Assignment 4 EE/CS 5841

by Sarala Ravindra (sravindr@mtu.edu)

Difference between SVC and linear SVC multi class classification not complete

The multi-class handling in the modules SVC and LinearSVC ,both from the sklearn package ,is different.This is becuase in implementation the SVC utilizes 'libsvm' optimization while the LinearSVC makes use of 'liblinear' optimization.The 'libsvm' employs a binay classifier thus 'one versus one' approach is used if the number of classes is more than two in SVC.For instance if there are 3 classes 'A',B' and 'C' the model is trained to have a separate classifier for separtion of each A ,B and C from one another .For n class n*(n-1)/2 classifiers are employed in SVC.

The 'liblinear''s approch is one versus rest in multi class classsification thus for n class n classifiers are used in Linear SVC.Due to lesser classifiers with increase in number of classes the linear SVC computation requirements are lesser compared to the SVC model specially when the number of classes is too high.

MNIST data extraction and preprocessing

The MNIST dataset has 60000 train samples and 10000 test samples.The samples are hand written digits which are size-normalized and centered in a fixed-size image.Each image is 28 cross 28 in size.The below code extracts the MNIST data and is stored as train_images , test_images which are the train and test samples respectively and their corresponding labels are stored as train_lab and test_lab respectively.

To save computing time the data which is well behaved (learned from our my use of this dataset for my previous assignments)is randonmly sampled to 30,000 train samples.The data is also scaled to zero mean and unit variance dataset.This is very much necessary to achieve faster computation on my poor performing computer specially while using the SVC from sklearn.

In [2]:

```
%%time

import numpy as np


#  train and test data extraction

dataset_train = np.load("train_data.npz")


train_images = dataset_train['x']                       #train images
extraction
train_lab = dataset_train['y']                          #train labels extract
on
dataset_test = np.load("test_data.npz")
```

```
test_images = dataset_test['x']                          #test images extracti
n
test_lab = dataset_test['y']                             #test labels extracti
n


train_img = np.reshape(train_images,[60000,28*28])
train_lab = np.ravel(train_lab,order = 'C')
test_img = np.reshape(test_images,[10000,28*28])


##  scale data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
train_img = scaler.fit_transform(train_img)
test_img = scaler.transform(test_img)

## sample data

h = np.random.randint(0,60000,(10000))

train_img = train_img[h]
train_lab =train_lab[h]
```

```
Wall time: 3.69 s
```

Question 1 Linear SVM classification

linear SVM with varying C

The program below is written to output the training and testing accuracies of the linear SVM classifier built using the 'LinearSVC' from sklearn for various values of parameter 'C'.

In [112]:

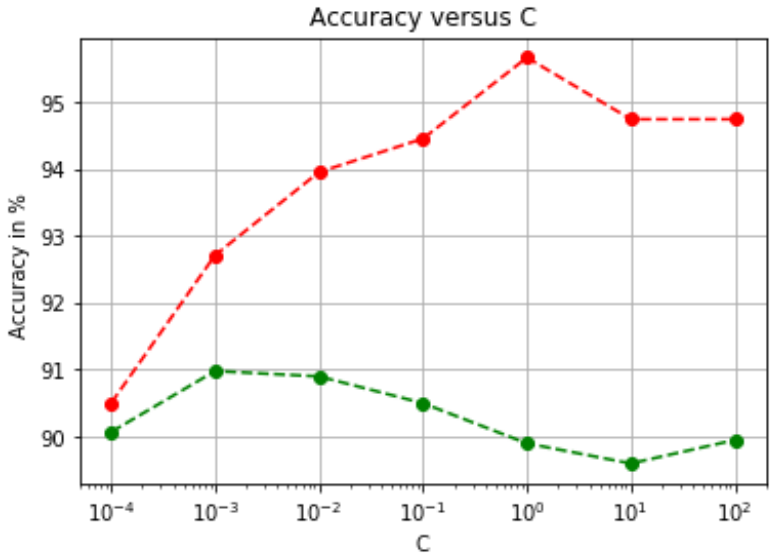
```
%%time
from sklearn.svm import LinearSVC
import numpy as np

Cs = [0.0001 , 0.001 , 0.01 , 0.1, 1, 10 ,100]

## training accuracies  for various C values

## testing accuracies  for various C values
test_correct =  np.zeros((7))
train_correct =  np.zeros((7))


for i in range(7):
    # define the classifier model
    clf = LinearSVC(C=Cs[i], class_weight=None, dual=False,
fit_intercept=True,
                    intercept_scaling=1, loss='squared_hinge', max_iter=100
```

```
,
                    multi_class='ovr', penalty='l2', random_state=0, tol=0.
001,
                    verbose=0)

    # fit data into the model
    clf.fit(train_img,train_lab)

    # pridict test labels
    predict_lab_test = clf.predict(test_img)
    predict_lab_train = clf.predict(train_img)



    # calculate test and train accuracies

    for l in range(10000):
        if(predict_lab_test[l] == test_lab[l]):
            test_correct[i] = test_correct[i]+1

    for l in range(30000):
        if(predict_lab_train[l] == train_lab[l]):
            train_correct[i] = train_correct[i]+1

    print('For C   = ',Cs[i],'the training accuracy
is',str.format('{0:.2f}',train_correct[i]/300),'% and the testing accuracy
is',str.format('{0:.2f}',test_correct[i]/100),'%')
```

```
For C   =  0.0001 the training accuracy is 90.50 % and the testing accuracy
is 90.07 %
For C   =  0.001 the training accuracy is 92.71 % and the testing accuracy i
s 90.98 %
For C   =  0.01 the training accuracy is 93.95 % and the testing accuracy is
90.90 %
For C   =  0.1 the training accuracy is 94.45 % and the testing accuracy is
90.50 %
For C   =  1 the training accuracy is 94.66 % and the testing accuracy is 89
.90 %
For C   =  10 the training accuracy is 94.74 % and the testing accuracy is 8
9.60 %
For C   =  100 the training accuracy is 94.74 % and the testing accuracy is
89.55 %
CPU times: user 52min 41s, sys: 548 ms, total: 52min 42s
Wall time: 52min 40s
```

In [139]:

```
import matplotlib.pyplot as plt
train_percent=np.divide(train_correct,300)
test_percent= np.divide(test_correct,100)
plt.plot(Cs,train_percent,'--ro',Cs,test_percent,'--go')
plt.xscale('log')
plt.xlabel('C')
plt.ylabel('Accuracy in % ')
plt.title('Accuracy versus C')
plt.grid(True)
plt.show()
```



plot: red line is the training accuracy green line is the testing accuracy

The parameter C is the regularization term associated with the errorsof classification in the objective function of the linear SVC design.The performane of Linear SVC classifier depends on the pick of maximum of the minimum margin between the hyper planes and the ablility to classify the test data with minimum misclassification.It's the parameter C which plays role with this aspect of the classifier.Thus it is important to choose best value of C.

Parameter C ia a kind of measure to the desire of achieving correct classification with the training dataset.Thatt is large value of C gives us higher accuracy classification scores on the training data whilw perform comparitively poor on the test data classification.This is because with increase in C value the minimum margin value of the model is reduced giving a overfit on the train data but reducing the generalization of the classifier model resuting in higher misclassification on the test data.Again smaller C means the model is designed too generally neglecting the train data properties ,this means both low training and testing error.This is because the train data is the base for for classification knowledge.

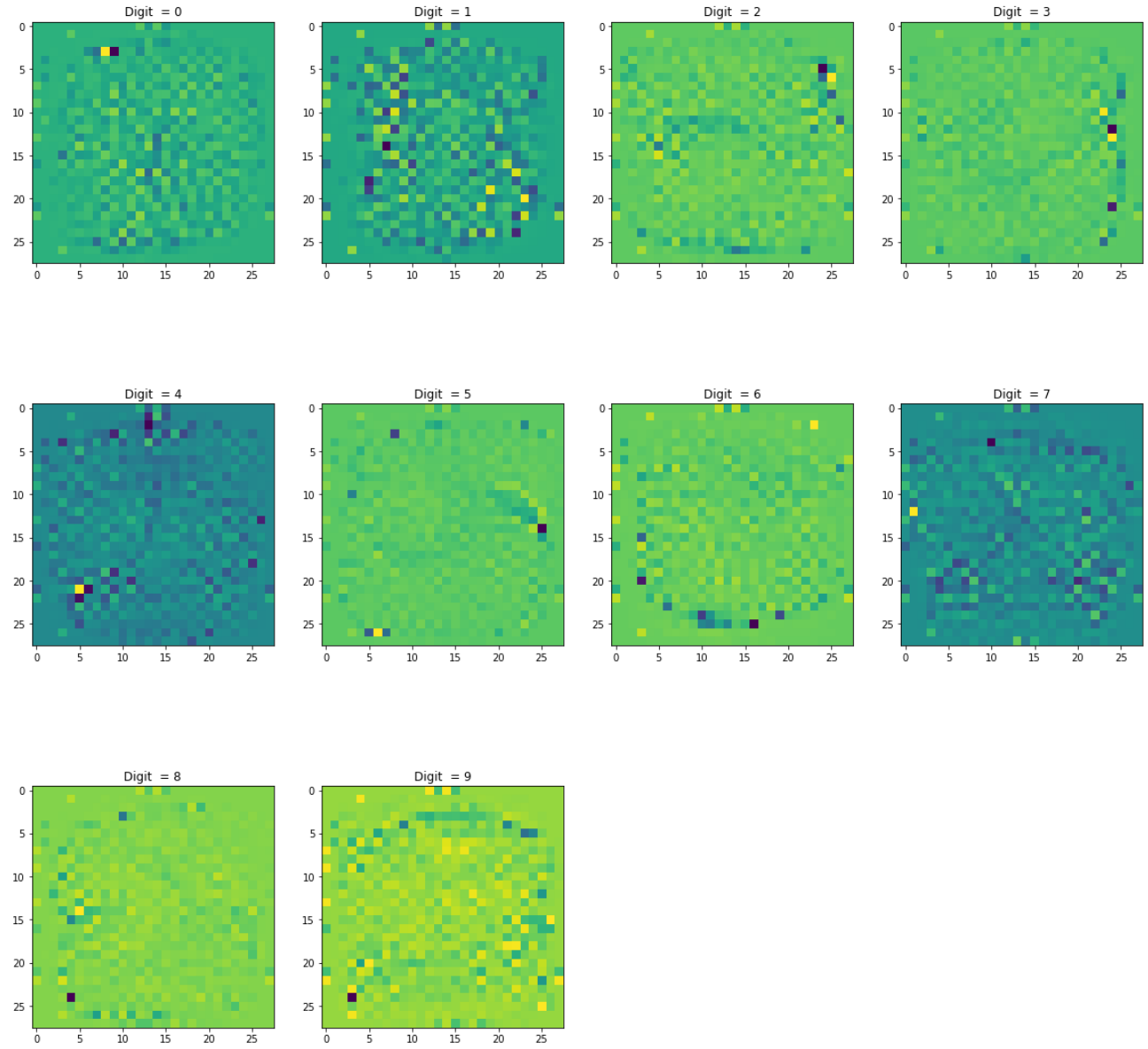
Thus,larger C means higher model complexity (overfitting) therefore higher variance and smaller bias and smaller C means underfiting implying lower variance and higher bias.

The accuracy versus C plot displayed above is in terms with the discussion above.ver small C values results in lower train and test accuracier ddue to high bias in the model while large C overfits the model giving high train data accuracy and low test accuracy.

From the plot it can be seen that the best value of C is 0.001 with test accuracy of 90.98 % and train accuracy of 92.71 % as the model seems to have achieved a trade of between the variance and bias.

In [158]:

```
import matplotlib.pyplot as plt
s=np.zeros((28,28))
plt.figure(figsize=(20,20))
for img in range(10):
    s=np.reshape(clf.coef_[img,:],((28,28)))
    plt.subplot(3,4,img+1)
    plt.imshow(s)
    plt.title('Digit  = %i' % img)
plt.show()
```



The above images give the weights assigned to each feature for a given class when compared to the rest of the digit (one vs rest scheme).Those pixels which contribute a lot in separating the digit from the rest of the digit is heavily weighed.

In comparision to the logistic regression in our previous assignment the linear SVM performs better in prediction.the logistic model is a probablistic approach which weighs the features taking their dependencies with other features for a class.The images of the weights per class in logistic regression model shows a spread over region where the digit is 'likly' to occur when written free hand.while the images produced by the linear SVC model here gives the weights of the features of the support vectors per class.Thus , the images generated for the logistic and the Linear SVC look similar but they are not the same.

Unlike the logistic regression model where each feature plays role in classification decsion only a subset of data points called the support vectors takes part in core classification decsion model.It is for this reason the Linear SVC is doing better than logistic in classification digits.

Q1 with penalty l1

In [159]:

```
%%time
```

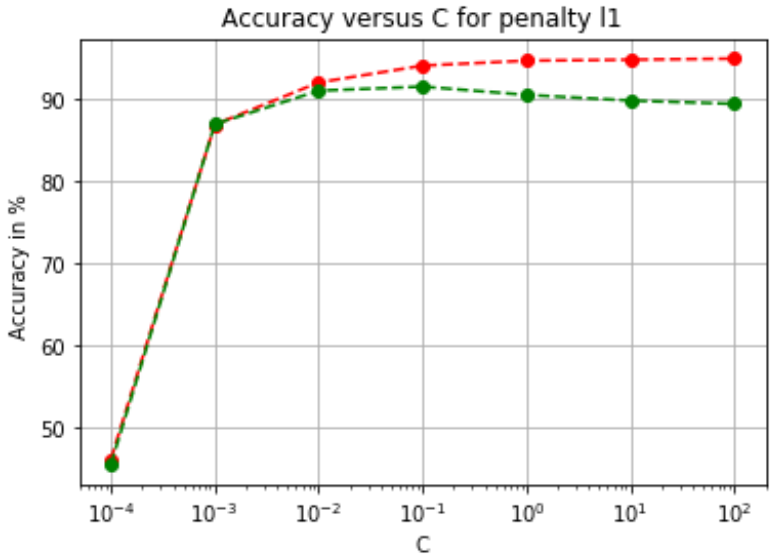
```
%%time
from sklearn.svm import LinearSVC
import numpy as np

Cs = [0.0001 , 0.001 , 0.01 , 0.1, 1, 10 ,100]

## training accuracies  for various C values

## testing accuracies  for various C values
test_correct =  np.zeros((7))
train_correct =  np.zeros((7))


for i in range(7):
    # define the classifier model
    clf = LinearSVC(C=Cs[i], class_weight=None, dual=False,
fit_intercept=True,
                    intercept_scaling=1, loss='squared_hinge', max_iter=100
,
                    multi_class='ovr', penalty='l1', random_state=0, tol=0.
001,
                    verbose=0)

    # fit data into the model
    clf.fit(train_img,train_lab)

    # pridict test labels
    predict_lab_test = clf.predict(test_img)
    predict_lab_train = clf.predict(train_img)



    # calculate test and train accuracies

    for l in range(10000):
        if(predict_lab_test[l] == test_lab[l]):
            test_correct[i] = test_correct[i]+1

    for l in range(30000):
        if(predict_lab_train[l] == train_lab[l]):
            train_correct[i] = train_correct[i]+1

    print('For C  = ',Cs[i],'the training accuracy
is',str.format('{0:.2f}',train_correct[i]/300),'% and the testing accuracy
is',str.format('{0:.2f}',test_correct[i]/100),'%')
```

```
For C  =  0.0001 the training accuracy is 46.18 % and the testing accuracy
is 45.57 %
For C  =  0.001 the training accuracy is 86.91 % and the testing accuracy i
s 86.96 %
For C  =  0.01 the training accuracy is 92.04 % and the testing accuracy is
91.03 %
For C  =  0.1 the training accuracy is 94.08 % and the testing accuracy is
91.53 %
For C  =  1 the training accuracy is 94.68 % and the testing accuracy is 90
.52 %
For C  =  10 the training accuracy is 94.80 % and the testing accuracy is 8
9.83 %
```

```
For C  =  100 the training accuracy is 94.93 % and the testing accuracy is
89.43 %
CPU times: user 1h 15min 40s, sys: 2.02 s, total: 1h 15min 42s
Wall time: 1h 15min 40s
```

In [160]:

```
import matplotlib.pyplot as plt
train_percent=np.divide(train_correct,300)
test_percent= np.divide(test_correct,100)
plt.plot(Cs,train_percent,'--ro',Cs,test_percent,'--go')
# plt.axis([.0001,100, 60, 100])
plt.xscale('log')
plt.xlabel('C')
plt.ylabel('Accuracy in % ')
plt.title('Accuracy versus C for penalty l1')
plt.grid(True)
plt.show()
```



plot: red line is the training accuracy green line is the testing accuracy

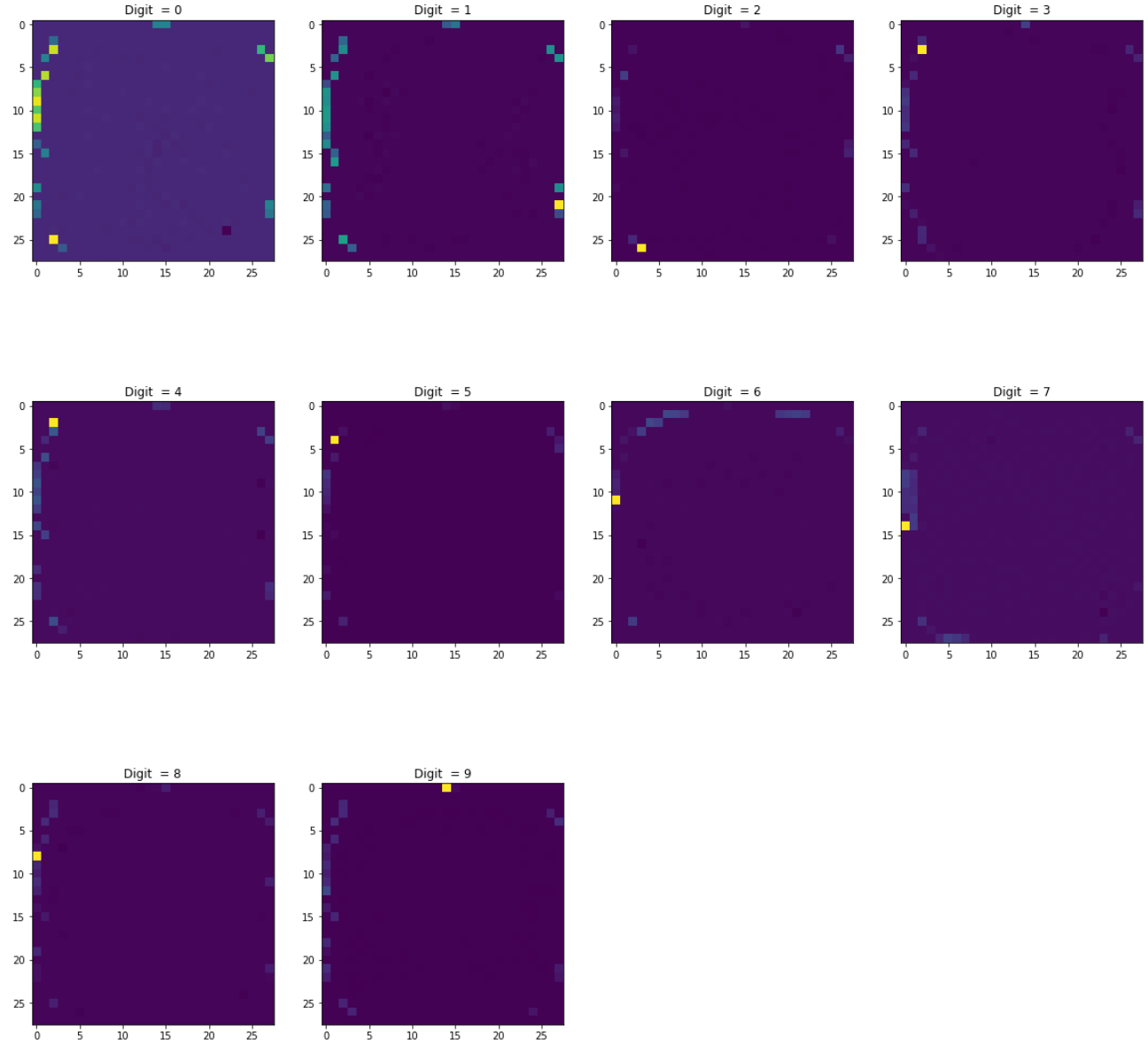
The classification model is penalised (constrained) to achieve some kind of regularization so as to nottooverfit the model.Previously l2 norm was used and the plot above is for the l1 norm.

The l1 norm can yield reduced model by eliminating of features (driving the features to zero).While l2 only shrinks the feature coefficients and does not eliminate them.l1 is good when there are large number of features which do not significantlyy contribute to the decision model which is the case here.l1 performs better than l2 penalization with l1's best test accuracy being 91.53% for C=0.1 and l2's best accuracy being 90.08 % for C=0.001. As in the previous plot of accuracy versus C ,the classifier has higher variance and lower bias for large C and lower variance and higher bias for smaller C. Based on the above plot I would choose C=0.1 with test accuracy 91.53% and train accuracy 94.08 %.

In [161]:

```
import matplotlib.pyplot as plt
s=np.zeros((28,28))
plt.figure(figsize=(20,20))
for img in range(10):
    s=np.reshape(clf.coef_[img,:],((28,28)))
```

```
    plt.subplot(3,4,img+1)
    plt.imshow(s)
    plt.title('Digit  = %i' % img)
plt.show()
```



The above images show the weight of the features of the support vectors for each class.Note that compared to the images generated with l2 penalty ,the l1 penalty images have a lot of feature weights driven to zero given a sparse decsion model.Thus The image is a lot blue (more zeros in feature weights) compared with the l2 penalty images.

Question 2 SVC with different kernels

Due to computation time constraints the train data which was reduced to 30,000 samples is further reduced down to 10,000 samples.

rbf kernel SVM

In [5]:

```
%%time
```

```
#Import Libraries

from sklearn.svm import SVC
import matplotlib.pyplot as plt

# Create SVM classification object


Cs =  [0.0001 , 0.001 , 0.01 , 0.1, 1, 10 ,100]
gammas =  [0.0001 , 0.001 , 0.01 , 0.1, 1, 10 ,100]

test_correct =  np.zeros((7,7))
train_correct =  np.zeros((7,7))

for cr in range(7):
    for gr in range(7):

        clf = SVC(C=Cs[cr], cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma=gammas[gr], kern
el='rbf',
            max_iter=10000, probability=False, random_state=None,
shrinking=True,
            tol=0.001, verbose=False)

        clf.fit(train_img,train_lab)

        # pridict test labels
        predict_lab_test = clf.predict(test_img)
        predict_lab_train = clf.predict(train_img)



        # calculate test and train accuracies

        for l in range(10000):
            if(predict_lab_test[l] == test_lab[l]):
                test_correct[cr,gr] = test_correct[cr,gr]+1

        for l in range(10000):
            if(predict_lab_train[l] == train_lab[l]):
                train_correct[cr,gr] = train_correct[cr,gr]+1

        print('For C  = ',Cs[cr],'and gamma = ',gammas[gr],'the training ac
curacy is',str.format('{0:.2f}',train_correct[cr,gr]/100),'% and the testin
g accuracy is',str.format('{0:.2f}',test_correct[cr,gr]/100),'%')
```

```
For C  =  0.0001 and gamma =  0.0001 the training accuracy is 11.09 % and t
he testing accuracy is 11.35 %
For C  =  0.0001 and gamma =  0.001 the training accuracy is 11.09 % and th
e testing accuracy is 11.35 %
For C  =  0.0001 and gamma =  0.01 the training accuracy is 11.09 % and the
testing accuracy is 11.35 %
For C  =  0.0001 and gamma =  0.1 the training accuracy is 11.09 % and the
testing accuracy is 11.35 %
For C  =  0.0001 and gamma =  1 the training accuracy is 11.09 % and the te
sting accuracy is 11.35 %
For C  =  0.0001 and gamma =  10 the training accuracy is 11.09 % and the t
esting accuracy is 11.35 %
```

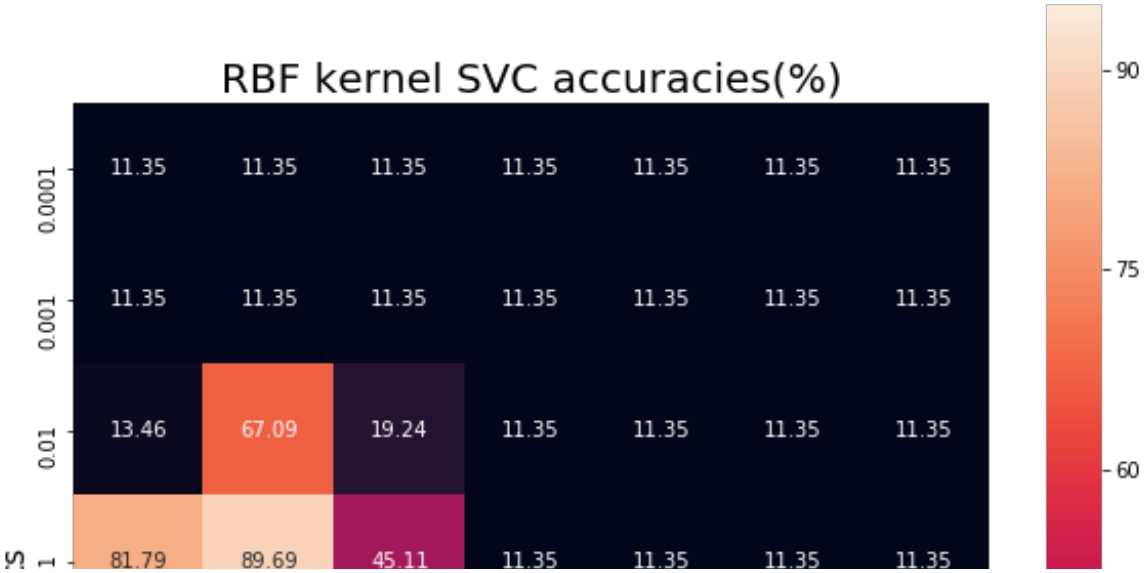
esting accuracy is 11.35 %
For C = 0.0001 and gamma = 100 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 0.0001 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 0.001 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 0.01 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 0.1 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 1 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 10 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.001 and gamma = 100 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.01 and gamma = 0.0001 the training accuracy is 13.37 % and the testing accuracy is 13.46 %
For C = 0.01 and gamma = 0.001 the training accuracy is 67.16 % and the testing accuracy is 67.09 %
For C = 0.01 and gamma = 0.01 the training accuracy is 19.17 % and the testing accuracy is 19.24 %
For C = 0.01 and gamma = 0.1 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.01 and gamma = 1 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.01 and gamma = 10 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.01 and gamma = 100 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.1 and gamma = 0.0001 the training accuracy is 81.61 % and the testing accuracy is 81.79 %
For C = 0.1 and gamma = 0.001 the training accuracy is 90.66 % and the testing accuracy is 89.69 %
For C = 0.1 and gamma = 0.01 the training accuracy is 48.15 % and the testing accuracy is 45.11 %
For C = 0.1 and gamma = 0.1 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.1 and gamma = 1 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.1 and gamma = 10 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 0.1 and gamma = 100 the training accuracy is 11.09 % and the testing accuracy is 11.35 %
For C = 1 and gamma = 0.0001 the training accuracy is 92.46 % and the testing accuracy is 91.61 %
For C = 1 and gamma = 0.001 the training accuracy is 97.75 % and the testing accuracy is 94.06 %
For C = 1 and gamma = 0.01 the training accuracy is 99.99 % and the testing accuracy is 77.81 %
For C = 1 and gamma = 0.1 the training accuracy is 100.00 % and the testing accuracy is 17.47 %
For C = 1 and gamma = 1 the training accuracy is 100.00 % and the testing accuracy is 11.35 %
For C = 1 and gamma = 10 the training accuracy is 100.00 % and the testing accuracy is 11.35 %
For C = 1 and gamma = 100 the training accuracy is 100.00 % and the testing accuracy is 11.35 %
For C = 10 and gamma = 0.0001 the training accuracy is 96.48 % and the testing accuracy is 93.38 %

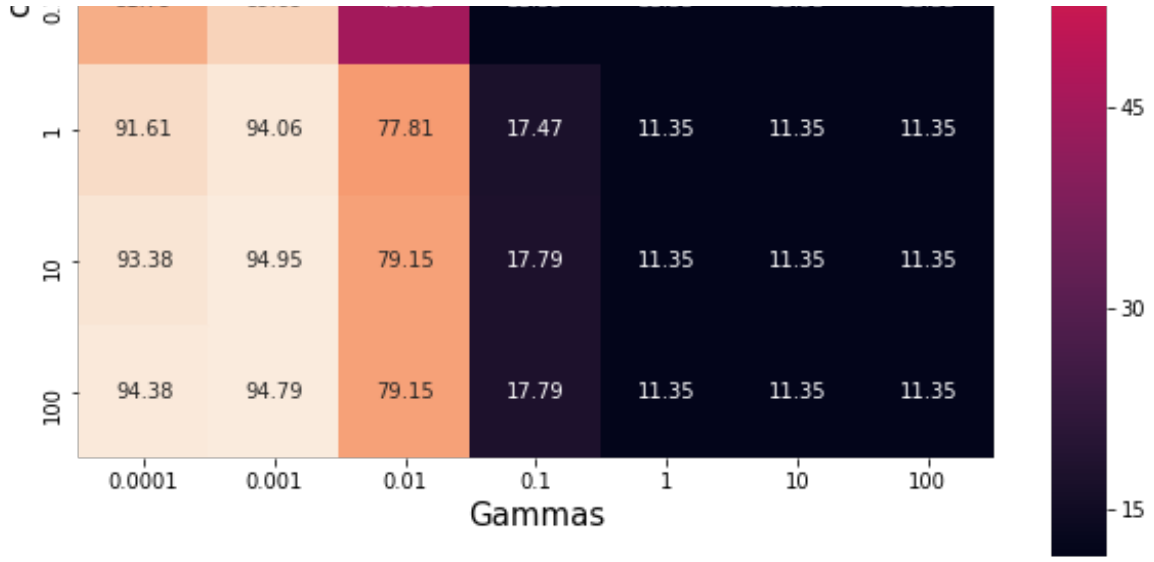
```
esting accuracy is 95.50 %
For C   =  10 and gamma =  0.001 the training accuracy is 99.94 % and the te
sting accuracy is 94.95 %
For C   =  10 and gamma =  0.01 the training accuracy is 100.00 % and the te
sting accuracy is 79.15 %
For C   =  10 and gamma =  0.1 the training accuracy is 100.00 % and the tes
ting accuracy is 17.79 %
For C   =  10 and gamma =  1 the training accuracy is 100.00 % and the testi
ng accuracy is 11.35 %
For C   =  10 and gamma =  10 the training accuracy is 100.00 % and the test
ing accuracy is 11.35 %
For C   =  10 and gamma =  100 the training accuracy is 100.00 % and the tes
ting accuracy is 11.35 %
For C   =  100 and gamma =  0.0001 the training accuracy is 99.44 % and the
testing accuracy is 93.38 %
For C   =  100 and gamma =  0.001 the training accuracy is 100.00 % and the
testing accuracy is 94.79 %
For C   =  100 and gamma =  0.01 the training accuracy is 100.00 % and the t
esting accuracy is 79.15 %
For C   =  100 and gamma =  0.1 the training accuracy is 100.00 % and the te
sting accuracy is 17.79 %
For C   =  100 and gamma =  1 the training accuracy is 100.00 % and the test
ing accuracy is 11.35 %
For C   =  100 and gamma =  10 the training accuracy is 100.00 % and the tes
ting accuracy is 11.35 %
For C   =  100 and gamma =  100 the training accuracy is 100.00 % and the te
sting accuracy is 11.35 %
Wall time: 5h 59min 56s
```

Testing and training accuracies in table

In [59]:

```
import seaborn as sns
plt.figure(figsize=(10,10))
RBF_kernel_test=test_correct/100
sns.heatmap(RBF_kernel_test, annot=True, fmt=".2f",square=True,xticklabels=
Cs, yticklabels=gammas)
plt.xlabel('Gammas',size=15)
plt.ylabel('cs',size=15)
plt.title('RBF kernel SVC accuracies(%)',size=20)
plt.show()
```



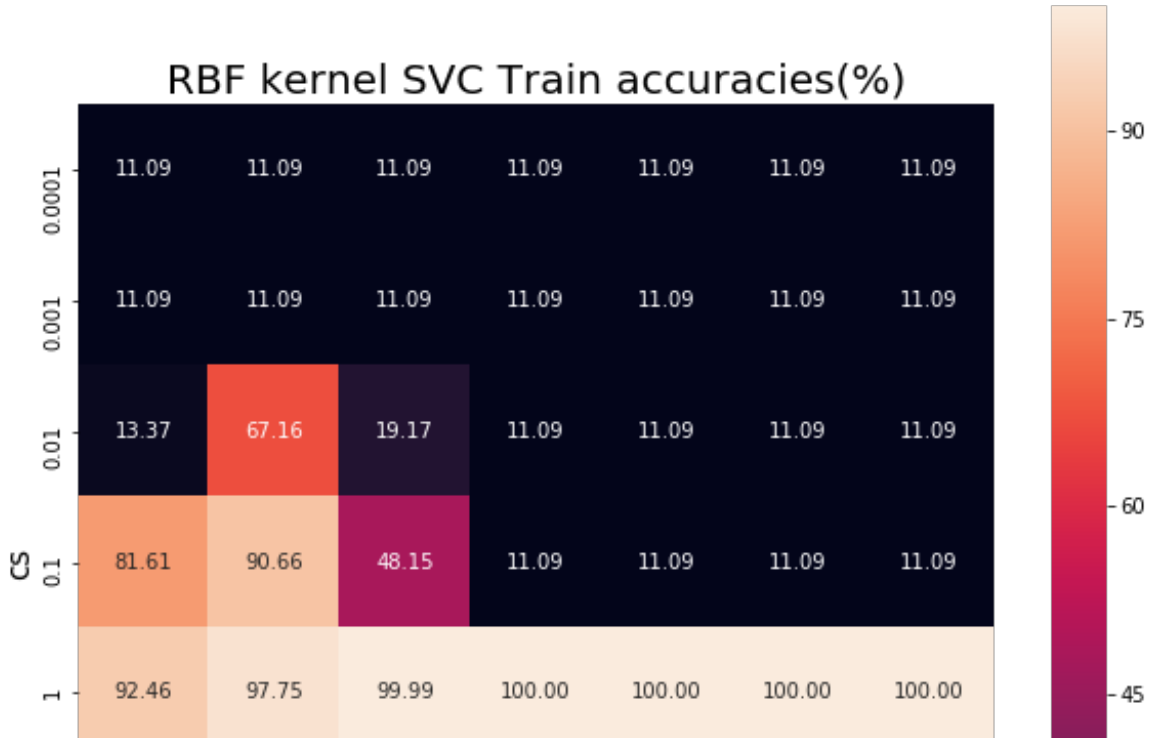


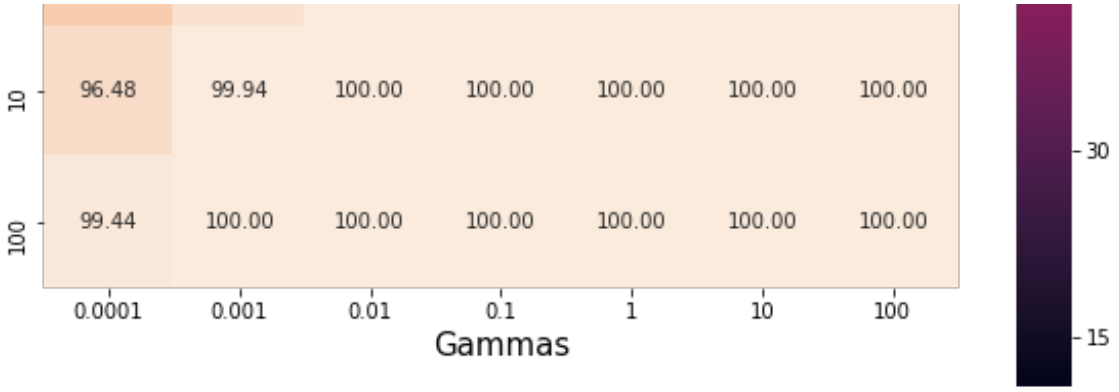
test accuracies for rbf kernel SVM

As gamma increases to a large value or C reduces to a very small value the test accuracies goes down due to high bias in the model from table above. the below table gives the train accuracies ,large c overfit the model while large gamma induces high bias in the model so the train accuracy is about 100 percent in that band.Very low C has less accuracy due to high bias.

In [72]:

```
import seaborn as sns
plt.figure(figsize=(10,10))
# RBF_kernel_train=train_correct/100
sns.heatmap(RBF_kernel_train, annot=True, fmt=".2f",square=True,xticklabels
=Cs, yticklabels=gammas)
plt.xlabel('Gammas',size=15)
plt.ylabel('cs',size=15)
plt.title('RBF kernel SVC Train accuracies(%)',size=20)
plt.show()
```





The parameters gamma and C are crucial in the SVM classification model performance. The gamma function gives the radius of influence of the support vector in the classification of any other data point in the dataset.Larger the value of gamma lower is the influencing range and vice-versa. As already discussed large C values overfit the train data thus the train accuracies is very high for largr Cs.Smaller C means less variance and high bias as there is less penalising of the misclassification of data. The classifier is more sensitive to gamma than C.Small gamma values constraint the model with support vector selection .So small gamma means larger variance and high bias.

Trade off between these regularisation parameters yeilds us the a testing accuracy of 94.95 % for C=10 and gamma 0.001.

In comparision with the linear SVC the radial basis function SVM performs better yeilding relatively higher accuracy.This is because the SVM has additional regularisation terms not just the C parameter.Here it is the rbf kernel gamma parameter which regulates the support vector influence on the decsion function.

I would choose C = 10 and gamma = 0.001 due to comparitvely better trade-off result

images using dual*coeff* and n_support using best parameters

In [73]:

```
%%time

#Import Libraries

from sklearn.svm import SVC

# Create SVM classification object

clf = SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=10000, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

clf.fit(train_img,train_lab)
```

```
Wall time: 33.9 s
```

In [74]:

```
print(clf.n_support_)
```

```
[264 159 444 454 389 457 288 442 438 430]
```

n*support* gives us the number of support vectors associated with each class.From the above result ,class O is associated with 264 support vectors in all of the n/(n-1)/2 (that's 45) classifiers and same goes with the rest of the classes.

from this we can say that alpha values from coloum 264 to (264+159)=425 are associated with digit 1

In [84]:

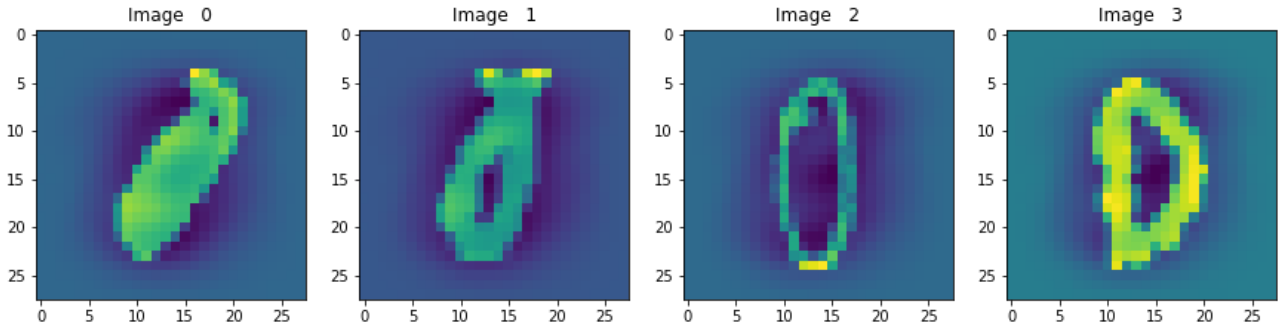
```
np.argmax(clf.dual_coef_,axis=1)
```

Out[84]:

```
array([118, 299, 299, 331,  85, 129, 267, 118, 264], dtype=int64)
```

In [100]:

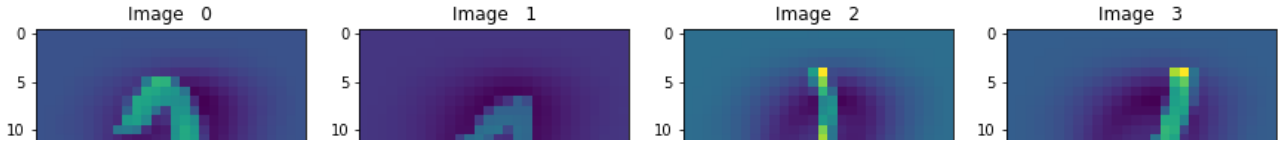
```
import matplotlib.pyplot as plt
s=np.zeros((28,28))
plt.figure(figsize=(15,15))
d = [118,129,85,100]
for img in range(4):
    s=np.reshape(clf.support_vectors_[d[img],:],((28,28)))
    plt.subplot(1,4,img+1)
    plt.imshow(s)
    plt.title('Image    %i' % img)
plt.show()
```

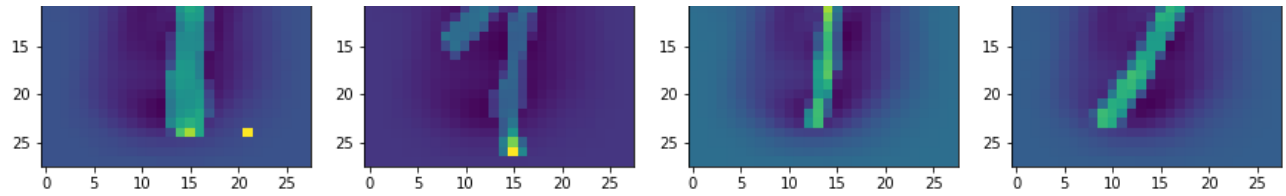


these support vectors rae associated with digit 0 classification.The image backgroud shows the digit with which digit 0 is being distinguished.With close look you can observe 3 in image 2,8 in image 3.

In [101]:

```
s=np.zeros((28,28))
plt.figure(figsize=(15,15))
d = [267,331,420,300]
for img in range(4):
    s=np.reshape(clf.support_vectors_[d[img],:],((28,28)))
    plt.subplot(1,4,img+1)
    plt.imshow(s)
    plt.title('Image    %i' % img)
plt.show()
```





In [ ]:

```
the same explanation goes here these vectors are the suppoort vectors assio
ciated with digit 1 with the other digit with whivh the support vector is a
ssociated with.
It looks like image 1 support vector is associated with digit 1 and digit 9
```

poly kernel SVM

In [7]:

```
%%time

#Import Libraries

from sklearn.svm import SVC
import matplotlib.pyplot as plt

# Create SVM classification object


C_ =  [0.0001 , 0.001 , 0.01 , 0.1, 1]
Ds =  [1,3,4,5,7]

test_correct =  np.zeros((5,5))
train_correct =  np.zeros((5,5))

for cr in range(5):
    for dr in range(5):

        pclf = SVC(C=C_[cr], degree=Ds[dr], kernel='poly',max_iter=10000)

        pclf.fit(train_img,train_lab)

        # pridict test labels
        predict_lab_test = pclf.predict(test_img)
        predict_lab_train = pclf.predict(train_img)



        # calculate test and train accuracies

        for l in range(10000):
            if(predict_lab_test[l] == test_lab[l]):
                test_correct[cr,dr] = test_correct[cr,dr]+1

        for l in range(10000):
            if(predict_lab_train[l] == train_lab[l]):
                train_correct[cr,dr] = train_correct[cr,dr]+1

        print('For C  = ',C_[cr],'and degree = ',Ds[dr],'the training accur
```

```
acy is',str.format('{0:.2f}',train_correct[cr,dr]/100),'% and the testing a
ccuracy is',str.format('{0:.2f}',test_correct[cr,dr]/100),'%')
```

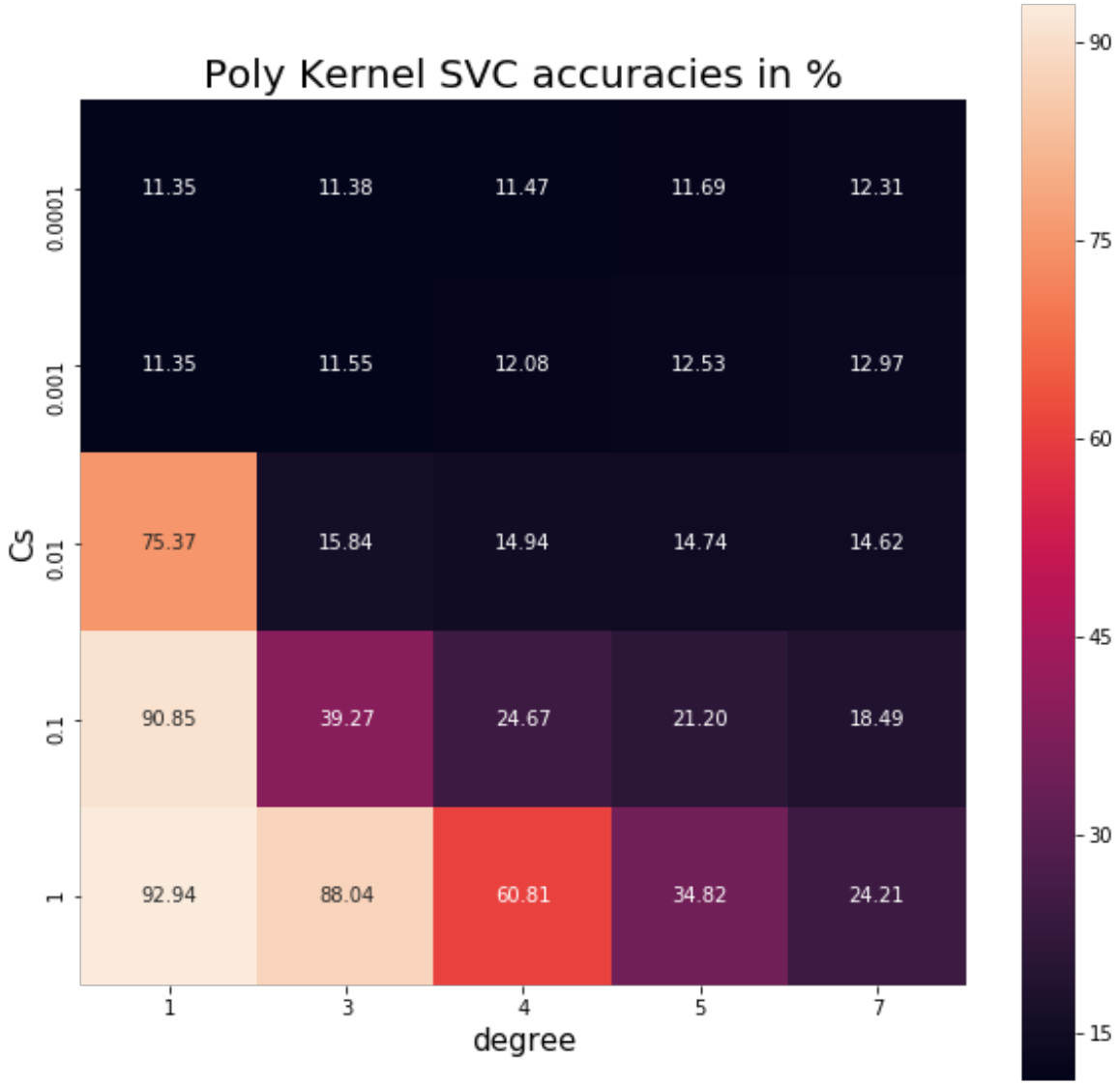
```
For C  =  0.0001 and degree =  1 the training accuracy is 11.09 % and the t
esting accuracy is 11.35 %
For C  =  0.0001 and degree =  3 the training accuracy is 11.28 % and the t
esting accuracy is 11.38 %
For C  =  0.0001 and degree =  4 the training accuracy is 11.53 % and the t
esting accuracy is 11.47 %
For C  =  0.0001 and degree =  5 the training accuracy is 12.04 % and the t
esting accuracy is 11.69 %
For C  =  0.0001 and degree =  7 the training accuracy is 13.14 % and the t
esting accuracy is 12.31 %
For C  =  0.001 and degree =  1 the training accuracy is 11.11 % and the te
sting accuracy is 11.35 %
For C  =  0.001 and degree =  3 the training accuracy is 11.64 % and the te
sting accuracy is 11.55 %
For C  =  0.001 and degree =  4 the training accuracy is 12.47 % and the te
sting accuracy is 12.08 %
For C  =  0.001 and degree =  5 the training accuracy is 13.17 % and the te
sting accuracy is 12.53 %
For C  =  0.001 and degree =  7 the training accuracy is 14.64 % and the te
sting accuracy is 12.97 %
For C  =  0.01 and degree =  1 the training accuracy is 75.60 % and the tes
ting accuracy is 75.37 %
For C  =  0.01 and degree =  3 the training accuracy is 16.76 % and the tes
ting accuracy is 15.84 %
For C  =  0.01 and degree =  4 the training accuracy is 16.48 % and the tes
ting accuracy is 14.94 %
For C  =  0.01 and degree =  5 the training accuracy is 16.94 % and the tes
ting accuracy is 14.74 %
For C  =  0.01 and degree =  7 the training accuracy is 18.27 % and the tes
ting accuracy is 14.62 %
For C  =  0.1 and degree =  1 the training accuracy is 91.39 % and the test
ing accuracy is 90.85 %
For C  =  0.1 and degree =  3 the training accuracy is 43.66 % and the test
ing accuracy is 39.27 %
For C  =  0.1 and degree =  4 the training accuracy is 28.64 % and the test
ing accuracy is 24.67 %
For C  =  0.1 and degree =  5 the training accuracy is 25.43 % and the test
ing accuracy is 21.20 %
For C  =  0.1 and degree =  7 the training accuracy is 23.94 % and the test
ing accuracy is 18.49 %
For C  =  1 and degree =  1 the training accuracy is 95.27 % and the testin
g accuracy is 92.94 %
For C  =  1 and degree =  3 the training accuracy is 91.85 % and the testin
g accuracy is 88.04 %
For C  =  1 and degree =  4 the training accuracy is 67.98 % and the testin
g accuracy is 60.81 %
For C  =  1 and degree =  5 the training accuracy is 43.85 % and the testin
g accuracy is 34.82 %
For C  =  1 and degree =  7 the training accuracy is 34.11 % and the testin
g accuracy is 24.21 %
Wall time: 3h 1min 1s
```

In [ ]:

```
# table generation giving accuracies for different degree and polynomial
```

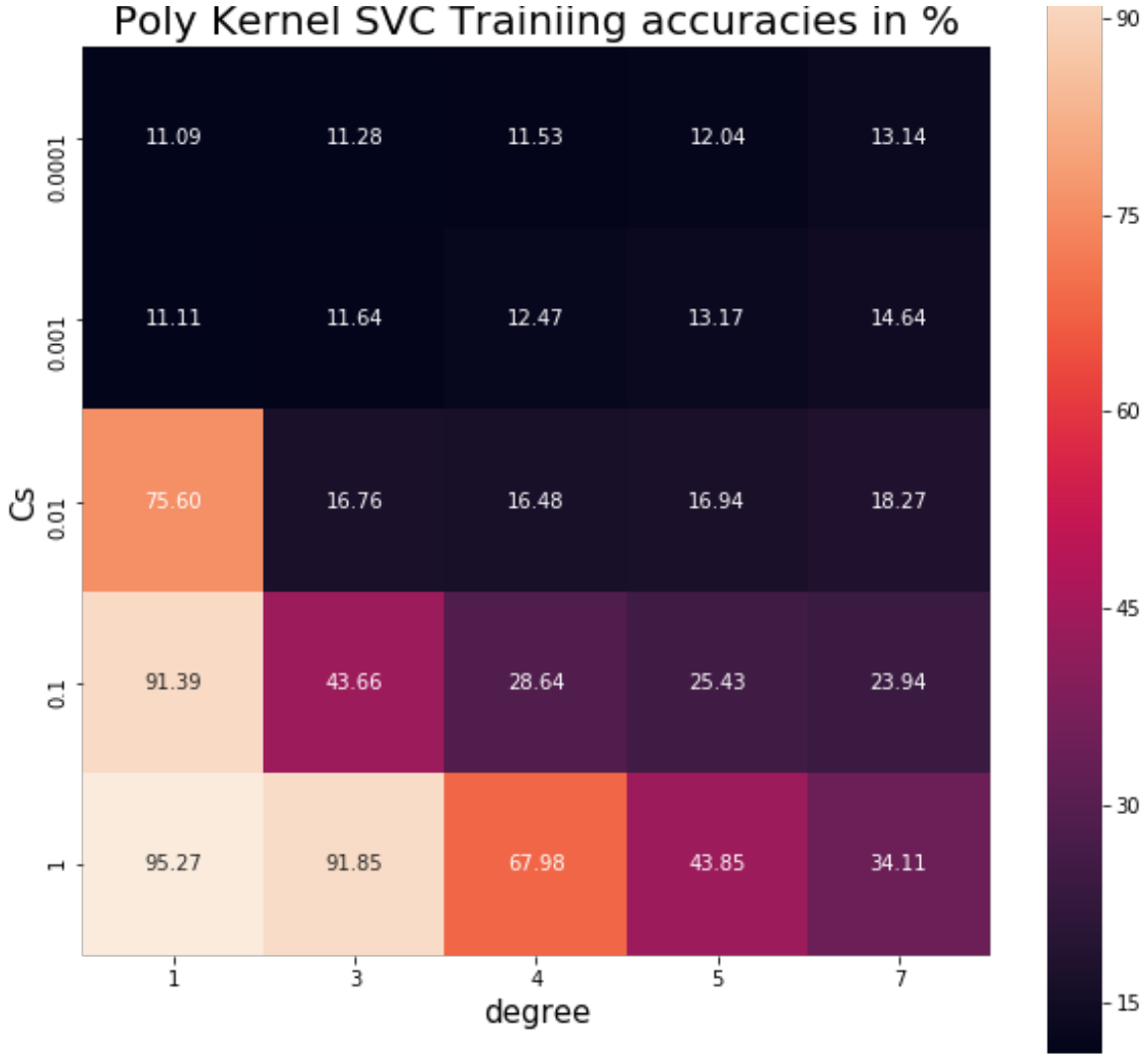
In [58]:

```
import seaborn as sns
plt.figure(figsize=(10,10))
poly_kernel_test=test_correct/100
sns.heatmap(poly_kernel_test, annot=True, fmt=".2f",square=True,xticklabels
=Ds, yticklabels=C_)
plt.xlabel('degree',size=15)
plt.ylabel('Cs',size=15)
plt.title('Poly Kernel SVC Testing accuracies in %',size=20)
plt.show()
```



In [70]:

```
import seaborn as sns
plt.figure(figsize=(10,10))
poly_kernel_train=train_correct/100
sns.heatmap(poly_kernel_train, annot=True,
fmt=".2f",square=True,xticklabels=Ds, yticklabels=C_)
plt.xlabel('degree',size=15)
plt.ylabel('Cs',size=15)
plt.title('Poly Kernel SVC Trainiing accuracies in %',size=20)
plt.show()
```



The results show that the data can be better classified when degree=1, meaning it is a dataset for which linear classifier can be used.At low C bands the test accuracy is the least due to the underfit.As C increase the accuracy gets better.As the given data seems to be linearly seprable(with soft margins) a degree of 1 seems to work the best.

I choose C = 1 and degree = 1 as best parameters

images using dual*coeff* and n_support using best parameters

In [93]:

```
print(pclf.n_support_)  # print number of support vector for each class
```

```
[260 257 385 448 415 480 283 382 495 499]
```

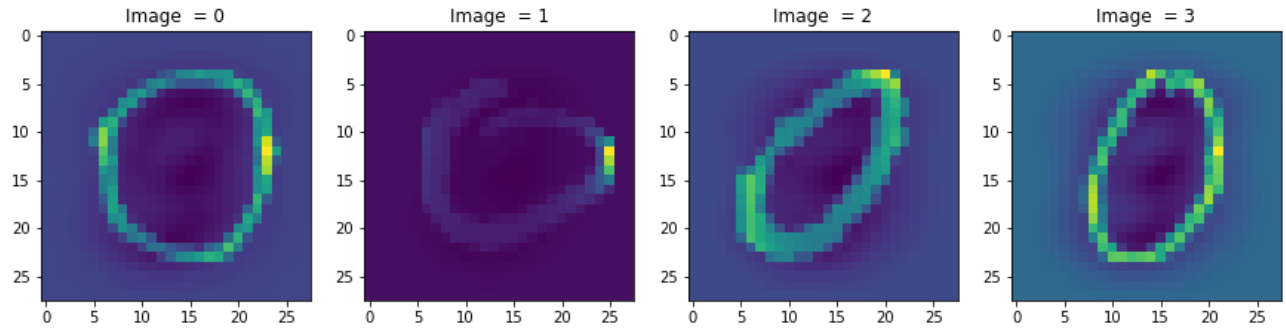
In [63]:

```
# fit poly SVM model
pclf = SVC(C=1, degree=1, kernel='poly',max_iter=10000)
pclf.fit(train_img,train_lab)
```

In [95]:

```
import matplotlib.pyplot as plt
```

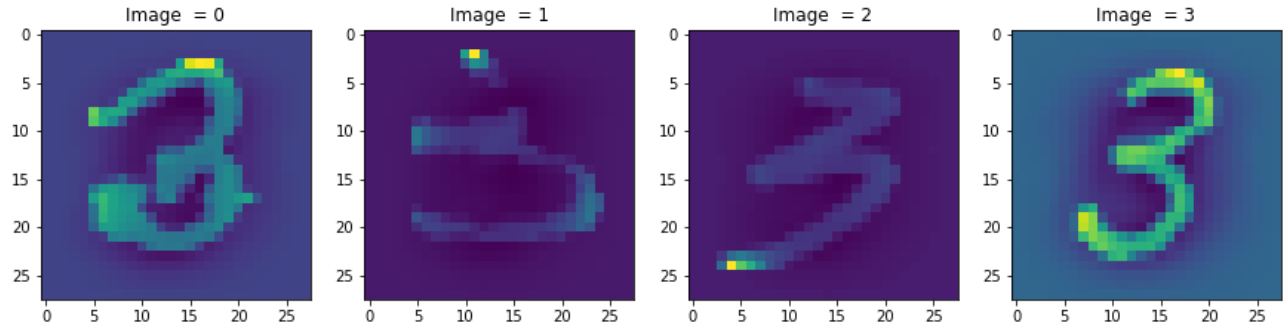
```
s=np.zeros((28,28))
plt.figure(figsize=(15,15))
d = [1,2,17,25]
for img in range(4):
    s=np.reshape(pclf.support_vectors_[d[img],:],((28,28)))
    plt.subplot(1,4,img+1)
    plt.imshow(s)
    plt.title('Image  = %i' % img)
plt.show()
```



support vectors associated with digit 0 shown.Look image 1 seems to be assosiated with digit 3 too

In [97]:

```
s=np.zeros((28,28))
plt.figure(figsize=(15,15))
d = [920,1000,1057,1129]
for img in range(4):
    s=np.reshape(pclf.support_vectors_[d[img],:],((28,28)))
    plt.subplot(1,4,img+1)
    plt.imshow(s)
    plt.title('Image  = %i' % img)
plt.show()
```



support vectors associated with digit 3 shown.Look image 3 seems to be assosiated with digit 9 too.Thus support vectors carry features weighed appropriately to distinguish one class from the other.