1. D

for misclassified case:

we know that $(w^T x_n + b)$ is classified wrong $\Rightarrow y_n \cdot (w^T x_n + b) < 0$ (∵ different sign means wrong)

$\because \xi_n \geq 1 - y_n(w^T x_n + b) \quad \because \xi_n > 1$ (for wrong classified case)

$\xi_n \leq 1$ (for right cases)

number of wrong cases: $\sum_{n=1}^{N} [\![\xi_n^* > 1]\!] \leq \sum_{n=1}^{N} \xi_n^*$

∵ wrong cases' $\xi_n^* > 1$

∴ add up wrong cases' ξ_n^* will be larger than the number of wrong cases

$\Rightarrow$ we just have to make sure the element in the $\sum$ will still be larger than 1 for the wrong cases, then it will still be an upper bound

$(\xi_n^* > 1)$

- $\frac{\xi_n^*}{2} > \frac{1}{2} \neq 1 \quad \Rightarrow$ no
- $\sqrt{\xi_n^*} > \sqrt{1} = 1 \quad \Rightarrow$ yes $\quad \Rightarrow$ 4
- $\xi_n^* > 1 \quad \Rightarrow$ yes
- $\lfloor \xi_n^* \rfloor > \lfloor 1 \rfloor \Rightarrow$ yes
- $\log_2(1+\xi_n^*) > \log_2(1+1) > 1 \Rightarrow$ yes

2. D

for soft-margin SVM.

we have a condition:

$$y_n(w^T z_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad (\text{optimal solution } \alpha^* \text{ that } \alpha_n^* = C)$$

$$y_n\left(\sum_{m=1}^{N} y_m \alpha_m^* K(x_n, x_m) + b\right) \geq 1 - \xi_n$$

" every example is bounded SV

$$\because y_n\left(\sum_{m=1}^{N} y_m \alpha_m^* K(x_n, x_m) + b\right) = 1 - \xi_n, \ \because \xi_n \geq 0$$

$$\exists \ y_n\left(\sum_{m=1}^{N} y_m \alpha_m^* K(x_n, x_m) + b\right) \leq 1$$

$$\begin{cases} y_n = 1 \Rightarrow & b \leq 1 - \sum_{m=1}^{N} y_m \alpha_m^* K(x_n, x_m) \quad (\text{upper bound}) \\ y_n = -1 \Rightarrow & b \geq -1 - \sum_{m=1}^{N} y_m \alpha_m^* K(x_n, x_m) \quad (\text{lower bound}) \end{cases}$$

$\Rightarrow$ 因為至少有一個 $y_n < 0$

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

3、A

Apply Lagrange Multiplier on P_2

$$L = \frac{1}{2}w^T w + C\cdot\sum_{n=1}^{N}\xi_n^2 + \sum_{n=1}^{N}\alpha_n\cdot(1-\xi_n - y_n(w^T z_n + b))$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N}\alpha_n y_n = 0$$

$$\frac{\partial L}{\partial w_i} = 0 \Rightarrow w = \sum_{n=1}^{N}\alpha_n y_n z_n$$

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow \xi_n = \frac{\alpha_n}{2C}$$

將求得的 constraint，w, ξ_n 代回 Lagrange dual

$$\Rightarrow \max_{0\le\alpha_n\le C}\left(\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(x_n,x_m) + \sum_{n=1}^{N}\xi_n\left(C\cdot\frac{\alpha_n}{2C} - \alpha_n\right) + \sum_{n=1}^{N}\alpha_n - \sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m k(x_n,x_m)\right)$$

$$\Rightarrow \max_{0\le\alpha_n\le C}\left(-\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(x_n,x_m) + \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\alpha_n\xi_n\right)$$

$\because \alpha_n \ge 0$

$\therefore \max \alpha_n = \max -\frac{1}{\alpha_n}$

$$\Rightarrow \max_{0\le\alpha_n\le C}\left(-\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(x_n,x_m) + \sum_{n=1}^{N}\alpha_n + \frac{1}{2}\sum_{n=1}^{N}\frac{1}{\alpha_n}\xi_n\right)$$

$$\Rightarrow \max_{0\le\alpha_n\le C}\left(-\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(x_n,x_m) + \sum_{n=1}^{N}\alpha_n + \frac{1}{2}\sum_{n=1}^{N}\frac{1}{2C}\right)$$

$$\Rightarrow \max_{0\le\alpha_n\le C}\left(-\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m\left[K(x_n,x_m) + \frac{1}{2C}[n=m]\right] + \sum_{m=1}^{N}\alpha_n\right)$$

$\max(k) = -\min(-k)$

$$\Rightarrow \min_{\alpha}\left(\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m\left[K(x_n,x_m) + \frac{1}{2C}[n=m]\right] - \sum_{m=1}^{N}\alpha_n\right)$$

$$\Rightarrow \xi^* = \frac{1}{2C}\cdot\alpha^* \quad \text{※}$$

4. E

i: d個

s: 2個

θ: $\frac{2R-2L}{2} = (R-L)$個

∴ $\Phi_{ds}(x)$會有 $d \cdot 2 \cdot (R-L) = 2d(R-L)$ 項

假設 $\Phi_{ds}(x)$ 和 $\Phi_{ds}(x')$ 每一項的 sign 都相同 ∴ $K_{ds}(x, x') = 2d(R-L)$

若其中有 sign 不同 ∴ 則代表 x_i 和 x'_i 不同，然後當 θ 在 x_i 和 x'_i 之間會 sign 不同

∴ 共會有 $\frac{|x_i - x'_i|}{2}$ 個 θ 造成 sign 不同

∴ 每個 θ 又會配上 2個 s ∴ 在 i 上會有 $|x_i - x'_i|$ 個不同

∴ 對整個 x 和 x' 而言 ∴ 會有 $|x - x'|_1$ 個 sign 不同

∴ 我們起初假設 sign 相同，所以大家都算成 1，但若 sign 不同，應該算成 -1

∴ 會需要減 $2 \cdot |x - x'|_1$ 才能讓 $K_{ds}(x, x')$ 是正確的 $(1 - 2 = -1)$

∴

$$K_{ds}(\Phi_{ds}(x), \Phi_{ds}(x')) = 2d(R-L) - 2\|x - x'\|_1$$

5. B

uniform blending ∋取 sign

∋需要超過半數才能決定

∋ want tightest upper bound ∋盡可能的讓錯誤分散

∋共 2M+1 個 g_t

∋至少要 $\lceil \frac{2M+1}{2} \rceil = M+1$ 才能讓錯誤有效發生

∋能夠讓錯誤都成功達成錯誤，就是把所有錯誤都分成 M+1 個一組

看能分成幾組，就代表 G(x) 最多能產生幾個錯誤

let N = test size

∋ total 錯 $= \sum_{m=1}^{2M+1} N \cdot e_t$ ∋可以分 $\frac{\sum_{m=1}^{2M+1} N \cdot e_t}{M+1}$ 組，造成 $\frac{\sum_{m=1}^{2M+1} N \cdot e_t}{M+1}$ 個錯 for G(x)

∋ $E_{out} = \frac{\frac{1}{M+1} \cdot \sum_{m=1}^{2M+1} N \cdot e_t}{N} = \frac{1}{M+1} \sum_{m=1}^{2M+1} e_t$ ※

6. B

$1 - \frac{P_{N'}^{1126}}{1126^{N'}} \geq 0.95$

$P_{N'}^{1126}$：不能重複選，每次看前面剩下的人選再選一個

$1126^{N'}$：人可以重複選，每次都有 1126 個 choice

↳不重複的机率

∋ $\frac{P_{N'}^{1126}}{1126^{N'}} \leq 0.25$

(a) $N'=54$ ∋ $0.2752 > 0.25$

(b) $N'=56$ ∋ $0.2492 < 0.25$

(c) $N'=58$ ∋ $0.2248 < 0.25$

(d) $N'=60$ ∋ $0.2021 < 0.25$

∋smallest N' from options

$= 56$ ※

7. C

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

$$\Rightarrow u_n(y_n - w^T x_n)^2 = (\tilde{y}_n - w^T \tilde{x}_n)^2$$

$$\Rightarrow \tilde{y}_n = \sqrt{u_n}\, y_n \quad ; \quad \tilde{x}_n = \sqrt{u_n}\, x_n$$

8. E

(a) $0.5(1-1-0) + 0.5(1-0.25-0.25) = 0.25$

(b) $0.1(1-0.64-0.04) + 0.3(1-0.75^2-0.25^2) = 0.3365$

(c) $0.9(1-0.49-0.09) + 0.1(1-0-1) = 0.378$

(d) $0.8(1-0.64-0.04) + 0.2(1-0.81-0.01) = 0.292$

(e) $0.8(1-0.81-0.01) + 0.2(1-0.81-0.01) = 0.18$

$\Rightarrow$ best branch $\Rightarrow$ minimum $|D_c \text{ with } h| \cdot \text{Impurity}(D_c) \Rightarrow$ (e)

9. D

$$U_t = \sum_{n=1}^{N} u_n^{(t)}$$

$$U_1 = \sum_{n=1}^{N} \frac{1}{N} = 1$$

$$U_2 = \underbrace{N\cdot(\varepsilon_1)}_{\text{錯的數量}} \cdot \frac{1}{N} \cdot \sqrt{\frac{\varepsilon_1}{1-\varepsilon_1}} + \underbrace{N\cdot(1-\varepsilon_1)}_{\text{對的量}} \cdot \frac{1}{N} \cdot \sqrt{\frac{1-\varepsilon_1}{\varepsilon_1}} = \frac{2(1-\varepsilon_1)\cdot\varepsilon_1}{\sqrt{(1-\varepsilon_1)\,\varepsilon_1}} = 2\sqrt{(1-\varepsilon_1)\,\varepsilon_1}$$

$$U_3 = N\cdot(\varepsilon_2)\cdot\left[\frac{N}{1+N\cdot\varepsilon_1}\cdot\sqrt{\frac{\varepsilon_1}{1-\varepsilon_1}} + \frac{N}{1+N\cdot\varepsilon_1}\sqrt{\frac{1-\varepsilon_1}{\varepsilon_1}}\right]\cdot\sqrt{\frac{\varepsilon_2}{1-\varepsilon_2}}$$

$$+ N\cdot(1-\varepsilon_2)\left[\frac{N}{1+N\cdot\varepsilon_1}\cdot\sqrt{\frac{\varepsilon_1}{1-\varepsilon_1}} + \frac{N}{1+N\cdot\varepsilon_1}\cdot\sqrt{\frac{1-\varepsilon_1}{\varepsilon_1}}\right]\cdot\sqrt{\frac{1-\varepsilon_2}{\varepsilon_2}}$$

$$= \sqrt{\varepsilon_2(1-\varepsilon_2)}\cdot 4\cdot\sqrt{(1-\varepsilon_1)\varepsilon_1} = 2^2\cdot\sqrt{\varepsilon_2(1-\varepsilon_2)}\cdot\sqrt{\varepsilon_1(1-\varepsilon_1)}$$

;

so we cand find a pattern

$$\text{ans } U_T = 2^T\cdot\prod_{t=1}^{T}\sqrt{\varepsilon_t(1-\varepsilon_t)}$$ ※

10. B

gradient boost:

if α_t is one-variable linear regression optimal

if η is too, since description let α_t be the steepest η

$$\exists \min_{\eta} E_{in}(\eta) = \frac{1}{N}\sum_{n=1}^{N}(y_n - s_n - \eta g_t(x_n))^2$$

令 $\frac{dE_{in}}{d\eta} = 0$ to get optimal η

$$\exists \frac{1}{N}\sum_{n=1}^{N} 2\cdot(y_n - s_n - \eta\cdot g_t(x_n))\cdot(-g_t(x_n)) = 0$$

$$\exists \eta = \frac{\sum_{n=1}^{N}(y_n - s_n)\cdot g_t(x_n)}{\sum_{n=1}^{N}[g_t(x_n)]^2}$$

$s_n = s_n + \alpha_t \cdot g_t(x_n)$

$= s_n + \eta \cdot g_t(x_n)$

$$\exists \sum_{n=1}^{N}(s_n - y_n)g_t(x_n) = \sum_{n=1}^{N}(s_n + \eta g_t(x_n) - y_n)\cdot g_t(x_n)$$

$$= \sum_{n=1}^{N}(s_n - y_n)g_t(x_n) + \eta\sum_{n=1}^{N}[g_t(x_n)]^2$$

$$= \sum_{n=1}^{N}(s_n - y_n)g_t(x_n) + \frac{\sum_{n=1}^{N}(y_n - s_n)g_t(x_n)}{\sum_{n=1}^{N}[g_t(x_n)]^2}\cdot\sum_{n=1}^{N}[g_t(x_n)]^2$$

$$= \sum_{n=1}^{N}(s_n - y_n)g_t(x_n) + -1\cdot\sum_{n=1}^{N}(s_n - y_n)g_t(x_n)$$

$$= 0$$

—— ※

Q11 - Q16

```
from libsvm.svmutil import *
import numpy as np
from numpy import loadtxt
```

```
file = open('train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = []
    numbers.append(int(number_strings[0]))
    for i in range(1, 17):
      sub = number_strings[i].split(':')
      numbers.append(float(sub[1]))
    data.append(numbers)

x = np.array(data)
y_tr = x[:, 0]
x_tr = x[:, 1:]

file = open('test.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = []
    numbers.append(int(number_strings[0]))
    for i in range(1, 17):
      sub = number_strings[i].split(':')
      numbers.append(float(sub[1]))
    data.append(numbers)

x = np.array(data)
y_test = x[:, 0]
x_test = x[:, 1:]
```

```
def err(predict, ans):
  sum = 0
  num = len(predict)
  for i in range(0, num):
    if(int(predict[i]) != int(ans[i])):
      sum += 1
  return float(float(sum)/float(num))
```

Q11 C

```
#Q11
import math
pk = 1
tr = np.copy(y_tr)
tr[tr != pk] = -1

prob = svm_problem(tr, x_tr)
param = svm_parameter('-s 0 -t 0 -c 1')
model = svm_train(prob, param)
coef = model.get_sv_coef()
SV = model.get_SV()
W = np.zeros(16)
for i in range(0, len(coef)):
  for j in range(0, len(W)):
    #print(coef[i][0])
    #print(SV[i][j+1])
    W[j] += coef[i][0]*SV[i][j+1]

length = 0
for i in W:
  length += i*i
print(math.sqrt(length))
```

output: 6.309673609961579

Q12 D Q13 B

```
#Q12 #Q13
pk = [2, 3, 4, 5, 6]
Ein = []
nSV = []
for i in pk:
  tr = np.copy(y_tr)
  tr[tr != i] = -1

  prob = svm_problem(tr, x_tr)
  param = svm_parameter('-s 0 -t 1 -d 2 -c 1 -r 1 -g 1')
  model = svm_train(prob, param)
  p_label, p_accu, p_val = svm_predict(tr, x_tr, model)
  num = model.get_nr_sv()
  Ein.append((err(p_label, tr), i))
  nSV.append((num, i))

max = __builtins__.max
print(Ein)
print(max(Ein))
print(nSV)
print(min(nSV))
```

get maxEin = 0.014857142857142857 with class: 5

get minnSV = 368 with class: 3

Q14 D

```
#Q14
C = [0.01, 0.1, 1, 10, 100]
Eout = []

tr = np.copy(y_tr)
tr[tr != 7] = -1
test = np.copy(y_test)
test[y_test != 7] = -1

for i in C:
  prob = svm_problem(tr, x_tr)
  param = svm_parameter('-s 0 -t 2 -g 1 -c '+ str(i))
  model = svm_train(prob, param)
  p_label, p_accu, p_val = svm_predict(test, x_test, model)
  Eout.append((err(p_label, test), i))

print(Eout)
print(min(Eout))
```

get minEout = 0.004, with C = 10

Q15 C

```
#Q15
gamma = [0.1, 1, 10, 100, 1000]
Eout = []

tr = np.copy(y_tr)
tr[tr != 7] = -1
test = np.copy(y_test)
test[y_test != 7] = -1

for i in gamma:
  prob = svm_problem(tr, x_tr)
  param = svm_parameter('-s 0 -t 2 -c 0.1 -g '+str(i))
  model = svm_train(prob, param)
  p_label, p_accu, p_val = svm_predict(test, x_test, model)
  Eout.append((err(p_label, test), i))

print(Eout)
print(min(Eout))
```

get minEout = 0.0402, with gamma = 10

Q16 A

```
#Q16
from sklearn.model_selection import train_test_split
import statistics
from statistics import mode

gamma = [0.1, 1, 10, 100, 1000]
choose = []
print(len(gamma))

tr = np.copy(y_tr)
tr[tr != 7] = -1
tr[tr == 7] = 1

for j in range(0, 500):
  x_trtr, x_tt, y_trtr, y_tt = train_test_split(x_tr, tr, test_size=200)
  mini = 2
  tar = 0
  print(j)
  for i in range(0, len(gamma)):
    #print(len(gamma))
    print(gamma[i])
    prob = svm_problem(y_trtr, x_trtr)
    param = svm_parameter('-s 0 -t 2 -c 0.1 -g '+str(gamma[i]))
    model = svm_train(prob, param)
    p_label, p_accu, p_val = svm_predict(y_tt, x_tt, model)
    Eval = err(p_label, y_tt)
    if(Eval < mini):
      mini = Eval
      tar = gamma[i]
  choose.append(tar)

print(len(choose))
print(choose)
print(mode(choose))
```

get appear most gamma = 0.1

Q17-Q20

```
#data preprocess
import math
y_ada_tr = []
x_ada_tr = []
index_tr = []
y_ada_test = []
x_ada_test = []
index_test = []

cnt = 0
for i in range(0, len(y_tr)):
  if y_tr[i] == 11:
    y_ada_tr.append(1)
    x_ada_tr.append(x_tr[i])
    index_tr.append(cnt)
    cnt +=1
  elif y_tr[i] == 26:
    y_ada_tr.append(-1)
    x_ada_tr.append(x_tr[i])
    index_tr.append(cnt)
    cnt += 1

cnt = 0
for i in range(0, len(y_test)):
  if(y_test[i] == 11):
    y_ada_test.append(1)
    x_ada_test.append(x_test[i])
    index_test.append(cnt)
    cnt += 1
  elif y_test[i] == 26:
    y_ada_test.append(-1)
    x_ada_test.append(x_test[i])
    index_test.append(cnt)
    cnt += 1

x_ada_tr = np.array(x_ada_tr)
y_ada_tr = np.array(y_ada_tr)
x_ada_test = np.array(x_ada_test)
y_ada_test = np.array(y_ada_test)
index_tr = np.array(index_tr)
index_test = np.array(index_test)

x_sort = []
y_sort = []
index_sort = []

for i in range(0, len(x_ada_tr[0])):
  temp = x_ada_tr[:, i]
  x_sort.append(temp[temp.argsort()])
  y_sort.append(y_ada_tr[temp.argsort()])
  index_sort.append(index_tr[temp.argsort()])


x_sort = np.array(x_sort)
```

```
y_sort = np.array(y_sort)
index_sort = np.array(index_sort)

u = np.full((1,len(x_ada_tr)), float(1/len(x_ada_tr)), dtype = float)
s = np.array([1, -1])
thres = []
for i in range(0, len(x_sort)):
  sub = []
  sub.append(-math.inf)
  for j in range(0, len(x_sort[i])-1):
    if(x_sort[i][j] != x_sort[i][j+1]):
      sub.append(float((x_sort[i][j] + x_sort[i][j+1])/2))
  thres.append(sub)
thres = np.array(thres)

n_feature = len(x_sort)
N = len(x_sort[0])
NN = len(x_ada_test)
```

```
def errada(besti, bests, bestth):
  ein = 0
  for k in range(0, N):
    ss = 1
    if(x_sort[besti][k] < bestth):
      ss = -1
    ss *= bests
    if(ss != y_sort[besti][k]):
      ein += 1
  ein = float(ein/N)
  return ein
```

Q17 A Q18 C

```
#Q17#Q18
minein = 2
maxein = 0
Gin = np.zeros(N)
Gout = np.zeros(NN)
for t in range(0, 1000):
  mini = 2
  besti = 0
  bests = 0
  bestth = 0
  for i in range(0, n_feature):
    for sign in s:
      for th in thres[i]:
        einu = 0
        for j in range(0, N):
          ss = 1
          if(x_sort[i][j] < th):
            ss = -1
          ss *= sign
          if(ss != y_sort[i][j]):
            einu += u[0][index_sort[i][j]]
        einu = float(einu/N)
        if(einu < mini):
          mini = einu
          mineinu = einu
          besti = i
          bests = sign
          bestth = th
  ein = errada(besti, bests, bestth)
  if(ein < minein):
    minein = ein
  if(ein > maxein):
    maxein = ein

  sum = float(0)
  wr = float(0)
  for j in range(0, N):
    sum += u[0][index_sort[besti][j]]
    ss = 1
    if(x_sort[besti][j] < bestth):
      ss = -1
    ss *= bests
    if(ss != y_sort[besti][j]):
      wr += u[0][index_sort[besti][j]]
  epi = float(wr/sum)
  incor = math.sqrt(float((1-epi)/epi))
  cor = math.sqrt(float((epi)/(1-epi)))
  for j in range(0, N):
    ss = 1
    if(x_ada_tr[j][besti] < bestth):
      ss = -1
    ss *= bests
    if(ss != y_ada_tr[j]):
      u[0][j] *= incor
```

```
    else:
      u[0][j] *= cor
    Gin[j] += ss

  for j in range(0, NN):
    ss = 1
    if(x_ada_test[j][besti] < bestth):
      ss = -1
    ss *= bests
    Gout[j] += ss


print(minein)
print(maxein)
```

get minEin(gt) = 0.09846547314578005 = 0.1

get maxEin(gt) = 0.571611253196931 = 0.57

Q19 A Q20 A

```
for j in range(0, N):
  if(Gin[j] >= 0):
    Gin[j] = 1
  else:
    Gin[j] = -1

for j in range(0, NN):
  if(Gout[j] >= 0):
    Gout[j] = 1
  else:
    Gout[j] = -1

print("einG:",err(Gin, y_ada_tr))
print("eoutG:",err(Gout, y_ada_test))
```

get Ein(G) = 0.0

get Eout(G) = 0.002793296089385475