Assignment 1: One layer network

DD2424 - Deep Learning
Svenja Räther

March 2020

1 Introduction

This assignment 1 in the course DD2424 Deep Learning in Data Science trains and tests a one layer network with multiple outputs to classify images from the CIFAR-10 dataset. The network is trained using mini-batch gradient descent applied to a cost function that computes the cross-entropy loss of the classifier applied to the labelled training data and an L2 regularization term on the weight matrix.

2 Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. The dataset is divided into five training batches and one test batch, each with 10000 images. It is available here. A preview of the pictures is given in Figure 1.

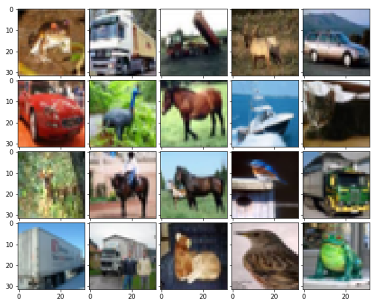


Figure 1: Preview of the first 10 pictures from the CIFAR-10 dataset in batch 1

3 Computing gradients

To calculate the gradients analytically, the function *ComputeGradients* is used. It follows the computations given for efficient gradient computations for a mini-batch presented in lecture 3 of the course.

3.1 Forward pass

The forward pass takes the input parameters X, W and b into the function *EvaluateClassifier* and returns the matrix P, which P contains the probability for each label for the image in the corresponding column of X. This is achieved by applying the *softmax* activation function to the calculated scores.

3.2 Backward pass

In order to be able to learn the parameters W and b, we need to calculate the gradients for those within each backpropagation step. Therefore, mini-batch gradient descent is used. The implemented efficient algorithm does this by calculation the G-batch which is used in the gradient calculations. A regularization term is added to the W-gradient.

3.3 Testing against numerical gradients

For verification of the correctness of the analytical computation of the gradients, those were compared to the numerically computed gradients. The code for calculating the numerical gradients was given by the course instructor and can be used by calling the functions *ComputeGradsNum* and *ComputeGradsNumSlow*. To compare those with the analytical gradients even for very small values, the function *ComputeRelativeError* is used to compute the relative errors.
Once this is computed, I checked whether the max difference in any value for the gradients of W and b is smaller than 1e-4. Once this requirement was fulfilled, I was confident that the implementation of the numerical gradients works as expected and I moved on with the assignment.

4 Results

To test the one layer network four different setups were executed. The tested parameters for each setup and the results are displayed in the corresponding subsections.

- Setup 1: lambda=0, n epochs=40, n batch=100, eta=.1
- Setup 2: lambda=0, n epochs=40, n batch=100, eta=.001
- Setup 3: lambda=.1, n epochs=40, n batch=100, eta=.001
- Setup 4: lambda=1, n epochs=40, n batch=100, eta=.001

Training corresponds to the first batch in the CIFAR-10 dataset, Validation to the second and Testing to the third. The following key takeouts can be stated by the comparison of the results.

Importance of the correct learning rate:

Backgroud: Choosing the correct learning rate has an effect on how good the network will be able to learn the weights and how well it performs. It controls the step rate that is used to go along the gradients. If the learning rate is too high, it happens that the desired minimum is overshoot, if it is zero, no steps will be taken.

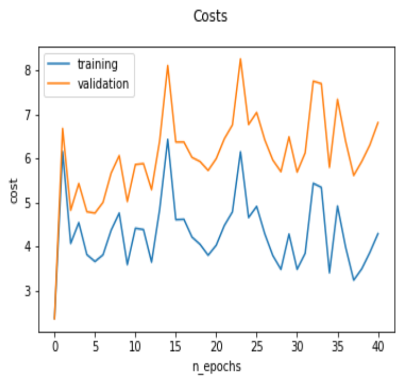
Observation: By comparing the results of setup 1 *(eta = .1)* with setup 2 *(eta = .001)* it is notable that the testing accuracy is much better on the second setup with *38.8%* compared to setup 1 with *29.5%*. Looking at the graphs for cost and accuracy, setup 1 shows changing variations in its directions. It seems to jump up and down in cost and leads to a not so good result on Training (*4.28*) and even worse on Validation (*6.81*). The accuracy shows a good improvement among the first few epochs but then also starts to follow a zigzack like curve. In setup 2, the curves for cost and accuracy follow one defined trend and allow for a smooth convergence. The learning slowly evolves into the desired direction. The overall results seem to be much better for this choice of eta.
To avoid oscillating curves (unstable learning) and results as the ones pictured in setup 1, it is important to chose a correct learning rate.

Effect of increasing the amount of regularization:

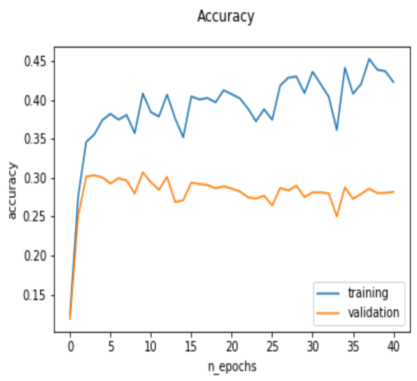
Backgroud: Regularization is a measure to control the overfitting of a model. Increasing the lambda value penalizes the weights and leads to a more generalized version of the model. This should also lead to a decreased gap between the testing and validation curve and result in better predictions on unseen data. However, having a very large lambda can lead to underfitting. Therefore, the right choice is important.

Observation: Increasing the regularization parameter lambda from *0 to .1* as described in the change from setup 2 to setup 3 decreases the gap between values for Training and Validation in accuracy and cost. Setup 3 results in an accuracy on Test of *39.27* even though we encounter a smaller accuracy on the Train for setup 3(*44.85*) compared to setup 2 (*45.53*). As described above, overfitting is reduced by the usage of the regularization term.
By setting lambda to 1 as seen it setup 4, the values move even closer together which is visible in the graphs. However, in this setup the test accuracy decreases to *37.55* and the curves show a very fast convergence towards one value. A too high value on lambda is thus also a bad choice as it might result in underfitting.

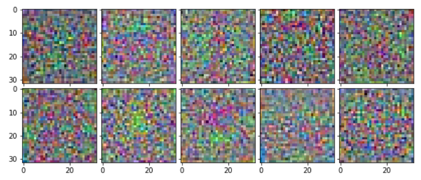
4.1 Setup 1



(a) Cost for setup 1



(b) Accuracy for setup 1



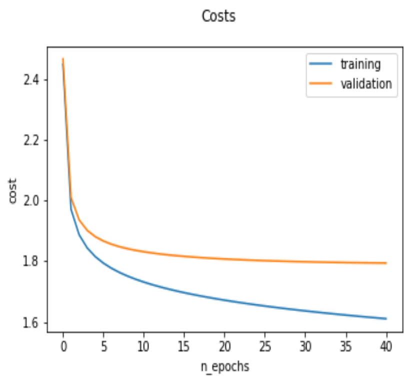
(c) Weigths for setup 1

Figure 2: Plots for setup 1

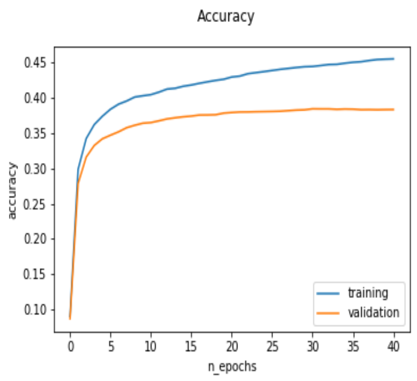
Measure	Training	Validation	Testing
Cost	4.2872	6.8123	-
Accuracy	0.4229	0.2815	0.295

Table 1: Results for setup 1

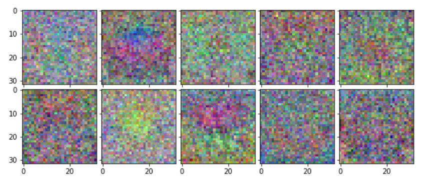
4.2 Setup 2



(a) Cost for setup 2



(b) Accuracy for setup 1



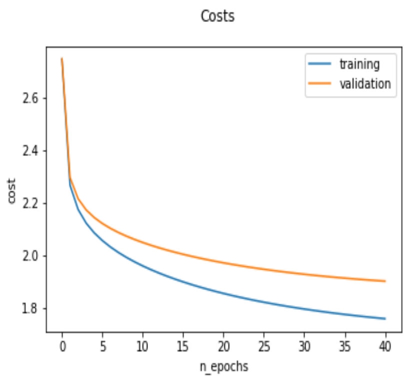
(c) Weigths for setup 2

Figure 3: Plots for setup 2

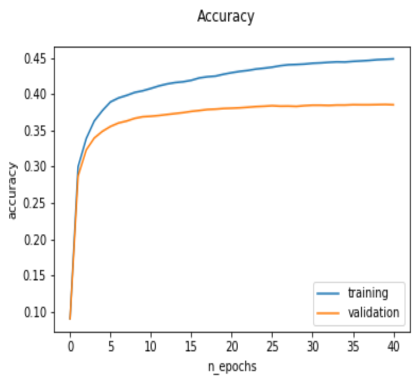
Measure	**Training**	**Validation**	**Testing**
Cost	1.6107	1.7944	-
Accuracy	0.4553	0.3835	0.388

Table 2: Results for setup 2

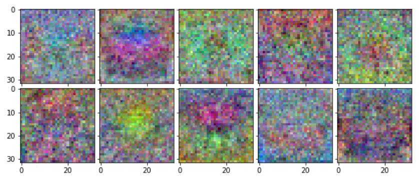
4.3 Setup 3



(a) Cost for setup 3



(b) Accuracy for setup 3



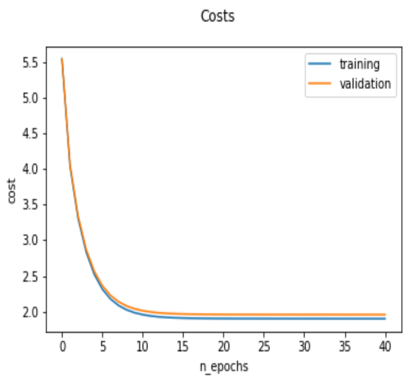
(c) Weigths for setup 3

Figure 4: Plots for setup 3

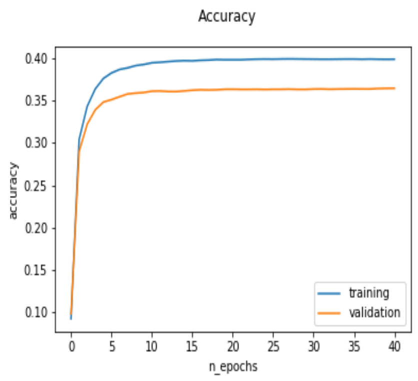
Measure	Training	Validation	Testing
Cost	1.7605	1.9028	-
Accuracy	0.4485	0.3853	0.3927

Table 3: Results for setup 3

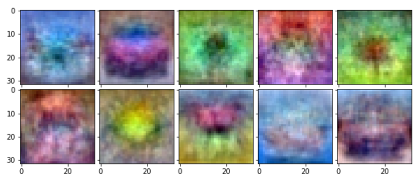
4.4 Setup 4



(a) Cost for setup 4



(b) Accuracy for setup 4



(c) Weigths for setup 4

Figure 5: Plots for setup 4

Measure	Training	Validation	Testing
Cost	1.8995	1.9568	-
Accuracy	0.3988	0.3645	0.3755

Table 4: Results for setup 4