University of Mannheim

Business Informatics and Mathematics

Fall 2023

IE 675b Machine Learning

Dr. Rainer Gemulla

Assignment 2: Logistic Regression

November 5, 2023

Laila Albalkhi

Jonah Niklas Bjørgsvik Wiecek

lalbalkh | 1968154 | albalkhl@uwindsor.ca

jwiecek | 1966868 | jwiecek@students.uni-mannheim.com

Contents

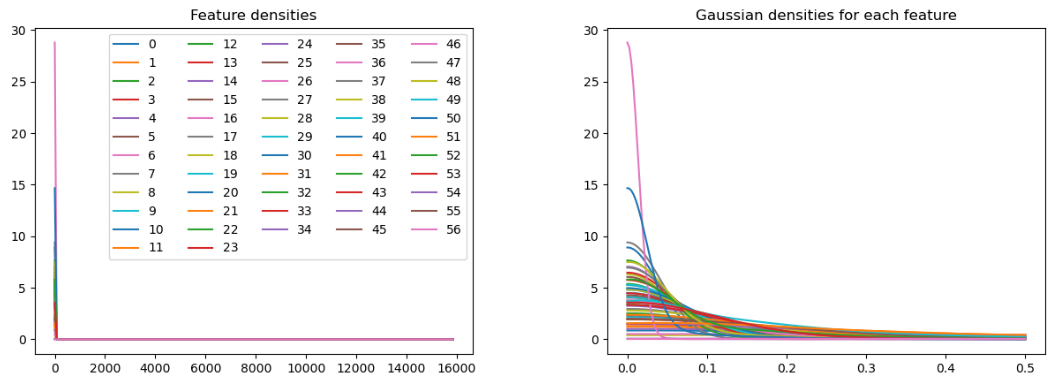


Figure 1.1: Kernel density plot and Gaussian density plot of all features.

1 Dataset Statistics

1.1 Kernel Density Plot

The kernel density plot shown in Figure 1.1 does not provide much information about the feature distributions or the values of their respective probability densities. We see a range of values that each feature can take on the x-axis and the probability density for that value plotted on the y-axis. In the figure on the left-hand side, we see that the maximum value of the dataset lies somewhere below 16000, and that the highest probability density for multiple features lies around a value of 0. For larger values In the figure on the right-hand side, we can truncate the x-axis and see a clearer picture of the Gaussian distributions plotted. We see that the highest probability densities lie somewhere between 0 and 0.2. For example, we see that the bright pink feature (feature 46) has a very high probability density for the value 0. Looking into the dataset details, we see that this feature describes the frequency percentage of the word "table".

When interpreting these figures, it's important to remember the structure and definition of the data itself. We know that the first 54 features are word frequency percentages; values from 0 to 100, but most word frequencies are under 10%. The last three features are length statistics, which explains the very large values we see in the data set. Through more exploration, we discover the largest value in the dataset to be 15841, which corresponds to the last feature describing the total length of uppercase letters. The minimum value is 0.0, describing a feature with no frequency percentage. The mean is approximately 6.1, and the standard deviation is extremely high at 93.5. This re-affirms that we have different features on different scales in our dataset.

From the interpretation of these figures, we can conclude that the majority of values for the features in our dataset lie between 0 and less than 1, and the probability density for features with extremely large values is very close to 0.

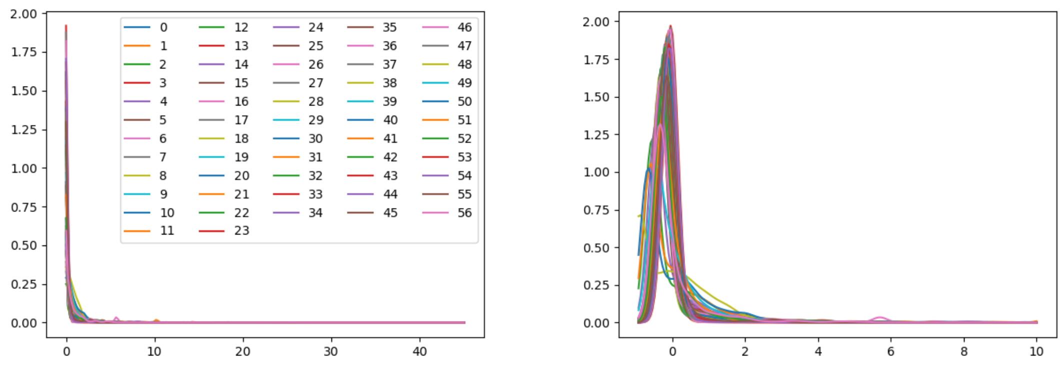


Figure 1.2: Kernel density plot of all normalized features.

1.2 Normalizing Data

We now normalize both our training and test data using z-scores to mean 0 and variance 1. When normalizing our test data, we use the statistics (in particular, the mean and standard deviation) calculated from the training data. This ensures that we don't over-fit our model to the test data, ensuring proper generalization and consistency within our model.

1.3 Normalized Kernel Density Plot

At first glance from the figure on the left, we don't see much of a difference in the kernel density plot. We can truncate the x-axis values again to see a clearer representation of each feature's distribution. We now see that there is less of a significant difference between the probability densities. We now have negative values in our dataset and the large values have decreased significantly.

We can explore this further and see that, as expected, our mean is now 0 and our variance is 1 for the training dataset. For the test dataset, we expect the mean and the variance to be around 0 and 1 but not exactly because we normalized using the mean and standard deviation of the training set. Looking at the mean vector for the testset we see that it indeed varies from -0.06 to 0.09.The variance for the different features ranges from values 0.05 to 2.75. This is as expected. We also see that the curves of the KDE's are a bit wider, so the peaks don't have to be as high for the area under the curve to be 1. This lets us see more details of the distribution.

2 Maximum Likelihood Estimation

2.1 Using a Bias Term

Let's recall that in logistic regression models, we assume that the logarithmic odds on our y values are linear functions of our x values. Furthermore, linear functions can be described by all the parameters $w_0, ..., w_D$. We define η as the following:

$$\eta = w_0 + w_1x_1 + w_2x_2 + ... + w_Dx_D \tag{1}$$
$$= w_0 + \sum_{j=1}^{D} w_jx_j \tag{2}$$
$$= w_0 + w^Tx \tag{3}$$
$$= w_0 + \langle w, x\rangle \tag{4}$$

where w_j is the weight for each corresponding feature x_j, and w_0 is the bias term. Note that we collected the parameters into a weight vector w to result in the dot product shown in equation (4). We now re-scale and shift the features by multiplying by a constant α and shifting by a constant δ.

$$w_0 + \langle w, \alpha x + \delta\rangle \tag{5}$$

We can use the bi-linearity property of the dot product here, and keeping in mind that α and δ are constants, we can perform some manipulation as follows:

$$= w_0 + \langle w, \alpha x\rangle + \langle w, \delta\rangle \tag{6}$$
$$= w_0 + \langle \alpha w, x\rangle + \delta w \tag{7}$$
$$= w_0 + \delta w + \langle \alpha w, x\rangle \tag{8}$$
$$= w_0' + \langle w', x\rangle \tag{9}$$

where $w_0' = w_0 + \delta w$ and $w' = \alpha w$. We've now shown that when we use a bias term, we can reconstruct the definition of η to give us a scaled weight matrix and a shifted bias vector. When we use this definition in our logistic regression calculations to find the maximum likelihood, we will get the same estimates.

The reason we computed z-scores is then to ensure that we have all features on the same scale, to improve the interpretability of the features. Now that the values have been normalized, we can better interpret the contribution of each value by their relative effect on our target variable. We have less of a discrepancy in the values of our features and can relate them to each other with more accuracy now that they lie on the same scale. In addition, normalizing our data will prove to be useful when we later perform L2 regularization on our model, since the scale of features can significantly affect the impact of regularization.

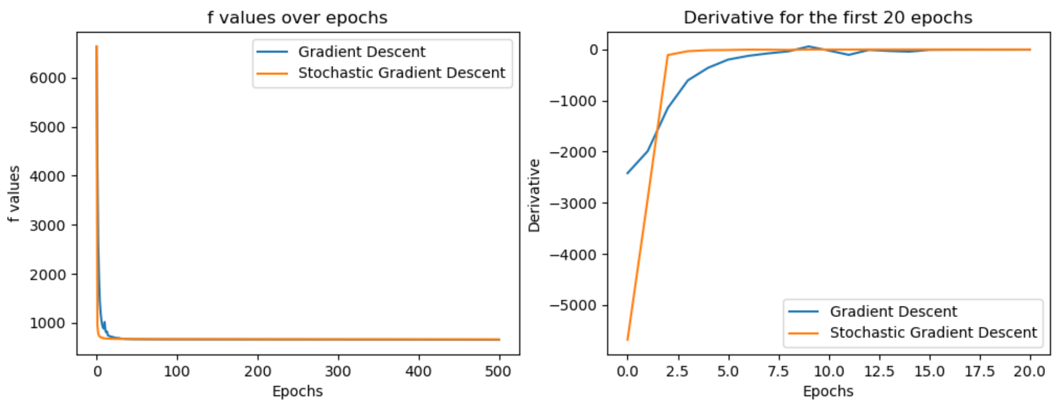


Figure 2.1: Gradient Descent vs. Stochastic Gradient Descent

2.2 Comparing Gradient Descent and Stochastic Gradient Descent

One of the first things we can notice about gradient descent (GD) and stochastic gradient descent (SGD) is their difference in run time. SGD takes much longer to compute because it must perform many more updates. SGD updates the parameters for each training example, whereas GD updates the parameters only once per iteration after computing the gradient using the entire dataset. SGD also takes longer to converge to the minimum than GD, because its path to the minimum value is more erratic than GD.

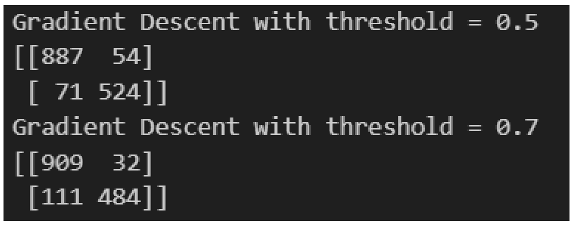
We plot the two methods side by side here in Figure 2.1. We see that there is a slight increase in the f-values in gradient descent and the bold-driver heuristic is visualized as the learning rate decreases before increasing again. We can take a closer look at the convergence rate by taking the derivative as shown in the figure on the right. This temporary peak is due to the fact that the direction of GD is based on the average of all gradients in the dataset, and in some cases, this leads to increases because the direction of the steepest descent may not decrease the loss function for all data points. As for SGD, the steepest descent varies from step to step, helping escape local minima and descending downward quickly.

3 Prediction

3.1 Choosing a Classification Threshold

From the context of our dataset, we need to carefully decide on a classification threshold to determine which emails we consider to be spam and which to label as legitimate. Intuitively, false positives (marking legitimate emails as spam) are very undesirable. Taking this into consideration, we can choose to maximize the recall of the negative class (not spam), effectively ensuring that we accurately identify all emails that are not spam. Alternatively, we can choose to maximize the precision of the positive class (spam), ensuring that of all emails we classify as spam, the majority are accurately classified as such.

We explore these threshold values in greater detail in Figure 3.1. We see that as the recall on the negative class increases, the overall accuracy and F1-score decrease. In order to get a good recall but without sacrificing too much accuracy, we decided on a threshold value of 0.7. This means that if a particular email has a probability of 70% of being spam, then we will classify it as such. Otherwise, we will risk misclassifying it as legitimate. We can see in the confusion matrix in Figure 3.1 that with this threshold value, we are more accurate in our classification of emails that are not spam, but we have more spam emails that pass through undetected. This choice is based on our intuitive reasoning but could be adjusted with a clearly-defined cost algorithm.



```
Gradient Descent with threshold = 0.5
[[887  54]
 [ 71 524]]
Gradient Descent with threshold = 0.7
[[909  32]
 [111 484]]
```

Figure 3.1: Confusion matrix with default choice of $threshold$ compared to 0.7

3.2 GD vs SGD

When comparing classification results, GD performs slightly better in recall for the negative class (not spam). In general, however, the two methods are very similar in comparison. For our choice of threshold, only one more email got classified the wrong class in SGD compared to GD.

Another performance metric is to study the ROC curves of the respective models. GD has a higher area under the ROC curve than SGD. In other words; for the same choices of threshold we could get higher precision and recall. This means that GD is better at distinguishing between the two classes. By studying the area under the ROC curves we can find a threshold that maximizes both recall and precision. Doing this gives us an optimal choice of $threshold = 0.456$. But we keep our choice of 0.7 since we know that there are greater consequences for classifying legitimate emails as spam and vice versa. We could find an "analytical" answer to this problem if we had a specified cost matrix.

3.3 Composition of the Weight Vector

We interpret the positive and negative weights as contributing to the probability of the email being classified as spam or not spam, respectively. Large positive weight values contribute to spam emails and large negative weight values contribute to legitimate emails. We can look at the peaks of the

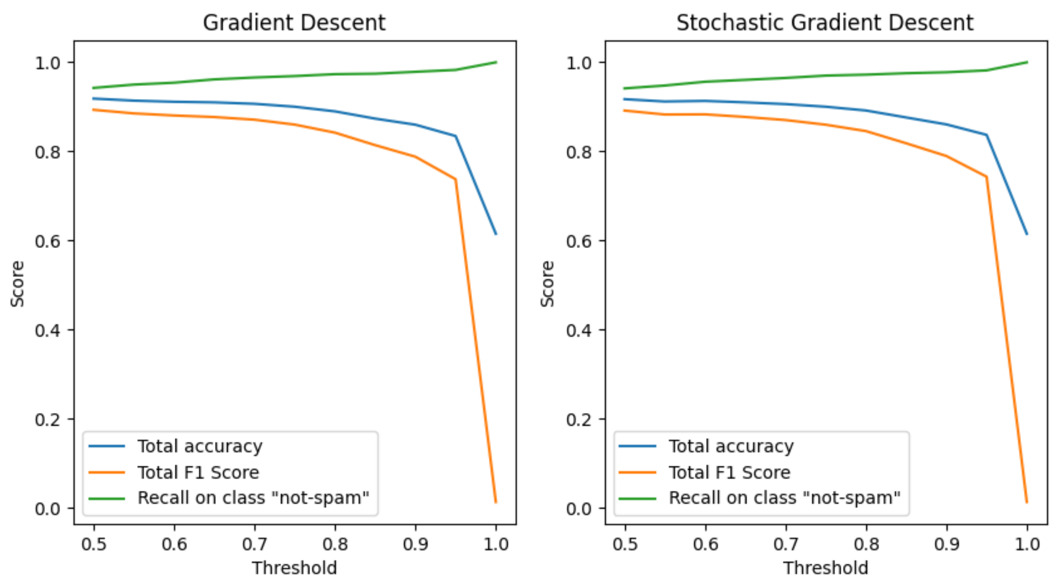


Figure 3.2: Choosing an optimal threshold for classification.

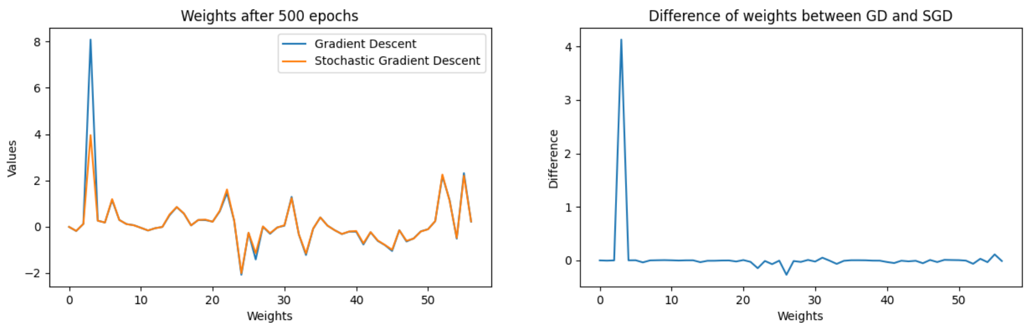


Figure 3.3: Weight comparison using GD vs SGD.

graph in Figure 3.2 to see which weights have the largest values, both positive and negative. We see that in GD, the weights have higher absolute values than in SGD. In other words, SGD adds a form of smoothing to the weights. This is because calculating the SGD uses an approximation of the gradient. We will analyze the weights using the results from GD in this section. We see that the average weight across all features is 0.19, and we will use this average to compare the contribution of each feature. We see a fairly even distribution between positive and negative weights.

We first take a look at the largest positive values of weights; the features that most contribute to an email being classified as spam. The feature that represents the frequency percentage of the word "*3d*" has the highest weight at approximately 8.09. This isn't entirely intuitive, but we can justify this by thinking of spam emails that boast free 3D products or services. The next largest weight is attributed to the feature describing the longest number of consecutive capitalized letters, with a value of 2.31. This is intuitively associated with spam emails that have a long string of capitalized letters, aimed to grab the attention of the recipients. We also have large weights associated with frequencies of characters and numbers, namely *$, 000, 857*, and *#*. Spam emails are often about money, which makes these weights intuitive.

We now take a look at the largest negative values; the features that contribute to legitimate emails. The two smallest weights are attributed with the features that describe the frequency percentage of the words "*hp*" and "*george*", with values -2.08 and -1.42 respectively. This is intuitive, as the training data has personal identifiable information, potentially regarding an employee named George working at Hewlett-Packard. Other word frequencies with negative weights were "*edu*", "*meeting*", "*conference*", and "*project*". These are also intuitive as they are professional topics typically found in legitimate email conversations.

4 Maximum Aposteriori estimation

4.1 Effect of λ

We tested λ ranging from 0 to 1000. Both the training- and test likelihood were strictly decreasing, which makes sense since the regularization is just subtracting a term from the un-regularized likelihood, and this term is linearly proportional with λ. The interesting thing, however, is that there are some values of λ (of the ones we tested) that have a higher prediction accuracy than the previous value tested. Here are the findings:

$$\lambda = 0.1 \text{ has a higher prediction accuracy than } \lambda = 0.0 \tag{10}$$

$$\lambda = 1 \text{ has a higher prediction accuracy than } \lambda = 0.9 \tag{11}$$

$$\lambda = 7 \text{ has a higher prediction accuracy than } \lambda = 6 \tag{12}$$

$$\lambda = 9 \text{ has a higher prediction accuracy than } \lambda = 8 \tag{13}$$

$$\lambda = 50 \text{ has a higher prediction accuracy than } \lambda = 40 \tag{14}$$

These small increasing steps can also be seen in Figure 4.1. This change can be interpreted as a smoothing effect that led to a change in weights, that in turn generalized better on the test data. In Figure 4.1the max accuracy is marked with a cross, so given our threshold value of 0.7, the highest prediction accuracy was with a value of λ of 0.1.

This lambda value is slightly surprising.[1] While studying the weights, we discovered that the most important feature for spam classification was the frequency of the word "*3d*", and word frequencies

[1]See section 4.3. After exploring different thresholds we realize why lambda is so small here

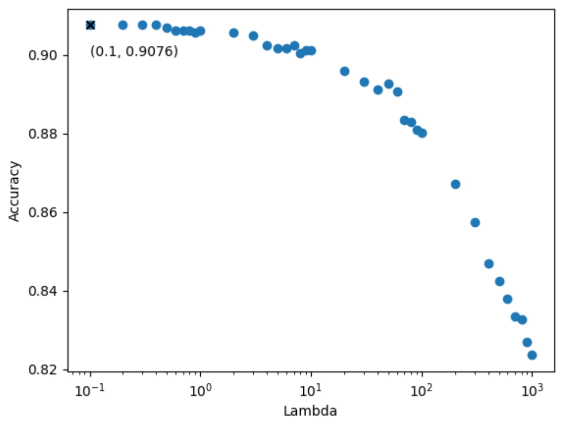


Figure 4.1: Prediction accuracy over different values of λ.

of "*hp*" and "*george*" contributed to legitimate emails. This initially felt very specific and potentially over-fitted to the training data, implying that adding a prior would yield better results on unseen data. Since all of the weights lie in the interval $[-3, 9]$, it makes sense that we don't need a huge penalty term. However, it was quite surprising that the optimal lambda was so close to 0. This can be interpreted as an indication of high similarity between the test data and the training data.

4.2 Weight Vector with Varying λ Values

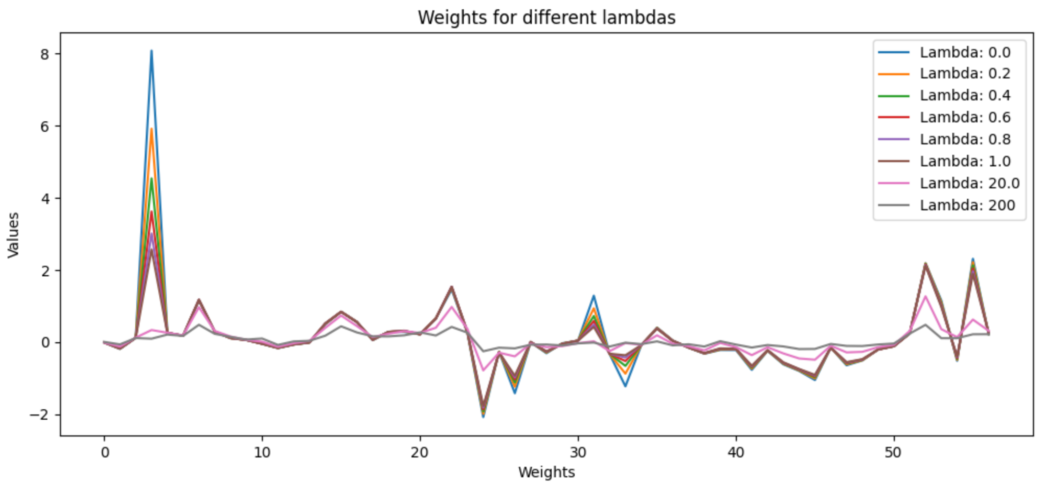


Figure 4.2: Smoothing of weights with increasing values of λ.

When varying choices for λ, we effectively scale down the weight vector, pushing the weights closer to 0. This can be seen in Figure 4.2 where the weights of $\lambda = 200$ can be seen in grey. This line still represents the same peaks as the original weight distribution in a compressed manner. Choosing an even larger value of $\lambda = 2000$ for instance, would plot the weights on the graph in nearly a straight line. We can see that the distribution of the weight vector changes in Figure 4.3, where we compare weights trained with regularization with a rather high penalty to the weights without

l_2 regularization. We see that the maximum and minimum values have the same features, but the differences between peaks or valleys respectively are much smaller in scale. And of course, the absolute differences are much smaller since the range of the weights is much smaller as well.

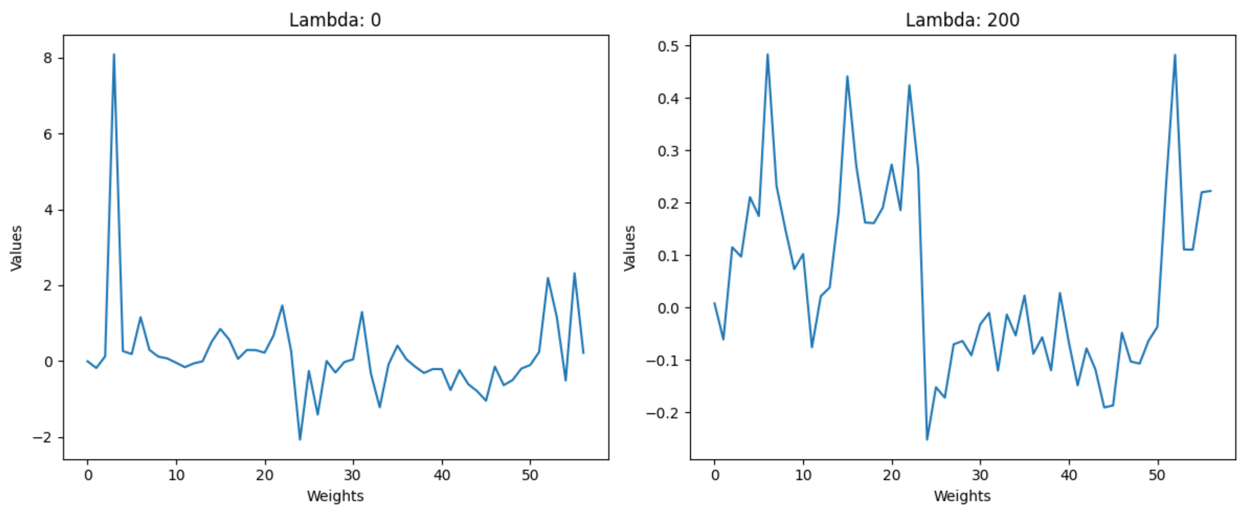


Figure 4.3: Weight composition with and without l_2 regularization.

4.3 Exploration: Optimal λ for Different Thresholds?

The change of the weights when trained with normalization is of course independent of the threshold. The threshold is only used for selecting the label in classification. However, we discovered that for a threshold of 0.5, we actually get better results with a higher regularization than when we had a threshold of 0.7. The best accuracy for the model with threshold 0.5 is when $\lambda = 4$.

This made us realize we won't get good results with regularization when we have a high threshold because using regularization or a prior will result in the weights being smaller. If the weights are smaller, then the linear combination of the input and weights, η, will also be smaller. Lastly, our probabilities from the sigmoid function will also be smaller. That means regularization will basically push the probabilities closer to the center, which is 0.5.

This discussion regarding optimal λ values needs to be done with cross-validation on the training set only, because if we select a λ that optimizes performance on the test set, we are over-fitting. Nonetheless, it is interesting to observe what the optimal value would be and try to interpret the intuition behind it.