EECS545(Section001): Homework #3

Due on FEB.23, 2022 at 11:59pm

Instructor:Honglak Lee

Tiejin Chen **tiejin@umich.edu**

Problem 1

Proof. Since log is a strictly incresing function Hence we can get:

$$w_{ML} = \underset{w}{argmax} \prod_{i=1}^{N} p(y_i|x_i;w) = \underset{w}{argmax} \sum_{i=1}^{N} ln(p(y_i|x_i;w))$$

We can actually calculate this formula further with $p(y_i|x_i;w) = \sigma(w^T x_i)^{y_i}(1-\sigma(w^T x_i))^{1-y_i}$. However, we do not need to calculate a specific expression of it to prove the result. Now let us consider MAP, we have:

$$w_{MAP} = \underset{w}{argmax} \prod_{i=1}^{N} p(W)p(y_i|x_i;w) = \underset{w}{argmax}\, Nln(p(w)) + \sum_{i=1}^{N} p(y_i|x_i;w)$$

Now, let consider $ln(p(w))$, we have:

$$p(w) = Cexp(-\frac{1}{2}w^T\Sigma^{-1}w)$$

And we know $\Sigma = \tau^2 I$, hence Σ^{-1} is a diagonal martix with all diagonal element equal to $\frac{1}{\tau^2}$. Now we take log of it, and throw away the constant to get:

$$ln(p(w)) = -\frac{1}{2}w^T\Sigma^{-1}w = -\frac{1}{2\tau^2}\left\|w\right\|_2^2$$

Then we have:

$$w_{MAP} = \underset{w}{argmax} -\lambda\left\|w\right\|_2^2 + \sum_{i=1}^{N} p(y_i|x_i;w)$$

where $\lambda = \frac{N}{2\tau^2}$ is a constant greater than 0. Then let us prove what we what.

Assumed $\left\|w_{MAP}\right\|_2 > \left\|w_{ML}\right\|_2$. Then we have:

$$\left\|w_{MAP}\right\|_2^2 > \left\|w_{ML}\right\|_2^2, \sum_{i=1}^{N} p(y_i|x_i;w_{MAP}) < \sum_{i=1}^{N} p(y_i|x_i;w_{ML})$$

Then second inequality holds because $w_{ML} = \underset{w}{argmax} \sum_{i=1}^{N} ln(p(y_i|x_i;w))$. Then we have:

$$\sum_{i=1}^{N} p(y_i|x_i;w_{ML}) - \lambda\left\|w_{ML}\right\|_2^2 > \sum_{i=1}^{N} p(y_i|x_i;w_{MAP}) - -\lambda\left\|w_{MAP}\right\|_2^2$$

which is contradict with the fact that $w_{MAP} = \underset{w}{argmax} -\lambda\left\|w\right\|_2^2 + \sum_{i=1}^{N} p(y_i|x_i;w)$. Thus we prove that $\left\|w_{MAP}\right\|_2 \leq \left\|w_{ML}\right\|_2$ by contridiction.

□

Problem 2

we will use K_1, K_2, K_3 to donate the matrix of k_1, k_2, k_3. And Y is any vector belongs to $\mathbb{R}^N$.

(a)

It is a kernel.we have $K_{ij} = k_1(x_i, x_j) + k_2(x_i, x_j) = K_{1_{ij}} + K_{2_{ij}}$. And it is symmertic.Thus, it is easy to find:

$$K = K_1 + K_2$$

Problem 2 continued on next page...

Then we have:

$$Y^T KY = Y^T(K_1 + K_2)Y = Y^T(K_1)Y + Y^T(K_2)Y \geq 0$$

Since $Y^T(K_1)Y \geq 0$ and $Y^T(K_2)Y \geq 0$. Thus,K is symmetric and positive semi-definite. Hence it is a kernel.

(b)

No, it is not. let $k_2 = 2k_1$. Then $k = -k_1$. And $K = -K_1$. Hence:

$$Y^T KY = -Y^T K_1 Y < 0$$

which shows that K is not semi-definite.

(c)

Yes, it is. we have $k = ak_1$, $K = aK_1$. Hence:

$$Y^T KY = aY^T KY \geq 0$$

since a is positive. Hence K is symmetric and positive semi-definite. Hence it is a kernel.

(d)

No, it is not. we have $k = ak_1$, $K = -aK_1$. Hence:

$$Y^T KY = -aY^T KY < 0$$

since a is positive. Hence K is not positive semi-definite.

(e)

We can have:

$$K(x, z) = K_1(x, z) \otimes K_2(x, z)$$

where $\otimes$ presents hadamard production of two matrix.For $K_1(x, z)$, we can have it Spectral decomposition:

$$K_1(x, z) = Q^T \Lambda Q = Q^T \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} Q = U^T U$$

where $U = \Lambda^{\frac{1}{2}} Q$. Then we assumed $Y = (y_1, ..., y_n)$. Then we have:

$$Y^T KY = \sum_{i=1}^{N} \sum_{j=1}^{N} K_{1_{ij}} K_{2_{ij}} y_i^T y_j$$

Using the result we get:

$$Y^T KY = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ki}^T u_{kj} K_{2_{ij}} y_i^T y_j$$

And we find that:

$$u_{ki}^T u_{kj} K_{2_{ij}} y_i^T y_j = K_{2_{ij}} (u_{ki} y_i)^T u_{kj} y_j = K_{2_{ij}} x_i^T x_j$$

Thus we can get:

$$Y^T KY = X^T K_2 X \geq 0$$

Thus, K is a symmertic and psd. Hence it is a valid kernel.

(f)

No, it is not a valid kernel. For example, if we have $n = 2, f(x) = 1,\ if\ x = x_1; f(x) = -2,\ if\ x = x_2$, Then we can get the K is:

$$\begin{bmatrix} 1 & -4 \\ -4 & 4 \end{bmatrix}$$

Then we can calculate the eigenvalue of K. To get the first eigenvalue is 6.77200187 and the second eigenvalue is -1.77200187(by numpy). For any psd matrix, we know that all the eigenvalue should be no less than 0. Hence K is not a psd matrix. Thus it is not a valid kernel.

(g)

It is a valid kernel. Assumed we have a finite set $\{x_1, ..., x_N\}$ in R^D. Then $\{\phi(x_1), .., \phi(x_N)\}$ is a finite set in R^M. Then with Mercer theorem, we know K_3 where $K_{3_{ij}} = k_3(\phi(x_i), \phi(x_j))$ is a psd and symmetric matrix. Then we have $K = K_3$. Thus, it is a valid kernel.

(h)

It is a valid kernel. Firstly, we prove $k_1(x,z)^n$ is a valid kernel. We can prove this by induction. If $n = 0$, the Kernel matrix will be a zero function, and thus psd. Then if $n = m$ it holds, then when $n = m+1$, $k_1(x,z)^{m+1} = k_1(x,z)^m k_1(x,z)$ then, it is a valid kernel by part(e). Hence $k_1(x,z)^n$ is a valid kernel. Then by part(c), every term in $p(k_1(x,z))$ is a valid kernel. Finally, using part(a) one by one, we can get $k(x,z)$ is a valid kernel.

(i)

we consider the $D = 2$ have:

$$k(x,z) = (x_1z_1 + x_2z_2 + 1)^2 = 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 = \phi(x)^T\phi(z)$$

Thus we can get :

$$\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$$

(j)

We have:

$$k(x,z) = exp(-\frac{1}{2\sigma^2}(x^Tx + z^Tz - 2x^Tz)) = exp(-\frac{1}{2\sigma^2}x^Tx - \frac{1}{2\sigma^2}z^Tz)exp(\frac{1}{\sigma^2}x^Tz)$$

Then we can using Taylor's formula of $exp(\frac{1}{\sigma^2}x^Tz)$ to get:

$$k(x,z) = exp(-\frac{1}{2\sigma^2}x^Tx)exp(-\frac{1}{2\sigma^2}z^Tz)(1 + \frac{x^Tz}{1!\sigma^2} + \frac{(x^Tz)^2}{2!(\sigma^2)^2} + ...)$$

And the last term can been written as:

$$(1 + \frac{x^Tz}{1!\sigma^2} + \frac{(x^Tz)^2}{2!(\sigma^2)^2} + ...) = (1 \times 1 + \sqrt{\frac{1}{1!\sigma^2}}x^T\sqrt{\frac{1}{1!\sigma^2}}z + \sqrt{\frac{1}{2!(\sigma^2)^2}}x^T\sqrt{\frac{1}{2!(\sigma^2)^2}}z + ...$$

Thus we can get the $\phi(x)$:

$$\phi(x) = exp(-\frac{1}{2\sigma^2}x^Tx)(1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!(\sigma^2)^2}}x, .., \sqrt{\frac{1}{n!(\sigma^2)^n}}x, ...)^T$$

Problem 3

(a)

(i)

We know $w_{t+1} = w_t + y_n\phi(x_n)$, Assumed $\alpha_t = (a_1, a_2, .., a_N)$,Thus:

$$w_{t+1} = \Phi^T\alpha_t + y_n\phi(x_n) = \Phi^T\alpha_{t+1}$$

where $\alpha_{t+1} = (a_1, a_2, .., a_n + y_n, ..., a_N)$. Thus, we prove that $\frac{w}{t+1} = \Phi^T\alpha_{t+1}$

(ii)

we will use induction to prove this. First, when t=0, we know $w_0 = 0$, so that we can rewrite it as:

$$w_0 = \Phi^T\alpha_0$$

where α_0 is a all zero vector. Hence we prove that for $t = 0$,w_0 can be expressed as $\Phi^T\alpha_0$. Then we assumed for $t = n$, w_n can be writen as $\Phi^T\alpha_n$. Then by part(i), we know that w_{n+1} can be expressed as $\Phi^T\alpha_{n+1}$.

Thus, by induction, we can prove that for any $0 \leq t \leq T$, w_t can be expressed as $\Phi^T\alpha_t$.

(b)

(i)

we have know that in previous part that if $\alpha_t = (a_1, a_2, .., a_N)$, then $\alpha_{t+1} = (a_1, a_2, .., a_n + y_n, ..., a_N)$. Hence maximum number of elements that differ between is 1.

(ii)

We have:

$$h(\phi(x_n), w_t) = w^t\phi(x) = \alpha_t^T\Phi\phi(x_n)$$

Assumed that $\alpha_t = (a_1, a_2, .., a_N)$. Then we have:

$$\alpha_t^T\Phi\phi(x) = \sum_{i=1}^{N} a_i\phi^T(x_i)\phi(x_n) = \sum_{i=1}^{N} a_i k(x_i, x_n)$$

So we prove what we want.

Problem 3 continued on next page. . .

(c)

Algorithm 1 Kernelize Perceptron training algorithm

$a_0 \leftarrow 0$
for $t = 0$ to $T - 1$ **do**
  Pick a random training example(x_n, y_n)
  Assumed $a_t = (a_{t_1}, ..., a_{t_N})$
  $h \leftarrow \sum_{i=1}^{N} a_{ti} k(x_i, x_n)$
  if $y_n h < 0$ **then**
    $a_{(t+1)_n} \leftarrow a_{t_n} + y_n$, update$a_{t+1}$
  end if
end for
return a_T

Problem 4

(a)

We have constraints that:

$$\varepsilon_n \geq 1 - y_n h(x_n)$$

$$\varepsilon_n \geq 0$$

Hence, once ε_n is greater than $max(0, 1 - y_n h(x_n))$, then it will greater than both of them. We can write it as:

$$\varepsilon_n \geq max(0, 1 - y_n h(x_n))$$

Now look at the objective time, we need to make ε_n as small as possible. Hence, we take $\varepsilon_n = max(0, 1 - y_n h(x_n)) == max(0, 1 - y_n(w^T x_n + b))$. Thusm the objective function can be writeen as:

$$\underset{w,b}{min}\, E(w, b), E(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} max(0, 1 - y_i(w^T x_i + b))$$

Hence we prove thay are equivalent.

(b)

We have:

$$\nabla_w \frac{1}{2} \|w\|^2 = w$$

$$\nabla_w 0 = 0$$

$$\nabla_w 1 - y_i(w^T x_i + b) = y_i x_i$$

Thus, if $1 - y_i(w^T x_i + b) > 0 \to y_i(w^T x_i + b) < 1$, the grident of w is $y_i x_i$ otherwise it will be zero. Thus:

$$\nabla_w max(0, 1 - y_i(w^T x_i + b)) = \sum_{i=1}^{N} I(y_i(w^T x_i + b) < 1) y_i x_i$$

Therefore:

$$\nabla_w E(w, b) = w - C \sum_{i=1}^{N} I(y_i(w^T x_i + b) < 1) y_i x_i$$

For b, we have:

$$\frac{\partial[1 - y_i(w^T x_i + b)]}{\partial b} = x_i$$

Similar to the analysis above, we can get:

$$\frac{\partial E(w,b)}{\partial b} = -C\sum_{i=1}^{N} I(y_i(w^T x_i + b) < 1)y_i$$

(c)

We can get the result:

w:[96. -36.64285714 233.57142857 88.28571429],b:-0.0689285714285714 [Iter 5: accuracy = 54.1667%

w:[-1.98076923 -11.71153846 25.35576923 11.32692308],b:-0.2867253946337398 [Iter 50: accuracy = 95.8333%

w:[-1.99019608 -4.81862745 11.45098039 5.74019608],b:-0.295685263623663 [Iter 100: accuracy = 95.8333%

w:[-0.499501 -0.3243513 1.05538922 1.28293413],b:-0.31806328958006436 [Iter 1000: accuracy = 95.8333%

w:[-0.3517593 -0.2779888 0.88644542 1.00329868],b:-0.33290319844971034 [Iter 5000: accuracy = 95.8333%

w:[-0.33655448 -0.28065645 0.89411863 0.98642119],b:-0.33432381099331177 [Iter 6000: accuracy = 95.8333%

(d)

With similar analysis we have in part(b) we can have:

$$\nabla_w E_i(w,b) = \frac{1}{N}w - CI(y_i(w^T x_i + b) < 1)y_i x_i$$

$$\frac{\partial E_i(w,b)}{\partial b} = -CI(y_i(w^T x_i + b) < 1)y_i$$

(e)

We can get the result:

w:[-1.60513517 -2.82975568 7.75514067 4.70009547],b:[-0.03916667] [Iter 5: accuracy = 95.8333%

w:[-1.68902612 -0.17971377 2.50267745 2.78270712],b:[-0.07074783] [Iter 50: accuracy = 95.8333%

w:[-1.21320347 0.08608695 1.68120436 2.20196636],b:[-0.07740539] [Iter 100: accuracy = 95.8333%

w:[-0.49457636 -0.18894245 0.95385434 1.14885559],b:[-0.10334756] [Iter 1000: accuracy = 95.8333%

w:[-0.42353581 -0.2382758 0.8887035 1.06173562],b:[-0.12178864] [Iter 5000: accuracy = 95.8333%

w:[-0.4428795 -0.21702285 0.90732014 1.06339658],b:[-0.12332915] [Iter 6000: accuracy = 95.8333%

Problem 5

(a)

By default hyperparameter given by LinearSVC, we get the test error is 0.3750%

(b)

We get the following results:

training size:50 Number of support vector:35

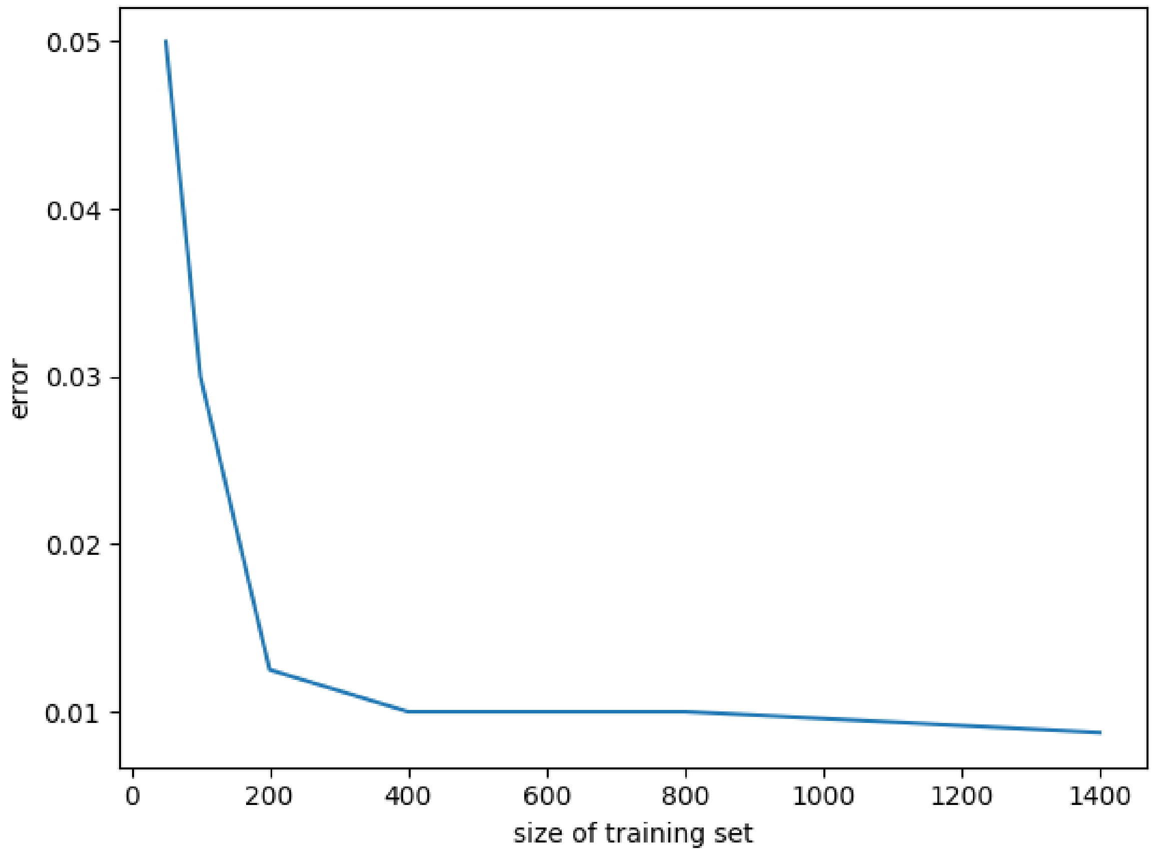
training size:100 Number of support vector:55

training size:200 Number of support vector:87

training size:400 Number of support vector:129

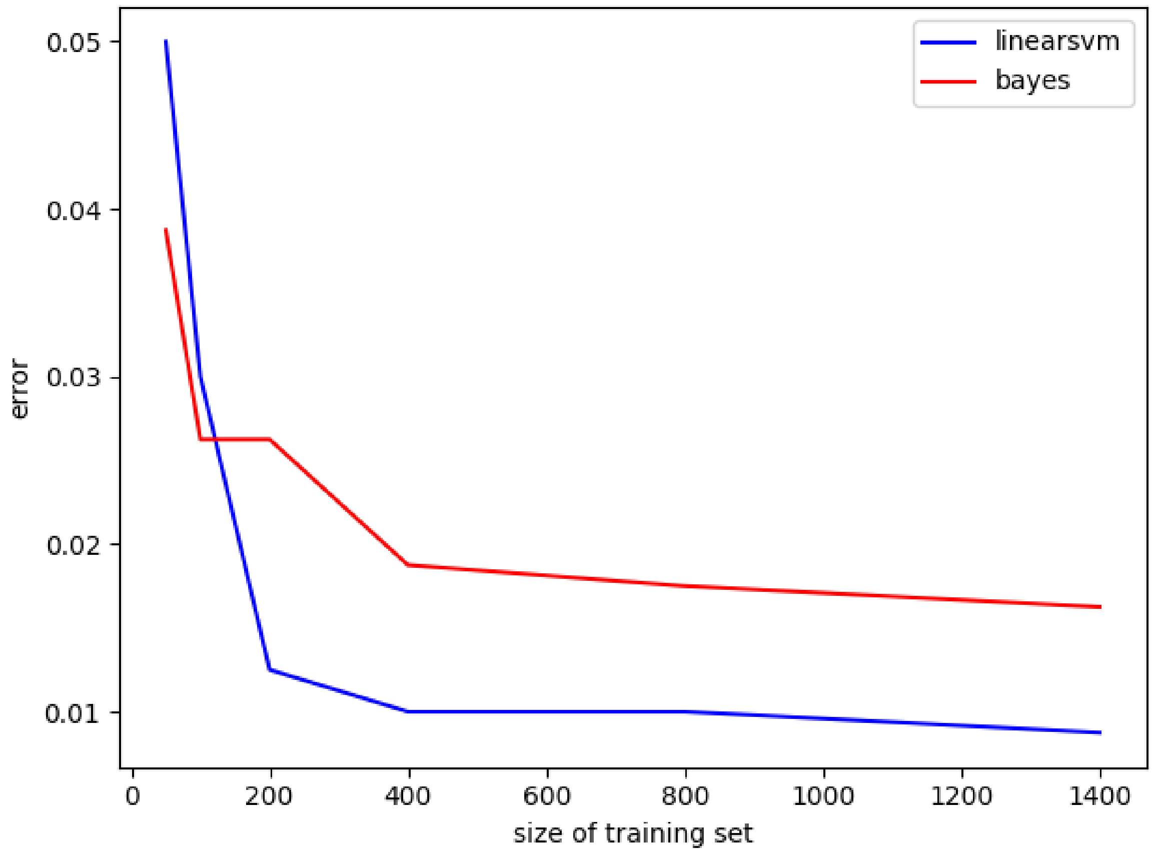
training size:800 Number of support vector:196

training size:1400 Number of support vector:234 We have the plot:



(c)

We get the plot with two methods and training size and error:



It is very clear that Linear SVM are better under all training set size.