TDT4265 Assignment 1

Anders Thallaug Fagerli

Task 1

a)

For logistic regression, the cost function is given as

$$C^n(w) = -(y^n \ln(\hat{y}^n) + (1-y^n)\ln(1-\hat{y}^n)), \quad \hat{y}^n = f(x) = \frac{1}{1+e^{-w^\top x^n}}.$$

We take the partial derivative, and seperate the gradient into two terms,

$$\frac{\partial C^n(w)}{\partial w_i} = -y^n \frac{\partial}{\partial w_i}\Big[\ln(\hat{y}^n)\Big] - (1-y^n)\frac{\partial}{\partial w_i}\Big[\ln(1-\hat{y}^n)\Big].$$

Using the chain rule, the first term is derived as

$$\begin{aligned}
\frac{\partial}{\partial w_i}\Big[\ln(\hat{y}^n)\Big] &= \frac{1}{\hat{y}^n} \cdot \frac{\partial}{\partial w_i}\Big[\hat{y}^n\Big] \\
&= \frac{1}{\hat{y}^n} \cdot x_i^n f(x^n)(1-f(x^n)) \\
&= \frac{1}{\hat{y}^n} \cdot x_i^n \hat{y}^n(1-\hat{y}^n) \\
&= x_i^n(1-\hat{y}^n).
\end{aligned}$$

Similarly, the second term is derived as

$$\begin{aligned}
\frac{\partial}{\partial w_i}\Big[\ln(1-\hat{y}^n)\Big] &= \frac{1}{1-\hat{y}^n} \cdot \frac{\partial}{\partial w_i}\Big[1-\hat{y}^n\Big] \\
&= -\frac{1}{1-\hat{y}^n} \cdot x_i^n f(x^n)(1-f(x^n)) \\
&= -\frac{1}{1-\hat{y}^n} \cdot x_i^n \hat{y}^n(1-\hat{y}^n) \\
&= -x_i^n \hat{y}^n.
\end{aligned}$$

In total, we then get

$$\begin{aligned}
\frac{\partial C^n(w)}{\partial w_i} &= -y^n x_i^n(1-\hat{y}^n) + (1-y^n)x_i^n\hat{y}^n \\
&= -y^n x_i^n + y^n x_i^n \hat{y}^n + x_i^n \hat{y}^n - y^n x_i^n \hat{y}^n \\
&= -(y^n - \hat{y}^n)x_i^n.
\end{aligned}$$

b)

For softmax regression, the multi-class cross entropy cost is given as

$$C^n(w) = -\sum_{k=1}^{K} y_k^n \ln(\hat{y}_k^n), \quad \hat{y}_k^n = \frac{e^{w_k^\top x^n}}{\sum_{k'}^{K} e^{w_{k'}^\top x^n}}. \tag{1}$$

Writing out the terms, this can be written as

$$C^n(w) = -\sum_{k=1}^{K} y_k^n w_k^\top x^n + \sum_{k=1}^{K} y_k^n \ln\left(\sum_{k'}^{K} e^{w_{k'}^\top x^n}\right).$$

The gradient can then for the softmax regression be split into two terms,

$$\frac{\partial C^n(w)}{\partial w_{kj}} = -\frac{\partial}{\partial w_{kj}}\left[\sum_{k=1}^{K} y_k^n w_k^\top x^n\right] + \frac{\partial}{\partial w_{kj}}\left[\sum_{k=1}^{K} y_k^n \ln\left(\sum_{k'}^{K} e^{w_{k'}^\top x^n}\right)\right].$$

The first term is simplest, and gives

$$\frac{\partial}{\partial w_{kj}}\left[\sum_{k=1}^{K} y_k^n w_k^\top x^n\right] = y_k^n x_j^n.$$

For the second term, all $k' \neq k$ can be treated as constants, and disappear in the differentiation. The partial derivative is thus derived as

$$\begin{aligned}
\frac{\partial}{\partial w_{kj}}\left[\sum_{k=1}^{K} y_k^n \ln\left(\sum_{k'}^{K} e^{w_{k'}^\top x^n}\right)\right] &= \sum_{k=1}^{K} y_k^n \frac{\partial}{\partial w_{kj}}\left[\ln\left(\sum_{k'}^{K} e^{w_{k'}^\top x^n}\right)\right] \\
&= \frac{1}{\sum_{k'}^{K} e^{w_{k'}^\top x^n}} \cdot x_j^n e^{w_k^\top x^n} \sum_{k=1}^{K} y_k^n \\
&= \hat{y}_k^n x_j^n \sum_{k=1}^{K} y_k^n, \qquad \sum_{k=1}^{K} y_k^n = 1 \\
&= \hat{y}_k^n x_j^n.
\end{aligned}$$

In total, this gives the gradient

$$\frac{\partial C^n(w)}{\partial w_{kj}} = -x_j^n(y_k^n - \hat{y}_k^n)$$

Task 2

b)

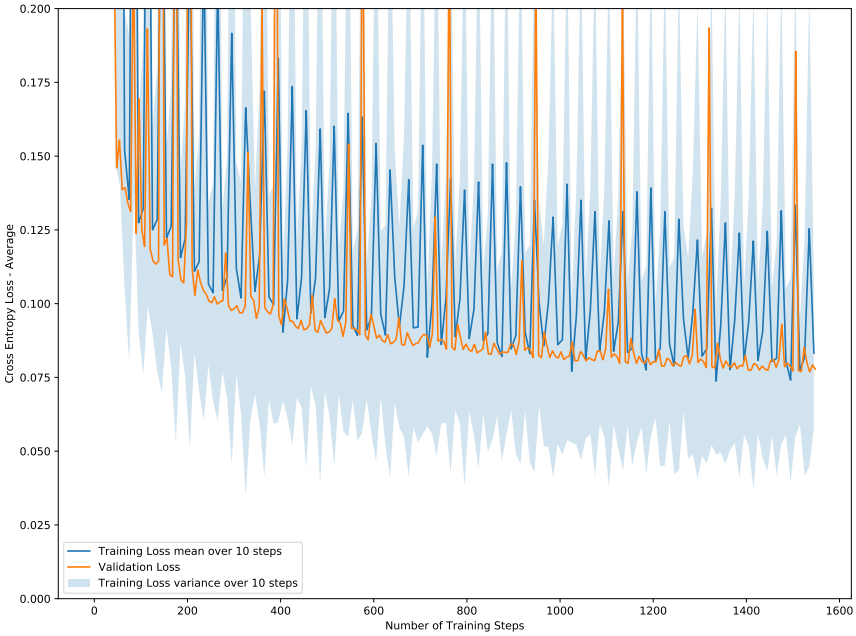

Figure 1: *Training and validation loss over training.*

c)

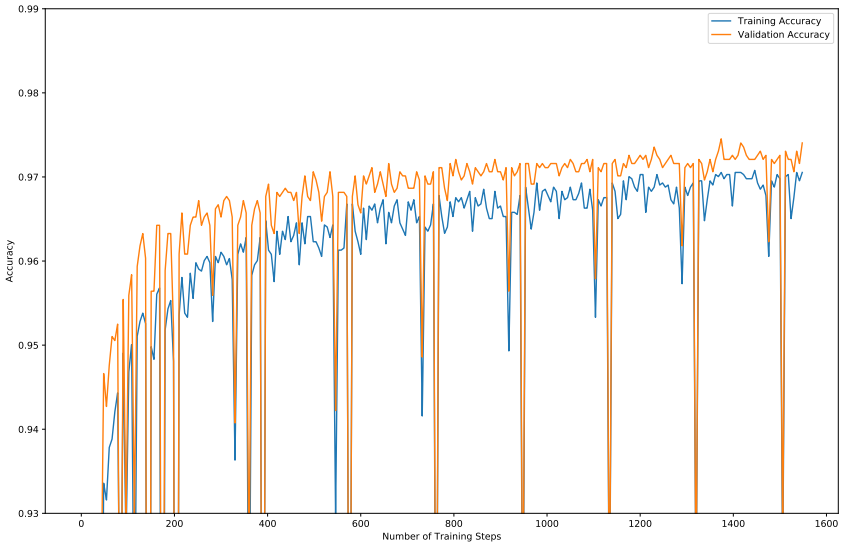

Figure 2: *Accuracy on training and validition set over training.*

d)

Early stopping kicked in after 35 epochs. Side note: one should perhaps use the weights at the validation loss minimum as final weights, but I have not implemented this in the code, as it does not seem the "Starter code" was made with this in mind. The final weights are therefore the weights at the stopping criteria.

e)

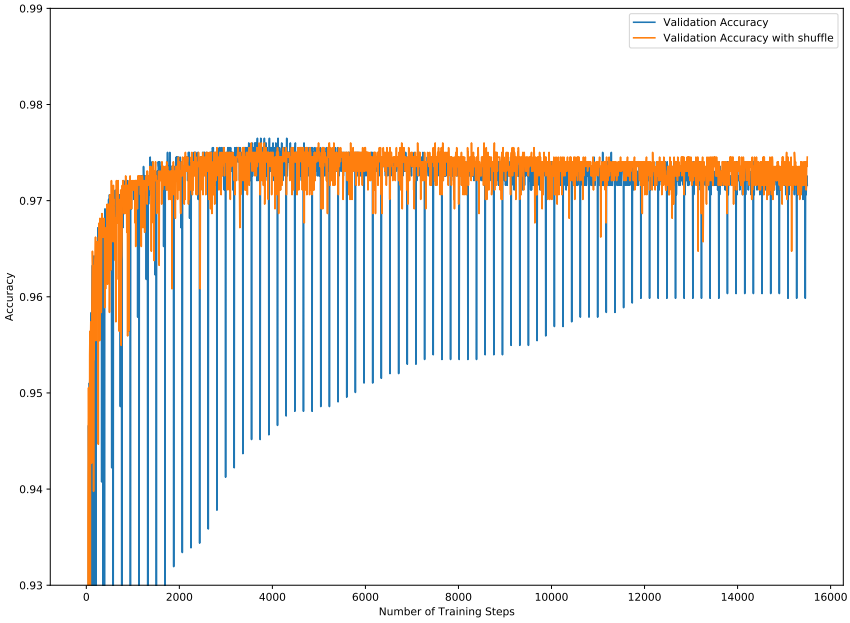

Figure 3: *Accuracy on validation set with and without dataset shuffling. Early stopping is turned off to better see the difference.*

From Figure 3 we see that the validation accuracy has less spikes with dataset shuffling than without. We also see that the spikes come periodically, and by closer inspection, the spikes are seen to come from batch indice [2304, 2431]. This may indicate that this specific batch is not representing the validation set well, and training on this batch will therefore give worse accuracy on the validation set, as the weights are updated to best fit this specific batch. When the data is shuffled, the training step will train over a more generalized set of data that represents the overall distribution of data better. The "poor" batch will effectively no longer exist, and we get the benifit of training over a larger distribution of data while still doing batch gradient descent.

Task 3

b)

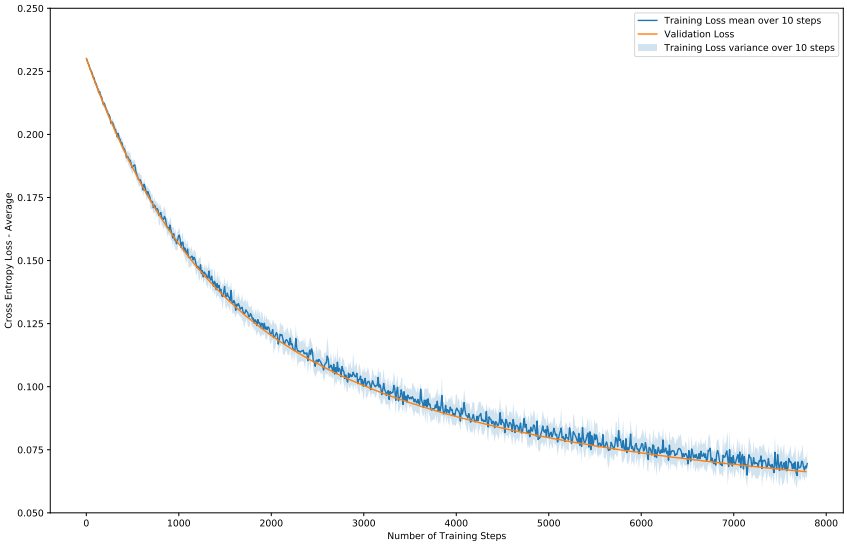

Figure 4: *Training and validation loss over training.*

c)

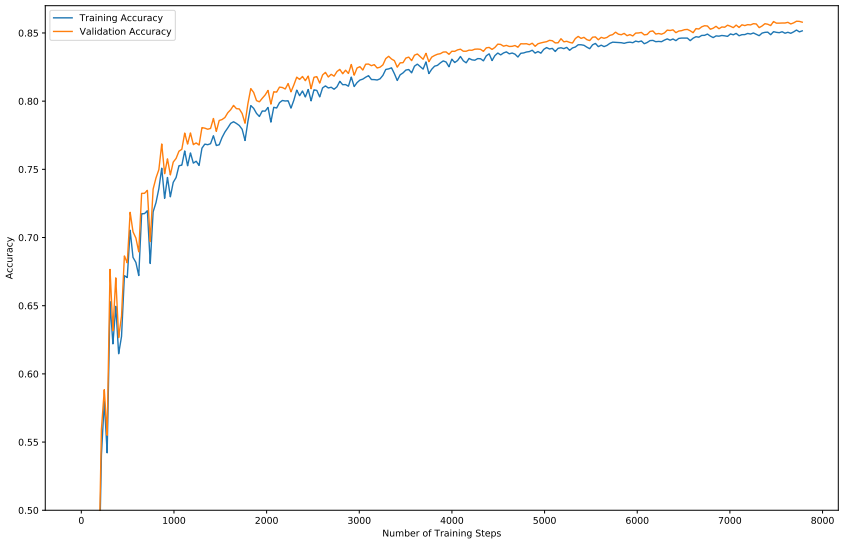

Figure 5: *Accuracy on training and validation set over training.*

d)

From Figure 4 and Figure 5 we see that the validation set loss and accuracy follows the training set, so we cannot say there are signs of overfitting based on this. One would typically see that the validation set accuracy deterioates after a while when overfitting, and that the training set achieves a lower loss and greater accuracy than the validation set. As this is not the case, we cannot say there are signs of overfitting, but a test set would help determining this. We should however not exclude overfitting solely based on the validation set, as the data in the validation set may be very similar to the training set, and therefore not represent the whole distribution of data well enough. However, I think it is odd that the model acheives better accuracy on the validation set than on the training set, so there may be something wrong in the code.

Task 4

a)

The new objective function with regularization is

$$J(w) = C(w) + \lambda R(w), \qquad R(w) = ||w||^2 = \sum_{i,j} w_{i,j}^2,$$

where $C(w)$ is defined by (1). The new gradient for the backward pass is then

$$\begin{aligned}\frac{\partial J^n(w)}{\partial w_{kj}} &= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial C^n(w)}{\partial w_{kj}} + \frac{\partial R(w)}{\partial w_{kj}} \\ &= -\frac{1}{N}\sum_{n=1}^{N} x_j^n(y_k^n - \hat{y}_k^n) + 2\lambda w_{kj}\end{aligned}$$

b)

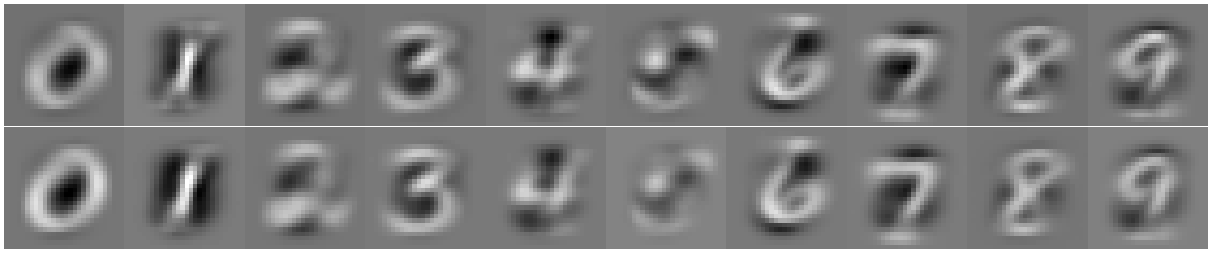

Figure 6: *Weights for model with $\lambda = 0$ (top row) and $\lambda = 1$ (bottom row).*

In Figure 6 we see the weights without regularization (top row) and with regularization (bottom row). Although the difference isn't as apparent as in the assignment text, we see that the bottom row is slightly more smoothed out. This is because the weights are overall smaller in value for the regularized model, as we in a sense have penalized the use of larger weights by the parameter λ. For the unregularized model there is no constraint on the magnitude of the weights, leading to possibly large contrasts between large and low values, which gives a more "noisy" image.

c)

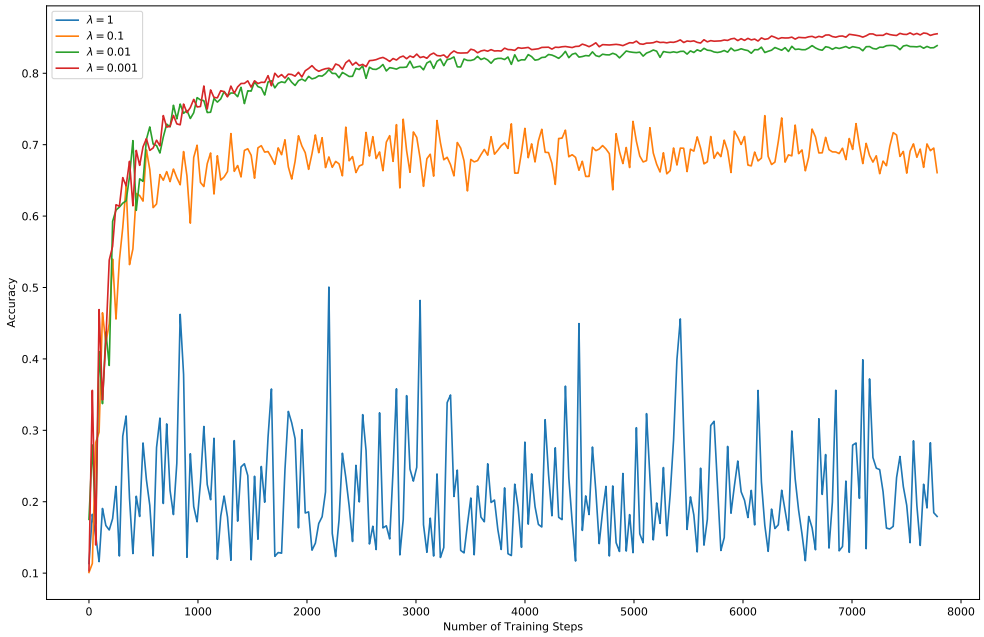

Figure 7: *Validation set accuracy for different* λ.

d)

From Figure 7 we see that the validation set accuracy degrades with increased regularization. I think this somewhat goes against the intentions of regularizing the model, as we use regularization for better generalization, which should give decent performance on the validation set. It may however be the case that the model is simple enough as it is, and by penalizing the complexity we get a so simple model that we underfit the data.

e)

From Figure 8 we see that the length decreases with increasing λ, meaning the average values of w decrease. This is evidence that regularization penalizes the weights, leading to simpler models.

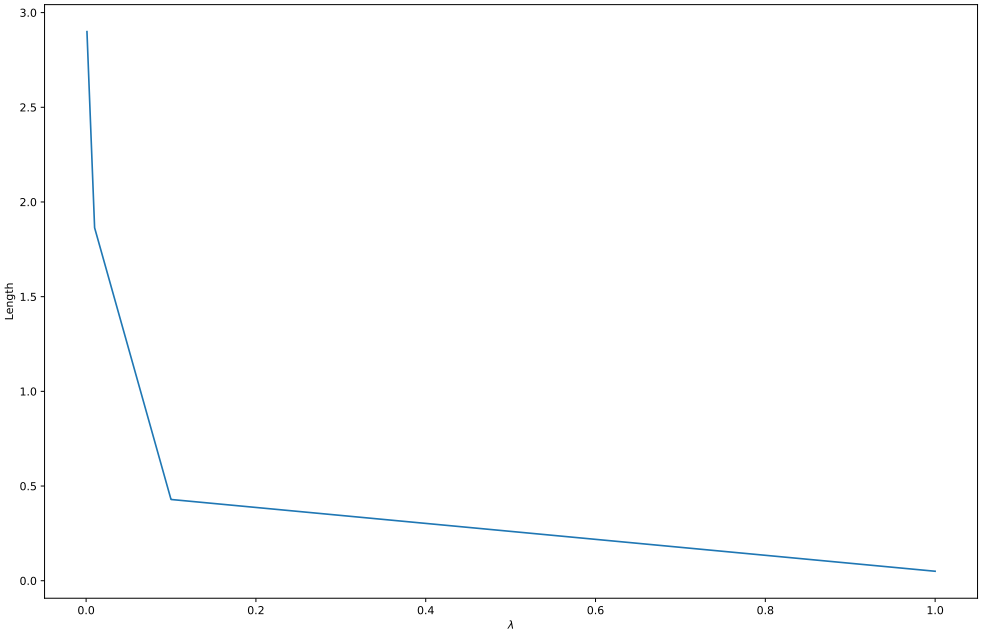

Figure 8: *L_2 norm of weight vector w for different λ.*