Machine Learning HWS21
Assignment 2: Logistic Regression

Timur Michael Carstensen - 1722194

07.11.2021

Contents

1 Dataset Statistics

1.1 a)

In kernel density estimation, the probability density function of a random variable is estimated using a finite set of observations. The kernel density plot then visualises the probability density over a continuous interval. In the case of the given dataset, the features have different data scales. More specifically, the first 54 features are given as relative frequencies whereas the last three are encoded as absolute values. Hence, Figure 1, which visualises the kernel densities for all features, is not really helpful, apart from telling us that most density is around zero which implies that most observations have rather small values, relative or absolute.

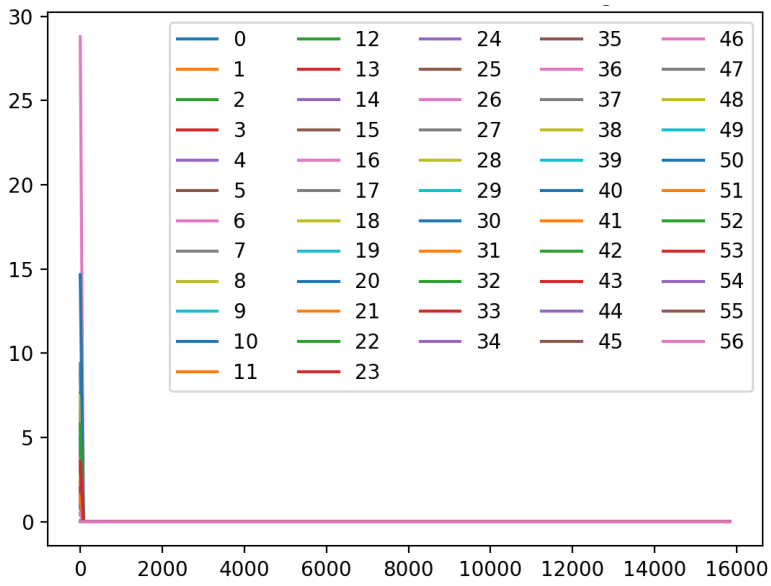

Figure 1: Kernel density plot for all features

Figure 3 and Figure 2 show the kernel density plots for the differently scaled features separately with truncated x-axes. In Figure 3 it is visible that most density is in the interval between 0 and 0.1 for the relative features. The same applies for absolute features, as shown in Figure 2. In general, we can say that our data is highly right skewed. Given the different scales, it is sensible to normalise our data to be able to visualise relative and absolute scaled features in one kernel density plot.

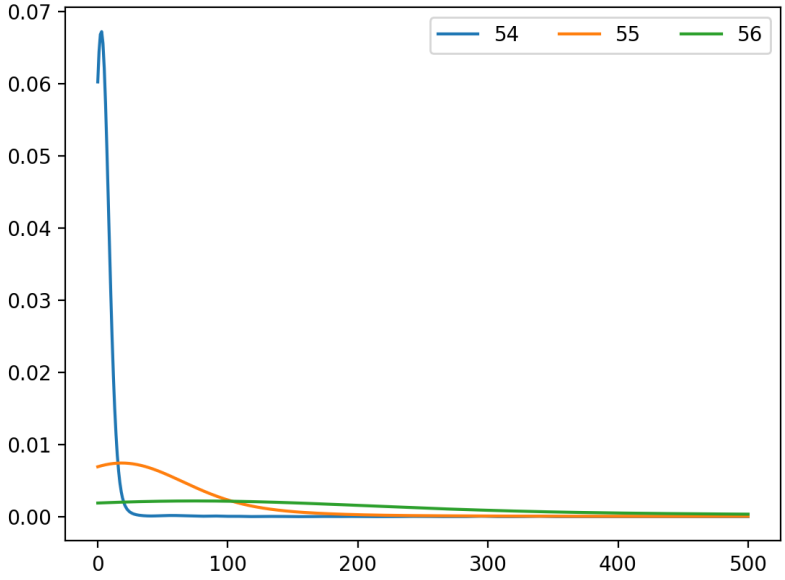

Figure 2: Kernel density plot for absolute features with truncated x-axis

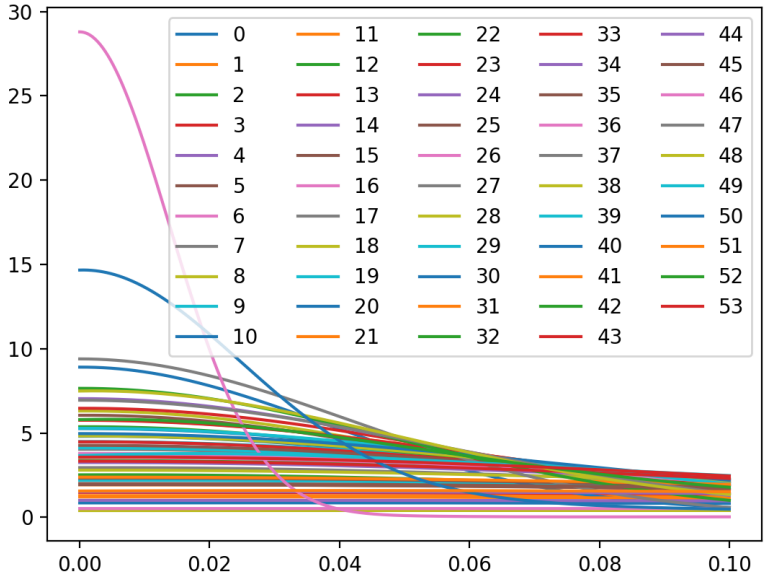

Figure 3: Kernel density plot for relative features with truncated x-axis

1.2 b)

cf. code. The test must be normalised using the mean and standard deviation of the training data. If one were to use the mean and standard deviation of the test set to normalise the test data, we would have different scales for train and test data and would not be able to classify correctly.

1.3 c)

After normalising the data, the mean of each feature is zero and most of the density therefore falls in the area around zero as can be seen in Figure 4. Similarly, now one must also consider negative values.
The data is still highly right skewed but now there is more density on values not directly around zero which shows that the normalisation has slightly lessened the effect of large numbers amounts of small observations.
Now, as mentioned in subsection 1.1, one can plot all features (relative and absolute) in one plot. Also, when increasing the x-axis range (cf. code), one can observe local maxima around 7 which is due to the squished data range that results from normalisation.

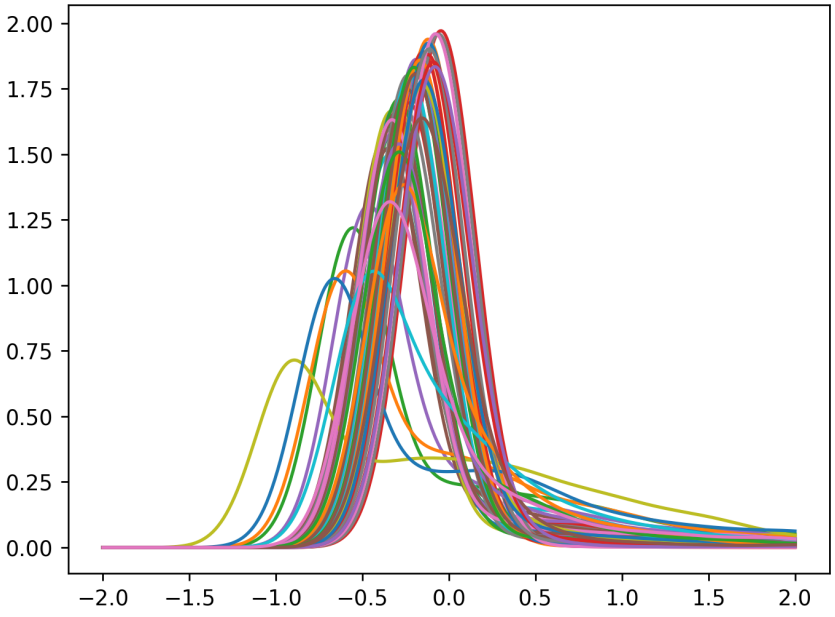

Figure 4: Kernel density plot for all normalised features

2 Maximum Likelihood Estimation

2.1 a)

By scaling the design matrix X by a scalar a and adding a constant b, the likelihood function with a bias term becomes:

$$p(y \mid aX + b, w) = \sum_{i=1}^{N} \sigma(w_0 + \sum_{j=1}^{D} w_j(ax_{ij} + b)) \tag{1}$$

$$p(y \mid aX + b, w) = \sum_{i=1}^{N} \sigma(w_0 + \sum_{j=1}^{D} w_j b + \sum_{j=1}^{D} a w_j x_{ij}) \tag{2}$$

Now one can see that the shifting (adding b) only acts as an additive term to our bias, the effect of which should be neglibile in the final result. Scaling the design matrix by a only scales the weight vector without changing the feature values. This does also not affect our likelihood as the initialisation of our weight vector is irrelevant due to our objective function (log-likelihood - not shown here) being convex.

Normalisation is performed due to the differing scales of the attributes (relative frequencies and absolute values). While this does not pose a problem for MLE estimation, during MAP estimation it could. If one were to use two vastly different scales for body measurements (e.g. weight in grams and height in metres), the one with the greater scale (metres) would receive larger weights. However, large weights are penalized in MAP estimation with a gaussion prior which would then not produce the optimal results. Finally, standardisation also helps to make the weights more interpretable across features.

2.2 b)

cf. code

2.3 c)

cf. code. Epoch implementation done with only vectorized operations.

2.4 d)

cf. code

2.5 e)

As shown in Figure 5, after only 10 epochs, SGD is already very close to the optimum and after 15 there is no more significant change in the objective function value. GD, however, takes comparatively longer to converge to the optimum and, in the end, gets closer to the global optimum than SGD.

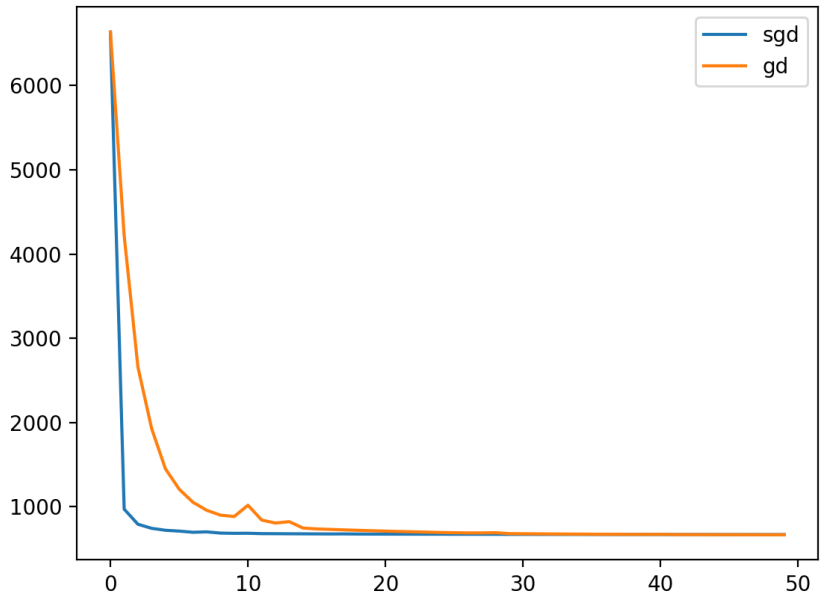

Figure 5: Log-likelihood development of GD and SGD from epoch 0 to 50

Figure 6 shows the learning rate development of GD and SGD. In the beginning they are the same but then SGD drops much faster and consistently stays below GD's learning rate. This is because SGD converges to the optimum (or close to it) much faster and then just "jitters" around it until the last epoch is reached. That is, GD continues to improve the objective functions value as we get a more accurate gradient in each epoch but therefore needs more epochs to reach the optimum. SGD on the other hand converges to its optimum (albeit a worse objective function value as compared with GD) in far fewer epochs. So, in the case of SGD, one could stop at a lower number of epochs (e.g. 50) if compute time was an issue. Hence, there is a trade-off between GD and SGD as one would technically need fewer epochs for the latter, while the individual epoch is much slower to compute (due to a lack of vectorization) as compared with the former.

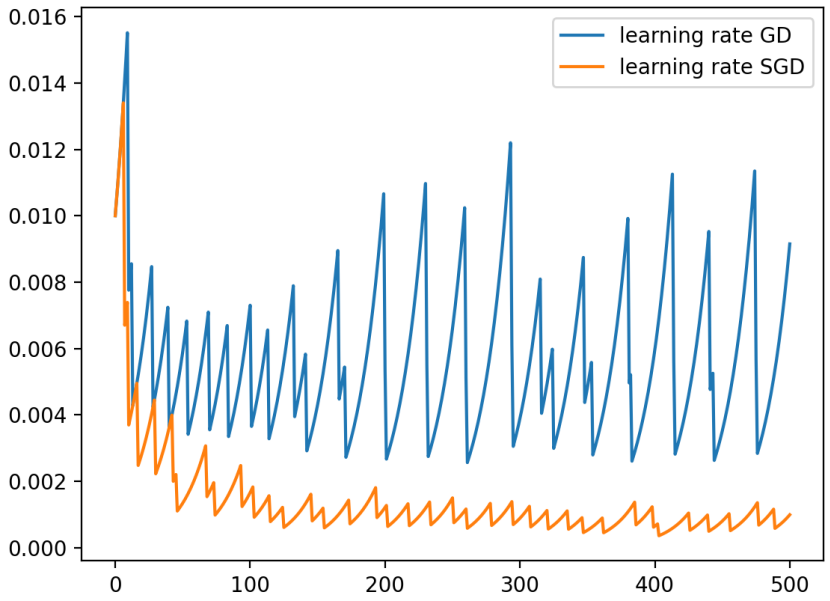

Figure 6: Learning rate development of GD and SGD

It is also interesting to note that the final weights for most of the features are the same (cf. Figure 7) with the only noteable exception being the 4th one (word_freq_3d). While both methods assign by far the most weight to the previously mentioned feature, the weights still differ dramatically which is due to the randomization in SGD. That the 4th feature gets a high weight, no matter the method, is also somewhat intuitive as most observations in which we observe the term 3d are spam in our training data set.

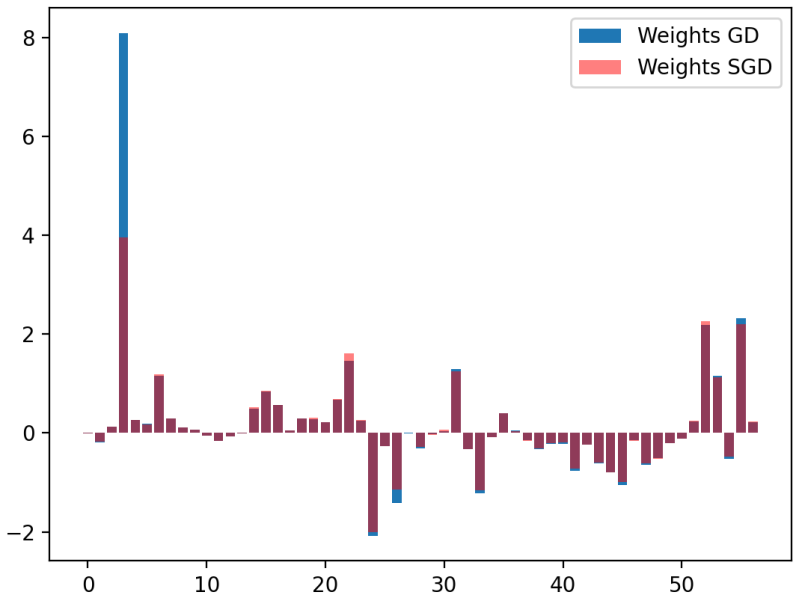

Figure 7: Final weight vectors for GD and SGD

3 Prediction

In prediction, first the observations' attributes are scaled by the "final" weight vector of each column and then passed into the sigma function to squish the result into the range between 0 and 1. Then, all values > 0.5 are assigned the positive class while the rest is assigned the negative class.
Given that the main difference between SGD and GD in regards to to the weight vector was already covered in subsection 2.5, the following will only cover GD.

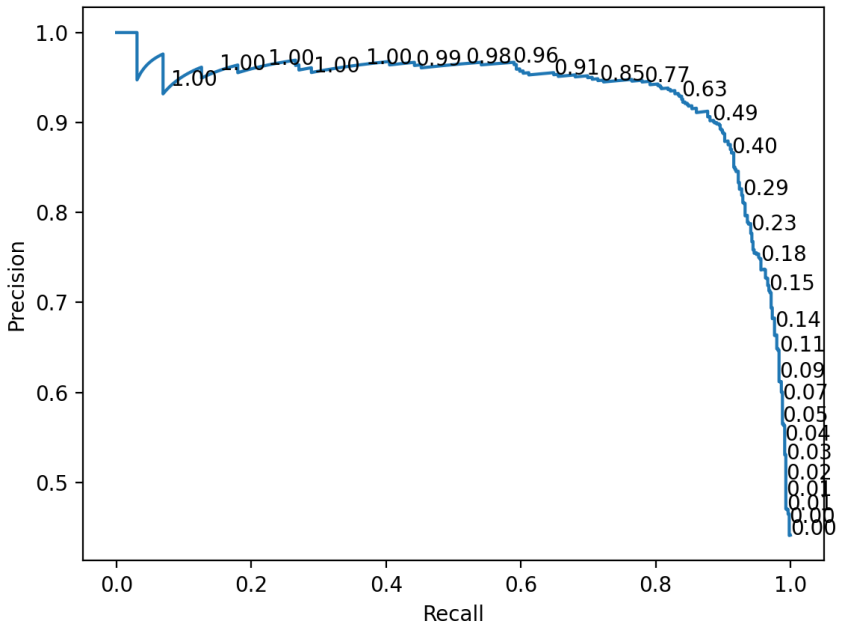

Figure 8: Precision recall curve

The model performs quite well with F1-scores of 0.93 for the negative and 0.89 for the positive class. Another interesting perspective is offered by the precision-recall curve for the positive class in Figure 8. Here, precision and recall are plotted for different values of the decision boundary (which usually is set at 0.5 for logistic regression). If one were to try to maximise the detected spam emails, they could decrease the threshold (decision boundary) and hence increase recall. However, this would worsen precision and lead to many non-spam mails being classified as spam (which is not desirable). Hence, the 0.5 decision boundary seems reasonable as it maximises precision and recall.

To make the weights interpretable, one must calculate the odds ratio for a given weight [Mol20]:

$$\frac{odds_{w_{j+1}}}{odds} = \exp(w_j) \tag{3}$$

Again, the most important feature by the odds ratio is the above mentioned word_freq_3d with an odds ratio of 3249.5 for GD and 52.21 for SGD. That is, the odds ratio would increase by 3249.5 for a one unit increase of the feature in a given example. The odds ratio here is much higher than for any of the other features and is explained if we take a look at our training data set. The feature is only ever observed for a total of 35 times in our training set and in those cases the corresponding email was spam 29 times

(i.e. 83 percent). In the real world it is reasonable to assume that the term 3d does not appear as often in spam emails as it does in the training set. Hence, this could be seen as overfitting of our model. Here, one must take into account the origin of the data (a researcher at HP, 3d would make sense to appear in the spam emails targeted at him if he was involved in computer graphics research).
Further, capital_run_length_longest, char_freq_$, word_freq_000 are the next most important features in decreasing order (ordered by the odds ratio). These are a little more intuitive as it is reasonable to assume that spam emails have more words/sentences typed out in capital letters or make use of otherwise less frequently used characters such as "000" or "$".
Generally, to avoid overfitting as is the case for "3d", it would be necessary to use a prior / perform regularization.

4 Maximum Aposteriori Estimation

4.1 a)

cf. code

4.2 b)

Given the same number of epochs, negative log-likelihood increases with an increase in the regularization parameter λ as our observations are obscured by the prior. That is, with an increase in λ, one places more weight on the prior and less weight on the observations (cf. ML assignment 1). This is confirmed when checking the values of our loss function for differing values of λ: when employing no prior ($\lambda = 0$) one gets the same value as before with GD (655). For $\lambda = 1$ we achieve 683, for $\lambda = 2$ 695, for $\lambda = 10$ 754, for $\lambda = 100$ 988 and for $\lambda = 500$ 1304.

The values of the loss function for test set also increase with increasing values of λ. This makes sense as the test set has a very similar distribution to the training set. If both sets had differing distributions, increases in λ might actually have improved the value of the loss function.

The accuracy of the predictions increases slightly with increases in λ at first. Starting from 91.86% at $\lambda = 0$, accuracy increases to 91.99% for $\lambda = 10$. Generally, prediction accuracy does not change much until one reaches very high values of λ.

4.3 c)

With increases in the regularization parameter, the weights become more similar and generally smaller / move closer toward zero. Hence, the enormous effect of the "3d" feature that was discussed in section 3 is lessened the greater the λ. As it was speculated that the high weight on the "3d" was due to overfitting, it makes sense that accuracy slightly increases for increasing an λ up until a certain point.

References

[Mol20] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.