Machine Learning Handin2

Group member
Yidan Chen(202003411,202003411@post.au.dk)
Cheng Huang(202101639,202101639@post.au.dk)

November 9, 2021

1 Part I: Derivative

We already know that

$$
\begin{aligned}
&L(z) = -\sum_{i=1}^{k} y_i ln(softmax(z)_j) = -ln(softmax(z)_j)\\
&\delta i, j = 1 \quad if \quad i = j\\
&\delta i, j = 0 \quad otherwise
\end{aligned}
$$

We can do the following operations

$$
\begin{aligned}
\frac{\partial L}{\partial z_i} &= \frac{\partial - ln(softmax(z)_j)}{\partial z_j}\\
&= \frac{\partial}{\partial z_j} - ln(\frac{e^{zj}}{\sum_{\alpha=1}^{k} e^{z\alpha}})\\
&= \frac{\partial}{\partial zj}(-lne^{zj} + ln\sum_{\alpha=1}^{k} e^{z\alpha})\\
&= -\frac{\partial lne^{zj}}{\partial zi} + \frac{\partial ln\sum_{\alpha=1}^{k} e^{z\alpha}}{\partial zi}
\end{aligned}
$$

We can divide it into two parts

Firstly

$$
-\frac{\partial lne^{zj}}{\partial zi} = \begin{cases} -1 & i = j \\ 0 & i! = j \end{cases}
$$

Secondly

$$
\begin{aligned}
\frac{\partial ln\sum_{\alpha=1}^{k} e^{z\alpha}}{\partial zi} &= \frac{1}{\sum_{\alpha=1}^{k} e^{z\alpha}} \frac{\partial \sum_{\alpha=1}^{k} e^{z\alpha}}{\partial z_i}\\
&= \frac{e^{zi}}{\sum_{\alpha=1}^{k} e^{z\alpha}}
\end{aligned}
$$

Through above we can get the conclusion that the

$$
\begin{aligned}
The\,first\,part = -\delta_{i,j}\\
The\,second\,part = softmax(z)_i
\end{aligned}
$$

Therefore

$$
\frac{\partial L}{\partial z_i} = -\delta_{i,j} + \frac{1}{\sum_{\alpha=1}^{k} e^{z\alpha}} e^{zi} = -\delta i, j + softmax(z)_i
$$

2 Part II: Implementation and test

2.1 Code for forward pass and backwards pass

```
### YOUR CODE HERE - FORWARD PASS
n = X.shape[0]
xw1 = np.dot(X, W1) + b1
xw1_out = relu(xw1)
softmax_in = np.dot(xw1_out, W2) + b2
nn = softmax(softmax_in)
cost=np.mean(-1*(np.log(nn)[labels.nonzero()])+c*(np.sum(W1*W1)+np.sum(W2*W2)))
### END CODE
```

```
### YOUR CODE HERE - BACKWARDS PASS
d_nn = nn - labels
d_b2 = np.mean(d_nn, axis=0).reshape(b2.shape)
d_w2 = 1/n * np.dot(xw1_out.T, d_nn) + 2*c*W2
d_cc = np.dot(d_nn, W2.T)
new_d_cc = d_cc.copy()
new_d_cc[xw1<0] = 0
d_b1 = np.mean(new_d_cc, axis=0).reshape(b1.shape)
d_w1 = 1/n * np.dot(X.T, new_d_cc) + 2*W1*c
### END CODE
```

The following picture shows the Loss Per Epoch and Accuracy Per Epoch

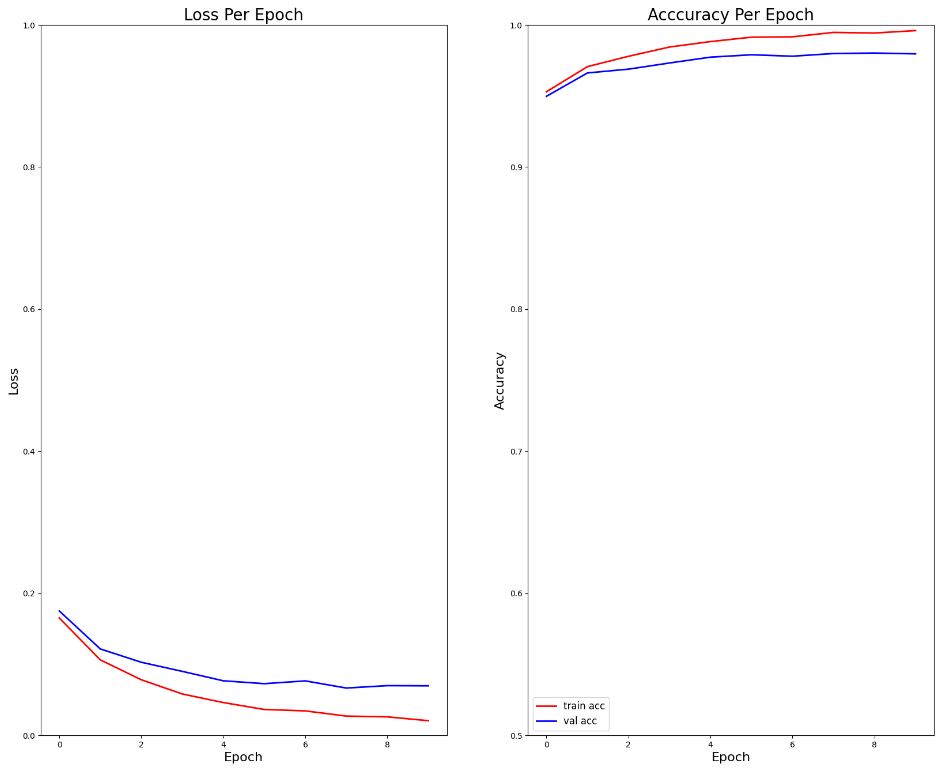

Figure 1: epoch plots

We end up with **in sample accuracy 0.9922** and **test sample accuracy 0.9773977397739774**

```
args Namespace(batch_size=-1, epochs=-1, hidden=-1, lr=-1)
in sample accuracy 0.9922
test sample accuracy 0.9773977397739774
outputting to file epoch_plots.png
```

Figure 2: in sample and test accuracy