Machine Learning Techniques Homework 5

Wu, Bo-Run (r08942073)

25th December 2020

Question 1.

Solution: [d]

After apply the polynomial transform to the three example data, the $\mathbf{X}$ becomes $[x_1 = \begin{bmatrix}1\\-2\\4\end{bmatrix}, x_2 = \begin{bmatrix}1\\0\\0\end{bmatrix}, x_3 = \begin{bmatrix}1\\2\\4\end{bmatrix}]$ and we can get the following functions $\begin{cases} -(w_1 - 2w_2 + 4w_3 + b) \geq 1 & (1) \\ +(w_1 + b) \geq 1 & (2) \\ -(w_1 + 2w_2 + 4w_3 + b) \geq 1 & (3) \end{cases}$, by applying the $y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$ By combining the equation above, we will get $\begin{cases} (1) + (2) & w_2 - 2w_3 \geq 1 \\ (2) + (3) & -w_2 - 2w_3 \geq 1 \\ (1) + 2 * (2) + (3) & -2w_3 \geq 1 \end{cases}$.

Therefore, the region of $\mathbf{w}$ for each elements will be $\begin{cases} w_1 \geq 1 - b \\ w_2 = 0 \\ w_3 \leq -\frac{1}{2} \end{cases}$. To minimize the $\frac{1}{2}\mathbf{w}^T\mathbf{w}$, we will get the optimal solutions $w_1^* = 0,\ w_2^* = 0,\ w_3^* = -\frac{1}{2},\ b^* = 1$.

Question 2.

Solution: [b]

From Question 1, we can know the margin with the equation $margin(b, w) = \frac{1}{||w||} = 2$

Question 3.

Solution: [e]

Because for $x_1,\ x_2\ ...\ x_M$ the $y_{1...M} = +1$, and for $x_{M+1},\ x_{M+2}\ ...\ x_N$ the $y_{M+1...N} = -1$. We can know that the boundary is between x_M and x_{M+1}, and the largest margin will be $\frac{1}{2}(x_{M+1} - x_M)$.

Question 4.

Solution: [a]

For $\rho = 0$, we can get 4 dichotomies. Then for $\rho > 0$, we can get at least get 2 dichotomies, because there is no boundary limits to SVM. Therefore, other 2 dichotomies happen in the place which two points distance is greater than 2ρ, causing the probability for two points to be $(1-2\rho)^2$. The expectation will be $2+2(1-2\rho)^2$.

Question 5.

Solution: [c]

From page 5 of Lecture 202, we can know that the Lagrange function is objective plus Lagrange multipliers multiply by constraints. Follow page 9, 10, 11 and 13 of Lecture 202 to do the simplification. The objective in the original function $-\sum_{n=1}^{N} \alpha_n$ will become $-(\sum_{n=1}^{N} \rho_+ [\![y_n = +1]\!]\alpha_n + \sum_{n=1}^{N} \rho_- [\![y_n = -1]\!]\alpha_n)$

Question 6.

Solution: [e]

We can consider the boundary condition $y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$ for both α. As we can see from the equation that we can scale the w and b to reach the boundary condition. Furthermore, we can see the ρ_+ and ρ_- as the two sides of the margin to a even margin of $\frac{\rho_+ + \rho_-}{2}$. With the boundary condition we can scale the original w and b with $\frac{2}{\rho_+ + \rho_-}$, we can get that $\alpha^* = \frac{\rho_+ + \rho_-}{2}\alpha$

Question 7.

Solution: [d]

Valid kernel must be positive semi-definite, which means that the eigenvalues must be all non-negative. Assume we have a kernel $\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, and apply it to the option (d) which will give us $\log_2\left(\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\right) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

Question 8.

Solution: [c]

Squared distance between two examples $\mathbf{x}$ and $\mathbf{x}'$ is $||\phi(\mathbf{x}) - \phi(\mathbf{x}')||^2$ in the **Z**-space. The kernel function is $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}') = exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$. Our goal is to maximize $||\phi(\mathbf{x}) - \phi(\mathbf{x}')||^2 = \phi(\mathbf{x})^T\phi(\mathbf{x}) - 2\phi(\mathbf{x})\phi(\mathbf{x}') + \phi(\mathbf{x})^T\phi(\mathbf{x}) = 2 - exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$, which upper bound of this equation is 2.

Question 9.

Solution: [d]

$E_{in}(\hat{h}) = 0$ means that all the point classify correctly. We can see it as $\left(\sum_{n=1}^N y_n^2 \alpha_n K(\mathbf{x}_n, \mathbf{x}) + b\right) \geq 0$. With the conditions listed in the problem the equation become $\sum_{n=1}^N y_n^2 K(\mathbf{x}_n, \mathbf{x}) \geq 0 \Rightarrow \sum_{n=1}^N K(\mathbf{x}_n, \mathbf{x}) \geq 0$ and with a tighter lower bound. We substitute $K(\mathbf{x}_n, \mathbf{x})$ with $exp(-\gamma||\mathbf{x}_n - \mathbf{x}||^2)$ get $\sum_{n=1}^N exp(-\gamma||\mathbf{x}_n - \mathbf{x}||^2) < 1 \Rightarrow (N-1)exp(-\gamma\xi^2) < 1 \Rightarrow \gamma > \frac{\ln N-1}{\xi^2}$

Question 10.

Solution: [c]

Because we assume $\mathbf{w}_t$ as a linear combination of $\phi(\mathbf{x}_n)i_{n=1}^N$,which $\mathbf{w}_t = \sum_{n=1}^N \alpha_{t,n}\phi(\mathbf{x}_n)$ and the algorithm updates $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$ when the current $\mathbf{w}_t$ makes a mistake $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\phi(\mathbf{x}_{n(t)})$. We can substitute $\mathbf{w}_t$ as $\sum_{n=1}^N \alpha_{t,n}\phi(\mathbf{x}_n)$ and get $\mathbf{w}_{t+1} \leftarrow \sum_{n=1}^N \alpha_{t,n}\phi(\mathbf{x}_n) + y_{n(t)}\phi(\mathbf{x}_{n(t)})$. The α_t updates when makes a mistake $\alpha_{t+1} \leftarrow \alpha_t + y_{n(t)}$

Question 11.

Solution: [a]

From Problem 10, we can get $\mathbf{w}_t = \sum_{n=1}^N \alpha_{t,n}\phi(\mathbf{x}_n) \rightarrow \mathbf{w}_t^T = \sum_{n=1}^N \alpha_{t,n}\phi(\mathbf{x}_n)^T$ and we multiply $\phi(\mathbf{x})$ get $\mathbf{w}_t^T\phi(\mathbf{x}) = \sum_{n=1}^N \alpha_{t,n}\phi(\mathbf{x_n})^T\phi(\mathbf{x}) = \sum_{n=1}^N \alpha_{t,n}K(\mathbf{x}_n, \mathbf{x})$

Question 12.

Solution: [b]

From complementary slackness, we can know that $\alpha_n(1-\xi_n-y_n(\mathbf{w}^T\mathbf{z}_n+b))=0$ and $(C-\alpha_n)\xi_n=0$. By assumption, we know that $\alpha_n=C$ and substitute it into the complementary slackness get $1-\xi_n-y_n(\mathbf{w}^T\mathbf{x}_n+b)=0 \rightarrow b=\frac{1-\xi_n}{y_n}-\sum_{m=1}^{N} y_m\alpha_m K(x_n,x_m) \leq \frac{1}{y_n}-\sum_{m=1}^{N} y_m\alpha_m K(x_n,x_m)$

Question 13.

Solution: [e]

We get the Lagrange function $\mathcal{L}((b,w,\xi),\alpha)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{n=1}^{N} C\xi_n^2+\sum_{n=1}^{N}\alpha_n(1-\xi_n-y_n(\mathbf{w}^T z_n+b))$. Then, we simplify the equation by $\begin{cases}\frac{\partial\mathcal{L}}{\partial\xi_n}=2C\xi_n-\alpha_n=0\\ \frac{\partial\mathcal{L}}{\partial b}=\sum_{n=1}^{N}-\alpha_n y_n=0\\ \frac{\partial\mathcal{L}}{\partial w_i}=w_i-\alpha_n y_n x_{n,i}=0\end{cases}$ get $\mathcal{L}((b,w,\xi),\alpha)=\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\alpha_n\alpha_m y_n y_m z_n z_m+\sum_{n=1}^{N}\frac{1}{4C}\alpha_n^2-\sum_{n=1}^{N}\alpha_n=\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\alpha_n\alpha_m y_n y_m(K(\mathbf{x_n},\mathbf{x_m})+\frac{1}{2C}[\![n=m]\!])$

Question 14.

Solution: [e]

From Question 13, we can get $\xi_n^*=\frac{\alpha_n^*}{2C}$

Question 15.

Solution: [d]

```python
import numpy as np
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem


def main():
    y_train, x_train = svm_read_problem('../Data/hw5/satimage.scale')

    y_train = [temp == 3.0 for temp in y_train]

    model = svm_train(y_train, x_train, '-t 0 -c 10')

    sv_coef = np.array(model.get_sv_coef())
    sv_indices = [index-1 for index in model.get_sv_indices()]

    _, x_support = svm_read_problem('../Data/hw5/satimage.scale', return_scipy=True)

    x_support = x_support.toarray()[sv_indices]

    print(f'||w|| is {np.linalg.norm(x_support.T @ sv_coef)}')


if __name__ == '__main__':
    main()
```

Question 16.

Solution: [b]

```
import numpy as np
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem


def main():
    y_train, x_train = svm_read_problem('../Data/hw5/satimage.scale')

    for i in range(5):

        print(f'Label␣{i+1}')

        temp_y_train = [temp == i+1 for temp in y_train]

        model = svm_train(temp_y_train, x_train, '-t␣1␣-d␣2␣-g␣1␣-r␣1␣-c␣10')

        p_label, p_acc, p_val = svm_predict(temp_y_train, x_train, model)


if __name__ == '__main__':
    main()
```

Question 17.

Solution: [c]

```
import numpy as np
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem


def main():
    y_train, x_train = svm_read_problem('../Data/hw5/satimage.scale')

    for i in range(5):

        print(f'Label␣{i+1}')

        temp_y_train = [temp == i+1 for temp in y_train]

        model = svm_train(temp_y_train, x_train, '-t␣1␣-d␣2␣-g␣1␣-r␣1␣-c␣10')

        p_label, p_acc, p_val = svm_predict(temp_y_train, x_train, model)
```

```
if __name__ == '__main__':
    main()
```

Question 18.

Solution: [d, e]

```
import numpy as np
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem


def main():
    y_train, x_train = svm_read_problem('../Data/hw5/satimage.scale')

    y_train = [temp == 6.0 for temp in y_train]

    y_test, x_test = svm_read_problem('../Data/hw5/satimage.scale.t')

    y_test = [temp == 6.0 for temp in y_test]

    for c in [0.01, 0.1, 1, 10, 100]:

        print(f'----------')
        print(f'C {c}')

        model = svm_train(y_train, x_train, f'-t 2 -g 10 -c {c}')

        p_label, p_acc, p_val = svm_predict(y_test, x_test, model)

        print(f'Eout = {p_acc[1]}')


if __name__ == '__main__':
    main()
```

Question 19.

Solution: [b]

```
import numpy as np
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem


def main():
    y_train, x_train = svm_read_problem('../Data/hw5/satimage.scale')

    y_train = [temp == 6.0 for temp in y_train]

    y_test, x_test = svm_read_problem('../Data/hw5/satimage.scale.t')
    y_test = [temp == 6.0 for temp in y_test]

    for g in [0.1, 1, 10, 100, 1000]:

        print(f'----------')
        print(f'Gamma␣{g}')

        model = svm_train(y_train, x_train, f'-t␣2␣-g␣{g}␣-c␣0.1')

        p_label, p_acc, p_val = svm_predict(y_test, x_test, model)

        print(f'Eout␣=␣{p_acc[1]}')


if __name__ == '__main__':
    main()
```

Question 20.

Solution: [b]

```
import numpy as np
from tqdm import tqdm
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
from concurrent.futures import ProcessPoolExecutor, ThreadPoolExecutor

ITER = 1000

def one_iter(y, x):
    np.random.seed()
    shuffle = np.random.permutation(len(y))

    Evals = list()
    for g in [0.1, 1, 10, 100, 1000]:
        model = svm_train(y[shuffle[200:]], x[shuffle[200:]], f'-t␣2␣-g␣{g}␣-c␣0.1␣-q')

        p_label, p_acc, p_val = svm_predict(y[shuffle[:200]], x[shuffle[:200]], model, '-q

        Evals.append((p_acc[1], g))

    return min(Evals)

def main():
    y_train, x_train = svm_read_problem('../Data/hw5/satimage.scale', return_scipy=True)

    y_train = y_train == 6.0

    gammas = {g:0 for g in [0.1, 1, 10, 100 ,1000]}
    with ProcessPoolExecutor() as executor:
        for _, idx in tqdm(executor.map(one_iter, [y_train]*ITER, [x_train]*ITER), total=IT
            gammas[idx] += 1

    print(gammas)

if __name__ == '__main__':
    main()
```