Assignment 02
Learning from Data, Related Challenges, Classification

submitted for

EN3150 - Pattern Recognition
Department of Electronic and Telecommunication Engineering
University of Moratuwa

Udugamasooriya P. H. J.
220658U
Progress on GitHub ⬀

20 August 2025

1 Linear Regression

Question 01 The main reason is the presence of outliers.

Based on the inliers, we see that the desired line is an almost horizontal one. To see why a slanted line has been produced instead of a horizontal line, we look at the behavior of the squaring function;
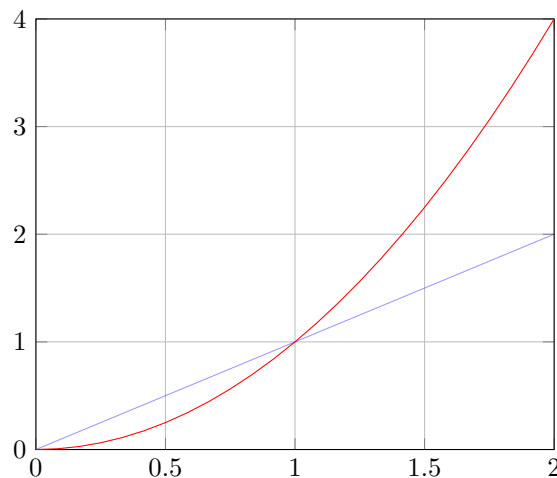

Figure 1: $y = x^2$

Note that for $|x| < 1$, $x^2 < |x| < 1$, i.e., the squares of numbers less than 1 are less than the numbers themselves (in absolute value). This means that the squared loss function can remain small enough for prediction errors that are just slightly below 1, but increases rapidly as it increases beyond 1.

In the given situation, even though a horizontal line would give near-zero error for the inliers, the error due to outliers would be very large. It is reasonable then, that a more optimal, i.e., with a lesser mean squared error, could be achieved with a slanted line, with lesser distance between outliers and the corresponding predictions.

Such a line would increase the gap between the inliers and the corresponding predictions and therefore increase the loss, but *not by too much*, due to the behavior of the square function. Instead, the reduction in the already-large gap between the outliers and the corresponding predictions, *will* cause a *rapid* decrease in the loss, *exceeding* the increase in the loss due to the inliers.

Question 02 Scheme 1.

The issue we wish to mitigate is the rapid increase in loss due to outliers, while keeping the loss due to inliers significant, with the intention that a model that better fits the inliers and is not strongly affected by the outliers as described above would be preferred.

With Scheme 1, we do not completely neglect the effect of outliers, but significantly scale down their contribution to the loss function, so they do not dominate the loss when the loss due to the inliers is small.

Scheme 2 is obviously not suitable, as it amplifies the loss from outliers over inliers; minimizing such a function would require the error due to outliers to be significantly reduced, causing more slanted line to be preferred, as described in the previous problem.

Question 03 In this case, the prediction to be made is of a categorical nature, i.e., a probability distribution over several possible classes, rather than a single continuous value.

In linear regression, we try to fit a continuous linear model to a given set of data points. In doing so, we make the following assumptions:

- the error between an actual data point and the corresponding prediction is normally distributed
- the data points vary continuously

However, in the case of categorical data, the variation of data points tends to be more discrete in nature, and the error between continuous-valued predictions and the data points is no longer normally distributed.

More specifically, if each voxel is treated as a feature, we expect the data matrix $\mathbf{X}$ to have almost linearly dependent rows/columns, as we expect a high correlation between the voxels due to spatial smoothness in the images. This would lead to instability in the computation of $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$ required for the ordinary least squares weights.

On a final note, linear regression performs best when there are fewer features than data points for training. In our case, we expect the input images (data points) to consist of large numbers of voxels (features), but that there would be fewer such images than voxels. This might also lead to unstable coefficients, and sometimes overfitting, in the absence of regularization.

Question 05

2   Logistic Regression

Question 02 The following is a snippet of an error that was encountered:

```
------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

----> logreg.fit(X_train, y_train)

ValueError: could not convert string to float: 'Adelie'
```

The error indicates that not all data items given to `logreg.train` could be converted to the `float` data type. Clearly, all data *must* be *numeric* in order to enable the computation of weights and gradients required for logistic regression.

The error arises from the line `X = df_filtered.drop(['class_encoded'], axis=1)` in Listing 1 provided; X is prepared by dropping only the 'class_encoded' column, and therefore will retain the 'species', 'island', and 'sex' columns from df_filtered, whose entries are strings.

In our case, 'species' is not a feature—it is the label we intend to predict—and therefore mustn't even be contained in X at all.

We have two options regarding 'island' and 'sex'; if we believe the label we are trying to predict, i.e., the species of the penguin, depends on either of these features, then we could form X by replacing these columns with numerically encoded versions of them, or, if we believe they have little or no influence onf the label, we could drop them altogether.

It is unlikely that the sex of a penguin has any influence on its species, so we will drop this feature entirely. A penguin's habitat ('island'), on the other hand, may be considered as influencing its species, as different species may have adaptations for different habitats. Whether or not to include it as a feature then depends on one's exact requirements. For the purposes of this problem, we will look at both options.

Hence, we resolve the error as follows. We drop the 'species' and 'sex' columns entirely from 'df_filtered' to form X_. Then, from X_ we form X_1 by one-hot encoding the contents of the 'island' column and including them as new features, and X_2 by dropping the column entirely. The relevant lines of code are given in Listing 1.

```
X_ = df_filtered.drop(['species', 'sex', 'class_encoded'], axis=1)

# Form X_1 by one-hot encoding the island names and including them as new
    features
ohe = OneHotEncoder(sparse_output=False, drop='first', handle_unknown='ignore')
islands_encoded = ohe.fit_transform(X_[['island']])
X_1 = pd.concat(
    [
        X_.drop('island', axis=1).reset_index(drop=True),
        pd.DataFrame(islands_encoded, columns=ohe.get_feature_names_out(['island'
            ]))
    ], axis=1
)

# Form X_2 by simply dropping the 'island' column
X_2 = X_.drop(['island'], axis=1)
```

Listing 1: Fix for the error

We then run the code given in Listing 2 to train the logistic regression model with both X_1 and X_2. The outputs generated by each one respectively, are as follows:

```
Accuracy : 0.5813953488372093
[[ 2.75836572e-03 -8.48924277e-05  4.43272550e-04 -2.85530563e-04
   1.84666104e-04 -1.04725548e-04]] [-8.62161382e-06]

Accuracy : 0.5813953488372093
[[ 2.76291513e-03 -8.24983988e-05  4.82711005e-04 -2.87644233e-04]]
 [-8.35428389e-06]
```

The same accuracy has been achieved in both cases, and the coefficients for common features in both cases are very similar. We further note that in both cases, the following warning was raised: `ConvergenceWarning: The max_iter was reached which means the coef_ did not converge.`

Question 03 It is clear that SAGA (Stochastic Average Gradient Ameliorated) has performed poorly on the current dataset, as indicated both by the relatively low accuracy of about 0.58 on its perfor-

mance on the data, and the fact that the coefficients did not converge.

Being a stochastic method, SAGA performs best when the size of the dataset is large, ideally with more than tens of thousands of data points. This is because statistically, the gradient computed using a minibatch of the data points approaches the true gradient only when averaged over sufficiently many samples.

Hence, the main reason for the poor performance of SAGA on our specific dataset is its size. Our dataset consists only of 214 data points—too few for SAGA to perform optimally.

More specifically, we attribute the poor performance of SAGA to the following issues that may arise due to a dataset of this scale:

- the gradient computation has a high tendency to be noisy
- a lot more iterations will be needed in order for any convergence to occur, as indicated by the `ConvergenceWarning`
- the overhead associated with each iteration is not "offset" by a fewer number of iterations, as is usually the case with larger datasets
- sparsity?

Question 04 The same code snippet in Listing 2 given was run using the 'liblinear' solver in place of 'saga' solver with training and test sets constructed using both X_1 and X_2 as described above, yielding respectively the following outputs;

```
Accuracy : 1.0
[[ 1.51189823 -1.38940877 -0.14156163 -0.00378048  0.72408487 -0.5530247 ]
 [-0.07450833]

Accuracy : 1.0
[[ 1.59665154 -1.42501103 -0.15238046 -0.003951  ]] [-0.0755452]
```

In each case, an accuracy of 1.0 is reported.

Question 05 'liblinear' is a coordinate-descent method which uses all data points to compute the partial derivative with respect to a single coefficient at a time, and updates it over a number of iterations. Hence, it performs better on smaller datasets, with several hundreds of, or a few thousand data points, such as in our case.

As such, 'liblinear' does not incur as much overhead per iteration as a stochastic method such as SAGA. Thus, when the number of data points is small, it will converge faster than a stochastic method, with lesser memory and resource usage. Therefore, 'liblinear' outperforms SAGA on our current dataset.

Question 06 The `random_state` parameter passed to a `LogisticRegression` instance from `sklearn` controls how the coefficients will be initialized, and in the case of a stochastic method, how data points will be shuffled and selected randomly for iterations.

Hence, performing different runs with different seed values for `random_state` is expected to give different results and different accuracies, as each value would result in different coefficient initializations, and different minbatches of data points being used for updates.

Question 07 Sagasolver with feature scaling

Question 08 Categorical

3   First and Second Order Methods for Logistic Regression

Question 02 Method to initialize weights

Question 03 Loss function

Question 04 Newton's Method

Question 05 Plots

Question 06 Iterations

Question 07 Convergence