Edgar R. Chavez

02/03/25

Machine Learning – CSE6363

Assignment 1 Report

Linear Models

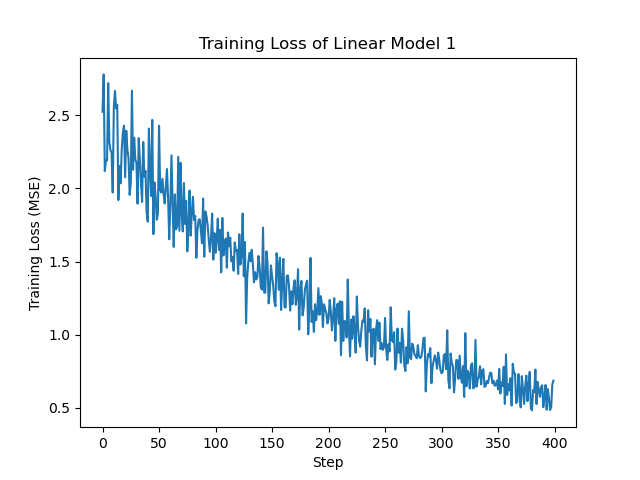
In this assignment we are tasked with implementing linear models for regression and classification for the iris dataset with four features: sepal length, sepal width, petal length and petal width. For linear regression we select four different combinations of input and output features to use to train four different models. To observe the effects of regularization, we pick one of the trained models, inspect the weights and train an identical model again with L2 regularization. Lastly create a single model that predicts the petal length and width given the sepal length and width. For each model created we test its performance on unseen data from the best trained weights.

For logistic regression we need to compare 3 variants of input features: petal length/width, sepal length/width and all the features. For the first two we include visualizations of the classifier using plot\_decision\_regision. For each trained model compute the accuracy on the test set, these should load the best trained weights.

The iris dataset was split training and testing sets, 10% is used for our test sets. Our iris dataset is also standardized using scikit-learn standard scaler.

**Linear Regression Model 1:**

The first model used three features: sepal length, sepal width, and petal length to predict the petal width. From the training loss we can see that that the loss is steadily decreasing. We had a training loss of approximately 0.6857 and a validation loss of 0.8674. The training score using MSE is 0.6036340242887303. With weights of [ 0.38537754 -0.10863772 0.5967069 ] and a Bias of 0.6932852685445741. When evaluating the test set, we get a score of 0.834933332930648, the gap between our test score and train score shows slight overfitting.



**Linear Regression Model 2:**

The second model used two features: sepal length and sepal width to predict the petal length. From the training loss we can see a slight decrease in the loss with some variation. We had a training loss of 0.18556179620965257 and a validation loss of 0.33439036405077943. The training score using MSE is 0.24582333609898524. With weights of [ 0.70417024 -0.26015478] and a bias of 0.29213645005675964. When evaluating the test set, we get a score of 0.2770098789581659, this shows that our model generalized well to unseen data given this combination of features.

**A graph of a training loss of a model

Description automatically generated**

**Linear Regression Model 3:**

The third model used one feature: sepal length to predict the sepal width. From the training loss we see that our loss is not decreasing which could be a sign that our feature was not a good choice. We had a training loss of 1.1603336717952353 and a validation loss of 1.27941778147726. The training score using MSE is 1.0648831439325712. With weights of 0.1666793508171112 and a bias of -0.050994633756714765. When evaluating the test set we get a score of 0.7227832152180904, in this case our model could be underfitting since our choice of feature did not tell us enough information.

**A graph of a training loss of linear model

Description automatically generated**

**Linear Regression Model 4:**

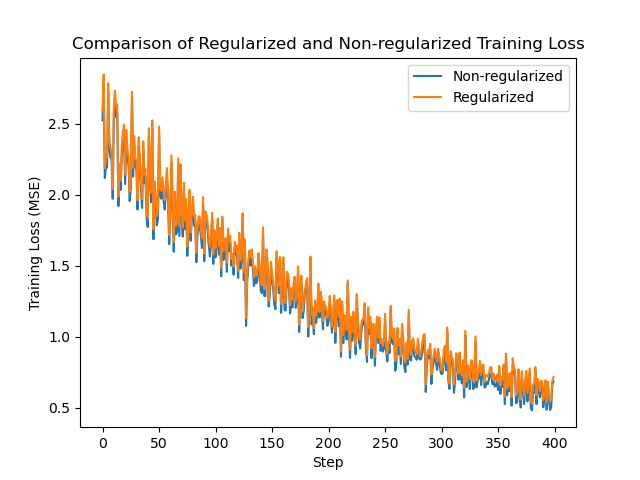
The fourth model used three features: sepal length, petal length and petal width to predict the sepal width. From the training loss we see that our is steadily decreasing though the loss is still generally high. We had a training loss of 1.4229523447193104 and a validation loss of 1.5838882439299429. The training score using MSE is 1.2312330560779607. With weights of [ 0.25323606 -0.60209254 0.20579298] and a bias of 0.6917314748975284. When evaluating the test set, we get a score of 0.7964004770141327, from this again we could determine that our model is underfitting and based on the previous model and this model trying to predict the sepal length does not yield good results.

A graph of a training loss of a model

Description automatically generated

**Linear Regression with Regularization Model:**

The regularized model uses the same features as our first model. From the loss comparison graph we see that both non-regularized and regularized models are very similar. Model weights with regularization: [ .37190483 -0.10667485 0.57030412] and model weights without regularization: [ 0.38537754 -0.10863772 0.5967069 ] Model bias with regularization: 0.6930755163090524 and model bias without regularization: 0.6932852685445741. Model score with regularization: 0.60133917772057 and model score without regularization: 0.6036340242887303.



**Linear Regression with Multiple Outputs Model:**

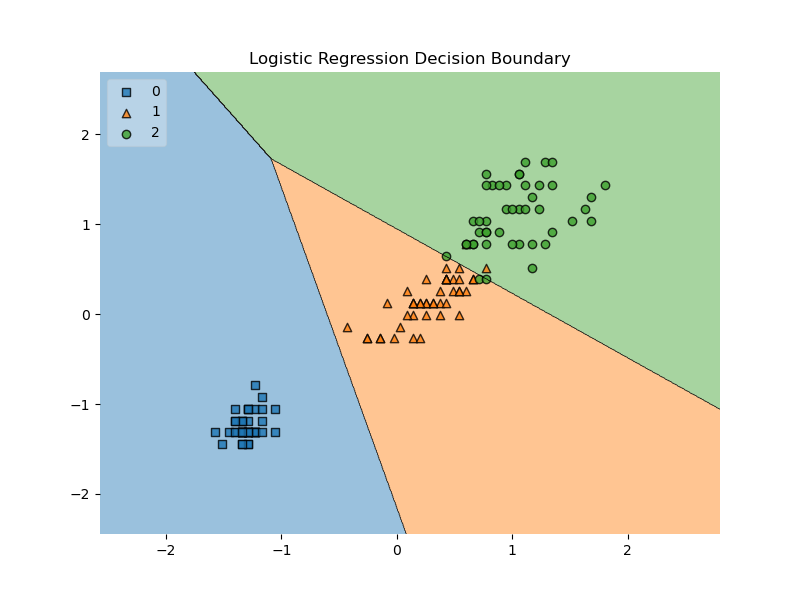
The multiple outputs models use the sepal length and sepal width to predict petal length and petal width. From our loss function we see that our loss is decreasing. Our training loss is 0.6127464601674029 and our validation loss is 0.6941568702310914. The training score using MSE is 0.7052970901007217. The weights are [[0.72417236 0.45045464] [0.10506063 0.51570058]] and the bias is [-0.10814122 -0.10940386].

A graph with blue lines

Description automatically generated

**Logistic Regression Model 1:**

The first model used two features: petal length and petal width to classify each flower in one of three different iris flowers. From both the loss graph and the decision boundary we see that model was able to train well and accurately classify the samples. We have a training accuracy of 0.9703703703703703 and a training loss of 0.24806895107897664. Our model weights are [[-2.24180552 0.75287497 2.49506894] [-1.01838849 -0.178327 2.25145502]] and bias is [ 0.18586946 1.99738764 -0.30608394].

 A graph with a line

Description automatically generated

When evaluating the test set we see that our model was able to generalize well to unseen data with a test accuracy of 0.9333333333333333

A diagram of a logistic regression decision boundary

Description automatically generated

**Logistic Regression Model 2:**

The second model used two features sepal length and sepal width to classify each flower sample in one of three different iris flower categories. From both graphs we see that are loss is decreasing and we are able to classify one type of flower extremely accurately, but it’s the other two classes that the model seems to struggle with. We have a training accuracy of 0.8222222222222222, and a training loss of 0.45727769375713145. Our model weights are [[-1.93320459 0.78275783 2.15658514] [1.85920766 -0.57321229 -0.23125584]] and bias are [0.03743264 1.057673 0.78206752]

A diagram of a logistic regression decision boundary

Description automatically generated A graph with blue lines

Description automatically generated

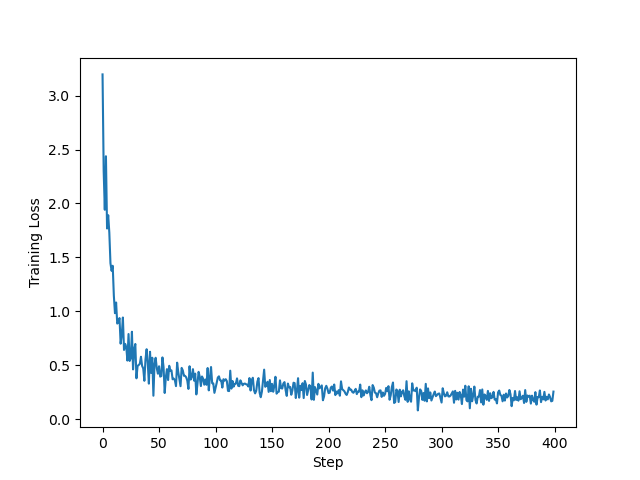
When evaluating the test set we see that our model does generalize well to one of the classes but not so well to the other two classes. This brings our test accuracy to a low 0.6666666666666666.

A diagram of a logistic regression decision boundary

Description automatically generated

**Logistic Regression Model 3:**

The third model used all the features in our dataset to predict the type of iris flower. From our loss graph we a steady decrease in the loss. Overall we have a training accuracy of 0.9407407407407408. Our model weights are [[-0.95246985 0.72218691 1.23642133] [ 1.76451865 -0.39854137 -0.31123776] [-0.44763989 1.02783297 1.29698008] [-1.43412086 -0.94060643 1.98813988]] and the bias are [-1.37641946 -0.11035212 -1.90946423]. When evaluating our test set we have a test accuracy of 0.8666666666666667.



**Questions:**

1. **What are the pros and cons of using the normal equations to solve for the weights in linear regression as opposed to using gradient descent?**

Some benefits of using the normal equations is that it gives the exact solution without having to iteratively update anything since no hyperparameters are needed. The formula is straightforward being but as our dataset increases this method uses a lot of memory and becomes impractical. This approach does work well with small datasets.

1. **Why is the softmax function used in multi-class logistic regression (Hint: the model itself produces logits?)**

Softmax function is to take the raw output values (logits) that are produced by the model and transforms them into a probability distribution across all the classes. Also ensures that these probabilities sum to 1.