Relu in between but this time I added a sigmoid layer at the end. With this layer I can assure that the result will either be 0 or 1.

Choice of hyperparameters

This one is not really stable if I put a learning rate greater than 0.005 the model will change to much and will look really random. For the evaluation of the loss, I choose to use a Binary_cross_entropy.

Learning Curves



with:

- Blue: Average loss per epoch
- Green: Average loss per validation epoch
- Orange: Average accuracy per training epoch
- Red: Average accuracy per validation epoch

Summary

The curves on this one are not really stable, and I might have missed something.
I think that the error is coming from the dataloader because I'm not sure of the way I did it.

Fashion Mnist

Data Preprocessing

For this one the data was already split into train and validation dataset so I didn’t had to do this part.

First of all I normalized the data by dividing each vector target by 255 (Using the value of grey goes from [0,255] to [0,1] wich is way easier to compute.). Then I transformed each label vector from a number to an array because my model output will be an array and if I use torch. argmax()) the criterion will be detached from the model. Then as for the last dataset, I put everything into a dataset and a dataloader to speed up training.

During the initialisation of the model I figured out that the conv2d layer needed a third dimension representing the color panel but since our dataset don’t have color panel I just add an empty dimension.

Model architecture

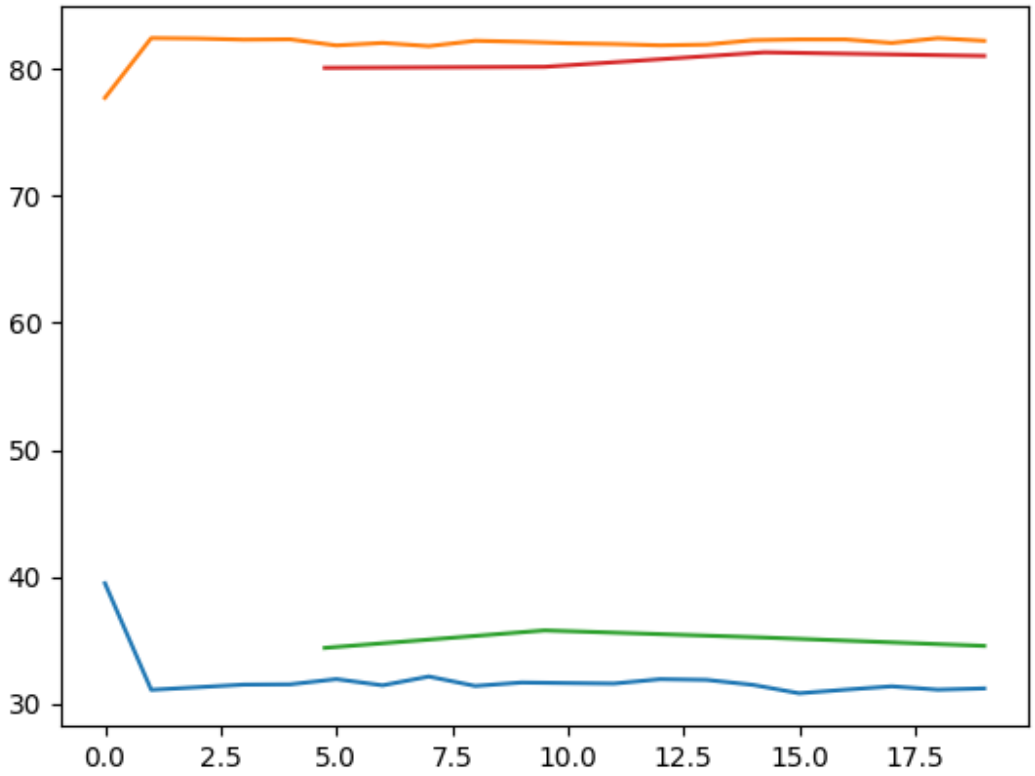
This model is a classical convolutional neural network:

1. Two conv2d layer with maxpooling (one with a stride the other not) to take in account the whole picture.
2. a view() function to flatten the result
3. a linear regression going to the 10 output node representing the 10 possible label

Choice of hyperparameters

For this one I used the classical combo with cross entropy loss and adam as optimizer, for the learning rate I choose a pretty fast one (0.01) because it doesn’t create any instability and it can reduce the risk of staying stuck.

Learning Curve



with:

- Orange: Accuracy per epoch
- Red: Accuracy per validation epoch
- Green: loss per validation epoch
- Blue: loss per validation epoch

Summary

This one was the hardest one, especially due to the format of the label and the need of a third dimensions for conv2d. I learned a lot about how tensor are attached to prediction due to torch.argmax(). I also learned that stopping a model when the training is not improving anything anymore is a good practice but I didn't had time to implement it.