Homework #2

Answer Sheet[1]

DEADLINE: 05/09/2017, 14:00
INSTRUCTOR: Hsuan-Tien Lin

王冠鈞 b03902027

1. To calculate the gradient $\nabla F(A, B)$, we just have to calculate the partial derivatives of them. For simplicity, in the function $F(A, B)$:

$$F(A,B) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + \exp(-y_n(A \cdot (\mathbf{w}_{\text{SVM}}^T \boldsymbol{\phi}(\mathbf{x}_n) + b_{\text{SVM}}) + B)))$$

, let $b_n = -y_n(Az_n + B)$, and $a_n = 1 + \exp(-y_n(Az_n + B)) = 1 + \exp(b_n)$. Then we can compute the partial derivatives with chain rule:

$$\frac{\partial F}{\partial A} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{d\ln a_n}{da_n}\cdot\frac{da_n}{db_n}\cdot\frac{db_n}{dA}\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{a_n}\cdot\exp(b_n)\cdot(-y_n z_n)\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(b_n)}{1+\exp(b_n)}\cdot(-y_n z_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\theta(b_n)\cdot(-y_n z_n) = \frac{1}{N}\sum_{n=1}^{N} -y_n p_n z_n$$

$$\frac{\partial F}{\partial B} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{d\ln a_n}{da_n}\cdot\frac{da_n}{db_n}\cdot\frac{db_n}{dB}\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{a_n}\cdot\exp(b_n)\cdot(-y_n)\right)$$

$$\frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(b_n)}{1+\exp(b_n)}\cdot(-y_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\theta(b_n)\cdot(-y_n) = \frac{1}{N}\sum_{n=1}^{N} -y_n p_n$$

Thus the gradient can be written in the form of vector/matrix as below, in terms of y_n, p_n, z_n, N:

$$\nabla F(A,B) = \frac{1}{N}\sum_{n=1}^{N}[-y_n p_n z_n, -y_n p_n]^T$$

2. According to the definition of the Hessian matrix, the answer should be[2]:

$$\begin{pmatrix} \frac{\partial^2 F}{\partial A^2} & \frac{\partial^2 F}{\partial A \partial B} \\ \frac{\partial^2 F}{\partial B \partial A} & \frac{\partial^2 F}{\partial B^2} \end{pmatrix}$$

, and we just have to compute their 2nd order derivatives:

$$\frac{\partial^2 F}{\partial A^2} = \frac{\partial}{\partial A}\left(\frac{1}{N}\sum_{n=1}^{N} -y_n p_n z_n\right) = \frac{1}{N}\sum_{n=1}^{N} -y_n z_n \left(\frac{dp_n}{d(\exp(b_n))}\cdot\frac{d\exp(b_n)}{db_n}\cdot\frac{db_n}{dA}\right)$$

[1]Some answers in this homework are referenced from/inspired by the questions/choices in the previous years of ML course homework.

[2]Since $y_n \in \{-1, 1\}$, $y_n^2 = 1$

$$= \frac{1}{N}\sum_{n=1}^{N} -y_n z_n \left(\frac{1}{(1+\exp(b_n))^2}\cdot\exp(b_n)\cdot(-y_n z_n)\right) = \frac{1}{N}\sum_{n=1}^{N}(-y_n z_n)^2 \cdot \frac{\exp(b_n)}{1+\exp(b_n)} \cdot \left(1-\frac{\exp(b_n)}{1+\exp(b_n)}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N} z_n^2 p_n(1-p_n)$$

Similarly, we can get $\frac{\partial^2 F}{\partial A \partial B} = \frac{\partial^2 F}{\partial B \partial A} = \frac{1}{N}\sum_{n=1}^{N} z_n p_n(1-p_n)$ and $\frac{\partial^2 F}{\partial B^2} = \frac{1}{N}\sum_{n=1}^{N} p_n(1-p_n)$. In conclusion, we can get the Hessian matrix:

$$H(F) = \frac{1}{N}\sum_{n=1}^{N}\begin{pmatrix} z_n^2 p_n(1-p_n) & z_n p_n(1-p_n) \\ z_n p_n(1-p_n) & p_n(1-p_n) \end{pmatrix}$$

3. In the Gaussian kernel $K(\mathbf{x},\mathbf{x}') = \exp(-\gamma||\mathbf{x}-\mathbf{x}'||^2)$, when $\gamma \to \infty$ and all $\mathbf{x}_n$ are different (i.e. $|\mathbf{x}_i - \mathbf{x}_j||^2 > 0, \forall i \neq j$), $\exp(-\gamma_{\to\infty}||\mathbf{x}-\mathbf{x}'||^2) \to 0$, and if $\mathbf{x} = \mathbf{x}'$, $\exp(-\gamma||\mathbf{x}-\mathbf{x}'||^2) = 1$. That is, the resulting kernel matrix K will have all terms 0 (since all $\mathbf{x}_n$ are different) except the diagonal terms, which will be 1. Thus, K will become a unit matrix, i.e. $K = I$.
The optimal $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = (\lambda I + K)^{-1}\mathbf{y} = (\lambda I + I)^{-1}\mathbf{y} = ((\lambda+1)I)^{-1}\mathbf{y} = \frac{1}{\lambda+1}\mathbf{y}$$

4. Now that if $\gamma \to 0$, $K(\mathbf{x},\mathbf{x}') = \exp(-\gamma_{\to 0}||\mathbf{x}-\mathbf{x}'||^2) \to 1$. It implies that the resulting kernel matrix K will be an all-1 matrix (i.e. $k_{ij} = 1, \forall 1 \leq i,j \leq N$), say it $\mathbf{1}_N$, where N is the number of data. And the optimal $\boldsymbol{\beta}$ is:[3]

$$\boldsymbol{\beta} = (\lambda I + K)^{-1}\mathbf{y} = \boldsymbol{\beta} = (\lambda I + \mathbf{1}_N)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1+\lambda & 1 & \cdots & 1 \\ 1 & 1+\lambda & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1+\lambda \end{bmatrix}^{-1}\mathbf{y} = \frac{1}{\lambda(\lambda+N)}\begin{bmatrix} \lambda+N-1 & -1 & \cdots & -1 \\ -1 & \lambda+N-1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \cdots & -1 & \lambda+N-1 \end{bmatrix}\mathbf{y}$$

5. In P_2, the violations $\xi_n^\vee, \xi_n^\wedge$ are squared, which means that when it violates with an amount of $\xi_n^\vee$ ($\xi_n^\wedge$), it will impose a penalty of $(\xi_n^\vee)^2$ ($(\xi_n^\wedge)^2$). Tracing back to the unconstrained problem of P_1[4], the term $\max(0, |\mathbf{w}^T\mathbf{z}_n + b - y_n| - \epsilon)$ means that it will impose a linear penalty if the difference between $\mathbf{w}^T\mathbf{z}_n + b$ and y_n is larger than ϵ. So intuitively, if we want a quadratic penalty that will lead to $(\xi_n^\vee)^2$ and $(\xi_n^\wedge)^2$, we just have to make the $(\max(\cdots))$ term sqared, i.e. the unconstrained problem will become:

$$\min_{b,\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}(\max(0, |\mathbf{w}^T\mathbf{z}_n + b - y_n| - \epsilon))^2$$

6. From the result of the previous problem and according to the the representer theorem, we can replace $\mathbf{w}^T$ with $\mathbf{w}_*$ and replace $\mathbf{z}_m\mathbf{z_n}$[5] with $K(\mathbf{x}_n, \mathbf{x_m})$. Thus the derived dual problem is:

$$\min_{b,\boldsymbol{\beta}} F(b,\boldsymbol{\beta}) = \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}\beta_n\beta_m K(\mathbf{x}_n,\mathbf{x}_m) + C\sum_{n=1}^{N}\left(\max\left(0, |\sum_{m=1}^{N}\beta_m K(\mathbf{x}_n,\mathbf{x}_m) + b - y_n| - \epsilon\right)\right)^2$$

[3]The inverse matrix is computed with Wolfram Alpha
[4]Lecture 206, page 13 in handout
[5]One of the $\mathbf{z}$ comes from the previous result and the other comes from $\mathbf{w}_*$.

$$\left(= \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}\beta_n\beta_m K(\mathbf{x}_n,\mathbf{x}_m) + C\sum_{n=1}^{N}\left(\max\left(0,|s_n-y_n|-\epsilon\right)\right)^2 \right)$$

We need to derive $\frac{\partial f(b,\boldsymbol{\beta})}{\partial \beta_m}$, but there are some unpleasant terms in the function, and we might have to split it into several cases.

(a) For the former term, $\frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}\beta_n\beta_m K(\mathbf{x}_n,\mathbf{x}_m)$, it's easy to find its derivative of β_m, which is $\sum_{n=1}^{N}\beta_n K(\mathbf{x}_n,\mathbf{x}_m)$.

(b) For the latter, we first have to find out whether the maximum is 0 or the other one.

- If 0 is larger than $|s_n-y_n|-\epsilon$, then the derivative is 0.
- If 0 is smaller than $|s_n-y_n|-\epsilon$, then we have to calculate the derivative of $|s_n-y_n|-\epsilon$.

For this condition, we can add 0/1 term $[\![|s_n-y_n|-\epsilon \geq 0]\!]$ so that if the maximum is 0 it will set the whole term to 0, and vice versa.
Then, we have to remove the absolute value inside the maximum:

- If $s_n \geq y_n$, then $\frac{\partial}{\partial\beta_m}(|s_n-y_n|-\epsilon) = \frac{\partial}{\partial\beta_m}(s_n-y_n-\epsilon) = K(\mathbf{x}_n,\mathbf{x}_m)$
- If $s_n < y_n$, then $\frac{\partial}{\partial\beta_m}(|s_n-y_n|-\epsilon) = \frac{\partial}{\partial\beta_m}(y_n-s_n-\epsilon) = -K(\mathbf{x}_n,\mathbf{x}_m)$

To combine, we can use $\text{sign}(s_n-y_n)$ to determine its sign and the derivation $\frac{\partial}{\partial\beta_m}(|s_n-y_n|-\epsilon) = \text{sign}(s_n-y_n)K(\mathbf{x}_n,\mathbf{x}_m)$ In total, the derivative of the latter term will be:

$$\frac{\partial}{\partial\beta_m}C\sum_{n=1}^{N}\left(\max\left(0,|s_n-y_n|-\epsilon\right)\right)^2 = \frac{\partial C\sum_{n=1}^{N}\left(\max\left(0,|s_n-y_n|-\epsilon\right)\right)^2}{\partial(|s_n-y_n|-\epsilon)}\cdot\frac{\partial(|s_n-y_n|-\epsilon)}{\partial\beta_m}$$

$$= 2C\sum_{n=1}^{N}[\![|s_n-y_n|-\epsilon\geq 0]\!](|s_n-y_n|-\epsilon)\cdot\text{sign}(s_n-y_n)K(\mathbf{x}_n,\mathbf{x}_m)$$

In conclusion, we derived the derivation of β_m:

$$\frac{\partial f(b,\boldsymbol{\beta})}{\partial\beta_m} = \sum_{n=1}^{N}\beta_n K(\mathbf{x}_n,\mathbf{x}_m) + 2C\sum_{n=1}^{N}[\![|s_n-y_n|-\epsilon\geq 0]\!](|s_n-y_n|-\epsilon)\text{sign}(s_n-y_n)K(\mathbf{x}_n,\mathbf{x}_m)$$

7. Let the independently generated examples by f are $(x_1,x_1^2),(x_2,x_2^2)$. Since the model it uses is the minimization of the squared error in linear regression, we can write down its in-data error:

$$E_{\text{in}} = \sum_i (h(x_i)-y_i)^2 = (w_1x_1+w_0-x_1^2)^2 + (w_1x_2+w_0-x_2^2)^2$$

To get a solution of w_1, w_0, we first do some differentiation:

$$\frac{\partial E_{\text{in}}}{\partial w_1} = 2(w_1x_1+w_0-x_1^2)x_1 + 2(w_1x_2+w_0-x_2^2)x_2 = 0$$

$$\frac{\partial E_{\text{in}}}{\partial w_2} = 2(w_1x_1+w_0-x_1^2) + 2(w_1x_2+w_0-x_2^2) = 0$$

Then we solve,

$$(x_1+x_2)w_1 + 2w_0 = x_1^2+x_2^2$$
$$(x_1^2+x_2^2)w_1 + (x_1+x_2)w_0 = x_1^3+x_2^3$$

From the equations above we get:

$$\begin{cases} w_1 = x_1 + x_2 \\ w_2 = -x_1 x_2 \end{cases}$$

Thus we get $h(x) = (x_1 + x_2)x - x_1x_2$, and $\bar{g}(x) = E[h(x)] = E[(x_1 + x_2)x - x_1x_2] = E[x_1 + x_2]x - E[x_1x_2]$. Since the distribution for generating x_1, x_2 is uniform in $[0, 1]$, the pdf of the distribution is $\frac{1}{1-0} = 1$, and thus the expectation for x_1, x_2 is $\int_0^1 x \cdot 1dx = \frac{1}{2} \Rightarrow E[x_1 + x_2] = 1, E[x_1x_2] = \frac{1}{4}$. Thus $\bar{g}(x) = x - \frac{1}{4}$.

8. I've come up with a method to obtain $g(\tilde{\mathbf{x}})$ with $\tilde{N}$ queries. The method is shown as followed:

(a) First, construct an arbitrary hypothesis h_1, and send a query to obtain RMSE(h_1), and we can get the normal squared error of h_1:

$$E(h_1) = \tilde{N} \cdot \text{RMSE}(h_1)^2 = (\tilde{y}_1 - h_1(\tilde{\mathbf{x}}_1))^2 + (\tilde{y}_2 - h_1(\tilde{\mathbf{x}}_2))^2 + \cdots + (\tilde{y}_{\tilde{N}-1} - h_1(\tilde{\mathbf{x}}_{\tilde{N}-1}))^2 + (\tilde{y}_{\tilde{N}} - h_1(\tilde{\mathbf{x}}_{\tilde{N}}))^2$$

(b) Then, construct another hypothesis h_2 such that for all $\tilde{\mathbf{x}}_i$ except $\tilde{\mathbf{x}}_1$, $h_2(\tilde{\mathbf{x}}_i) = h_1(\tilde{\mathbf{x}}_i)$, and then query for RMSE(h_2) in order to get $E(h_2)$.

(c) Now that we get the value of two errors, from the properties above, we can do such operations:

$$E(h_1) - E(h_2) = (\tilde{y}_1 - h_1(\tilde{\mathbf{x}}_1))^2 - (\tilde{y}_1 - h_2(\tilde{\mathbf{x}}_1))^2$$

Since we know $E(h_1), E(h_2), h_1(\tilde{\mathbf{x}}_1), h_2(\tilde{\mathbf{x}}_1)$, we can get $\tilde{y}_1$ by solving the quadratic equation above.

(d) After that, we can construct a new hypothesis h_3 such that $\tilde{y}_1 - h_3(\tilde{\mathbf{x}}_1) = 0$, and except for $\tilde{\mathbf{x}}_2$ (and $\tilde{\mathbf{x}}_1$), $h_3(\tilde{\mathbf{x}}_i) = h_2(\tilde{\mathbf{x}}_i)$; with h_2 and the query of error of h_3, we can get $\tilde{y}_2$. Continue the same process until $h_{\tilde{N}}$, and at this time we've queried for $\tilde{N} - 1$ times and have got $\tilde{y}_1, \cdots, \tilde{y}_{\tilde{N}-1}$.

(e) As for $\tilde{y}_{\tilde{N}}$, we can construct a $h_{\tilde{N}+1}$, such that $E(h_{\tilde{N}+1}) = (\tilde{y}_{\tilde{N}} - h_{\tilde{N}}(\tilde{\mathbf{x}}_{\tilde{N}}))^2$ and can be known without sending a query.[6] Then we can finally get $\tilde{y}_{\tilde{N}}$.

The total number of queries above is $\tilde{N}$.

9. I've come up with a method to obtain $\mathbf{g}^T\tilde{\mathbf{y}}$ with only 2 times. The method is shown as followed:

(a) Arbitrarily generate a hypothesis g_1, and let a new hypothesis $g_2 = g_1 + g$.

(b) Send a query for g_1, and another for g_2 so that we can compute the squared errors $E(g_1), E(g_2)$:

$$E(g_1) = \tilde{N} \cdot \text{RMSE}(g_1)^2 = (\tilde{y}_1 - g_1(\tilde{\mathbf{x}}_1))^2 + \cdots + (\tilde{y}_{\tilde{N}} - g_1(\tilde{\mathbf{x}}_{\tilde{N}}))^2$$

$$= ((g_1(\tilde{\mathbf{x}}_1))^2 + \cdots + (g_1(\tilde{\mathbf{x}}_{\tilde{N}}))^2) - 2(g_1(\tilde{\mathbf{x}}_1)\tilde{y}_1 + \cdots + \tilde{g}_1(\tilde{\mathbf{x}}_{\tilde{N}})y_{\tilde{N}}) + (\tilde{y}_1^2 + \cdots + \tilde{y}_{\tilde{N}}^2)$$

$$E(g_2) = ((g_2(\tilde{\mathbf{x}}_1))^2 + \cdots + (g_2(\tilde{\mathbf{x}}_{\tilde{N}}))^2) - 2(g_2(\tilde{\mathbf{x}}_1)\tilde{y}_1 + \cdots + g_2(\tilde{\mathbf{x}}_{\tilde{N}})y_{\tilde{N}}) + (\tilde{y}_1^2 + \cdots + \tilde{y}_{\tilde{N}}^2)$$

[6] Since $E(h_{\tilde{N}}) = (\tilde{y}_{\tilde{N}-1} - h_{\tilde{N}}(\tilde{\mathbf{x}}_{\tilde{N}-1}))^2 + (\tilde{y}_{\tilde{N}} - h_{\tilde{N}}(\tilde{\mathbf{x}}_{\tilde{N}}))^2$ and $E(h_{\tilde{N}})$, $(\tilde{y}_{\tilde{N}-1} - h_{\tilde{N}}(\tilde{\mathbf{x}}_{\tilde{N}-1}))^2$ are known, we can construct $h_{\tilde{N}+1}$ such that $\tilde{y}_{\tilde{N}-1} - h_{\tilde{N}+1}(\tilde{\mathbf{x}}_{\tilde{N}-1}) = 0$ and $h_{\tilde{N}+1}(\tilde{\mathbf{x}}_{\tilde{N}}) = h_{\tilde{N}}(\tilde{\mathbf{x}}_{\tilde{N}})$

(c) Since g_1, g_2 are constructed by our own, the terms $((g_1(\tilde{\mathbf{x}}_1))^2 + \cdots + (g_1(\tilde{\mathbf{x}}_{\tilde{N}}))^2)$ and $((g_2(\tilde{\mathbf{x}}_1))^2 + \cdots + (g_2(\tilde{\mathbf{x}}_{\tilde{N}}))^2)$ are already known, and we can compute the difference:

$$[E(g_2) - ((g_2(\tilde{\mathbf{x}}_1))^2 + \cdots + (g_2(\tilde{\mathbf{x}}_{\tilde{N}}))^2)] - [E(g_1) - ((g_1(\tilde{\mathbf{x}}_1))^2 + \cdots + (g_1(\tilde{\mathbf{x}}_{\tilde{N}}))^2)]$$

$$= -2[(g_2(\tilde{\mathbf{x}}_1) - g_1(\tilde{\mathbf{x}}_1))\tilde{y}_1 + \cdots + (g_2(\tilde{\mathbf{x}}_{\tilde{N}}) - g_1(\tilde{\mathbf{x}}_{\tilde{N}}))\tilde{y}_{\tilde{N}}] = -2(g(\tilde{\mathbf{x}}_1)\tilde{y}_1 + \cdots + g(\tilde{\mathbf{x}}_{\tilde{N}})y_{\tilde{N}})$$

$$= -2\mathbf{g}^T\tilde{\mathbf{y}}$$

Thus we can get $\mathbf{g}^T\tilde{\mathbf{y}}$ from the above method with 2 queries.

10. Since RMSE is monotonic to squared error, we can just minimize its squared error ($\min_{\alpha_1,\alpha_2,\cdots,\alpha_K} E(\sum_{k=1}^{K}\alpha_k g_k)$) so as to get its minimum RMSE. Expanding the squared error, we get:

$$E(\sum_{k=1}^{K}\alpha_k g_k) = \sum_{n=1}^{\tilde{N}}\left(\tilde{y}_n - \sum_{i=1}^{K}\alpha_i g_i(\tilde{x}_n)\right)^2 = \sum_{n=1}^{\tilde{N}}\tilde{y}_n^2 + \text{terms containing } \alpha_1, \cdots, \alpha_K$$

Since $\sum_{n=1}^{\tilde{N}}\tilde{y}_n^2$ is constant, we can simply ignore it in the minimization process. As for the latter terms, we first arrange all the terms containing α_i:

$$E_{\alpha_i} = \alpha_i^2\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)^2 - 2\alpha_i\mathbf{g}_i^T\tilde{\mathbf{y}} + 2\alpha_i\sum_{0\le(j\neq i)\le K}\left(\alpha_j\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)g_j(\tilde{\mathbf{x}}_n)\right)$$

Then its first order condition, i.e. $\frac{\partial E_{\alpha_i}}{\partial\alpha_i}(=\frac{\partial E}{\partial\alpha_i}) = 0$, is:

$$\frac{\partial E}{\partial\alpha_i} = 2\alpha_i\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)^2 - 2\mathbf{g}_i^T\tilde{\mathbf{y}} + 2\sum_{0\le(j\neq i)\le K}\left(\alpha_j\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)g_j(\tilde{\mathbf{x}}_n)\right)$$

$$= \left(2\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)g_1(\tilde{\mathbf{x}}_n)\right)\alpha_1 + \cdots + \left(2\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)^2\right)\alpha_i + \cdots + \left(2\sum_{n=1}^{\tilde{N}} g_i(\tilde{\mathbf{x}}_n)g_1(\tilde{\mathbf{x}}_n)\right)\alpha_K = 2\mathbf{g}_i^T\tilde{\mathbf{y}}$$

. which is an linear equation of $(\alpha_1, \cdots, \alpha_K)$. That is, we have K first order conditions $\frac{\partial E}{\partial\alpha_1}, \cdots, \frac{\partial E}{\partial\alpha_K}$, so we have K such equations to solve all α_i. We can follow the steps (algorithm) below:

(a) For $i = 1$ to K, make 2 queries, as the result of the previous question, to get $\mathbf{g}_k^T\tilde{\mathbf{y}}$.

(b) Construct the following matrices:

$$A = \sum_{n=1}^{\tilde{N}}\begin{bmatrix} (g_1(\tilde{\mathbf{x}}_n))^2 & \cdots & g_1(\tilde{\mathbf{x}}_n)g_i(\tilde{\mathbf{x}}_n) & \cdots & g_1(\tilde{\mathbf{x}}_n)g_K(\tilde{\mathbf{x}}_n) \\ \vdots & \ddots & \vdots & & \vdots \\ g_i(\tilde{\mathbf{x}}_n)g_1(\tilde{\mathbf{x}}_n) & \cdots & (g_i(\tilde{\mathbf{x}}_n))^2 & \cdots & g_i(\tilde{\mathbf{x}}_n)g_K(\tilde{\mathbf{x}}_n) \\ \vdots & & \vdots & \ddots & \vdots \\ g_K(\tilde{\mathbf{x}}_n)g_1(\tilde{\mathbf{x}}_n) & \cdots & g_K(\tilde{\mathbf{x}}_n)g_i(\tilde{\mathbf{x}}_n) & \cdots & (g_K(\tilde{\mathbf{x}}_n))^2 \end{bmatrix}, B = \begin{bmatrix} \mathbf{g}_1^T\tilde{\mathbf{y}} \\ \vdots \\ \mathbf{g}_i^T\tilde{\mathbf{y}} \\ \vdots \\ \mathbf{g}_K^T\tilde{\mathbf{y}} \end{bmatrix}$$

(c) Solve the following linear equation:

$$A\boldsymbol{\alpha} = B$$

, where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_K]^T$. Return the solution.

The algorithm made 2 queries in every iteration of the first step, so in total it made $2K$ queries.

11. In my result, there are multiple combinations:

$$(\gamma, \lambda) = \{(32, 0.001), (32, 1), (32, 1000), (2, 0.001), (2, 1), (2, 1000), (0.125, 0.001)\}$$

that will lead to the minimum $E_{in}(g)$, which is 0.[7]

12. The combination $(\gamma, \lambda) = (0.125, 1000)$ leads to the lowest $E_{out}(g)$, which is 39%.

13. In my result, there are multiple combinations:

$$(\gamma, C) = \{(32, 1), (32, 1000), (2, 1), (2, 1000), (0.125, 1000)\}$$

that will lead to the minimum $E_{in}(g)$, which is 0.

14. The combination $(\gamma, C) = (0.125, 1)$ leads to the lowest $E_{out}(g)$, which is 42%.

15. In my implementation, the number of iterations is 200, and I didn't add $x_0 = 1$ into my data. Moreover, in every iteration, I performed bootstrapping on the first 400 data and sampled 400 data from them (which may be sampled multiple times) as $\tilde{\mathcal{D}}_t$.
After running a few times, although the results may differ, using the parameter $\lambda = 0.1$ seems to give us a lowest $E_{in}(g)$, which is 32%.

16. After running a few times, although the results may differ, using the parameter $\lambda = 0.01$ seems to give us a lowest $E_{out}(g)$, which is about 37%.
The result of this question emphasizes that using bagging along with linear kernel may have better results than using Gaussian kernel alone.

17. The problem:

$$\min_{\alpha_t} \frac{1}{N} \sum_{n=1}^{N} \max \left(1 - y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n), 0 \right)$$

, its constrained form is:

$$\begin{aligned} \min_{\alpha_t} \quad & \frac{1}{N} \sum_{n=1}^{N} \xi_n \\ \text{subject to} \quad & y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \text{ , for } n = 1, 2, \cdots, N \end{aligned}$$

That's the most that I can derive.

[7] By the way, in Questions 11, 12, 15, 16, I used the package `numpy` for matrix computation.