Homework 3

March 7, 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
import seaborn as sns

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

import matplotlib.pyplot as plt
from jupyterthemes import jtplot
jtplot.style()
```

In [2]:
```
#load data
data = pd.read_csv("breast-cancer-wisconsin.data")
data = data.drop(['1000025'], axis = 1)
data.head()
```

Out[2]:

	5	1	1.1	1.2	2	1.3	3	1.4	1.5	2.1
0	5	4	4	5	7	10	3	2	1	2
1	3	1	1	1	2	2	3	1	1	2
2	6	8	8	1	3	4	3	7	1	2
3	4	1	1	3	2	1	3	1	1	2
4	8	10	10	8	7	10	9	7	1	4

In [3]:
```
#normalize data to have 0 mean and 1 standard deviation
# normalize everything except labels
selection = ['5', '1', '1.1', '1.2', '2', '1.3', '3', '1.4', '1.5']
data_norm = (data[selection] - data.mean()[selection]) / (data.std()[selection])
# scale down labels to be 1 and 0
```

```
data_norm = data_norm.join((data['2.1'] - 2)*0.5)
# data_norm
```

In [4]:
```
#split data
d_train, d_test = train_test_split(data_norm, test_size = 0.3)
d_train_x, d_train_y = d_train[selection], d_train['2.1']
d_test_x, d_test_y = d_test[selection], d_test['2.1']
```

In [5]:
```
#grid search polynomial kernal
def grid_search_poly(X, y):
    Cs = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]
    degree = [1, 2, 3, 4, 5]
    param_grid = {'C': Cs, 'degree' : degree}
    search = GridSearchCV(SVC(kernel = 'poly'), param_grid, cv=ShuffleSplit(test_size=(
    search.fit(X, y)
    print(search.best_params_)
    return search.cv_results_
```

In [6]:
```
#pass in all data because we split using shuffleSplit
results = grid_search_poly(data_norm[selection], data_norm['2.1'])
results = pd.DataFrame(results)
results
```

```
{'C': 0.1, 'degree': 1}
```

Out[6]:

	mean_fit_time	mean_score_time	mean_test_score	mean_train_score	param_C	\
0	0.003819	0.001304	0.647317	0.650524	0.0001	
1	0.003816	0.001200	0.647317	0.650524	0.0001	
2	0.003606	0.001203	0.647317	0.650524	0.0001	
3	0.004113	0.001301	0.658049	0.664990	0.0001	
4	0.003506	0.001605	0.683902	0.684277	0.0001	
5	0.003810	0.001307	0.647317	0.650524	0.001	
6	0.004104	0.001510	0.647317	0.650524	0.001	
7	0.004110	0.001602	0.702927	0.705870	0.001	
8	0.004306	0.001105	0.718537	0.726625	0.001	
9	0.003597	0.001310	0.741463	0.754507	0.001	
10	0.003108	0.000909	0.951220	0.949686	0.01	
11	0.003908	0.001200	0.767317	0.774214	0.01	
12	0.003011	0.001201	0.839512	0.840042	0.01	
13	0.003710	0.000903	0.805366	0.816143	0.01	
14	0.003106	0.001002	0.827805	0.838784	0.01	
15	0.002302	0.000902	0.974634	0.969602	0.1	
16	0.003911	0.001200	0.864390	0.874423	0.1	
17	0.002600	0.000905	0.927317	0.926415	0.1	
18	0.003013	0.000602	0.870732	0.884486	0.1	
19	0.002604	0.001006	0.878049	0.893711	0.1	
20	0.001608	0.000501	0.970732	0.971488	1	
21	0.002907	0.000697	0.943415	0.953040	1	

22	0.002515	0.001010	0.955610	0.966667	1
23	0.003107	0.000799	0.926341	0.941090	1
24	0.002408	0.000999	0.928780	0.939623	1
25	0.002003	0.000607	0.968780	0.974004	10
26	0.003305	0.000401	0.940976	0.973375	10
27	0.001911	0.000398	0.955610	0.985115	10
28	0.003106	0.000606	0.936585	0.981342	10
29	0.002002	0.000407	0.945854	0.978826	10
30	0.003506	0.000401	0.966341	0.974423	100
31	0.008726	0.000501	0.939512	0.983857	100
32	0.002606	0.000201	0.951220	0.990147	100
33	0.004910	0.000599	0.931220	0.987841	100
34	0.002202	0.000504	0.945854	0.987841	100

	param_degree	params	rank_test_score	\
0	1	{'C': 0.0001, 'degree': 1}	31	
1	2	{'C': 0.0001, 'degree': 2}	31	
2	3	{'C': 0.0001, 'degree': 3}	31	
3	4	{'C': 0.0001, 'degree': 4}	30	
4	5	{'C': 0.0001, 'degree': 5}	29	
5	1	{'C': 0.001, 'degree': 1}	31	
6	2	{'C': 0.001, 'degree': 2}	31	
7	3	{'C': 0.001, 'degree': 3}	28	
8	4	{'C': 0.001, 'degree': 4}	27	
9	5	{'C': 0.001, 'degree': 5}	26	
10	1	{'C': 0.01, 'degree': 1}	7	
11	2	{'C': 0.01, 'degree': 2}	25	
12	3	{'C': 0.01, 'degree': 3}	22	
13	4	{'C': 0.01, 'degree': 4}	24	
14	5	{'C': 0.01, 'degree': 5}	23	
15	1	{'C': 0.1, 'degree': 1}	1	
16	2	{'C': 0.1, 'degree': 2}	21	
17	3	{'C': 0.1, 'degree': 3}	17	
18	4	{'C': 0.1, 'degree': 4}	20	
19	5	{'C': 0.1, 'degree': 5}	19	
20	1	{'C': 1, 'degree': 1}	2	
21	2	{'C': 1, 'degree': 2}	11	
22	3	{'C': 1, 'degree': 3}	5	
23	4	{'C': 1, 'degree': 4}	18	
24	5	{'C': 1, 'degree': 5}	16	
25	1	{'C': 10, 'degree': 1}	3	
26	2	{'C': 10, 'degree': 2}	12	
27	3	{'C': 10, 'degree': 3}	5	
28	4	{'C': 10, 'degree': 4}	14	
29	5	{'C': 10, 'degree': 5}	9	
30	1	{'C': 100, 'degree': 1}	4	
31	2	{'C': 100, 'degree': 2}	13	
32	3	{'C': 100, 'degree': 3}	7	

33	4	{'C': 100, 'degree': 4}	15
34	5	{'C': 100, 'degree': 5}	9

	split0_test_score	split0_train_score	...	split7_test_score \
0	0.697561	0.628931	...	0.648780
1	0.697561	0.628931	...	0.648780
2	0.697561	0.628931	...	0.648780
3	0.712195	0.643606	...	0.658537
4	0.726829	0.666667	...	0.692683
5	0.697561	0.628931	...	0.648780
6	0.697561	0.628931	...	0.648780
7	0.736585	0.700210	...	0.717073
8	0.746341	0.721174	...	0.726829
9	0.760976	0.750524	...	0.760976
10	0.965854	0.947589	...	0.956098
11	0.795122	0.771488	...	0.780488
12	0.873171	0.825996	...	0.863415
13	0.829268	0.800839	...	0.809756
14	0.863415	0.819706	...	0.843902
15	0.970732	0.972746	...	0.970732
16	0.887805	0.872117	...	0.873171
17	0.946341	0.926625	...	0.941463
18	0.892683	0.880503	...	0.882927
19	0.897561	0.888889	...	0.887805
20	0.965854	0.979036	...	0.970732
21	0.931707	0.953878	...	0.946341
22	0.960976	0.962264	...	0.960976
23	0.931707	0.939203	...	0.926829
24	0.936585	0.939203	...	0.941463
25	0.970732	0.983229	...	0.965854
26	0.917073	0.981132	...	0.936585
27	0.951220	0.983229	...	0.956098
28	0.917073	0.976939	...	0.956098
29	0.946341	0.974843	...	0.965854
30	0.951220	0.985325	...	0.960976
31	0.912195	0.987421	...	0.951220
32	0.926829	0.989518	...	0.960976
33	0.907317	0.983229	...	0.926829
34	0.931707	0.983229	...	0.960976

	split7_train_score	split8_test_score	split8_train_score \
0	0.649895	0.643902	0.651992
1	0.649895	0.643902	0.651992
2	0.649895	0.643902	0.651992
3	0.664570	0.653659	0.666667
4	0.679245	0.682927	0.685535
5	0.649895	0.643902	0.651992
6	0.649895	0.643902	0.651992

7	0.696017	0.702439	0.706499
8	0.719078	0.702439	0.737945
9	0.754717	0.721951	0.765199
10	0.951782	0.951220	0.947589
11	0.767296	0.746341	0.784067
12	0.832285	0.834146	0.834382
13	0.807128	0.785366	0.817610
14	0.834382	0.829268	0.834382
15	0.972746	0.975610	0.968553
16	0.876310	0.863415	0.865828
17	0.920335	0.926829	0.926625
18	0.886792	0.878049	0.882600
19	0.899371	0.878049	0.893082
20	0.972746	0.975610	0.966457
21	0.955975	0.951220	0.953878
22	0.970650	0.956098	0.964361
23	0.935010	0.926829	0.941300
24	0.932914	0.931707	0.937107
25	0.976939	0.970732	0.972746
26	0.976939	0.946341	0.972746
27	0.987421	0.951220	0.989518
28	0.985325	0.936585	0.983229
29	0.981132	0.951220	0.981132
30	0.972746	0.970732	0.972746
31	0.987421	0.926829	0.989518
32	0.993711	0.951220	0.991614
33	0.991614	0.931707	0.991614
34	0.991614	0.951220	0.991614

	split9_test_score	split9_train_score	std_fit_time	std_score_time	\
0	0.643902	0.651992	0.000596	4.592342e-04	
1	0.643902	0.651992	0.000604	4.032394e-04	
2	0.643902	0.651992	0.000489	4.006749e-04	
3	0.653659	0.666667	0.000534	4.549588e-04	
4	0.692683	0.679245	0.000671	4.915570e-04	
5	0.643902	0.651992	0.000758	4.573331e-04	
6	0.643902	0.651992	0.000294	4.959051e-04	
7	0.717073	0.700210	0.001381	4.883856e-04	
8	0.736585	0.719078	0.000790	3.004681e-04	
9	0.756098	0.754717	0.000500	4.554763e-04	
10	0.941463	0.951782	0.000943	3.035352e-04	
11	0.770732	0.773585	0.000706	4.026301e-04	
12	0.834146	0.842767	0.000449	4.030970e-04	
13	0.809756	0.823899	0.000453	3.010280e-04	
14	0.814634	0.842767	0.000530	5.859473e-07	
15	0.965854	0.970650	0.000639	3.007939e-04	
16	0.853659	0.870021	0.000546	4.016424e-04	
17	0.926829	0.930818	0.000662	5.406621e-04	

18	0.873171	0.882600	0.000783	4.912642e-04
19	0.882927	0.884696	0.000663	9.219773e-06
20	0.960976	0.972746	0.000487	5.011797e-04
21	0.921951	0.951782	0.000697	4.568257e-04
22	0.941463	0.972746	0.000666	4.489582e-04
23	0.926829	0.943396	0.000546	3.999259e-04
24	0.921951	0.945493	0.000668	4.415712e-04
25	0.965854	0.976939	0.000634	4.955439e-04
26	0.931707	0.968553	0.000455	4.916438e-04
27	0.936585	0.981132	0.000302	4.875715e-04
28	0.912195	0.983229	0.000535	4.945451e-04
29	0.931707	0.976939	0.000009	4.986193e-04
30	0.970732	0.976939	0.001020	4.911191e-04
31	0.941463	0.979036	0.004205	5.013943e-04
32	0.946341	0.989518	0.000492	4.010678e-04
33	0.917073	0.987421	0.001051	4.892257e-04
34	0.926829	0.987421	0.000396	5.041360e-04

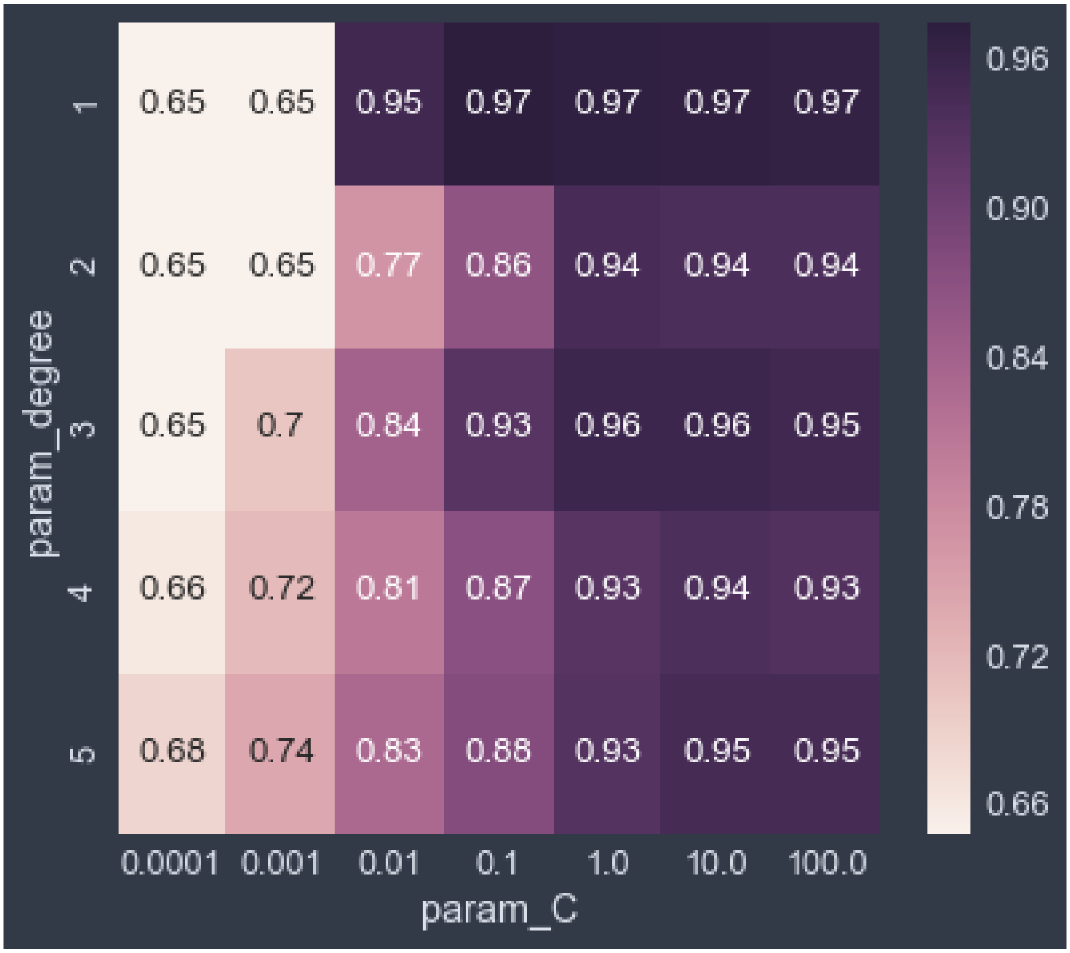
	std_test_score	std_train_score
0	0.027599	0.011861
1	0.027599	0.011861
2	0.027599	0.011861
3	0.027976	0.011627
4	0.027835	0.012964
5	0.027599	0.011861
6	0.027599	0.011861
7	0.030499	0.010100
8	0.029272	0.011297
9	0.025347	0.007813
10	0.010462	0.003978
11	0.022253	0.006223
12	0.022927	0.009746
13	0.018696	0.009700
14	0.024199	0.009764
15	0.007169	0.003539
16	0.014438	0.004912
17	0.010336	0.003916
18	0.012952	0.005503
19	0.012720	0.005224
20	0.008726	0.004108
21	0.010951	0.004000
22	0.008293	0.005341
23	0.005093	0.004027
24	0.009805	0.003956
25	0.004975	0.004317
26	0.011220	0.005707
27	0.014868	0.003172
28	0.018639	0.003172

```
29         0.012989         0.003308
30         0.008001         0.005118
31         0.014003         0.004501
32         0.014305         0.002107
33         0.024140         0.002935
34         0.014213         0.002935

[35 rows x 32 columns]
```

In [7]:
```
accuracies = results.pivot(index='param_degree', columns='param_C', values='mean_test_s
accuracies
```

Out[7]:
```
param_C       0.0001    0.0010    0.0100    0.1000    1.0000    10.0000   \
param_degree
1             0.647317  0.647317  0.951220  0.974634  0.970732  0.968780
2             0.647317  0.647317  0.767317  0.864390  0.943415  0.940976
3             0.647317  0.702927  0.839512  0.927317  0.955610  0.955610
4             0.658049  0.718537  0.805366  0.870732  0.926341  0.936585
5             0.683902  0.741463  0.827805  0.878049  0.928780  0.945854

param_C       100.0000
param_degree
1             0.966341
2             0.939512
3             0.951220
4             0.931220
5             0.945854
```

In [8]:
```
sns.heatmap(accuracies, annot=True)
plt.show()
```




```
In [9]: #best accuracy
        svc = SVC(kernel = 'poly', C = 0.1, degree = 1)
        svc.fit(d_test_x, d_test_y)
        svc.score(d_test_x, d_test_y)
```

```
Out[9]: 0.98048780487804876
```

From the grid search the best parameters are C = 0.1 and degree = 1. Using these parameters the best accuracy we get is 98%.

Generally, our model is innaccurate if we have too low of a C because we fail to penalize misclassified points. If our C is too high the SVM tries to classify outliers and the model overfits. Also with a high degree our model tends to overfit because it allows too many degrees of freedom and begins to fit more noise.

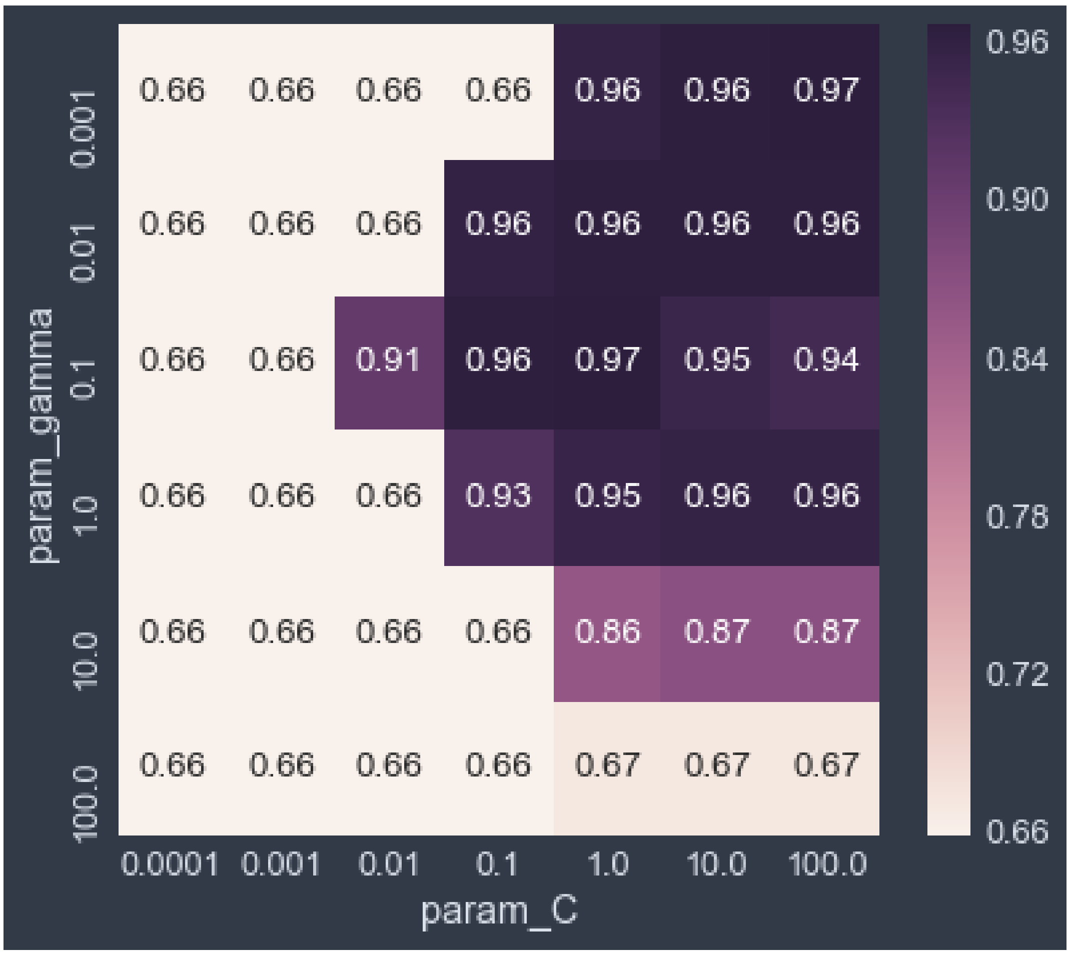
```
In [10]: #grid search polynomial kernal
         def grid_search_rbf(X, y):
             Cs = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]
             gamma = [0.001, 0.01, 0.1, 1, 10, 100]
```

```
    param_grid = {'C': Cs, 'gamma' : gamma}
    search = GridSearchCV(SVC(kernel = 'rbf'), param_grid, cv=5)
    search.fit(X, y)
    print(search.best_score_, search.best_params_)
    return search.cv_results_
```

```
In [13]: resultsCV = grid_search_rbf(d_train_x, d_train_y)
         resultsCV = pd.DataFrame(resultsCV)
         accuraciesCV = resultsCV.pivot(index='param_gamma', columns='param_C', values='mean_te
```

```
0.966457023061 {'C': 1, 'gamma': 0.1}
```

```
In [14]: sns.heatmap(accuraciesCV, annot=True)
         plt.show()
```



The best accuracy with cross validated grid search is 97% using C = 1, and gamma = 0.01

```
In [ ]:
```