Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Advanced Machine Learning - Project 1

Wojciech Kosiuk, Adam Majczyk, Damian Skowroński

Version 1.0

31.03.2024

Contents

1 Methodology (Tasks: 1, 3.1, 3.2) **2**

1.1 Task 1 - Datasets - selection and preprocessing 2

1.2 Task 3.1 - Stopping Rule 2

1.3 Task 3.2 - Performance measurements 2

2 Convergence analysis (Task 3.3) **3**

3 Comparison of classification performance (Task 3.4) **5**

4 Comparison of classification performance of models with and without interactions (Task 3.5) **6**

4.1 Models quality 6

4.2 Convergence of the models 7

5 Final conclusions **8**

1 Methodology (Tasks: 1, 3.1, 3.2)

1.1 Task 1 - Datasets - selection and preprocessing

Dataset Selection

The used datasets come from the OpenML platform. They are namely:

- Small datasets (i.e. ≤ 10 variables):
  1. ***banknote-authentication*** [1] (ID: 1462), 4 predictors, 1372 instances.
  2. ***puma8NH*** [2] (ID: 816), 8 predictors, 8192 instances, binarized version.
  3. ***phoneme*** [3] (ID: 1489), 5 predictors, 5404 instances.
- Large datasets (i.e. > 10 variables):
  1. ***bank32nh*** [4] (ID: 833), 32 predictors, 8192 instances, binarized version.
  2. ***PizzaCutter1*** [5] (ID: 1443), 37 predictors, 661 instances.
  3. ***MegaWatt1*** [6] (ID: 1442), 37 predictors, 253 instances.
  4. ***analcatdata_authorship*** [7] (ID: 970), 70 predictors, 841 instances.
  5. ***mc1*** [8] (ID: 1056), 38 predictors, 9466 instances.
  6. ***twonorm*** [9] (ID: 1507), 20 predictors, 7400 instances.

All of them are listed as dense, binary classification problems with only numeric variables.

Preprocessing

The datasets underwent preprocessing to adhere to the project requirements. This included:

1. Imputation of the mean of variable for missing values.
2. Removal of collinear predictive variables (variables with an absolute value of the Pearson correlation coefficient larger than 0.7 were dropped). Subsequently, the datasets were examined to ensure they still met the necessary variable count.

1.2 Task 3.1 - Stopping Rule

The stopping rule is based on the moving average and is outlined in Algorithm 1:

Algorithm 1: Stopping rule

$tol \leftarrow select\_tolerance\_value(default = 10^{-5})$
$lb \leftarrow select\_lookback\_value(default = 5)$
for *e in range(epochs)* **do**
  if $e > lb$ *and avg(*$cost_{e-1}$*, ..,* $cost_{e-lb}$*) -* $cost_e$ $<= tol$ **then**
    $stop\_iterations()$
  end
end

This stopping rule allows for fluctuations in the cost function, as long as the average cost over the preceding lb epochs is greater than the cost of the current epoch by more than tol (i.e. the average reduction of cost over the last lb epochs is greater than tol). It allows for noisy/unstable algorithms (such as SGD) to continue training, while also ensuring a general decrease in the average cost.

1.3 Task 3.2 - Performance measurements

For the performance metric, **Balanced Accuracy** was selected. There were 3 methods of training Logistic Regression to be tested: SGD, ADAM, IWLS. Each model was tested on 9 datasets.

For each of the 9 dataset there were 5 train-test splits (80-20) created. Each split for each dataset received a set random state (1-5) to ensure reproducibility of the results (1-5 was also chosen, so that each method of training was compared on the same splits - the comparison is therefore fair). The balanced accuracy was collected for each split. It was calculated on the test portion of each split. This in total resulted on 135 training processes (3 methods $\times$ 9 datasets $\times$ 5 splits).

The 3 methods of training were trained until they converged (i.e. triggered the stopping rule in Algorithm 1) or reached a limit of 500 epochs. The coefficients from the last epoch before triggering the stopping rule or from the 500-th epoch were considered as the result.

2 Convergence analysis (Task 3.3)

NOTE: To make the convergence comparison fair, it was decided to train both SGD and ADAM with batch sizes equal to 1 (since the standard definition of SGD requires the batch size to be 1).

Training time vs number ofepochs

Table 1: Comparison of aggregated training time and number of epochs per method.

method	training time (seconds)				number of epochs			
	mean	std	min	max	mean	std	min	max
SGD	3.45	6.36	0.02	21.72	122.60	135.55	8	416
ADAM	6.90	5.66	0.92	21.44	235.04	137.45	97	500
IWLS	65.17	114.27	0.05	437.70	44.82	42.24	11	144

Table 1 illustrates significant variations in the performance of the methods across the epxeriments. SGD tends to achive the fastest convergence time, closely followed by ADAM, while IWLS exhibits considerably slower performance. In terms of the number of epochs, IWLS generally requires fewer epochs, ADAM the most, and SGD falls in between.

This observation suggests that each epoch of IWLS requires significantly more time to compute compared to the other two methods. However, the algorithm demonstrates rapid convergence in terms of the number of epochs. The prolonged computation time is likely attributed to the computational complexity involved in calculating matrix inverses, which is particularly noticeable in larger datasets.

Interestingly, throughout the entire experiment, it was SGD, not IWLS, that triggered the stopping rule after the fewest number of epochs.

Epochs and cost values

Upon closer examination of Figure 1, the experiment results split by dataset are consistent with previous observations (see Table 1):

- For every dataset except *analcatdata_authorship* and *mc1*, IWLS converged in the fewest number of epochs. In these two datasets, SGD converged the fastest, but its resulting cost was the worst out of the three methods.
- ADAM consistently required the most iterations, and it usually converged before reaching the maximum limit of 500 epochs. An exception is observed in the *MegaWatt1* dataset, where ADAM failed to converge in any of the attempts, suggesting it could benefit from additional training.

Additional conclusions include:

- For small datasets (upper row, 3 graphs), each method exhibited less variance in training and yielded similar results in terms of cost. However, for larger datasets, the results were less consistent.
- Across all datasets, IWLS consistently achieved the lowest cost (indicating the best performance) and generally outperformed the other methods.

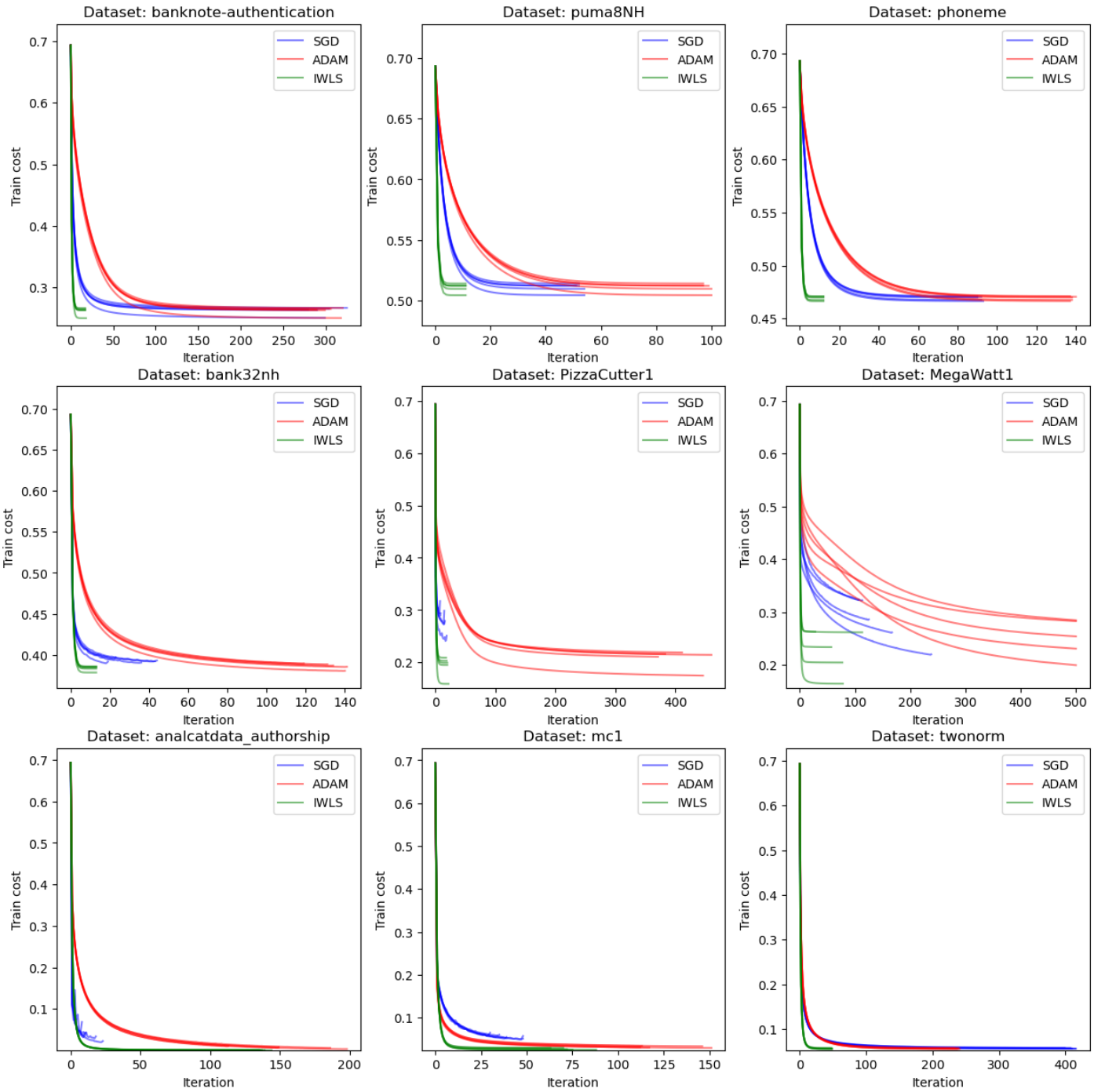

Figure 1: Comparison of the number of epochs (iterations) required for each method to converge per dataset.

3 Comparison of classification performance (Task 3.4)

To compare the performance of the implemented numerical methods for Logistic Regression, 4 other popular algorithms were chosen - namely LDA, QDA, Decision Tree and Random Forest. They were trained and tested on the same splits as described in Subsection 1.3 to ensure a fair comparison with the tested methods for Logistic Regression.

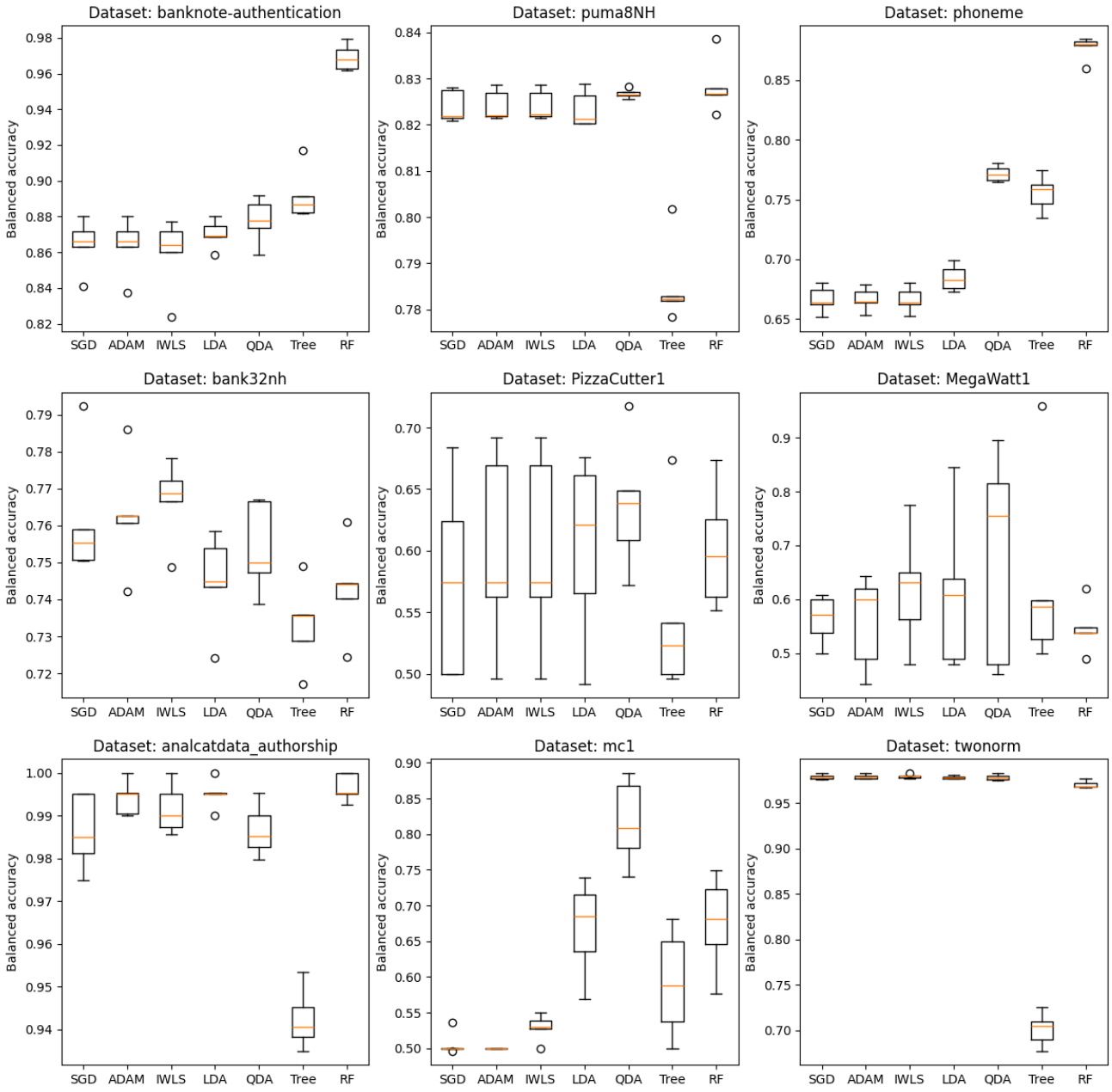

Figure 2: Balanced accuracy of implemented models against other popular algorithms.

In 3 datasets (*banknote-authentication*, *phoneme*, *mc1*) implementations from Python library (i.e. LDA, QDA, Decision-Tree, RF) have significantly better results. Two of these datasets are small, having only 4 and 5 predictors respectively. Decision tree emerged as the least effective algorithm, showing notably lower results on several datasets, while Random Forest emerged as the top-performing model on four datasets. LDA and QDA demonstrate performance closest to that of SGD, ADAM, and IWLS.

Comparing models of our own implementations (i.e. SGD, ADAM, IWLS), in the majority of datasets, the results of these models are similar. However, datasets *PizzaCutter1*, *bank32nh*, and *analcatdata_authorship* exhibit a trend where the SGD model performs worse than the other two methods (as also evident in Figure 1 when examining cost). Additionaly, in datasets *bank32nh* and *mc1*, the IWLS method achieved the best results. While there is a clear correlation between cost (Figure 1) and accuracy, it's not as pronounced as one might expect.

4 Comparison of classification performance of models with and without interactions (Task 3.5)

Models were compared with and without interactions, by training each of them on both its original features and the new features created by multiplying pairs of (predictive and non-intercept) features. This experiment was conducted only on the 3 small datasets.

4.1 Models quality

In Figure 3 a comparison of the balanced accuracy metric was made on models with and without interactions. In case of *banknote-authentication* and *phoneme* datasets, using models with interactions improved balanced accuracy by 4 percentage points. These datasets have 4 and 5 predictors, respectively. Such a small number of features may be the reason why adding interactions as additional features may improve the models' learning process. This hypothesis is further supported by the results on the *puma8NH* dataset which has 8 predictors. Introducing interactions decreased variance of the models. However, it is important to notice that the variance is relatively small compared to other datasets.

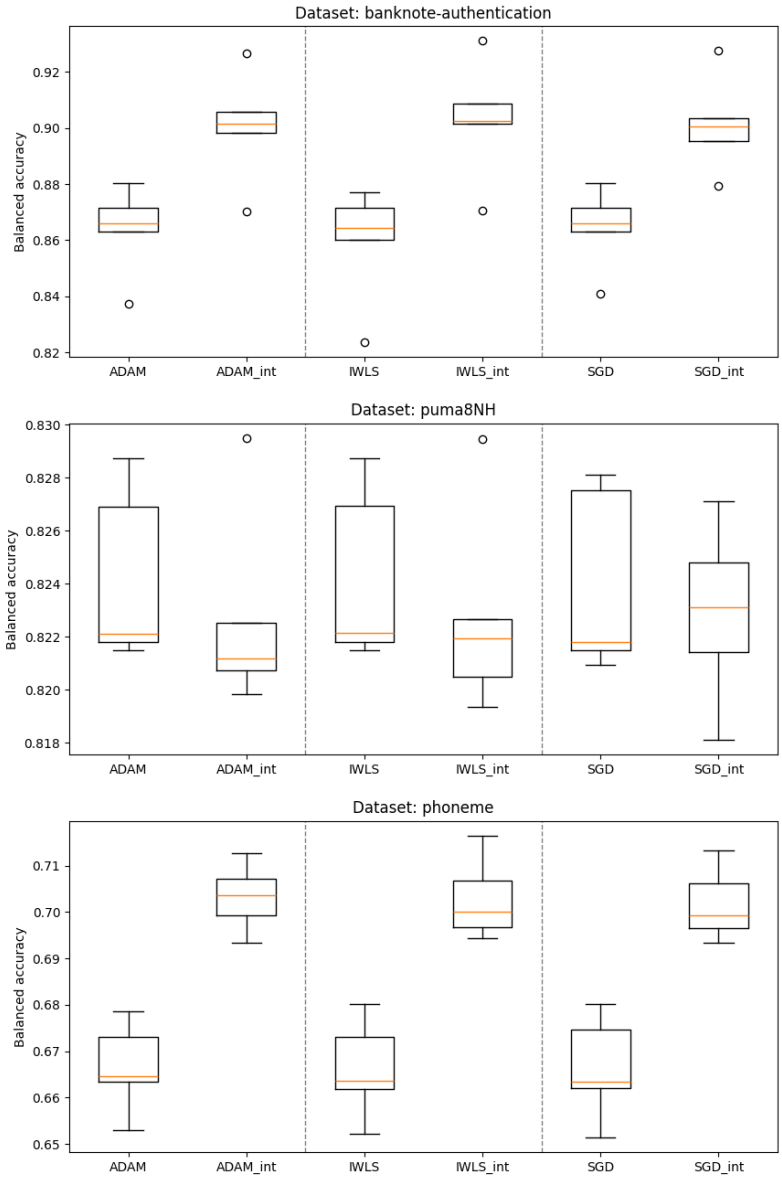

Figure 3: Balanced accuracy of models with and without interactions (***__int*** denotes the model with added interactions).

4.2 Convergence of the models

The comparison also extended to examination of the training process. The goal was to observe the influence of providing additional features on the speed of convergence.

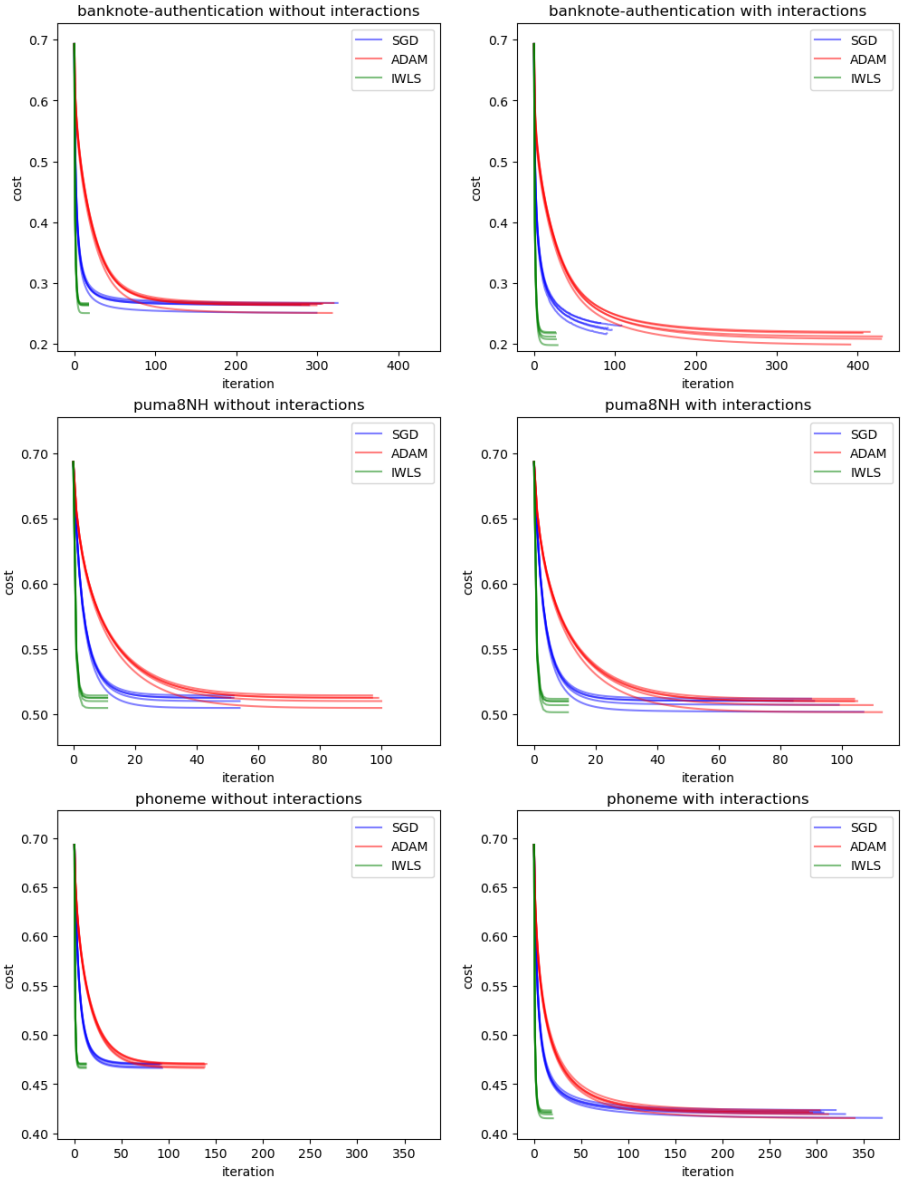

Figure 4: Convergence of models with and without interactions

In general, models with interactions tend to converge more slowly, which is an expected behavior due to the increased number of features. This difference is particularly evident in the *phoneme* dataset, where the model without interactions required a maximum of 140 iterations, while the model with interactions sometimes needed more than 350 iterations. Similarly, in the *banknote-authentication* dataset, models with interactions typically required an additional 100 iterations. However, this effect is not visible on the *puma8NH* dataset.

5 Final conclusions

Based on the conducted experiments, the following may be concluded:

- In terms of epochs, IWLS was the fastest algorithm. Despite that, it required the most real time to train.
- SGD usually converged faster than ADAM both in terms of epochs and real time.
- ADAM converged in a more stable manner (fewer fluctuations) than SGD.
- SGD and ADAM usually exhibited similar predictive performance. IWLS may be deemed as the best in terms of accuracy out of the 3 Logistic Regression methods.
- Other methods, such as Random Forest and QDA, often perform better or at least comparable to Logistic Regression.

References

[1] Volker Lohweg. Banknote Authentication. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C55P57.

[2] Luis Torgo. puma8NH. DELVE. Source: http://www.ncc.up.pt/ ltorgo/Regression/DataSets.html.

[3] Dominique Van Cappel. phoneme. KEEL, ELENA. Source: https://sci2s.ugr.es/keel/dataset.php?cod=105#sub2.

[4] Luis Torgo. bank32NH. DELVE. Source: http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html.

[5] Deter Bergman and Hans Bauer Jesus. PizzaCutter1. Source: UNKNOWN.

[6] Deter Bergman and Hans Bauer Jesus. MegaWatt1. Source: UNKNOWN.

[7] Jeffrey S. Simonoff. analcatdata_authorship. Springer-Verlag, New York, 2003. Source: http://www.stern.nyu.edu/~jsimonof/AnalCatData.

[8] J. Sayyad Shirabad and T.J. Menzies. The promise repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[9] Michael Revow. Twonorm. Source: http://www.cs.toronto.edu/~delve/data/twonorm/desc.html.

List of Tables

1 Comparison of aggregated training time and number of epochs per method. 3

List of Figures

1 Comparison of the number of epochs (iterations) required for each method to converge per dataset. 4

2 Balanced accuracy of implemented models against other popular algorithms. 5

3 Balanced accuracy of models with and without interactions (***__int*** denotes the model with added interactions). 6

4 Convergence of models with and without interactions 7