QUESTION 1

(a) ① $E[L(Y=keep,t)]$

$$= \sum_{t \in T} L(Y=keep, t) \cdot Pr(Y=keep, t)$$

they're ind

$$= 0 \cdot 0.9 + 1 \cdot 0.1$$

$$= 0.1$$

② $E[L(Y=remove,t)]$

$$= \sum_{t \in T} L(Y=remove,t) \cdot Pr(Y=remove, t)$$

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

$$= 90$$

(b) $\vec{x}$: Feature Vector

Since we know $Pr(t=spam|X)$, we can calculate $Pr(t=nonspam|X) = 1 - Pr(t=spam|X)$.

* Bayes optimal Decision rule:

Find a y^* from {keep, remove}, that minimizes $E[L(Y=y^*,t)]$, as follows:

week 2 P.g. 52

$$\begin{cases} y^* = keep: E[L(Y=keep,t)] = 1 \cdot Pr(t=spam|X) + 0 \cdot [1 - Pr(t=spam|X)] \\ y^* = remove: E[L(Y=remove,t)] = 0 \cdot Pr(t=spam|X) + 100 \cdot [1 - Pr(t=spam|X)] \end{cases}$$

Plug in $Pr(t=spam|X)$, compare the two values, choose the one that outputs lower expectation.

(c) First, let's compute:

Note $Pr(t=spam) = 0.1$

$Pr(t=spam, X_1, X_2)$	$X_2=0$	$X_2=1$
$X_1=0$	$\frac{200}{4691}$	$\frac{100}{103}$
$X_1=1$	$\frac{200}{209}$	1

$Pr(X_1=0, X_2=0|t=spam) = 0.4$

$Pr(X_1=0, X_2=0|t=nonspam) = 0.998$

$$\Rightarrow Pr(t=spam|X_1=0, X_2=0) = \frac{Pr(X_1=0, X_2=0|t=spam) \cdot Pr(t=spam)}{\sum_{t \in T} Pr(X_1=0, X_2=0|t) \cdot Pr(t)} = \frac{\frac{200}{4691}}{}$$

Similarly,

$Pr(t=spam|X_1=0, X_2=1) = \frac{100}{103}$

$Pr(t=spam|X_1=1, X_2=0) = \frac{200}{209}$

$Pr(t=spam|X_1, X_2=1) = 1$

As in Table 1, so we know $Pr(t=spam|X)$, we also can get $Pr(t=nonspam|X)$. And then we can apply optimal rule from (b).

So, we can find expected Loss of Keep and Remove as follows:

$E[L(Y=keep, X_1, X_2)]$ / $E[L(Y=remove, X_1, X_2)]$	$X_2=0$	$X_2=1$
$X_1=0$	0.0426 95.7365	0.9709 2.9126
$X_1=1$	0.9569 4.3062	1 0

Hence, we have the following actions based on X_1 and X_2.

If
- $X_1 = 0, X_2 = 0 \longrightarrow$ keep (0.0426)
- $X_1 = 0, X_2 = 1 \longrightarrow$ keep (0.9709)
- $X_1 = 1, X_2 = 0 \longrightarrow$ keep (0.9569)
- $X_1 = 1, X_2 = 1 \longrightarrow$ remove (0)

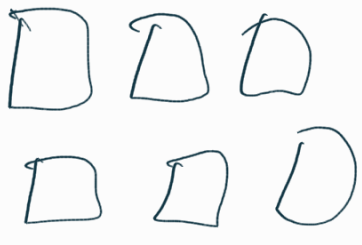
**(d) Based on (c), we know four best actions for four possible combinations of X_1 and X_2.

So, $E[L(X_k, t)] = 0.0426 + 0.9709 + 0.9569 + 0$

$= 1.9704$

$$\frac{\partial L}{\partial w_j} = \frac{\partial \xi}{\partial z} \cdot \frac{\partial z}{\partial w_j}$$

$$\frac{\partial \xi}{\partial z} = \begin{cases} \text{*11} \\ 20,11 \end{cases}$$



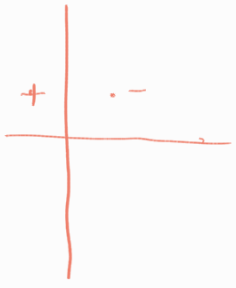
*Question 2

**(a) proof:

- If two points lie in a half-space, the line segment connecting them also lie in the same halfspace
- suppose some feasible weights exist ....
- but $-1 \to 3$ passes 1, 1 exisits in another half-space → Contradiction.

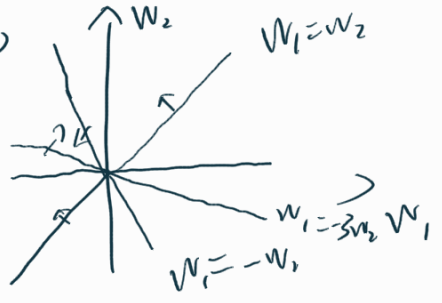
**(b) The feature maps:

x	$\psi_1(x)$	$\psi_2(x)$	t
-1	-1	1	1
1	1	1	0
3	3	9	1



$$\Rightarrow \begin{cases} -1w_1 + w_2 \geq 0 \\ w_1 + w_2 < 0 \\ 3w_1 + 9w_2 \geq 0 \end{cases}$$ (Hard threshold)

→



AND a pair $(w_1, w_2) = (-1, \frac{1}{2})$ will classify all the examples correctly.

$-(-1) + \frac{1}{2} = \frac{3}{2} \geq 0$

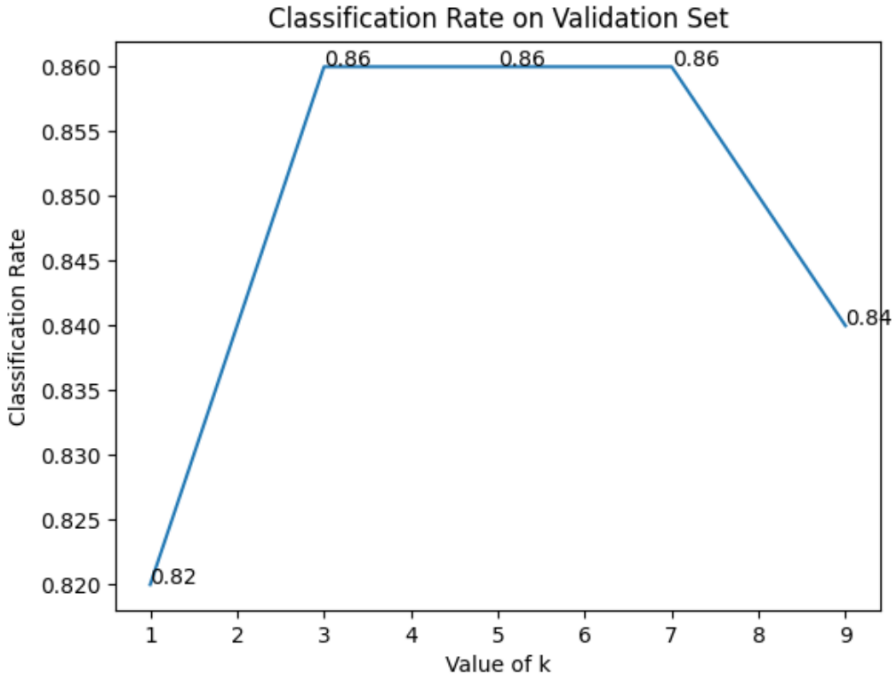
$-1 + \frac{1}{2} = -\frac{1}{2} < 0$

$3 \times (-1) + 9 \times \frac{1}{2} = \frac{3}{2} \geq 0$

* Question 3

** part(1)

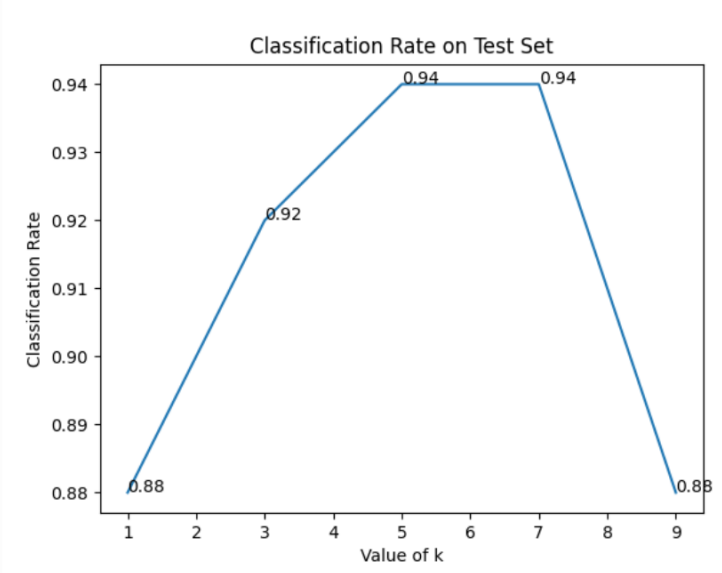
*** a)



***, b)

OVERALL, the classifier performs good on validation set the $K \in \{1, 3, 5, 7, 9\}$. As shown in the plot, the classifier predicts correctly for around 82% data points.

However, different Ks do perform differently. I would like to choose $K^* = 5$, the classification rate for K^* is 0.86. AND the classification rate for $K^* + 2$ (7) is 0.86, $K^* - 2$ (3) is 0.86.



The test performance of these values of K are similar to that from validation:

① K=5, 7 are still the best choices.

② K=3 is not the best choice anymore, but still a better choice than K=1 and K=9.

Such findings are expected, because we know when $K \to 0$, the model underfits, and when $K \to \infty$, the model overfits, both won't results in best rate, a middle value of K is therefore prefetted.

**Part(2)

***(a) Implemented and uploaded to markus.

***(b) Run_check_grad is very small as follows:

```
[[-0.10461503 -0.10461503]
 [-0.2001114  -0.20011139]
 [-0.20279696 -0.20279696]
 [-0.19851759 -0.19851759]
 [-0.13922366 -0.13922366]
 [ 0.05024501  0.05024502]
 [ 0.27444364  0.27444364]
 [ 0.12492774  0.12492773]
 [ 0.02181367  0.02181367]
 [-0.05183629 -0.05183628]
 [ 0.01158533  0.01158533]]
diff = 2.6438378464677305e-08
```

I'd assume 2.64×10^{-8} as "small enough".

THE BEST HYPERPARAMETERS for two datasets.

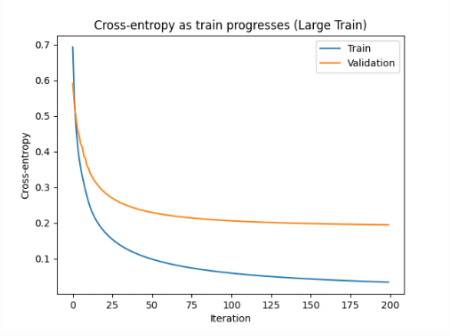
- load_train( ):

  learning_rate: 0.3
  num_iterations: 200
  CE on train: 0.1789
  Classification rate on train (error): 0.9875 (0.0125)
  CE on validation: 0.2928
  Classification rate on validation (error): 0.9 (0.1)

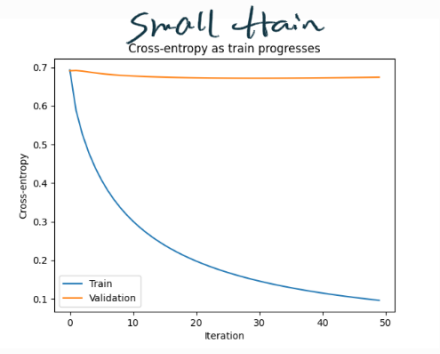
- load_train_small( ):

  learning_rate: 0.05
  num_iterations: 50
  CE on train: 0.0941
  Classification rate on train (error): 1
  CE on validation: 0.6743
  Classification rate on validation (error): 0.64 (0.36)

***(c) Look at the following two images.



Small train



*I run my code several times with diff hyperparameters, the results do change a little bit, and I'll select based on the following rules:

① better on validation, to avoid overfit.
② the curve that changes faster -> to avoid long time of running.

*QUESTION 4

**(a) I'll apply "Direct Solution":

I'll use $\mathcal{E}$ to denote cost function.

So, we can compute:

"$\mathcal{E} = \frac{1}{2}\sum_{i=1}^{N} a^i (y^i - \sum_{j=1}^{N} w^j \cdot x_j^i)^2 + \frac{\lambda}{2}\sum_{j=1}^{N} w_j^2$"

$$\frac{\partial \mathcal{E}}{\partial w_k} = \frac{1}{2}\sum_{i=1}^{N} a^i (y^i - \sum_{j=1}^{N} w^j \cdot x_j^i)\cdot(-x_k^i) + \frac{\lambda}{2}\cdot 2\cdot w^k$$

$$= \sum_{i=1}^{N} a^i \cdot x_k^i \left(\sum_{j=1}^{N} w^j \cdot x_j^i - y^i\right) + \lambda \cdot w^k$$

we want $\frac{\partial \mathcal{E}}{\partial w_k} = 0$, so:

$$\lambda \cdot w^k = \sum_{i=1}^{N} a^i \cdot x_k^i (y^i - \sum_{j=1}^{N} w^j \cdot x_j^i)$$

$$\lambda \cdot w^k = A \cdot X_k \cdot (Y - W^* \cdot X^T)$$

#Assume $X = [\ \ ]$, $W = \begin{matrix}\square\\ \square\end{matrix}$, $Y = \begin{matrix}\square\\ \square\end{matrix}$, X_k = Kth column

$\Leftrightarrow \lambda \cdot I \cdot w^* = A \cdot X^T (Y - w^* \cdot X^T)$ #X^T because we want A apply to X_j in one sample.

$\Leftrightarrow \lambda I \cdot w^* = AX^T Y - AX^T \cdot X \cdot w^*$

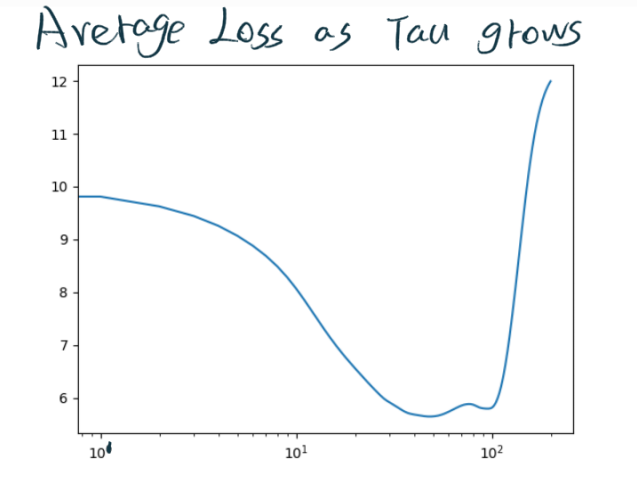
$\Leftrightarrow \lambda I w^* + AX^T \cdot X \cdot w^* = AX^T Y$

$\Leftrightarrow \lambda I \cdot w^* + X^T A \cdot X \cdot w^* = X^T \cdot AY$ #Since A is scalar matrix and $A_{ii} = a^i > 0$, so $AX^T = X^T A$

$\Leftrightarrow W^* = (\lambda I + X^T A X)^{-1} \cdot X^T A Y$

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

**(b)



**(c) When $\tau \to \infty$, the algorithm will perform bad, resulting in big loss in average. When $\tau \to 0$, it will predict badly and have a big loss as well.

It happened actually. For example, when $\tau = 10$ and $\tau = 1000$, the averge is bigger, and there's a trend $\tau = 10$ decrease a middle point increase (zero around 50)