1. d.

$$X=\begin{bmatrix}-2\\0\\2\end{bmatrix} \quad y=\begin{bmatrix}-1\\1\\-1\end{bmatrix} \quad \phi(X)=\begin{bmatrix}1&-2&4\\1&0&0\\1&2&4\end{bmatrix}$$

$2w_1-4w_2-b\geq 1$ $\quad 4w_1\geq 0$ $\quad -4w_2\geq 2$

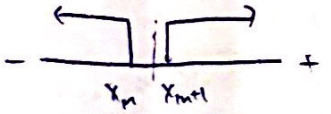
$b\geq 1$ $\quad w_1\geq 0$ $\quad w_2\geq -\frac{1}{2}$

$-2w_1-4w_2-b\geq 1$ #

2. b.

$$\frac{1}{\|w\|}=\frac{1}{\sqrt{w_1^2+w_2^2}}=\frac{1}{\sqrt{\frac{1}{4}}}=2 \ \#$$

3. e



$+ \quad \frac{1}{2}(x_{n+1}-x_n)$

5. C

$\frac{1}{2}w^Tw+\sum a_n(\rho-y_n(w^Tz_n+b))$ $\quad \frac{\partial L(b,w,a)}{\partial b}=0=-\sum y_n a_n$

$\frac{1}{2}w^Tw+\sum a_n(\rho-y_n(w^Tz_n))$ $\quad y_n(w^Tz+b)\geq \rho$

$-\frac{1}{2}\sum\|a_n y_n z_n\|^2+\sum \rho a_n$ $\quad \frac{\partial L(b,v,a)}{\partial w}=0=w_i-\sum a_n y_n z_n$

$w=\sum a_n y_n z_n$

$\min \frac{1}{2}\sum\|a_n y_n z_n\|^2-\sum \rho a_n$

6. a.

$$a\begin{cases}w=\sum a_n y_n z_n\\ b=y_n-w^Tz\end{cases}$$

7 d.

$k_{y_2} K(x, x')$ 非半正定矩陣

10. C

$w_{t+1} \leftarrow \partial_t \phi(x) + \gamma \phi(x)$

$w_{t+1} \leftarrow \phi(x)(\partial_t + \gamma)$

11. C

$w\phi(x)$

$= \partial_n y_n \phi(x_n)\phi(x)$

$= y_n \partial_n k(x_n, x)$

12. A.

$\partial_n = C$

$\partial_n (1 - \xi - \underbrace{y_n (w^T z_n + b)}_{\text{"0}}) = 0$

$(C - \partial_n)\xi = 0$

$\xi = 1 - y_n (w^T z_n + b)$ → violation amount

15~20　X b e d b a

Read data

```
def read_data(file):
    f = open(file)
    data = f.read()
    f.close()

    all_data = []
    all_label = []
    for line in data.split("\n"):
        i = 0
        line_data = []
        for sub_line in line.split(" "):
            for value in sub_line.split(":"):
                if i == 0:
                    if value != "":
                        all_label.append(int(value))
                elif i != 0 and i % 2 == 0:
                    if value != "":
                        line_data.append(float(value))
                i += 1
        if line_data != []:
            all_data.append(line_data)
    return all_data, all_label


def redefine_label(label, class_number):
    new_label = []
    for i in range(len(label)):
        if label[i] == class_number:
            new_label.append(1)
        else:
            new_label.append(-1)
    return new_label


train_data, train_label = read_data("./train.txt")
test_data, test_label = read_data("./test.txt")
```

16, 17

```
train_label = redefine_label(train_label, 2)
test_label = redefine_label(test_label, 2)
m = svm_train(train_label, train_data, "-c 10 -t 1 -
d 2")
p_label, p_acc, p_val = svm_predict(train_label, train_
data, m)
```

18

```
best_Eout = 100
best_c = 0
Cs = [0.01, 0.1, 1, 10, 100]
train_label = redefine_label(train_label, 6)
test_label = redefine_label(test_label, 6)
for c in Cs:
    m = svm_train(train_label, train_data, "-c %f -
t 2 -g 10" % c)
    p_label, p_acc, p_val = svm_predict(test_label, tes
t_data, m)
    print(p_acc[0])
    if 100 - p_acc[0] < best_Eout:
        best_Eout = 100 - p_acc[0]
        best_c = c
print("best:", best_Eout, best_c)
```

19

```
best_Eout = 100
best_gamma = 10000
gammas = [0.1, 1, 10, 100, 1000]
train_label = redefine_label(train_label, 6)
test_label = redefine_label(test_label, 6)
for gamma in gammas:
    m = svm_train(train_label, train_data, "-c 0.1 -
t 2 -g %f" % gamma)
    p_label, p_acc, p_val = svm_predict(test_label, tes
t_data, m)
    print(p_acc[0])
    if 100 - p_acc[0] < best_Eout:
        best_Eout = 100 - p_acc[0]
        best_gamma = gamma
```

```
print("best:", best_Eout, best_gamma)
```

20

```
def train_val_split(data, label, val_index):
    train_data = []
    train_label = []
    val_data = []
    val_label = []
    for i in range(len(data)):
        if i in val_index:
            val_data.append(data[i])
            val_label.append(label[i])
        else:
            train_data.append(data[i])
            train_label.append(label[i])
    return train_data, train_label, val_data, val_label


train_label = redefine_label(train_label, 6)
test_label = redefine_label(test_label, 6)
all_index = np.arange(0, len(train_data))
val_index = random.sample(range(0, len(train_data)), 20
0)
train_data, train_label, val_data, val_label = train_va
l_split(
    train_data, train_label, val_index)

gammas = [0.1, 1, 10, 100, 1000]
smallest_gamma = []

best_Eout = 100
best_gamma = 10000
for epoch in range(10):
    print(epoch)
    for gamma in gammas:
        m = svm_train(train_label, train_data, "-
c 0.1 -t 2 -g %f" % gamma)
        p_label, p_acc, p_val = svm_predict(val_label,
val_data, m)
        if 100 - p_acc[0] <= best_Eout:
```

```
            if gamma < best_gamma:
                best_gamma = gamma
    smallest_gamma.append(best_gamma)
print(smallest_gamma)
counts = np.bincount(smallest_gamma)
print(np.argmax(counts))
```