Homework 1 report - network implementation

Luka Boljević (63210482)

1 Introduction

The primary goal of this homework was to implement a neural network from scratch, using mini-batch gradient descent or Adam as the optimizer, and optionally using L2 regularization and/or exponential learning rate decay. The neural network will then be used for classification on the CIFAR-10 dataset [1].

The CIFAR-10 dataset consists of 60000 32 by 32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The classes are mutually exclusive, and they are: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*.

2 Network

We will be dealing with a multi-class classification problem, so the loss function that is used is categorical cross entropy. For that reason, the activation function on the last layer is softmax, while in the hidden layers, we used sigmoid. As stated, the two available optimizers will be mini-batch gradient descent and Adam. Optionally, L2 regularization and/or exponential learning rate decay can be used with both of the optimizers.

3 Training and experiments

3.1 Initial network

To achieve 48% classification accuracy on the test set, we tried a lot of network architectures. However, the one that is not too complex was the network with the following:

- Optimizer: Mini-batch gradient descent
- Hidden layers structure (each value represents one layer): 500, 500
- Number of epochs: 40
- Mini-batch size: 64
- Learning rate η: 0.3
- Learning rate schedule: no
- Regularization parameter λ: 0.1

This network was able to achieve **51.2%** classification accuracy. An architecture that is exactly the same, but uses mini-batch size 128, trains a bit faster but achieves a slightly worse result ($\sim$ 50% CA). Various other architectures were tried, but we weren't able to find the one that would consistently increase the aforementioned 51.2% CA.

3.2 Adam vs. Gradient Descent

For the next experiment, we kept all parameters the same (hidden layers, number of epochs, mini-batch size, learning rate schedule and λ) as in the initial network (Subsection 3.1), but set $\eta = 0.001$ and used the Adam optimizer. This network was able to achieve classification accuracy **46.9%**, compared to the 51.2% its gradient descent counterpart achieved.

3.3 No vs. L2 regularization

Starting from the initial network again (Subsection 3.1), we wanted to see the effect L2 regularization had. The initial network used $\lambda = 0.1$, so we'll test $\lambda = 0.0$ (so, no regularization), and $\lambda = 0.01$. The results can be seen in Table 1.

λ	0.01	0.0
CA	50.56%	51.25%

Table 1: Experimenting with different values for the regularization parameter λ. The initial network settings, shown in Subsection 3.1, used $\lambda = 0.1$ and achieved 51.2% CA. The newly obtained results aren't much different from the initial network.

3.4 No vs. exponential learning rate decay

In the last experiment, we tested using exponential learning rate decay, with different decay rates k, considering the initial network did not use a learning rate schedule. We tested $k = 0.001$, $k = 0.01$ and $k = 0.1$. Using $k = 0.0$ is equivalent to using no learning rate schedule. Results can be found in Table 2.

4 Observations

In this section, we will write a few observations while experimenting and trying out different architectures and hyperparameters. They will be split into paragraphs in no particular order.

A network with a single hidden layer generally didn't suffice, i.e. it couldn't learn very well. Using 2 or 3 hidden layers was more than enough, each of which had 200 to 500

k	0.001	0.01	0.1
CA	39.34%	28.09%	19.64%

Table 2: Experimenting with different values for the decay rate k when using exponential learning rate decay. The initial network settings, shown in Subsection 3.1, used $k = 0.0$ (so, no learning rate schedule) and achieved 51.2% CA. We can see that, the faster our learning rate decays, the worse the network performance is, i.e. the sooner it stops learning.

neurons. Using more than 500 neurons per hidden layer was unnecessary, as it just made the network train slower without much merit.

For gradient descent, $\eta = 0.2$ and $\eta = 0.3$ generally worked well. We mostly used those two values of our later experiments. For smaller η, say 0.01, the network trained slower (expectedly).

For Adam, we always used the typical values $\beta_1 = 0.9$ and $\beta_2 = 0.999$, as we found it easier to vary the other hyperparameters. Furthermore, using $\eta > 0.001$ (for example, 0.01, 0.1) would make the network stop learning after a few epochs, without really learning much at all. We mostly $\eta = 0.001$ for all our tests with Adam.

As far as we were able to test, a network using Adam learned and improved slower than a network using gradient descent, and the Adam network would require more epochs (say, 70, 80, compared to 40 with GD) to reach the classification accuracy of GD. However, it was more stable and overfitted noticeably less often than GD.

Exponential learning rate decay was generally not a huge improvement. The only reasonable results were obtained for decay rate $k = 0.001$, for only a few cherry picked architectures. Any higher value for k would decrease the learning rate way too fast, forcing the network to stop learning too quickly, without having learned much at all. A different learning rate schedule would have probably been better, for example step decay.

References

[1] Alex Krizhevsky and Vinod Nair and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). Link, 2009.