Machine Learning - Project 1

Amine Lamouchi, Charbel Chucri, Quinten Dewulf
Department of Computer Science, EPFL, Switzerland

I. Introduction

The aim of this project is to train a model that classifies events measured by the ATLAS experiment at CERN in order to distinguish between signal (Higgs boson) and background. Different methods were tested in order to improve the accuracy of our models on this data.

II. Data Preprocessing

A. Exploratory Phase

After looking at the different features in the training set, we noticed that some columns had missing values labeled with -999. Moreover, we found a pattern for those missing values: they mostly depend on the feature "PRI_jet_num". In particular, samples with jet number 0 have 11 features with missing values, samples with jet number 1 have 7 such features and other samples have significantly less features wit missing values. Hence, we decided to split our data into 3 independent datasets corresponding to samples with jet number $\{0\}$, $\{1\}$ and $\{2, 3\}$.

Then, when predicting a new instance, we will use the model corresponding to its jet number. This preprocessing step yielded an improvement of around 0.018 in categorical accuracy for all models tried.

B. Dealing with Missing Data

For each of the 3 datasets, we removed features that had a majority of missing entries in the training data or a standard deviation of 0 (this happened for samples with equal jet number). Note that different datasets had different removed features. This fix dealt with the majority of missing entries but not all of them. For what remained, we replaced -999 with the mean of the available values for the corresponding feature.

When given an instance to predict, we skip the features that we removed while training and change any -999 entry with the mean of the corresponding feature found during training.

C. Feature Expansion

To mitigate the data lost by the missing entries, we decided to expand the input by adding a feature x_i^2 for each feature x_i.

D. Standardizing

Once the above steps are done, we standardize the training and test datasets by using the means and standard deviations found during training.

III. Hyper-Parameter Tuning

Using regularised logistic regression led us to consider the problem of hyper-parameter tuning. In particular, we focused on tuning the Ridge regularization parameter λ. To achieve this, we first performed grid search on a fixed set of values and picked the hyper-parameter associated with the lowest test loss. One downside of this approach is that it leads to an optimistically biased evaluation of the model performance since we are using the same test set to evaluate the different models. To overcome this problem we nested the grid search inside a k-fold cross validation procedure. The resulting algorithm is the following:

- Divide the data set into k-folds
- Repeat k times:
  - Perform grid search on k-1 folds
  - Pick the hyper-parameter associated with the lowest test loss on the kth fold
- After this loop we get a list of k hyper-parameters, each of them is associated with the lowest test loss on a given fold.
- Iterate over the list of k hyper-parameters and pick the one associated with the lowest test loss on the entire data set.

IV. Newton's method with a stochastic Hessian

A. Motivation

Newton's method is an optimization method that requires fewer steps to converge by computing more informative 'weight update vectors'. However, computing update vectors is computationally expensive. We thus approximated Newton's method update vector by swapping out the Hessian matrix in the formula with a stochastic Hessian matrix.

B. Experiment

We've set up an experiment to show that Newton's method with a stochastic Hessian will converge faster than traditional gradient descent. We used gradient descent for a logistic regression model evaluated by log loss. The hyper-parameters are identical for all models, so the only element that varies in the different optimization algorithms are the update vectors. The weight update vector for gradient descent being $-\nabla\mathcal{L}(\mathbf{w})$ and $-(\mathbf{H}_{stoc}(\mathbf{w}))^{-1}\nabla\mathcal{L}(\mathbf{w})$ for Newton's method with a stochastic Hessian.

The models are being trained on 80% of the normalized training data, the identical initial weights all undergo 500

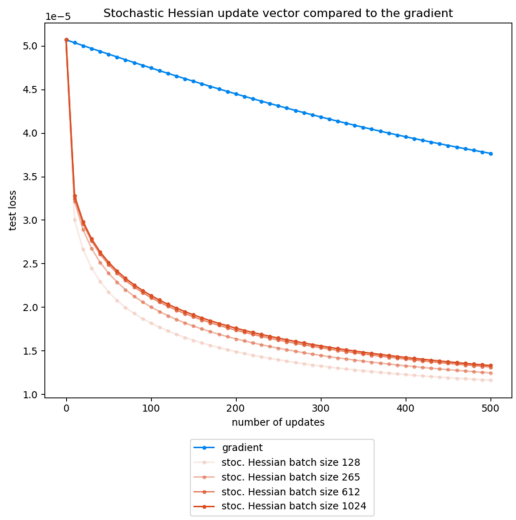

Figure 1. Stochastic Hessian update vs GD without feature expansion

updates, while being tested on the test data (the remaining 20% of the normalized training data) every 10 updates. Because the computation of the stochastic Hessian is a non-deterministic process, the displayed test loss of a model trained using a stochastic Hessian is the average of the real test loss of 10 models trained with this algorithm.

C. Results

The results show that the test loss decreases more rapidly for models using an update vector computed with a stochastic Hessian matrix. This implies that our approximation of the Newton's method update vector is more informative than the gradient.

Figure 1 specifically shows that the update method using the stochastic Hessian needs fewer updates to converge compared to gradient descent. Thanks to the reduced computational cost of the stochastic Hessian, this method converged overall faster than gradient descent.

However, note that this was in a setting where we did not have feature expansion. Adding the latter changed the results significantly, due to numerical errors that became more apparent in the stochastic Hessian method.

It is also interesting that the update vectors using a stochastic Hessian computed out of a smaller batch of samples consistently deliver a lower test loss. This can be explained by the fact that smaller batch size can lead to better generalization (and thus better test loss) as seen in [1].

V. Results

We tested multiple models on our data. We optimized the hyper-parameters when applicable as described above and then we quantified how good a model is using cross-validation locally. The results of the different models are summarized in table I. Note that we emphasize the difference between using feature expansion and not using it as it yielded important difference in particular with respect to the Hessian method.

Without Feature Expansion	
Method	Accuracy
log.reg. + stoc. Hessian	0.767 $\pm$0.015
log.reg. + GD	0.754 $\pm$0.026
lin.reg. + least-squares	0.753 $\pm$0.015
With Feature Expansion	
Method	Accuracy
log.reg. + GD	0.816 $\pm$0.015
reg.log.reg. + GD	0.794 $\pm$0.021
lin.reg. + least-squares	0.778 $\pm$0.007
log.reg. + stoc. Hessian	0.644 $\pm$0.099

Table I
Accuracy of different models tested

VI. Summary

Pre-processing by dividing the data set into 3 and using feature expansion boosted our accuracy. Logistic regression with gradient descent and Stochastic Hessian performed the best but the former achieved the best results overall. We believe it performed better than its Hessian variant because the latter is prone to numerical errors, which was apparent when used with the expanded features.

References

[1] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.