# Problem Set 2 : Gradient Descent Techniques for Linear Classifiers and Loss Optimization

Aman Dongre

October 2024

## 1 Introduction

This problem set focuses on optimizing linear classifiers like SVM and Softmax using gradient descent. Gradient descent is an iterative method that adjusts model parameters (weights and biases) to minimize the classification error measured by loss functions like hinge loss (SVM) and cross-entropy loss (Softmax). We also explore L2 regularization, which reduces overfitting by penalizing large weights, helping the models generalize better.

## 2 Methodology

- Implementation of Score function:
  The score function of a linear classifier maps an input feature vector $\mathbf{x}$ to a score that indicates how likely $\mathbf{x}$ belongs to a particular class. In a binary linear classifier, the score function can be written as:

$$\text{score}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

  Where:

  - $\mathbf{x}$ is the input feature vector,
  - $\mathbf{w}$ is the weight vector,
  - $b$ is the bias term (a scalar),

  - $\mathbf{w}^T\mathbf{x}$ is the dot product between the weight vector $\mathbf{w}$ and the feature vector $\mathbf{x}$.

- Describing the hinge loss and softmax loss :

## 2.1 SVM Classifier (Hinge Loss)

The hinge loss is used in SVM (Support Vector Machines). The loss for a given sample can be written as:

$$L(x_i, y_i) = \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

Where:

- $\mathbf{x}_i$ is the input sample,
- $y_i$ is the true label, which is either $+1$ or $-1$,
- $\mathbf{w}$ is the weight vector,
- $b$ is the bias,
- $\mathbf{w}^T\mathbf{x}_i + b$ is the score from the classifier.

The hinge loss only incurs a penalty when the correct class score does not exceed a margin of 1. For multiple samples, the total loss is averaged.

## 2.2 Softmax Classifier (Cross-Entropy Loss)

For the softmax classifier, the softmax function converts raw scores into probabilities, and the cross-entropy loss penalizes the deviation between the predicted probabilities and the true labels.

The softmax function for the score $z$ for class $i$ is:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

The cross-entropy loss is then:

$$L(x_i, y_i) = -\log(p_{y_i})$$

Where:

- $p_{y_i}$ is the predicted probability for the true class $y_i$.
- **Describing Regularization in Machine Learning :**

  Regularization is a crucial technique in machine learning that helps prevent overfitting, especially in models with a large number of parameters. By adding a penalty term to the loss function, regularization discourages complex models that may not generalize well to unseen data.

## 2.3 L2 Regularization

L2 regularization, also known as Ridge regularization, adds the squared magnitude of the coefficients as a penalty term to the loss function. The L2 regularization term is defined as:

$$R(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 = \frac{\lambda}{2}\sum_{j=1}^{n} w_j^2$$

Where:

* $\lambda$ is the regularization strength (hyperparameter) that controls the amount of regularization applied,
* $\mathbf{w}$ is the weight vector,
* $n$ is the number of features.

The overall loss function with L2 regularization can be expressed as:

$$L(\mathbf{x}, y) = L_{\text{original}}(\mathbf{x}, y) + R(\mathbf{w})$$

Where $L_{\text{original}}$ is the original loss function (e.g., hinge loss for SVM or cross-entropy loss for softmax).

**Benefits of L2 Regularization:**

* **Prevents Overfitting**: By penalizing large weights, L2 regularization helps the model to generalize better.
* **Smooth Solutions**: L2 regularization results in smaller and more evenly distributed weights, which often leads to smoother decision boundaries.

## 2.4 Other Regularization Techniques

While L2 regularization is widely used, other regularization techniques include:

* **L1 Regularization (Lasso)**: This technique adds the absolute value of the weights as a penalty, promoting sparsity in the model by driving some weights to exactly zero.
* **Elastic Net**: A combination of L1 and L2 regularization, which provides a balance between the two approaches.

In summary, regularization is an essential component of machine learning models, enhancing their ability to generalize to new data by preventing overfitting.

– **Explaining the Gradient Descent Method :**

## 2.5 Gradient Descent Method

Gradient descent is an iterative optimization algorithm used to minimize a loss function in machine learning models, particularly in the context of linear classifiers. The goal is to adjust the model parameters to reduce the error between predicted and true values.

## 2.6 Mathematical Formula

Given a loss function $L(\mathbf{w})$ dependent on the model parameters $\mathbf{w}$, the update rule for gradient descent can be expressed as:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$$

Where:

* $\mathbf{w}^{(t)}$ represents the model parameters at iteration $t$,
* $\eta$ is the learning rate, a hyperparameter that determines the size of the steps taken towards the minimum,
* $\nabla L(\mathbf{w}^{(t)})$ is the gradient of the loss function with respect to the model parameters.

## 2.7 Gradient Calculation

To compute the gradient, we take the derivative of the loss function. For example, in the case of the hinge loss used in SVM classifiers, the gradient with respect to the weights $\mathbf{w}$ can be calculated as follows:

$$\nabla L(\mathbf{w}) = -y_i \mathbf{x}_i \quad \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) < 1$$

Where:

* $y_i$ is the true label,
* $\mathbf{x}_i$ is the input feature vector.

For the softmax classifier using cross-entropy loss, the gradient can be computed similarly.

## 2.8 Iterative Process

The gradient descent process is repeated for a specified number of iterations or until convergence is reached, typically defined by a threshold for the change in the loss function or the parameters. The convergence criteria can be expressed as:

$$\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\| < \epsilon$$

Where $\epsilon$ is a small positive constant indicating the desired precision.

## 2.9 Conclusion

Gradient descent is a powerful and widely used optimization technique that facilitates the training of linear classifiers by effectively minimizing the loss function..

# 3 Results

Spiral Dataset Plot:

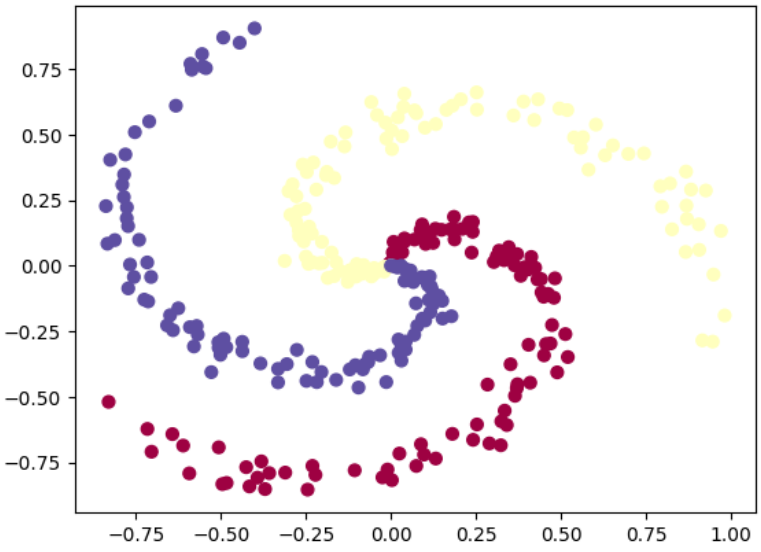


Figure 1: Spiral Dataset

**Task 1: Score Function :**
Linear Classifier Scores: [0.66 0.6 ]

**Task 2: Loss Functions :**
SVM Hinge Loss: 0.8586 Softmax Cross-Entropy Loss: 1.2250516039575707

**Task 3: Regularization :**
SVM Hinge Loss with L2 Regularization: 0.83266 Softmax Cross-Entropy Loss with L2 Regularization: 1.1290516039575709

**Task 4: Gradient Descent :**
Iteration 0: Loss = 0.83124593266
Iteration 10: Loss = 0.817072430822
Iteration 20: Loss = 0.802959598414
Iteration 30: Loss = 0.788906233513
Iteration 40: Loss = 0.774911157999
Iteration 50: Loss = 0.760973217094
Iteration 60: Loss = 0.747091278896
Iteration 70: Loss = 0.733264233928
Iteration 80: Loss = 0.719683751787

Iteration 90: Loss = 0.716525503862
Updated SVM Weights: [0.54308276 0.63281988]
Updated SVM Bias: 0.3867
Iteration 0: Loss = 1.128478627853
Iteration 10: Loss = 1.12284689827
Iteration 20: Loss = 1.11738834606
Iteration 30: Loss = 1.11209607806
Iteration 40: Loss = 1.10696360655
Iteration 50: Loss = 1.10198481535
Iteration 60: Loss = 1.09715392941
Iteration 70: Loss = 1.09246548745
Iteration 80: Loss = 1.08791431730
Iteration 90: Loss = 1.08349551368
Updated Softmax Weights: [[0.27441813 0.65848257] [0.09733377 0.81197434] [0.71399867 0.15816896]]
Updated Softmax Bias: [0.1296234 0.20476071 0.26561589]

# 4 Summary of Findings

In this problem set, we explored various aspects of linear classifiers and optimization techniques using gradient descent. Key findings include:

- We implemented the score function for a linear classifier, which computes the likelihood of an input feature vector belonging to a particular class.
- We derived and implemented the loss functions for Support Vector Machines (SVM) using hinge loss and softmax classifiers using cross-entropy loss.
- We incorporated L2 regularization into the loss functions to prevent overfitting and improve model generalization.
- Gradient descent was applied to minimize the loss functions, enabling us to update model parameters iteratively based on the computed gradients.

# 5 Challenges Faced

During the implementation of this problem set, we encountered several challenges:

- **Formatting Issues**: Managing the correct formatting in LaTeX was initially difficult. Ensuring that mathematical equations, figures, and sections were displayed correctly required careful attention to syntax and placement options.
- **Debugging the Code**: Ensuring the correctness of the implementation of loss functions and gradient descent required extensive debugging, especially in cases where unexpected results occurred.

In summary, this problem set provided valuable insights into the functioning of linear classifiers and the optimization techniques used in machine learning, while also highlighting the importance of meticulousness in LaTeX formatting and code implementation.