FEW-SHOT LOGISTIC REGRESSION FOR MULTIMODAL DATA

TECHNICAL REPORT

Thomas Zarri
2nd year Computer Science, Durham University
thomas.zarri@durham.ac.uk

1 Comprehensive dataset exploration

1.1 KNOW

Basic statistics of each modality of data are shown in table 1. This reveals that there are a large number of classes in comparison to the number of samples, justifying a few-shot learning paradigm to account for the small number of training samples per class.

Table 1: Dataset Summary Statistics

Modality	Count	Number of classes	Samples per class	Features
Brain EEG	16540	1654	10	561
Image	16540	1654	10	100
Text	16540	1654	10	512

Analysing each feature shows that they are not scaled and have greatly varying standard deviations, minimums, and maximums. For each feature, we can calculate the percentage of outliers by IQR to gauge the validity of that feature. Some of the worst features by outlier count are shown in table 2.

Table 2: Dataset Summary Statistics

Modality	Feature Number	Mean	Standard Deviation	Percentage of outliers
Brain EEG	475	0.3148	1.037	0.9138
Image	24	0.0	5.045	1.8400
Text	286	-0.2040	0.3073	0.7282

1.2 SEE

A histogram can be used to display the distribution of the mean of each feature in figure 1. This displays the clear skew in the image feature means, and the large positive outliers that offset the brain and text feature means.

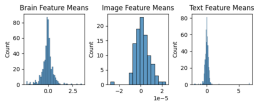


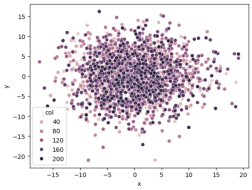
Figure 1: Histogram of the mean of each feature

A box-and-whisker plot can be used to visualise the number of outliers by IQR in each feature, seen in figure 2. The circles represent values outside the whisker length of the histogram, equivalent to our IQR test. Notably, the image feature has 2 extreme outlier instances, suggesting they should be culled before scaling to ensure the scaling isn't biased by extreme outliers.

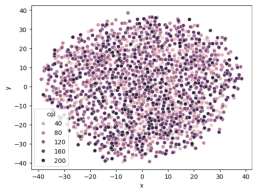
Figure 2: Box plots for each modality's worst feature

1.3 FIND

tSNE visualises the high dimensionality of the original training set in a lower dimensional space. A subset of 200 classes will be taken, and decomposed with **PCA** to perform an initial reduction in the dimensionality of the data, then displaying the **PCA**-embedded data. Then the **tSNE** model will be fit to create a more separable visualisation. Figure 3**a** displays the data embedded with only a reduction from **PCA**, while figure 3**b** shows the data embedded using **tSNE**, with improved separability. Regardless, there are no obvious clusters, indicating a more sophisticated approach to classification is required.



(a) PCA



(b) tSNE

Figure 3: Dimensionality Reduction Figures

2 Custom Model implementation

2.1 IMPLEMENT

Logistic Regression will be used alongside a **One Vs Rest** classifier. The number of models a **OvR** classifier requires scales linearly with the number of classes - which is useful since we have a large number of classes. **Logistic Regression** was chosen as it makes no assumptions about the distribution of classes in feature space - whereas, for example, **KNN** assumes similar classes will have similar distance to each other. This is relevant considering that despite **PCA/tSNE** embedding, we couldn't see clear clusters within the class distribution.

Training will occur by minimising a standard log-loss function, and calculating weight derivatives to perform gradient descent. A **learning rate** of 0.1, running for 100 epochs, and running without regularization is our initial hyper-parameter configuration.

Loss equation, where y represents our training labels, C our regularization strength, m our training samples, and p our predicted probabilities. Note the penalty term for L2 regularization.

$$-ylog(p) + (1-y)log(1-p) + \frac{C}{2m}\sum \theta^2 \tag{1}$$

Derivative for gradient descent, where m is the number of samples, x is our training data, h is our hypothesis function, C is our regularization strength, and y is our predicted probabilities, note this function is already vectorised, and includes the partial derivative of the penalty erm.

$$\frac{1}{m}(x^T(h_\theta(x) - y) - C\theta) \tag{2}$$

2.2 COMPARE

SkLearn's model took significantly less time than our model to train, this could be attributed to stopping rules, a faster/more efficient solver, or parallelization of training binary models across the **OvR** classifier. These results are shown in table 3.

Table 3: Training summary

Model	Training time	Final Accuracy
Custom	3.763	0.1733
Sklearn	0.812	0.1467

Figure 4 compares common performance metrics between the custom model and **SkLearn**'s implementation. The custom model has higher accuracy, recall, and f1-score, where the **SkLearn** implementation has a higher precision. This could be due to **SkLearn** using a different solver, and/or using regularization by default.

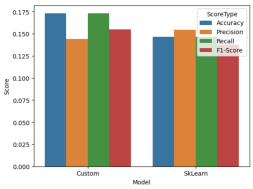


Figure 4: Comparison of common classification metrics

2.3 IMPROVE

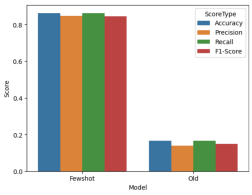
The problem of limited samples per class - identified in section 1 - is likely the largest hindrance to the model's performance. One way to counteract this is by adopting a few-shot learning paradigm. Following will be implemented to attempt to adapt to the few-shot paradigm:

- Combine modalities to increase features of training data, giving the model more to learn from
- Implement **L2 regularization** to prevent **overfitting**, considering model has small number of training examples
- Reduce train-test split to ratio 3:7, to enforce the few-shot learning paradigm

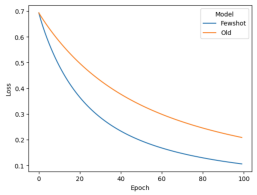
3 Result Analysis and Visualisation

3.1 PERFORMANCE

Figure 5**a** compares common classification metrics: **accuracy** (the percentage of correct predictions); **precision** (the ratio of correctly predicted positive instances, to all predicted positive instances); **recall** (the ratio of all correctly predicted positive instances, to all true positive instances); and **F1-score** (the product of precision and recall divided by the sum of precision and recall). The few-shot model sees dramatically better performance across all metrics, and converges faster, as seen in figure 5**b**.



(a) Comparison of common classification metrics



(b) Convergence of few-shot model vs old model

Figure 5: Comparison of few-shot model and old model

3.2 VISUALISATION

3.3 ABLATION

Evidently, the problem of a small number of samples per class has been greatly diminished, as the few-shot paradigm can learn from fewer samples per class, and at a higher accuracy than the old model.

4 Paradigm Design and Data Splitting

4.1 PARADIGM

Train-test splits with fewer training samples could allow for cheaper and easier data collection, and allow for a faster training time, as it would allow for less training data.

The initial configuration of hyper-parameters is also unlikely to be optimal, where a more optimal configuration could lead to higher accuracy or faster convergence.

4.2 ADJUSTMENT

K-folds cross-validation will be used to test performance of various train/test splits, ranging from very few training samples per class (3) to our baseline (7).

A hyper-parameter grid search will be used to attempt to find the best hyper-parameters for the model - this is being performed with a reduced data set for performance reasons. The following hyper-parameters will be concurrently varied:

learning rate, **regularization strength**, and **epochs**. This is also running with a K-folds cross-validation to ensure validity of the results.

4.3 REFLECTION

The figure 6 shows how metrics decrease with train-test split, as the model has fewer training samples to learn from, with a notably sharper drop between 3 to 4 training samples per class.

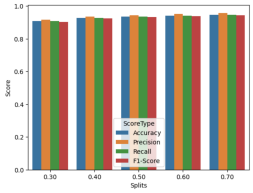
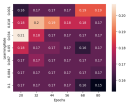
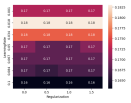


Figure 6: Varying train test splits, with classification metrics

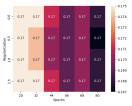
Figure 7 displays the relationships between hyper-parameters and the model accuracy. Learning rate and epochs have a weak diagonal pattern, where learning rate must decrease as epochs increase, or vice versa, for optimal performance. Figure 7**b** shows no relation between the hyper-parameters, but a slight favourite in terms of learning rate. Figure 7**c** shows better results with a very small number of epochs.



(a) Learning Rate vs Epochs



(b) Learning Rate vs Regularization



(c) Regularization vs Epochs

Figure 7: Hyper-parameter relation figures

The best models from the hyper-parameter grid search based on accuracy, and their respective scores are show in figure 8. These results also show better performance with a low number of epochs, and a consistent learning rate of 0.3, with regularization having little effect.

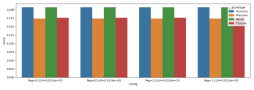
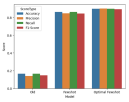
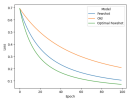


Figure 8: Top 4 configurations by accuracy

The final optimised few-shot model will have the optimal hyper-parameters obtained from figure 8, and will be using a modified train-test split that strikes a balance between reduced training samples, and loss of performance. Figure 9 shows a full comparison of the old model, the few-shot model, and our optimised model. Figure 9**a** shows the optimised model having marginally better performance over the few-shot model, while figure 9**b** shows faster convergence and a lower final loss for the optimised model.



(a) Classification metrics



(b) Loss graph

Figure 9: Final result metrics figures

5 Citations

References

[1] Du, Changde and Fu, Kaicheng and Li, Jinpeng and He, Huiguang. Decoding Visual Neural Representations by Multimodal Learning of Brain-Visual-Linguistic Features. in *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 9, pp. 10760-10777, 1 Sept. 2023*