Problem Set 2: Linear Classifiers and Gradient Descent

Po Yu Chen

October 14, 2024

1 Introduction

The objective of this assignment is to implement a linear classifier using Support Vector Machine (SVM) with hinge loss and a Softmax classifier with cross-entropy loss. We calculate the score function, apply L2 regularization to reduce overfitting, and use gradient descent to optimize the loss. The classifier is tested on a spiral dataset, which highlights the challenges of linear models in complex classification tasks.

2 Methodology

2.1 Score Function Implementation

The score function is the core component of the linear classifier that computes the raw class scores for each input data point. It is defined as:

$$\mathbf{S} = \mathbf{X}\mathbf{W} + \mathbf{b} \tag{1}$$

Where:

- $\mathbf{X} \in R^{N \times D}$ is the input data matrix, where N is the number of examples and D is the number of features.
- $\mathbf{W} \in R^{D \times K}$ is the weight matrix, where K is the number of classes.
- $\mathbf{b} \in R^{K}$ is the bias vector for each class.
- $\mathbf{S} \in R^{N \times K}$ is the matrix of raw scores for each class.

The score function computes the dot product between the input features and weights, and adds the bias, resulting in a set of scores for each class for every data point.

2.2 Hinge Loss

The hinge loss is used in SVM to create a margin between the correct class and incorrect classes, ensuring that the correct class score is higher by a margin. The hinge loss for a given training example is calculated as:

$$L_i = \sum_{j \neq y_i} \max(0, S_j - S_{y_i} + \Delta) \tag{2}$$

Where:

- S_j is the score for class j.
- S_{y_i} is the score for the correct class y_i.
- Δ is the margin parameter, typically set to 1.

The total hinge loss over all training examples without regularization is:

$$L_{\text{Hinge}} = \frac{1}{N} \sum_i L_i \tag{3}$$

This loss function encourages the correct class score to be at least one unit greater than the incorrect class scores, thus maximizing the margin between classes.

2.3 Softmax Loss

The Softmax classifier uses the cross-entropy loss, which measures the difference between the predicted probabilities and the true labels. First, the scores are converted to probabilities using the softmax function:

$$P_j = \frac{e^{S_j}}{\sum_k e^{S_k}} \tag{4}$$

Where:

- S_j is the score for class j.
- P_j is the probability that the sample belongs to class j.

The cross-entropy loss for a given training example is:

$$L_i = -\log(P_{y_i}) \tag{5}$$

Where P_{y_i} is the probability assigned to the correct class y_i. The total softmax loss over all training examples without regularization is:

$$L_{\text{Softmax}} = \frac{1}{N} \sum_i L_i \tag{6}$$

This loss encourages the classifier to assign high probabilities to the correct classes, which leads to improved accuracy.

2.4 Task 4: Regularization

To prevent overfitting, we applied L2 regularization to both the hinge loss and softmax loss functions. Regularization works by adding a penalty to the loss function that depends on the magnitude of the model weights. In our implementation, the L2 regularization term penalizes large weight values, helping to keep the model simple and ensuring that it generalizes well to unseen data. This regularization term was crucial in avoiding overfitting, especially given the complexity of the spiral dataset used for training.

2.5 Task 5: Gradient Descent

We optimized both classifiers using the gradient descent algorithm. Gradient descent is an iterative optimization method that updates the weights and biases of the model in the direction that reduces the loss function. In each iteration, we calculated the gradient of the loss with respect to the model parameters and used these gradients to adjust the parameters. The learning rate was an important hyperparameter in this process, controlling the size of each step taken. By carefully tuning the learning rate and applying gradient updates over multiple iterations, we were able to minimize the loss functions for both classifiers and achieve better performance on the dataset.

3 Results

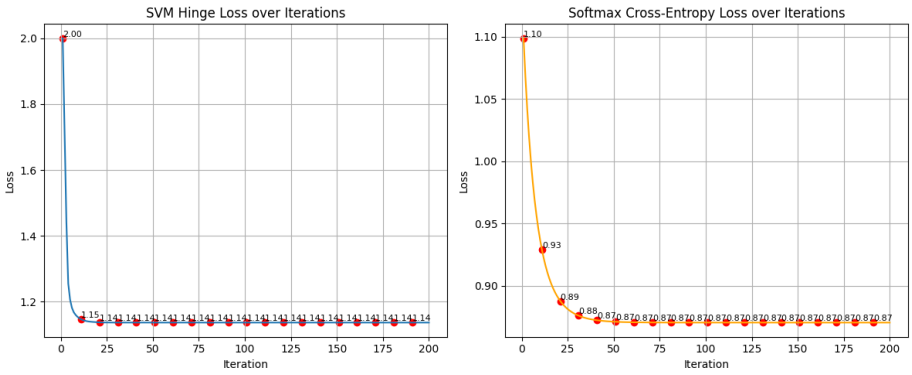

Figure 1: Loss over Iterations for SVM and Softmax Classifiers

This chart shows the loss values over 200 iterations for both the SVM and Softmax classifiers during training. The SVM classifier experienced an initial rapid decrease in hinge loss, stabilizing after a few iterations to a consistent value around 1.41. This indicates that the model learned effectively within the first 20 iterations and remained stable afterward. The Softmax classifier's cross-entropy loss similarly showed a steep initial decline, reaching a plateau around

0.87 after approximately 30 iterations. This behavior suggests that the Softmax model quickly approached an optimal state and demonstrated a strong fit to the training data.

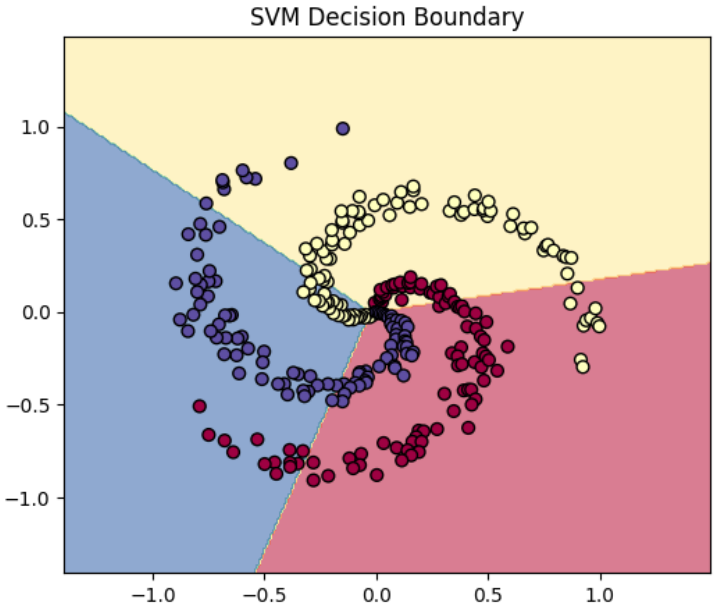

Figure 2: SVM Decision Boundary for Spiral Dataset

The SVM classifier generated linear decision boundaries that partition the complex spiral dataset into three distinct regions. However, due to the inherent complexity of the spiral dataset, which is not linearly separable, the SVM's linear decision boundaries result in some misclassification of data points.

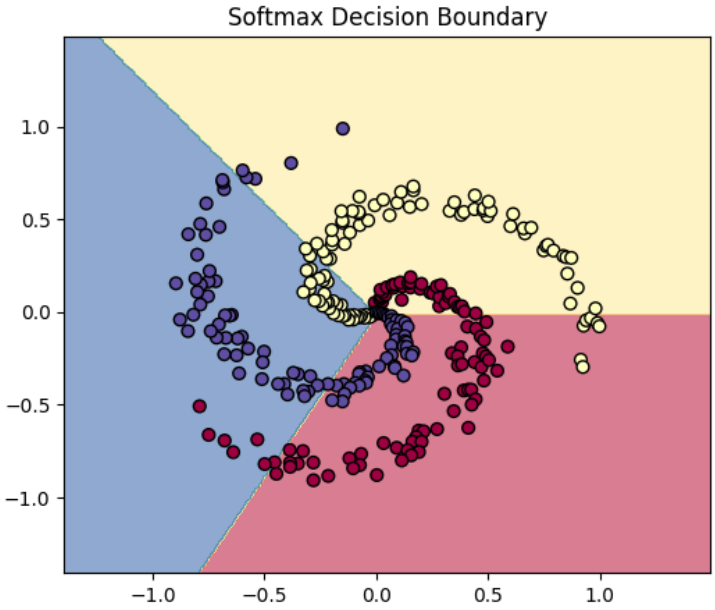

Figure 3: Softmax Decision Boundary for Spiral Dataset

The Softmax classifier also produces linear decision boundaries similar to those of the SVM. Given the non-linear and intertwined structure of the spiral dataset, these linear boundaries result in visible classification errors.

4 Conclusion

In this assignment, we successfully implemented SVM and Softmax classifiers using gradient descent, observing their strengths and limitations on a complex dataset. The Softmax classifier performed slightly better in terms of minimizing loss, but both models struggled with the non-linearly separable spiral data. A significant challenge was optimizing the learning rate and regularization to ensure convergence and prevent overfitting.