Assignment-2

Sai teja Ramadev

2015EE10466

Support vector Machine Assignment

Binary classification of first two classes with first 10 features: **svm_image_classifier.py**

```
C_range = 10. ** np.arange(-1, 3)
gamma_range = 10. ** np.arange(-8, -3)
degree_range = np.arange(1,4)
```

with linear kernel:

train accuracy = 0.99,test accuracy = 0.983333333333,same for all C values

With poly: used grid search for optimal values.

optimal values: C=0.001, gamma=0.001, degree=1, train score =0.991666666667, test score=1.0

after compiling svm_image_classifier.py code you will see output as below,

accuracy: 0.98958, params: {'kernel': 'poly', 'C': 1.0, 'degree': 3, 'max_iter': 100, 'coef0': -16, 'gamma': 9.9999999999999995e-08}

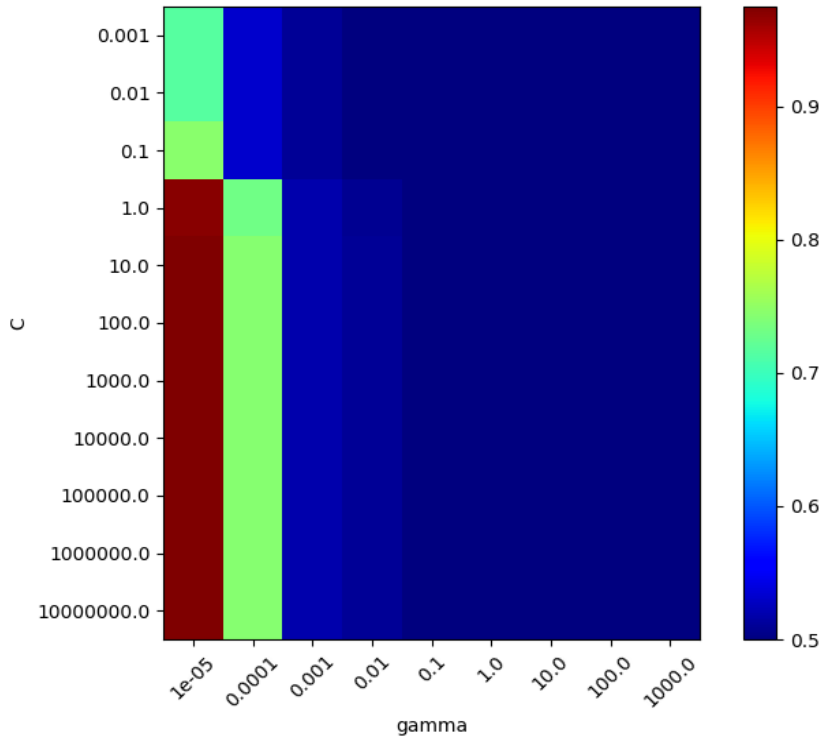
accuracy: 0.98125, params: {'kernel': 'poly', 'C': 1.0, 'degree': 3, 'max_iter': 100, 'coef0': -16, 'gamma': 9.999999999999995e-07}

So we can see that as gamma is increasing validation error increasing which means overfitting

With rbf: used grid search to obtain optimal values

C=10.0, gamma=0.00001, train score =1, test score= 0.983333333333.

C vs gamma



Binary classification of first two classes with all features: **svm_all_features_binary.py**

```
C_range = 10. ** np.arange(-3, 8)
gamma_range = 10. ** np.arange(-5, 4)
degree_range = np.arange(1,10)
```

with linear kernel:

train accuracy = 0.75,test accuracy = 0.746,same for all C values

With poly: used grid search for optimal values.Optimal values are C=0.001, gamma=0.001, degree=3,train accuracy= 0.997916666667, test accuracy = 0.991666666667

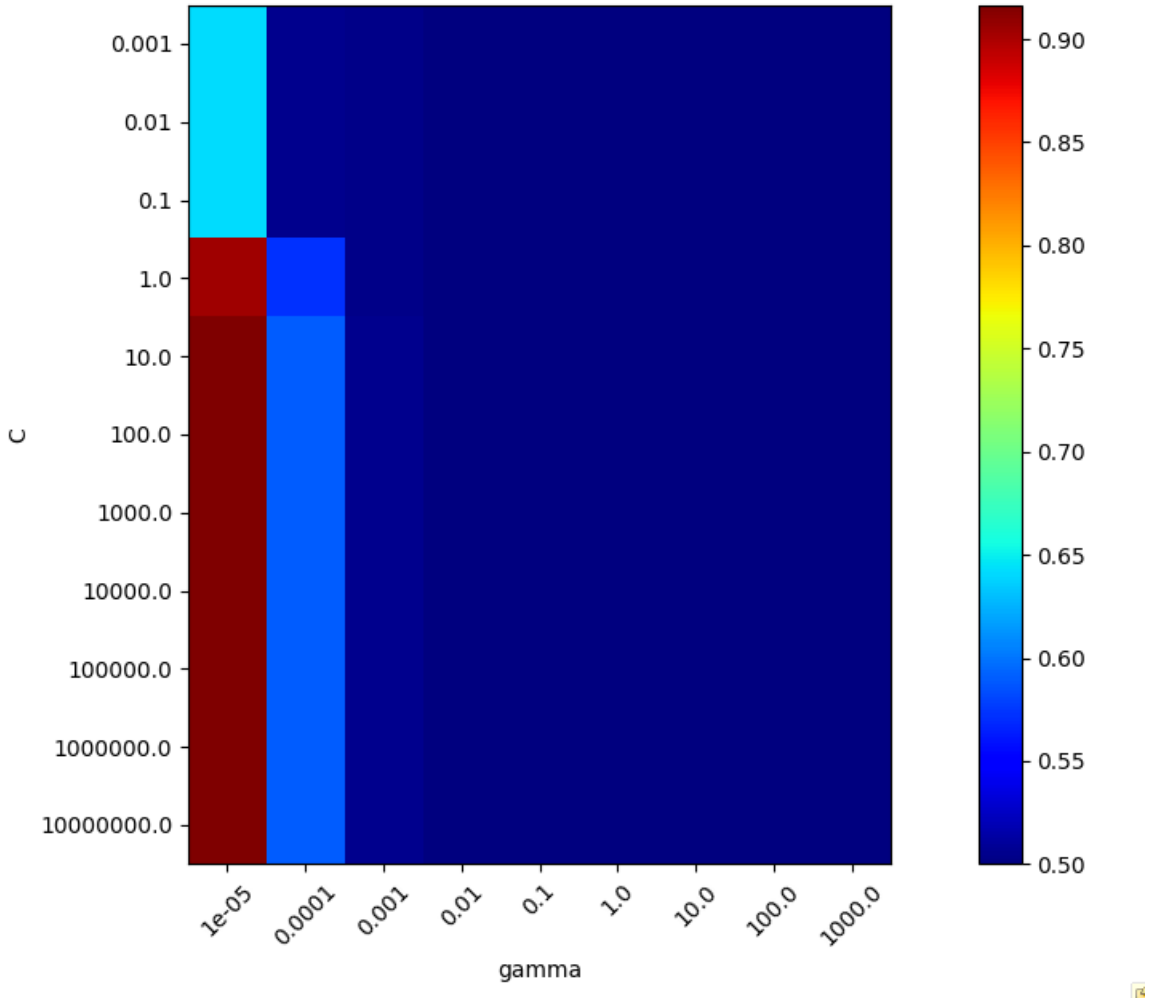
mean: 0.97292, params: {'kernel': 'poly', 'C': 10000000.0, 'degree': 2, 'max_iter': 10, 'coef0': -7, 'gamma': 0.001}

mean: 0.98750, params: {'kernel': 'poly', 'C': 10000000.0, 'degree': 2, 'max_iter': 10, 'coef0': -7, 'gamma': 0.0001}

So we can see that as gamma is increasing validation error increasing which means underfitting

With rbf: C=10.0, gamma=0.00001,train accuracy= 0.916666666667, test accuracy=0.925

C vs gamma



Multi-Classification with 10 features: **svm_multi_10features.py**

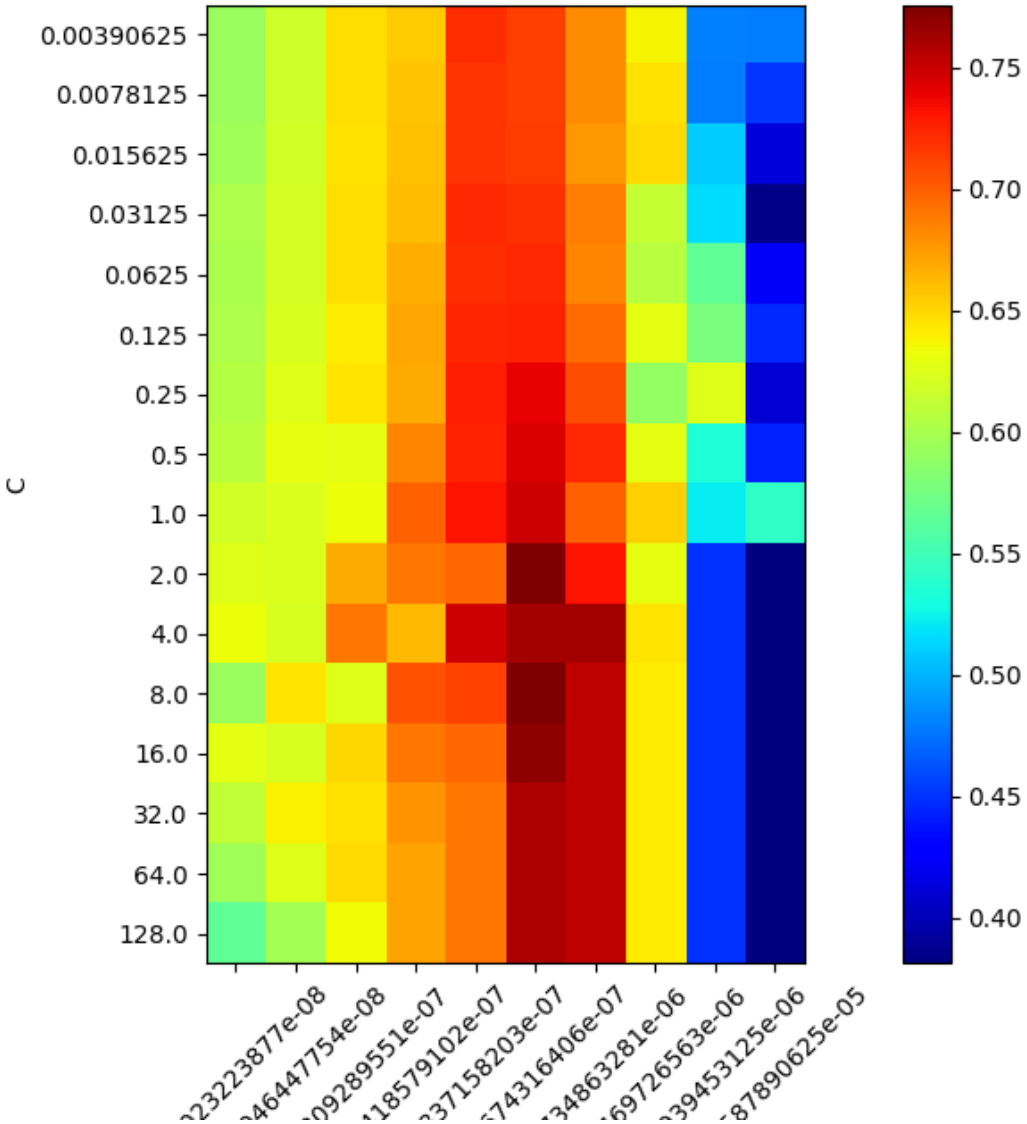
```
C_range = 2. ** np.arange(-8, 8)
gamma_range = 2. ** np.arange(-25, -15)
degree_range = np.arange(1,20)
```

With linear:train accuracy = 0.41,test accuracy = 0.39, almost same for all C values

with poly:C=0.03125, gamma=9.5367e-07,training accuracy=0.67,testing accuracy = 0.72,degree=3

with rbf: train accuracy = 0.770833333333,test accuracy = 0.74, C=8.0, gamma=9.5367431640625e-07

C vs gamma



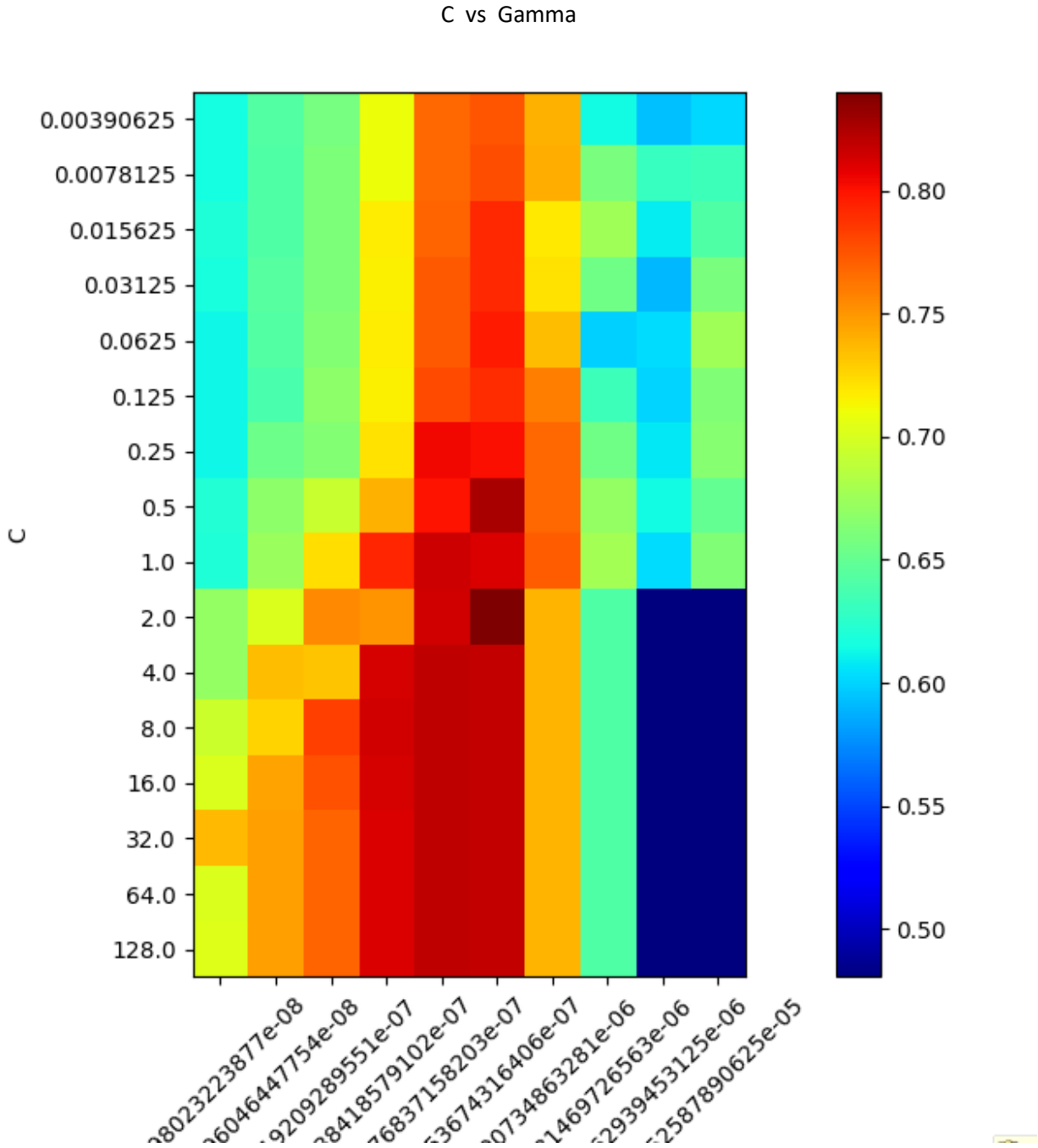
Multi-Classification with all features: **svm_multiu.py**

```
C_range = 2. ** np.arange(-8, 8)
gamma_range = 2. ** np.arange(-25, -15)
degree_range = np.arange(1,20)
```

With linear: Training Accuracy = 0.63, Testing accuracy = 0.57, almost no change for all values of C
with poly: C=0.015625, gamma=9.5367e-07,training accuracy=0.77,testing accuracy = 0.72,degree=5

With rbf: C= 2.0,gamma= 9.5367431640625e-07,Training Accuracy = 0.844166666667,

Testing accuracy = 0.821666666667.



Part -2: part2.py

with linear kernel: Training Accuracy = 0.52340416213, Testing accuracy = 0.51215484446.

So linear kernel is underfitting or unable to learn the model accurately no matter what are the values of c. This is because of large dimensions of data

The below are the examined C, gamma and degree (for poly kernel) ranges .

```
C_range = 2. ** np.arange(-5, 15)
gamma_range = 2. ** np.arange(-15, 3)
degree_range = np.arange(1,20)
```

With poly:C=0.03125, gamma=2.4414062e-05,training accuracy=0.805,testing accuracy = 0.79,degree=8

With rbf:

C=0.0625, gamma=6.103515625e-05,training accuracy=0.925,testing accuracy = 0.915

C vs gamma