

HW5 Solutions

● Graded

1 Day, 5 Hours Late

Student

Chih-Hao Liao

Total Points

100 / 100 pts

Question 1

Problem 1

20 / 20 pts

✓ - 0 pts Correct

- 4 pts Explain the first bound in the wrong way
- 4 pts Explain the second bound in the wrong way
- 4 pts Explain the third bound in the wrong way
- 4 pts Explain the forth bound in the wrong way
- 4 pts Explain the fifth bound in the wrong way
- 20 pts wrong answer or empty page

Question 2

Problem 2

20 / 20 pts

✓ - 0 pts Correct

- 5 pts Should be max
- 5 pts Should be $y_n < 0$
- 5 pts Some minor mistakes.
- 10 pts Some mistakes.
- 15 pts Some major mistakes.
- 15 pts Some reasonable effort
- 20 pts Totally incorrect.

Question 3

Problem 4

20 / 20 pts

- 20 pts No solution or wrong answer

✓ - 0 pts Correct

- 5 pts Some minor mistakes / missing some details

- 10 pts Major mistake or Several minor mistakes

- 10 pts Missing some derivation steps / details

- 15 pts Some reasonable effort but fundamentally wrong

- 20 pts Totally wrong solution or almost missing solution

- 20 pts Showed little to no derivation steps

- 20 pts No page selected or wrong page selected

- 10 pts Using examples to infer result without rigorous proof

Question 4

Problem 9

20 / 20 pts

✓ - 0 pts Correct

- 5 pts 1. One minor mistake during the derivation steps
2. One missing derivation step

- 10 pts 1. One major mistake
2. Missing several derivation steps
3. Several minor mistakes

- 15 pts Some reasonable effort but fundamentally wrong

- 20 pts 1. Little or no derivation steps
2. Wrong solution or no solution
3. No page selected
4. Wrong page selected

Question 5

Problem 10

20 / 20 pts

✓ - 0 pts Correct

- 5 pts minor error

- 10 pts Skip too many steps

 Incorrect $\eta(\alpha_t)$

- 15 pts Skip essential steps

- 15 pts Some reasonable effort but fundamentally wrong

- 15 pts Without rigorous proof

- 20 pts 1. No page selected or wrong page selected

 2. Wrong solution or no solution

Question 6

Coding Part

0 / 0 pts

6.1 Problem 17

0 / 0 pts

✓ - 0 pts Correct

- 200 pts No submission

- 80 pts Unclear solution

- 80 pts Using `sklearn.tree.DecisionTreeClassifier` does not meet the requirement of the problem.

- 200 pts Unclear submission.

- 80 pts No submission for P17-P20

- 120 pts Unclear solution for P11-P16

- 20 pts No submission for P15

- 80 pts Unclear submission for P17-P20

- 20 pts Lateness penalty

Question assigned to the following page: [1](#)

HTML_2023_HW5

tags: Personal

Discuss with anonymous and TAs.

Support Vector Machines

P1

$$\begin{aligned} & \min_{b,w,\xi} \frac{1}{2} w^T w + C \cdot \sum_{n=1}^N \xi_n \\ & \text{s.t. } y_n(w^T z_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \quad \forall n \end{aligned}$$

Since we have the slack variable equations above, we know that ξ is used to record the number of margin violation, and it shows that

- For (x_n, y_n) violate the margin: $\xi_n = 1 - y_n(w^T z_n + b)$
- For (x_n, y_n) not violate the margin: $\xi_n = 0$

To determine the number of misclassified examples, we need to consider the values of ξ_n , when $\xi_n > 1$, it indicates a misclassified example. Therefore, we can simply check the summation of misclassified examples by given $\xi = 1$, which means samples are on the boundaries, and the summation of ξ should not larger than n , which is the number of all the samples.

As a result, we have

Questions assigned to the following page: [1](#) and [2](#)

$$\begin{aligned}
\sum_{n=1}^N \frac{\xi_n^*}{2} &\Rightarrow \sum_{n=1}^N \frac{1}{2} \neq n \\
\sum_{n=1}^N \sqrt{\xi_n^*} &\Rightarrow \sum_{n=1}^N 1 = n \\
\sum_{n=1}^N \xi_n^* &\Rightarrow \sum_{n=1}^N 1 = n \\
\sum_{n=1}^N \lfloor \xi_n^* \rfloor &\Rightarrow \sum_{n=1}^N 1 = n \\
\sum_{n=1}^N \log_2(1 + \xi_n^*) &\Rightarrow \sum_{n=1}^N 1 = n
\end{aligned}$$

As a result, only choice $\sum_{n=1}^N \frac{\xi_n^*}{2}$ is not the upper bound of the number of misclassified examples, and there are 4 correct outcomes reach the goal.

As a result, we should choose $[d]$ as our solution.

P2

Consider a soft-margin primal SVM, by complementary slackness, we have

$\alpha_n(1 - \xi_n - y_n(w^T z_n + b)) = 0$ if $\forall n : \alpha_n^* = C \neq 0$, which entails that

$1 - \xi_n - y_n(w^T z_n + b) = 0$, hence, $y_n b = 1 - \xi_n - y_n \sum_{m=1}^N y_m \alpha_m K(x_n, x_m)$. Another restriction is that $\xi_n \geq 0$, $\xi_n = (1 - y_n(w^T z_n + b)) \geq 0$, hence, $y_n(w^T z_n + b) \leq 1$

Given $y_n = 1$, the upper bound of b can be obtained, so we let $y_n = -1$

$$\begin{aligned}
y_n b &= -1 \cdot b = 1 - \underbrace{\xi_n}_{\geq 0} + \sum_{m=1}^N y_m \alpha_m K(x_n, x_m) \geq 1 + \sum_{m=1}^N y_m \alpha_m K(x_n, x_m) \\
b &\geq -1 - \sum_{m=1}^N y_m \alpha_m^* K(x_n, x_m)
\end{aligned}$$

In order to find the smallest b^* , we have to take the maximum of the equation above, which means

$$b^* \geq \max_{n: y_n < 0} \left(-1 - \sum_{m=1}^N y_m \alpha_m^* K(x_n^T x_m) \right)$$

As a result, we should choose $[d]$ as our solution.

Question assigned to the following page: [3](#)

P3

Adding Lagrange multipliers α_n to P_2 , we have

$$\begin{aligned}\mathcal{L}(\alpha, w, b) &= \frac{1}{2}w^T w + C \sum_{n=1}^N \xi_n^2 + \sum_{n=1}^N \alpha_n(1 - \xi_n - y_n(w^T \phi(x_n) + b)) \\ \frac{\partial \mathcal{L}(\alpha, w, b)}{\partial b} &= - \sum_{n=1}^N \alpha_n y_n = 0 \\ \frac{\partial \mathcal{L}(\alpha, w, b)}{\partial \xi_n} &= 2C\xi_n - \alpha_n \Rightarrow \xi_n = \frac{\alpha_n}{2C} \\ \frac{\partial \mathcal{L}(\alpha, w, b)}{\partial w} &= w - \sum_{n=1}^N \alpha_n y_n \phi(x_n) = 0 \Rightarrow w = \sum_{n=1}^N \alpha_n y_n \phi(x_n)\end{aligned}$$

Since both ξ_n and α_n are the n-th element in the vector ξ and α_n , the optimal $\xi_n^* = \frac{\alpha_n^*}{2C}$ (vector).

As a result, we should choose [a] as our solution.

P4

The $K_{d,s}(x, x')$ actually means that how classifiers classify x and x' in the same group since classifiers classify x and x' into different groups. From the observation, if $x = x'$, any θ will make both classified into the same group. For any L and R , ($L < R$), there are $(R - L + 1)$ even integers between $2L$ and $2R$, and there are $(R - L)$ odd integers, which means there are $(R - L)$ candidate θ , and each θ can generate 2 decision stump. Therefore,

$K_{ds}(x, x') = 2d(R - L)$ if $x = x'$. Moreover, we can see that if a classifier $g_{s,i,\theta}$ is on the same side of x and x' (both x and x' are bigger or smaller than θ), this entails that a classifier $g_{s,i,\theta}$ classify x and x' different when it is between x and x' , and there exists $\frac{|x_i - x'_i|}{2}$ different choices of θ for fixed i and s . That means there are total $2 \sum_{i=1}^d \frac{|x_i - x'_i|}{2} = \|x - x'\|_1$ classify x and x' different.

$$\begin{aligned}K_{ds}(x, x') &= \left(\sum_{\text{All combinations}} [g_{s,i,\theta}(x) = g_{s,i,\theta}(x')] \right) - \left(\sum_{\text{All combinations}} [g_{s,i,\theta}(x) \neq g_{s,i,\theta}(x')] \right) \\ &= \underbrace{2d(R - L)}_{\text{All combinations}} - \|x - x'\|_1 - \|x - x'\|_1, \text{ where } x \neq x' \\ &= 2d(R - L) - 2\|x - x'\|_1\end{aligned}$$

As a result, we should choose [e] as our solution.

No questions assigned to the following page.

Bagging and Boosting

P5

In order to find the upper bound of $E_{out}(G)$, we have to classify a point incorrectly, and for $2M + 1$ binary classifiers, there must be $\frac{(2M+1)+1}{2}$ classifiers that classified the points incorrectly. If there exists N points, there exists $(\sum_{t=1}^{2M+1} e_t)N$ points that classified incorrectly. Therefore, the number of points that classified incorrectly by aggregated binary classifiers can be represented as below.

$$\frac{(\sum_{t=1}^{2M+1} e_t)N}{\frac{(2M+1)+1}{2}}$$

And the upper bound of $E_{out}(G)$ equals to

$$\frac{(\sum_{t=1}^{2M+1} e_t)N}{\frac{(2M+1)+1}{2}N} = \frac{1}{M+1} \sum_{t=1}^{2M+1} e_t$$

As a result, we should choose [b] as our solution.

P6

The probability P that every sample doesn't duplicated equals to

$$P = \frac{1127}{1127} \times \frac{1126}{1127} \times \frac{1125}{1127} \times \cdots \times \frac{1127 - N}{1127} = \prod_{n=0}^{N-1}$$

And the probability of samples that duplicate at least once equals to $1 - P$, therefore, we have

$$\frac{\binom{1127}{n} \cdot n!}{1127^n} \leq (1 - 0.75) = 0.25$$

No questions assigned to the following page.

```

1 |     from math import comb, factorial
2 |
3 |     N=1127;P=0.75;target_n = 0
4 |     for n in range(1,100):
5 |         if (((factorial(n)*comb(N, n))/(N**n))<=(1-P)):
6 |             print("Probability: ", (factorial(n)*comb(N, n))/(N**n))
7 |             target_n = n; break
8 |
9 |     print("Smallest N:", target_n)

```

```

1 | Probability:  0.24920995191245734
2 | Smallest N: 56

```

As a result, we should choose [b] as our solution.

P7

$$\min_w E_{in}^u(w) = \frac{1}{N} \sum_{n=1}^N u_n (y_n - w^T x_n)^2$$

Since $u_n \geq 0$, we can rewrite the equation as below.

$$E_{in}^u(w) = \frac{1}{N} \sum_{n=1}^N u_n (y_n - w^T x_n)^2 = \frac{1}{N} \sum_{n=1}^N (\sqrt{u_n} y_n - w^T \sqrt{u_n} x_n)^2$$

Then we can let $(\tilde{x}_n, \tilde{y}_n) = (\sqrt{u_n} x_n, \sqrt{u_n} y_n)$, then the equation above can be represented as below.

$$E_{in}^u(w) = \frac{1}{N} \sum_{n=1}^N (\tilde{y}_n - w^T \tilde{x}_n)^2$$

As a result, we should choose [c] as our solution.

P8

[a]

- Gini index for the first part: $1 - (1)^2 - (0)^2 = 0$
- Gini index for the second part: $1 - (0.5)^2 - (0.5)^2 = 0.5$
- Overall Gini index: $0 \cdot 0.5 + 0.5 \cdot 0.5 = 0.25$

Question assigned to the following page: [4](#)

[b]

- Gini index for the first part: $1 - (0.8)^2 - (0.2)^2 = 0.32$
- Gini index for the second part: $1 - (0.75)^2 - (0.25)^2 = 0.375$
- Overall Gini index: $0.32 \cdot 0.7 + 0.375 \cdot 0.3 = 0.3365$

[c]

- Gini index for the first part: $1 - (0.7)^2 - (0.3)^2 = 0.42$
- Gini index for the second part: $1 - (0)^2 - (1)^2 = 0$
- Overall Gini index: $0.42 \cdot 0.9 + 0 \cdot 0.1 = 0.378$

[d]

- Gini index for the first part: $1 - (0.8)^2 - (0.2)^2 = 0.32$
- Gini index for the second part: $1 - (0.9)^2 - (0.1)^2 = 0.18$
- Overall Gini index: $0.32 \cdot 0.8 + 0.18 \cdot 0.2 = 0.292$

[e]

- Gini index for the first part: $1 - (0.9)^2 - (0.1)^2 = 0.18$
- Gini index for the second part: $1 - (0.9)^2 - (0.1)^2 = 0.18$
- Overall Gini index: $0.18 \cdot 0.8 + 0.18 \cdot 0.2 = 0.18$

Since [e] has minimum Gini index, [e] has the best branch for building a CART decision tree.

As a result, we should choose [e] as our solution.

Ref: Day 22 : 決策樹 (<https://ithelp.ithome.com.tw/articles/10276079>)

P9

Assume that when iteration at t , there are a incorrect, and a terms correct, and from the lecture slide Lecture 12: Bagging and Boosting P.34 , we have

Questions assigned to the following page: [4](#) and [5](#)

$$\begin{aligned}
\blacklozenge_t &= \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \geq 1, 0 \leq \epsilon_t \leq \frac{1}{2} \\
U_{t+1} &= \sum_{n=1}^N u_n^{t+1} \\
&= \sum_{n=1}^N u_n^t \times \blacklozenge_t \cdot \llbracket y_n \neq g_t(x_n) \rrbracket + \sum_{n=1}^N u_n^t \times \frac{1}{\blacklozenge_t} \times \llbracket y_n \neq g_t(x_n) \rrbracket \\
&= \epsilon_t \times \blacklozenge_t \times \sum_{n=1}^N u_n^t + \frac{1 - \epsilon_t}{\blacklozenge_t} \times \sum_{n=1}^N u_n^t \\
&= U_t \times (\epsilon_t \blacklozenge_t + \frac{1 - \epsilon_t}{\blacklozenge_t}) \\
&= 2U_t \sqrt{\epsilon_t(1 - \epsilon_t)} \\
&= 2\sqrt{\epsilon_t(1 - \epsilon_t)}(2U_{t-1} \sqrt{\epsilon_{t-1}(1 - \epsilon_{t-1})}) \\
&= 2^T \prod_{t=1}^T \sqrt{\epsilon_t(1 - \epsilon_t)}
\end{aligned}$$

As a result, we should choose $[d]$ as our solution.

P10

From the lecture slide, we have

$$\min_{\eta} \frac{1}{N} \sum_{n=1}^N \text{err}\left(\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta g_t(x_n), y_n\right) \text{ with } \text{err}(s, y) = (s - y)^2$$

and we can rewrite the equation as below.

$$\begin{aligned}
\text{eq1} &\Rightarrow \min_{\eta} \frac{1}{N} \sum_{n=1}^N (s_n + \eta g_t(x_n) - y_n)^2 \\
\frac{\partial(\text{eq1})}{\partial \eta} &= \frac{1}{N} \sum_{n=1}^N \cdot 2 \cdot (s_n + \eta g_t(x_n) - y_n) \cdot (g_t(x_n)) = 0 \\
&\Rightarrow \sum_{n=1}^N (s_n + \eta g_t(x_n) - y_n) \cdot (g_t(x_n)) = 0 \\
\therefore s_{n+1} &= s_n + \eta g_t(x_n) \\
\therefore \Rightarrow \sum_{n=1}^N (s_{n+1} - y_n) \cdot g_t(x_n) &= 0
\end{aligned}$$

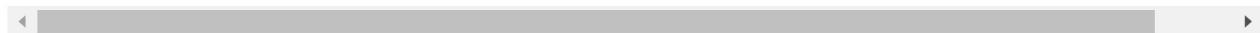
Question assigned to the following page: [5](#)

As a result, we should choose $[b]$ as our solution.

Experiments with Soft-Margin SVM and AdaBoos

P11

```
1 import numpy as np
2 from libsvm.svmutil import *
3
4 def main(args):
5     y_train, x_train = svm_read_problem(args['svm_train'])
6     target = 1
7
8     for i in range(len(y_train)):
9         y_train[i] = 1 if y_train[i] == target else 0
10
11    prob = svm_problem(y_train, x_train)
12    param = svm_parameter('-s 0 -t 0 -c {} -q'.format(args['C']))
13    m = svm_train(prob, param)
14    support_vector_coefficients = m.get_sv_coef()
15    support_vectors = m.get_SV()
16
17    height = len(support_vector_coefficients)
18    width = 0
19    primal = list()
20
21    for i in range(len(support_vectors)):
22        width = max(max(support_vectors[i].keys()), width)
23
24    for i in range(width):
25        accuracy = 0
26        for j in range(height):
27            if(i+1) in support_vectors[i]:
28                accuracy += support_vectors[j][i+1] * support_vector_coefficients[j]
29        primal.append(accuracy)
30    primal = np.array(primal)
31
32    print("||w||: ", sum(primal * primal) ** 0.5)
```



No questions assigned to the following page.

```
1 if __name__ == '__main__':
2     args = {
3         'C': 1,
4         'svm_train': "svm_letter.scale.tr",
5         'svm_test': "svm_letter.scale.t",
6     }
7
8     main(args)
```

```
1 ||w||: 6.305469435447087
```

As a result, we should choose $[c]$ as our solution.

P12

```
1 import numpy as np
2 from libsvm.svmutil import *
3
4 def main(args):
5     for classifier in range(2,7):
6         print(f'==> "{classifier}" versus "not {classifier}"')
7         y_train, x_train = svm_read_problem(args['svm_train'])
8         for type in range(len(y_train)):
9             y_train[type] = 1 if y_train[type] == classifier else 0
10
11         prob = svm_problem(y_train, x_train)
12         param = svm_parameter('-s 0 -t 1 -d {Q} -g 1 -r 1 -c {C} -q'.format(C=args[
13             'C'], Q='Q'))
14         m = svm_train(prob, param)
15         p_label, p_acc, p_val = svm_predict(y_train, x_train, m)
16         print("sVM: ", len(m.get_SV()))
```

```
1 if __name__ == '__main__':
2     args = {
3         'adaboost_train': "adaboost_letter.scale.tr",
4         'adaboost_test': "adaboost_letter.scale.t",
5         'C': 1,
6         'Q': 2,
7         'svm_train': "svm_letter.scale.tr",
8         'svm_test': "svm_letter.scale.t",
9     }
10
11     main(args)
```

No questions assigned to the following page.

```
1 === "2" versus "not 2"
2 Accuracy = 98.8667% (10381/10500) (classification)
3 SVM: 589
4 === "3" versus "not 3"
5 Accuracy = 99.3238% (10429/10500) (classification)
6 SVM: 368
7 === "4" versus "not 4"
8 Accuracy = 99.0381% (10399/10500) (classification)
9 SVM: 497
10 === "5" versus "not 5"
11 Accuracy = 98.5143% (10344/10500) (classification)
12 SVM: 643
13 === "6" versus "not 6"
14 Accuracy = 98.8762% (10382/10500) (classification)
15 SVM: 502
```

As a result, we should choose [d] as our solution.

P13

We use the same code above.

```
1 === "2" versus "not 2"
2 Accuracy = 98.8667% (10381/10500) (classification)
3 SVM: 589
4 === "3" versus "not 3"
5 Accuracy = 99.3238% (10429/10500) (classification)
6 SVM: 368
7 === "4" versus "not 4"
8 Accuracy = 99.0381% (10399/10500) (classification)
9 SVM: 497
10 === "5" versus "not 5"
11 Accuracy = 98.5143% (10344/10500) (classification)
12 SVM: 643
13 === "6" versus "not 6"
14 Accuracy = 98.8762% (10382/10500) (classification)
15 SVM: 502
```

As a result, we should choose [b] as our solution.

No questions assigned to the following page.

P14

```
1 import numpy as np
2 from libsvm.svmutil import *
3
4 def main(args):
5     choices = [0.01, 0.1, 1, 10, 100]
6     y_train, x_train = svm_read_problem(args['svm_train'])
7     y_test, x_test = svm_read_problem(args['svm_test'])
8
9     for type in range(len(y_train)):
10         y_train[type] = 1 if y_train[type] == args['classifier'] else 0
11     for type in range(len(y_test)):
12         y_test[type] = 1 if y_test[type] == args['classifier'] else 0
13
14     for choice in choices:
15         print(f'== choice: {choice}')
16         prob = svm_problem(y_train, x_train)
17         param = svm_parameter('-s 0 -t 2 -g {gamma} -c {C} -q'.format(C=choice, gamma=choice))
18         m = svm_train(prob, param)
19         p_label, p_acc, p_val = svm_predict(y_test, x_test, m)
```



```
1 if __name__ == '__main__':
2     args = {
3         'classifier': 7,
4         'gamma': 1,
5         'svm_train': "svm_letter.scale.tr",
6         'svm_test': "svm_letter.scale.t",
7     }
8
9     main(args)
```

```
1 === choice: 0.01
2 Accuracy = 95.48% (4774/5000) (classification)
3 === choice: 0.1
4 Accuracy = 95.48% (4774/5000) (classification)
5 === choice: 1
6 Accuracy = 98.58% (4929/5000) (classification)
7 === choice: 10
8 Accuracy = 99.6% (4980/5000) (classification)
9 === choice: 100
10 Accuracy = 99.46% (4973/5000) (classification)
```

As a result, we should choose [d] as our solution.

No questions assigned to the following page.

P15

We use the same code above, then update the `choices`, `param`, and the parameter of args. The updated lines show below.

```
1 choices = [0.1, 1, 10, 100, 1000]
2 param = svm_parameter('-s 0 -t 2 -g {gamma} -c {C} -q'
3                         .format(C=args['C'], gamma=choice))
4
5 if __name__ == '__main__':
6     args = {
7         'C': 0.1,
8         'classifier': 7,
9         'svm_train': "svm_letter.scale.tr",
10        'svm_test': "svm_letter.scale.t",
11    }
12
13 main(args)
```

```
1 === choice: 0.1
2 Accuracy = 95.48% (4774/5000) (classification)
3 === choice: 1
4 Accuracy = 95.48% (4774/5000) (classification)
5 === choice: 10
6 Accuracy = 95.98% (4799/5000) (classification)
7 === choice: 100
8 Accuracy = 95.48% (4774/5000) (classification)
9 === choice: 1000
10 Accuracy = 95.48% (4774/5000) (classification)
```

As a result, we should choose [c] as our solution.

No questions assigned to the following page.

P16

```
1 import numpy as np
2 import random
3 from libsvm.svmutil import *
4 from tqdm import trange
5
6 def main(args):
7     choices = [0.1, 1, 10, 100, 1000]
8     y_train, x_train = svm_read_problem(args['svm_train'])
9
10    for type in range(len(y_train)):
11        y_train[type] = 1 if y_train[type] == args['classifier'] else 0
12
13    gamma_count = [0, 0, 0, 0, 0]
14    for i in trange(args['repeat_time']):
15        sampling = random.sample(range(len(y_train)), args['sampling'])
16        training_id = [i for i in range((len(y_train))) if i not in sampling]
17        validation_id = [i for i in range((len(y_train))) if i in sampling]
18
19        min = 1
20        min_gamma = 0
21        for i, g in enumerate(choices):
22            prob = svm_problem([y_train[idx] for idx in training_id], [x_train[idx]
23                param = svm_parameter('-s 0 -t 2 -g {gamma} -c {C} -q'.format(C=args['C'
24                m = svm_train(prob, param)
25                p_label, p_acc, p_val = svm_predict([y_train[idx] for idx in validation
26                print(p_acc)
27                if p_acc[1] < min:
28                    min_gamma = i
29                    min = p_acc[1]
30                gamma_count[min_gamma] += 1
31                print("gamma_count:", gamma_count)
32                print("final gamma_count:", gamma_count)
```

Question assigned to the following page: [6.1](#)

```
1 if __name__ == '__main__':
2     args = {
3         'C': 0.1,
4         'classifier': 7,
5         'sampling': 200,
6         'svm_train': "svm_letter.scale.tr",
7         'svm_test': "svm_letter.scale.t",
8         'repeat_time': 500,
9     }
10
11     main(args)
```

```
1 final gamma_count: [305, 0, 195, 0, 0]
```

As a result, we should choose [a] as our solution.

P17

```
1 import math
2 import numpy as np
3 from libsvm.svmutil import *
4 from tqdm import trange
5
6 def adaboost(args, x, y):
7     u = np.ones((args['size'], )) / args['size']    # weights
8     E_in_u, alpha = np.zeros((args['T'],)), np.zeros((args['T'],))
9     s, i, theta = np.zeros((args['T'],)), np.zeros((args['T'],), dtype=int), np.zer
10    for t in trange(args['T']):
11        # get optimal parameters
12        E_in_u[t], s[t], i[t], theta[t] = decision_stump_multi(args, x, y, u)
13        diamond = np.sqrt((1-E_in_u[t])/E_in_u[t])
14        # update u, which are weights
15        for idx in range(len(y)):
16            if y[idx] != s[t] * sign(x[idx][i[t]] - theta[t]):
17                u[idx] *= diamond
18            else:
19                u[idx] /= diamond
20        alpha[t] = np.log(diamond)
21    return E_in_u, s, i, theta, alpha
```



Question assigned to the following page: [6.1](#)

```

1 def decision_stump(args, x, y, u):
2     # h(x) = s * sign (x - theta)
3     s = 1; theta = float('-inf'); error = float('inf')
4     sorted_x = np.array(sorted(x))
5     theta_list = np.array([float('-inf')] + [(sorted_x[i]+sorted_x[i+1])/2 for i
6
7         for theta_hypothesis in theta_list:
8             y_pos = np.where(x >= theta_hypothesis, 1, -1)
9             y_neg = np.where(x < theta_hypothesis, 1, -1)
10            error_pos = np.sum(u[np.where(y_pos != y)]) / np.sum(u)
11            error_neg = np.sum(u[np.where(y_neg != y)]) / np.sum(u)
12
13            if error_pos > error_neg:
14                if error_neg < error:
15                    error = error_neg
16                    theta = theta_hypothesis
17                    s = -1
18                else:
19                    if error_pos < error:
20                        error = error_pos
21                        theta = theta_hypothesis
22                        s = 1
23            return error, s, theta
24
25 def decision_stump_multi(args, x, y, u):
26     s = np.zeros((args['dimension'],))
27     theta = np.zeros((args['dimension'],))
28     error = np.zeros((args['dimension'],))
29
30     for dim in range(args['dimension']):
31         error[dim], s[dim], theta[dim] = decision_stump(args, x[:,dim], y, u)
32     i = np.argmin(error)
33     return error[i], s[i], i, theta[i]

```



Question assigned to the following page: [6.1](#)

```
1 def get_E_in_list(args, x, y, s, i, theta):
2     error_list = list()
3     for t in range(args['T']):
4         error = 0
5         for n in range(len(y)):
6             if(y[n] != (s[t] * np.sign(x[n][i[t]] - theta[t]))):
7                 error += 1
8             error_list.append(error/len(y))
9     return error_list
10
11 def predict_G(args, x, y, s, i, theta, alpha):
12     error = 0
13     for n in range(len(x)):
14         temp = 0
15         for alpha_n in range(len(alpha)):
16             temp += alpha[alpha_n] * (s[alpha_n] * np.sign(x[n][i[alpha_n]] - theta[alpha_n]))
17         guess = np.sign(temp)
18         if(y[n] != guess):
19             error += 1
20     return error/len(y)
21
22 def sign(value):
23     if value >= 0:
24         return 1
25     else:
26         return -1
```

Question assigned to the following page: [6.1](#)

```

1 def transformation(args, y, x):
2     transformed_x, transformed_y = list(), list()
3     # idx:value => value only
4     for n in range(len(y)):
5         if(y[n] == args['label_pos'] or y[n] == args['label_neg']):
6             transformed_y.append(y[n])
7             row = list()
8             for index, value in x[n].items():
9                 row.append(value)
10            transformed_x.append(row)
11    # one-versus-one: label "label_pos" versus label "label_neg"
12    for label in range(len(transformed_y)):
13        transformed_y[label] = 1 if transformed_y[label] == args['label_pos'] else
14    return np.array(transformed_y), np.array(transformed_x)
15
16 def main(args):
17     y_train, x_train = transformation(args, *svm_read_problem(args['adaboost_train']))
18     y_test, x_test = transformation(args, *svm_read_problem(args['adaboost_test']))
19
20     # hyper-parameters
21     args['dimension'] = x_train.shape[1]
22     args['size'] = len(x_train)
23     # adaboost and decision_stump
24     E_in_u, s, i, theta, alpha = adaboost(args, x_train, y_train)    # only return c
25     # P17, P18
26     E_in_list = get_E_in_list(args, x_train, y_train, s, i, theta)
27     print("min E_in(g_t): ", min(E_in_list))
28     print("max E_in(g_t): ", max(E_in_list))
29     # P19, P20
30     E_in_G = predict_G(args, x_train, y_train, s, i, theta, alpha)
31     E_out_G = predict_G(args, x_test, y_test, s, i, theta, alpha)
32     print("E_in(G): ", E_in_G)
33     print("E_out(G): ", E_out_G)
34     print("==Fuck you HTML==")

```

Question assigned to the following page: [6.1](#)

```
1 if __name__ == '__main__':
2     args = {
3         'adaboost_train': "adaboost_letter.scale.tr",
4         'adaboost_test': "adaboost_letter.scale.t",
5         'dimension': 16,
6         'label_pos': 11,
7         'label_neg': 26,
8         'size': 0,
9         'T': 1000,
10        'type': 1, # 1: best, 0: worst
11    }
12
13    main(args)
```

```
1 min E_in(g_t): 0.09846547314578005
2 max E_in(g_t): 0.571611253196931
3 E_in(G): 0.0
4 E_out(G): 0.002793296089385475
```

As a result, we should choose [a] as our solution.

P18

We use the same code above.

```
1 min E_in(g_t): 0.09846547314578005
2 max E_in(g_t): 0.571611253196931
3 E_in(G): 0.0
4 E_out(G): 0.002793296089385475
```

As a result, we should choose [c] as our solution.

P19

We use the same code above.

```
1 min E_in(g_t): 0.09846547314578005
2 max E_in(g_t): 0.571611253196931
3 E_in(G): 0.0
4 E_out(G): 0.002793296089385475
```

As a result, we should choose [a] as our solution.

No questions assigned to the following page.

P20

We use the same code above.

```
1 | min E_in(g_t): 0.09846547314578005
2 | max E_in(g_t): 0.571611253196931
3 | E_in(G): 0.0
4 | E_out(G): 0.002793296089385475
```

As a result, we should choose [a] as our solution.