Page No.
Date

Ans 1:

$$p(y_i=1 / f(x_i)) = \frac{1}{1+e^{-f(x_i)}}$$

$$= \sigma(f(x_i)) \quad ; \sigma(x) \text{ is sigmoid function.}$$

$$p(y_i=-1 / f(x_i)) = 1 - \frac{1}{1+e^{-f(x_i)}}$$

$$= \frac{e^{-f(x_i)}}{1+e^{-f(x_i)}}$$

$$= \frac{1}{1+e^{f(x_i)}}$$

$$= \sigma(-f(x_i))$$

$$p(y_i / f(x_i)) = \sigma(y_i f(x_i))$$

Assuming data points to be independent and identically distributed (i.i.d):

$$\text{log likelihood, } L = \prod_{i=1}^{N} p(y_i / f(x_i))$$

$$\text{Negative log likelihood, } -\ln L = -\sum_{i=1}^{N} \ln [p(y_i / f(x_i))]$$

$$= -\sum_{i=1}^{N} \ln [\sigma(y_i f(x_i))]$$

$$= -\sum_{i=1}^{N} \ln \left[\frac{1}{1+e^{-y_i f(x_i)}} \right]$$

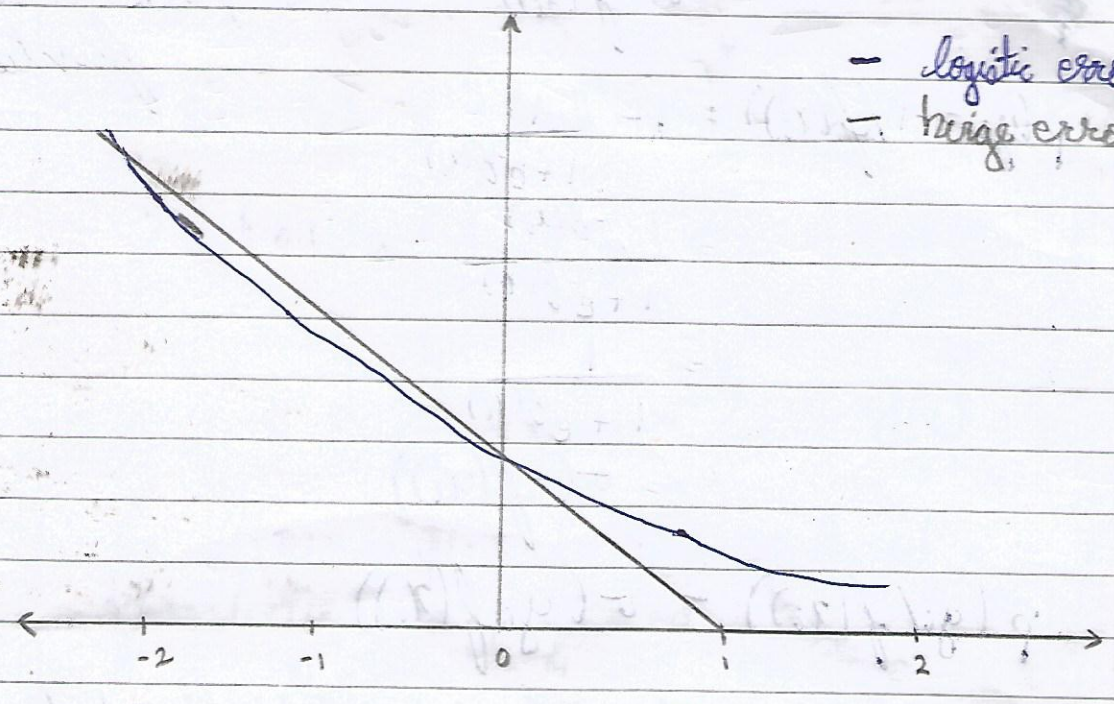
$$= \sum_{i=1}^{N} \ln [1+e^{-y_i f(x_i)}]$$

$$= \sum_{i=1}^{N} E_{LR}(y_i, f(x_i))$$

where $E_{LR}(y_i, f(x_i)) = \ln(1 + \exp(-y_i f(x_i)))$

With addition of a quadratic regularization term;

$$\rightarrow \sum_{i=1}^{N} E_{LR}(y_i, f(x_i)) + \lambda \|w\|^2$$



– logistic error function
–. hinge error function

~~log~~ Hinge error $= \sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda \|w\|^2$.

where $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$.

Both logistic and hinge loss are continuous approximation of misclassification error. Hinge loss is more sparse and covers less area than logistic error function.

Page No.
Date

Theory Extra Credit

Ans:

$$L(w, b, \xi_i \ldots \xi_N, \hat{\xi}_i \ldots \hat{\xi}_N) =$$

$$C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N}(\mu_i \xi_i + \hat{\mu}_i \hat{\xi}_i)$$

$$- \sum_{i=1}^{N} a_i(\varepsilon + \xi_i + f(x_i) - y_i) - \sum_{i=1}^{N} \hat{a}_i(\varepsilon + \hat{\xi}_i - f(x_i) + y_i) \quad (1)$$

$$\# \Rightarrow \frac{dL}{dw} = w - \sum_{i=1}^{N} a_i \phi(x_i) + \sum_{i=1}^{N} \hat{a}_i \phi(x_i) = 0$$

Here $f(x) = w^T\phi(x) + b$ was replaced and then differentiation was performed. ; $w = \sum_{i=1}^{N} \phi(x_i)[a_i - \hat{a}_i]$ (2)

$$\# \frac{dL}{db} = -\sum_{i=1}^{N} a_i + \sum_{i=1}^{N} \hat{a}_i = 0$$

$$\sum_{i=1}^{N}(\hat{a}_i - a_i) = 0 \quad (3)$$

$$\# \frac{dL}{d\xi_i} = C - \mu_i - a_i = 0$$

$$C = (\mu_i + a_i) \quad (4)$$

$$\# \frac{dL}{d\hat{\xi}_i} = C - \hat{\mu}_i - \hat{a}_i = 0$$

$$C = (\hat{\mu}_i + \hat{a}_i) \quad (5)$$

Now, combining like terms in 1;

$$\sum_{i=1}^{N} \xi_i (C - \mu_i - a_i) + \sum_{i=1}^{N} \hat{\xi}_i [C - \hat{\mu}_i - \hat{a}_i]$$

$$+ \varepsilon \left[-\sum_{i=1}^{N} a_i - \sum_{i=1}^{N} \hat{a}_i \right] + \left[\sum_{i=1}^{N} a_i y_i - \sum_{i=1}^{N} \hat{a}_i y_i \right]$$

$$+ \frac{1}{2} \|w\|^2 + \sum_{i=1}^{N} \left[\hat{a}_i w^T \phi(x_i) - a_i w^T \phi(x_i) \right]$$

Now, from 4 $\quad C = \mu_i + a_i$

from 5 $\quad C = \hat{\mu}_i + \hat{a}_i$

$\therefore$ (6) and (7) becomes 0, where (6) $\sum_{i=1}^{N} \xi_i (C - \mu_i - a_i) =$

(7) $\sum_{i=1}^{N} \hat{\xi}_i (C - \hat{\mu}_i - \hat{a}_i)$

Thus;

$$-\varepsilon \sum_{i=1}^{N} \left[a_i + \hat{a}_i \right] + \left[\sum_{i=1}^{N} (a_i - \hat{a}_i) y_i \right]$$

$$+ \frac{1}{2} \|w\|^2 + \sum_{i=1}^{N} \left[\hat{a}_i w^T \phi(x_i) - a_i w^T \phi(x_i) \right]$$

From 2;

$$\|w\|^2 = w^T w$$

$$\frac{1}{2} \sum_{i=1}^{N} \phi(x_i) \left[a_i - \hat{a}_i \right] \sum_{j=1}^{N} \phi(x_j)^T \left[a_j - \hat{a}_j \right]$$

$$+ \sum_{i=1}^{N}$$

$$\Rightarrow -\varepsilon\sum_{i=1}^{N}(a_i+\hat{a}_i)+\sum_{i=1}^{N}(a_i-\hat{a}_i)y_i$$

$$+\frac{1}{2}\left[\sum_{i=1}^{N}\phi(x_i)^T(a_i-\hat{a}_i)\sum_{j=1}^{N}\phi(x_j)(a_j-\hat{a}_j)\right]$$

$$+\sum_{i=1}^{N}\left[\hat{a}_i\omega^T\phi(x_i)-a_i\omega^T\phi(x_i)\right]$$

$$\Rightarrow -\varepsilon\sum_{i=1}^{N}(a_i+\hat{a}_i)+\sum_{i=1}^{N}(a_i-\hat{a}_i)y_i$$

$$+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}k(x_i,x_j)(a_i-\hat{a}_i)(a_j-\hat{a}_j)$$

$$+\sum_{i=1}^{N}\left[\phi(x_i)\sum_{j=1}^{N}\phi(x_j)(a_j-\hat{a}_j)-\phi(x_i)\sum_{j=1}^{N}\phi(x_j)\right]$$

$$+\sum_{i=1}^{N}\left[\hat{a}_i\phi(x_i)+\sum_{i=1}^{N}\phi(x_i)\left[\hat{a}_i\sum_{j=1}^{N}\phi(x_j)^T(a_j-\hat{a}_j)\right.\right.$$

$$\left.\left.-a_i\sum_{j=1}^{N}\phi(x_j)^T(a_j-\hat{a}_j)\right]\right.;\ k(x_i,x_j)=\phi(x_i)^T\phi(x_j)$$

$$\Rightarrow -\varepsilon\sum_{i=1}^{N}(a_i+\hat{a}_i)+\sum_{i=1}^{N}(a_i-\hat{a}_i)y_i$$

$$+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}k(x_i,x_j)(a_i-\hat{a}_i)(a_j-\hat{a}_j)$$

$$-\sum_{i=1}^{N}\sum_{j=1}^{N}\phi(x_j)^T\phi(x_i)(\hat{a}_j-a_j)(\hat{a}_i-a_i)$$

$$\Rightarrow -\varepsilon\sum_{i=1}^{N}(a_i+\hat{a}_i)+\sum_{i=1}^{N}(a_i-\hat{a}_i)y_i$$

$$+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}k(x_i,x_j)(a_i-\hat{a}_i)(a_j-\hat{a}_j)$$

$$-\sum_{i=1}^{N}\sum_{j=1}^{N}k(x_i,x_j)(a_i-\hat{a}_i)(a_j-\hat{a}_j)$$

$$\Rightarrow -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_i-\hat{a}_i)(a_j-\hat{a}_j)k(x_i,x_j)$$

$$-\varepsilon\sum_{i=1}^{N}(a_i+\hat{a}_i)+\sum_{i=1}^{N}(a_i-\hat{a}_i)y_i\ ;\ \text{Hence proved.}$$

Ans 2).

2.1).

First I have made two list of lists containing image of 28*28 pixels and label belonging to that image. Later I selected those image and labels where label is equal to 3 or 8. 2000 samples per class from the training set belonging to 3 or 8 and a test set of 500 samples per class belonging to 3 or 8 from the testing set were selected. Then soft-margin linear SVM formulation was used and five-fold cross-validation with grid search was performed for finding an appropriate regularization parameter, C. Soft-margin SVM was trained from that C.

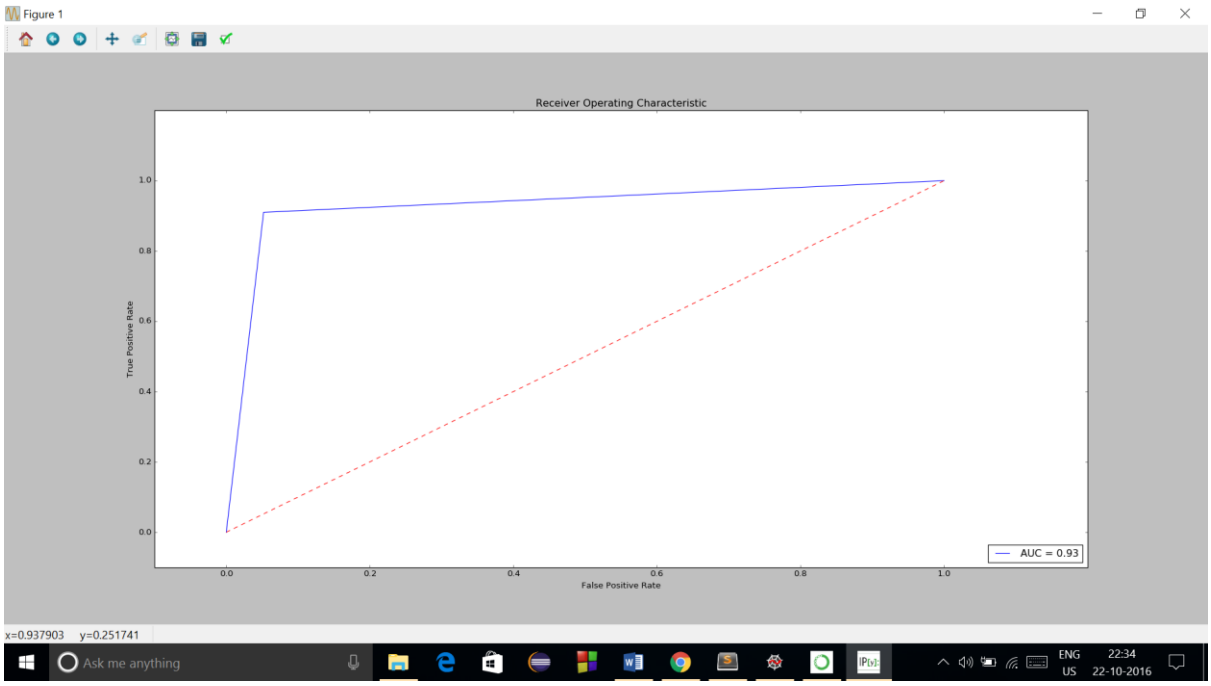
- grid.best_estimator_

SVC(C=0.1, cache_size=200, class_weight=None, coef0=0.0,

decision_function_shape=None, degree=3, gamma='auto', kernel='linear',

max_iter=-1, probability=False, random_state=None, shrinking=True,

tol=0.001, verbose=False)



C = 0.1, Accuracy = 0.51

C = 1, Accuracy = 0.5

C = 10, Accuracy = 0.5

C = 100, Accuracy = 0.5

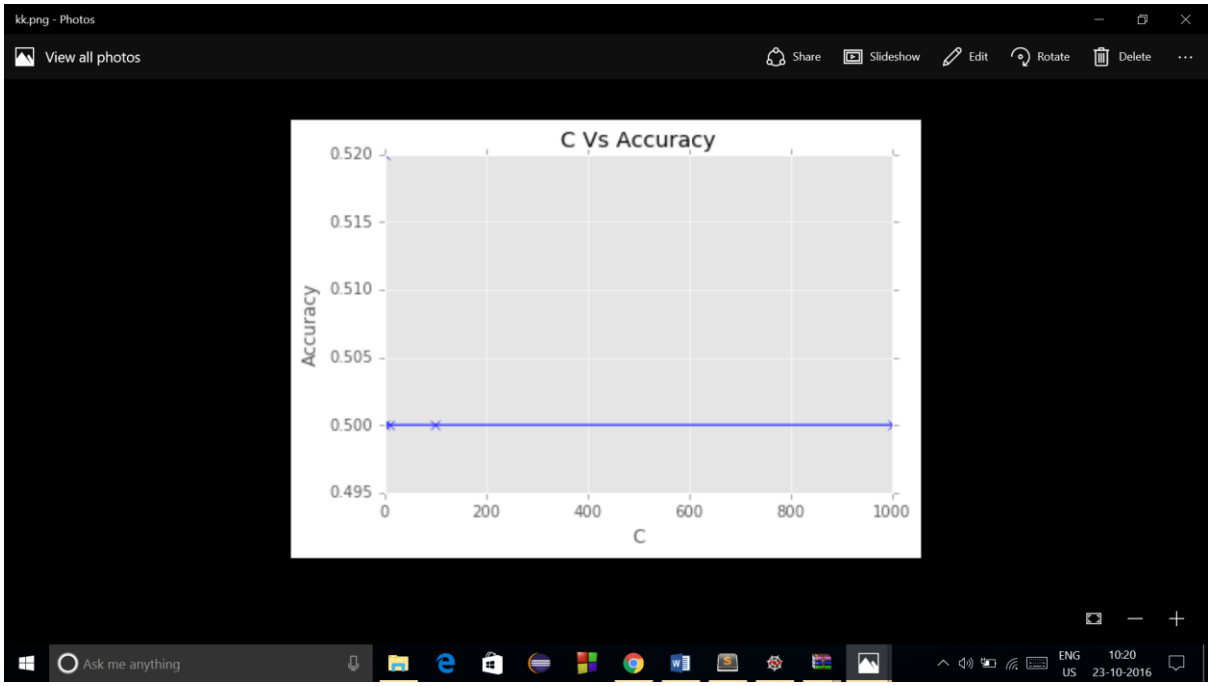
C = 1000, Accuracy = 0.5

param_grid = { 'kernel': ['linear'], 'C': [0.1, 1, 10, 100, 1000] }

svr = SVC()

grid = GridSearchCV(svr, param_grid, cv = 5)

grid.fit(trainimage, trainlabel)



2.2):

First I have made two list of lists containing image of 28*28 pixels and label belonging to that image. 2000 samples per class from the training set and a test set of 500 samples per class from the testing set were selected. Then soft-margin linear SVM formulation was used and five-fold cross-validation with grid search was performed for finding an appropriate regularization parameter, C. Soft-margin SVM was trained from that C. There was a gridsearch performed to find C.

param_grid = {

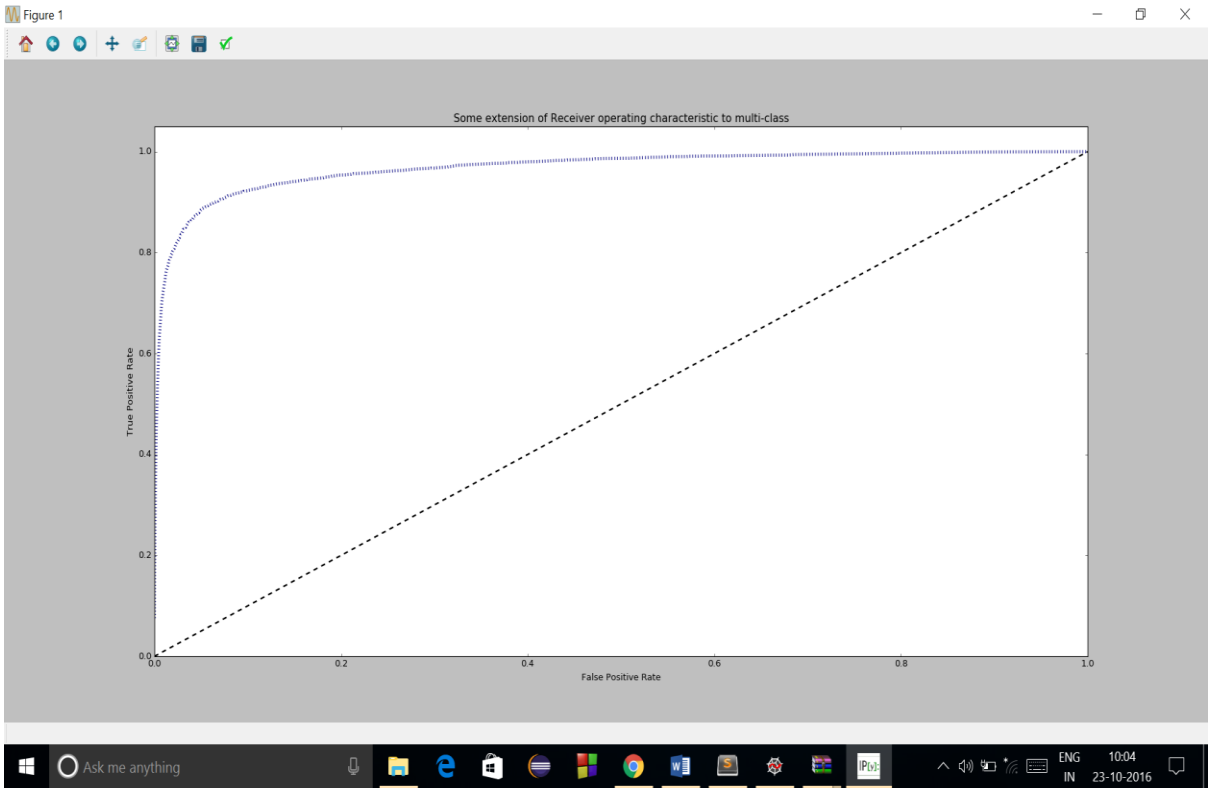
'estimator__C': [0.1, 1, 10, 100, 1000]

}

svr = OneVsRestClassifier(LinearSVC())

grid = GridSearchCV(svr, param_grid, cv = 5)

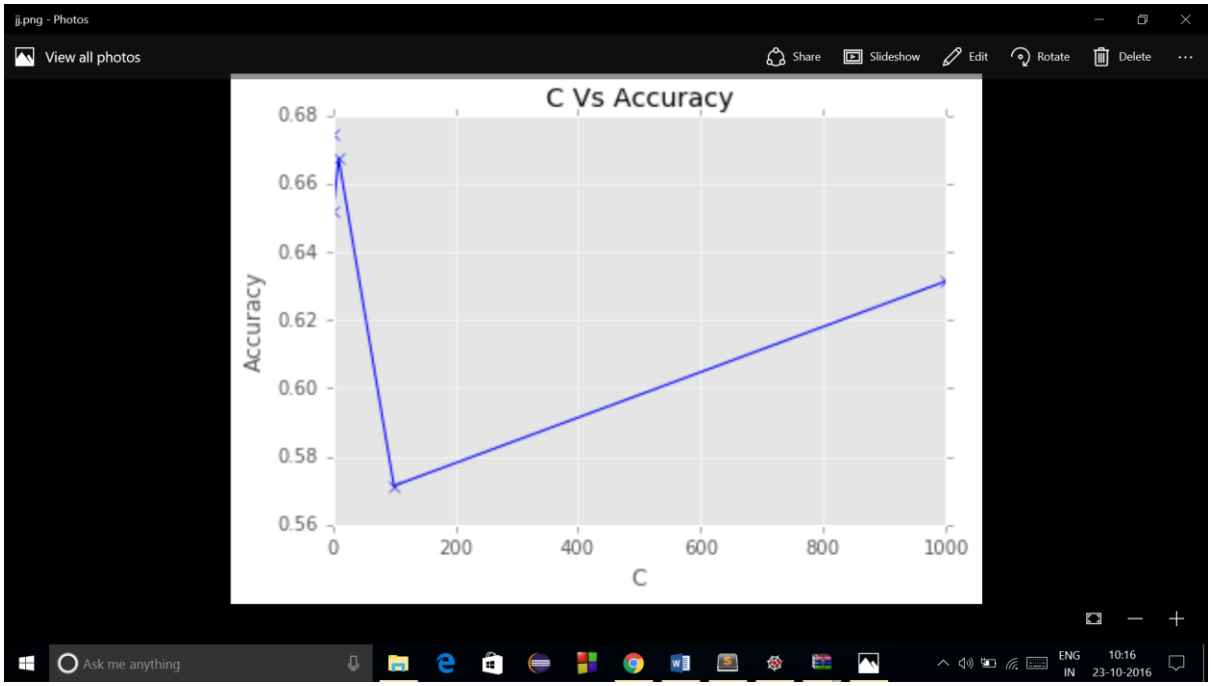
grid.fit(trainimage, trainlabel)



OneVsRestClassifier(estimator=LinearSVC(C=0.1, class_weight=None, dual=True, fit_intercept=True,
intercept_scaling=1, loss='squared_hinge', max_iter=1000,
multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
verbose=0),
n_jobs=1

roc_auc:

0.96792355555555565



C = 0.1, Accuracy = 0.6744

C = 1, Accuracy = 0.6516

C = 10, Accuracy = 0.6674

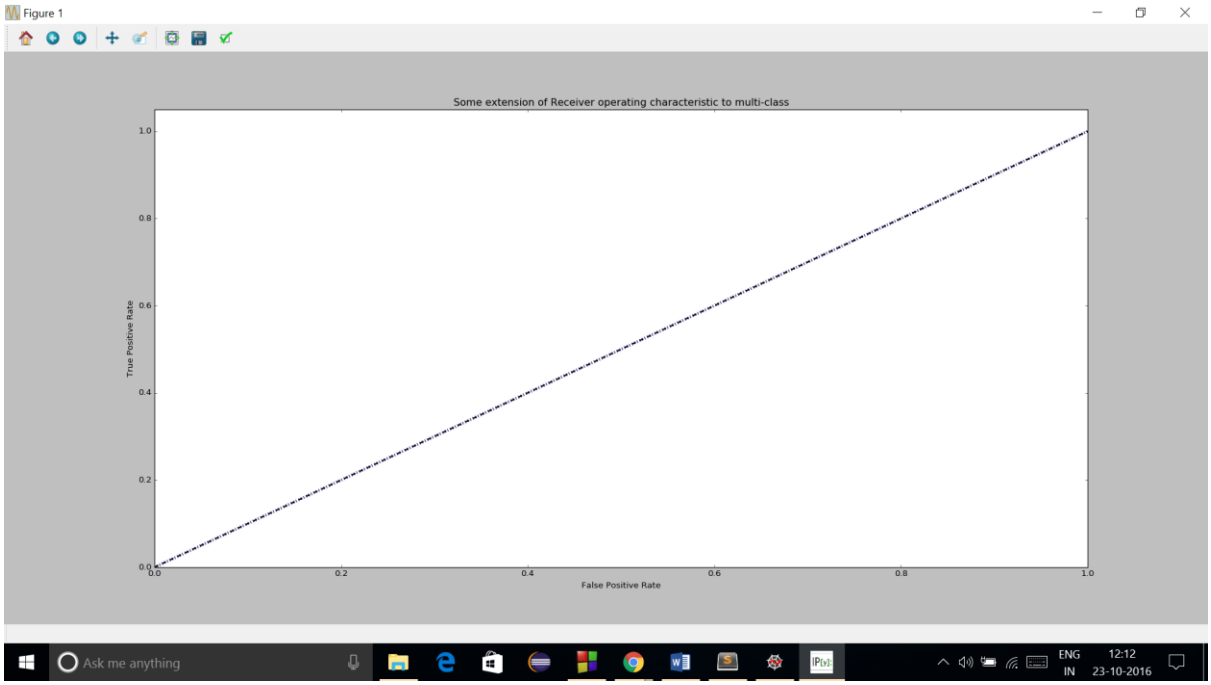
C = 100, Accuracy = 0.5714

C = 1000, Accuracy = 0.6314

Auc:

0: 0.99516133333333323, 1: 0.99492133333333321, 2: 0.96737022222222213, 3: 0.96847066666666659, 4: 0.98848044444444438, 5: 0.96452977777777782, 6: 0.98314533333333343, 7: 0.97833422222222222, 8: 0.88355511111111107, 9: 0.95334311111111114, 'macro': 0.96792355555555565}

2.3).



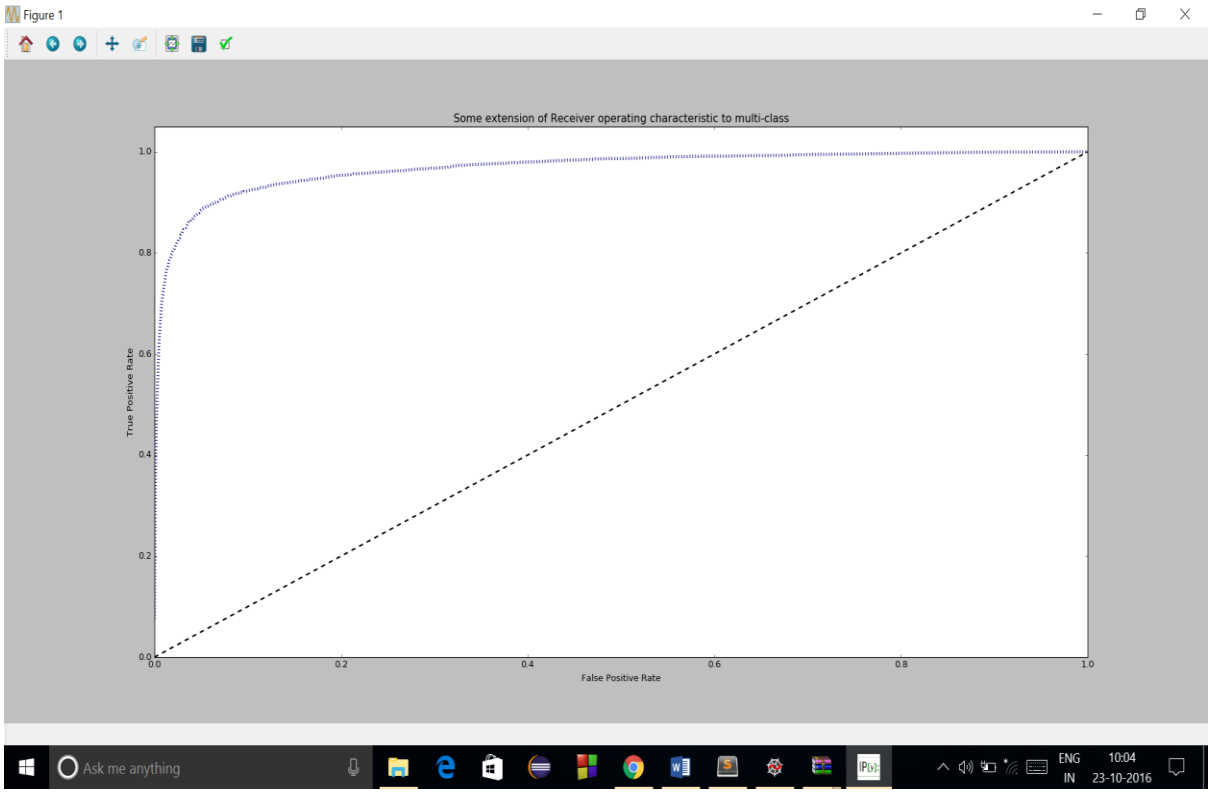
0: 0.5, 1: 0.5, 2: 0.5, 3: 0.5,

4: 0.5, 5: 0.5, 6: 0.5, 7: 0.5, 8: 0.5, 9: 0.5, 'macro': 0.51

C = 1, Gamma = 0.000001

I have chosen this C and gamma as others were giving me lower accuracy through function estimator.score().

Accuracy of 2 and 3 together:



Ans 3).

Chose 150 randomly selected samples per class from the training set. Then a test set of 100 samples per class was selected. Then KPCA and KNN was performed with k = 3. Five-fold cross-validation was performed with grid search to estimate the gamma. Code is provided in the source section.

gamma: [0.1, 1, 10, 100, 1000], then gamma = 0.1 as the best one.

0.290666666667

Predict_proba returns this:

```
array([[ 0.33333333,  0.33333333,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.33333333,  0.33333333,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.66666667,  0.33333333,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       ...,
       [ 0.33333333,  0.33333333,  0.        , ...,  0.        ,
         0.        ,  0.33333333],
```

```
[ 0.        , 0.33333333, 0.        , ..., 0.        ,
  0.        , 0.33333333],
[ 0.        , 0.33333333, 0.        , ..., 0.        ,
  0.        , 0.33333333]])
```