DD2424 Deep Learning Assignment1

Ershad Taherifard (ershadt@kth.se)

April 2020

1 Introduction

This paper trains and tests a one layer network with multiple outputs to classify images from the CIFAR-10 dataset. The dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. They are divided into five training batches and one test batch, each with 10000 images.

2 Methods

The network was implemented from scratch in Python. The network was trained using mini-batch gradients descent on a cost function that computes the cross-entropy loss of the classifier applied to the labelled training data and an L_2 regularization term on the weight matrix.

To make sure the gradient calculations were correct, the analytically calculated gradients were compared with a numerical calculated gradients.

The network's performance was examined by altering its parameters, that is the regularization and the learning rate.

Finally, once the weight matrices ware trained using the different parameter settings, they were all visualized.

3 Results

3.1 Gradients

The analytically calculated gradients were in fact correct. This is seen by calculating the relative error between the a numerically computed gradient value and an analytically computed gradient. The numerically computed gradient is based on the centered difference formula.

$$\frac{|g_a - g_n|}{\max\left(\text{eps}, |g_a| + |g_n|\right)}$$

$Batchsize =$	10	100
$\sum gn_W =$	$-1.4210854715202004\text{e}-14$	$-6.217248937900877\text{e}-15$
$\sum gn_b =$	0.0	$-1.9081958235744878\text{e}-17$
$\sum ga_W =$	$1.4677155490971927\text{e}-08$	$8.126832540256146\text{e}-09$
$\sum ga_b =$	$2.2204455288332703\text{e}-11$	$4.4408915780835834\text{e}-11$
$\varepsilon_b =$	$1.614688079270477\text{e}-10$	$1.1438445433401461\text{e}-10$
$\varepsilon_W =$	$8.179941424082782\text{e}-11$	$1.9930892644255269\text{e}-10$

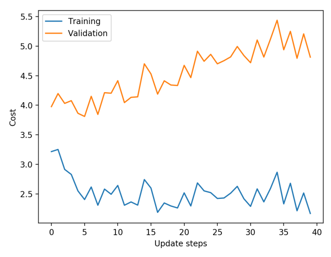
3.2 Parameter settings

The following parameter settings were applied:

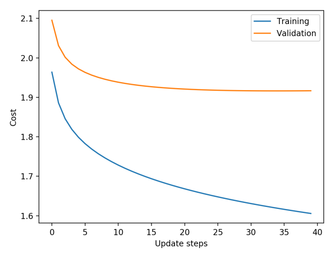
- $\lambda = 0$, n epochs = 40, $n_{batch} = 100$, eta = 0.1
- $\lambda = 0$, n epochs = 40, $n_{batch} = 100$, eta = 0.001
- $\lambda = 0.1$, n epochs = 40, $n_{batch} = 100$, eta = 0.001
- $\lambda = 1$, n epochs = 40, $n_{batch} = 100$, eta = 0.001

3.3 Loss and costs

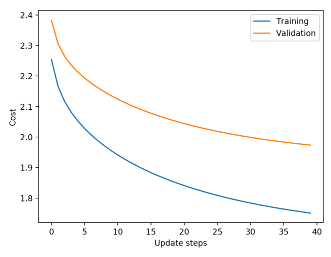
(a) Learning rate = 0.1 $\lambda = 0$



(b) Learning rate = 0.001 $\lambda = 0$



(c) Learning rate = 0.001 $\lambda = 0.1$



(d) Learning rate = 0.001 $\lambda = 1$

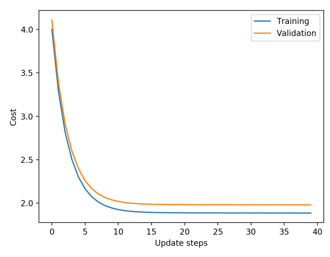
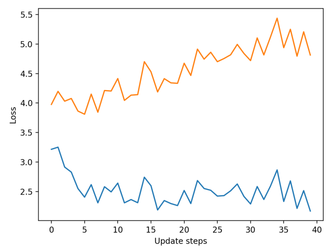
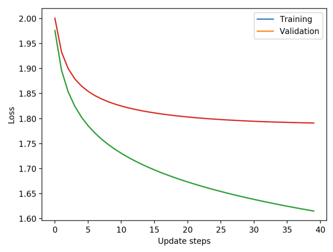


Figure 1: The cost functions

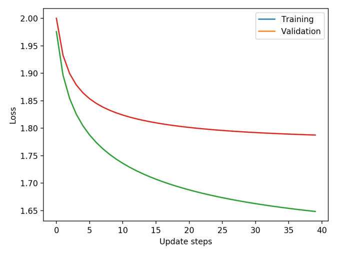
(a) Learning rate = 0.1 $\lambda = 0$



(b) Learning rate = 0.001 $\lambda = 0$



(c) Learning rate = 0.001 $\lambda = 0.1$



(d) Learning rate = 0.001 $\lambda = 1$

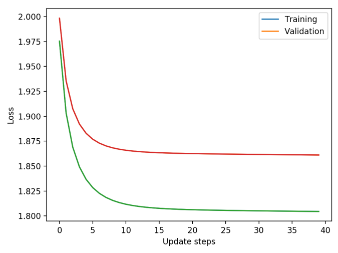


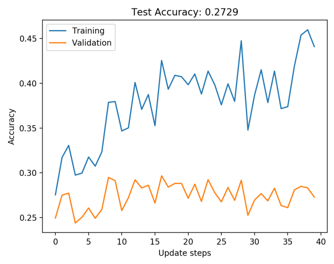
Figure 2: The loss functions

3.4 Accuracies

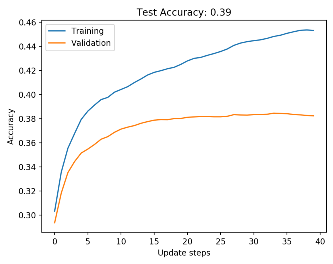
The following table show the final accuracy the network achieves.

Accuracies	**Training data**
$\lambda = 0$ $eta = 0.1 =$	0.2729
$\lambda = 0$ $eta = 0.001$	0.39
$\lambda = 0.1$ $eta = 0.001$	0.3904
$\lambda = 1$ $eta = 0.001$	0.3758

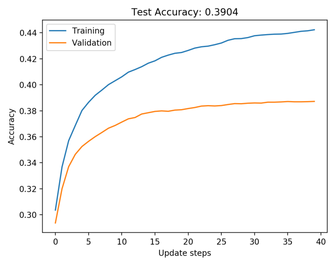
(a) Learning rate = 0.1 $\lambda = 0$



(b) Learning rate = 0.001 $\lambda = 0$



(c) Learning rate = 0.001 $\lambda = 0.1$



(d) Learning rate = 0.001 $\lambda = 1$

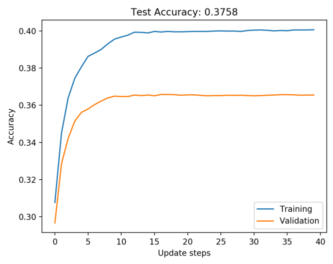
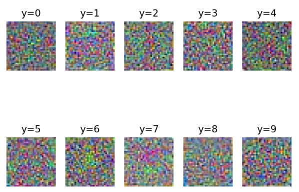


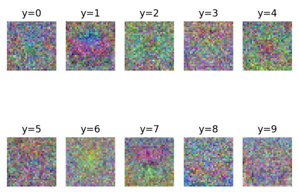
Figure 3: The accuracies.

3.5 Weight matrix

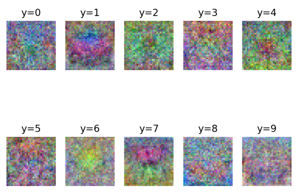
(a) Learning rate = 0.1 $\lambda = 0$



(b) Learning rate = 0.001 $\lambda = 0$



(c) Learning rate = 0.001 $\lambda = 0.1$



(d) Learning rate = 0.001 $\lambda = 1$

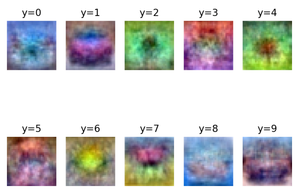


Figure 4: We can see that the weight matrix starts to visualise the distinct classes.

4 Conclusion

When training any supervised learning algorithms one must take the bias-variance trade-off into consideration. To overcome the overfitting issue, that is avoiding high variance, one can increase the regularization term. When (λ is low, the training data overfits and results in high accuracy but the performs poorly on the validation data. As λ increases, the accuracy the training data decreases but the model is able to perform better on the validation data since it is now more general and not influenced by outliers. But if λ is too high, the model will underfit and give us bad results once again. The second parameter of the model was the learning rate which is the length of each update step. Higher learning rate resulted in quicker convergence towards the local minima, however if too high it will start to fluctuate and overshoot the minima. But if the learning rate is to small the learning will be more stable but slower.