Machine Learning HW5

B03902089 資工三 林良翰

Transforms: Explicit versus Implicit

1.

- $\phi_1(X) = 2x_2^2 - 4x_1 + 1$ and $\phi_2(X) = x_1^2 - 2x_2 - 3$
- $X_i = (x_1, x_2) \to Z_i = (\phi_1(X_i), \phi_2(X_i)) = (z_1, z_2)$
- $X_1 = (1, 0) \to Z_1 = (-3, -2), Y_1 = -1$
  $X_2 = (0, 1) \to Z_2 = (3, -5), Y_2 = -1$
  $X_3 = (0, -1) \to Z_3 = (3, -1), Y_3 = -1$
  $X_4 = (-1, 0) \to Z_4 = (5, -2), Y_4 = +1$
  $X_5 = (0, 2) \to Z_5 = (5, -7), Y_5 = +1$
  $X_6 = (0, -2) \to Z_6 = (9, 1), Y_6 = +1$
  $X_7 = (-2, 0) \to Z_7 = (9, 1), Y_7 = +1$
- $z_1 = 4$ is the optimal separting “hyperplane” in Z space

2.

- Polynomial kernel with penalty parameter $C = 10^6$, independent term $\zeta = 2$, kernel coefficient $\gamma = 1$, degree $d = 2$.
- Optimal $\alpha \approx [0.0,\ 0.4591,\ 0.4741,\ 0.5333,\ 0.1962,\ 0.2037,\ 0.0]$
- Support vectors: $[(0, 1),\ (0, -1),\ (-1, 0),\ (0, 2),\ (0, -2)]$

```
from sklearn import svm

X = [[1, 0], [0, 1], [0, -1], [-1, 0], [0, 2], [0, -2], [-2, 0]]
y = [-1, -1, -1, 1, 1, 1, 1]

clf = svm.SVC(C=1000000.0, kernel='poly', coef0=2, degree=2, gamma=1)
clf.fit(X, y)
print("support vectors:")
print(clf.support_vectors_)
print("alpha * y:", clf.dual_coef_)
```

3.

- $b = y_s - \sum_{SV\ indices\ n} \alpha_n y_n K(x_n, x_s)$ with support vector x_s and label y_s.

- $w = \left(\sum_{SV\ indices\ n} \alpha_n y_n K(x_n, x) \right) + b$ with a new vector x to predict.
- The corresponding nonlinear curve $\approx \frac{8}{15}(x_1)^2 + \frac{2}{3}(x_2)^2 - \frac{32}{15}x_1 - \frac{5}{3}$

```
from sklearn import svm
import numpy as np

b = []
for i in clf.support_:
  b.append([])
  for j, ya in zip(clf.support_, clf.dual_coef_[0]):
    b[-1].append(ya * (X[i][0] * X[j][0] + X[i][1] * X[j][1] + 2)**2)
  b[-1] = (y[i] - sum(b[-1]))
print("b:", np.array(b))

ayk = []
# ay is alpha * y
for x, ya in zip(clf.support_vectors_, clf.dual_coef_[0]):
  # kernel is (2 + XX')^2 = 4 + 4XX' + (XX')(XX')
  # the coefficient is 4 + 4X + XX --> (x1)^2 + (x2)^2 + 4(x1) + 4(x2) + 4
  ayk.append([ya * x[0]**2, ya * x[1]**2, ya * 4 * x[0], ya * 4 * x[1], ya * 4]
print("w:", np.sum(np.array(ayk), axis=0))
```

4.

- $z_1 = 2(x_2)^2 - 4x_1 + 1 = 4$ and $\frac{8}{15}(x_1)^2 + \frac{2}{3}(x_2)^2 - \frac{32}{15}x_1 - \frac{5}{3}$ are different because they are learned with respect to different Z space.

Dual Problem of L2-Error Soft-Margin Support Vector Machines

5.

- $\mathcal{L}((b, w, \xi), \alpha, \beta) = \frac{1}{2}w^T w + C \sum_{n=1}^{N} (\xi_n)^2 + \sum_{n=1}^{N} \alpha_n \left(1 - \xi_n - y_n \left(w^T x_n + b\right)\right) + \sum_{n=1}^{N} \beta_n (-\xi_n)$
- Partial differentiated by ξ_n
  $\frac{\partial \mathcal{L}((b,w,\xi),\alpha,\beta)}{\partial \xi_n} = 2C\xi_n - \alpha_n - \beta_n = 0, \Rightarrow 2C\xi_n - \alpha_n = \beta_n \geq 0$
  $0 \leq \alpha_n \leq 2C\xi_n \Rightarrow \beta$ can be removed. $\xi \geq 0$ is explicit.
- $\mathcal{L}((b, w, \xi), \alpha) = \frac{1}{2}w^T w + C \sum_{n=1}^{N} (\xi_n)^2 + \sum_{n=1}^{N} \alpha_n \left(1 - \xi_n - y_n \left(w^T x_n + b\right)\right) + \sum_{n=1}^{N} \left(2C\xi_n - \alpha_n\right)(-\xi_n)$
  $\mathcal{L}((b, w, \xi), \alpha) = \frac{1}{2}w^T w + \sum_{n=1}^{N} \alpha_n \left(1 - y_n \left(w^T x_n + b\right)\right) + \sum_{n=1}^{N} C(\xi_n)^2 - \alpha_n \xi_n - 2C\xi_n + \alpha_n \xi_n$
  $\mathcal{L}((b, w, \xi), \alpha) = \frac{1}{2}w^T w + \sum_{n=1}^{N} \alpha_n \left(1 - y_n \left(w^T x_n + b\right)\right) - \sum_{n=1}^{N} C(\xi_n)^2$

6.

- $\mathcal{L}((b, w, \xi), \alpha) = \frac{1}{2} w^T w + C \sum_{n=1}^{N} (\xi_n)^2 + \sum_{n=1}^{N} \alpha_n \left(1 - \xi_n - y_n \left(w^T x_n + b\right)\right)$
- Partial differentiated by ξ_n
  $\frac{\partial \mathcal{L}((b,w,\xi),\alpha)}{\partial \xi_n} = 2C\xi_n - \alpha_n = 0, \Rightarrow C\xi_n - \alpha_n = -C\xi_n$
- Finally we obtain
  $\mathcal{L}((b, w, \xi), \alpha) = \frac{1}{2} w^T w + \sum_{n=1}^{N} \alpha_n \left(1 - y_n \left(w^T x_n + b\right)\right) - C \sum_{n=1}^{N} (\xi_n)^2$

7.

- $L((b, w, \xi), \alpha) = \frac{1}{2} w^T w + \sum_{n=1}^{N} C (\xi_n)^2 + \sum_{n=1}^{N} \alpha_n \left(1 - \xi_n - y_n \left(w^T x_n + b\right)\right)$
- $\frac{\partial L((b,w,\xi),\alpha)}{\partial b} = \sum_{n=1}^{N} -\alpha_n y_n = 0 \Rightarrow b$ can be removed.
  $\Rightarrow L((b, w, \xi), \alpha) = \frac{1}{2} w^T w + \sum_{n=1}^{N} C (\xi_n)^2 + \sum_{n=1}^{N} \alpha_n \left(1 - \xi_n - y_n w^T x_n\right)$
- $\frac{\partial L((b,w,\xi),\alpha)}{\partial w_i} = w_i - \alpha_n y_n x_{n,i} = 0 \Rightarrow w = \sum_{n=1}^{N} \alpha_n y_n x_n$
  $\Rightarrow L((b, w, \xi), \alpha) = -\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n y_n x_n \right\|^2 + \sum_{n=1}^{N} C (\xi_n)^2 + \sum_{n=1}^{N} \alpha_n - \sum_{n=1}^{N} \alpha_n \xi_n$
- $\frac{\partial L((b,w,\xi),\alpha)}{\partial \xi_n} = 2C\xi_n - \alpha_n = 0 \Rightarrow \xi_n = \frac{\alpha_n}{2C}$
  $\Rightarrow L((b, w, \xi), \alpha) = -\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n y_n x_n \right\|^2 - \frac{1}{4C} \sum_{n=1}^{N} (\alpha_n)^2 + \sum_{n=1}^{N} \alpha_n$
- KKT conditions
  - Primal feasible: $y_n \left(w^T x_n + b\right) \geq 1 - \xi_n$
  - Dual feasible: $\alpha_n \geq 0$
  - Dual-inner optimal: $\sum_{n=1}^{N} -\alpha_n y_n = 0, w = \sum_{n=1}^{N} \alpha_n y_n x_n$
  - Primal-inner optimal: $\alpha_n \left(1 - \xi_n - y_n \left(w^T x_n + b\right)\right) = 0$

8.

- If we use $z_n = \phi(x_n)$, it will cost more computation power to calculate $\phi(x_n)\phi(x_m)$. Therefore we use a kernel $K(x_n, x_m)$ to compute the transformation and inner product in an efficient way.
- Optimization problem with kernel trick:
  - Quadratic coefficient: $q_{n,m} = y_n y_m z_n^T z_m = y_n y_m K(x_n, x_m), p = -1_N, (A, c)$ for equation and bound constraints.
  - $\alpha = QP(Q_D, p, A, c)$
  - Optimal bias from free SV (x_s, y_s): $b = y_s - \sum_{n=1}^{N} \alpha_n y_n K(x_n, x_s)$
  - Optimal hypothesis g_{svm} for test input x: $g_{svm}(x) = sign\left(\sum_{n=1}^{N} \alpha_n y_n K(x_n, x) + b\right)$

Operation of Kernels

9.

- Valid kernel $\Rightarrow$ positive-semidefinite matrix $\Rightarrow$ eigenvalue non-negative
  $x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, K = \begin{bmatrix} K_{11} & \cdots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \cdots & K_{NN} \end{bmatrix}$
  We need to prove $x^T K x = \sum_{i=1}^{N} \sum_{j=1}^{N} x_j K_{ij} x_j \geq 0$
- Denote K as $K_1\left(x, x'\right)$, and set $K = 0.5I = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, $eigen\,(K) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$
- [a]
  $eigen\left((1-K)^1\right) = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \Rightarrow$ not valid kernel
- [b]
  Any matrix with 0-th power always results into matrix filled with ones.
  $eigen\left((1-K)^0\right) = eigen\left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right) = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \Rightarrow$ valid kernel
- [c]
  Positive semi-definite matrix is closed under addition and multiplication
  $\Rightarrow I + K^1 + K^2 + K^3 + \cdots + K^n$ is valid kernel.
  We have known that $0 < K < 1 \Rightarrow \lim_{n\to\infty} K^n = 0$, thus:
  $\lim_{n\to\infty} I + K^1 + K^2 + K^3 + \cdots + K^n = \frac{(I-K^n)\cdot I}{I-K} = (I-K)^{-1}$ is also a valid kernel.
- [d]
  From [c], we have known is a valid kernel, and we known its closeness under multiplication and addition.
  $(I-K)^{-1}(I-K)^{-1} = (I-K)^{-2}$ is also a valid kernel.

10. Kernel Scaling and Shifting

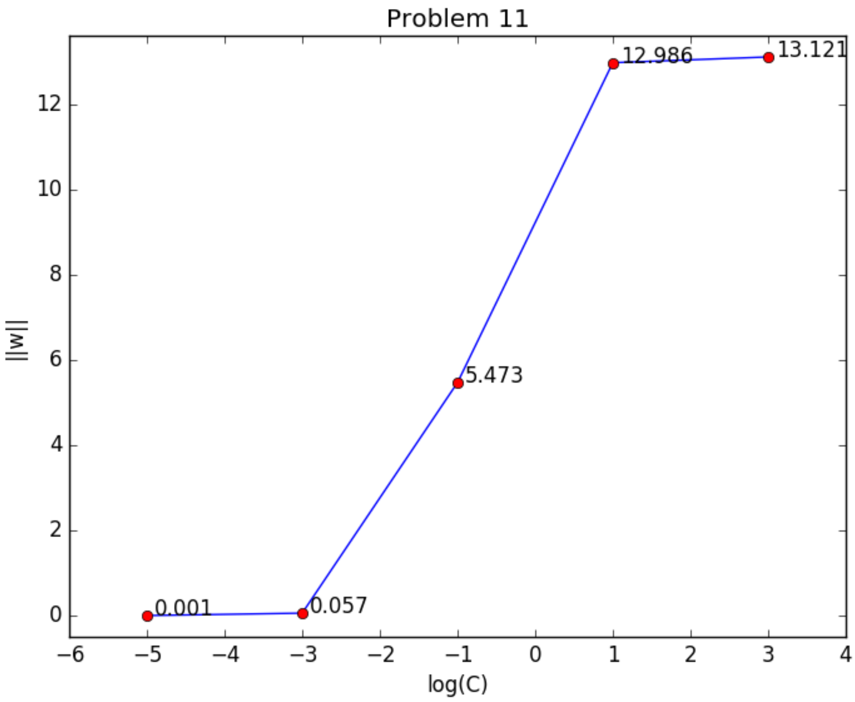
- $\tilde{K}\left(x, x'\right) = pK\left(x, x'\right) + q$
- We need to prove $\tilde{g}_{svm}(x) = g_{svm}(x)$
- $b = y_s - \sum_{SV\ indices\ n}^{N} \alpha_n y_n K(x_n, x_s)$ on bounded SV $\left(x_s, y_s\right)$

$$\begin{aligned} g_{svm}(x) &= sign\left(\left(\sum_{SV\ indices\ n}^{N} \alpha_n y_n K(x_n, x)\right) + b\right) \\ &= sign\left(\left(\sum_{SV\ indices\ n}^{N} \alpha_n y_n K(x_n, x)\right) + y_s - \sum_{SV\ indices\ n}^{N} \alpha_n y_n K(x_n, x_s)\right) \\ &= sign\left(\left(\sum_{SV\ indices\ n}^{N} \alpha_n y_n \left(K(x_n, x) - K(x_n, x_s)\right)\right) + y_s\right) \end{aligned}$$

- $\tilde{b} = y_s - \sum_{SV\ indices\ n}^{N} \tilde{\alpha}_n y_n \tilde{K}(x_n, x_s) = y_s - \sum_{SV\ indices\ n}^{N} \tilde{\alpha}_n y_n \left(pK(x_n, x_s) + q\right)$
  on bounded SV $\left(x_s, y_s\right)$
  $$\begin{aligned}\tilde{g}_{svm}(x) &= sign\left(\left(\sum_{SV\ indices\ n}^{N} \tilde{\alpha}_n y_n \tilde{K}(x_n, x)\right) + \tilde{b}\right) \\ &= sign\left(\left(\sum_{SV\ indices\ n}^{N} \tilde{\alpha}_n y_n \left(pK(x_n, x) + q\right)\right) + y_s - \sum_{SV\ indices\ n}^{N} \alpha_n y_n \left(pK(x_n, x_s) + q\right)\right) \\ &= sign\left(\left(\sum_{SV\ indices\ n}^{N} p\tilde{\alpha}_n y_n \left(K(x_n, x) - K(x_n, x_s)\right)\right) + y_s\right) \\ &= g_{svm}(x)\end{aligned}$$
- $\tilde{\alpha}_n = \frac{1}{p}\alpha_n \Rightarrow \tilde{C} = \frac{1}{p}C$

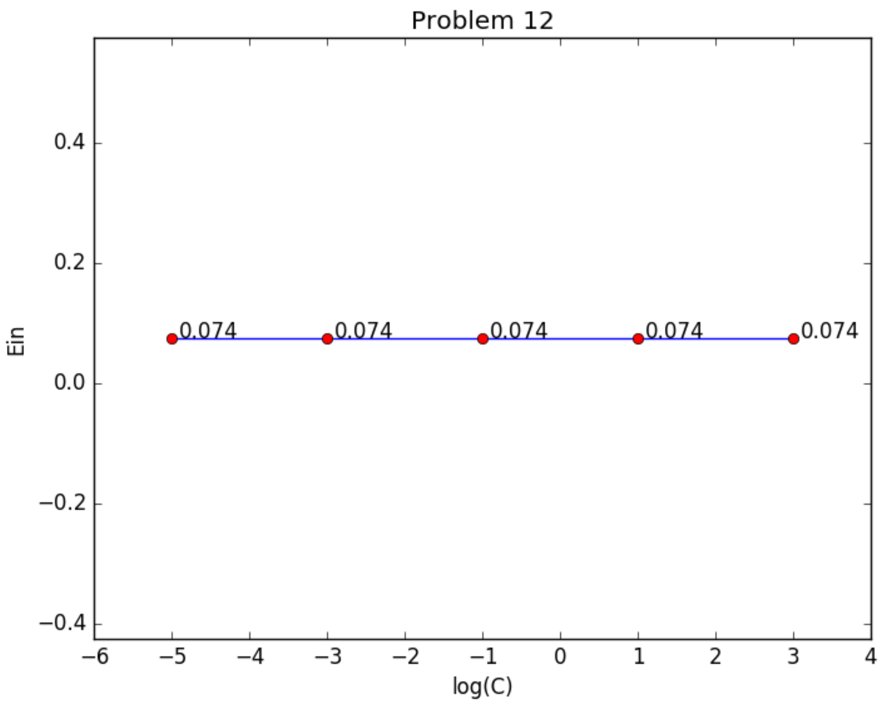
Experiments with Soft-Margin Support Vector Machine

11.



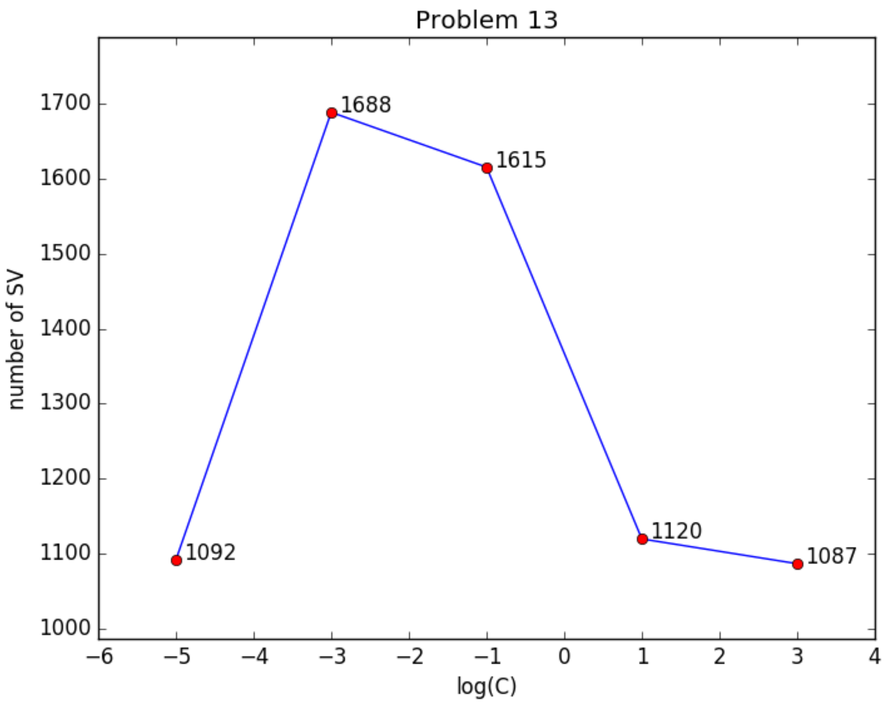
- Larger C will cause larger $\|w\|$.

12.



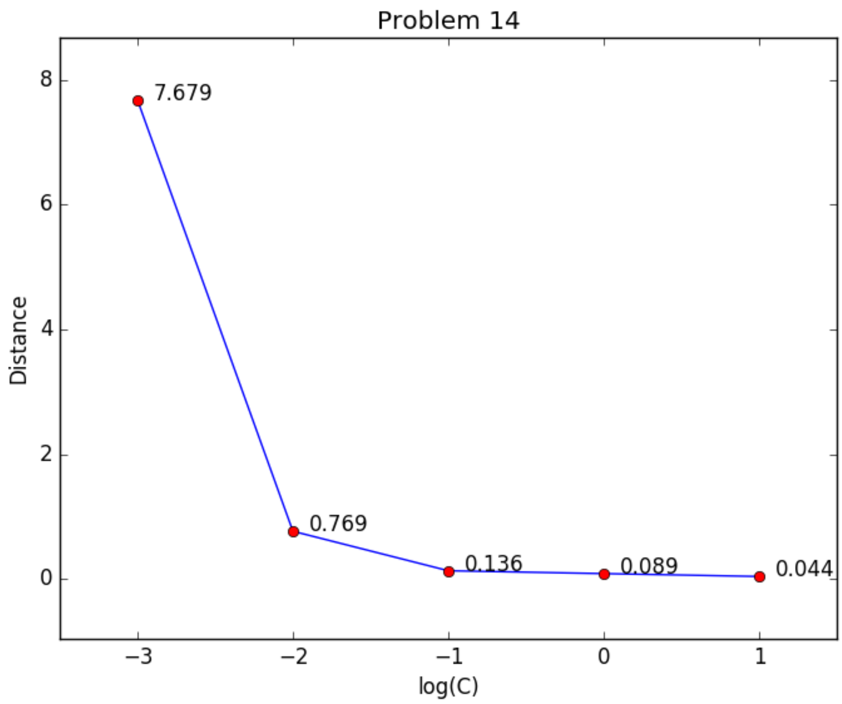
- All E_{in} are the same.

13.



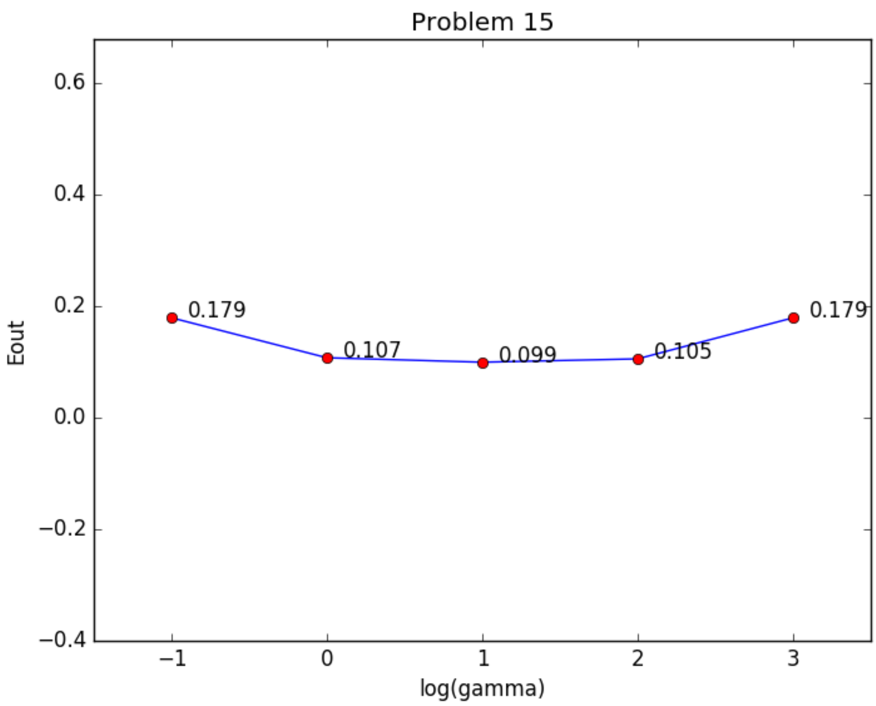
- When $\log_{10} C$ is around $-3 \sim -1$, the number of SVs is higher.

14.



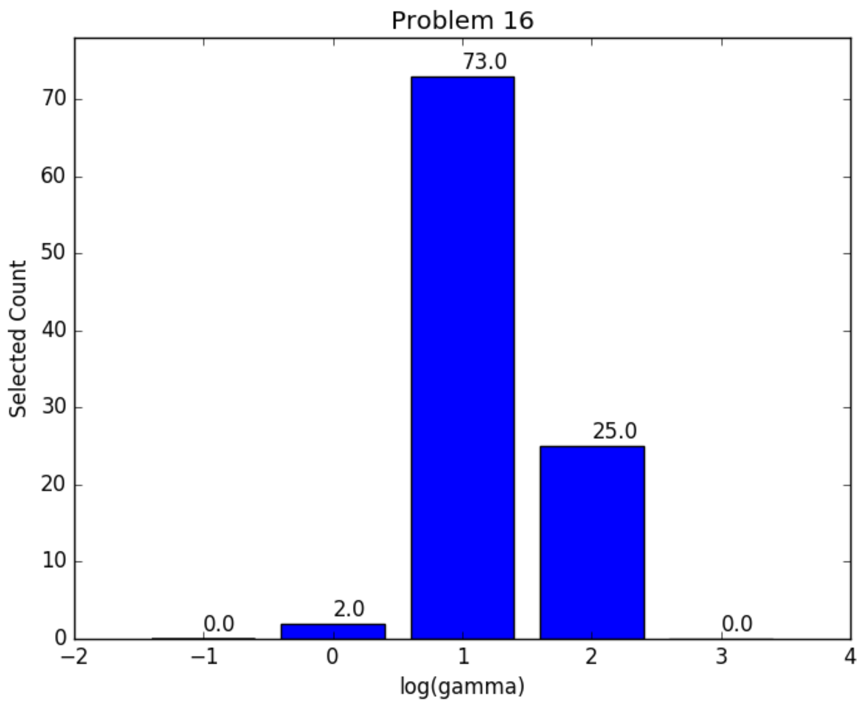
- Larger C will cause smaller distance between free SVs and hyperplane.

15.



- When $\log_{10}\gamma = 1$, E_{out} is the lowest.

16.



- By 100 iteration of validation, we found $\log_{10} \gamma = 1$ has the lowest E_{val}.