Team Rap: Rasmus Jørgensen (201909451), Mathias Weller (201907948), & Anna Blume (201907691)

# Machine Learning assignment 2

**Part I: Derivative**

This is our derivation of the derivative:

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

$$\frac{dL}{dz_i} = \left(-\ln\left(softmax(z)_j\right)\right)'$$

$$= -\frac{1}{softmax(z)_j} \cdot \left(softmax(z)_j\right)' \text{ , chain rule}$$

$$= -\frac{1}{softmax(z)_j} \cdot \left(\frac{e_j^z}{\sum e^{z_k}}\right)'$$

$$= -\frac{1}{softmax(z)_j} \cdot \frac{\delta_{i,j} \cdot e^{z_j} \cdot \sum e^{z_k} - e^{z_j} \cdot e^{z_i}}{(\sum e^{z_k})^2} \text{ , division rule}$$

$$= -\frac{\sum e^{z_k}}{e^{z_j}} \cdot \frac{\delta_{i,j} \cdot e^{z_j} \cdot \sum e^{z_k} - e^{z_j} \cdot e^{z_i}}{(\sum e^{z_k})^2}$$

$$= -\frac{\delta_{i,j} \cdot \sum e^{z_k} - e^{z_i}}{\sum e^{z_k}}$$

$$= -\frac{\delta_{i,j} \cdot \sum e^{z_k}}{\sum e^{z_k}} + \frac{e^{z_i}}{\sum e^{z_k}}$$

$$= -\delta_{i,j} + \frac{e^{z_i}}{\sum e^{z_k}}$$

$$= -\delta_{i,j} + softmax(z)_i$$

where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise.

Team Rap: Rasmus Jørgensen (201909451), Mathias Weller (201907948), & Anna Blume (201907691)

**Part II: Implementation and test**

We have not implemented that we can calculate the gradient for the entire input at once, so we do both forward and backward pass for each row in X.

Forward pass:

```
#Forward pass
currentRow = X[i,:]
a = currentRow @ W1
b = a + b1
rel = relu(b)[0]
d = rel @ W2
e = d + b2
softmaxVec = softmax(e)[0]
probability = softmaxVec[y[i]]
f = -np.log(probability)
#Cost
nll[i] = f
```

Backward pass:

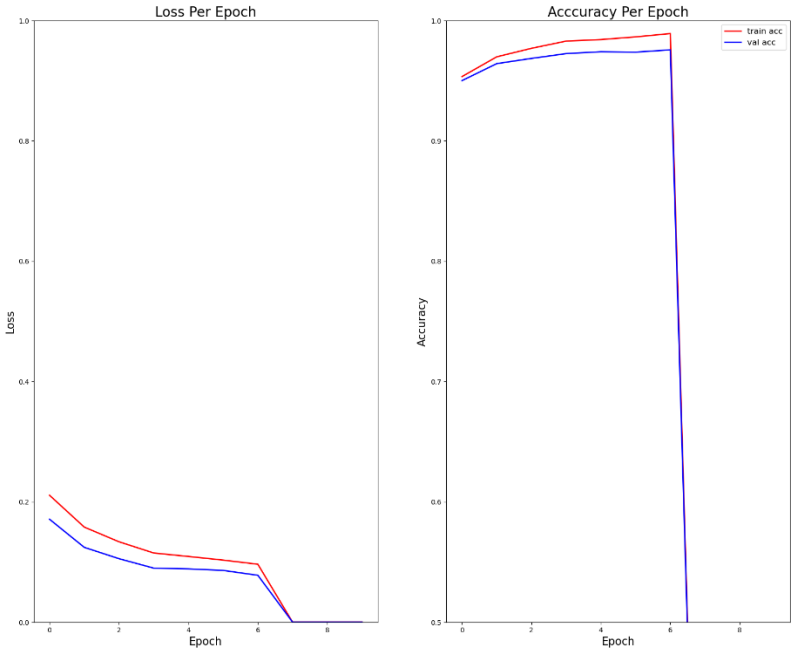
```
#Backward pass
df_de = (softmaxVec - labels[i])
df_db2 = df_de
df_dd = df_de
df_dc = df_dd @ W2.T
df_dw2 = rel[:, np.newaxis] @ df_dd[:,np.newaxis].T
dc_db = np.diag(rel > 0).astype(int)
df_db = df_dc @ dc_db
df_da = df_db
df_db1 = df_db
df_dw1 = currentRow[:, np.newaxis] @ df_da[:,np.newaxis].T

W1_grad += df_dw1
W2_grad += df_dw2
b1_grad += df_db1
b2_grad += df_db2
```

After this we add the gradients for the weight decay to the gradients of W1 and W2 (since this only needs to be added once).

```
Val_loss 0.10551628768207254
Val_acc 0.9685555555555555
Start epoch  3
Train_loss 0.11496183068501209
Train_acc 0.9829607843137255
Val_loss 0.08981077748229496
Val_acc 0.9725555555555555
Start epoch  4
Train_loss 0.10919021895996146
Train_acc 0.9842156862745098
Val_loss 0.08857105252075169
Val_acc 0.9741111111111111
Start epoch  5
Train_loss 0.10317178921518239
Train_acc 0.9864509803921568
Val_loss 0.0859835048091689
Val_acc 0.9737777777777777
Start epoch  6
Train_loss 0.0962390943272102
Train_acc 0.9892156862745098
Val_loss 0.07791856046835044
Val_acc 0.9756666666666667
Stopping early
in sample accuracy 0.9871833333333333
test sample accuracy 0.9762976297629763
outputting to file epoch_plots.png

(base) C:\Users\rsmj9\Documents\Datalogi\5.
```




The loss and accuracy “drop off” when we stop early after 6 epochs. We stop early if our validation accuracy is above 0.975 and the difference in validation accuracy from one epoch to the next is less than 0.005.