(PSL) Coding Assignment 5

Fall 2023

Contents

Introduction

In this assignment, you will be implementing a linear Support Vector Machine (SVM) classifier from scratch using stochastic gradient descent (SGD).

As we discussed in class, traditional SVMs often solve the dual problem, which involves a quadratic objective function subject to linear constraints. While this approach can be efficient for small-scale tasks, it becomes less practical for large-scale problems. In such cases, we can leverage the benefits of SGD to directly solve the primal problem.

The SGD Algorithm

The SGD algorithm works as follows:

1. Start by choosing a random initial value of parameters
2. Loop Over Epochs:
   - In each epoch, go through the entire dataset once. An epoch is a complete pass through all the training data.
3. Loop Over Data Points:
   - Within each epoch, iterate over each data point in your training dataset.
4. Update the Gradient:
   - For each data point, calculate the gradient of the loss function with respect to the current parameter values. This gradient represents the direction of steepest ascent.
5. Calculate Step Sizes:
   - For each parameter, calculate the step size as : step size = gradient * learning rate.
6. Update Parameters:
   - Update new parameters as : new params = old params - step size
7. Repeat Until Convergence:
   - Repeat steps 3 to 6 for each data point in the dataset. Continue this process for a fixed number of epochs or until convergence criteria are met.

Pegasos Algorithm

The Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) algorithm, proposed by Shalev-Shwartz et al. (2011) [Paper Link], is an application of SGD.

Recall that the primal problem of linear SVM can be expressed as the following the Loss + Penalty format:

$$\frac{\lambda}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i(x_i^t\beta + \alpha)\right]_+$$

where α is the intercept and β is the p-dimensional coefficient vector.

The **Pegasos Algorithm** can be summarized as follows:

1. initialize $\beta = 0_{p\times 1}$, $\alpha_1 = 0$, and $t = 0$
2. for *epoch = 1, 2, . . . , T* do
   - for i *= 1, 2, . . . , n* do
     - $t = t + 1, \quad \eta_t = \frac{1}{t\lambda}$
     - update $\beta_{t+1} \Leftarrow \beta_t - \eta_t\Delta_t$
     - update $\alpha_{t+1} \Leftarrow \alpha_t - \eta_t\delta_t$

Here η_t is the learning rate, and Δ_t and δ_t are the (sub)gradient of $J_i(\beta, \alpha)$ when β_t and $\alpha = \alpha_t$:

$$J_i(\beta, \alpha) = \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2 + \left[1 - y_i(x_i^t\beta + \alpha)\right]_+$$

$$\Delta_t = \begin{cases} \lambda\beta_t - y_i x_i & \text{if } \ y_i(x_i^t\beta_t + \alpha_t) < 1 \\ \beta_t & \text{otherwise} \end{cases}$$

$$\delta_t = \begin{cases} -y_i & \text{if } \ y_i(x_i^t\beta_t + \alpha_t) < 1 \\ 0 & \text{otherwise} \end{cases}$$

Test Your Function

- Implement the **Pegasos Algorithm**.
  - Use a fixed number of epochs, e.g., T = 20.
  - In each epoch, before going through the dataset, consider randomizing the order of the data points. To achieve this, you should set random seeds for shuffling. For this assignment, the seeds used for shuffling **do not need to be associated with your UIN**.
- Test your code with the provided training (400 samples) and test (600 samples) datasets, which are subsets of the MNIST data. Each dataset consists of 257 columns, with the first 256 columns representing the features, and the last column indicating the label (either 5 or 6).
  - [coding4_train.csv]
  - [coding4_test.csv]
- Report **confusion tables** on the **training and test** datasets.
- Your code should obtain **less than 15%** test error.