CSC311, Fall 2022, Homework 2

Huipeng Lei, 1005908154

Due Monday, October 17, 2022

Problem 1

a) The expected loss $\mathbb{E}[\mathcal{J}(y,t)]$ for $y = Keep$:

$$\begin{aligned}\mathbb{E}[\mathcal{J}(y = Keep, t)] &= P(t = NonSpam) * \mathcal{J}(y = Keep, t = NonSpam) \\ &\quad + P(t = Spam) * \mathcal{J}(y = Keep, t = Spam) \\ &= 0 + 0.2 * 1 \\ &= 0.2\end{aligned}$$

The expected loss $\mathbb{E}[\mathcal{J}(y,t)]$ for $y = Remove$:

$$\begin{aligned}\mathbb{E}[\mathcal{J}(y = Remove, t)] &= P(t = NonSpam) * \mathcal{J}(y = Remove, t = NonSpam) \\ &\quad + P(t = Spam) * \mathcal{J}(y = Remove, t = Spam) \\ &= 0.8 * 500 + 0 \\ &= 400\end{aligned}$$

b) Denote $S = Spam, NS = NonSpam, K = Keep, R = Remove$.
Given the conditional probability $P(t = S|x)$, the Bayes optimal $y* \in \{Keep, Remove\}$ can be obtained by having $y_* = \underset{y \in \{Keep, Remove\}}{argmin} \mathbb{E}[\mathcal{J}(y,t)|x]$. Where
When $\underline{y = Keep}$:

$$\begin{aligned}\mathbb{E}[\mathcal{J}(y = K, t)|x] &= \mathcal{J}(y = K, t = NS) * P(t = NS|x) \\ &\quad + \mathcal{J}(y = K, t = S) * P(t = S|x) \\ &= 0 * P(t = NS|x) + 1 * P(t = S|x) \\ &= P(t = S|x)\end{aligned}$$

When $\underline{y = Remove}$:
Note: $P(t = S|x) = 1 - P(t = NS|x)$

$$\begin{aligned}\mathbb{E}[\mathcal{J}(y = R, t)|x] &= \mathcal{J}(y = R, t = NS) * P(t = NS|x) \\ &\quad + \mathcal{J}(y = R, t = S) * P(t = S|x) \\ &= 500 * P(t = NS|x) + 0 * P(t = S|x) \\ &= 500 * (1 - P(t = S|x))\end{aligned}$$

Since we want to minimize $\mathbb{E}[\mathcal{J}(y,t)|x]$, we can choose the Bayes optimal as

$$\begin{aligned}y_* &= \begin{cases} Keep, & P(t = S|x) \leq 500 * (1 - P(t = S|x)) \\ Remove, & P(t = S|x) > 500 * (1 - P(t = S|x) \end{cases} \\ &= \begin{cases} Keep, & P(t = S|x) \leq \frac{500}{501} \\ Remove, & P(t = S|x) > \frac{500}{501} \end{cases}\end{aligned}$$

c) We know

$$P(x_1, x_2) = P((x_1, x_2)|t = S) * P(t = S) + P((x_1, x_2)|t = NS) * P(t = NS)$$

- $P(x_1 = 0, x_2 = 0) = 0.45 * 0.2 + 0.996 * 0.8 = 0.8868$
- $P(x_1 = 0, x_2 = 1) = 0.25 * 0.2 + 0.002 * 0.8 = 0.0516$
- $P(x_1 = 1, x_2 = 0) = 0.18 * 0.2 + 0.002 * 0.8 = 0.0376$
- $P(x_1 = 1, x_2 = 1) = 0.12 * 0.2 + 0 * 0.8 = 0.024$

From b), we only need to find what $P(t = S|(x_1, x_2))$ is for different combinations of x_1 and x_2 and see which combination decides what the Bayes Optimal is. The formula is

$$P(t = S|(x_1, x_2)) = \frac{P((x_1, x_2)|t = S) * P(t = S)}{P(x_1, x_2)}$$

1. For $x_1 = 0, x_2 = 0$:
$$P(t = S|(x_1, x_2)) = \frac{0.45 * 0.2}{0.8868} = \frac{75}{739} = 0.101488... \leq \frac{500}{501}$$
2. For $x_1 = 0, x_2 = 1$:
$$P(t = S|(x_1, x_2)) = \frac{0.25 * 0.2}{0.0516} = \frac{125}{129} = 0.968992... \leq \frac{500}{501}$$
3. For $x_1 = 1, x_2 = 0$:
$$P(t = S|(x_1, x_2)) = \frac{0.18 * 0.2}{0.0376} = \frac{45}{47} = 0.9574468... \leq \frac{500}{501}$$
4. For $x_1 = 1, x_2 = 1$:
$$P(t = S|(x_1, x_2)) = \frac{0.12 * 0.2}{0.024} = 1 > \frac{500}{501}$$

Based on part b and the result we get, given features x_1, x_2, the Bayes Optimal decision for y_* would be

$$y_* = \begin{cases} Keep, & x_1 = 0, x_2 = 0 \vee x_1 = 0, x_2 = 1 \vee x_1 = 1, x_2 = 0 \\ Remove, & x_1 = 1, x_2 = 1 \end{cases}$$

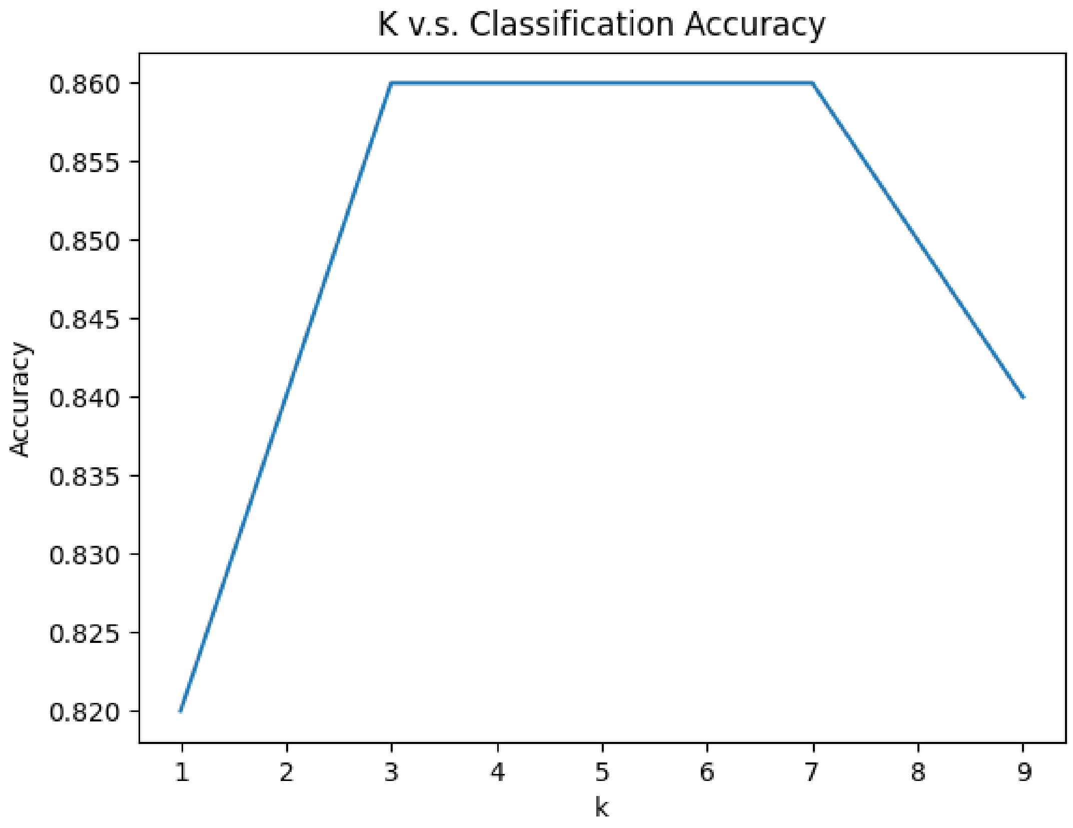
d) According to the Bayes decision rule from part c. The expected loss $\mathbb{E}[\mathcal{J}(y_*, t)]$ is

$$\begin{aligned}
\mathbb{E}[\mathcal{J}(y_*, t)] &= 1 * P(y_* = K, t = S) + 500 * P(y_* = R, t = NS) \\
&= 1 * \Big[P(t = S|(x_1 = 0, x_2 = 0)) * P(x_1 = 0, x_2 = 0) \\
&\quad + P(t = S|(x_1 = 0, x_2 = 1)) * P(x_1 = 0, x_2 = 1) \\
&\quad + P(t = S|(x_1 = 1, x_2 = 0)) * P(x_1 = 1, x_2 = 0)\Big] \\
&\quad + 500 * (1 - P(t = S|(x_1 = 1, x_2 = 1))) * P(x_1 = 1, x_2 = 1) \\
&= 1 * \Big[\frac{75}{739} * 0.8868 + \frac{125}{129} * 0.0516 + \frac{45}{47} * 0.0376\Big] + 500 * (1 - 1) * 0.024 \\
&= 0.176
\end{aligned}$$

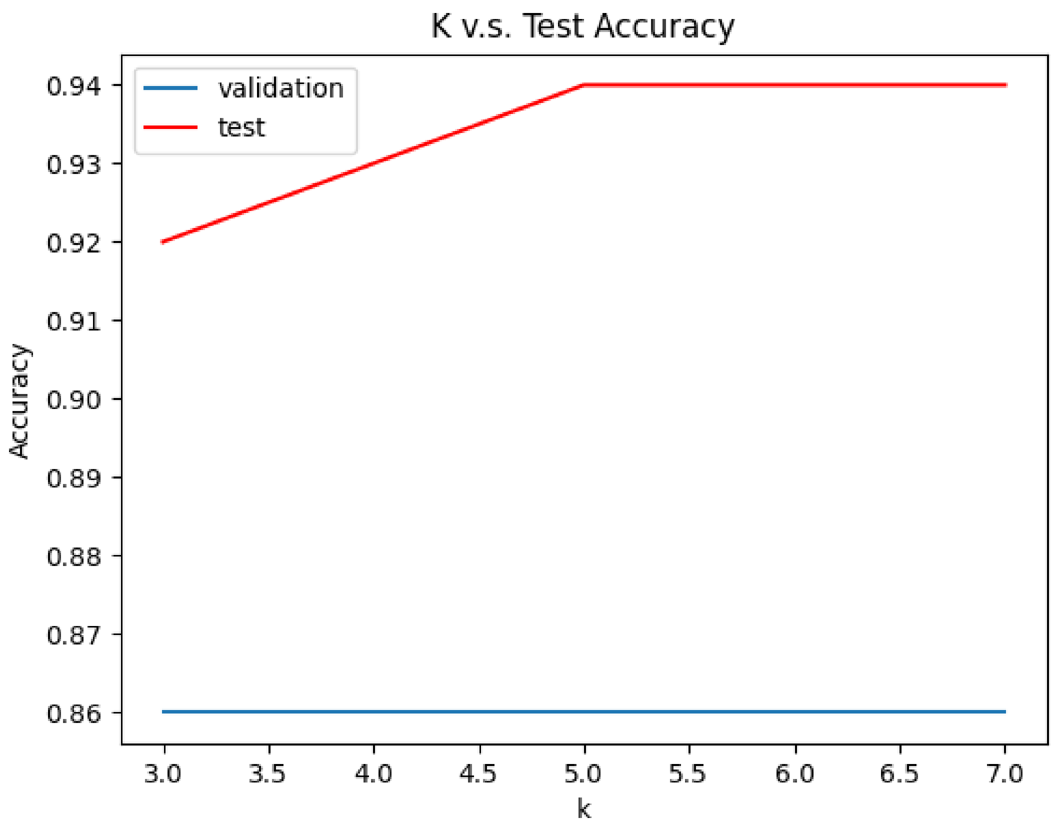
Problem 2

a) From definition, a data-set is linearly separable if the training examples can be perfectly separated by a linear decision rule. However, if we want to visualize this data-set in Data Space, we see that the half-spaces are not convex. For instance, let's draw a line segment connecting two points $(x_1 = -2, x_2 = -1)$ and $(x_1 = 2, x_2 = 3)$ that lie in the positive half-space, we see that the line intersects $(x_1 = 1, x_2 = 2)$, which lies in the negative half-space. Therefore, the data-set is not linearly separable. Furthermore, there exists no linear line that separates this data-set perfectly.

Problem 3



3.1 a)



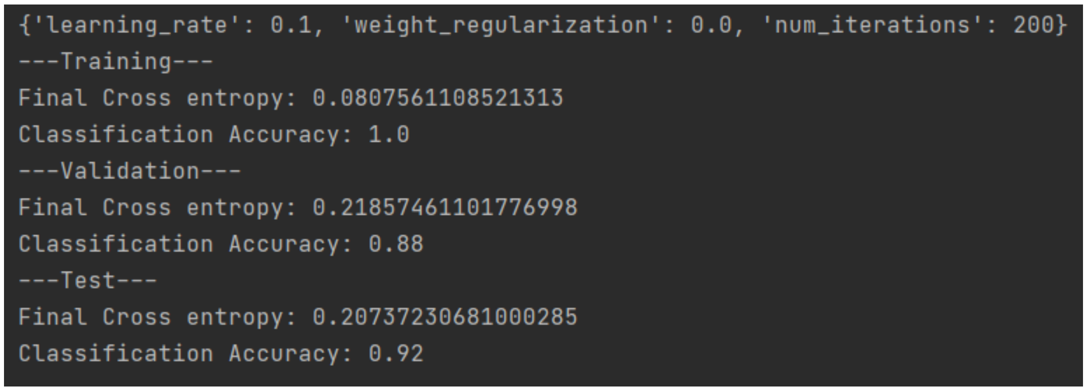
b)

We chose $k^* = 5$ since it has the highest validation accuracy for all $k \in \{1, 3, 5, 7, 9\}$ as shown in part a. From the plot shown above, we see running K-Nearest-Neibors on test inputs has higher accuracy than running KNN on validation inputs. The validation accuracy stays the same for all $k^* - 2, k^*, k^* + 2$ at around 86%, but we see an increase of the test accuracy from 92% when $k = k^* - 2$ to 94% for both $k = k^*$ and $k = k^* + 2$. Moreover, as k changes, the test and validation accuracies changes in the same direction (i.e. positively correlated).

3.2 b) The best hyper-parameter settings I've found is:

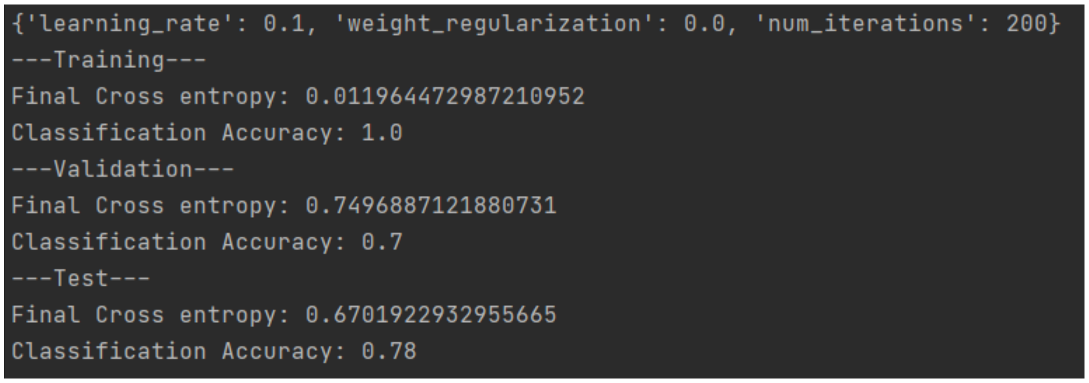
- Learning rate = 0.1
- Weight regularization = 0
- Number of iterations = 200
- Initialize weights = $(M + 1)$ x 1 zero vector

Training set comes from **mnist_train**



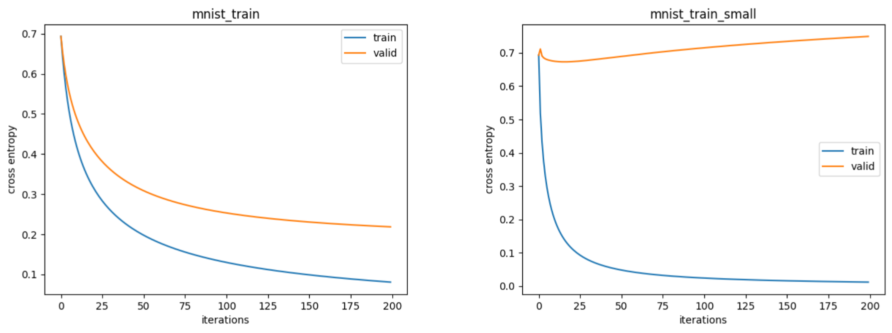
```
{'learning_rate': 0.1, 'weight_regularization': 0.0, 'num_iterations': 200}
---Training---
Final Cross entropy: 0.0807561108521313
Classification Accuracy: 1.0
---Validation---
Final Cross entropy: 0.21857461101776998
Classification Accuracy: 0.88
---Test---
Final Cross entropy: 0.20737230681000285
Classification Accuracy: 0.92
```

Training set comes from **mnist_train_small**



```
{'learning_rate': 0.1, 'weight_regularization': 0.0, 'num_iterations': 200}
---Training---
Final Cross entropy: 0.011964472987210952
Classification Accuracy: 1.0
---Validation---
Final Cross entropy: 0.7496887121880731
Classification Accuracy: 0.7
---Test---
Final Cross entropy: 0.6701922932955665
Classification Accuracy: 0.78
```

c) After running the code five times with the same hyperparameter settings, the result remains the same.



The cross-entropy decreases as the training progresses.

Problem 4

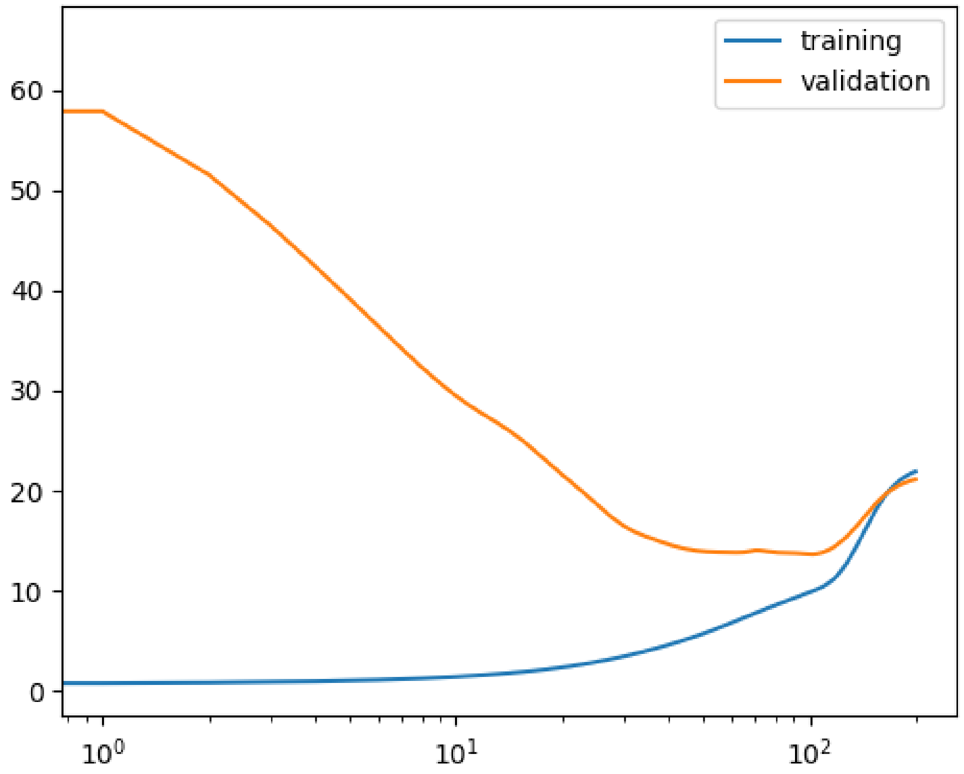
a) Let $\mathcal{J}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T x^{(i)})^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$. We want to find $\mathbf{w}^*$ that minimizes $\mathcal{J}$ by setting the derivative of J w.r.t $\mathbf{w}$ to 0. (i.e. $\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = 0$)
Note:

$$\frac{\partial \mathbf{w}^T}{\partial \mathbf{w}} = 1$$

$$\frac{\partial \mathbf{w}^T\mathbf{w}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}\cdot\mathbf{w}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}}{\partial \mathbf{w}} * \mathbf{w} + \frac{\partial \mathbf{w}}{\partial \mathbf{w}} * \mathbf{w} = 2\mathbf{w}$$

Find $\mathbf{w}^*$:

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \mathbf{w}} &= \sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T x^{(i)})(-x^{(i)}) + \lambda\mathbf{w} \\
&= \sum_{i=1}^{N}(-x^{(i)})a^{(i)}(y^{(i)} - \mathbf{w}^T x^{(i)}) + \lambda\mathbf{w} \\
&= -\mathbf{X}^T\mathbf{A}(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w} \\
&= -\mathbf{X}^T\mathbf{A}\mathbf{y} + \mathbf{X}^T\mathbf{A}\mathbf{w} + \lambda\mathbf{w} \\
0 &\overset{\text{set}}{=} -\mathbf{X}^T\mathbf{A}\mathbf{y} + (\mathbf{X}^T\mathbf{A} + \lambda\mathbf{I})\mathbf{w} \\
(\mathbf{X}^T\mathbf{A} + \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^T\mathbf{A}\mathbf{y} \\
\mathbf{w}^* &= (\mathbf{X}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{A}\mathbf{y}
\end{aligned}$$



c)

d) As $\tau \rightarrow \inf$, with $||x - x^{(i)}||$ and $||x - x^{(j)}||$ are fixed, $a^{(i)} \rightarrow \frac{1}{N}$ because the terms in exponent tends to zero, i.e. $\frac{||x-x^{(i)}||^2}{2\tau^2} \rightarrow 0$. Therefore, the algorithm behaves almost the same as traditional $l2$ regularized linear regressions.

As $\tau \rightarrow 0$, the terms in exponent tends to infinity, which makes $a^{(i)}$ super small. Since τ acts as a bandwidth where it gives more weight to points that are closer to the test example, less to those that are further away. But since $\tau \rightarrow 0$, the weights for any points are super small, i.e. zero. In this case, the algorithm only focuses on the point exactly at the test example, which causes overfitting.

e) **Advantage**: Locally weighted regression works well when there exists a non-linear relationship between the data and the targets, while traditional linear regression cannot be used to make prediction for a non-linear data-set.

Disadvantage: Higher computational cost than traditional linear regressions since it uses local fitting of data points and the model computes for each single data point.