(CS535) Deep Learning Homework 2

Ayoung Kang (ID:933-610-350)

In this assignment, we trained a one hidden layer fully connected neural network on the attached 2-class (class 0: airplane, class 1: ship) dataset extracted from CIFAR-10. Figure 1 shows some of the training examples.

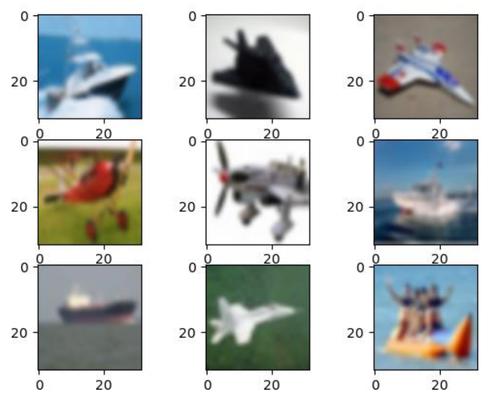

Figure 1. Training examples

1) Write a function that evaluates the trained network (5 points), as well as computes all the subgradients of W1 and W2 using backpropagation (5 points).

Please, see the source code. A function that evaluates the trained network is defined as `evaluate()`, and `LinearTransform`, `ReLU`, `SigmoidCrossEntropy` classes have their own `backward` function, which computes all the subgradients.

2) Write a function that performs stochastic mini-batch gradient descent training (5 points). You may use the deterministic approach of permuting the sequence of the data. Use the momentum approach described in the course slides.

Please, see the source code. A function `train()` performs stochastic mini-batch gradient descent training.

3) Train the network on the attached 2-class dataset extracted from CIFAR-10: (data can be found in the cifar-2class-py2.zip downloadfile on Canvas.). The data has 10,000 training examples in 3072 dimensions and 2,000 testing examples. For this assignment, just treat each dimension as uncorrelated to each other. Train on all the training examples, tune your parameters (number of

hidden units, learning rate, mini-batch size, momentum) until you reach a good performance on the testing set. What accuracy can you achieve? (20 points based on the report).

Based on the results from question 5, I tuned the parameters as belows and achieved 84.99% test accuracy in epoch 71 (Table1). Figure 2 shows the train and test accuracies when using following parameters

- num_epochs = 100
  batch_size = 256
  hidden_units = 256
  learning_rate = 0.0001
  momentum = 0.8

```
[Epoch 71, mb 39]    Avg.Loss = 0.076
   Train Loss: 757.410    Avg. train Loss: 0.076    Train Acc.: 98.69%
   Test Loss: 765.138    Avg. test Loss:  0.383    Test Acc.:  84.99%

Best train accuracy: 99.79%    Best test accuracy: 84.99%
```

Table 1. Best test accuracy

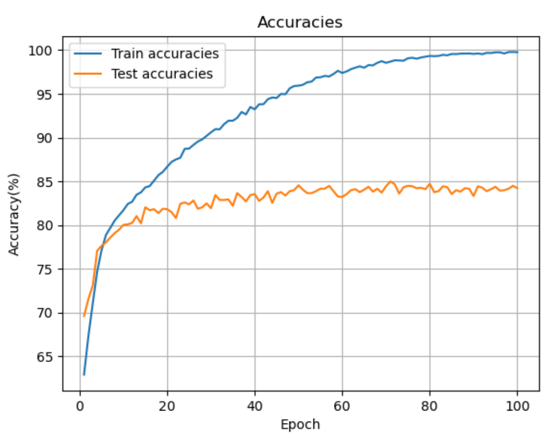

Figure 2. Train and test accuracies

(4) Training Monitoring: For each epoch in training, your function should evaluate the training objective, testing objective, training misclassification error rate (error is 1 for each example if misclassifies, 0 if correct), testing misclassification error rate (5 points).

Loss = sum of the loss over all examples

Avg. loss = loss / number of examples

Misclassification error rate (%) = 100 - Accuracy

```
[Epoch 1, mb 39]    Avg.Loss = 0.662
   Train Loss: 6604.716    Avg. train Loss: 0.660    Train Acc.: 62.89%
   Test Loss: 1078.465    Avg. test Loss:  0.539    Test Acc.:  69.59%
```

```
[Epoch 2, mb 39]    Avg.Loss = 0.594
   Train Loss: 5928.305    Avg. train Loss: 0.593    Train Acc.: 67.31%
   Test Loss: 1002.969     Avg. test Loss:  0.501    Test Acc.:  71.54%
[Epoch 3, mb 39]    Avg.Loss = 0.559
   Train Loss: 5577.929    Avg. train Loss: 0.558    Train Acc.: 71.07%
   Test Loss: 951.551      Avg. test Loss:  0.476    Test Acc.:  73.10%
[Epoch 4, mb 39]    Avg.Loss = 0.522
   Train Loss: 5213.781    Avg. train Loss: 0.521    Train Acc.: 74.68%
   Test Loss: 891.968      Avg. test Loss:  0.446    Test Acc.:  77.06%
[Epoch 5, mb 39]    Avg.Loss = 0.488
   Train Loss: 4867.399    Avg. train Loss: 0.487    Train Acc.: 77.11%
   Test Loss: 844.662      Avg. test Loss:  0.422    Test Acc.:  77.62%
[Epoch 6, mb 39]    Avg.Loss = 0.462
   Train Loss: 4608.564    Avg. train Loss: 0.461    Train Acc.: 78.86%
   Test Loss: 825.666      Avg. test Loss:  0.413    Test Acc.:  78.01%
[Epoch 7, mb 39]    Avg.Loss = 0.444
   Train Loss: 4435.966    Avg. train Loss: 0.444    Train Acc.: 79.68%
   Test Loss: 800.278      Avg. test Loss:  0.400    Test Acc.:  78.57%
[Epoch 8, mb 39]    Avg.Loss = 0.430
   Train Loss: 4294.923    Avg. train Loss: 0.429    Train Acc.: 80.51%
   Test Loss: 788.556      Avg. test Loss:  0.394    Test Acc.:  79.07%
[Epoch 9, mb 39]    Avg.Loss = 0.420
   Train Loss: 4191.114    Avg. train Loss: 0.419    Train Acc.: 81.11%
   Test Loss: 769.843      Avg. test Loss:  0.385    Test Acc.:  79.46%
[Epoch 10, mb 39]    Avg.Loss = 0.408
   Train Loss: 4076.326    Avg. train Loss: 0.408    Train Acc.: 81.70%
   Test Loss: 743.935      Avg. test Loss:  0.372    Test Acc.:  80.02%
```

(5) Tuning Parameters: please create three figures with following requirements. Save them into jpg format:

- **Test accuracy with different number of batch size**
  (Learning rate = 0.01, hidden units = 128, momentum = 0.8)

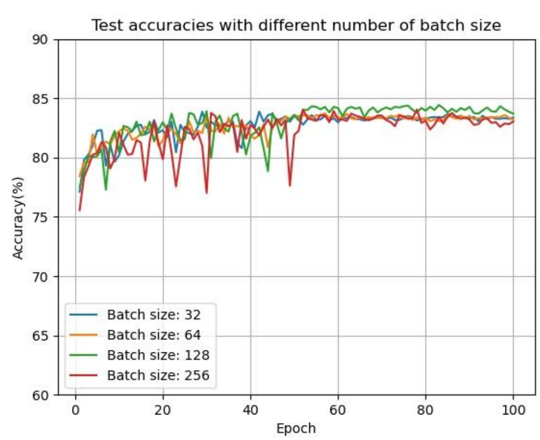

Figure 3. Test accuracies with batch size = [32, 64, 128, 256]

- **Test accuracy with different learning rate**
  (Batch size = 64, hidden unit = 128, momentum = 0.8)

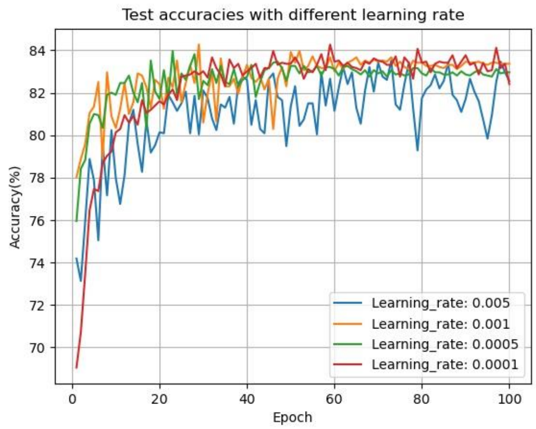

Figure 4. Test accuracies with learning rate = [0.0001, 0.0005, 0.001, 0.005]

- **Test accuracy with different number of hidden units**
  (Batch size = 64, learning rate = 0.001, momentum = 0.8)

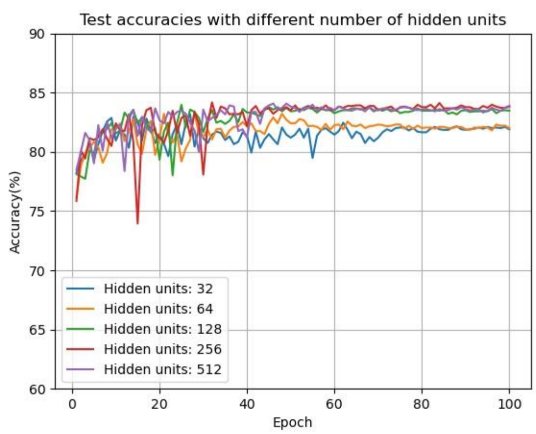

Figure 5. Test accuracy with hidden units = [32, 64, 128, 256, 512]

(6) Discussion about the performance of your neural network.

Since the network performs stochastic mini-batch gradient descent, the train and test accuracy curves do not have a linear trend. For this data set, it seems that a batch size does not have a huge impact on the test accuracy. When a batch size is one, it performs stochastic gradient descent, and it will take longer to train one epoch because we will lose speed up from vectorization. On the other hand, when using a larger batch size, one gradient step will take a longer time.

Figure 6 shows the train accuracies with different learning rates when we do not use the momentum approach. Without the momentum approach, the smaller the learning rate, the slower our neural network reaches the highest train accuracy. However, we can see that with the momentum approach, the network achieves the highest train accuracy much faster even when we use a small learning rate because momentum accelerates the gradient descent. We also observe that the bigger the learning rate, the more the test accuracy curve fluctuates.

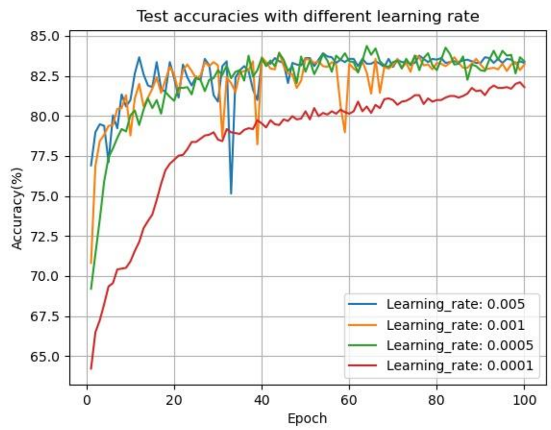

Figure 6. Test accuracies without momentum

As we increase the number of hidden units, we observe that the network achieves higher test accuracy. If we have too few hidden units, we will get higher train error due to underfitting. If we have too many hidden units, we may get low train error but still get high test error due to overfitting.