CSC311 Assignment 2

July 7, 2023

1. Moved to Assignment 3.

2. (a) Suppose that the dataset is linearly separable. Then, the data space can be separated into two half spaces, one containing $(-2, 1), (2, 3)$ and the other containing $(1, 2)$. Since half spaces are convex, any line connecting two points in a half space is also in the half space. Defining $a = (-2, -1), b = (2.3)$, the line

$$t(a) + (1 - t)(b) \text{ for some } t \in [0, 1]$$

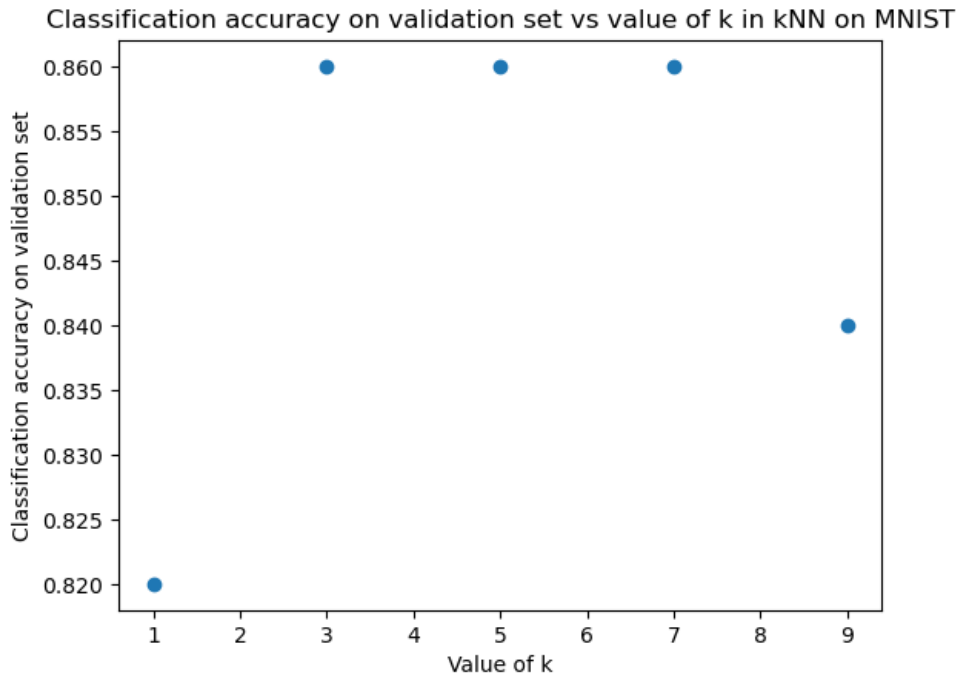
then must also lie in the half space of a and b. However, $(1, 2)$ is on the line since $t = 0.25$ gives $0.25(-2, -1) + 0.75(2, 3) = (1, 2)$, meaning $(1, 2)$ is in the same half space as a and b, which is a contradiction. Thus, the dataset is not linearly separable.

(b) The constraints are

$$\begin{aligned} -2w_1 + w_2 &\geq 0 \\ w_1 + 4w_2 &< 0 \\ 2w_1 + 9w_2 &\geq 0 \end{aligned}$$

(c) See the last page.

3. (a)



(b) I choose $k^* = 7$. Its accuracy is 0.860, which is high and indicates that underfitting is unlikely. Also, $k = 3, 5$ have very similar accuracies; compared to them, $k^* = 7$ overfits the least since $7 > 5 > 3$.

For $k = 5$, the validation accuracy is 0.86 and the test accuracy is 0.94.
For $k = 7$, the validation accuracy is 0.86 and the test accuracy is 0.94.
For $k = 9$, the validation accuracy is 0.84 and the test accuracy is 0.88.

As k increases, both accuracies tend to decrease since higher k leads to underfitting. Similarly, as k decreases, both accuracies also tend to decrease since lower k leads to overfitting.

(c) This is question 3.2 (a). See the code.

(d) This is question 3.2 (b). My value of run_check_grad is 2.8180701551567397e-08, which indicates that my implementation of part (a) is correct.

To find the best hyperparameters for mnist_train and mnist_train_small, I adjusted the learning rate and number of iterations. For the learning rate, I tried values from 0.001 to 0.1, and for the number of iterations I tried values from 100 to 2000. I selected the best one according to which one has the highest accuracies.

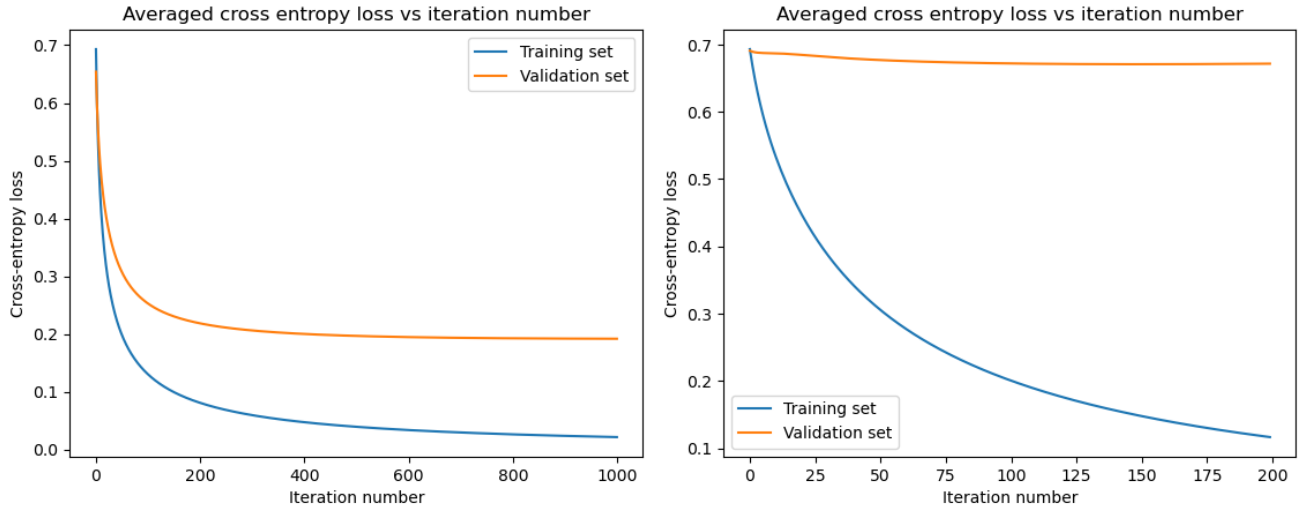
The best hyperparameters I found for mnist_train are $n = 1000$ and $\alpha = 0.1$. Its results are:

```
Training cross entropy loss: 0.021540517909554163, training classification accuracy: 1.0
Validation cross entropy loss: 0.19179991363518784, validation classification accuracy: 0.88
Test cross entropy loss: 0.21421419249319446, test classification accuracy: 0.92
```

The best hyperparameters I found for mnist_train_small iare $n = 200$ and $\alpha = 0.01$. Its results are:

```
Training cross entropy loss: 0.11659436923096456, training classification accuracy: 1.0
Validation cross entropy loss: 0.6716093269532207, validation classification accuracy: 0.64
Test cross entropy loss: 0.5411473066396657, test classification accuracy: 0.76
```

(e) This is question 3.2 (c). The left plot is for nmist_train and the right is for mnist_train_small.

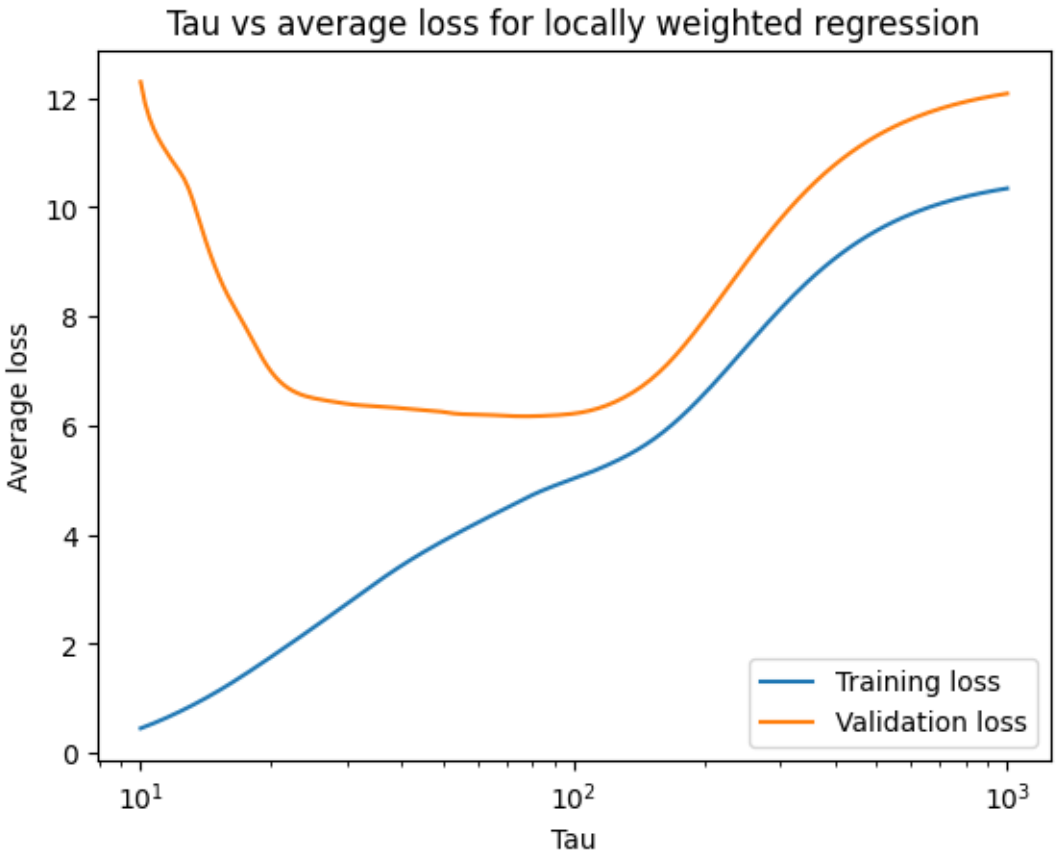


The results do not change after running my code several times. If they did, I would choose the best hyperparameters by taking the average of the best values.

4. (a) See the last page.

(b) See the code.

(c)



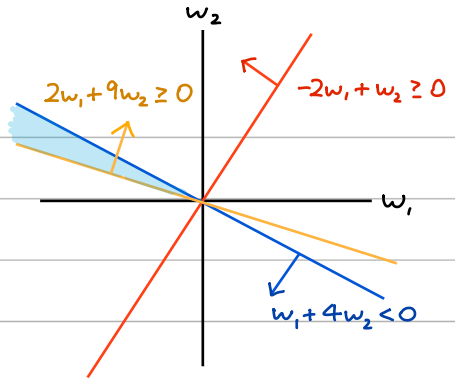
(d) As $\tau \to \infty$, $a^{(i)} \to \dfrac{e^{-0}}{e^{-0} + \ldots + e^{-0}} = \dfrac{1}{1 + \ldots + 1} = \dfrac{1}{N}$. This means the weights of the algorithm are being averaged out, meaning the algorithm behaves more like standard least squares regression for these values.

As $\tau \to 0$, $a^{(i)} \to \dfrac{e^{-\infty}}{e^{-\infty} + \ldots + e^{-\infty}} = \dfrac{0}{0 + \ldots + 0}$ which DNE. This means the algorithm behaves more erratically for these values, which is reflected by the higher loss around $\tau = 10^1$ on the above plot.

(e) An advantage is that it may be more locally accurate since the weights are calculated local to each point. For this reason, it also works well for non-linear data.

A disadvantage is that it is computationally slow and that it is sensitive to noise, including outliers, which may reduce the accuracy.

2.c)



The feasible area is the shaded region.

4.a) Since $\vec{w}^*$ corresponds to the minimum, we solve for $\vec{w}$ in $\frac{\partial}{\partial \vec{w}}\left(\frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)}-\vec{w}^T x^{(i)})^2+\frac{\lambda}{2}\|\vec{w}\|^2\right)=0$.

$$0=\frac{\partial}{\partial \vec{w}}\left(\frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)}-\vec{w}^T\vec{x}^{(i)})^2+\frac{\lambda}{2}\|\vec{w}\|^2\right)=\frac{1}{2}\sum_{i=1}^{N} a^{(i)}\frac{\partial}{\partial \vec{w}}(y^{(i)}-\vec{w}^T\vec{x}^{(i)})^2+\frac{\partial}{\partial \vec{w}}\frac{\lambda}{2}\|\vec{w}\|^2$$

$$=\frac{1}{2}\sum_{i=1}^{N} 2a^{(i)}(y^{(i)}-\vec{w}^T\vec{x}^{(i)})\frac{\partial}{\partial \vec{w}}(y^{(i)}-\vec{w}^T\vec{x}^{(i)})+\frac{\lambda}{2}2\vec{w} \qquad \text{since } \frac{\partial}{\partial \vec{w}}\|\vec{w}\|^2=2\vec{w}$$

$$=\sum_{i=1}^{N} a^{(i)}(y^{(i)}-\vec{w}^T\vec{x}^{(i)})(-\vec{x}^{(i)})+\lambda\vec{w} \qquad \text{since } \frac{\partial}{\partial \vec{w}}\vec{w}^T\vec{x}^{(i)}=\frac{\partial}{\partial \vec{w}}\left(\sum_{k=1}^{D} w_k x_k^{(i)}\right)$$

$$=(w_1,\dots,w_D)=\vec{w}$$

$$=-\sum_{i=1}^{N}\vec{x}^{(i)}a^{(i)}y^{(i)}+\sum_{i=1}^{N}\vec{x}^{(i)}a^{(i)}\vec{w}^T\vec{x}^{(i)}+\lambda\vec{w} \qquad (1)$$

Notice that:

$$\sum_{i=1}^{N}\vec{x}^{(i)}a^{(i)}y^{(i)}=\begin{bmatrix} x_1^{(1)}a^{(1)}y^{(1)}+\dots+x_1^{(N)}a^{(N)}y^{(N)} \\ \vdots \\ x_D^{(1)}a^{(1)}y^{(1)}+\dots+x_D^{(N)}a^{(N)}y^{(N)}\end{bmatrix}=\begin{bmatrix} x_1^{(1)} & \dots & x_1^{(N)} \\ \vdots & \ddots & \vdots \\ x_D^{(1)} & \dots & x_D^{(N)}\end{bmatrix}\begin{bmatrix} a^{(1)}y^{(1)} \\ \vdots \\ a^{(N)}y^{(N)}\end{bmatrix}$$

$$=\begin{bmatrix} x_1^{(1)} & \dots & x_1^{(N)} \\ \vdots & \ddots & \vdots \\ x_D^{(1)} & \dots & x_D^{(N)}\end{bmatrix}\begin{bmatrix} a^{(1)} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a^{(N)}\end{bmatrix}\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)}\end{bmatrix}=X^TA\vec{y}$$

Similarly,

$$\sum_{i=1}^{N}\vec{x}^{(i)}a^{(i)}\vec{w}^T\vec{x}^{(i)}=\begin{bmatrix} x_1^{(1)} & \dots & x_1^{(N)} \\ \vdots & \ddots & \vdots \\ x_D^{(1)} & \dots & x_D^{(N)}\end{bmatrix}\begin{bmatrix} a^{(1)} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a^{(N)}\end{bmatrix}\begin{bmatrix} w_1x_1^{(1)}+\dots+w_Dx_D^{(1)} \\ \vdots \\ w_1x_1^{(N)}+\dots+w_Dx_D^{(N)}\end{bmatrix}$$

$$=X^TA\begin{bmatrix} x_1^{(1)} & \dots & x_D^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \dots & x_D^{(N)}\end{bmatrix}\begin{bmatrix} w_1 \\ \vdots \\ w_D\end{bmatrix}=X^TAX\vec{w}$$

Then, (1) becomes

$$0=-X^TA\vec{y}+X^TAX\vec{w}+\lambda\vec{w} \implies X^TA\vec{y}=X^TAX\vec{w}+\lambda I\vec{w}=(X^TAX+\lambda I)\vec{w}$$

where I is the identity matrix

so $\vec{w}=(X^TAX+\lambda I)^{-1}X^TA\vec{y}$.