Implementing Backpropagation

Data Preprocessing

Given data contained a total of 17 features. Most of which are very unrelated to the final label, the ‘Outcome1’ column. So, feature selection is very predominant during the preprocessing of the dataset.

In the train dataset, three columns are dropped as per feature selection part, ‘Row_ID’, ‘Outbreak_Related’, ‘Reporting_PHU_ID’. Of these, ‘Row_ID’ and ‘Reporting_PHU_ID’ are just the id’s corresponding to row and PHU. Having these in the data is irrelevant as these in no way affect the final label(‘Reporting_PHU_ID’ affects the output but, there is already ‘Reporting_PHU’ which has the same relation as ‘Reporting_PHU_ID’). And in the feature ‘Outbreak_Related’ almost 181361 entries are missing. There are two alternatives, one to extrapolate the data to fill the missing values or to just drop the column as there are too many missing values, and filling them all could affect the performance. The latter solution is preferred by me. There are also columns with fewer missing values, like ‘Test_Reported_Date’ with 7219 missing entries and ‘Specimen_Date’ with 1273 missing entries. For these columns, the value ‘Ambig’ is used in place of missing entries. This way when converting the data into categorical, the missing values will be assigned to a category, keeping the integrity of the data intact.

In the test dataset, two columns are dropped, ‘Outbreak_Related’ and ‘Reporting_PHU_ID’ because ‘Row_ID’ is required while making predictions. There are no missing values in the test dataset, so it is left unchanged further.

After the cleaning of datasets, they are converted into categorical form. Many of the entries of the train and test dataset are strings. So, they are all converted into integers which represent the class to which the string belongs to. In the train dataset, ‘Outcome1’ column is converted into categorical with three classes 0, 1 and 2. 0 is Resolved class, 1 is Fatal class and 2 is Non Resolved.

The train dataset is now split into train, val portions, for training and validation. 0.2% of the train dataset is used for validation and 0.8% of the train dataset is used for training.

The train dataset and test dataset is then normalized, except the final label, ‘Outcome1’ column of the train dataset.

The ‘Outcome1’ column of the train dataset is then converted to the one hot form.

Model Architecture
All the layers of the model are fully connected layers. First input layer has 14 nodes, because the input has 14 features. The first hidden layer has 9 nodes, and the second hidden layer has 5 hidden nodes. Which are hand picked based on random search in the parameter space. Output layer has 3 nodes, to accustom for the one hot output corresponding to the three output classes, 0, 1 and 2.

Loss function
The loss function used is mean squared error. Which is the squared mean of deviations of predicted values from true values

Optimizer
The optimizer used is simple gradient descent. Weights of the neural network are updated for each sample point. Each weight is decreased by the amount of learning rate times the gradient of that weight with respect to the loss function.

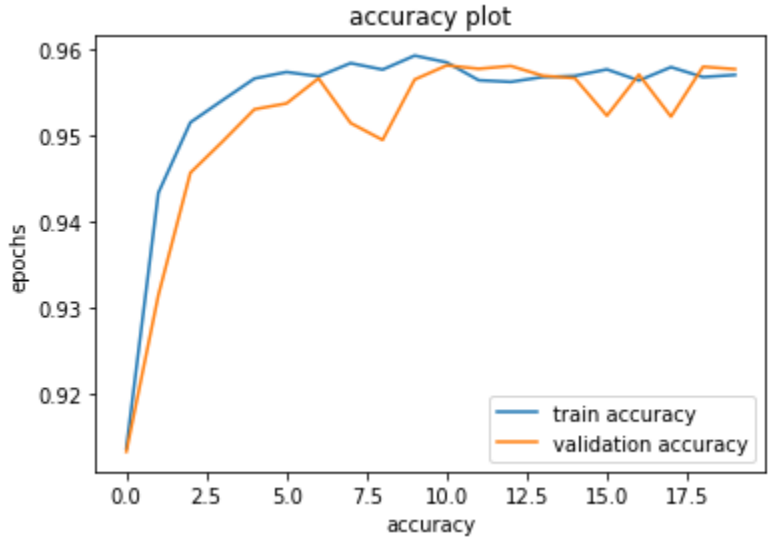
Hyperparamters
Learning rate: 0.09
Epochs: 20
Layers: [14: input, 9: hidden1, 5: hidden2, 3: output]
Activations: [tanh, tanh, tanh, softmax]

Evaluating Model Performance
The model performance is evaluated using the validation split. The metrics used are the loss value, classification accuracy and f1 score.

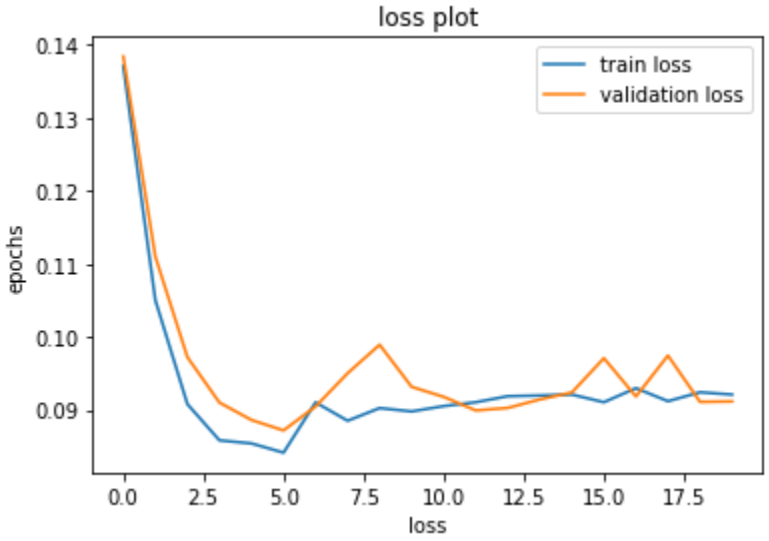
Epoch	Train Loss	Train Accuracy	Train F1 Score	Validation Loss	Validation Accuracy	Validation F1 Score
0	0.1371	0.9136	0.9037	0.1384	0.9132	0.9043
1	0.1050	0.9433	0.9318	0.1110	0.9314	0.9182
2	0.0908	0.9514	0.9420	0.0972	0.9456	0.9360
3	0.0858	0.9540	0.9443	0.0910	0.9492	0.9394
4	0.0854	0.9565	0.9465	0.0886	0.9529	0.9425
5	0.0841	0.9573	0.9470	0.0872	0.9536	0.9428
6	0.0910	0.9568	0.9468	0.0904	0.9565	0.9463
7	0.0885	0.9583	0.9479	0.0950	0.9514	0.9382

8	0.0902	0.9576	0.9471	0.0989	0.9494	0.9382
9	0.0898	0.9592	0.9489	0.0931	0.9564	0.9456
10	0.0905	0.9584	0.9483	0.0918	0.9580	0.9476
11	0.0910	0.9563	0.9465	0.0899	0.9576	0.9475
12	0.0919	0.9561	0.9463	0.0902	0.9580	0.9478
13	0.0920	0.9567	0.9468	0.0914	0.9568	0.9466
14	0.0921	0.9568	0.9468	0.0924	0.9566	0.9462
15	0.0910	0.9576	0.9473	0.0971	0.9522	0.9415
16	0.0930	0.9563	0.9464	0.0918	0.9570	0.9468
17	0.0912	0.9578	0.9476	0.0974	0.9521	0.9413
18	0.0924	0.9567	0.9468	0.0911	0.9579	0.9477
19	0.0921	0.9569	0.9470	0.0911	0.9576	0.9474

Accuracy plot



Loss plot



Observations

The loss decreased but the loss for the final epoch is not the best loss.
On the other hand the accuracy increased for the majority of the time.
Main target is the validation accuracy and validation f1 score.
Final validation accuracy is 95.76% and validation f1 score is 0.9474.
The curve is a bit shaky, maybe having a lower learning rate and training for more epochs could turn out to be fruitful. But due to time constraint couldn't do it.