COMP2261 ARTIFICIAL INTELLIGENCE

TECHNICAL REPORT

Patrick Q. Quilty
thsv47

January 23, 2025

2.1 Comprehensive Dataset Exploration

2.1.1 Know

Through a preliminary exploration of the dataset the following can be found:

Classification	Samples	Classes	Size
Seen	16540	1654	10
Unseen	16000	200	80

Modality	Features	Used	Unused
Brain	1700	561	1139
Text	512	512	0
Image	1000	100	900

All data belongs to 1 of 2 classifications, seen and unseen. The seen classification has 1654 unique classes, each appearing 10 times within the dataset, giving a total of 16540 samples. The unseen classification has 200 unique classes, each with 80 occurrences, giving 16000 samples.

There are 3 different modalities of data in the dataset. Brain activity data consists of a total of 1700 distinct features, measuring brain EEG readings, however, the vast majority of these features are disregarded, as in the development of out model we are only considering readings that occurred between 70 and 400ms of the subject being shown the image. The text features are obtained from descriptions associated with the images shown to the subject, and all 512 of these features are considered. And the finally image features are derived from the images shown to the subject, only 100 of these features are used.

2.1.2 See

The data in the above section can be supported by the following figures:

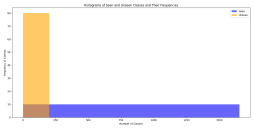

Figure 1: Number of Seen & Unseen Classes

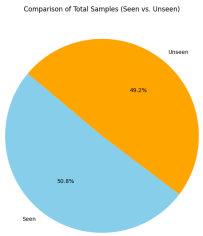

Figure 2: Total makeup of Samples

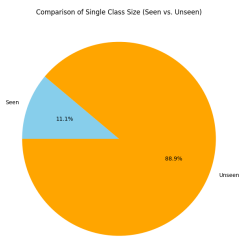

Figure 3: Comparison of Class sizes

2.1.3 Find

While the total number of samples in both seen and unseen is balanced, the number of samples per class is significantly different, with there being roughly 8 times the amount of classes in seen versus unseen, meaning that there are far more

samples per unseen class. This can lead to a bias towards unseen classes during model training, a model may develop a bias towards unseen classes, as they have a far higher per-class frequency distribution, as classes with more data contribute more to the loss function in training.

2.2 Custom Model Implementation

2.2.1 Implement

Due to the nature of the dataset, I chose to implement a logistic regression model, as it has no inherent bias towards majority classes, as the error is minimised across all of the features.

Logistic regression is a statistical model that can be used in binary classification. It consists of two classes, one is designated as the positive class 1, the other the negative class 0. The model outputs 1 or 0 as its prediction for whether a sample belongs to the positive or negative class, given its features.

In order to be adapted to allow for multi-class classification, a one-versus-rest approach is taken. In this approach separate logistic regression classifiers are trained for each class, assigning that class as the positive class and all other classes together as the negative. These individual models are then adapted to return the probability of belonging to their positive class. The class that has the highest probability is then chosen as the prediction.

The model is trained using gradient descent, an optimisation algorithm that minimises errors between the predicted results and their actual values through a loss function. It iteratively adjusts the model's weights to improve accuracy.

The first step in the training is to compute a weighted sum z of the features X by taking the dot product of their weights:

$$z = X \cdot \text{weights} \tag{1}$$

The weighted sum z is then passed into the sigmoid function in order to convert it into probabilities σ:

$$\text{predictions} = \sigma = \frac{1}{1 + e^{-z}} \tag{2}$$

The gradient is then calculated, giving the magnitude and direction of the weight;s adjustments. Here, m is the number of training samples, σ is the predictions and y is the real values.

$$\text{gradient} = \frac{1}{m} X \cdot (\sigma - y) \tag{3}$$

The weights are then updated using a learning rate and the gradient:

$$\text{weights} = \text{weights} - \text{learning rate} \cdot \text{gradient} \tag{4}$$

After all iterations have been completed, the optimised weights are stored, to be used in predictions.

To do a prediction, an array of features in passed into the model, using the weights calculated in the training of the model, a weighted sum is calculated:

$$z = X \cdot \text{weights} \tag{5}$$

Then, z is converted into a probability by using the sigmoid function, giving a probability between 0 and 1:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

The class with the highest probability of occurring is then chosen.

2.2.2 Compare

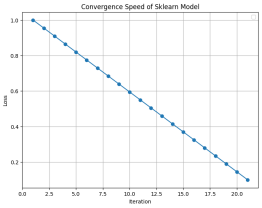

Figure 4: Sklearn Model

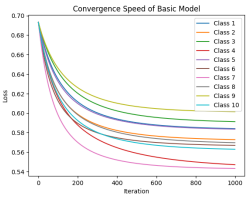

Figure 5: Basic Model

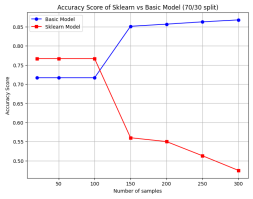

Figure 6: Sklearn vs Basic

Figure 4 shows the convergence speed of the sklearn logistic regression model. The sklearn model demonstrates a speed of convergence that is linear, steadily approaching 0, and achieves this in 21 iterations on a sample size of 10 classes. Whereas, the basic model has multiple convergence speeds, one for each class, following a logarithmic curve, this is shown in figure 5. In figure 6, it can be seen that the basic model's accuracy, although initially worse than that off the sklearn model, improves as the number of samples used increases, whereas, the sklearn model's accuracy decreases.

2.2.3 Improve

To improve the model, the sigmoid function used during training and validation is replaced with the softmax function. Softmax generates probability distributions across multiple classes, making it better for multi-class classification when compared with the sigmoid function. This significantly improves the both speed at which the model be trained and make predictions, as a separate model does not need to be trained for each class. Because of this, the model can achieve higher accuracy when trained for the same amount of time, as it can complete more training iterations, leading to better weight optimisation.

2.3 Results and Visualisation

2.3.1 Performance

Model	Accuracy	Precision	Recall	F1-score
Baseline	0.7667	0.7667	0.7667	0.7479
Basic	0.7167	0.7923	0.7167	0.7006
Improved	0.8334	0.8658	0.8334	0.8305

The baseline model has balanced performance with identical values for accuracy, precision and recall. But it has a slightly lowered F1-score showing the harmonic mean of precision and recall. This model is the benchmark for comparison.

The basic model has a decline in accuracy from the baseline, showing it got less correct predictions. But, its precision was improved meaning that this model is better then the baseline at identifying true positives relative to false positives. The model's recall stays the same as its accuracy meaning that it retrieves the same proportion of true positives. Its F1-score is lower, showing a relationship between gains in precision and loss in recall.

The improved model has the highest scores in all 4 categories out of the 3 models, significantly outperforming the baseline in all metrics. The increase in accuracy shows that the model makes more correct guesses. The improved precision shows that the model is better at identifying true positives with fewer false positives. The F1-score also improves suggesting that a good balance between precision and recall exists.

2.3.2 Visualisation

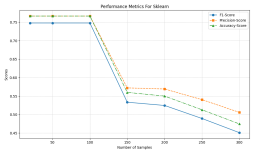

Figure 7: Sklearn Model

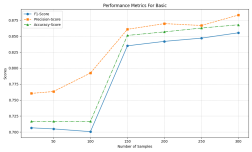

Figure 8: Basic Model

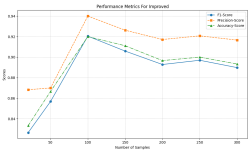

Figure 9: Improved Model

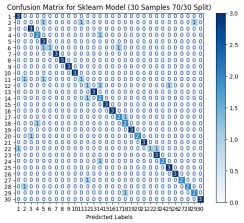

Figure 10: Sklearn Model

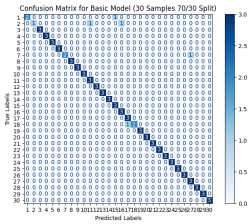

Figure 11: Basic Model

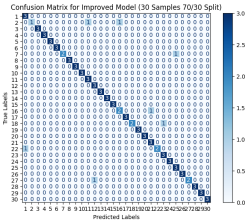

Figure 12: Improved Model

2.3.3 Ablation

The problem found in section 2.1, was that the dataset was imbalanced, which can cause biases in model predictions towards the majority class (any class in the unseen classification). However, the choice of model, logistic regression, does overcome this issue. This is because, in training logistic regression, gradient descent is used to attempt to find optimal weights for each feature in the training set, by using a loss function, this naturally accounts for imbalances as, no matter what, optimised weights are found. But, for very unbalanced datasets, it may be more effective to use extra steps such as data resampling or a weighted loss function.

2.4 Paradigm

2.4.1 Paradigm

A better data splitting technique then just picking an arbitrary index to split categories at is data stratification. Data stratification gives a more representative splitting of data, which can lead to better performance and more accurate predictions. It does this by preserving class distribution between the training and validation sets, meaning that there is less likely to be biased training and validation. The result of this is more accuracy when using small datasets, as arbitrarily splitting data is more likely to create subsets that do not correctly represent the overall dataset.

2.4.2 Adjustment

To adjust the model to use stratified data splitting, you need to ensure that the distribution of classes is preserved in both the training and validation set. This is done by first analysing the dataset to find out the proportion of each class label, the dataset is then turned into subsets for each class, from these subsets a proportional number of samples is selected randomly for both the training and validation set, making sure that the distribution remains the same.

2.4.3 Reflection

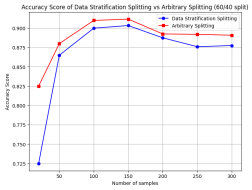

Figure 13: 60/40 split

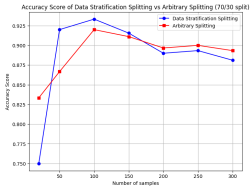

Figure 14: 70/30 split

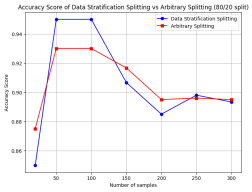

Figure 15: 80/20 split

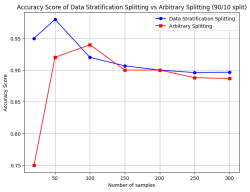

Figure 16: 90/10 split

In figures 13 and 14, it can be seen, that when there is a reasonably sized sample size the old method of data splitting, arbitrary data splitting and the new method of stratification splitting have very similar accuracy scores, following the same pattern as the number of samples increases. However, figures 15 and 16 show that as the split between the training and validation set becomes more unbalanced, stratification splitting significantly outperforms the original method, in some ranges of sample size.