Testing for Linear Kernel SVM.

```
IPython console
Console 1/A
In [75]: print("Best C value", bestpar,"Prediction accuracy: %0.2f" %metrics.accuracy_score(predicted, test_y))
Best C value 5.29831690628e-07 Prediction accuracy: 0.91

In [76]:
```

Use svm:LinearSVC module for a faster implementation than svm.SVC when testing for parameter C of the Linear Kernel

Approach:

1. Search for a range for testing using the GridsearchCV
2. Zoom on range until accuracy rises and falls.
3. Best parameter for C is at the peak

Test results for different C range:

For range detection

```
In [30]: clf = GS(svm.LinearSVC(),dict( C=np.logspace(-4,
0, 15)),n_jobs=-1).fit(train, train_y)

In [31]: means = clf.cv_results_['mean_test_score']
    ...: stds = clf.cv_results_['std_test_score']
    ...: for mean, std, params in zip(means, stds,
clf.cv_results_['params']):
    ...:     print("%0.3f (+/-%0.03f) for %r"
    ...:               % (mean, std * 2, params))
0.888 (+/-0.003) for {'C': 0.0001}
0.878 (+/-0.008) for {'C': 0.00019306977288832496}
0.875 (+/-0.005) for {'C': 0.00037275937203149379}
0.867 (+/-0.014) for {'C': 0.00071968567300115217}
0.861 (+/-0.006) for {'C': 0.0013894954943731374}
0.843 (+/-0.066) for {'C': 0.0026826957952797246}
0.858 (+/-0.002) for {'C': 0.0051794746792312128}
0.851 (+/-0.037) for {'C': 0.01}
0.864 (+/-0.019) for {'C': 0.019306977288832496}
0.851 (+/-0.019) for {'C': 0.037275937203149381}
0.842 (+/-0.025) for {'C': 0.071968567300115138}
0.852 (+/-0.011) for {'C': 0.13894954943731375}
0.859 (+/-0.008) for {'C': 0.26826957952797248}
0.855 (+/-0.010) for {'C': 0.51794746792312074}
0.846 (+/-0.011) for {'C': 1.0}

In [32]: clf.best_params_
Out[32]: {'C': 0.0001}
```

First Zoom on range

```
IPython console
Console 1/A
 [   0 1635   33 ...,    6   63   12]
 [  19   10 1237 ...,   21   50   14]
 ...,
 [   0    3   29 ..., 1449   15  108]
 [   8    5   50 ...,    6 1024   22]
 [   4    1    7 ...,   61   49 1186]]

In [52]: clf = svm.LinearSVC( C=0.0001).fit(train,train_y)

In [53]: predicted = clf.predict(test_x)
    ...: print("Prediction accuracy: ",metrics.accuracy_score(predicted, test_y))
Prediction accuracy:  0.89280714619

In [54]: clf = GS(svm.LinearSVC(),dict( C=np.logspace(-6, 0, 15)),n_jobs=-1).fit(train, train_y)

In [55]: means = clf.cv_results_['mean_test_score']
    ...: stds = clf.cv_results_['std_test_score']
    ...: for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    ...:         print("%0.3f (+/-%0.03f) for %r"
    ...:               % (mean, std * 2, params))
    ...: print()
0.906 (+/-0.003) for {'C': 9.9999999999999995e-07}
0.904 (+/-0.001) for {'C': 2.6826957952797274e-06}
0.902 (+/-0.000) for {'C': 7.1968567300115137e-06}
0.899 (+/-0.001) for {'C': 1.9306977288832496e-05}
0.894 (+/-0.002) for {'C': 5.1794746792312125e-05}
0.887 (+/-0.006) for {'C': 0.00013894954943731373}
0.861 (+/-0.019) for {'C': 0.00037275937203149379}
0.860 (+/-0.020) for {'C': 0.001}
0.828 (+/-0.077) for {'C': 0.0026826957952797246}
0.856 (+/-0.012) for {'C': 0.0071968567300115137}
0.849 (+/-0.026) for {'C': 0.019306977288832496}
0.860 (+/-0.011) for {'C': 0.051794746792312128}
0.848 (+/-0.017) for {'C': 0.13894954943731361}
0.859 (+/-0.015) for {'C': 0.37275937203149379}
0.863 (+/-0.005) for {'C': 1.0}


In [56]: clf.best_params_
Out[56]: {'C': 9.9999999999999995e-07}
```

Peak Best C

```
Console 1/A
In [60]: clf = GS(svm.LinearSVC(),dict( C=np.logspace(-10, -6, 30)),n_jobs=-1).fit(train, train_y)

In [61]: means = clf.cv_results_['mean_test_score']
    ...: stds = clf.cv_results_['std_test_score']
    ...: for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    ...:         print("%0.3f (+/-%0.03f) for %r"
    ...:               % (mean, std * 2, params))
    ...: print()
0.788 (+/-0.005) for {'C': 1e-10}
0.796 (+/-0.006) for {'C': 1.3738237958832609e-10}
0.805 (+/-0.006) for {'C': 1.8873918221350996e-10}
0.814 (+/-0.007) for {'C': 2.5929437974046672e-10}
0.823 (+/-0.007) for {'C': 3.5622478902624368e-10}
0.831 (+/-0.005) for {'C': 4.8939009184774994e-10}
0.839 (+/-0.004) for {'C': 6.7233575364993349e-10}
0.847 (+/-0.006) for {'C': 9.2367085718738469e-10}
0.854 (+/-0.006) for {'C': 1.2689610031679233e-09}
0.862 (+/-0.005) for {'C': 1.7433288221999873e-09}
0.869 (+/-0.005) for {'C': 2.3950266199874909e-09}
0.873 (+/-0.005) for {'C': 3.2903445623126709e-09}
0.878 (+/-0.003) for {'C': 4.5203536563602409e-09}
0.882 (+/-0.002) for {'C': 6.2101694189156032e-09}
0.887 (+/-0.002) for {'C': 8.5316785241728148e-09}
0.891 (+/-0.002) for {'C': 1.1721022975334793e-08}
0.894 (+/-0.001) for {'C': 1.6102620275609394e-08}
0.896 (+/-0.001) for {'C': 2.2122162910704503e-08}
0.898 (+/-0.001) for {'C': 3.0391953823131947e-08}
0.900 (+/-0.001) for {'C': 4.1753189365604005e-08}
0.902 (+/-0.001) for {'C': 5.7361525104486813e-08}
0.903 (+/-0.002) for {'C': 7.8804628156699041e-08}
0.904 (+/-0.002) for {'C': 1.0826367338740541e-07}
0.905 (+/-0.003) for {'C': 1.4873521072935117e-07}
0.906 (+/-0.002) for {'C': 2.0433597178569396e-07}
0.907 (+/-0.003) for {'C': 2.8072162039411759e-07}
0.907 (+/-0.002) for {'C': 3.856620421163472e-07}
0.907 (+/-0.002) for {'C': 5.2983169062837019e-07}
0.906 (+/-0.003) for {'C': 7.2789538439831463e-07}
0.906 (+/-0.003) for {'C': 9.9999999999999995e-07}
```

Testing for Rbf kernel

Since cross validation takes too long time using LibSVC,

```
In [14]: cl = svm.LinearSVC( C=bestpar).fit(train,train_y)
    ...: predicted = clf.predict(test)
    ...: print("Best C value", bestpar,"Prediction accuracy: %0.2f"\
    ...:       %metrics.accuracy_score(predicted, test_y))
Best C value 0.0193069772888 Prediction accuracy: 0.91

In [15]: %time grid = GS(svm.SVC(cache_size = 600), param_grid=dict( gamma=np.logspace(-5, 0, 5),\
    ...:                    C=[bestpar]),n_jobs=-1 ).fit(train, train_y)
Wall time: 2h 16min 15s

In [16]: grid.best_score_
Out[16]: 0.84869809992962697

In [17]: grid.best_params_
Out[17]: {'C': 0.019306977288832496, 'gamma': 0.0031622776601683794}
```

the optimal approach to try naively default parameter with C=1 , and gamma set to auto and find precision . This gave a precision of 0.93. meaning better combination with C =1

Finding the exteme for the parameter low or high (On low C's, High gamma chosen), also the time is significantly increased as dealing with 64 most variant features.

IPython console

Console 1/A

```
In [13]: p_train = pca.fit_transform(train)

In [14]: pca.fit_transform(test)
Out[14]:
array([[ -5.53509273e-01,  -3.57359188e+00,   1.53774803e+00, ...,
          2.60978147e-01,  -8.84283350e-01,   7.03719415e-01],
       [  3.77424502e-01,   3.48850549e+00,  -9.84052326e-01, ...,
         -4.95779813e-04,  -2.33123306e-01,   2.56946477e-01],
       [  5.69535404e-01,   1.49414408e+00,   3.06421289e-01, ...,
          3.16258858e-01,   6.54428372e-01,   3.45091259e-01],
       ...,
       [  1.26663559e+00,  -3.73722373e+00,   2.40171170e-01, ...,
          9.10279977e-02,   1.43748799e-01,  -3.09191415e-01],
       [  1.30092761e+00,  -2.39982311e-01,  -2.68950933e+00, ...,
         -4.10833915e-01,  -3.50129891e-01,   3.37071682e-01],
       [ -1.45429500e+00,  -1.61655536e+00,  -3.07826224e-01, ...,
          4.91443081e-01,   2.25218407e-02,  -4.65111830e-01]])

In [15]: p_test =pca.fit_transform(test)

In [16]: %time grid = GS(svm.SVC(cache_size = 600), param_grid=dict( gamma=[1.0], C=[0.01,0.001,1]),n_jobs=-1 ).fit(p_train,
train_y)
Wall time: 15min 17s

In [17]: grid.best_score_
Out[17]: 0.21630430756694694

In [18]: grid.best_params_
Out[18]: {'C': 1, 'gamma': 1.0}

In [19]: %time grid = GS(svm.SVC(cache_size = 600), param_grid=dict( gamma=[1.0], C=np.logspace(-4,1,5)),n_jobs=-1
).fit(p_train, train_y)
Wall time: 20min 53s

In [20]: grid.best_params_
Out[20]: {'C': 10.0, 'gamma': 1.0}

In [21]: grid.best_score_
Out[21]: 0.22523056409496647
```

High C and low gamma test

```
IPython console
Console 1/A

In [25]: %time grid = GS(svm.SVC(cache_size = 600), param_grid=dict( gamma=[0.01,0.1], C=[1.0]),n_jobs=-1 ).fit(p_train,
train_y)
Wall time: 4min 40s

In [26]: means = grid.cv_results_['mean_test_score']
    ...: stds = grid.cv_results_['std_test_score']
    ...: for mean, std, params in zip(means, stds, grid.cv_results_['params']):
    ...:     print("%0.3f (+/-%0.03f) for %r"
    ...:               % (mean, std * 2, params))
0.964 (+/-0.003) for {'C': 1.0, 'gamma': 0.01}
0.967 (+/-0.003) for {'C': 1.0, 'gamma': 0.1}

In [27]:
```

Zoom in on range

```
IPython console
Console 1/A   Console 3/A

In [31]: %time grid = GS(svm.SVC(cache_size = 600), param_grid=dict( gamma=np.logspace(-2,-1,12,endpoint=True),
C=[1.0]),n_jobs=-1 ).fit(p_train, train_y)
Wall time: 10min 33s

In [32]: grid.best_score_
Out[32]: 0.97788806992851585

In [33]: means = grid.cv_results_['mean_test_score']
    ...: stds = grid.cv_results_['std_test_score']
    ...: for mean, std, params in zip(means, stds, grid.cv_results_['params']):
    ...:     print("%0.3f (+/-%0.03f) for %r"
    ...:               % (mean, std * 2, params))
0.964 (+/-0.003) for {'C': 1.0, 'gamma': 0.01}
0.967 (+/-0.002) for {'C': 1.0, 'gamma': 0.012328467394420659}
0.970 (+/-0.004) for {'C': 1.0, 'gamma': 0.015199110829529339}
0.973 (+/-0.004) for {'C': 1.0, 'gamma': 0.018738174228603841}
0.975 (+/-0.004) for {'C': 1.0, 'gamma': 0.023101297000831605}
0.977 (+/-0.004) for {'C': 1.0, 'gamma': 0.028480358684358019}
0.977 (+/-0.004) for {'C': 1.0, 'gamma': 0.035111917342151307}
0.978 (+/-0.005) for {'C': 1.0, 'gamma': 0.043287612810830572}
0.977 (+/-0.005) for {'C': 1.0, 'gamma': 0.0533669923120631}
0.976 (+/-0.005) for {'C': 1.0, 'gamma': 0.065793322465756823}
0.974 (+/-0.004) for {'C': 1.0, 'gamma': 0.081113083078968723}
0.967 (+/-0.003) for {'C': 1.0, 'gamma': 0.10000000000000001}

In [34]: grid.best_params_
Out[34]: {'C': 1.0, 'gamma': 0.043287612810830572}

In [35]: |
```

Test results

```
In [41]: print("Classification report for classifier %s:\n%s\n"
    ...:       % (clf, metrics.classification_report(test_y, predicted)))
    ...: print("Confusion matrix:\n%s" % metrics.confusion_matrix(test_y, predicted))
Classification report for classifier SVC(C=1.0, cache_size=600, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.043287612810830572,
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False):
             precision    recall  f1-score   support

          0       0.98      0.99      0.99      1507
          1       0.99      0.99      0.99      1679
          2       0.96      0.99      0.97      1455
          3       0.98      0.97      0.97      1544
          4       0.98      0.98      0.98      1410
          5       0.98      0.97      0.98      1359
          6       0.99      0.98      0.99      1457
          7       0.98      0.98      0.98      1590
          8       0.97      0.97      0.97      1474
          9       0.97      0.95      0.96      1526

avg / total       0.98      0.98      0.98     15001
```