HW 2

Arnab Dey
Student ID: 5563169
Email: dey00011@umn.edu

Solution 1.a

Kernel function $K_i(x,x')$ is defined as

$$K_i(x,x') = \phi_i(x)^T\phi_i(x'),$$

where $x, x \in \mathbb{R}^p$, and $\phi(x) \in \mathbb{R}^d$ is a feature space transformation. Note, that a valid kernel function is symmetric and positive semidefinite. Let $K_1, K_2, \ldots, K_m$ be a set of valid kernel functions. Therefore, for any $w_j \geq 0, j = 1, 2, \ldots, m$

$$\begin{aligned}
K(x,x') &= \sum_{j=1}^{m} w_j K_j(x,x') \\
&= \sum_{j=1}^{m} w_j \phi_j(x)^T \phi_j(x') \\
&= \begin{bmatrix} \sqrt{w_1}\phi_1(x)^T & \sqrt{w_2}\phi_2(x)^T & \cdots & \sqrt{w_m}\phi_m(x)^T \end{bmatrix} \begin{bmatrix} \sqrt{w_1}\phi_1(x') \\ \sqrt{w_2}\phi_2(x') \\ \cdots \\ \sqrt{w_m}\phi_m(x') \end{bmatrix} \\
&= \phi(x)^T\phi(x'),
\end{aligned}$$

where, $\phi(x) := \begin{bmatrix} \sqrt{w_1}\phi_1(x)^T & \sqrt{w_2}\phi_2(x)^T & \cdots & \sqrt{w_m}\phi_m(x)^T \end{bmatrix}^T$. If $\phi_j(x) \in \mathbb{R}^{d_j}, j = 1, 2, \ldots, m$ then $\phi(x) \in \mathbb{R}^{(\sum_{j=1}^{m} d_j)}$. Therefore,

$$K(x',x) = \phi(x')^T\phi(x) = \phi(x)^T\phi(x') = K(x,x').$$

Therefore, $K(x,x')$ is symmetric. Also, for any $y \in \mathbb{R}^N$,

$$\begin{aligned}
y^T K y &= \sum_{i=1}^{N}\sum_{j=1}^{N} y_i K_{ij} y_j \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} y_i \phi(x_i)^T \phi(x_j) y_j \\
&= \|\sum_{i=1}^{N} y_i \phi(x_i)\|^2 \geq 0,
\end{aligned}$$

where $K_{ij} := \phi(x_i)^T\phi(x_j) = K(x_i,x_j)$ is the ij^{th} entry of Gram matrix K. Therefore, K is positive semidefinite. Thus, $K(x,x')$ is a valid kernel.

Solution 1.b

Consider,

$$K(x,x') = K_1(x,x') + K_2(x,x'),$$

where K_1 and K_2 are valid kernel functions. Let,

$$K_1(x,x') = \phi_1(x)^T\phi_1(x'),\ K_2(x,x') = \phi_2(x)^T\phi_2(x'),$$

where $\phi_1(x) \in \mathbb{R}^{d_1}$ and $\phi_2(x) \in \mathbb{R}^{d_2}$ are feature transformations. Now,

$$\begin{aligned} K(x,x') &= \phi_1(x)^T\phi_1(x') + \phi_2(x)^T\phi_2(x') \\ &= \begin{bmatrix}\phi_1(x)^T & \phi_2(x)^T\end{bmatrix}\begin{bmatrix}\phi_1(x') \\ \phi_2(x')\end{bmatrix} \\ &= \phi(x)^T\phi(x'), \end{aligned}$$

where $\phi(x) := \begin{bmatrix}\phi_1(x)^T & \phi_2(x)^T\end{bmatrix}^T \in \mathbb{R}^{d_1+d_2}$. Now,

$$K(x',x) = \phi(x')^T\phi(x) = \phi(x)^T\phi(x') = K(x,x').$$

Therefore, $K(x,x')$ is symmetric. Also, for any $y \in \mathbb{R}^N$,

$$\begin{aligned} y^TKy &= \sum_{i=1}^N\sum_{j=1}^N y_iK_{ij}y_j \\ &= \sum_{i=1}^N\sum_{j=1}^N y_i\phi(x_i)^T\phi(x_j)y_j \\ &= \|\sum_{i=1}^N y_i\phi(x_i)\|^2 \geq 0, \end{aligned}$$

where $K_{ij} := \phi(x_i)^T\phi(x_j) = K(x_i,x_j)$ is the ij^{th} entry of Gram matrix K. Therefore, K is positive semidefinite. Thus, $K(x,x')$ is a valid kernel.

Solution 2

In case of two classes, C_1 and C_2 with labels $+1/-1$, we are interested to find $w \in \mathbb{R}^{d_1}$ and b such that

$$\begin{cases} w^T\phi(x) + b \geq 1, & \text{if } x \in C_1 \\ w^T\phi(x) + b \leq -1 & \text{if } x \in C_2, \end{cases}$$

where $\phi(x) \in \mathbb{R}^{d_1}$ denotes feature space transformation of sample $x \in \mathbb{R}^d$. Note, that in case of linear SVM, $\phi(x) = x$. The training data comprises of N input samples $x_1, x_2, \ldots, x_N$ with corresponding target values $t_1, t_2, \ldots, t_N$ such that $t_1 = 1$ if $x \in C_1$ and -1 otherwise. The distance of x_n from the discriminant hyperplane $w^T\phi(x_n) + b$ is given by,

$$\frac{|w^T\phi(x_n) + b|}{\|w\|},$$

which can be written as,

$$\frac{t_n(w^T\phi(x_n) + b)}{\|w\|},$$

as $t_n = 1$ if $x_n \in C_1$ and -1 otherwise. In case of large margin classifier, we would like this be to at least some value ρ:

$$\frac{t_n(w^T\phi(x_n) + b)}{\|w\|} = \rho,$$

and our objective would be to maximize ρ. As we can see that there are infinite number of solutions which can be obtained by scaling w and b. Therefore, without loss of generality, for a unique solution, we fix $\rho\|w\| = 1$. Therefore, the problem can be formulated as,

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2, \\ &\text{such that } t_n(w^T\phi(x_n) + b) \geq 1,\ \forall n = 1, 2, \ldots, N. \end{aligned}$$

But in case of non-separable classes, the above optimization algorithm will not work. There can be two types of deviation of the samples from the optimal hyperplane: a sample may lie on the wrong side of the hyperplane and thus be miss-classified, or it may be on the right side of the hyper but within the margin, *i.e.* not sufficiently away from the separating hyperplane. To penalize the hyperplane for these two cases, we introduce slack variable $\xi \geq 0$, which store the deviation from the margin. Therefore, in case of non-separable classes we require,

$$t_n(w^T\phi(x_n) + b) \geq 1 - \xi_n. \tag{1}$$

If $\xi_n = 0$, then there is no problem with x_n. If $0 < \xi_n < 1$, x_n is correctly classified but in the margin. If $\xi_n \geq 1$, the sample x_n is misclassified. We deefine soft error as,

$$\sum_{n=1}^{N} \xi_n,$$

and add this as a penalty term so that our cost function becomes,

$$L = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n,$$

where C is th penalty factor. Adding the constraint (1) and $\xi_n \geq 0$ $\forall n = 1, 2, \ldots, N$, the Lagrangian becomes,

$$L(w, b, a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x) + b) - 1 + \xi_n\} - \sum_{n=1}^{N} \mu_n\xi_n. \tag{2}$$

Taking derivatives with respect to the parameters and setting them to 0, we get:

$$\begin{aligned}
\frac{\partial L}{\partial w} &= 0 \implies w = \sum_{n=1}^{N} a_n t_n \phi(x_n), \\
\frac{\partial L}{\partial b} &= 0 \implies \sum_{n=1}^{N} a_n t_n = 0, \\
\frac{\partial L}{\partial \xi_n} &= 0 \implies a_n = C - \mu_n.
\end{aligned} \tag{3}$$

Using this results to eliminate w, b and $\{\xi_n\}$ from (2), we get the dual Lagrangian form:

$$L_d = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m K(x_n, x_m), \tag{4}$$

where $K(x_n, x_m) = \phi(x_n)^T\phi(x_m)$ is know as Gram matrix or kernel matrix. We have to maximize (4) with respect to the dual variables $\{a_n\}$ subject to

$$\begin{aligned}
0 \leq a_n &\leq C, \\
\sum_{n=1}^{N} a_n t_n &= 0.
\end{aligned}$$

This is a quadratic programming which we solve using *cvxopt* python library. We have to vectorize (4) to make it compatible with *cvxopt*. Therefore, if we define a matrix H such that $H(n, m) = t_n t_m K(x_n, x_m)$, we can write the optimization problem (4) as follows:

$$\begin{aligned}
&\min_a \frac{1}{2}a^T H a - \mathbb{1}^T a, \\
&s.t. - a_n \leq 0, \\
&a_n \leq C, \\
&t^T a = 0,
\end{aligned} \tag{5}$$

where $a := [a_1 \, a_2 \, \dots \, a_N]^T$ and $t := [t_1 \, t_2 \, \dots \, t_N]^T$. Once we have the optimization problem in this form, we can use *cvxopt qp* library for quadratic programming. After solving the optimization problem and obtaining optimal values of a, we can compute w from (3) in case of linar SVM as $\phi(x_n) = x_n$, $\forall n = 1, 2, \dots, N$. b can be found from the fact that for any support vector that satisfies $0 < a_n < C$ have $\xi_n = 0$ hence satisfies,

$$t_n \left(\sum_{m \in \mathcal{S}} a_n t_n K(x_n, x_m) + b \right) = 1, \tag{6}$$

where, $\mathcal{S}$ is the set of the indices of the support vectors. For numerical stability, we take the average over all such vectors *i.e.*

$$b = \frac{1}{N_m} \sum_{n \in \mathcal{M}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m K(x_n, x_m) \right), \tag{7}$$

where $\mathcal{M}$ is the set of indices of datapoints having $0 < a_n < C$.

The discriminant function is given by,

$$\begin{aligned} g(x) &= w^T \phi(x) + b \\ &= \sum_{n=1}^{N} a_n t_n K(x, x_n) + b. \end{aligned} \tag{8}$$

We might see that a subset of the data points may have $a_n = 0$, in which case they do not contribute to the discriminant function. The remaining data points with $a_n > 0$ are the support vectors. In case of linear SVM, (8) can be written as

$$\begin{aligned} g(x) &= w^T x + b \\ &= \sum_{n=1}^{N} a_n t_n x^T x_n + b. \end{aligned} \tag{9}$$

In two class classification, we label $x \in C_1$ if $g(x) \geq 0$ and $x \in C_2$ if $g(x) < 0$.

Results: The result is summarized in Table 1, 2 and 3. From Table 1 we can infer that the data is

Table 1: Q2: Mean error over 10-fold cross validation: Linear SVM on 'hw2_data_2020.csv'

C	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
Training(%)	49.78	49.36	49.37	49.40	49.39	49.38	49.37	49.37
Validation(%)	50.7	50.09	49.84	49.74	49.73	49.71	49.7	49.7

Table 2: Q2: Std of error over 10-fold cross validation: Linear SVM on 'hw2_data_2020.csv'

C	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
Training(%)	0.78	0.36	0.28	0.28	0.26	0.26	0.27	0.27
Validation(%)	1.68	1.56	1.62	1.52	1.54	1.52	1.54	1.54

not linearly separable and thus error percentage is very high. Also, we can see that as we increase C the validation error rate decreases, which indicates more penalty is added to the optimization problem to penalize the data which are not correctly classified with linear discriminant function. Therefore, a high value of C might be desired for this dataset. I choose $C = 100$ as this gives the lowest average validation error and also for the penalty factor reason stated. Table 3 shows the error rate for $C = 100$ on the test data set.

Table 3: Q2: Test error for optimal C: Linear SVM on 'hw2_data_2020.csv'

C	100
Test(%)	50.8

Solution 3

SVM procedure with *cvxopt* for a general kernel function $K(x_n, x_m)$ has been summarized in the theory description of 'Solution 2'. Note, that in Q3, we have to test the SVM algorithm with Linear and RBF (Radial basis function) kernels.

Linear kernel: For this kernel

$$K(x_n, x_m) = \phi(x_n)^T \phi(x_m) = x_n^T x_m.$$

The discriminant function is given in (9. In case of linear kernels, we can compute the weights w explicitly but this is not required if we already have the kernel matrix (also known as Gram matrix) with us as the discriminant function involves kernel. Intercept is computed from (7).

Results: Results with linear kernel is summarized in Table 4, 5 and 6. From Table 4 we can infer that

Table 4: Q3: Mean error over 10-fold cross validation: Linear Kernel on 'hw2_data_2020.csv'

C	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
Training(%)	49.78	49.36	49.37	49.40	49.39	49.38	49.37	49.37
Validation(%)	50.7	50.09	49.84	49.74	49.73	49.71	49.7	49.7

Table 5: Q3: Std of error over 10-fold cross validation: Linear kernel on 'hw2_data_2020.csv'

C	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
Training(%)	0.78	0.36	0.28	0.28	0.26	0.26	0.27	0.27
Validation(%)	1.68	1.56	1.62	1.52	1.54	1.52	1.54	1.54

the data is not linearly separable and thus error percentage is very high. Also, we can see that as we increase C the validation error rate decreases, which indicates more penalty is added to the optimization problem to penalize the data which are not correctly classified with linear discriminant function. Therefore, a high value of C might be desired for this dataset. I choose $C = 100$ as this gives the lowest average validation error and also for the penalty factor reason stated. Table 6 shows the error rate for $C = 100$ on the test data set.

RBF kernel: RBF kernel is defined as

$$K(x_n, x_m) = \exp\left[-\gamma \|x_n - x_m\|^2\right], \tag{10}$$

where $\gamma > 0$ is a hyperparameter tuned via cross-validation. Usually γ can be interpreted as $\frac{1}{2\sigma^2}$, where σ is the feature variance. Therefore, I have computed variance over all data features and chose the following values of γ based on this variance estimate,

$$\gamma = [1.23 \times 10^{-6},\ 1.23 \times 10^{-5},\ 1.23 \times 10^{-4},\ 0.00123,\ 0.0123]$$

and as before and to compare the kernel with linear one, I have taken following C values to play with:

$$C = [10^{-4},\ 10^{-3},\ 10^{-2},\ 0.1,\ 1,\ 10,\ 100,\ 1000].$$

Table 6: Q3: Test error for optimal C: Linear kernel on 'hw2_data_2020.csv'

C	100
Test(%)	50.8

The choice of C comes from the fact that a high value of C means higher penalty for misclassification or for the samples with narrow margins. Therefore, the choice of these C covers a wide range of penalty weights. The discriminant function is shown in (8) and the kernel is (10). Intercept is computed from (7).

Results: Results with rbf kernel is summarized in Table 7, 8 and 11. From Table 9 we can see that the

Table 7: Q3: Train set mean error(%) over 10-fold cross validation: rbf Kernel on 'hw2_data_2020.csv'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 1.23 \times 10^{-6}$	49.61	33.62	0.19	0.79	0.99	1.04	1.04	1.03
$\gamma = 1.23 \times 10^{-4}$	49.61	34.48	0.31	0.32	0.41	0.42	0.42	0.43
$\gamma = 1.23 \times 10^{-3}$	49.61	40.82	15.5	10.02	9.39	9.33	9.22	1.26
$\gamma = 0.00123$	49.61	49.36	49.61	49.61	49.61	49.61	49.61	40.39
$\gamma = 0.0123$	49.61	50.09	49.61	49.61	49.61	49.61	49.54	47.41

Table 8: Q3: Train set std error(%) over 10-fold cross validation: rbf Kernel on 'hw2_data_2020.csv'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 1.23 \times 10^{-6}$	0.17	11.03	0.18	0.08	0.07	0.07	0.08	0.08
$\gamma = 1.23 \times 10^{-4}$	0.17	10.49	0.49	0.03	0.04	0.04	0.04	0.04
$\gamma = 1.23 \times 10^{-3}$	0.17	6.40	2.39	0.15	0.23	0.26	0.23	0.05
$\gamma = 0.00123$	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.19
$\gamma = 0.0123$	0.17	0.17	0.17	0.17	0.17	0.17	0.18	0.17

Table 9: Q3: Validation set mean error(%) over 10-fold cross validation: rbf Kernel on 'hw2_data_2020.csv'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 1.23 \times 10^{-6}$	49.61	36.97	6.82	14.4	16.77	17.18	17.2	17.2
$\gamma = 1.23 \times 10^{-4}$	49.61	37.58	6.7	8.71	9.98	10.1	10.1	10.08
$\gamma = 1.23 \times 10^{-3}$	49.61	42.75	22.65	17.83	17.43	17.42	17.33	11.88
$\gamma = 0.00123$	49.61	49.61	49.61	49.61	49.61	49.61	49.61	49.06
$\gamma = 0.0123$	49.61	49.61	49.61	49.61	49.61	49.61	49.61	49.46

Table 10: Q3: Validation set std error(%) over 10-fold cross validation: rbf Kernel on 'hw2_data_2020.csv'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 1.23 \times 10^{-6}$	1.55	9.83	1.76	1.95	1.28	1.19	1.19	1.19
$\gamma = 1.23 \times 10^{-4}$	1.55	9.59	2.31	1.26	1.10	1.09	1.09	1.06
$\gamma = 1.23 \times 10^{-3}$	1.55	6.56	2.86	1.11	0.95	0.97	1.01	1.25
$\gamma = 0.00123$	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.49
$\gamma = 0.0123$	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.44

optimal hyperparameter combination is $C = 0.01$ and $\gamma = 1.23 \times 10^{-5}$. Test error for these hyperparameters is shown in Table 11.

Table 11: Q3: Test error(%) for optimal C, γ: rbf kernel on 'hw2_data_2020.csv'

$C =$	0.01
$\gamma = 1.23 \times 10^{-5}$	4.55

Solution 4

In one-vs-all multiclass SVM with K_c different classes, we construct K_c separate SVMs, in which the k^{th} model discriminant function $g_k(x)$ is trained using the data from class C_k as positive examples and the data from the remaining $K_c - 1$ classes as the negative examples. The implementation for each SVM with linear and rbf kernel are same as explained in 'Solution 2' and 'Solution 3'. We make prediction for new input x using

$$\arg\max_k g_k(x).$$

i have taken the following γ values to play with and tune from cross validation. The interpretation for these range of values comes from the fact that γ can be interpreted as $\frac{1}{2\sigma^2}$, where σ^2 is the variance of the features. I have computed the feature variance and the selected this sweep range to cover a good amount of penalty factors.

$$\gamma = [0.01,\ 0.1,\ 1,\ 10,\ 100]$$

and as before and to compare the kernel with linear one, I have taken following C values to play with:

$$C = [10^{-4},\ 10^{-3},\ 10^{-2},\ 0.1,\ 1,\ 10,\ 100,\ 1000].$$

The choice of C comes from the fact that a high value of C means higher penalty for misclassification or for the samples with narrow margins. Therefore, the choice of these C covers a wide range of penalty weights.

Linear kernel results: The results of multi-class SVM on *mfeat-fou* with linear kernel are summarized in Table 12. 13 and 14. From Table 12, we can see that $C = 10$ produces the lowest validation error

Table 12: Q4: Mean error over 10-fold cross validation: Linear Kernel on 'mfeat-fou'

C	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
Training(%)	88.28	60.20	29.22	19.20	15.94	12.01	11.59	16.45
Validation(%)	90.12	61.18	30.75	22.12	19.25	19.00	21.06	25.56

Table 13: Q4: Std error over 10-fold cross validation: Linear Kernel on 'mfeat-fou'

C	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
Training(%)	2.96	5.49	2.68	1.16	0.47	0.38	2.50	3.48
Validation(%)	2.99	5.91	4.80	3.51	2.31	2.39	3.02	3.49

rate. This also indicates that a high penalty is required for misclassification to have a good performance on 'mfeat-fou' data. Table 14 shows the test error with $C = 10$.

RBF kernel results: The results of multi-class SVM on *mfeat-fou* with rbf kernel are summarized in Table 15, 16, 17, 18 and 19. From Table 17 we can see that the optimal hyperparametere combination is $C = 10$ and $\gamma = 1$. The test error with these hyperparameters is shown in Table 19.

Table 14: Q4: Test error for optimal C: Linear kernel on 'mfeat-fou'

C	10
Test(%)	17.25

Table 15: Q4: Train set mean error(%) over 10-fold cross validation: rbf Kernel on 'mfeat-fou'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 0.01$	89.76	87.09	64.57	29.63	17.54	17.06	19.88	63.0
$\gamma = 0.1$	89.93	67.81	35.37	20.56	15.5	13.22	38.56	53.61
$\gamma = 1$	86.85	55.91	23.45	13.76	8.59	2.58	3.31	3.51
$\gamma = 10$	81.40	58.64	20.75	17.70	15.47	15.19	15.19	15.20
$\gamma = 100$	74.48	32.53	19.78	19.29	19.27	19.09	19.09	19.09

Table 16: Q4: Train set std error(%) over 10-fold cross validation: rbf Kernel on 'mfeat-fou'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 0.01$	0.26	4.33	10.46	3.49	1.10	1.08	2.42	2.99
$\gamma = 0.1$	0.28	7.52	3.28	3.44	0.69	1.23	2.09	4.32
$\gamma = 1$	4.93	4.82	3.39	0.57	0.17	0.32	0.89	0.93
$\gamma = 10$	6.18	8.57	1.04	0.33	0.46	0.49	0.50	0.49
$\gamma = 100$	6.00	4.67	0.72	0.49	0.54	0.55	0.55	0.55

Table 17: Q4: Validation set mean error(%) over 10-fold cross validation: rbf Kernel on 'mfeat-fou'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 0.01$	90.75	87.62	64.12	32.31	22.75	23.0	28.5	68.81
$\gamma = 0.1$	89.93	70.75	36.37	25.12	21.5	23.18	47.62	63.06
$\gamma = 1$	86.62	56.0	26.68	19.87	16.12	16.06	19.0	19.12
$\gamma = 10$	84.87	61.12	23.56	20.93	19.87	19.25	19.25	19.25
$\gamma = 100$	79.87	37.5	23.75	22.68	22.62	22.5	22.5	22.5

Table 18: Q4: Validation set std error(%) over 10-fold cross validation: rbf Kernel on 'mfeat-fou'

$C =$	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	100	1000
$\gamma = 0.01$	2.14	3.93	10.57	4.57	3.41	2.92	4.65	5.91
$\gamma = 0.1$	2.56	7.84	3.90	5.88	3.44	3.32	4.99	7.81
$\gamma = 1$	4.63	7.09	5.22	3.37	2.14	2.46	3.30	3.07
$\gamma = 10$	5.44	9.57	3.50	3.09	2.75	2.25	2.25	2.25
$\gamma = 100$	7.03	5.22	3.06	2.63	2.84	2.80	2.80	2.80

Table 19: Q4: Test error(%) for optimal C, γ: rbf kernel on 'mfeat-fou'

$C =$	10.0
$\gamma = 1$	15.75