Department of Electrical Engineering

Indian Institute of Technology Delhi

ELL 409
Machine Intelligence and Learning
Assignment 1

Ayushmaan Pandey
2021EE30709

Yajat Kapoor
2021EE30471

Contents

1 Linear Classification : Implementation

1.1 Dataset Description

- Input features
  - Original Dataset : It has a grayscale image of the cloth associated with each datapoint.
  - Normalised Dataset : It has a 1x28x28 tensor associated with each data point, and each value (representing color intensity in grayscale) is a floating point number between 0 and 1 (inclusive).
- Label
  - y:The label, e the type of cloth, takes 10 possible values and ranges from 0 to 9 (inclusive).
- Sample images from each label:

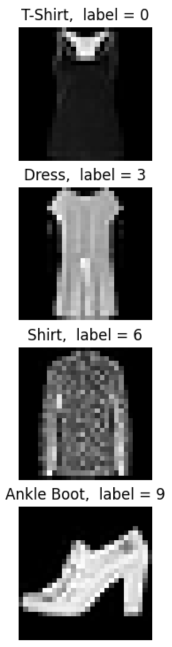

Figure 1: Example data points images from the dataset for each label

1.2 Problem Description

Task is to build a linear classification model to assign a class out of 10 possible labels to the cloth.

1.3 Data Preprocessing

We applied ToTensor() method to the testing and training datasets that converts the image into a tensor and also normalises its intensity in range 0 to 1.

1.4 Model

We have the following layers in the Sequential Learning based model:

- Flatten : It flattens the 2D 1x28x28 tensor into a 1D tensor of 1x784 dimension
- Linear layer : with 784 input features and 10 labels
- Batch Normalisation : This scales and shifts the input values so that their mean becomes 0 and variance 1
- Softmax activation function : softmax function, or normalized exponential function converts a vector of K real numbers into a probability distribution of K possible outcomes

1.5 Loss Function and Optimiser

Loss function : CrossEntropyLoss Optimiser : SGD

1.6 Results

Final accuracy of model is 77.80% . We can deduce that dataset is linear to a large degree since it has high accuracy with linear classification technique.

2 Linear Classification: : Hyperparameter Tuning and Regularization

2.1 Learning Rate tuning

We used ReduceLROnPlateau scheduler and obtained following data:

Initial Learning Rate	LR Factor	Patience	Accuracy
0.01	0.1	5	82.86%
0.10	0.1	5	83.46%
0.50	0.1	5	84.34%
0.90	0.1	5	83.45%
0.01	0.5	5	83.06%
0.10	0.5	5	83.06%
0.50	0.5	5	83.86%
0.90	0.5	5	82.76%
0.01	0.9	5	83.14%
0.10	0.9	5	83.83%
0.50	0.9	5	83.77%
0.90	0.9	5	83.73%
0.01	0.1	7	82.96%
0.10	0.1	7	83.53%
0.50	0.1	7	84.08%
0.90	0.1	7	83.89%
0.01	0.5	7	83.07%
0.10	0.5	7	83.36%
0.50	0.5	7	84.15%
0.90	0.5	7	81.50%
0.01	0.9	7	82.69%
0.10	0.9	7	83.98%
0.50	0.9	7	83.10%
0.90	0.9	7	84.31%
0.01	0.1	9	82.87%
0.10	0.1	9	83.43%
0.50	0.1	9	83.25%
0.90	0.1	9	83.77%
0.01	0.5	9	82.68%
0.10	0.5	9	83.93%
0.50	0.5	9	83.92%
0.90	0.5	9	84.28%
0.01	0.9	9	82.84%
0.10	0.9	9	83.37%
0.50	0.9	9	83.21%
0.90	0.9	9	82.61%

Table 1: LR tuning data from LR.log

So we fix initial LR = 0.5, LR factor = 0.1 and Patience = 5 for further tuning since it has highest accuracy.

2.2 L2 regularization tuning

we implemented weight decay and obtained following data:

Weight Decay Rate	Training Time (sec)	Accuracy
0.01	141.7	79.27%
0.1	168.0	71.24%
0.5	123.4	62.45%
0.9	130.4	62.03%

Table 2: Regularisation tuning data from weight_decay_rate.log

We observe that Weight decay rate = 0.01 gives highest accuracy, and weight decay rate = 0.5 gives fastest training time.So we fix Weight decay rate = 0.01 for further tuning.

2.3 Batch size selection

we varied the batch size and obtained following data:

Batch Size	Training Time (sec)	Accuracy
32	238.0	74.94%
64	150.3	63.84%
128	131.9	80.31%
256	64.8	66.17%

Table 3: Tuning data from batchSize.log

We observe that Batch size of 128 gives highest accuracy, and Batch size of 256 gives fastest training time. So we fix Batch size as 128 for further tuning.

2.4 Optimiser selection

we tried different optimisers and obtained following data:

Optimizer	Training Time (sec)	Accuracy
SGD	138.0	81.73%
Adam	102.9	57.48%
RMSprop	127.9	51.99%

Table 4: Tuning data from optimizer_name.log

We observe that the SGD optimizer gives highest accuracy, and the RMSprop optimizer gives fastest training time. So we select the SGD optimizer for final evaluation

2.5 Conclusion

- We finally obtained parameter values as :

Hyperparameter	Value
Initial Learning Rate	0.50
LR Factor	0.1
Patience	5
Weight Decay Rate	0.01
Batch Size	128
Optimizer	SGD

Table 5: Vertical Table

- The final accuracy is 81.73% and the final training time is 138.0 secs.
- We infer from the data that the choice of Optimizer had the most impact since an optimizer other than SGD caused significant decrease in accuracy.
- We notice that higher weight decay rate in L2 regularisation does not increase accuracy, and rather decreases it . This indicates absence of overfitting.
- Importance of tuning : Through this exercise in tuning the hyperparameters we found out how the values of hyperparameters are used to maximise model accuracy, achieve generalisation and minimise overfitting. Although our focus here was not to minimise resource consumption (like time and memory), that goal is also achievable during tuning though we lose some accuracy in the trade-off.
- Error and Learning rate plots:

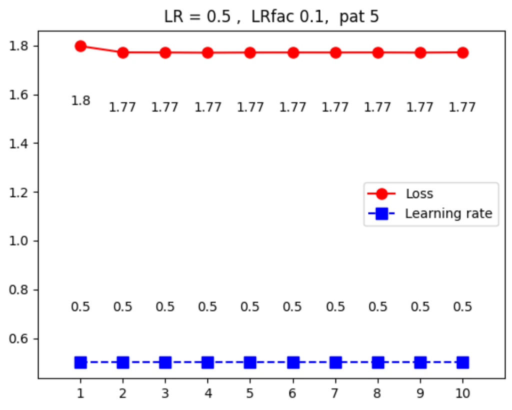

Figure 2: Plots of Error and Learning rate during training.

3 Linear Regression

3.1 Dataset Description

- X1: A numerical feature representing a sales-related metric.
- X2:A binary categorical feature (0 or 1) representing a certain retail promotion.
- X3: A categorical feature with three categories (A, B, or C) representing the store location.
- y:The target variable, representing the sales revenue (numerical).

3.2 Problem Description

Task is to build a linear regression model to predict the sales revenue based on the provided features. The goal is to understand how different factors, such as the sales-related metric, promotion, and store location, influence sales revenue

3.3 Data Preprocessing

In this preprocessing code, we start by loading a dataset from a CSV file. We then convert categorical data in the 'X3' column into numerical values using a predefined mapping. Next, we separate the target variable ('y') from the input features ('X'). To ensure compatibility with PyTorch, we convert the DataFrame into PyTorch tensors with double precision. Finally, we split the data into training and testing sets, with a 70-30 ratio, preparing it for model training and evaluation. This process involves data loading, encoding, type conversion, normalization (which is currently commented out), and dataset splitting to ensure it's ready for machine learning tasks.

3.4 Exploratory Data Analysis

3.4.1 Calculation of Correlation Coefficient

- Correlation Coefficient between X1 and y: 0.97
- Correlation Coefficient between X2 and y: 0.20
- Correlation Coefficient between X3 and y: 0.01

3.5 Feature selection

By analysing the correlation coefficient we can predict that target depends majorly on input X1 and has little dependence on X2 and almost no dependence on X3.So we would be analysing the MSE,MAE and R-squared for all the cases metioned in evaluation part:

3.6 Model

In our model building process, we have chosen a linear regression model with one output unit (nn.Linear(input_size, 1)), which is suited for regression tasks. We've used double precision for data handling. To optimize our model's performance and prevent overfitting, we've incorporated L2 regularization (weight_decay=l2_lambda) by adding a weight decay term to the stochastic gradient descent (SGD) optimizer. This regularization technique helps control the complexity of the model and enhances its generalization capabilities.

3.7 Evaluation

- Considering all inputs X1,X2,X3.
  - Mean Absolute Error: 1.503309898913856
  - Mean Squared Error: 3.619468594013942
  - R-squared: 0.9604338377368393
  - weights(w_1, w_2, w_3): [[3.03370084 2.03546008 0.06423518]]
  - bias(w_o): [-0.2653805]

Plot

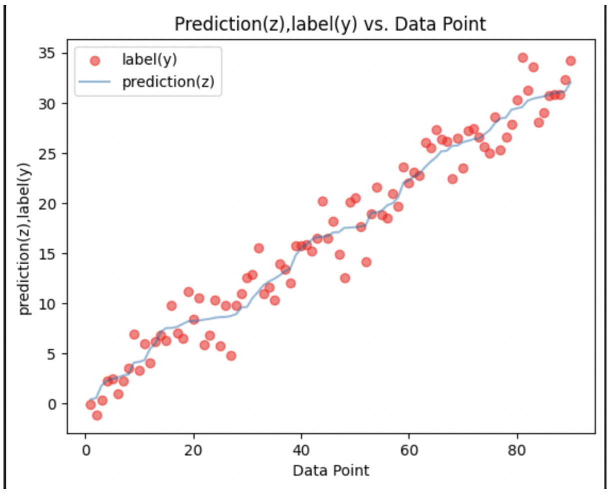

Figure 3: Plot by utilising only feature X1,X2,X3.Line plot is for prediction(z) and scatter plot is for label(y)

- X1 and X2
  - Mean Absolute Error: 1.5110948878061983
  - Mean Squared Error: 3.637135981691146
  - R-squared: 0.9602407069748379
  - weights(w_1, w_2): [[3.04144137 2.05359198]]
  - bias(w_o): [-0.1997372]
  - **Interpretation**: No impact of X3 since MSE, MAE, R-squared barely changes. So this shows X3 is not very relevant for the model.

Plot

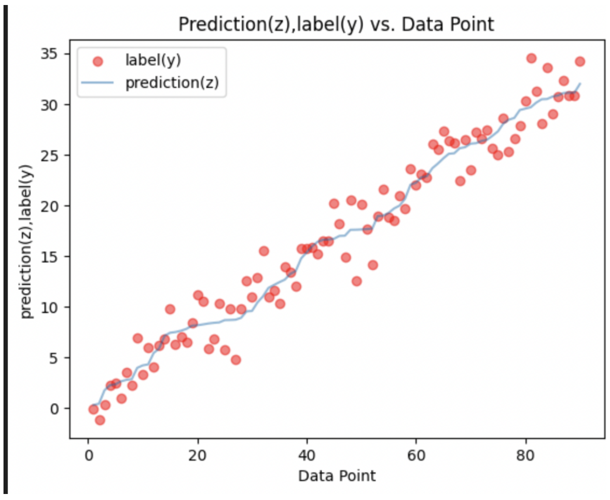

Figure 4: Plot by utilising only feature X1 and X2.Line plot is for prediction(z) and scatter plot is for label(y)

- X2 and X3
  - Mean Absolute Error: 8.403804828224809
  - Mean Squared Error: 92.6836900749908
  - R-squared: -0.013170255633788893
  - weights(w_2, w_3): [[4.15016945 0.77383881]]
  - bias(w_o): [12.15011626]
  - **Interpretation**: MSE,MAE are increased by a drastic amount on removing X1 and also R-squared decreases to a negative value which shows that prediction accuracy drops drastically by removing X1 and utlising only X2,X3 which shows that X1 is a relevant feature for the model and X2,X3 are not of much relevance to the model.

Plot

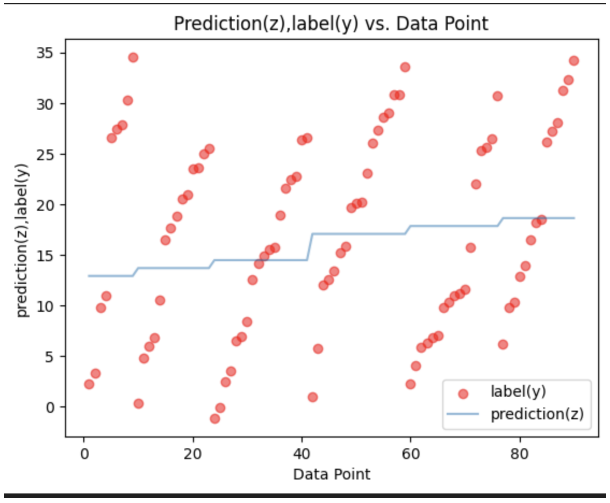

Figure 5: Plot by utilising only feature X2 and X3.Line plot is for prediction(z) and scatter plot is for label(y)

- X1
  - Mean Absolute Error: 1.737102831619866
  - Mean Squared Error: 4.6489699684482755
  - R-squared: 0.9491798601506316
  - weights(w_1): [[3.08141423]]
  - bias(w_o): [0.6259764]
  - **Interpretation** : MSE changes by some amount but not too much also R-squared and MAE almost remains the same so from this we can deduce that the X2,X3 are not very relevant for the model and X1 is a relevant feature for the model.

Plot

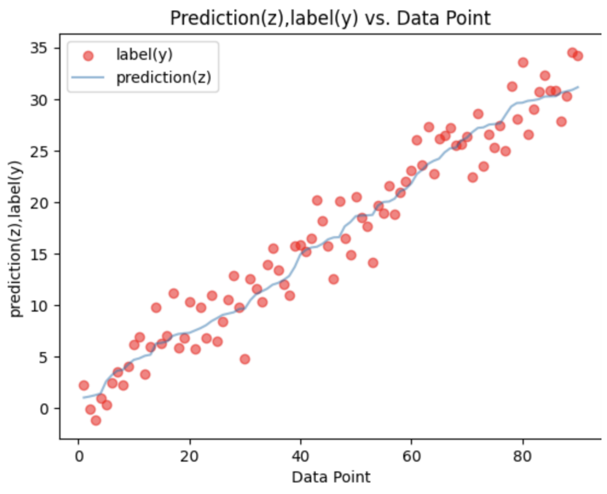

Figure 6: Plot by utilising only feature X1.Line plot is for prediction(z) and scatter plot is for label(y)

- X2
  - Mean Absolute Error: 8.355418976125321
  - Mean Squared Error: 90.82886054213607
  - R-squared: 0.007105783337458482
  - weights(w_2): [[3.81074856]]
  - bias(w_o): [13.93434314]
  - **Interpretation** : MSE,MAE are increased by a drastic amount on removing X1,X3 and also R-squared decreases to a very small value which shows that prediction accuracy drops drastically by removing X1,X3 which shows that X2 is not a relevant feature for the model.

Plot

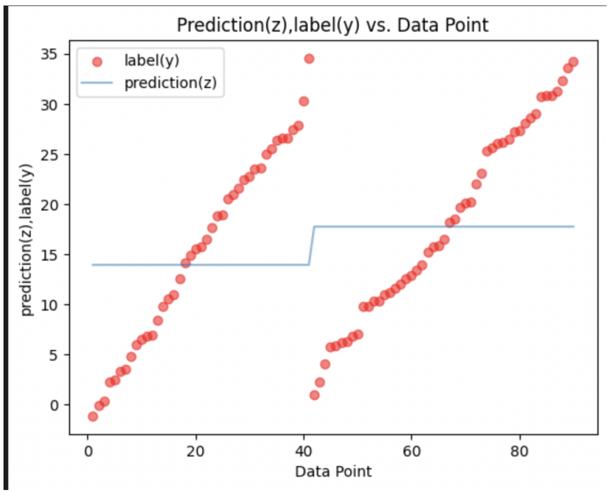

Figure 7: Plot by utilising only feature X2.Line plot is for prediction(z) and scatter plot is for label(y)

- X3
  - Mean Absolute Error: 8.341306538696177
  - Mean Squared Error: 94.41151884108365
  - R-squared: -0.03205798778188629
  - weights(w_3): [[0.72347634]]
  - bias(w_o): [14.33594914]
  - **Interpretation** : MSE,MAE are increased by a drastic amount on removing X1,X2 and also R-squared decreases to a very small negative value which shows that prediction accuracy drops drastically by removing X1,X2 which shows that X3 is not a relevant feature for the model.

Plot

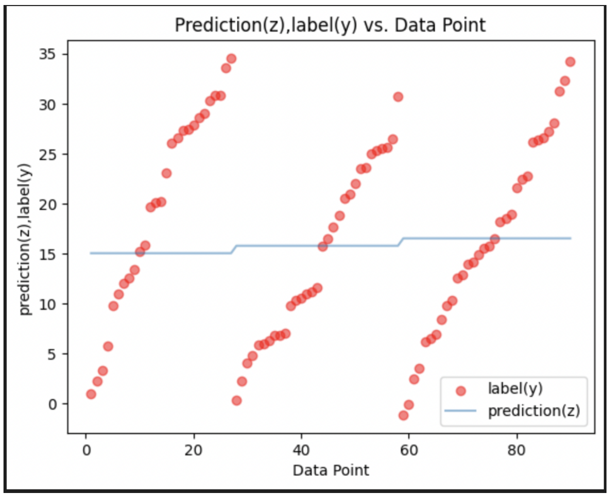

Figure 8: Plot by utilising only feature X3.Line plot is for prediction(z) and scatter plot is for label(y)

3.8 Predictions

3.8.1 Prediction with X1,X2,X3

Now using our model we can do predictions on new data and compare prediction of our model with the target.Some predictions are listed below:

- Prediction1:
  - Inputs: X1 = 6.6308,X2 = 0.0,X3 = 2.0
  - target: 20.9036
  - prediction: 19.9622
- Prediction2:
  - Inputs: X1 = 7.5838,X2 = 0.0,X3 = 3.0
  - target: 22.7401
  - prediction: 22.9605
- Prediction3:
  - Inputs: X1 = 3.8346,X2 = 1.0,X3 = 1.0
  - target: 12.0523
  - prediction: 13.4305

As we can clearly observe that model is able to closely predict sales revenue for new data as its a linear regression model so there is some difference in predicted and actual value as the model deals with real values.

3.8.2 Prediction with only X1

Now using our model we can do predictions on new data and compare prediction of our model with the target.Some predictions are listed below:

- Prediction1:
  - Inputs: X1 = 6.6308
  - target: 20.9036
  - prediction: 21.0582
- Prediction2:
  - Inputs: X1 = 7.5838
  - target: 22.7401
  - prediction: 23.9953
- Prediction3:
  - Inputs: X1 = 3.8346
  - target: 12.0523
  - prediction: 12.4407

As we can clearly observe that model is able to closely predict sales revenue for new data as its a linear regression model so there is some difference in predicted and actual value as the model deals with real values.