Machine Learning Lab Report

Chi Zhau Han

April 30, 2024

1 Gradient of Loss Function and Update Equations

Starting with the hinge loss with l2 regularisation loss function:

$$O = C\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0)) + \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$\mathbf{w}$: is the vector of weights to be trained in the model.
w_0 : is the bias term.
C : This is the regularization hyperparameter.
y_i : This is the true label for the i-th data point.
$\mathbf{x}_i$: This is the feature vector for the i-th data point.

The partial derivative of the hinge loss function is:

$$\frac{\partial}{\partial \mathbf{w}} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0)) = \begin{cases} -y_i\mathbf{x}_i & \text{if } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and for the biased term:

$$\frac{\partial}{\partial w_0} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0)) = \begin{cases} -y_i & \text{if } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and the l2 regularisation term:

$$\frac{\partial}{\partial \mathbf{w}}\mathbf{w}^\mathbf{T}\mathbf{w} = \mathbf{w}.$$

Combining the results we have the gradient of the objective function is

$$\frac{\partial O}{\partial \mathbf{w}} = C\sum_{i=1}^{N} \begin{cases} -y_i\mathbf{x}_i & \text{if } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0) > 0 \\ 0 & \text{otherwise} \end{cases} + \mathbf{w}$$

and the gradient with respect to bias term:

$$\frac{\partial O}{\partial w_0} = C\sum_{i=1}^{N} \begin{cases} -y_i & \text{if } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + w_0) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Finally, the update equations are

$$\mathbf{w} := (\mathbf{w}) - \eta\frac{\partial O}{\partial \mathbf{w}} \qquad w_0 := w_0 - \eta\frac{\partial O}{\partial w_0}$$

η : is the learning rate

2 Intepreting Performance over Iterations

Looking at Figure 1, The observed large decrease in the training objective function, or loss, indicates effective learning during the initial stages of training. The 'zigzag' pattern suggests a potential overshooting of the minimum point due to a high learning rate. However, the diminishing amplitude of these oscillations, coupled with a decreasing loss, indicates convergence towards a local minimum.

After approximately 50 iterations, the loss stabilizes around 65 (from print statement), a significant reduction from the initial value exceeding 850. This stabilization suggests that the model has effectively learned and minimized the error in predictions.

The inverse relationship between loss and accuracy is evident in the accuracy per iteration figure. As the loss decreases, the model's accuracy increases, indicating that the model is improving its predictions over iterations.

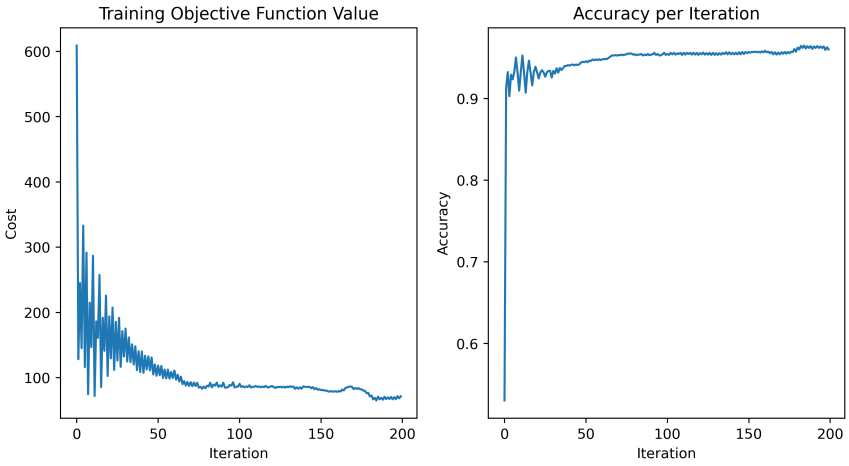

Figure 1: Objective Function and Train Accuracy during Training

2.1 Definition and Indication of Training Accuracy and Testing Accuracy

Training Accuracy: The accuracy of the model on the same data it was trained on. A high accuracy would indicate that a model has been trained to identify patterns of the training data well.

Testing Accuracy: The accuracy of the model on unseen data. A high testing accuracy indicates that a model is able to generalize well from the training data to new data.

3 Effect of η on Model Training

The learning rate η significantly influences the model's performance and the pattern of convergence. For larger η (10 and 1), the model converges quickly but exhibits a 'zigzag' pattern due to overshooting, resulting in test accuracies of 0.62 and 0.89 respectively.

As η decreases (0.1, 0.01, 0.001), the 'zigzag' pattern diminishes, and the convergence becomes smoother, indicating less overshooting. This results in improved stability and accuracy, with test accuracies of 0.93, 0.95, and 0.97 respectively.

For very small η (0.0001, 0.00001, 0.000001), the model's convergence significantly slows down, especially when $\eta = 0.000001$ the graph becomes almost a straight line, indicating a very smooth but slow optimization process. However, the test accuracy decreases to 0.95, 0.58, and 0.44 respectively.

In conclusion, a balance is needed between the speed of convergence and the stability of the model. For 200 iterations, $\eta = 0.001$ provides the highest test accuracy (0.97) with a reasonable convergence rate and a smooth convergence pattern.

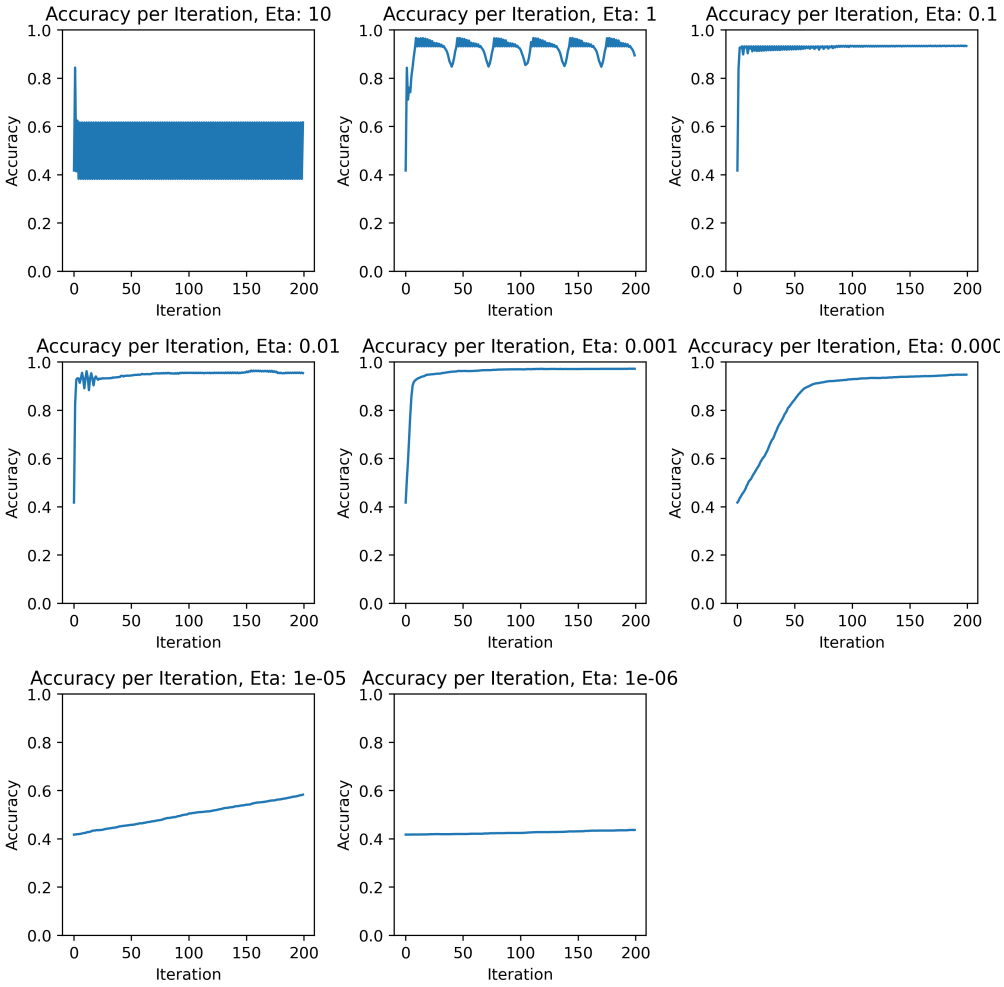

Figure 2: Training Accuracies over Iterations with Different η

4 Conclusion on Model Selection

Based on the model selection process using 5 fold cross validation, these conclusions are drawn. For the regression task, the model performs optimally with the Relu activation function and a hidden layer size of (100,). This configuration yields a mean R2 score of approximately 0.907, indicating a high degree of predictive accuracy. The low standard deviation of 0.0136 suggests that this configuration provides consistent performance across different subsets of data.

Conversely, the worst-performing model configuration uses the 'logistic' activation function and a hidden layer size of (3,3). This setup results in the lowest mean R2 score of approximately 0.642, indicating a significantly lower predictive accuracy compared to the optimal configuration.

Activation	Hidden Layer Sizes	Mean Score	Standard Deviation
relu	(3,)	0.889	0.013
relu	(100,)	0.905	0.010
relu	(3, 3)	0.607	0.560
relu	(100, 100)	0.908	0.014
logistic	(3,)	0.755	0.046
logistic	(100,)	0.894	0.007
logistic	(3, 3)	0.734	0.067
logistic	(100, 100)	0.893	0.006

Table 1: Grid Search CV Results

5 Comparison of Adam and SGD Algorithm

In terms of performance metrics, Adam outperforms SGD. Specifically, Adam achieves a lower Mean Squared Error (MSE) of 0.144 compared to SGD's 0.152, and a higher R2 score of 0.923 compared to SGD's 0.913 on the test set.

Based on Figure 3, In terms of convergence, Adam's loss curve also shows a steeper decrease, indicating faster convergence than SGD. However, Adam's loss curve also exhibits more fluctuations, which I think might be attributed to some variable learning rate.

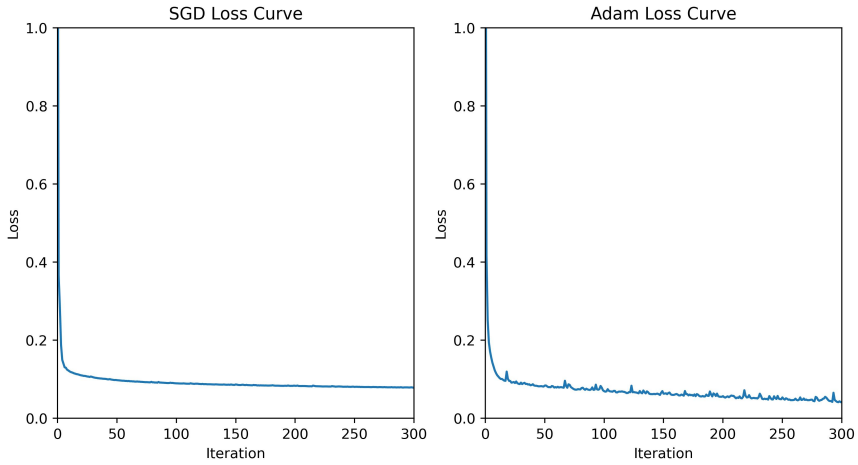

Figure 3: Loss Curve of ADAM and SGD algorithm

6 Developing a Robust MLP Regressor

During model development, I used 4-fold cross-validation to optimize the model. I focused on imputing missing data, reducing noise of data, and the architecture of the Multi-layer perceptron. For imputers, I considered simple imputer with both mean and median strategy, and 4 nearest neighbour imputer. For Reducing noise I compared the impacts between standard scaler and robust scaler. Lastly for the architecture, I considered tested two different hidden layer sizes: (50,50) and (25,25). I also compared the 'relu' and 'logistic' activation functions. I used the Adam algorithm as the search strategy, based on its superior performance over SGD based on the previous section.

The optimal configuration was found to be a Simple Imputer with median strategy, 'relu' activation, an MLP with hidden layer sizes of (50,50), and the Standard Scaler.

Mean Score	Std Deviation	Imputer	MLP Activation	MLP Hidden Layers	Scaler
-0.1803	0.0296	KNNImputer(4)	relu	(50, 50)	StandardScaler
-0.1803	0.0210	KNNImputer(4)	relu	(50, 50)	RobustScaler
-0.1771	0.0220	KNNImputer(4)	relu	(25, 25)	StandardScaler
-0.1791	0.0197	KNNImputer(4)	relu	(25, 25)	RobustScaler
-0.1963	0.0155	KNNImputer(4)	logistic	(50, 50)	StandardScaler
-0.1947	0.0161	KNNImputer(4)	logistic	(50, 50)	RobustScaler
-0.1991	0.0185	KNNImputer(4)	logistic	(25, 25)	StandardScaler
-0.1992	0.0162	KNNImputer(4)	logistic	(25, 25)	RobustScaler
-0.1700	0.0249	SimpleImputer	relu	(50, 50)	StandardScaler
-0.1750	0.0213	SimpleImputer	relu	(50, 50)	RobustScaler
-0.1772	0.0184	SimpleImputer	relu	(25, 25)	StandardScaler
-0.1859	0.0148	SimpleImputer	relu	(25, 25)	RobustScaler
-0.1963	0.0185	SimpleImputer	logistic	(50, 50)	StandardScaler
-0.1972	0.0160	SimpleImputer	logistic	(50, 50)	RobustScaler
-0.1977	0.0180	SimpleImputer	logistic	(25, 25)	StandardScaler
-0.1997	0.0185	SimpleImputer	logistic	(25, 25)	RobustScaler
-0.1728	0.0228	SimpleImputer(median)	relu	(50, 50)	StandardScaler
-0.1813	0.0309	SimpleImputer(median)	relu	(50, 50)	RobustScaler
-0.1777	0.0227	SimpleImputer(median)	relu	(25, 25)	StandardScaler
-0.1827	0.0223	SimpleImputer(median)	relu	(25, 25)	RobustScaler
-0.1936	0.0154	SimpleImputer(median)	logistic	(50, 50)	StandardScaler
-0.1967	0.0155	SimpleImputer(median)	logistic	(50, 50)	RobustScaler
-0.1974	0.0167	SimpleImputer(median)	logistic	(25, 25)	StandardScaler
-0.1983	0.0189	SimpleImputer(median)	logistic	(25, 25)	RobustScaler

Table 2: Grid Search CV Results for MLP Model

In the final optimization stage, I varied the regularization constant (alpha) and found the best constant to be alpha = 0.1, using Grid Search.

Mean Score	Std Deviation	alpha
0.9090	0.0103	0.0001
0.9075	0.0128	0.001
0.9087	0.0137	0.01
0.9097	0.0110	0.1
0.9062	0.0094	1.0

Table 3: Grid Search CV Results for Different Alpha Values