Homework 3

March 8, 2018

1 Homework 3: Hyperparameter Tuning with SVMs

The final deliverable for this homework will be this Jupyter notebook, which should include all relevant code, markdown cells before each code block describing what the code does, and any write-ups/images/plots that you wish to include.

To add a block click on `Insert > Insert Cell Below`. To make a markdown cell, click the drop-down menu at the top of this page and select `Markdown`.

The starter code for this homework is purposely very minimal. You should get used to coding from scratch. Just follow all the instructions in the PDF you will be fine.

```
In [1]: import numpy as np
        import pandas as pd

        from sklearn.svm import SVC
        from sklearn.model_selection import train_test_split, GridSearchCV

        from scipy.stats import zscore

        import matplotlib.pyplot as plt
```

1.1 2.3 Grid Search

Here I will be finding the optimal pair of (C, d) for an SVM model where C is the hyperparameter which determines the amount with which we penalize misclassified points and d is the degree of the polynomial, that gives the best results on a test set.

1.1.1 2.3.1 Specifications

First let's read in the data and normalize all the data points.

```
In [2]: df = pd.read_csv('breast-cancer-wisconsin.data',
                         names=['ID', 'radius', 'texture', 'perimeter',
                                'area', 'smoothness', 'compactness', 'concativity',
                                'concave points', 'symmetry', 'label'])
        df
```

```
Out[2]:           ID  radius  texture  perimeter  area  smoothness  compactness  \
        0    1000025       5        1          1     1           2            1
```

1	1002945	5	4	4	5	7	10
2	1015425	3	1	1	1	2	2
3	1016277	6	8	8	1	3	4
4	1017023	4	1	1	3	2	1
5	1017122	8	10	10	8	7	10
6	1018099	1	1	1	1	2	10
7	1018561	2	1	2	1	2	1
8	1033078	2	1	1	1	2	1
9	1033078	4	2	1	1	2	1
10	1035283	1	1	1	1	1	1
11	1036172	2	1	1	1	2	1
12	1041801	5	3	3	3	2	3
13	1043999	1	1	1	1	2	3
14	1044572	8	7	5	10	7	9
15	1047630	7	4	6	4	6	1
16	1048672	4	1	1	1	2	1
17	1049815	4	1	1	1	2	1
18	1050670	10	7	7	6	4	10
19	1050718	6	1	1	1	2	1
20	1054590	7	3	2	10	5	10
21	1054593	10	5	5	3	6	7
22	1056784	3	1	1	1	2	1
23	1059552	1	1	1	1	2	1
24	1065726	5	2	3	4	2	7
25	1066373	3	2	1	1	1	1
26	1066979	5	1	1	1	2	1
27	1067444	2	1	1	1	2	1
28	1070935	1	1	3	1	2	1
29	1070935	3	1	1	1	1	1
..	...	...	...	...	...	...	...
653	1350423	5	10	10	8	5	5
654	1352848	3	10	7	8	5	8
655	1353092	3	2	1	2	2	1
656	1354840	2	1	1	1	2	1
657	1354840	5	3	2	1	3	1
658	1355260	1	1	1	1	2	1
659	1365075	4	1	4	1	2	1
660	1365328	1	1	2	1	2	1
661	1368267	5	1	1	1	2	1
662	1368273	1	1	1	1	2	1
663	1368882	2	1	1	1	2	1
664	1369821	10	10	10	10	5	10
665	1371026	5	10	10	10	4	10
666	1371920	5	1	1	1	2	1
667	466906	1	1	1	1	2	1
668	466906	1	1	1	1	2	1
669	534555	1	1	1	1	2	1
670	536708	1	1	1	1	2	1

```
671   566346       3        1          1     1           2            1
672   603148       4        1          1     1           2            1
673   654546       1        1          1     1           2            1
674   654546       1        1          1     3           2            1
675   695091       5       10         10     5           4            5
676   714039       3        1          1     1           2            1
677   763235       3        1          1     1           2            1
678   776715       3        1          1     1           3            2
679   841769       2        1          1     1           2            1
680   888820       5       10         10     3           7            3
681   897471       4        8          6     4           3            4
682   897471       4        8          8     5           4            5

     concativity  concave points  symmetry  label
0              3               1         1      2
1              3               2         1      2
2              3               1         1      2
3              3               7         1      2
4              3               1         1      2
5              9               7         1      4
6              3               1         1      2
7              3               1         1      2
8              1               1         5      2
9              2               1         1      2
10             3               1         1      2
11             2               1         1      2
12             4               4         1      4
13             3               1         1      2
14             5               5         4      4
15             4               3         1      4
16             2               1         1      2
17             3               1         1      2
18             4               1         2      4
19             3               1         1      2
20             5               4         4      4
21             7              10         1      4
22             2               1         1      2
23             3               1         1      2
24             3               6         1      4
25             2               1         1      2
26             2               1         1      2
27             2               1         1      2
28             1               1         1      2
29             2               1         1      2
..           ...             ...       ...    ...
653            7              10         1      4
654            7               4         1      4
655            3               1         1      2
```

```
656              3                1          1      2
657              1                1          1      2
658              2                1          1      2
659              1                1          1      2
660              2                1          1      2
661              1                1          1      2
662              1                1          1      2
663              1                1          1      2
664             10               10          7      4
665              5                6          3      4
666              3                2          1      2
667              1                1          1      2
668              1                1          1      2
669              1                1          1      2
670              1                1          1      2
671              2                3          1      2
672              1                1          1      2
673              1                1          8      2
674              1                1          1      2
675              4                4          1      4
676              1                1          1      2
677              2                1          2      2
678              1                1          1      2
679              1                1          1      2
680              8               10          2      4
681             10                6          1      4
682             10                4          1      4

[683 rows x 11 columns]
```

```
In [10]: normalized_df = df.apply(zscore)
         for i in range(df.shape[0]):
             if normalized_df['label'][i] < 0:
                 normalized_df['label'][i] = 0
             else:
                 normalized_df['label'][i] = 1
         normalized_df
```

```
Out[10]:           ID    radius   texture  perimeter      area  smoothness  \
         0   -0.123664  0.197905 -0.702212  -0.741774 -0.639366   -0.555608
         1   -0.118956  0.197905  0.277252   0.262783  0.758032    1.695166
         2   -0.098833 -0.511643 -0.702212  -0.741774 -0.639366   -0.555608
         3   -0.097459  0.552679  1.583204   1.602192 -0.639366   -0.105454
         4   -0.096256 -0.156869 -0.702212  -0.741774  0.059333   -0.555608
         5   -0.096097  1.262227  2.236180   2.271896  1.806080    1.695166
         6   -0.094521 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
         7   -0.093777 -0.866417 -0.702212  -0.406921 -0.639366   -0.555608
         8   -0.070369 -0.866417 -0.702212  -0.741774 -0.639366   -0.555608
```

```
9   -0.070369 -0.156869 -0.375724  -0.741774 -0.639366   -0.555608
10  -0.066814 -1.221191 -0.702212  -0.741774 -0.639366   -1.005763
11  -0.065380 -0.866417 -0.702212  -0.741774 -0.639366   -0.555608
12  -0.056304  0.197905 -0.049236  -0.072069  0.059333   -0.555608
13  -0.052760 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
14  -0.051836  1.262227  1.256716   0.597635  2.504778    1.695166
15  -0.046905  0.907453  0.277252   0.932487  0.408682    1.245011
16  -0.045225 -0.156869 -0.702212  -0.741774 -0.639366   -0.555608
17  -0.043382 -0.156869 -0.702212  -0.741774 -0.639366   -0.555608
18  -0.042004  1.971775  1.256716   1.267340  1.107381    0.344701
19  -0.041926  0.552679 -0.702212  -0.741774 -0.639366   -0.555608
20  -0.035683  0.907453 -0.049236  -0.406921  2.504778    0.794856
21  -0.035678  1.971775  0.603740   0.597635  0.059333    1.245011
22  -0.032145 -0.511643 -0.702212  -0.741774 -0.639366   -0.555608
23  -0.027682 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
24  -0.017727  0.197905 -0.375724  -0.072069  0.408682   -0.555608
25  -0.016684 -0.511643 -0.375724  -0.741774 -0.639366   -1.005763
26  -0.015707  0.197905 -0.702212  -0.741774 -0.639366   -0.555608
27  -0.014957 -0.866417 -0.702212  -0.741774 -0.639366   -0.555608
28  -0.009328 -1.221191 -0.702212  -0.072069 -0.639366   -0.555608
29  -0.009328 -0.511643 -0.702212  -0.741774 -0.639366   -1.005763
..        ...       ...       ...        ...       ...         ...
653  0.441321  0.197905  2.236180   2.271896  1.806080    0.794856
654  0.445231 -0.511643  2.236180   1.267340  1.806080    0.794856
655  0.445625 -0.511643 -0.375724  -0.741774 -0.290016   -0.555608
656  0.448443 -0.866417 -0.702212  -0.741774 -0.639366   -0.555608
657  0.448443  0.197905 -0.049236  -0.406921 -0.639366   -0.105454
658  0.449120 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
659  0.464946 -0.156869 -0.702212   0.262783 -0.639366   -0.555608
660  0.465354 -1.221191 -0.702212  -0.406921 -0.639366   -0.555608
661  0.470093  0.197905 -0.702212  -0.741774 -0.639366   -0.555608
662  0.470103 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
663  0.471085 -0.866417 -0.702212  -0.741774 -0.639366   -0.555608
664  0.472599  1.971775  2.236180   2.271896  2.504778    0.794856
665  0.474542  0.197905  2.236180   2.271896  2.504778    0.344701
666  0.475983  0.197905 -0.702212  -0.741774 -0.639366   -0.555608
667 -0.983271 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
668 -0.983271 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
669 -0.874193 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
670 -0.870721 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
671 -0.822933 -0.511643 -0.702212  -0.741774 -0.639366   -0.555608
672 -0.763593 -0.156869 -0.702212  -0.741774 -0.639366   -0.555608
673 -0.680718 -1.221191 -0.702212  -0.741774 -0.639366   -0.555608
674 -0.680718 -1.221191 -0.702212  -0.741774  0.059333   -0.555608
675 -0.615343  0.197905  2.236180   2.271896  0.758032    0.344701
676 -0.584791 -0.511643 -0.702212  -0.741774 -0.639366   -0.555608
677 -0.505467 -0.511643 -0.702212  -0.741774 -0.639366   -0.555608
678 -0.483732 -0.511643 -0.702212  -0.741774 -0.639366   -0.105454
```

```
679 -0.378838 -0.866417 -0.702212  -0.741774 -0.639366   -0.555608
680 -0.302972  0.197905  2.236180   2.271896  0.059333    1.695166
681 -0.289023 -0.156869  1.583204   0.932487  0.408682   -0.105454
682 -0.289023 -0.156869  1.583204   1.602192  0.758032    0.344701

     compactness  concativity  concave points  symmetry  label
0      -0.698853    -0.181827       -0.612927 -0.348400    0.0
1       1.772867    -0.181827       -0.285105 -0.348400    0.0
2      -0.424217    -0.181827       -0.612927 -0.348400    0.0
3       0.125054    -0.181827        1.354008 -0.348400    0.0
4      -0.698853    -0.181827       -0.612927 -0.348400    0.0
5       1.772867     2.269251        1.354008 -0.348400    1.0
6       1.772867    -0.181827       -0.612927 -0.348400    0.0
7      -0.698853    -0.181827       -0.612927 -0.348400    0.0
8      -0.698853    -0.998853       -0.612927  1.961862    0.0
9      -0.698853    -0.590340       -0.612927 -0.348400    0.0
10     -0.698853    -0.181827       -0.612927 -0.348400    0.0
11     -0.698853    -0.590340       -0.612927 -0.348400    0.0
12     -0.149582     0.226686        0.370540 -0.348400    1.0
13     -0.149582    -0.181827       -0.612927 -0.348400    0.0
14      1.498232     0.635199        0.698363  1.384297    1.0
15     -0.698853     0.226686        0.042718 -0.348400    1.0
16     -0.698853    -0.590340       -0.612927 -0.348400    0.0
17     -0.698853    -0.181827       -0.612927 -0.348400    0.0
18      1.772867     0.226686       -0.612927  0.229166    1.0
19     -0.698853    -0.181827       -0.612927 -0.348400    0.0
20      1.772867     0.635199        0.370540  1.384297    1.0
21      0.948960     1.452225        2.337476 -0.348400    1.0
22     -0.698853    -0.590340       -0.612927 -0.348400    0.0
23     -0.698853    -0.181827       -0.612927 -0.348400    0.0
24      0.948960    -0.181827        1.026185 -0.348400    1.0
25     -0.698853    -0.590340       -0.612927 -0.348400    0.0
26     -0.698853    -0.590340       -0.612927 -0.348400    0.0
27     -0.698853    -0.590340       -0.612927 -0.348400    0.0
28     -0.698853    -0.998853       -0.612927 -0.348400    0.0
29     -0.698853    -0.590340       -0.612927 -0.348400    0.0
..           ...          ...             ...       ...    ...
653     0.399689     1.452225        2.337476 -0.348400    1.0
654     1.223596     1.452225        0.370540 -0.348400    1.0
655    -0.698853    -0.181827       -0.612927 -0.348400    0.0
656    -0.698853    -0.181827       -0.612927 -0.348400    0.0
657    -0.698853    -0.998853       -0.612927 -0.348400    0.0
658    -0.698853    -0.590340       -0.612927 -0.348400    0.0
659    -0.698853    -0.998853       -0.612927 -0.348400    0.0
660    -0.698853    -0.590340       -0.612927 -0.348400    0.0
661    -0.698853    -0.998853       -0.612927 -0.348400    0.0
662    -0.698853    -0.998853       -0.612927 -0.348400    0.0
663    -0.698853    -0.998853       -0.612927 -0.348400    0.0
```

```
664     1.772867     2.677764        2.337476  3.116993    1.0
665     1.772867     0.635199        1.026185  0.806731    1.0
666    -0.698853    -0.181827       -0.285105 -0.348400    0.0
667    -0.698853    -0.998853       -0.612927 -0.348400    0.0
668    -0.698853    -0.998853       -0.612927 -0.348400    0.0
669    -0.698853    -0.998853       -0.612927 -0.348400    0.0
670    -0.698853    -0.998853       -0.612927 -0.348400    0.0
671    -0.698853    -0.590340        0.042718 -0.348400    0.0
672    -0.698853    -0.998853       -0.612927 -0.348400    0.0
673    -0.698853    -0.998853       -0.612927  3.694559    0.0
674    -0.698853    -0.998853       -0.612927 -0.348400    0.0
675     0.399689     0.226686        0.370540 -0.348400    1.0
676    -0.698853    -0.998853       -0.612927 -0.348400    0.0
677    -0.698853    -0.590340       -0.612927  0.229166    0.0
678    -0.424217    -0.998853       -0.612927 -0.348400    0.0
679    -0.698853    -0.998853       -0.612927 -0.348400    0.0
680    -0.149582     1.860738        2.337476  0.229166    1.0
681     0.125054     2.677764        1.026185 -0.348400    1.0
682     0.399689     2.677764        0.370540 -0.348400    1.0

[683 rows x 11 columns]
```

Now let's split our data into a testing and training set.

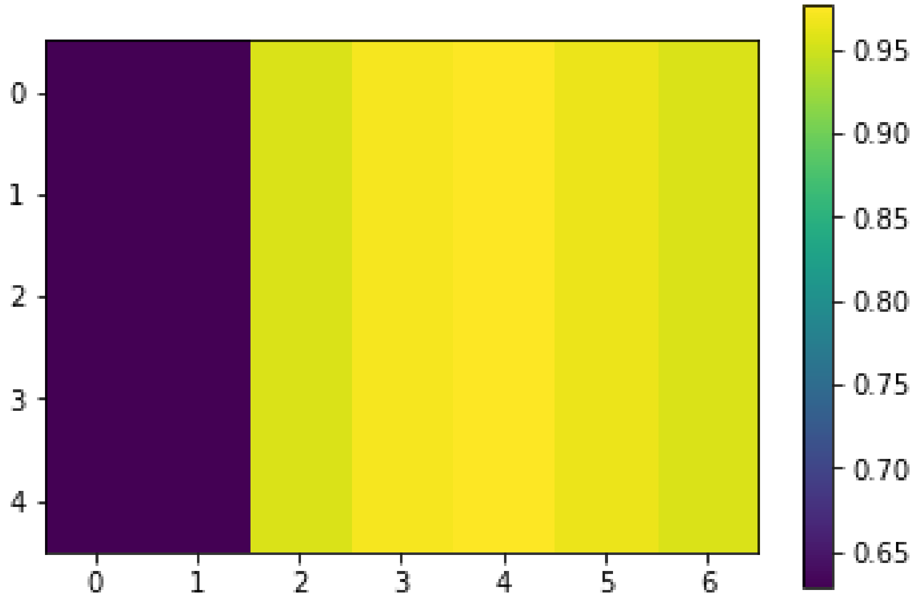
```
In [13]: C = [.0001, .001, .01, .1, 1, 10, 100]
         d = [1, 2, 3, 4, 5]
         train_set, test_set = train_test_split(normalized_df, test_size = .3)
```

Now let's test to see which pair of (C, d) will yield the highest accuracy.

```
In [14]: row, col = len(d), len(C);
         accuracies = [[0 for c in range(col)] for r in range(row)]
         for i in range(row):
             for j in range(col):
                 svc = SVC(C=C[j], degree=d[i], probability=True)
                 svc.fit(train_set[['radius', 'texture', 'perimeter', 'area',
                                    'smoothness', 'compactness', 'concativity',
                                    'concave points', 'symmetry']],
                         train_set['label'])
                 score = svc.score(test_set[['radius', 'texture', 'perimeter', 'area',
                                             'smoothness', 'compactness', 'concativity',
                                             'concave points', 'symmetry']],
                                   test_set['label'])
                 accuracies[i][j] = score
         accuracies
         array = np.array(accuracies)
         array

Out[14]: array([[ 0.62926829,  0.62926829,  0.95609756,  0.97073171,  0.97560976,
                  0.96585366,  0.95609756],
                [ 0.62926829,  0.62926829,  0.95609756,  0.97073171,  0.97560976,
                  0.96585366,  0.95609756],
                [ 0.62926829,  0.62926829,  0.95609756,  0.97073171,  0.97560976,
                  0.96585366,  0.95609756],
                [ 0.62926829,  0.62926829,  0.95609756,  0.97073171,  0.97560976,
                  0.96585366,  0.95609756],
                [ 0.62926829,  0.62926829,  0.95609756,  0.97073171,  0.97560976,
                  0.96585366,  0.95609756]])
```

```
In [15]: plt.imshow(array)
         plt.colorbar()
         plt.show()
```



```
In [16]: best = 0
         best_C = 0
         best_d = 0
         for i in range(row):
             for j in range(col):
                 if accuracies[i][j] > best:
                     best = accuracies[i][j]
                     best_C = C[j]
                     best_d = d[i]
         print(best, best_C, best_d)
```

```
0.975609756098 1 1
```

1.2 2.4 Cross Validation Search

Use cross validated grid search to find the best combination of and C on the breast cancer dataset.

1.2.1 2.4.1 Specifications

Using $C = \{.0001, .001, .01, .1, 1, 10, 100\}$ and $\gamma = \{.0001, .001, .01, .1, 1, 10\}$ for gridsearch, we wil report the best accuracy and which parameters gave the best accuracy.

```
In [20]: def svc_param_selection(X, y, nfolds):
             Cs = [.001, .01, .1, 1, 10, 100, 1000]
             gammas = [.001, .01, .1, 1, 10, 100]
             param_grid = {'C':Cs, 'gamma':gammas}
             search = GridSearchCV(SVC(kernel='rbf'), param_grid, cv=nfolds)
             search.fit(X, y)
             search.best_params_
             return search.best_params_, search.best_score_

         svc_param_selection(normalized_df[['radius', 'texture', 'perimeter',
                                            'area', 'smoothness', 'compactness',
                                            'concavity', 'concave points', 'symmetry']],
                             normalized_df['label'], 5)
```

```
Out[20]: ({'C': 1, 'gamma': 0.01}, 0.97071742313323572)
```