TDT4265 - Computer Vision and Deep Learning

Assignment 1

Written Spring 2020 By Dinosshan Thiagarajah and Kyrre S. Haugland

Task 1: Theory

Task 1a: Derive the gradient for Logistic Regression

$$
\begin{aligned}
C^n(w) &= -y^n ln(\hat{y}^n) + (1-y^n) ln(1-\hat{y}^n)\\
\frac{\partial C^n(w)}{\partial w_j} &= -\frac{\partial}{\partial w_j}(y^n ln(\hat{y}) + (1-y^n) ln(1-\hat{y}^n))\\
\textbf{insert } \hat{y}^n &= f_w(x^n)\\
&= -\frac{\partial}{\partial w_j}(y^n ln(f_w(x^n)) + (1-f_w(x^n)) ln(1-f_w(x^n)))\\
\textbf{using hint}&\\
&= -(y^n \frac{x_j^n f_w(x^n)}{f_w(x^n)}(1-f_w(x^n)) + \frac{(1-y^n)(-x_j^n f_w(x^n)(1-f_w(x^n))}{(1-f_w(x^n))})\\
&= -(y^n x_j^n (1-f_w(x^n)) + (1-y^n)(-x_j^n f_w(x^n)))\\
&= -(y^n - \hat{y}^n) x_j^n
\end{aligned}
$$

Task 1b: Derive the gradient for Softmax Regression

$$
\begin{aligned}
C^n(w) &= -\frac{1}{K}\sum_{k=1}^{K} y_k^n ln(\hat{y}_k^n)\\
\textbf{insert } \hat{y}_k^n &= \frac{e^{w_k^T x^n}}{\sum_{k'}^{K} e^{w_{k'}^T x^n}} \textbf{ and using hint}\\
\frac{\partial C^n(w)}{\partial w_{kj}} &= -\frac{\partial}{\partial w_{kj}}\frac{1}{K}\sum_{k=1}^{K}(y_k^n(ln(e^{w_k^T x^n}) - ln(\sum_{k'}^{K} e^{w_{k'}^T x^n})))\\
&= -\frac{\partial}{\partial w_{kj}}\frac{1}{K}\sum_{k=1}^{K}(y_k^n(w_k^T x^n) + \frac{\partial}{\partial w_{kj}}\frac{1}{K}\sum_{k=1}^{K} y_k^n ln(\sum_{k'}^{K} e^{w_{k'}^T x^n})))
\end{aligned}
$$

when k=j

$$
\frac{\partial}{\partial w_{kj}}\frac{1}{K}\sum_{k=1}^{K}(y_k^n(w_k^T x^n) = \begin{cases} -\frac{1}{K} y_k^n x_j^n & k = kj\\ 0 & k \neq kj \end{cases}
$$

Using chain rule

$$
\frac{\partial}{\partial w_{kj}}\frac{1}{K}\sum_{k=1}^{K} y_k^n ln(\sum_{k'}^{K} e^{w_{k'}^T x^n}))) = \frac{1}{K} y_k^n \frac{1}{\sum_{k'}^{K} e^{w_{k'}^T x^n}} e^{w_{k'}^T x^n} x_j^n
$$

Merging both sums

$$
= -\frac{1}{K} y_k^n x_j^n + \frac{1}{K} y_k^n \frac{1}{\sum_{k'}^{K} e^{w_{k'}^T x^n}} e^{w_{k'}^T x^n} x_j^n
$$

Using $\sum_{k=1}^{K} y_k^n = 1$ **and** $\hat{y}_k^n = \frac{e^{w_k^T x^n}}{\sum_{k'}^{K} e^{w_{k'}^T x^n}}$

$$
= -\frac{1}{K} x_j^n (y_k^n - \hat{y}_k^n)
$$

Task 2: Logistic Regression through Gradient Descent

Task 2b:

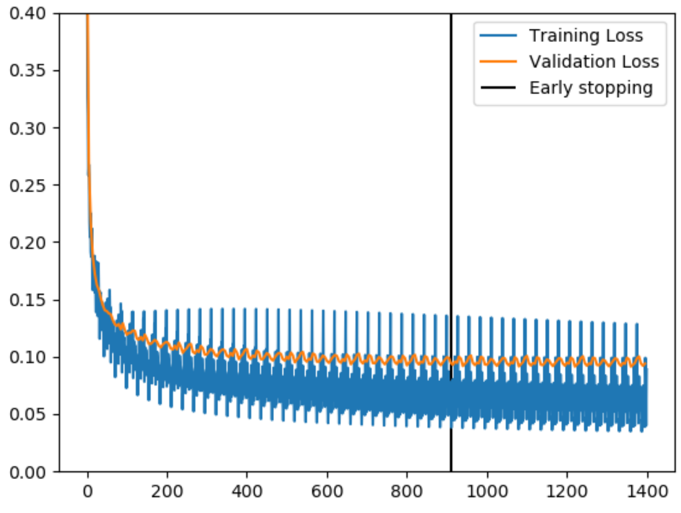

Figure 1: Plot of training and validation loss over training

Task 2c:

Datatset	Accuracy
Train	0.9722
Validation	0.9600
Test	0.9802

Table 1: Accuracy of trained model

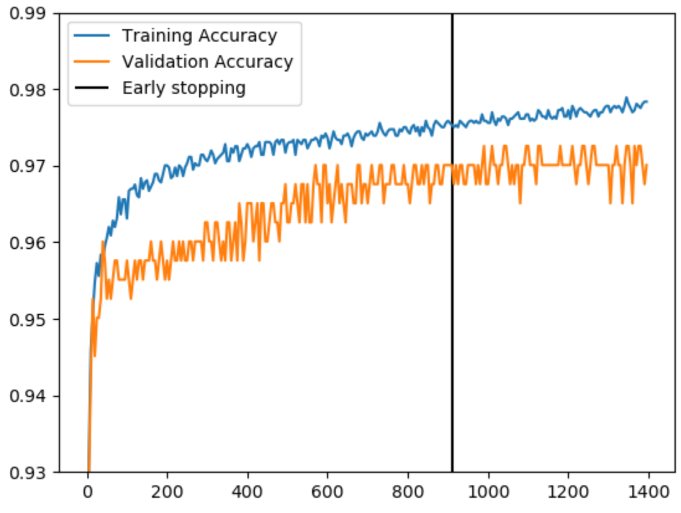

Figure 2: Plot of accuracy on the training set and validation set over training

Task 2d:

Stopping criteria that is chosen for this part is that the training is stopped if the validation loss increases consistently after passing through 20% of the train dataset 3 times. With this implementation, the early stopping kicked in at step number 910 as seen from task 2d and 2c. Since the early stopping kicked in at the right time, signs of overfitting is not observed at this stage. More sophisticated early stopping methods can be implemented as well. For instance filtering the validation loss data before using it as part of the stopping criteria could be an addition, but this is not implemented for this assignment. Further, without the early stopping, choosing the epoch number to 500 leads to signs of overfitting as validation accuracy decreases while the train accuracy increases.

Task 3: Regularization

Task 3a:

$$R = \|w\|^2 = \sum_{i,j} w_{i,j}^2$$

$$\frac{\partial}{\partial w_k} R(w) = \sum_{i,j} \frac{\partial}{\partial w_k} w_{i,j}^2$$

$$\frac{\partial}{\partial w_k} w_{i,j}^2 = \begin{cases} 0 \text{ if } k \neq i,j \\ 2w_k \text{ else} \end{cases}$$

then it follows:

$$\frac{\partial R(w)}{\partial w} = 2w$$

Hence the update term can be written as:

$$\Delta\omega = -\alpha \left(\frac{1}{N} \sum_{n=1}^{N} \frac{\partial J^n(\omega)}{\partial \omega_{kj}} \right) \tag{1}$$

$$= -\alpha \left(\frac{1}{N} \sum_{n=1}^{N} \frac{\partial C^n(\omega)}{\partial \omega_{kj}} + \lambda \frac{\partial R^n(\omega)}{\partial \omega_{kj}} \right) \tag{2}$$

$$\underline{= -\alpha \left(\frac{1}{N} \sum_{n=1}^{N} -\frac{1}{K} x_j^n (y_k^n - \hat{y}_n^k) + 2\lambda \omega_{k,j} \right)} \tag{3}$$

Task 3b:

From figure 3 we observe that for lower values of *lambda* the validation accuracy increases. This is due to reduced model complexity in this case is causing less generalization. As we want as high validation accuracy as possible $\lambda = 0.001$ is the most desirable values of this pool of *lambda* values.

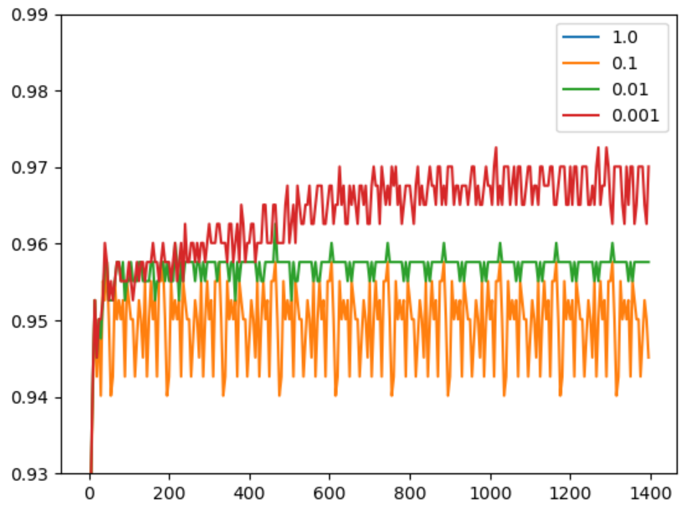

Figure 3: Plot of validation accuracy for different λ

Task 3c:

It can be seen below that the length of weight vector decreases with higher λ.

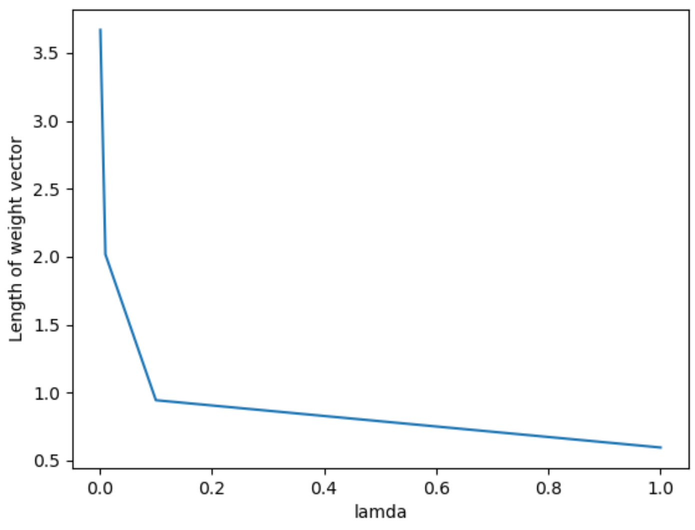

Figure 4: Plot of length of weight vector vs λ

Task 3d:

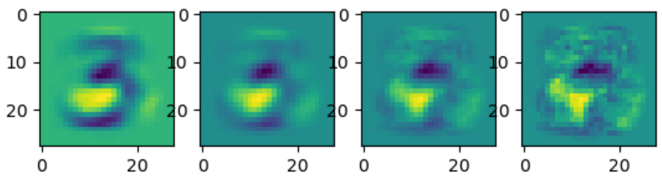

Figure 5: Task 3 weights for $\lambda = 1.0$ (leftmost image), to $\lambda = 0.001$ (rightmost image)

Task 4: Softmax Regression through Gradient Descent

Task 4b:

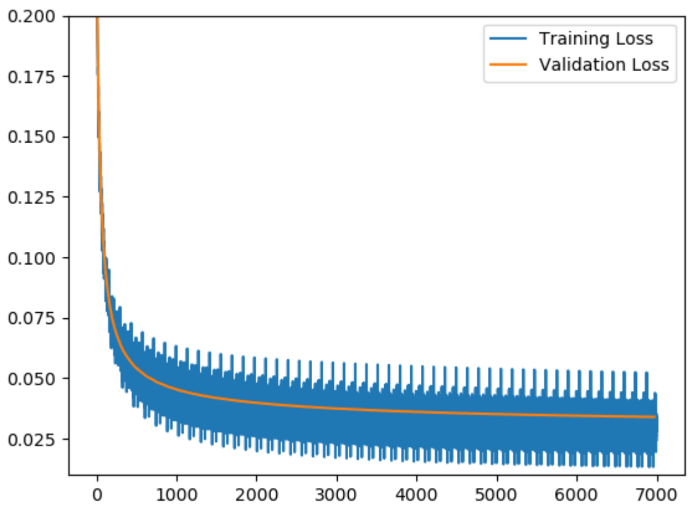

Figure 6: Plot of training and validation loss over training using the MNIST dataset

Datatset	Accuracy
Train	0.9208
Validation	0.9060
Test	0.9395

Table 2: Accuracy of trained model

Task 4c:

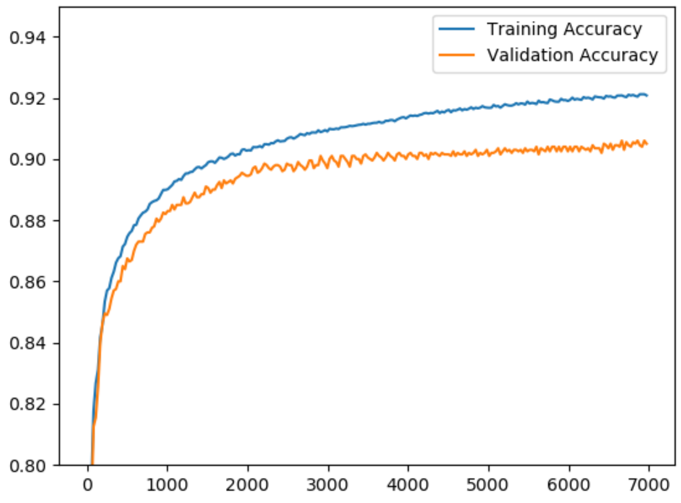

Figure 7: Plot of accuracy on the training set and validation set over training using the MNIST dataset

Task 4e:

In the figure below one can see the weights for using $\lambda = 0$ and $\lambda = 0.1$. Comparing top and bottom row, we see that the weights of $\lambda = 0.1$ are less noisy than when $\lambda = 0$. This is because ridge regression (L2 regularization) gives penalties to the model for being too complex. This complexity is seen through the noise, however choosing a too large λ will create a model that is not sufficiently complex, restricting generalization.

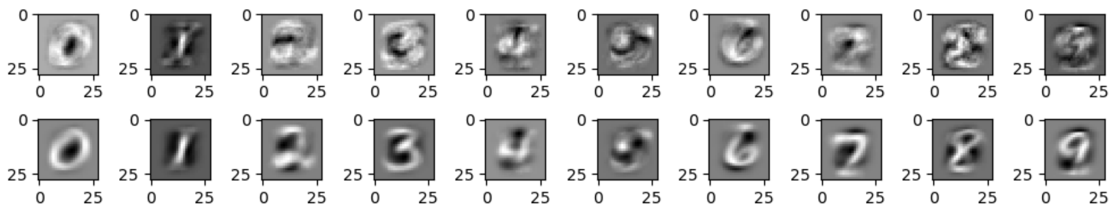

Figure 8: Task 4 visualization of weights for a model with $\lambda = 0.0$ (top row), and $\lambda = 0.1$ (bottom row)