CSC 461 Homework 3 Report

Richard Burke, Neeraj Adhikari

December 1, 2019

1 Introduction

Our group was given the task of implementing a Multiclass Support Vector Machine (SVM) with Stochastic Gradient Descent (SGD) using PyTorch that would best classify the CIFAR-10 dataset. The dataset consists of 60,000 32×32 color images which each image belongs to one of ten predefined classes. The ten classes were as follows: *plane*, *car*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. Using PyTorch's dataset loader, the data was split as 50,000 images belonging to the training dataset and the remaining 10,000 belonging to the test dataset.

2 Our Solution

We both began by looking into some tutorials and the documentation for using Pytorch. Once we got a feel for some of the basics of Pytorch we extracted the training dataset and test dataset as separate tensors in mini-batch mode with a batch size of 4. So, each mini-batch was a 4-dimensional tensor with first dimension signifying the items in the mini-batch and the rest of the dimensions the width, height and color channels of the image. We also normalized the images in the dataset so that each pixel color value, instead of being in the range 0–255, was in the range 0–1 first and then −0.5–0.5. This magnitude normalization and zero-centering was done for better results in gradient descent.

Being restricted to using specifically the multiclass SVM with SGD we still tried to manipulate the available hyperparameters as best we could to improve the accuracy of classification. Namely, we tested using different number of epochs ranging from 1 to 10 in which our best and consistent results were when this hyperparameter is set to the value 6. Another hyperparameter that improved results was the learning rate. We tested different learning rates that were as high as 0.1 to a learning rate as small as 10^{-7}. During these trials we found our solution yielded the best results when the value was set at 10^{-5}. At one point we also tried using PyTorch's built-in learning rate scheulder (`torch.optim.lr_scheduler.ReduceLROnPlateau`) for dynamic learning rate adjustment but the results from using that didnt show any improvements. Using those values the results consistently yielded an accuracy between 38–39% on the test dataset. Since we are using SVMs, which in their simplest formulation are linear classifiers, on complex data like these images we feel that being nearly four times more accurate than the theoretical accuracy of 10% with a random guess seemed to be a characteristic of a pretty successful implementation.

3 Statistics

Presented below are some accuracy percentages with various values of learning rate and epoch count.

Learning Rate	Epochs	Accuracy
10^{-2}	6	29%
10^{-5}	4	38%
10^{-4}	6	36%
10^{-7}	2	24%
10^{-7}	6	39%