Student Number: 15031625
Assignment 2
Report

Due to some unfortunate time constraints, the IEEE report template could not be followed. The report below includes the Data Preparation, Hyper-Parameter Optimisation and Active Learning sections. Each section includes the background, experimentation-setup, results and discussion of the results relevant to those sections.

Data Preparation

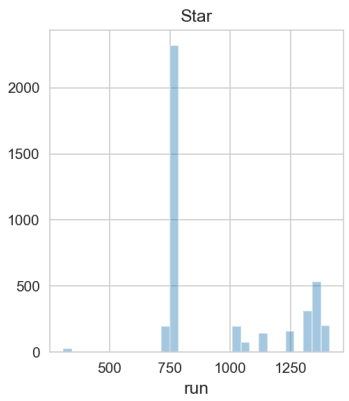
The data file is named “Skyserver_SQL2_27_2018 6_51_39 PM.csv”. Before pre-processing, it has a total of 18 fields. The data file was created by using an SQL table to join two separate tables, namely PhotoObj and SpecObj tables. These tables contain the image object data and the corresponding the spectral data respectively. The fields originating from the image object data are objid (object identifier); ra (J2000 Right Ascension r-band) and dec (J2000 Declination r-band) astronomical coordinates in the equatorial coordinate system; u, g, r, i and z band responses from the telescope; run (number), re-run (number), camcol(column number) and field (number). The fields originating from the spectral data are specobjid (object identifier); class (galaxy, star or quasar object class); redshift (Final Redshift), plate (plate Number), mjd (Modified Julian Date of the observation) and fiberid (fiberID).

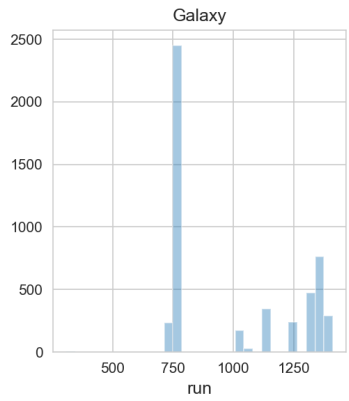
Every observation had a unique objid and specobjid. This indicates that these two fields were the primary keys of their respective tables. Both field columns have been removed from the data. They do not hold any information which can be used either alone or in combination with other fields to help during a classification task. Hence, unique identifiers should not be used during learning because their purpose is to assist when we query the data.

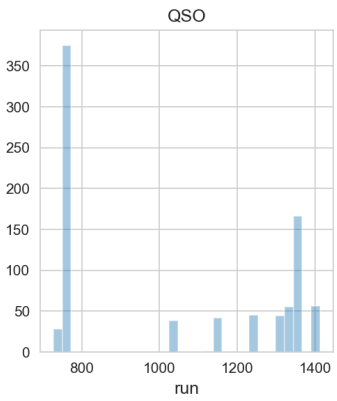
There is only one categorical column in our dataset, namely the class column. This is the target column that the classification task is attempting to predict, given the other feature columns for a particular observation. The feature columns are all numerical. The supervised neural network model that will be trained works with numerical data. For this reason, class column data values have been one-hot encoded. This implies that each class data-value would be a 3-bit string, with the bit index corresponding to the particular class being set to 1, while the other two bits set to 0.

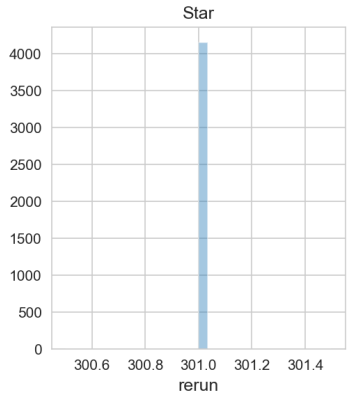
The neural network model does not make any assumptions with regard to how it deals with missing values. The above, partially pre-processed dataset has to be examined for missing values, after which appropriate measures need to be taken to rectify the incomplete observations. In the case of the data at hand, no action needs to be taken because there are no missing values for any of the fields.

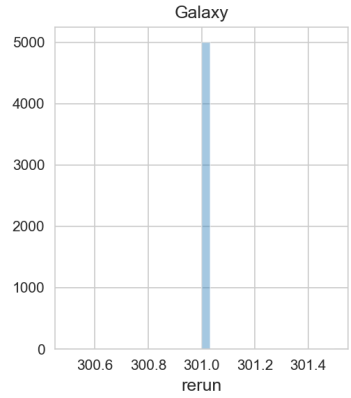
It is necessary to understand what each of the features mean in the specific problem domain in order gauge their importance. The neural network model will learn most efficiently when the features it makes use of play an active role with regard to which target class an observation can be given. The features run, rerun, camcol and field are described (in the Sloan Website) as being specific to the camera at point of imaging. This hints to the fact that their values play no significance in altering the values of the target class of an observation. This hypothesis needs to be confirmed analytically and univariate analysis can help in this regard.

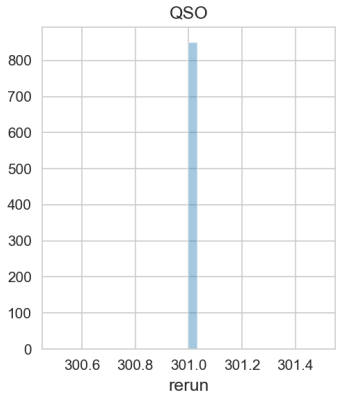

Star
2000
1500
1000
500
0
500
750
1000
1250
run

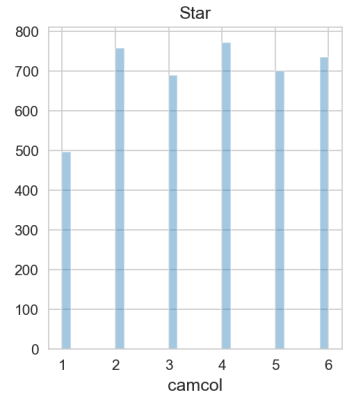

Galaxy
2500
2000
1500
1000
500
0
500
750
1000
1250
run

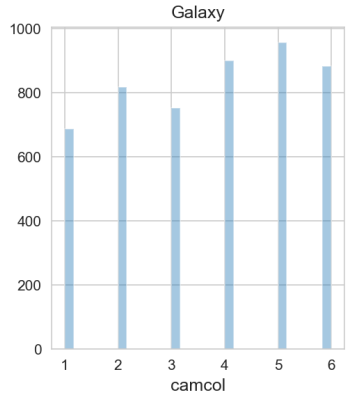

QSO
350
300
250
200
150
100
50
0
800
1000
1200
1400
run

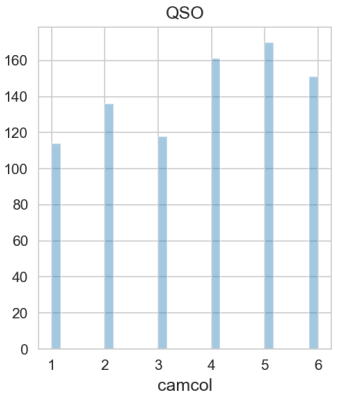

Star
4000
3500
3000
2500
2000
1500
1000
500
0
300.6
300.8
301.0
301.2
301.4
rerun

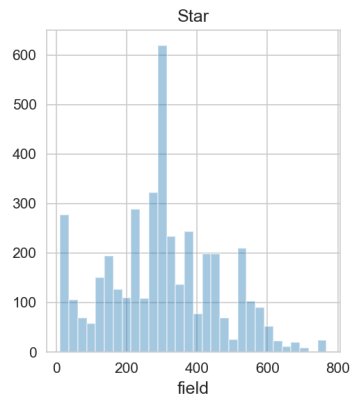

Galaxy
5000
4000
3000
2000
1000
0
300.6
300.8
301.0
301.2
301.4
rerun

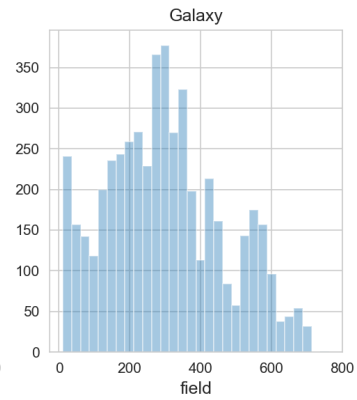

QSO
800
700
600
500
400
300
200
100
0
300.6
300.8
301.0
301.2
301.4
rerun

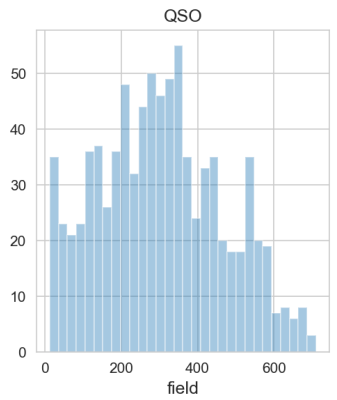

Star
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
camcol


Galaxy
1000
800
600
400
200
0
1
2
3
4
5
6
camcol


QSO
160
140
120
100
80
60
40
20
0
1
2
3
4
5
6
camcol


Star
600
500
400
300
200
100
0
0
200
400
600
800
field


Galaxy
350
300
250
200
150
100
50
0
0
200
400
600
800
field


QSO
50
40
30
20
10
0
0
200
400
600
field

The above figures are histograms for run, rerun, camcol and field feature columns, for each of the classes. It indicates how the class values for the respective columns are distributed over the range of that feature column. Rerun has the same value of 301 for every data observation as can be seen by the single bar for each of the classes. The frequency of each of the single bars' for the classes is exactly the frequency of the particular class in all the data observations. It therefore makes no sense to include this feature, and it is imputed. The run, camcol and field plots have almost the same distributions for each class. This indicates the specific classes have almost no influence with regard to how the values for the feature columns are distributed. The shape of the histogram irrespective of the class, would be the same as the shape of the histogram specific to a class. In classification machine learning, if we are aware of the distribution playing no role across classes, we are merely increasing the learning time and search space difficulty by including these features. The network would waste time trying to learn how the value of these features maps to specific classes. It might eventually learn the fact that there is no relationship, but this can be avoided in the first place.

The ranges of each of the features were clearly different. In order for the neural network to learn efficiently, feature scaling was performed. Two feature scaling approaches were tested namely, Min-Max Scaling and Standardisation. The training and generalisation errors, and classification accuracy averages were determined using the different schemes for a maximum of 100 epochs and 15 runs using random hyper-parameter choices, before hyper-parameter optimisation.

Averages	**Training Error**	**Generalisation Error**	**Classification Accuracy**
Min-Max Scaling	0.550966970191702	0.6084926029046377	0.8900000055631002
Standardisation	0.22191876133505625	0.2245498994986216	0.9751666585604349

The results indicate that learning speeds up when we use the standardisation technique. Learning can be described as a decrease in the average training error that leads to a corresponding decrease in the average generalisation error and increase in the average classification accuracy, after a certain period of training time. Z-scores are not forced to lie in the range (0,1) and the increased sparsity among the values for each feature column ensures that they are a lot more representative of the original input distribution. The hyperparameter optimisation process that follows will make use of the standardisation technique.

Hyper-parameter Optimisation

Data-Set Split

The dataset contains 10000 patterns. The patterns were split into a training, generalisation and test sets before the hyper-parameter optimisation process. 20% of the dataset is allocated to the test set. 20% of the remaining 80% of the dataset is allocated to the generalisation set and the rest is part of the training set. Mini-batch training has been used due to its increasing popularity in the research community. Mini-batches are built from the training set, which are used to train the neural network. The average of all mini-batch errors for a particular epoch gives the training error for that epoch. The average training error over all epochs represents the training error for the particular run.

The generalisation set is used to calculate the generalisation error at the end of every epoch. The average generalisation error over all epochs represents the generalisation error for the particular run. The test set is used to determine the classification accuracy of the trained model for the particular run, at the end of all the epochs.

In order to compare stable figures, averages and standard deviations are calculated over 20 runs. A different random number generator seed is used for each run, where the seed value is equal to the run count. The hyper-parameter tuning approach involves tuning a single parameter at a time; establishing the best parameter value for that parameter and tuning other parameters using the values that were tuned previously. The tuning process begins by tuning the most important hyper-parameters first (those that have highest impact on the generalisation ability of the network) and then moving on to less important hyper-parameters, using constant values for parameters that have been tuned and those that are still to be tuned. This approach ensures a network that can produce highest classification accuracy.

Number of Hidden Neurons

```
Initial Constant Hyper Parameter Values
learning_rate = 0.5
epochs = 100
batch_size = 16
hidden_layers = 1
initial_weight_standard_deviations = 0.03
activation function = relu
no regularization
cross entropy loss function
```

Averages	Training Error	Generalisation Error	Classification Accuracy
1 Neuron	1.0645232193570897	1.071277345220248	0.753
5 Neurons	0.3836956901809962	0.4485129797955354	0.9636666
10 Neurons	0.20841753246437442	0.2511313944806655	0.989
20 Neurons	0.19795355495894473	0.25188159043590225	0.98550004
50 Neurons	0.19151666754966243	0.2701160665353139	0.98

Standard Deviation	Training Error	Generalisation Error	Classification Accuracy
1 Neuron	0.44750609420627835	0.43318064692415725	0.18198398
5 Neurons	0.1954208153907608	0.2647182752754634	0.022628652
10 Neurons	0.01531053041088829	0.05444839810736198	0.0043204934
20 Neurons	0.00255152384021599	0.02104664010603016	0.003937004
50 Neurons	0.00134537883878565	0.01406209132905883	0.007071061

Occam’s Razor principle states that the simplest architecture is the best architecture. In order to test this principle, we started with an architecture that has only a single neuron. The training error averages and standard deviations decrease as we increase the number of neurons from one to 50. However, the average generalisation errors do not decrease when the number of neurons are increased to 20 and 50 respectively. The same can be seen for the classification accuracy. This indicates that if we add more than 10 neurons, we are slowly starting to overfit the data. 10 neurons seem have enough representational power to generalise to newer astronomical observations.

Number of Hidden Layers

It has been determined that 10 neurons should be enough to represent the search space for the astronomical observation classification task. We might potentially improve upon the Generalisation and Classification Accuracy if we split these neurons between more than one hidden layer. To test this hypothesis, we keep the first hidden layer to be 6 neurons, the second hidden layer to be 4 neurons, while the output layer is fixed at 3 neurons due to having 3 target classes. This makes a pyramidal architecture, which have been shown to perform well in practice. The results are as follows:

	Training Error	**Generalisation Error**	**Classification Accuracy**
Average	1.3190026302069582	1.335858381986618	0.6315
Standard Deviation	0.5050179938103797	0.48918504021354214	0.19162592

The addition of the extra hidden layer has impacted negatively on the results. From this, it is clear that the astronomical observation classification task is not high in complexity. More hidden layers effectively perform the “feature engineering step”. The outputs of neurons in hidden layers lower down the architecture create new higher level feature representations. This task does not require the architecture to go deeper. Recombinations of the raw, initial features themselves are enough for representation of the problem. This ought to be because there are image object features and spectral features which cannot be interpreted together with one another. E.g. The image bands and the redshift values cannot possibly be combined to produce a new feature. It makes more sense to interpret the features as they are. The negative results do not lend to experimentation with more hidden layers.

Activation Function

Sigmoid Activation Function – 10 Neurons	**Training Error**	**Generalisation Error**	**Classification Accuracy**
Average	0.12887430902690683	0.15756216863791148	0.9816666
Standard Deviation	0.00186938815936784	0.00403613705556832	0.007919738

Sigmoid Activation Function – 5 Neurons	**Training Error**	**Generalisation Error**	**Classification Accuracy**
Average	0.1396618326250144	0.15116019507249198	0.98849994
Standard Deviation	0.00369058974911922	0.00662611629171531	0.0010801279

The relu activation function introduces sparse activations to remedy the vanishing gradient problem associated with the sigmoid activation function (particularly for deep neural networks). But since the architecture contains a single layer with very few neurons, we are unlikely to have a vanishing gradient problem. The sigmoid function was tested, with half the number of neurons. This was done to confirm the hypotheses that those half number of neurons (5) would have probably not fired anyway in the 10 neuron network that used relu due to the property of sparse activations. The hypotheses was confirmed as the 5-neuron network was able to decrease the generalisation error and increase the classification accuracy.

Learning Rate

The learning rate could decrease by a constant factor (0.99) in each iteration. We would start with a high learning rate in the beginning (0.9) to promote exploration of the search space and slowly decrease the learning rate so that we could exploit the search space which has been refined by the backpropagation steps. The results are as follows:

Learning Rate Decay	Training Error	Generalisation Error	Classification Accuracy
Average	0.14503373104875594	0.16011788268884022	0.98333335
Standard Deviation	0.00202586129592116	0.00410982388557100	0.0024944327

Learning Rate Decay performed similarly to having a constant learning rate of 0.5. This indicates that learning rate decay over all the epochs, equates to learning about half of what we could at each epoch. The constant learning rate handles the exploration-exploitation tradeoff as the number of epochs are increased.

Regularisation

Weight Decay Regularization lambda=0.000001	Training Error	Generalisation Error	Classification Accuracy
Average	0.14150011909233295	0.15866469231744607	0.9865
Standard Deviation	0.00422648775818485	0.00963460970541959	0.0024832815

Dropout Regularization, keepProb = 0.9	Training Error	Generalisation Error	Classification Accuracy
Average	0.2522089564908529	0.25825936267773314	0.9608334
Standard Deviation	0.00730060780161488	0.00907391646028897	0.008409657

Both regularisation schemes were unable to improve performance significantly. This is due to the fact that the network was reasonably small in size ((5 hidden neurons * 12 features) + (3 output neurons * 5 hidden neurons) + 5 hidden layer bias + 3 output layer bias = 83). The lower the number of parameters to train, the lower the chance to overfit. Weight Decay regularization attempts to punish large weights because if there are a large number of such weights, the neural network may overfit. Since there are only few weights, the effect of weight decay does not bring any significant improvement. The value of lambda is kept low at 0.000001 because there is very little overfitting. Increasing the value of lambda impacts the neural network negatively.

However, weight decay does better than dropout regularisation. This is because dropout is too strict for this very minimal architecture. Even with the high keep probability of 0.9, losing a single node is detrimental to performance because the 5 layer architecture has been determined to be optimal in previous steps. Hence, no form of regularization is included in the final architecture.

Batch Size

Different batch sizes were tested. The results below indicate that a batch size of 64 produced the best results i.e. smallest standard deviations and best classification accuracy.

Averages	Training Error	Generalisation Error	Classification Accuracy
Batch Size = 8	0.14362831333074136	0.1616626115143299	0.9866667
Batch Size = 32	0.14837815797135698	0.16262698277831078	0.98
Batch Size = 64	0.16862179207404457	0.17785837997992834	0.9871667
Batch Size = 128	0.20699326867175594	0.21945257782936098	0.9828334

Standard Deviation	Training Error	Generalisation Error	Classification Accuracy
Batch Size = 8	0.00194515735061423	0.01037855128266098	0.0016499186
Batch Size = 32	0.00464278153237075	0.01105183437475441	0.010677097
Batch Size = 64	0.00448713348992250	0.00760441377887312	0.0009427969
Batch Size = 128	0.00453830534724460	0.01074109981482905	0.0034721026

Active Learning

Incremental Learning

Initially, a random 1% of the candidate training set is initialised as the initial actual set that is used for training the neural network. These patterns are removed from the candidate training set. Incrementally, patterns are added from the candidate training set to the actual training set. Patterns are added one at time, only when a specific criterion is met. The criterion used in this research is that a pattern is to be added, if and only if, the neural network does not generalise well. It is required to keep track of a generalisation factor every epoch in order to see whether generalisation is taking place. The generalisation factor is the ratio of the training error to the generalisation error for the particular epoch. The generalisation factor should not exceed a threshold value, in order to ascertain a particular level of generalisation. The research tested two different generalisation factors. If it is determined that the generalisation factor exceeds the threshold value, a new training pattern (the pattern that would incur largest error when fed forward through the network) from the candidate set is removed from that set and added to the actual training set to be used in the next epoch. The results are as follows:

Average Criterions	Training Error	Generalisation Error	Classification Accuracy
Generalisation Factor > 1.1	0.2354523851228024	0.3223134085536003	0.9756667
Generalisation Factor > 1.3	0.2584116874685336	0.31388797983527184	0.9773333

Standard Deviations Criterion	Training Error	Generalisation Error	Classification Accuracy
Generalisation Factor > 1.1	0.01591639778199914	0.08361188914245542	0.006859707
Generalisation Factor > 1.3	0.02551272008686725	0.06874139278376223	0.0027182563

Selective Learning

The selective learning approach in this research was applied every 10 epochs. This implies that every 10 epochs we train with a subset of patterns from the set of training patterns. Every subset may include any of the patterns that were present in previous subsets and the all the patterns in the training set are eligible for selection. In this research, a new selective learning approach is proposed. At each selection point, the cross-entropy errors of a feedforward pass of each of the training patterns in the training set is determined. The training patterns in the training set are sorted in ascending order of cross-entropy errors. The sorted training patterns are split into 3 i.e split of 1/3. The bottom 1/3 (least errors) and the top 1/3 (highest errors) are combined to produce the subset that is selected. This is based on the hypotheses that the network will learn more from patterns that produce large errors. The patterns that produce low errors are also included because these patterns are in direct conflict with those patterns that produce large errors. Training on these opposing patterns will cause the model to learn aggressively over the time period of 10 epochs. The middle 1/3 patterns are not included because those patterns would reduce the information gain by equally undoing the effects of the high error and low error patterns. The results are as follows:

	Training Error	Generalisation Error	Classification Accuracy
Average	0.29350776316842636	0.28467249423265456	0.9831667
Standard Deviation	0.01772012632316903	0.02717000648163884	0.00047139844

Comparison of the Chosen Selective and Incremental Learning Approaches

Neither the selective learning, nor the incremental learning approach was able to improve on the performance of the optimised active learning neural network model of this research. The training data did not play a crucial role in producing a model that generalises best. Passive learning was reasonable enough for the problem at hand. This may be because there were not many outliers in the available data – most data patterns were quite similar to one another and it does not matter what order we present the data to the neural network. However, the results do indicate that the particular selective learning approach that was used, was better than the incremental learning approach. This is because incremental learning had the extra generalisation factor threshold parameter to tune. It may also be because in incremental learning, the patterns are not removed from the actual training set – we merely add patterns on top of them. Some of those patterns already present in the training set may no longer be needed, but they are not removed and could be negatively impacting the performance. Selective learning is a lot more “situationally aware” since subsequent subsets do not build up on one another. Incremental Learning is nothing more than passive learning over a lesser number steps, determined by the generalisation factor.

Conclusion

The astronomical observation classification task was successful, producing a classification accuracy of 98.71667%. It is evident that input pre-processing and hyper-parameter optimisation are crucial in increasing the performance of the model. The task was simple enough not to require usage of any active learning approach.