Question 1:

a) policy that keep every email (y= Keep)

$$E[L(y,t)] = L(y, t=\text{Spam})\cdot Pr(t=\text{Spam}) + L(y, t=\text{NonSpam})\cdot Pr(t=\text{NonSpam})$$

$$= 1\cdot 0.1 + 0\cdot(1-0.1)$$
$$= 0.1$$

policy that remove every email (y= Remove)

$$E(L(y,t)] = L(y, t=\text{Spam})\cdot Pr(t=\text{Spam}) + L(y, t=\text{NonSpam})\cdot Pr(t=\text{NonSpam})$$

$$= 0\cdot 0.1 + 100\cdot 0.9$$
$$= 90$$

b) want to find $y=y^*$ which minimize $E[L(y,t)|x]$

$$E[L(y,t)|x] = L(y, t=\text{Spam})\cdot Pr(t=\text{Spam}|x) + L(y, t=\text{NonSpam})\cdot Pr(t=\text{NonSpam}|x)$$

Suppose y^* = keep

$$E(L(y^*,t)|x) = Pr(t=\text{Spam}|x)$$

Suppose y^* = Remove

$$E(L(y^*,t)|x) = 100\,Pr(t=\text{NonSpam}|x)$$
$$= 100(1-Pr(t=\text{Spam}|x))$$
$$= 100 - 100\,Pr(t=\text{Spam}|x)$$

we want to know when the expected loss of Remove is less than keep. given x.

i.e. $100 - 100 Pr(t=Spam|x) < Pr(t=Spam|x)$

$$100 < 101 Pr(t=Spam|x)$$

$$Pr(t=Spam|x) > \frac{100}{101}$$

Hence. when $Pr(t=Spam|x) > \frac{100}{101}$ $y^* =$ remove

otherwise $y^* =$ keep

c) want to find y^* that minimize $E[L(y,t)|x_i]$ where $i=1,2,3,4$.

$x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $x_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $x_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

for each $x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$ $x_{i1} \rightarrow$ input feature x_1, $x_{i2} \rightarrow$ input feature x_2

from part a, b we know:

$$E(L(y,t)|x_i) = L(y, t=Spam) \cdot Pr(t=Spam|x_i) + L(y, t=NonSpam) \cdot Pr(t=NonSpam|x_i)$$

also, using Bayes' Theorem.

$$Pr(t=Spam|x_i) = \frac{Pr(x_i|t=Spam)\, Pr(t=Spam)}{Pr(x_i)}$$

by $Pr(t=Spam)=0.1$ $= \dfrac{Pr(x_i|t=Spam) \times 0.1}{Pr(x_i|t=Spam) \times 0.1 + Pr(x_i|t=nonSpam) \times 0.4}$

Hence,

$$Pr(t=\text{Spam}\mid x_1) = \frac{0.4\times0.1}{0.4\times0.1+0.998\times0.9} = \frac{200}{4691}$$

$$Pr(t=\text{Spam}\mid x_2) = \frac{0.3\times0.1}{0.3\times0.1+0.001\times0.9} = \frac{100}{103}$$

$$Pr(t=\text{Spam}\mid x_3) = \frac{0.2\times0.1}{0.2\times0.1+0.001\times0.9} = \frac{200}{209}$$

$$Pr(t=\text{Spam}\mid x_4) = \frac{0.1\times0.1}{0.1\times0.1} = 1$$

since $Pr(t=\text{Spam}\mid x_1)$, $Pr(t=\text{Spam}\mid x_2)$, $Pr(t=\text{Spam}\mid x_3) \le \frac{100}{101}$

Hence $y^* = \text{Keep}$ for x_1, x_2, x_3

and $y^* = \text{Remove}$ for x_4

d)

$$P(x_1) = 0.4\times0.1+0.998\times0.9 = \frac{4691}{5000}$$
$$P(x_2) = 0.3\times0.1+0.001\times0.9 = \frac{309}{10000}$$
$$P(x_3) = 0.2\times0.1+0.001\times0.9 = \frac{209}{10000}$$
$$P(x_4) = 0.1\times0.1+0\times0.9 = \frac{1}{100}$$

$$E(L(y^*,t)) = E[E(L(y^*,t)\mid x)]$$
$$= \sum_x E(L(y^*,t)\mid x)\cdot P_X(x)$$
$$= \frac{200}{4691}\times\frac{4691}{5000}+\frac{100}{103}\times\frac{309}{10000}+\frac{200}{209}\times\frac{209}{10000}+0\times\frac{1}{100}$$
$$= 0.09$$

Question 2:

a) For a data set that is linearly separable. if two points lie in a half space, every point on the line segment connecting them should also lie in the same half space. So, assuming that this data set is linearly separable, every point on the line segment -1 and 3, which both in $t=1$ Hence $\forall x \in [-1,3]$ the corresponded t should in class $t=1$ However when $x=0$ $t=1$, contradict to the previous therefore, the data set provided is not linearly separable

b) After applying x^2, in the feature map, we have:

x	x^2	t
-1	1	1
1	1	0
3	9	1

let weight infront of feature x be w_1

let weight infront of feature x_2 be w_2

Hence $z = w_1 x + w_2 x^2$, from lecture we know $y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$

plug in $x=-1$ $x^2=1$, and $t=1$

$-w_1 + w_2 \geq 0$

plug in $x=1$, $x^2=1$ and $t=0$

$w_1 + w_2 < 0$

plug in $x=3$ $x^2=9$ and $t=1$, we know:

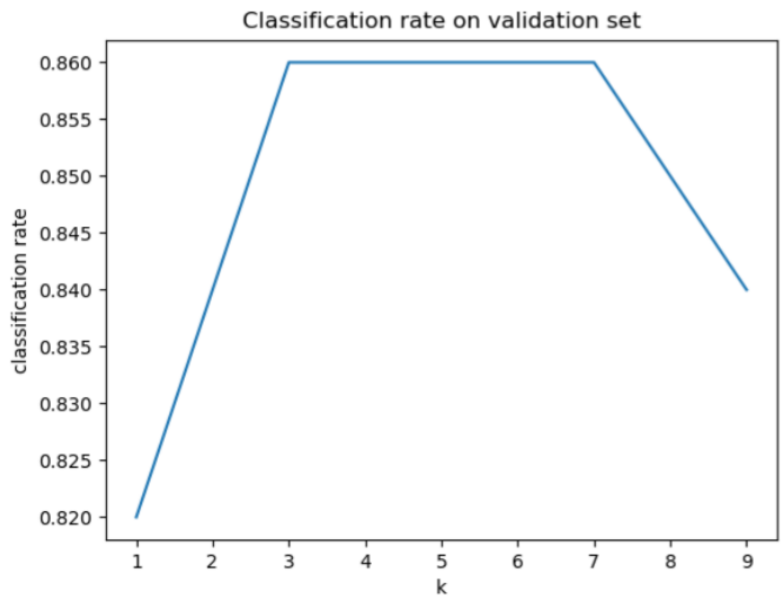
$3w_1 + 9w_2 \geq 0$

combining this 3, we know $\begin{cases} w_1 < 0 \\ w_2 \geq 0 \end{cases}$

Hence, choose $w_1 = -3$ $w_2 = 2$ correctly classifies all examples

Question 3:

3.1 a)



b) from the graph, we can see that initally, as k increases classification rate also increase. when k gets to 3 it reaches the max rate. 0.86. it's likely that KNN is less overfit when k increase and keep at 0.86 from k=3 to 7

start from k=7, classification rate decreases. as when k get too large. the model starts to underfit. and get worse as k get larger

Therefore we can see that the optimal range for k is from 3 to 7 there fore, we choose $k^*=5$ as it is at the middle of optimal which won't be too close to underfit nor overfit

The validation classification rate for k^*, k^*-2, k^*+2 are the same, 0.86

the test classification rate for k^* and k^*+2 is 0.94

test classification rate for k^*-2 is 0.92

The test performance of these k is generally better than validation performance
and the test performance is highly corresponded to validation performance
k^* and k^*+2 have the same rate and k^*-2 is only a little bit less than those two.

3.2

b) train on mnist_train.
The value returned by run_check_grad is $3.42e^{-8}$
its really small so our implementation should be correct

the hyperparameters I found worked best is

"learning_rate": 0.06, "num_iterations": 80

```
diff = 3.4218230576089815e-08
the train classification error is 0.03125
the valid classification error is 0.09999999999999998
the test classification error is 0.09999999999999998
the train CE is 0.20287437419432514
the valid CE is 0.3128086775228016
the test CE is 0.28134677646493883
```

train on mnist_train_small

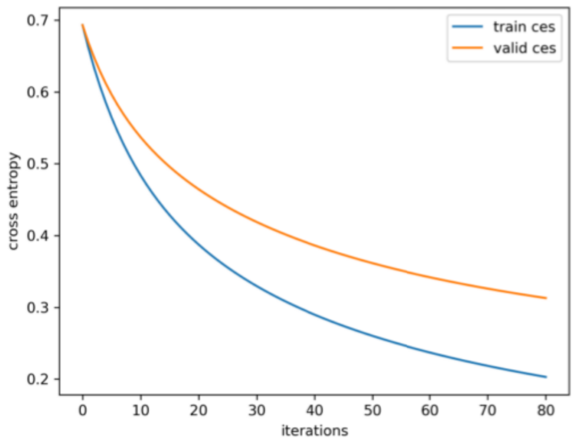
The value returned by run_check_grad is $3.32e^{-8}$
its really small so our implementation should be correct

the hyperparameters I found worked best is

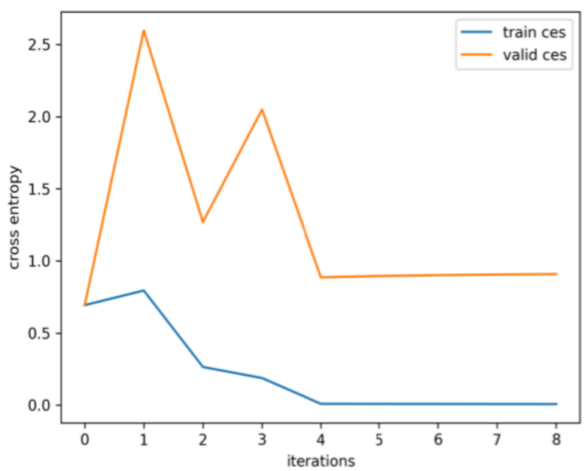
"learning_rate": 1 "num_iterations": 8

```
diff = 3.3193286201672526e-08
the train classification error is 0.0
the valid classification error is 0.3400000000000001
the test classification error is 0.21999999999999997
the train CE is 0.007723205012628585
the valid CE is 0.9084916056159195
the test CE is 0.7807530284634275
```

c) cross entropy graph for mnist_train:



cross entropy graph for mnist_train_small:



from two graphs. we can see that cross entropy for train follows a decreasing trend. while the cross entropy for validation set when train is small, fluctuate a lot.

for choosing the best parameters.

I also plot a graph for accuracies, first set the learning rate and to look at the plot on accuracy verses iteration to choose the best iteration.

for the train data since we have enough data, we can follow the convention set learning rate between 0.1 and 0.01 which I choose 0.06

for the small train data since we only have 10 training examples Hence the number of iteration must be small to avoid overfitting (I choose 8)

Hence. learning rate need to be large to make a difference. I chose 1

Question 4:

(a) let $W = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

let $f(w) = w^* = \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - w^T x^{(i)})^2 + \frac{\lambda}{2}\|w\|^2$

since we want to find w that minimize f,
we can set $\frac{\partial f}{\partial w_j} = 0$, first expand f to get $w_{j'}$ in the form

$$f(w) = \frac{1}{2}\sum_{i=1}^{N} a^{(i)}\Big(y^{(i)} - \sum_{j=1}^{m} w_j x^{(ij)}\Big)^2 + \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

$$\frac{f(w)}{dj'} = \sum_{i=1}^{N} a^{(i)}\Big(y^{(i)} - \sum_{j=1}^{m} w_j x^{(ij)}\Big)\cdot(-x^{ij'}) + w_{j'}\lambda$$

$$= \sum_{i=1}^{N}\Big(a^{(i)}y^{(i)}(-x^{ij'}) + a^{(i)}x^{ij'}\sum_{j=1}^{m} w_j x^{(ij)}\Big) + w_{j'}\lambda$$

$$= -\sum_{i=1}^{N}\big(a^{(i)}y^{(i)}x^{ij'}\big) + \sum_{i=1}^{N}\Big(a^{i}x^{(ij')}\sum_{j=1}^{m} w_j x^{(ij)}\Big) \underline{+ w_{j'}\lambda}$$

$$= -\sum_{i=1}^{N}\big(a^{(i)}y^{(i)}x^{ij'}\big) + \sum_{j=1}^{m}\sum_{i=1}^{N} a^{i}x^{ij'}w_j x^{(ij)} + w_{j'}\lambda$$

$$= -\sum_{i=1}^{N}\big(a^{i}y^{(i)}x^{ij'}\big) + \sum_{j=1}^{m}\sum_{i=1}^{N} a^{i}x^{ij'}w_j x^{ij} + \lambda I_{j=j'} w_j$$

$$= -\sum_{i=1}^{N}\big(a^{i}y^{(i)}x^{ij'}\big) + \sum_{j=1}^{m}\sum_{i=1}^{N}\big(a^{i}x^{ij'}\cdot x^{ij} + \lambda I_{j=j'}\big) w_j$$

set $\frac{f(w)}{dj'} = 0$

$$\sum_{j=1}^{m}\sum_{i=1}^{N}\big(a^{i}x^{ij'}\cdot x^{ij} + \lambda I_{j=j'}\big) w_j = \sum_{i=1}^{N}\big(a^{i}y^{(i)}x^{ij'}\big)$$

set all the $\frac{f(w)}{dj'} = 0$, we have.

$$\sum_{j'=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{N}(a^i x^{ij'} \cdot x^{ij} + \lambda I_{j=j'}) w_j = \sum_{j'=1}^{m}\sum_{i=1}^{N}(a^i y^{(i)} x^{ij'})$$

$$\sum_{j'=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{N} a^i x^{ij'} \cdot x^{ij} w_j + \lambda \sum_{j'=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{N} I_{j=j'}$$

$$= \sum_{j'=1}^{m}\sum_{i=1}^{N}(a^i y^{(i)} x^{ij'})$$

we can see that

$$\sum_{j'=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{N} a^i x^{ij'} \cdot x^{ij} w_j = X^T A X W$$

$$\lambda \sum_{j'=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{N} I_{j=j'} = \lambda I w^*$$

$$\sum_{j'=1}^{m}\sum_{i=1}^{N}(a^i y^{(i)} x^{ij'}) = X^T A y$$

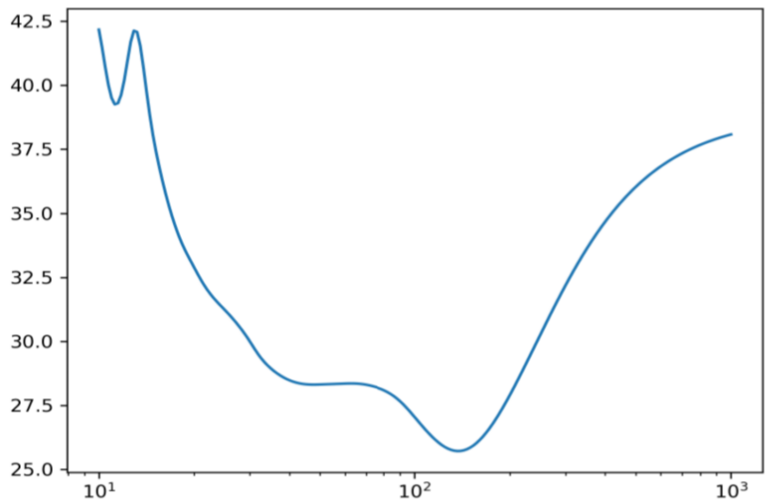
express it in Matrix form:

$$(X^T A X + \lambda I) w^* = X^T A y$$

$$w^* = (X^T A X + \lambda I)^{-1} X^T A y$$

as desired

c)



d) as $\tau \to \infty$ the algorithm behave the same as normal linear regression since each w_i is the same $= \frac{1}{n}$ (n is the num of training examples)

since each w_i is the same and relatively small, we don't emphasis anymore nor penalize any point, hence it should be have like a normal linear regression

as $\tau \to 0$ $w_i \to \infty$ which means for each x_i loss function emphasis alot on $(y^{(i)} - w^T x^{(i)})$ term which similar to set $k=1$ in kNN classifier.. Which of course won't perform too well in validation set.

We can see that our predition is correct from 4c graph. which as a 'V' shape. because when we find a balance between locally weighted and also covering enough neighbour will minimize the cost which get assured from the local minimum is in the middle