CS299:Machine Learning: Assignment2

Due on June 6, 2018

Andrew Ng

Bryan Zhang

Contents

Problem 1 **2**
1(a) . . . 2
1(d) . . . 3
1(b) . . . 3
1(c) . . . 5
i.Using a different constant learning rate . . . 5
ii.Decreasing the learning rate over time . . . 5
iii. Add a regularization term to the loss function. . . . 6
iv. linearly scale of input features . . . 7
v. Add zero-mean Gaussian noise to the training data or labels . . . 7

Problem 2 **7**
a . . . 7
b . . . 7
c . . . 8

Problem 3 **8**

Problem 4 **8**
a . . . 8
b . . . 9
c . . . 9
d . . . 9
e . . . 9
f . . . 10
g . . . 10
h . . . 11

Problem 5 **11**

Problem 6 **11**
b . . . 11
c . . . 12
d . . . 12
e . . . 12

Problem 1

I answer the subquestions in a more convenient order.

1(a)

The Most noticeable difference is that logistic regression converges quickly on dataset A but seems never converges for dataset B.
For dataset A, it only took 30395 iteration to fulfill the 1e-15 error margin. But for dataset B, the algorithm are still running even for 16490000 iterations.

1(d)

First of all, Let's formulate the gradient ascent update rules used by logistic regression algorithm in the $lr_debug.py$.

$$\theta_j = \theta_j + \alpha(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}y^{(i)}\frac{1}{1+e^{y^{(i)}x^{(i)}\cdot\theta^T}}) \tag{1}$$

Then we have to know what this update rule is maximizing. My guess is the log likelihood of being predicted correctly. After we compute the gradient, this can be proved. Assuming that the m training examples were generated independently. we can then write down the likelihood of the parameter as

$$\begin{aligned} L(\theta) &= p(Y|X;\theta) \\ &= \prod_{i=1}^{m}(1-p^{(i)}) \\ p^{(i)} &= \frac{1}{1+e^{y^{(i)}x^{(i)}\cdot\theta^T}} \\ \gamma^{(i)} &= y^{(i)}x^{(i)}\cdot\theta^T \end{aligned} \tag{2}$$

You can see intuitively that $\gamma^{(i)}$ will be negative if the algorithm makes a mistake, which will make $p^{(i)}$ bigger thus $L(\theta)$ smaller. As before, it will be easier to maximize the log likelihood:

$$\begin{aligned} l(\theta) &= log(\sqrt[m]{L(\theta)}) \\ &= \frac{1}{m}\sum_{i=1}^{m} log(1-p^{(i)}) \end{aligned} \tag{3}$$

Now, to maximize our log likelihood, we use the gradient ascent:

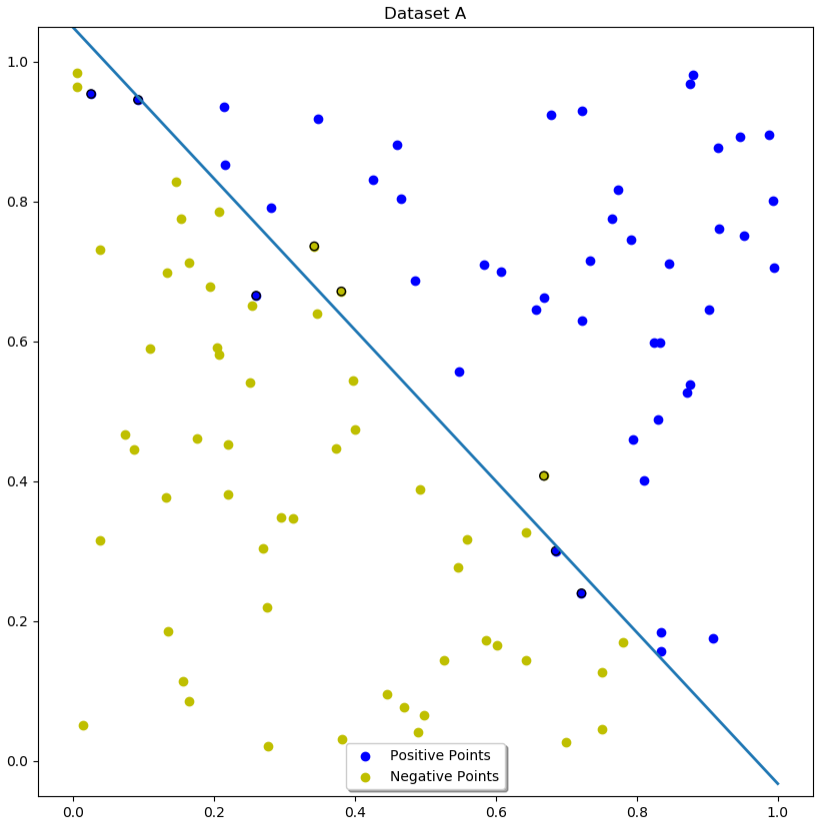
$$\begin{aligned} \nabla_\theta l(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta log(1-p^{(i)}) \\ &= \frac{1}{m}\sum_{i=1}^{m} -\frac{1}{1-p^{(i)}}\nabla_\theta p^{(i)} \\ &= \frac{1}{m}\sum_{i=1}^{m} -\frac{1+e^{y^{(i)}x^{(i)}\cdot\theta^T}}{e^{y^{(i)}x^{(i)}\cdot\theta^T}}(-\frac{1}{(1+e^{y^{(i)}x^{(i)}\cdot\theta^T})^2})e^{y^{(i)}x^{(i)}\cdot\theta^T}y^{(i)}x^{(i)} \\ &= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{1+e^{y^{(i)}x^{(i)}\cdot\theta^T}}y^{(i)}x^{(i)} \\ &= \frac{1}{m}\sum_{i=1}^{m} p^{(i)}y^{(i)}x^{(i)} \end{aligned} \tag{4}$$

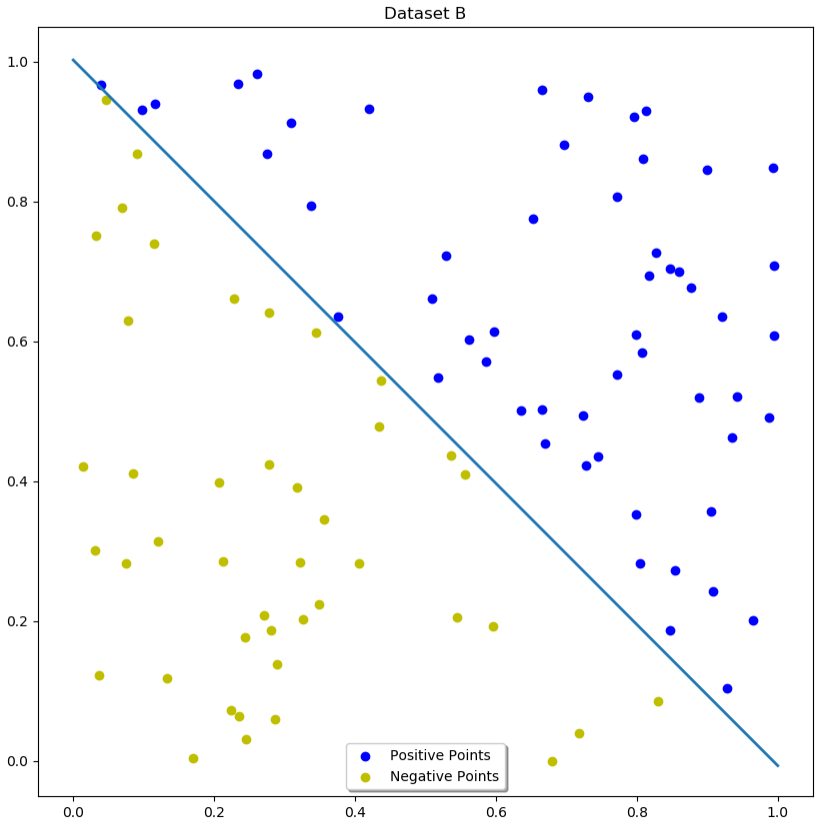
q.e.d. The logistic regression is try to maximizing the probability of the margins being positive. Geometrically, minimizing the logistic regression loss means the maximizing the margins toward a positive direction. As for Support Vector Machine, I believe it will help. Since this time, the svm is trying to maximize the biggest margins rather than the sum of all margins, which makes it more robust for discontinuity.

1(b)

My hypothesis: data set B can have θs with difference more than $3e^{-15}$ but which make the maximum margins. in other words, there is a flat discontinuity for dataset B, while the big gradient is just overshoot theta on the edges of the flat discontinuity, which is bigger than the required convergence

condition. For dataset A, this is not the case. Since for dataset A, mistakes has already been made. The separating line is just making the margins bigger. Also along separating line, dataset A has more data points to confine the line than dataset B. You can see the visualization for the dataset A's training process at here and for dataset B at here.





1(c)

i.Using a different constant learning rate

This won't help. Because if we slow the learning rate then to achieve the none-mistake separating line, the algorithms will take more iterations than the current learning rate. If we accelerate the learning rate, after achieving the none-mistakes separating line, the algorithm will not be stable since the current learning rate has already overshoot the line between edges of the falt discontinuity. The experiment results

learning rate	iterations
1e-1	$>$ 86000
1e-2	$>$86000
1	$>$86000
100	$>$86000
1000	$>$86000(Overflow Warning)

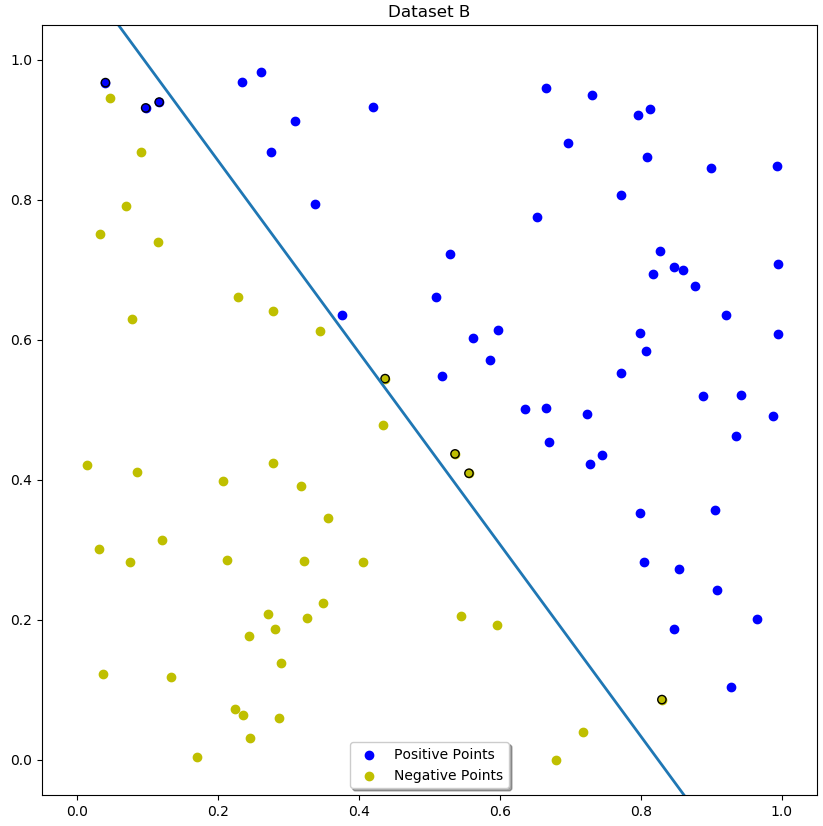
ii.Decreasing the learning rate over time

This will help. Because it will eliminate the iterations that overshoots and don't contribute the final accuracy. However, I found the decreasing factor matters two. With higher decreasing order, the training converges

before it reaches the good separating line. While if the factor has too slow order, then the training takes too many iterations. Below is my experiment result:

decreasing factor	learning rate	iterations	number of mistakes
$\frac{1}{t^2}$	10	12	43
	100	12	9
$\frac{1}{t}$	10	18	14
	100	19	7
$\frac{1}{\sqrt{t}}$	10	29	21
	100	29	7
$\frac{1}{log(t)}$	10	>86000(divide by zero warning)	Unkown
	100	>86000(divide by zero warning)	Unkown

Below is the best performance under the learning rate = 100 and decreasing factor = $\frac{1}{\sqrt{t}}$.



iii. Add a regularization term to the loss function.

I will add a term $||\theta||^2$ to the loss and thus a term of 2θ to the gradient. This doesn't help.Because the regularization is to penalize big thetas and give reward to more sparse thetas, which can't stabilize the overshoot in training dataset B.

iv. linearly scale of input features

This won't help. Because linear scaling of the input features will have no effect of the line slop and intercept, both of which loose the scale when they are computed by dividing of c. Here is my experiment result:

scaling factor	iterations
1	> 86000
0.5	>86000
0.1	>86000
1e-2	>86000
1e-4	>86000

v. Add zero-mean Gaussian noise to the training data or labels

This won't help either. Since once they are summed together to be theta the noise cancels out.

Problem 2

a

In logistic regression, we want to maximize the log likelihood:

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} h(x^{(i)}) + (1 - y^{(i)}) log(1 - h(x^{(i)})) \tag{5}$$

This time, instead of updating by gradient, we simply put its gradient to zero. And we have confidence to say that in this way, the parameter is also the learned parameter.

$$\nabla_\theta l(\theta) = x^T \cdot (y - h_\theta(x)) \ = 0 \tag{6}$$

If we look at the bias term column in x, we can have

$$\sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) = 0 \tag{7}$$

Relocating sum of probabilities and sum of the labels to opposite sides of the equation, we can prove the property.

b

If the model achieves perfect accuracy, which means it will predict all positive example with 1 and negative with 0.Then both sides of the property equation will be zero if the interval doesn't include 1. When it do includes 1, the proper equation still holds to be 1. Thus The model who achieves perfect accuracy is perfectly calibrated. But a perfect calibrated binary classifier can not achieve perfect accuracy. Because we can easily find a counter-example. Let half of example be negative and the prediction probabilities for all examples are $\frac{1}{2}$. Thus for intervals that contains $\frac{1}{2}$, the property holds to be one half on both sides of the equations. For intervals that don't contains $\frac{1}{2}$, the both sides of the equations are zero, which guarantees the property. Thus the model must be perfectly calibrated but can't achieve perfect accuracy.

Problem 2 continued on next page...

c

the objective function now is

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{1}{2}\theta^2 \tag{8}$$

Now we compute its gradient to θ.

$$\nabla_\theta J(\theta) = x^T \cdot (h_\theta(x^{(i)}) - y^{(i)}) + \theta \tag{9}$$

Then we set the gradient to zero to find the $\arg\min_\theta J(\theta)B$.

$$x^T \cdot (h_\theta(x^{(i)}) - y^{(i)}) = -\theta \tag{10}$$

Still we investigate the intercept term.

$$\sum_{i=1}^{m} P(y^{(i)=1}|x^{(i)};\theta) = \sum_{i=1}^{m} \mathbf{1}\{y^{(i)} = 1\} - \theta_1 \tag{11}$$

Thus, the L_2 regularization shift the probabilities toward zero by the extent of θ_1. This means Bigger θ_1 will predict lower probabilities.

Problem 3

If we assume that

$$||\theta_{MAP}||_2 > ||\theta_{ML}||_2 \tag{12}$$

Thus, θ_{MAP} is not equal to θ_{ML}, the $\arg\max_\theta \prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)$, which gives $p(y^{(i)}|x^{(i)},\theta_{MAP}) < p(y^{(i)}|x^{(i)},\theta_{ML})$. Also if prior $\theta \sim \mathcal{N}(0,\tau^2 I)$, $p(\theta_{MAP}) < P(\theta_{ML})$, for our assumption guarantees that θ_{MAP} are more away from the mean point zero. Using above two inequality, we can have this inequality,

$$p(\theta_{MAP})\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta_{MAP}) < p(\theta_{ML})\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta_{ML}) \tag{13}$$

This inequality contradicts our condition that,

$$\theta_{MAP}| = \arg\max_\theta p(\theta)\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta) \tag{14}$$

q.e.d

Problem 4

a

Yes, it is a kernel.

$$\begin{aligned} K_{ij} &= K_{1_{ij}} + K_{2_{ij}} \\ &= K_1(x^{(i)}, x^{(j)}) + K_2(x^{(i)}, x^{(j)}) \\ &= K(x^{(i)}, x^{(j)}) \end{aligned} \tag{15}$$

Problem 4 continued on next page...

Since K_1 and K_2 is symmetric, then $K_{1_{ij}} = K_{1_{ji}}$ and $K_{2_{ij}} = K_{2_{ji}}$. This gives,

$$\begin{aligned} K_{ij} &= K_{1_{ij}} + K_{2_{ij}} \\ &= K_{1_{ji}} + K_{2_{ji}} \\ &= K_{ji} \end{aligned} \tag{16}$$

Thus K is also symmetric. Then we are only step away to prove K is a Mercer Kernel. Since K is symmetric, we only need to prove for all vectors $\vec{x}$, $x^T K x \geq 0$.

$$x^T(K_1 + K_2)x = x^T K_1 x + x^T K_2 x \geq 0 \tag{17}$$

Since K_1 and K_2 is positive semidefinite, Thus we have proved K is a mercer kernel.

b

K is not necessarily a Mercer Kernel. Similarly with (a), we can easily prove that K is symmetric. But when it comes to positive semidefinite, we have this in the end.

$$x^T(K_1 - K_2)x = x^T K_1 x - x^T K_2 x \tag{18}$$

Since both term is bigger or equal to zero, $x^T(K_1 + k_2)$ is not always positive. q.e.d.

c

K is a Mercer Kernel. We can easily prove that K is symmetric by examining the elements that are symmetric to the principal diagonal. Then we focus same inequality.

$$\begin{aligned} x^T K x &= x^T a K_1 x \\ &= a x^T K_1 x \end{aligned} \tag{19}$$

Since a is a positive real number and K_1 is semidefinite, above equation is always bigger or equal to zero. q.e.d.

d

K is not a Mercer Kernel. Since $-a x^T K_1 x \leq 0$.

e

First, let's prove that the Kernel matrix is symmetric.

$$\begin{aligned} K_{ij} &= K(x^{(i)}, x^{(j)}) \\ &= K_1(x^{(i)}, x^{(j)}) K_1(x^{(i)}, x^{(j)}) \\ &= K_{1_{ij}} K_{2_{ij}} \end{aligned} \tag{20}$$

Since K_1 and K_2 are symmetric, $K_{1_{ij}} = K_{1_{ji}}$ and $K_{2_{ij}} = K_{2_{ji}}$. So the above equation can be transformed.

$$\begin{aligned} K_{ij} &= K_{1_{ij}} K_{2_{ij}} \\ &= K_{1_{ji}} K_{2_{ji}} \\ &= K_{ji} \end{aligned} \tag{21}$$

So the kernel matrix is symmetric.
Next we will prove the kernel matrix is positive semidefinite or the Schur Product Theorem.

Theorem 1. ***Schur Product Theorem***
If $A, B \in S\mathbb{R}^{n\times n}$, then their Hadamard product $A \otimes B$ is also positive semidefinite. Moreover, if both A and B are positive definite, then $A \otimes B$ is also positive definite.

Proof. [1]Since $A, B \succeq 0$, we can eigendecomposition A, B.

$$A = \sum^{n} u_i u_i^T, B = \sum^{n} v_i v_i^T$$

Then, using distributive, we can get

$$A \circ B = \sum_{i,j=1}^{n} u_i u_i^T \circ v_j v_j^T$$

Using the properties of Hadamard product, $(a \circ b)(c \circ d)^T = ac^T \circ bd^T$

$$A \circ B = \sum_{i,j=1}^{n} (u_i \circ v_j)(u_i \circ v_j)^T$$

Since this implies $A \circ B$ is Gram Matrix, it must be positive semidefinite. □

For $K = K_1 \circ K_2$ and K_1, K_2 are positive semidefinite, K is positive semidefinite. Thus K is a valid Mercer Kernel.

f

Still we first prove the kernel matrix is symmetric.

$$\begin{aligned} K_{ij} &= K(x^{(i)}, x^{(j)}) \\ &= f(x^{(i)}) f(x^{(j)}) \\ &= f(x^{(j)}) f(x^{(i)}) \\ &= K_{ji} \end{aligned} \tag{22}$$

But we will meet an obstacle when we want to prove its positive semi-definiteness. Because we don't know the exact mapping relation of function f. Thus K might not be semi-definite. A good counter example will be f(x) = -2 and the arbitrary vector to perform $a^T K a$ is one vector.Then the product must be negative.

g

Let's go to the symmetric proof!

$$\begin{aligned} K_{ij} &= K(x^{(i)}, x^{(j)}) \\ &= K_3(\phi(x^{(i)}), \phi(x^{(j)})) \\ &= K_3(\phi(x^{(j)}), \phi(x^{(i)})) \\ &= K_{ji} \end{aligned} \tag{23}$$

For its positive semi-definiteness, we can always construct a new finite set $\{\phi_i, i = 1, 2...m\}$ with respect to the original finite set $\{x^{(i),...x^{(m)}}\}$ where $\phi_i = \phi(x^{(i)})$. Because of this and that any finite set K_3 is positive semidefinite, K is a Mercer Kernel.

[1]I refer to this lecture note of UCSD and Schur product theorem item on wikipedia. I refer to this link for Hadamard Product property.

h

First prove the kernel's symmetry.

$$\begin{aligned} K_{ij} &= K(x^{(i)}, x^{(j)}) \\ &= p(K_1(x^{(i)}, x^{(j)})) \quad = p(K_{1_{ij}}) \\ &= P(K_{1_{ji}}) \\ &= K_{ji} \end{aligned} \tag{24}$$

Proving the kernel is positive semidefinite, we will use the conclusions we already proved, **e**, **c** and **a**. Since

$$p(K_1) = \sum_{i=0}^{m} a_i K_1^i$$

in which, all $a_i > 0$, thus

$$p(K_1) \succeq 0$$

q.e.d.

Problem 5

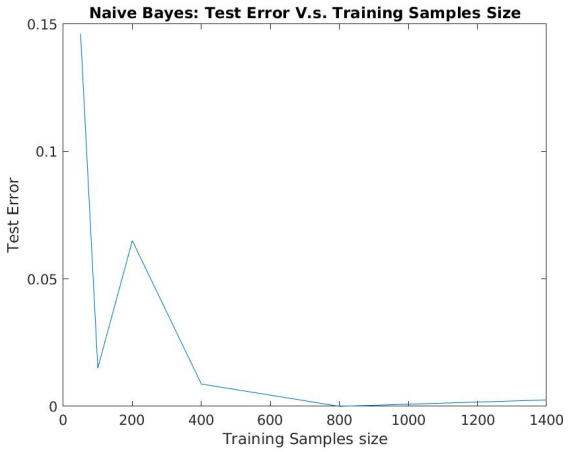
$$\begin{aligned} \theta^{(0)} &= \vec{0} \\ \theta^{(i)} &= \sum_{m=1}^{i} \alpha_m y^{(m)} \phi(x^{(m)}) \\ \alpha_m &= \alpha \mathbf{1} sign(\theta^{(m-1)} \phi(x^{(m)})) \\ \hat{y}^{(i)} &= sign(\theta^{(i-1)T} \phi(x^{(i)}) \\ &= sign(\sum_{m=1}^{i-1} \alpha_m y^{(m)} \phi(x^{(m)}) \phi(x^{(i)})) \\ &= sign(\sum_{m=1}^{i-1} \alpha_m y^{(m)} \phi(x^{(m)}) \phi(x^{(i)})) \\ &= sign(\sum_{m=1}^{i-1} \alpha_m y^{(m)} K(x^{(m)}, x^{(i)})) \\ K(x, z) &= \phi(x)\phi(z) \\ \theta^{(i+1)} &:= \theta^{(i)} + \alpha \mathbf{1}\{\hat{y}^{(i)} y^{(i+1)} > 0\} y^{(m)} \phi(x^{(m)}) \\ &= \theta^{(i)} + \alpha \mathbf{1}\{sign(\sum_{m=1}^{i-1} \alpha_m y^{(m)} K(x^{(m)}, x^{(i)}) y^{(i+1)} > 0\} y^{(m)} \phi(x^{(m)}) \end{aligned} \tag{25}$$

Problem 6

b

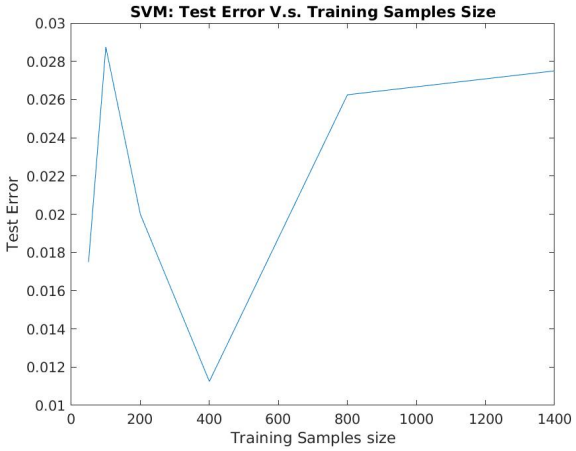
The top 5 indicative tokens are " news articl write re anybodi ".

c



When the training process hits 800, the test error are smallest, zero.

d



When the training process hits 400, the test error are smallest, zero.

e

Naive Bayes' training process is to count out the probability with a greater amount of sampling. So it is not easy to be over-fitted when tested. But SVM is to draw a separating planes. The chances to overfit the training dataset is very high.