Machine Learning Handin 2

Erik Holm Steenberg (201804655), Hans Brüner Dein (201706079),
Camilla Theresia Grøn Sørensen (201807007)
Instruktor: Mads Bech Toftrop

November 2022

Introduction

This is the report for the 2nd handin in the computer science course Machine Learning. The handin has two parts; One part about the derivative of the loss function and one part about the code for our neural network.

Part I: Derivative

First we need to calculate the derivative of the loss function of one data point, with respect to a single input variable z_i, given a one-hot-label label vector y with $y_j = 1$. Because of the one-hot-label label vector the loss function reduces to:

$$\text{softmax}(z)_j = \frac{e^{Z_j}}{\sum_{a=1}^{k} e^{Z_a}} \tag{1}$$

$$L(z) = -\sum_{i=1}^{k} y_i \ln(\text{softmax}(z)_i) = -\ln(\text{softmax}(z)_j) \tag{2}$$

$$\text{L}(z) = -\ln\left(\frac{e^{Z_j}}{\sum_{a=1}^{k} e^{Z_a}}\right) = -(\ln(e^{Z_j}) - \ln\left(\sum_{a=1}^{k} e^{Z_a}\right)) = -Z_j + \ln\left(\sum_{a=1}^{k} e^{Z_a}\right) \tag{3}$$

$$\frac{\partial L}{\partial z_i} = -\frac{\partial Z_j}{\partial z_i} + \frac{\partial}{\partial z_i}\ln\left(\sum_{a=1}^{k} e^{Z_a}\right) = -\delta_{i,j} + \frac{1}{\sum_{a=1}^{k} e^{Z_a}}\left(\frac{\partial}{\partial z_i}\sum_{a=1}^{k} e^{Z_a}\right) \tag{4}$$

$$= -\delta_{i,j} + \frac{e^{Z_i}}{\sum_{a=1}^{k} e^{Z_a}} = -\delta_{i,j} + \text{softmax(z)}_i \qquad \blacksquare \tag{5}$$

PART II: Implementation and test

Our implementation of the **forward pass** and **backward pass** are as follows:

```
### FORWARD PASS
XW1 = X @ W1 + b1
hidden = relu(XW1)
out = hidden @ W2 + b2
soft = softmax(out)
loss = -np.sum(labels*np.log(soft),axis=1)+c*(np.sum(W1**2)+np.sum(W2**2))
cost = np.mean(loss)
### BACKWARD PASS
d_out = (-labels + soft)/X.shape[0]
d_hidden = d_out@W2.T
diff = (hidden > 0)
d_XW1 = d_hidden * diff
d_w1 = X.T@d_XW1 + 2 * c * W1
d_w2 = hidden.T @ d_out + 2 * c * W2
d_b1 = np.sum(d_out @ W2.T * diff,axis=0,keepdims=True)
d_b2 = np.sum(d_out,axis=0,keepdims=True)
```

Running net_test.py grants us the following plots.

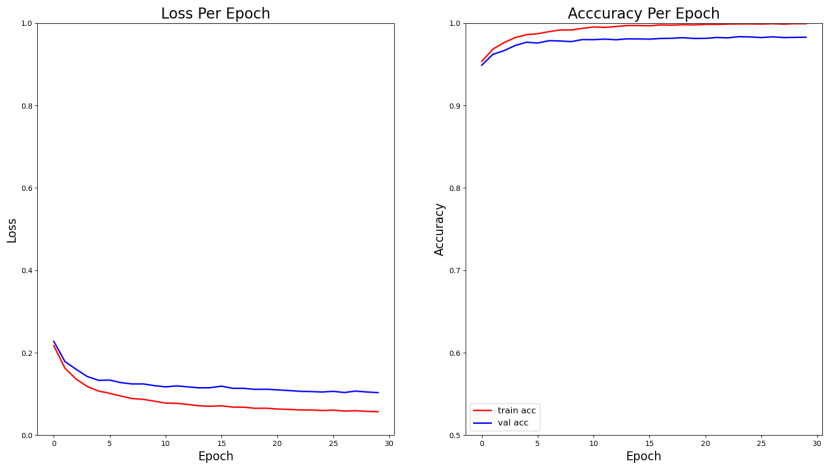

Figure 1: The two plots show respectively the loss per epoch and the accuracy per epoch for the training and validation set. The red line represents the training set, and the blue line represents the validation set. The loss is small and the accuracy is large in the beginning. This makes sense because by the time that we get the first point we have already trained using the training dataset multiple times.

The code also returns the final accuracies of loss and validation:

- In sample accuracy 0.9968833333333333
- Test sample accuracy 0.9821982198219822

The finished our report.