ANNADULT

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/adult.csv', index_col = F
df = df.dropna(axis=0)
onehot = pd.get_dummies(data=df, columns=['Column2','Column4', 'Column6', '
                        drop_first = True)
onehot = onehot.drop(['Column15'], axis=1)
df = onehot.join(df['Column15'])
df = df.dropna(axis=1)
df = df[df.notnull()]
```

In [3]:
```
X_and_Y = df.as_matrix()
np.random.shuffle(X_and_Y)
X_and_Y = X_and_Y[:5000, :101]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

```
(5000, 101) (5000, 100) (5000,)
```

In [4]:
```
#change to binary classification
for i in range(len(Y)):
    if Y[i] == ' >50K':
        Y[i] = 1
    else:
        Y[i] = 0
print(Y[:200])
```

```
[0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1
```

```
 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0
 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1
 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0]
```

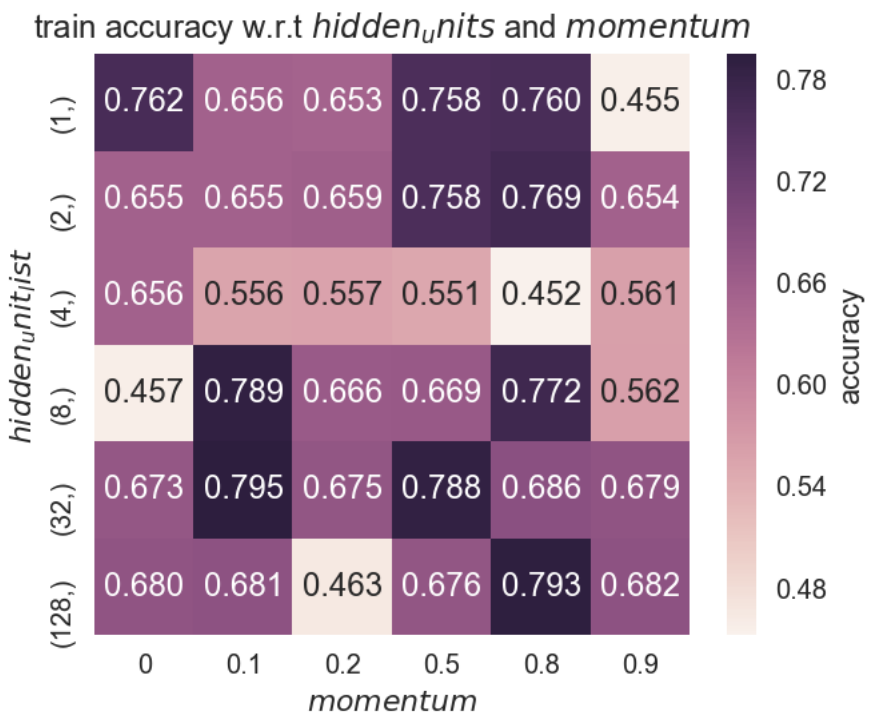
In [5]:
```
classifier = MLPClassifier(max_iter = 500)
hidden_unit_list = [(1,), (2,), (4,), (8,), (32,), (128,)]
momentum_list = [0,0.1, 0.2, 0.5, 0.8, 0.9]
params = {'hidden_layer_sizes':hidden_unit_list,'momentum':momentum_list}
grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

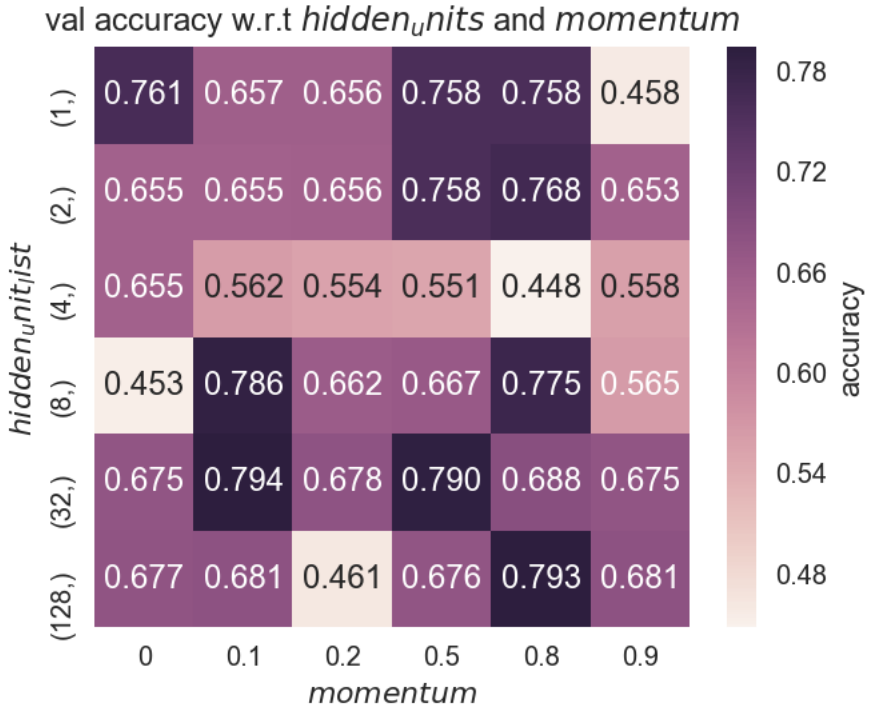
In [6]:
```
def draw_heatmap_RBF(acc, acc_desc, momentum_list, hidden_unit_list):
    plt.figure(figsize = (5,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f',
                     xticklabels=momentum_list, yticklabels=hidden_unit_lis
    ax.collections[0].colorbar.set_label("accuracy")
    ax.set(xlabel = '$momentum$', ylabel='$hidden_unit_list$')
    plt.title(acc_desc + ' w.r.t $hidden_units$ and $momentum$')
    sns.set_style("whitegrid", {'axes.grid' : False})
    plt.show()
```

In [7]:
```
Y = Y.astype('int')

X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
X_test      = X[int(0.8*len(X)):] # Get features from test set.
Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
grid_search.fit(X_train_val, Y_train_val)
```

Out[7]:
```
GridSearchCV(cv=5, error_score='raise',
       estimator=MLPClassifier(activation='relu', alpha=0.0001, batch_size=
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'hidden_layer_sizes': [(1,), (2,), (4,), (8,), (32,), (1
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring=None, verbose=0)
```

In [8]:
```
train_acc = grid_search.cv_results_['mean_train_score'].reshape(6,6)
draw_heatmap_RBF(train_acc, 'train accuracy', momentum_list, hidden_unit_li

val_acc = grid_search.cv_results_['mean_test_score'].reshape(6,6)
draw_heatmap_RBF(val_acc, 'val accuracy',momentum_list, hidden_unit_list)
print(train_acc.shape, val_acc.shape)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.762 0.656 0.653 0.758 0.760 0.455
0.655 0.655 0.659 0.758 0.769 0.654
0.656 0.556 0.557 0.551 0.452 0.561
0.457 0.789 0.666 0.669 0.772 0.562
0.673 0.795 0.675 0.788 0.686 0.679
0.680 0.681 0.463 0.676 0.793 0.682
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.78
0.72
0.66
0.60
0.54
0.48

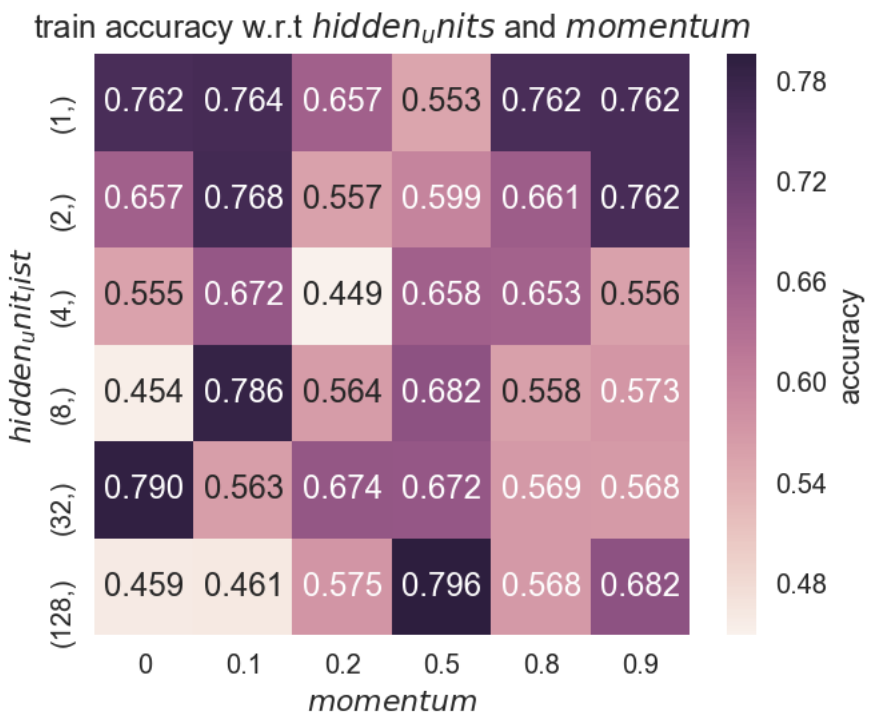

val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.761 0.657 0.656 0.758 0.758 0.458
0.655 0.655 0.656 0.758 0.768 0.653
0.655 0.562 0.554 0.551 0.448 0.558
0.453 0.786 0.662 0.667 0.775 0.565
0.675 0.794 0.678 0.790 0.688 0.675
0.677 0.681 0.461 0.676 0.793 0.681
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.78
0.72
0.66
0.60
0.54
0.48

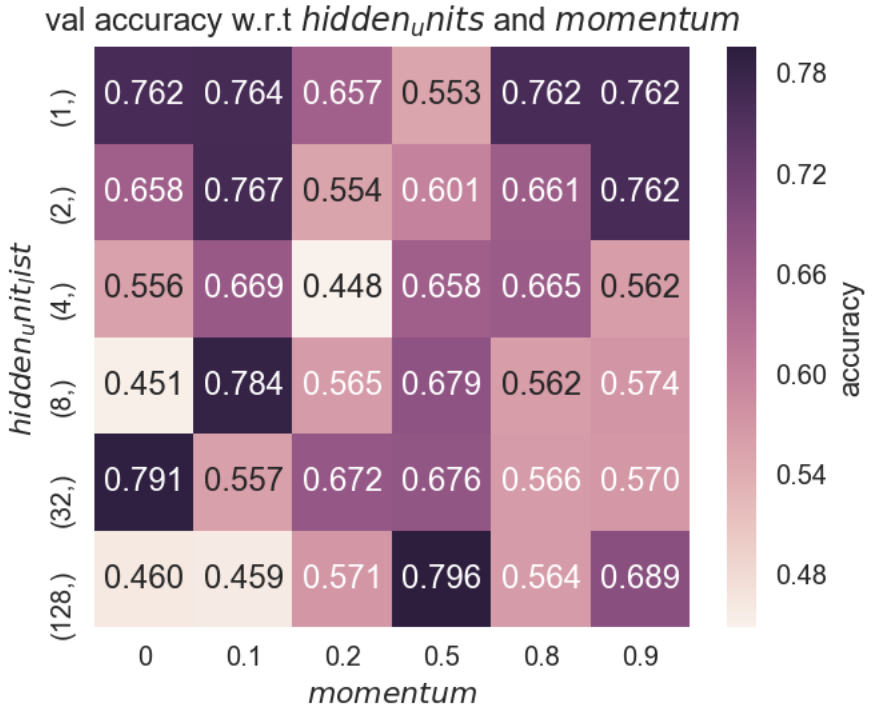
```
(6, 6) (6, 6)
```

```
In [9]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / len
        print(grid_search.best_params_)
        print(test_acc)
        tot_train = (sum(train_acc))/len(train_acc)
        tot_train = (sum(tot_train)/len(tot_train))
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        tot_val = (sum(tot_val)/len(tot_val))
        print(tot_val)
```

```
{'hidden_layer_sizes': (32,), 'momentum': 0.1}
0.778
0.658777141028
0.658395833333
```

```
In [10]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(6,6)
         draw_heatmap_RBF(train_acc2, 'train accuracy', momentum_list, hidden_unit_

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(6,6)
         draw_heatmap_RBF(val_acc2, 'val accuracy',momentum_list, hidden_unit_list)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.762 0.764 0.657 0.553 0.762 0.762
0.657 0.768 0.557 0.599 0.661 0.762
0.555 0.672 0.449 0.658 0.653 0.556
0.454 0.786 0.564 0.682 0.558 0.573
0.790 0.563 0.674 0.672 0.569 0.568
0.459 0.461 0.575 0.796 0.568 0.682
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.78
0.72
0.66
0.60
0.54
0.48

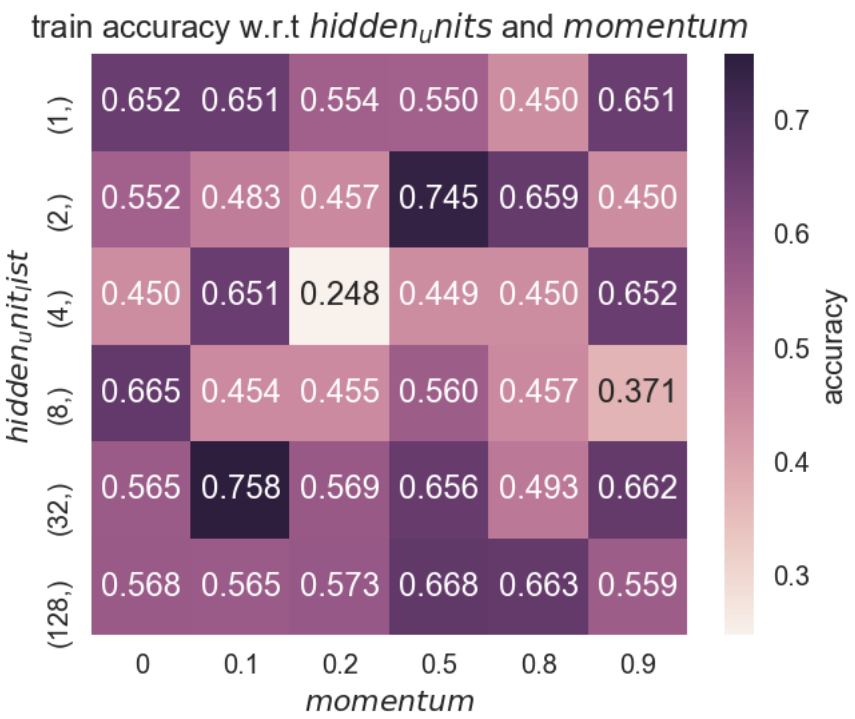

val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.762 0.764 0.657 0.553 0.762 0.762
0.658 0.767 0.554 0.601 0.661 0.762
0.556 0.669 0.448 0.658 0.665 0.562
0.451 0.784 0.565 0.679 0.562 0.574
0.791 0.557 0.672 0.676 0.566 0.570
0.460 0.459 0.571 0.796 0.564 0.689
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.78
0.72
0.66
0.60
0.54
0.48

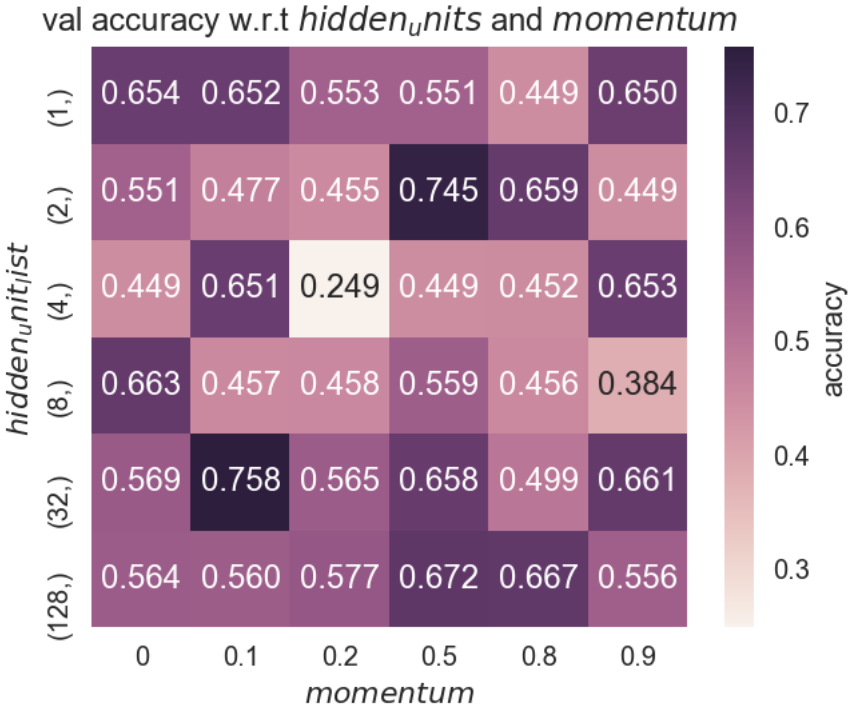
In [11]:
```
test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
print(grid_search.best_params_)
print(test_acc2)
tot_train2 = (sum(train_acc2))/len(train_acc2)
tot_train2 = (sum(tot_train2)/len(tot_train2))
print(tot_train2)
tot_val2 = (sum(val_acc2)/len(val_acc2))
tot_val2 = (sum(tot_val2)/len(tot_val2))
print(tot_val2)
```

```
{'hidden_layer_sizes': (128,), 'momentum': 0.5}
0.784
0.633383333333
0.6335
```

In [12]:
```
X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
X_test      = X[int(0.2*len(X)):] # Get features from test set.
Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
grid_search.fit(X_train_val, Y_train_val)
train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(6,6)
draw_heatmap_RBF(train_acc3, 'train accuracy', momentum_list, hidden_unit_

val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(6,6)
draw_heatmap_RBF(val_acc3, 'val accuracy',momentum_list, hidden_unit_list)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
```


train accuracy w.r.t hidden_units and momentum
hidden_units
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.652 0.651 0.554 0.550 0.450 0.651
0.552 0.483 0.457 0.745 0.659 0.450
0.450 0.651 0.248 0.449 0.450 0.652
0.665 0.454 0.455 0.560 0.457 0.371
0.565 0.758 0.569 0.656 0.493 0.662
0.568 0.565 0.573 0.668 0.663 0.559
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.7
0.6
0.5
0.4
0.3


val accuracy w.r.t hidden_units and momentum
hidden_units
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.654 0.652 0.553 0.551 0.449 0.650
0.551 0.477 0.455 0.745 0.659 0.449
0.449 0.651 0.249 0.449 0.452 0.653
0.663 0.457 0.458 0.559 0.456 0.384
0.569 0.758 0.565 0.658 0.499 0.661
0.564 0.560 0.577 0.672 0.667 0.556
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.7
0.6
0.5
0.4
0.3

```
In [13]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3))/len(train_acc3)
         tot_train3 = (sum(tot_train3)/len(tot_train3))
         print(tot_train3)
         tot_val3 = (sum(val_acc3)/len(val_acc3))
         tot_val3 = (sum(tot_val3)/len(tot_val3))
         print(tot_val3)
```

```
{'hidden_layer_sizes': (32,), 'momentum': 0.1}
0.24025
0.555919453943
0.556416666667
```

```
In [14]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = (tot_train + tot_train2 + tot_train3)/3
         avg_val = (tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)
```

```
0.60075 0.616026642768 0.616104166667
```

```
In [ ]:
```

ANNCov

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/covtype.data.gz', compres
X_and_Y = df.as_matrix()
X_and_Y = X_and_Y[:5000, :55]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

```
(5000, 55) (5000, 54) (5000,)
```

In [3]:
```
#change to binary classification
for i in range(len(Y)):
    if Y[i] == 2:
        Y[i] = 1
    else:
        Y[i] = 0
np.random.shuffle(X_and_Y)
```

In [4]:
```
classifier = MLPClassifier(max_iter = 500)
hidden_unit_list = [(1,), (2,), (4,), (8,), (32,), (128,)]
momentum_list = [0,0.1, 0.2, 0.5, 0.8, 0.9]
params = {'hidden_layer_sizes':hidden_unit_list,'momentum':momentum_list}
grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

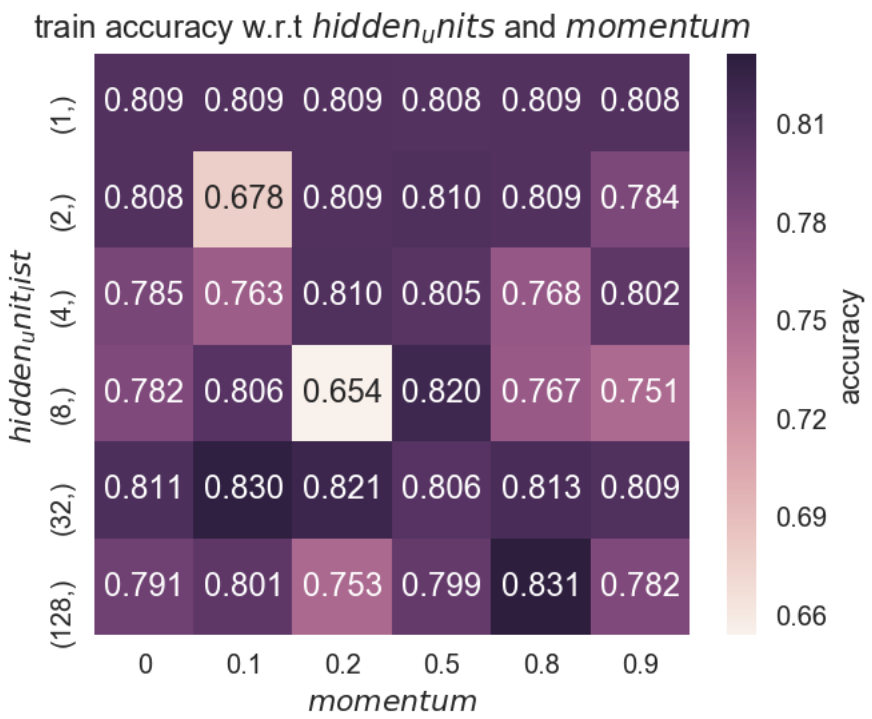
In [5]:
```
def draw_heatmap_RBF(acc, acc_desc, momentum_list, hidden_unit_list):
    plt.figure(figsize = (5,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f',
                     xticklabels=momentum_list, yticklabels=hidden_unit_lis
    ax.collections[0].colorbar.set_label("accuracy")
```

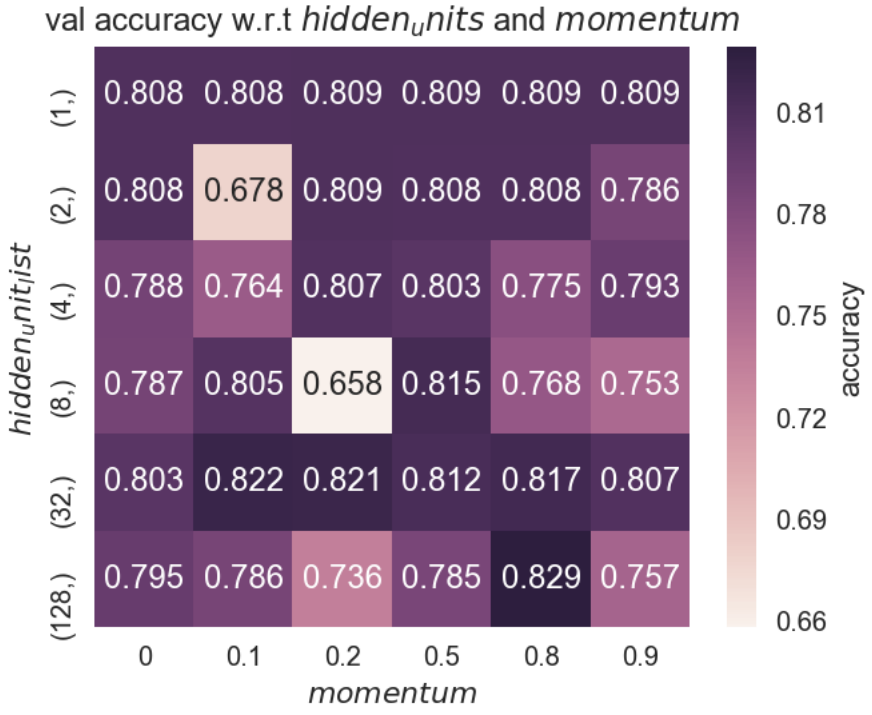
```
        ax.set(xlabel = '$momentum$', ylabel='$hidden_unit_list$')
        plt.title(acc_desc + ' w.r.t $hidden_units$ and $momentum$')
        sns.set_style("whitegrid", {'axes.grid' : False})
        plt.show()
```

```
In [6]: X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
        X_test      = X[int(0.8*len(X)):] # Get features from test set.
        Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
        Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
        grid_search.fit(X_train_val, Y_train_val)
```

```
Out[6]: GridSearchCV(cv=5, error_score='raise',
               estimator=MLPClassifier(activation='relu', alpha=0.0001, batch_size=
               beta_2=0.999, early_stopping=False, epsilon=1e-08,
               hidden_layer_sizes=(100,), learning_rate='constant',
               learning_rate_init=0.001, max_iter=500, momentum=0.9,
               nesterovs_momentum=True, power_t=0.5, random_state=None,
               shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
               verbose=False, warm_start=False),
               fit_params={}, iid=True, n_jobs=1,
               param_grid={'hidden_layer_sizes': [(1,), (2,), (4,), (8,), (32,), (1
               pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
               scoring=None, verbose=0)
```

```
In [7]: train_acc = grid_search.cv_results_['mean_train_score'].reshape(6,6)
        draw_heatmap_RBF(train_acc, 'train accuracy', momentum_list, hidden_unit_li

        val_acc = grid_search.cv_results_['mean_test_score'].reshape(6,6)
        draw_heatmap_RBF(val_acc, 'val accuracy',momentum_list, hidden_unit_list)
        print(train_acc.shape, val_acc.shape)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,) 0.809 0.809 0.809 0.808 0.809 0.808
(2,) 0.808 0.678 0.809 0.810 0.809 0.784
(4,) 0.785 0.763 0.810 0.805 0.768 0.802
(8,) 0.782 0.806 0.654 0.820 0.767 0.751
(32,) 0.811 0.830 0.821 0.806 0.813 0.809
(128,) 0.791 0.801 0.753 0.799 0.831 0.782
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.81 0.78 0.75 0.72 0.69 0.66

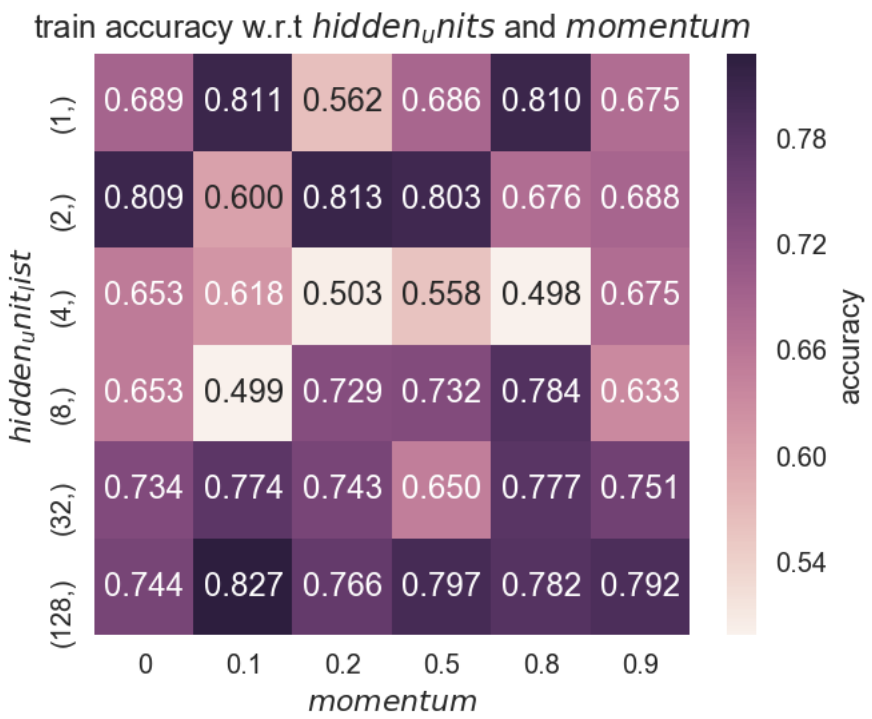

val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,) 0.808 0.808 0.809 0.809 0.809 0.809
(2,) 0.808 0.678 0.809 0.808 0.808 0.786
(4,) 0.788 0.764 0.807 0.803 0.775 0.793
(8,) 0.787 0.805 0.658 0.815 0.768 0.753
(32,) 0.803 0.822 0.821 0.812 0.817 0.807
(128,) 0.795 0.786 0.736 0.785 0.829 0.757
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.81 0.78 0.75 0.72 0.69 0.66

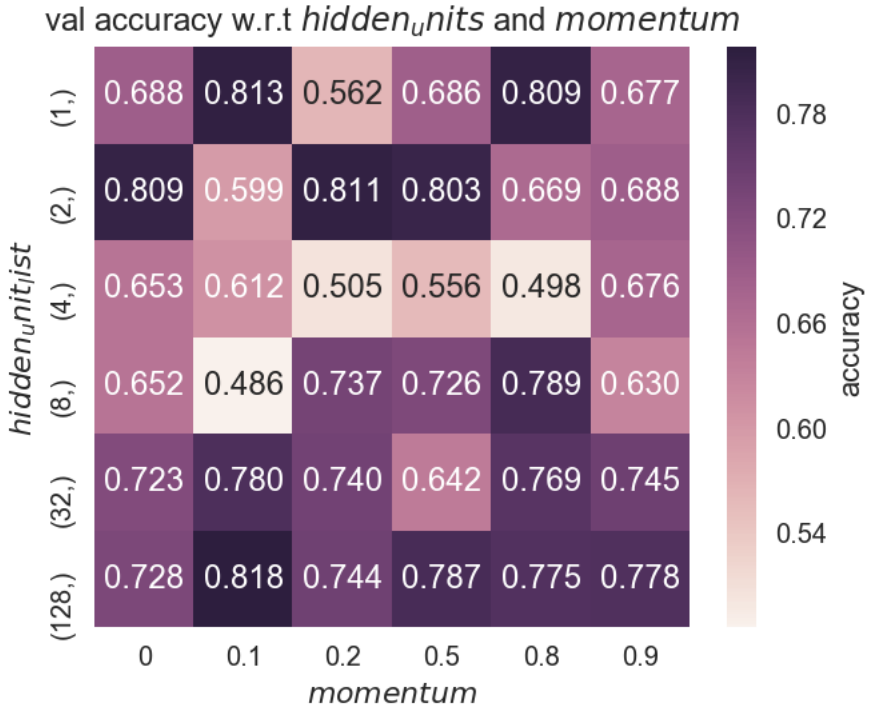
```
(6, 6) (6, 6)
```

```
In [8]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / len
        print(grid_search.best_params_)
        print(test_acc)
        tot_train = (sum(train_acc))/len(train_acc)
        tot_train = (sum(tot_train)/len(tot_train))
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        tot_val = (sum(tot_val)/len(tot_val))
        print(tot_val)
```

```
{'hidden_layer_sizes': (128,), 'momentum': 0.8}
0.843
0.791872336509
0.789923611111
```

```
In [9]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
        X_test      = X[int(0.5*len(X)):] # Get features from test set.
        Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
        Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
        grid_search.fit(X_train_val, Y_train_val)
        train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(6,6)
        draw_heatmap_RBF(train_acc2, 'train accuracy', momentum_list, hidden_unit_l

        val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(6,6)
        draw_heatmap_RBF(val_acc2, 'val accuracy',momentum_list, hidden_unit_list)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.689 0.811 0.562 0.686 0.810 0.675
0.809 0.600 0.813 0.803 0.676 0.688
0.653 0.618 0.503 0.558 0.498 0.675
0.653 0.499 0.729 0.732 0.784 0.633
0.734 0.774 0.743 0.650 0.777 0.751
0.744 0.827 0.766 0.797 0.782 0.792
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.78
0.72
0.66
0.60
0.54


val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.688 0.813 0.562 0.686 0.809 0.677
0.809 0.599 0.811 0.803 0.669 0.688
0.653 0.612 0.505 0.556 0.498 0.676
0.652 0.486 0.737 0.726 0.789 0.630
0.723 0.780 0.740 0.642 0.769 0.745
0.728 0.818 0.744 0.787 0.775 0.778
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.78
0.72
0.66
0.60
0.54

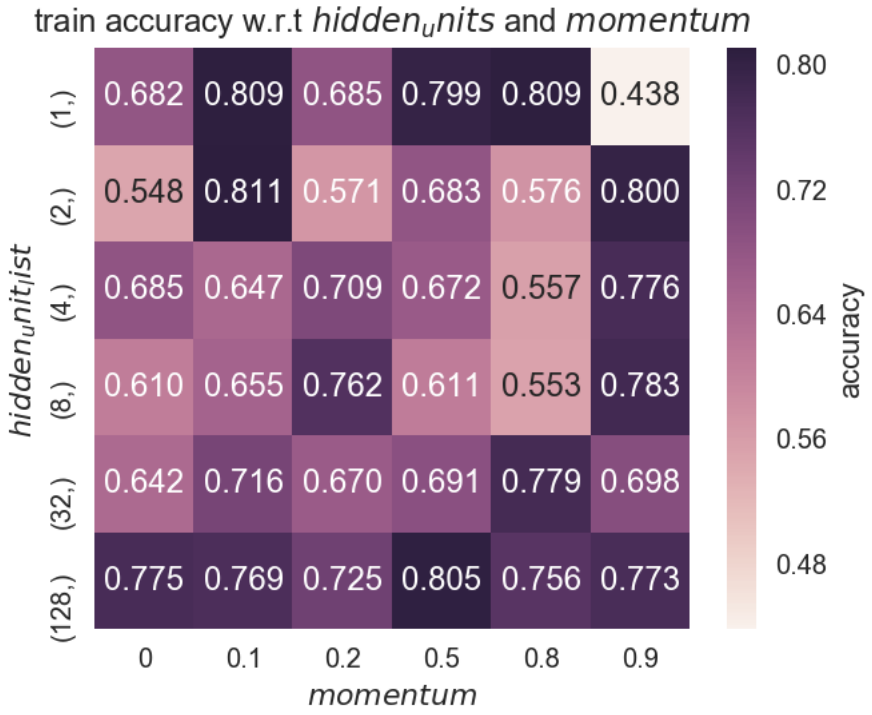
```
In [10]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2))/len(train_acc2)
         tot_train2 = (sum(tot_train2)/len(tot_train2))
         print(tot_train2)
         tot_val2 = (sum(val_acc2)/len(val_acc2))
         tot_val2 = (sum(tot_val2)/len(tot_val2))
         print(tot_val2)
```

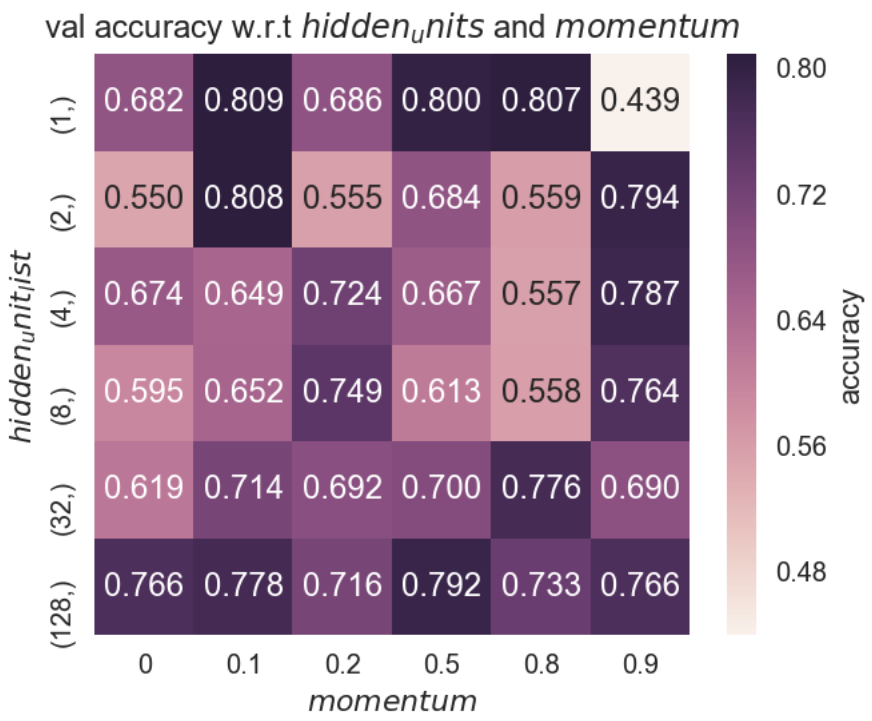
```
{'hidden_layer_sizes': (128,), 'momentum': 0.1}
0.8184
0.702573032145
0.699
```

```
In [11]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(6,6)
         draw_heatmap_RBF(train_acc3, 'train accuracy', momentum_list, hidden_unit_

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(6,6)
         draw_heatmap_RBF(val_acc3, 'val accuracy',momentum_list, hidden_unit_list)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
```

```
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
```






```
In [12]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3))/len(train_acc3)
         tot_train3 = (sum(tot_train3)/len(tot_train3))
         print(tot_train3)
         tot_val3 = (sum(val_acc3)/len(val_acc3))
         tot_val3 = (sum(tot_val3)/len(tot_val3))
         print(tot_val3)
```

```
{'hidden_layer_sizes': (1,), 'momentum': 0.1}
0.81075
0.695223363023
0.691777777778
```

```
In [13]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = (tot_train + tot_train2 + tot_train3)/3
         avg_val = (tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)
```

```
0.82405 0.729889577226 0.726900462963
```

```
In [ ]:
```

ANNLETTERS

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/letter-recognition.data',
df2 = df.copy()
df = df.drop([0], axis=1)
df = df.join(df2[0])
```

In [3]:
```
X_and_Y = df.as_matrix()
np.random.shuffle(X_and_Y)
X_and_Y = X_and_Y[:5000, :17]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

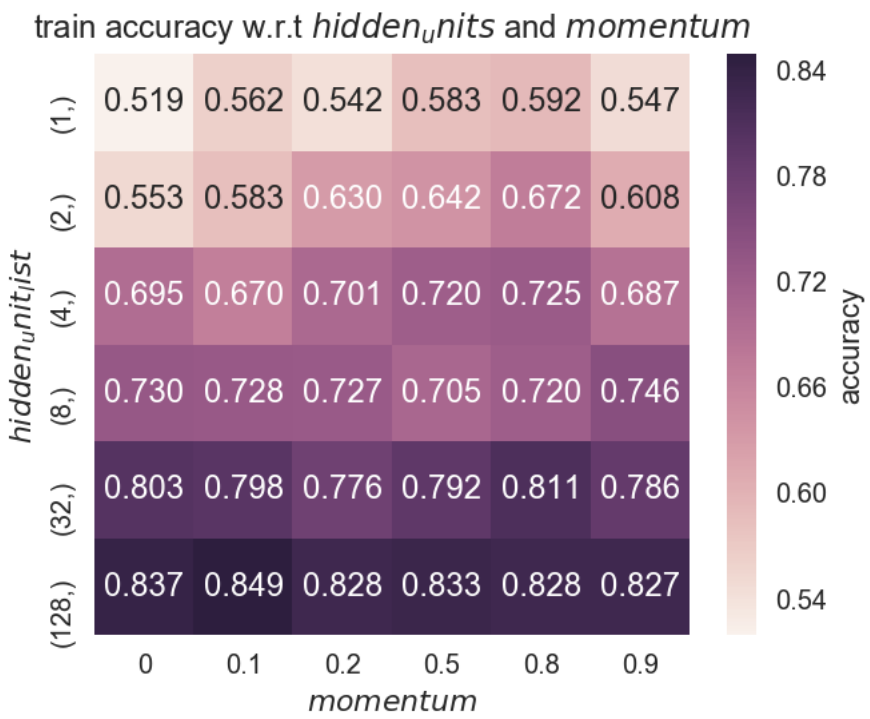
```
(5000, 17) (5000, 16) (5000,)
```

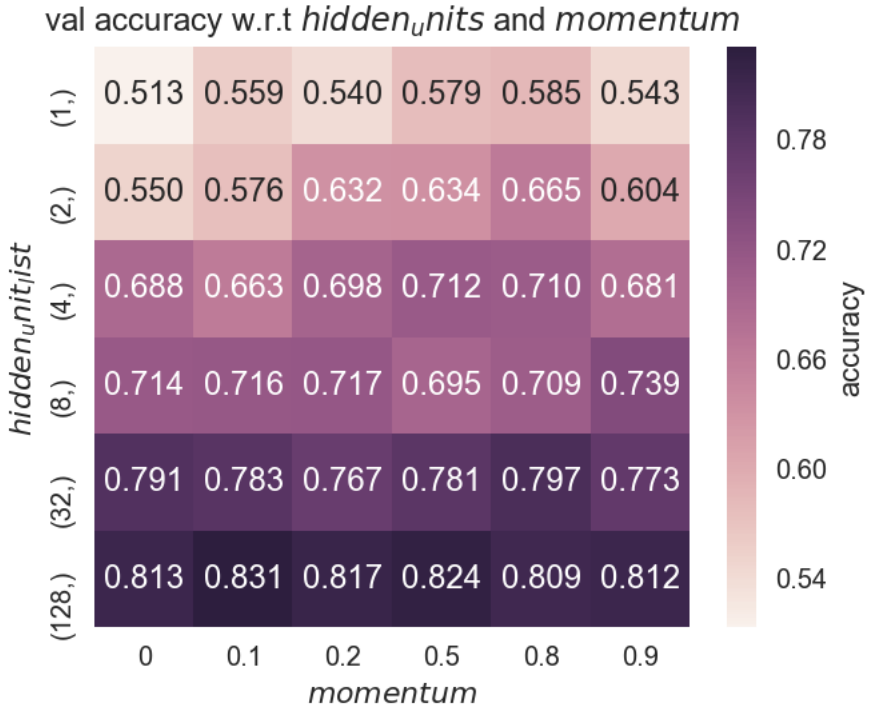
In [4]:
```
ordlist = []
for i in range(len(Y)):
    ordlist.append(ord(Y[i]))
```

In [5]:
```
for i in range(len(Y)):
    if ((ordlist[i]) >= 65 and (ordlist[i] <= 77)):
        Y[i] = 1
    else:
        Y[i] = 0
```

In [6]:
```
classifier = MLPClassifier(max_iter = 500)
hidden_unit_list = [(1,), (2,), (4,), (8,), (32,), (128,)]
momentum_list = [0,0.1, 0.2, 0.5, 0.8, 0.9]
params = {'hidden_layer_sizes':hidden_unit_list,'momentum':momentum_list}
grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

```
In [7]: def draw_heatmap_RBF(acc, acc_desc, momentum_list, hidden_unit_list):
            plt.figure(figsize = (5,4))
            ax = sns.heatmap(acc, annot=True, fmt='.3f',
                             xticklabels=momentum_list, yticklabels=hidden_unit_lis
            ax.collections[0].colorbar.set_label("accuracy")
            ax.set(xlabel = '$momentum$', ylabel='$hidden_unit_list$')
            plt.title(acc_desc + ' w.r.t $hidden_units$ and $momentum$')
            sns.set_style("whitegrid", {'axes.grid' : False})
            plt.show()

In [8]: Y = Y.astype('int')

        X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
        X_test      = X[int(0.8*len(X)):] # Get features from test set.
        Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
        Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
        grid_search.fit(X_train_val, Y_train_val)

Out[8]: GridSearchCV(cv=5, error_score='raise',
               estimator=MLPClassifier(activation='relu', alpha=0.0001, batch_size=
               beta_2=0.999, early_stopping=False, epsilon=1e-08,
               hidden_layer_sizes=(100,), learning_rate='constant',
               learning_rate_init=0.001, max_iter=500, momentum=0.9,
               nesterovs_momentum=True, power_t=0.5, random_state=None,
               shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
               verbose=False, warm_start=False),
               fit_params={}, iid=True, n_jobs=1,
               param_grid={'hidden_layer_sizes': [(1,), (2,), (4,), (8,), (32,), (1
               pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
               scoring=None, verbose=0)

In [9]: train_acc = grid_search.cv_results_['mean_train_score'].reshape(6,6)
        draw_heatmap_RBF(train_acc, 'train accuracy', momentum_list, hidden_unit_li

        val_acc = grid_search.cv_results_['mean_test_score'].reshape(6,6)
        draw_heatmap_RBF(val_acc, 'val accuracy',momentum_list, hidden_unit_list)
        print(train_acc.shape, val_acc.shape)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.519 0.562 0.542 0.583 0.592 0.547
0.553 0.583 0.630 0.642 0.672 0.608
0.695 0.670 0.701 0.720 0.725 0.687
0.730 0.728 0.727 0.705 0.720 0.746
0.803 0.798 0.776 0.792 0.811 0.786
0.837 0.849 0.828 0.833 0.828 0.827
0 0.1 0.2 0.5 0.8 0.9
momentum
0.84 0.78 0.72 0.66 0.60 0.54
accuracy


val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.513 0.559 0.540 0.579 0.585 0.543
0.550 0.576 0.632 0.634 0.665 0.604
0.688 0.663 0.698 0.712 0.710 0.681
0.714 0.716 0.717 0.695 0.709 0.739
0.791 0.783 0.767 0.781 0.797 0.773
0.813 0.831 0.817 0.824 0.809 0.812
0 0.1 0.2 0.5 0.8 0.9
momentum
0.78 0.72 0.66 0.60 0.54
accuracy

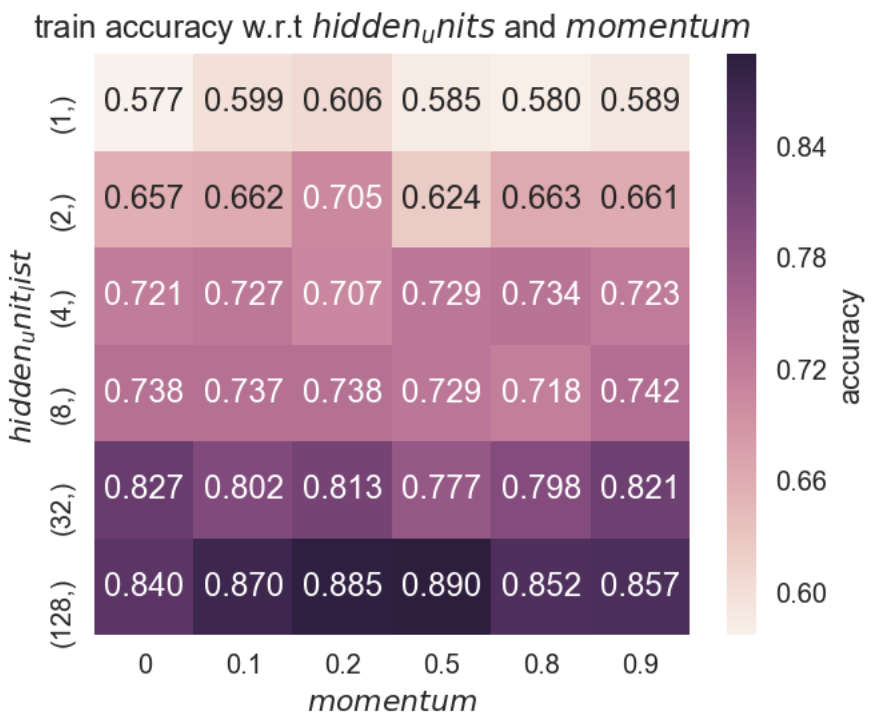
```
(6, 6) (6, 6)
```

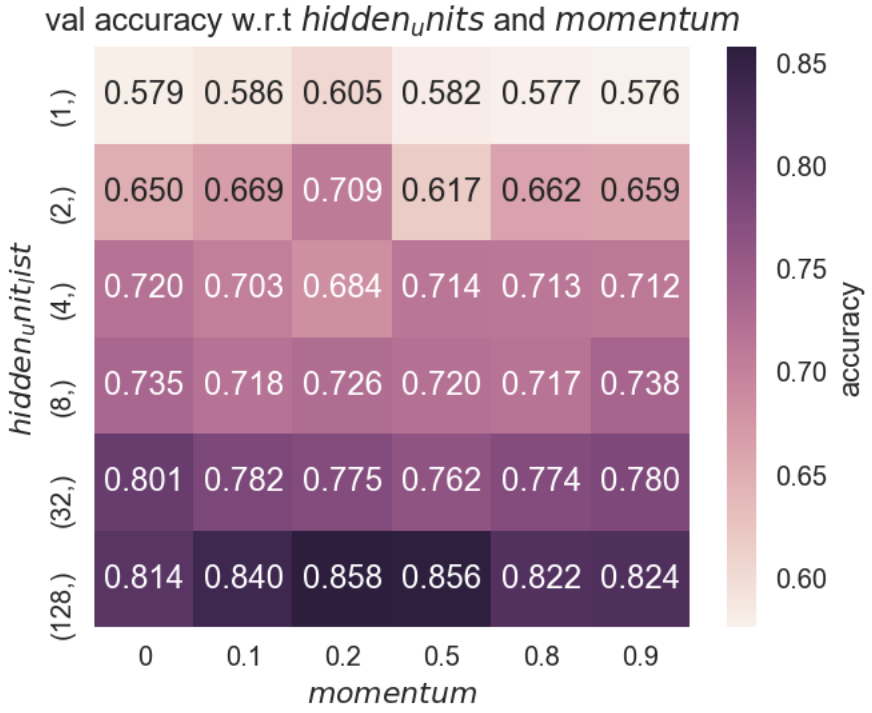
```
In [10]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / le
         print(grid_search.best_params_)
         print(test_acc)
         tot_train = (sum(train_acc))/len(train_acc)
         tot_train = (sum(tot_train)/len(tot_train))
         print(tot_train)
         tot_val = (sum(val_acc)/len(val_acc))
         tot_val = (sum(tot_val)/len(tot_val))
         print(tot_val)
```

```
{'hidden_layer_sizes': (128,), 'momentum': 0.1}
0.88
0.704265253377
0.694979166667
```

```
In [11]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(6,6)
         draw_heatmap_RBF(train_acc2, 'train accuracy', momentum_list, hidden_unit_

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(6,6)
         draw_heatmap_RBF(val_acc2, 'val accuracy',momentum_list, hidden_unit_list)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.577 0.599 0.606 0.585 0.580 0.589
0.657 0.662 0.705 0.624 0.663 0.661
0.721 0.727 0.707 0.729 0.734 0.723
0.738 0.737 0.738 0.729 0.718 0.742
0.827 0.802 0.813 0.777 0.798 0.821
0.840 0.870 0.885 0.890 0.852 0.857
0
0.1
0.2
0.5
0.8
0.9
momentum
0.84
0.78
0.72
0.66
0.60
accuracy


val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.579 0.586 0.605 0.582 0.577 0.576
0.650 0.669 0.709 0.617 0.662 0.659
0.720 0.703 0.684 0.714 0.713 0.712
0.735 0.718 0.726 0.720 0.717 0.738
0.801 0.782 0.775 0.762 0.774 0.780
0.814 0.840 0.858 0.856 0.822 0.824
0
0.1
0.2
0.5
0.8
0.9
momentum
0.85
0.80
0.75
0.70
0.65
0.60
accuracy

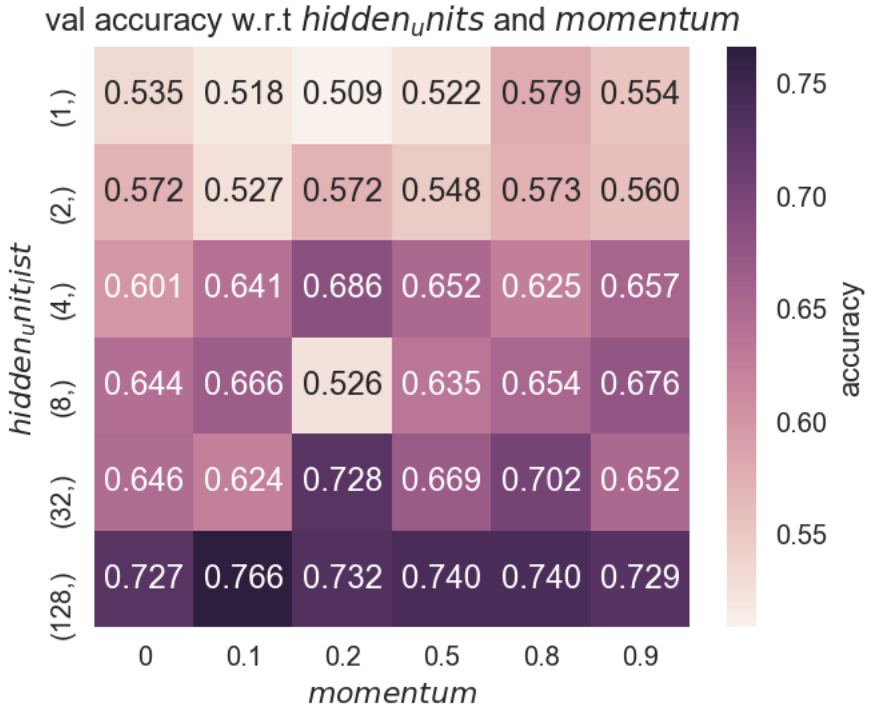
```
In [12]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2))/len(train_acc2)
         tot_train2 = (sum(tot_train2)/len(tot_train2))
         print(tot_train2)
         tot_val2 = (sum(val_acc2)/len(val_acc2))
         tot_val2 = (sum(tot_val2)/len(tot_val2))
         print(tot_val2)
```

```
{'hidden_layer_sizes': (128,), 'momentum': 0.2}
0.8572
0.730083333333
0.715588888889
```

```
In [13]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(6,6)
         draw_heatmap_RBF(train_acc3, 'train accuracy', momentum_list, hidden_unit_

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(6,6)
         draw_heatmap_RBF(val_acc3, 'val accuracy',momentum_list, hidden_unit_list)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_percep
  % (), ConvergenceWarning)
```


train accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.532 0.519 0.507 0.528 0.591 0.567
0.596 0.533 0.560 0.546 0.582 0.577
0.597 0.637 0.669 0.667 0.628 0.656
0.651 0.668 0.534 0.659 0.674 0.676
0.670 0.663 0.728 0.690 0.744 0.652
0.750 0.794 0.728 0.761 0.778 0.749
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.75
0.70
0.65
0.60
0.55


val accuracy w.r.t hidden_units and momentum
hidden_unit_list
(1,)
(2,)
(4,)
(8,)
(32,)
(128,)
0.535 0.518 0.509 0.522 0.579 0.554
0.572 0.527 0.572 0.548 0.573 0.560
0.601 0.641 0.686 0.652 0.625 0.657
0.644 0.666 0.526 0.635 0.654 0.676
0.646 0.624 0.728 0.669 0.702 0.652
0.727 0.766 0.732 0.740 0.740 0.729
0 0.1 0.2 0.5 0.8 0.9
momentum
accuracy
0.75
0.70
0.65
0.60
0.55

```
In [14]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3))/len(train_acc3)
         tot_train3 = (sum(tot_train3)/len(tot_train3))
         print(tot_train3)
         tot_val3 = (sum(val_acc3)/len(val_acc3))
         tot_val3 = (sum(tot_val3)/len(tot_val3))
         print(tot_val3)
```

```
{'hidden_layer_sizes': (128,), 'momentum': 0.1}
0.785
0.640501815953
0.630194444444
```

```
In [15]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = (tot_train + tot_train2 + tot_train3)/3
         avg_val = (tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)
```

```
0.840733333333 0.691616800888 0.680254166667
```

```
In [ ]:
```

DecisionTreeADULT

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn import tree
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/adult.csv', index_col = F
df = df.dropna(axis=0)
onehot = pd.get_dummies(data=df, columns=['Column2','Column4', 'Column6', '
                        drop_first = True)
```

In [3]:
```
onehot = onehot.drop(['Column15'], axis=1)
```

In [4]:
```
df = onehot.join(df['Column15'])
df = df.dropna(axis=1)
df = df[df.notnull()]
```

In [5]:
```
X_and_Y = df.as_matrix()
np.random.shuffle(X_and_Y)
X_and_Y = X_and_Y[:5000, :101]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

```
(5000, 101) (5000, 100) (5000,)
```

In [6]:
```
#change to binary classification
for i in range(len(Y)):
    if Y[i] == ' >50K':
        Y[i] = 1
    else:
        Y[i] = 0
```

In [7]:
```
print(Y[:200])
```

```
[0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0
 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0
 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0
 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0]
```

In [8]:
```
depth_list = []
for i in range(1, 21, 1):
    depth_list.append(i)
print(depth_list)
```

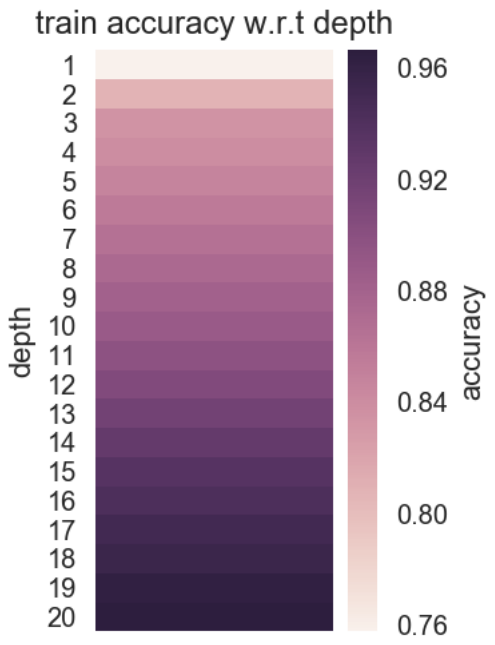
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

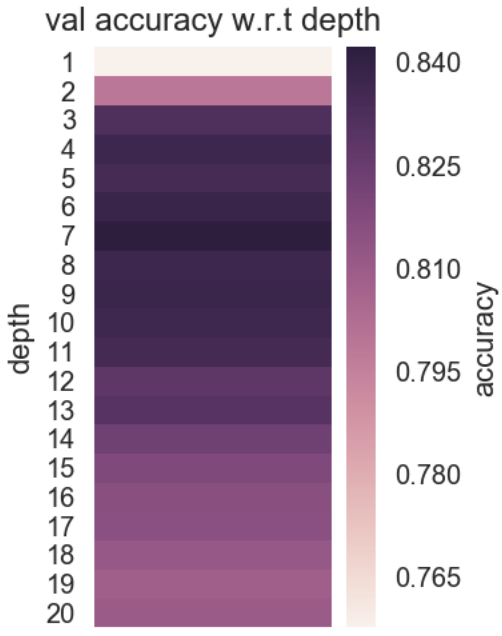
In [9]:
```
classifier = tree.DecisionTreeClassifier(criterion = 'entropy')
params = {'max_depth':depth_list}
grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

In [10]:
```
def draw_heatmap_linear(acc, acc_desc, depth_list):
    plt.figure(figsize = (2,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=depth_list, x
    ax.collections[0].colorbar.set_label("accuracy")
    ax.set(ylabel='depth')
    plt.title(acc_desc + ' w.r.t depth')
    sns.set_style("whitegrid", {'axes.grid' : False})
    plt.show()
```

In [11]:
```
Y = Y.astype('int')
X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
X_test      = X[int(0.8*len(X)):] # Get features from test set.
Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
```

In [12]:
```
grid_search.fit(X_train_val, Y_train_val)
train_acc = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

val_acc = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```



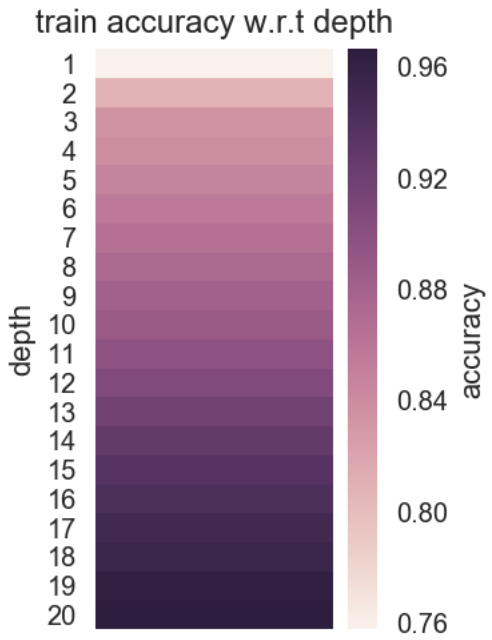


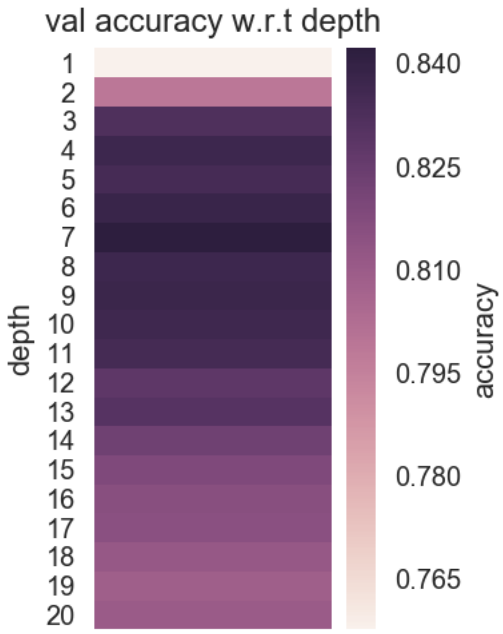

```
In [13]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / le
         print(grid_search.best_params_)
         print(test_acc)
         tot_train = (sum(train_acc)/len(train_acc))
```

```
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        print(tot_val)
```

```
{'max_depth': 7}
0.84
[ 0.8909721]
[ 0.822575]
```

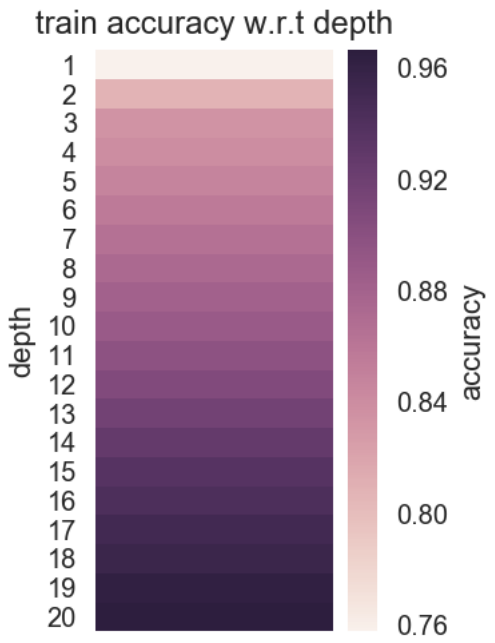
```
In [14]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```

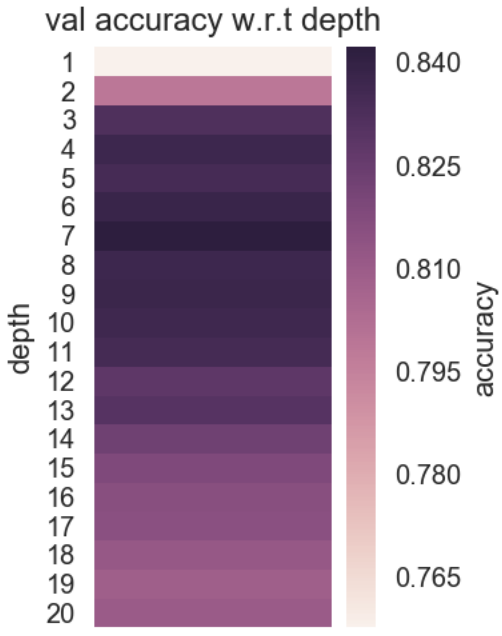





```
In [15]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2)/len(train_acc2))
         print(tot_train2)
         tot_val2 = (sum(val_acc2)/len(val_acc2))
         print(tot_val2)

{'max_depth': 8}
0.8472
[ 0.8934459]
[ 0.81122]
```

```
In [16]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```






```
In [17]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3)/len(train_acc3))
```

```
        print(tot_train3)
        tot_val3 = (sum(val_acc3)/len(val_acc3))
        print(tot_val3)
```

```
{'max_depth': 9}
0.81575
[ 0.89980536]
[ 0.7917]
```

In [18]:
```
avg_test = (test_acc + test_acc2 + test_acc3)/3
avg_train = sum(tot_train + tot_train2 + tot_train3)/3
avg_val = sum(tot_val + tot_val2 + tot_val3)/3
print(avg_test, avg_train, avg_val)
```

```
0.834316666667 0.894741119271 0.808498333333
```

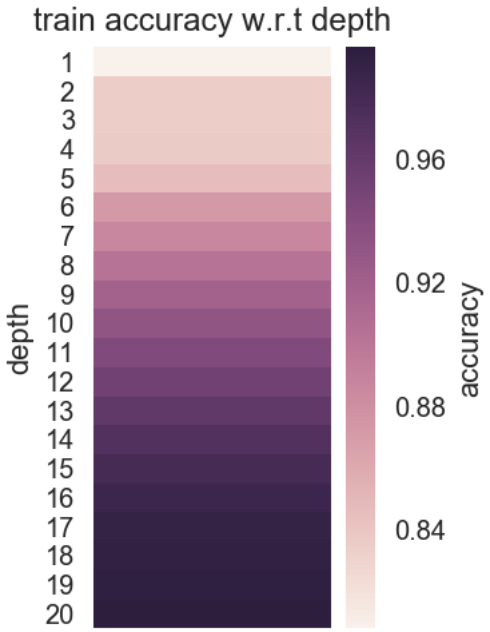
In [ ]:

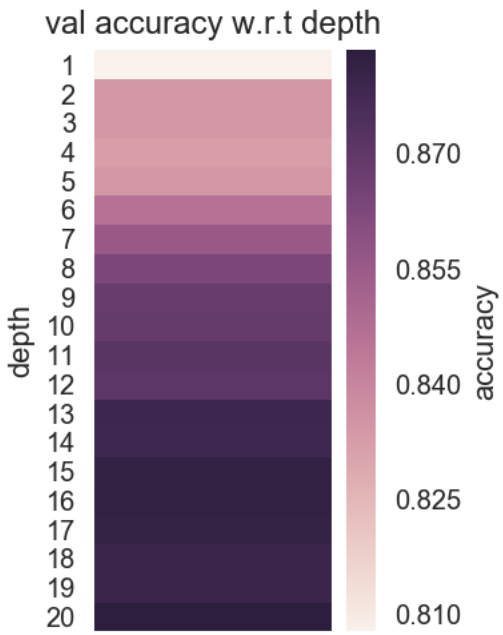
DecisionTreeCOV

March 24, 2018

```
In [1]: import scipy.io as sio
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import pandas as pd
        from sklearn import tree
        from sklearn.model_selection import GridSearchCV
        %config InlineBackend.figure_format = 'retina'

In [2]: df = pd.read_csv('C:/Users/Preston Wong/Downloads/covtype.data.gz', compres
        df.shape

Out[2]: (581011, 55)

In [3]: X_and_Y = df.as_matrix()
        X_and_Y = X_and_Y[:5000, :55]
        X = X_and_Y[:, 0:-1]
        Y = X_and_Y[:, -1]
        print(X_and_Y.shape, X.shape, Y.shape)

(5000, 55) (5000, 54) (5000,)


In [4]: #change to binary classification
        for i in range(len(Y)):
            if Y[i] == 2:
                Y[i] = 1
            else:
                Y[i] = 0
        np.random.shuffle(X_and_Y)

In [5]: depth_list = []
        for i in range(1, 21, 1):
            depth_list.append(i)
        print(depth_list)

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

```
In [6]: classifier = tree.DecisionTreeClassifier(criterion = 'entropy')
        params = {'max_depth':depth_list}
        grid_search = GridSearchCV(classifier, params, return_train_score = True, c

In [7]: def draw_heatmap_linear(acc, acc_desc, depth_list):
            plt.figure(figsize = (2,4))
            ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=depth_list, xt
            ax.collections[0].colorbar.set_label("accuracy")
            ax.set(ylabel='depth')
            plt.title(acc_desc + ' w.r.t depth')
            sns.set_style("whitegrid", {'axes.grid' : False})
            plt.show()

In [8]: X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
        X_test      = X[int(0.8*len(X)):] # Get features from test set.
        Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
        Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
        grid_search.fit(X_train_val, Y_train_val)
        train_acc = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
        draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

        val_acc = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
        draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```

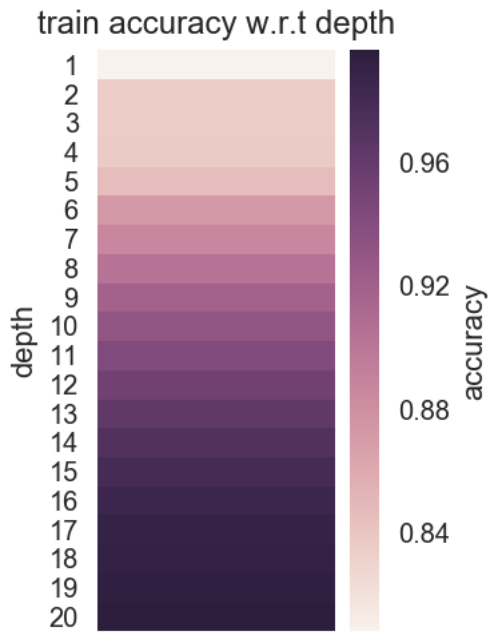


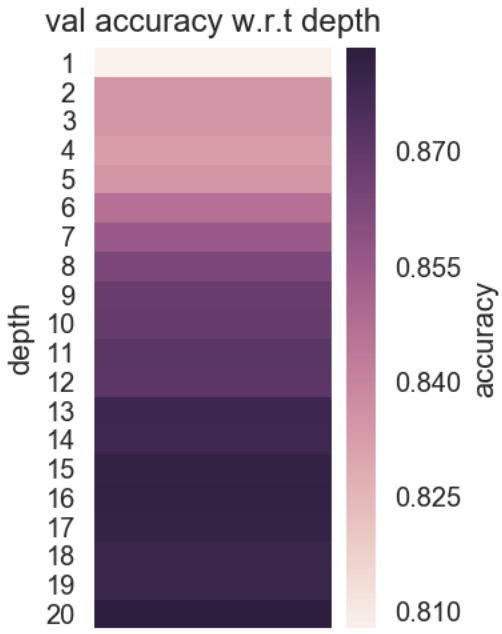



```
In [9]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / len
        print(grid_search.best_params_)
        print(test_acc)
        tot_train = (sum(train_acc)/len(train_acc))
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        print(tot_val)
```

```
{'max_depth': 20}
0.893
[ 0.92231545]
[ 0.861775]
```

```
In [10]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```



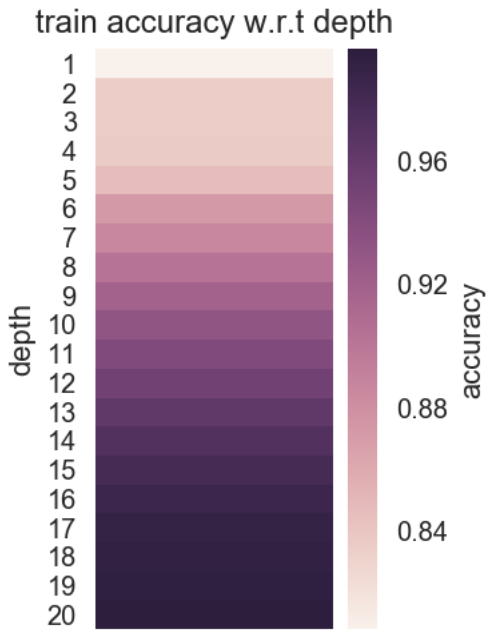


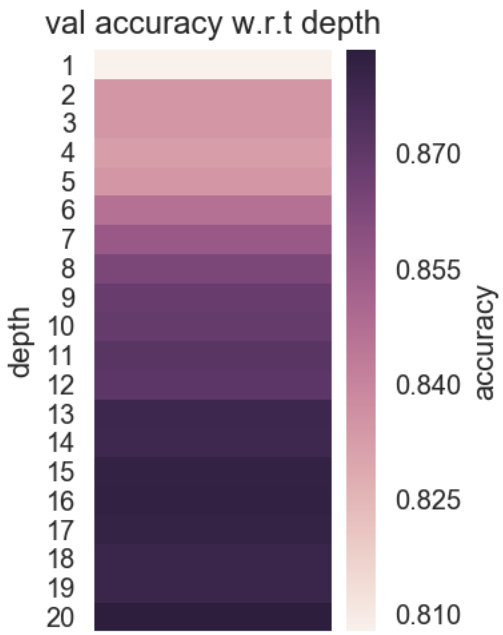

```
In [11]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2)/len(train_acc2))
```

```
        print(tot_train2)
        tot_val2 = (sum(val_acc2)/len(val_acc2))
        print(tot_val2)
```

```
{'max_depth': 13}
0.8864
[ 0.92602057]
[ 0.86132]
```

```
In [12]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```






```
In [13]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3)/len(train_acc3))
         print(tot_train3)
         tot_val3 = (sum(val_acc3)/len(val_acc3))
         print(tot_val3)

{'max_depth': 18}
0.86325
[ 0.93723405]
[ 0.8538]


In [14]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = sum(tot_train + tot_train2 + tot_train3)/3
         avg_val = sum(tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)

0.880883333333 0.928523357211 0.858965
```

DecisionTreeLETTERS

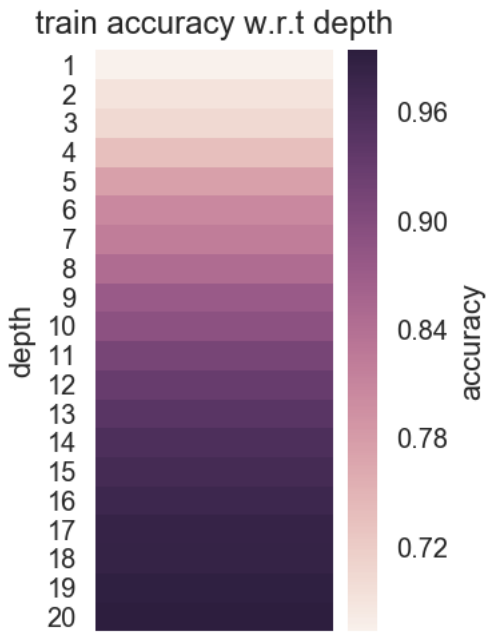
March 24, 2018

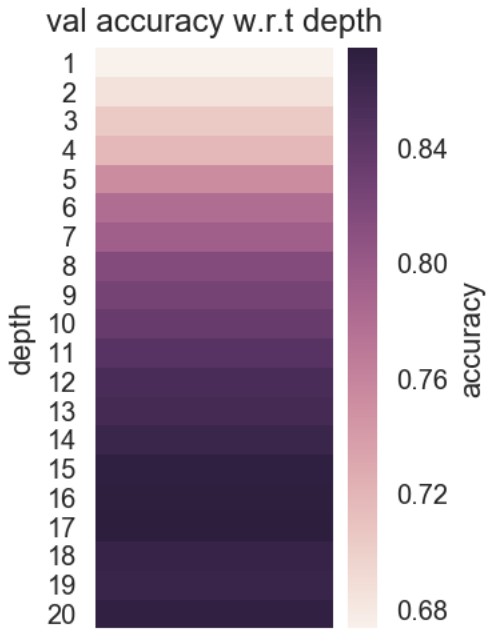
```
In [1]: import scipy.io as sio
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import pandas as pd
        from sklearn import tree
        from sklearn.model_selection import GridSearchCV
        from string import ascii_uppercase
        %config InlineBackend.figure_format = 'retina'

In [2]: df = pd.read_csv('C:/Users/Preston Wong/Downloads/letter-recognition.data',

In [3]: df2 = df.copy()
        df = df.drop([0], axis=1)
        df = df.join(df2[0])

In [4]: df.shape

Out[4]: (20000, 17)

In [5]: X_and_Y = df.as_matrix()
        #np.random.shuffle(X_and_Y)
        X_and_Y = X_and_Y[:5000, :17]
        X = X_and_Y[:, 0:-1]
        Y = X_and_Y[:, -1]
        print(X_and_Y.shape, X.shape, Y.shape)

(5000, 17) (5000, 16) (5000,)


In [6]: #change to binary classification
        #alphabet_list = []
        ordlist = []
        for i in range(len(Y)):
            #alphabet_list.append(Y[i])
            ordlist.append(ord(Y[i]))

In [7]: #print(alphabet_list)
        #print(ordlist)
```

```
In [8]: for i in range(len(Y)):
            if ((ordlist[i]) >= 65 and (ordlist[i] <= 77)):
                Y[i] = 1
            else:
                Y[i] = 0

In [9]: depth_list = []
        for i in range(1, 21, 1):
            depth_list.append(i)
        print(depth_list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

```
In [10]: classifier = tree.DecisionTreeClassifier(criterion = 'entropy')
         params = {'max_depth':depth_list}
         grid_search = GridSearchCV(classifier, params, return_train_score = True,

In [11]: def draw_heatmap_linear(acc, acc_desc, depth_list):
             plt.figure(figsize = (2,4))
             ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=depth_list, x
             ax.collections[0].colorbar.set_label("accuracy")
             ax.set(ylabel='depth')
             plt.title(acc_desc + ' w.r.t depth')
             sns.set_style("whitegrid", {'axes.grid' : False})
             plt.show()

In [12]: Y = Y.astype('int')
         X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
         X_test      = X[int(0.8*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.

In [13]: grid_search.fit(X_train_val, Y_train_val)
         train_acc = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```



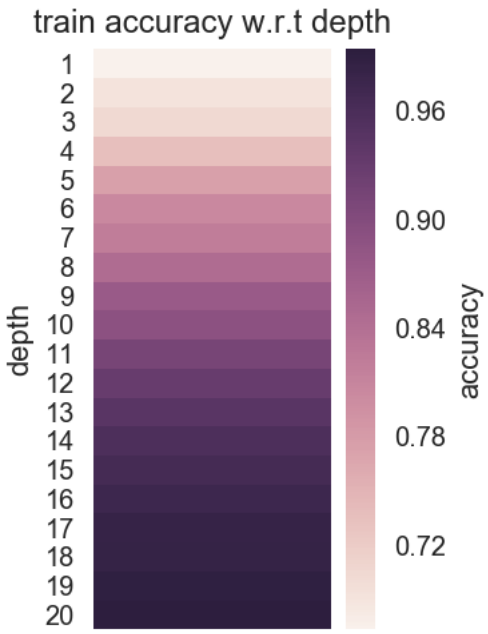


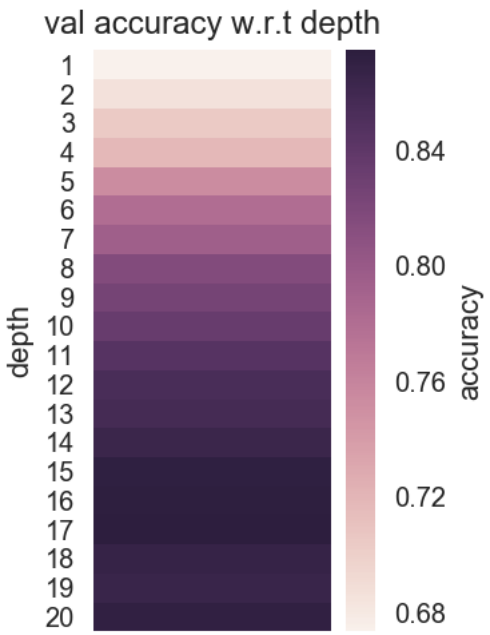

```
In [14]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / le
         print(grid_search.best_params_)
         print(test_acc)
         tot_train = (sum(train_acc)/len(train_acc))
```

```
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        print(tot_val)
```

```
{'max_depth': 17}
0.88
[ 0.87318125]
[ 0.8118375]
```

```
In [15]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```

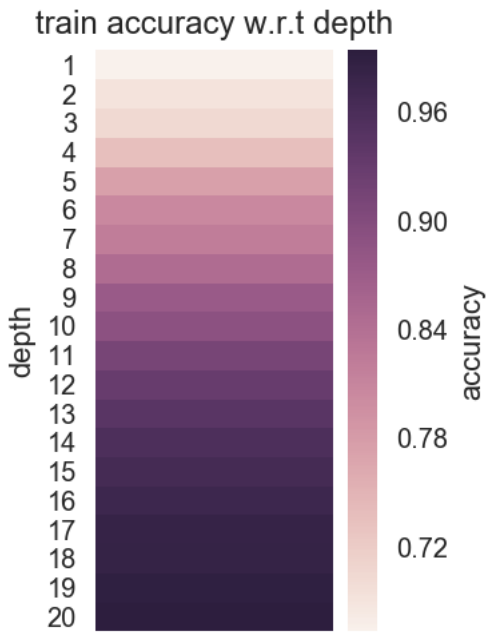


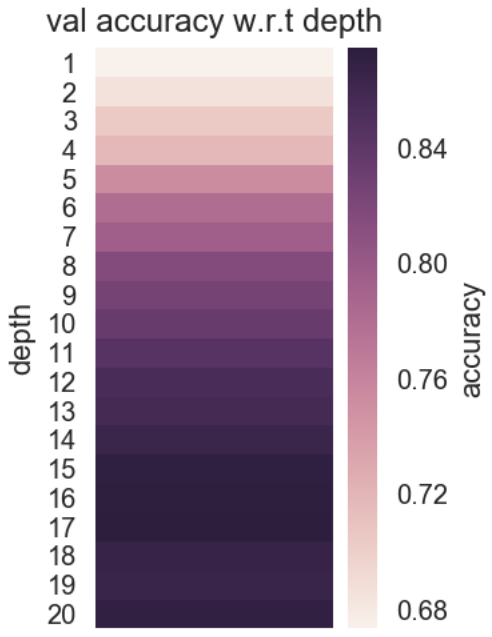



```
In [16]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2)/len(train_acc2))
         print(tot_train2)
         tot_val2 = (sum(val_acc2)/len(val_acc2))
         print(tot_val2)
```

```
{'max_depth': 17}
0.8468
[ 0.8841622]
[ 0.80396]
```

```
In [17]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```






```
In [18]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3)/len(train_acc3))
```

```
        print(tot_train3)
        tot_val3 = (sum(val_acc3)/len(val_acc3))
        print(tot_val3)
```

```
{'max_depth': 18}
0.8175
[ 0.88418269]
[ 0.75975]
```

In [19]:
```
avg_test = (test_acc + test_acc2 + test_acc3)/3
avg_train = sum(tot_train + tot_train2 + tot_train3)/3
avg_val = sum(tot_val + tot_val2 + tot_val3)/3
print(avg_test, avg_train, avg_val)
```

```
0.8481 0.880508712379 0.791849166667
```

In [ ]:

RandomForestADULT

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/adult.csv', index_col = F
df = df.dropna(axis=0)
onehot = pd.get_dummies(data=df, columns=['Column2','Column4', 'Column6', '
                        drop_first = True)
onehot = onehot.drop(['Column15'], axis=1)
df = onehot.join(df['Column15'])
df = df.dropna(axis=1)
df = df[df.notnull()]
```

In [3]:
```
X_and_Y = df.as_matrix()
np.random.shuffle(X_and_Y)
X_and_Y = X_and_Y[:5000, :101]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

```
(5000, 101) (5000, 100) (5000,)
```

In [4]:
```
#change to binary classification
for i in range(len(Y)):
    if Y[i] == ' >50K':
        Y[i] = 1
    else:
        Y[i] = 0
print(Y[:200])
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
```

```
 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1
 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1]
```

In [5]:
```
print(Y[:200])
```

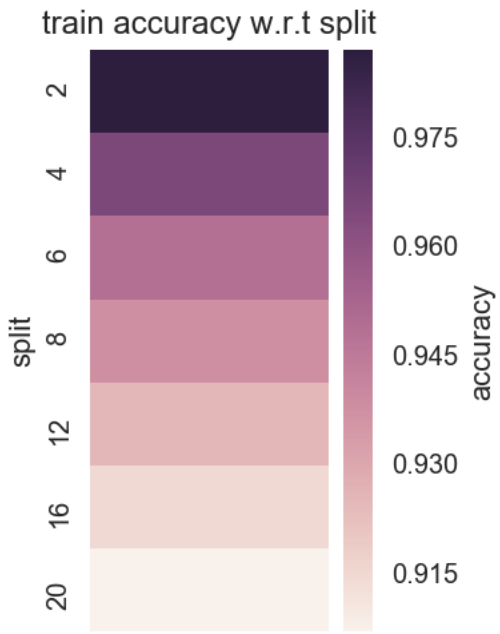
```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0
 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1
 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1]
```

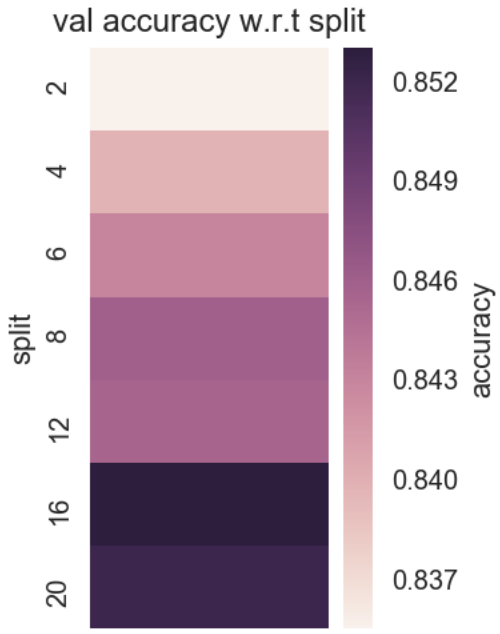
In [6]:
```
classifier = RandomForestClassifier()
split_list = [2,4,6,8,12,16,20]
params = {'min_samples_split':split_list}
grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

In [7]:
```
def draw_heatmap_linear(acc, acc_desc, split_list):
    plt.figure(figsize = (2,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=split_list, xt
    ax.collections[0].colorbar.set_label("accuracy")
    ax.set(ylabel='split')
    plt.title(acc_desc + ' w.r.t split')
    sns.set_style("whitegrid", {'axes.grid' : False})
    plt.show()
```

In [8]:
```
Y = Y.astype('int')
X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
X_test      = X[int(0.8*len(X)):] # Get features from test set.
Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
print(X_train_val.shape, X_test.shape, Y_train_val.shape, Y_test.shape)
grid_search.fit(X_train_val, Y_train_val)

train_acc = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
draw_heatmap_linear(train_acc, 'train accuracy', split_list)

val_acc = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```

```
(4000, 100) (1000, 100) (4000,) (1000,)
```



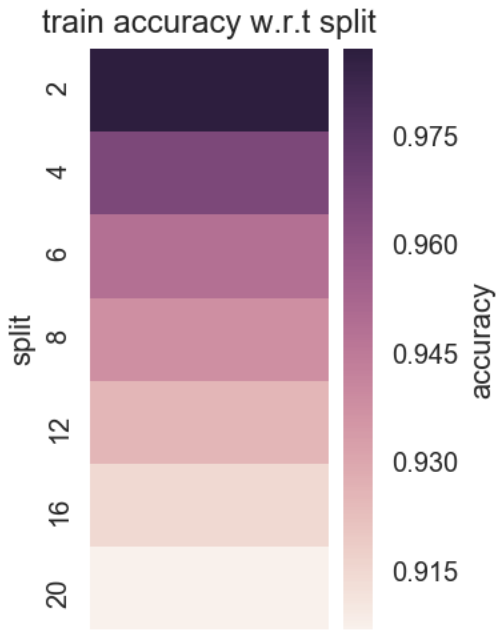


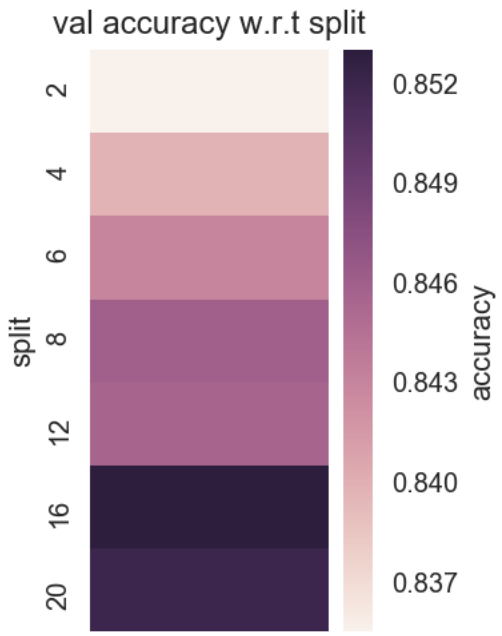

```
In [9]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / len
        print(grid_search.best_params_)
        print(test_acc)
        tot_train = (sum(train_acc)/len(train_acc))
```

```
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        print(tot_val)
```

```
{'min_samples_split': 16}
0.854
[ 0.94078526]
[ 0.84496429]
```

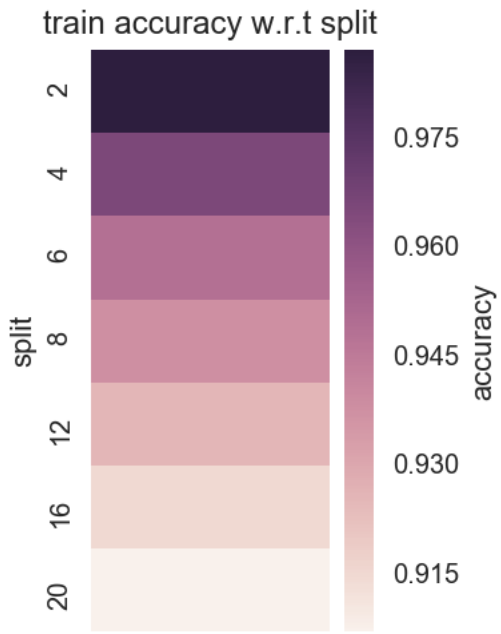
```
In [10]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', split_list)

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```

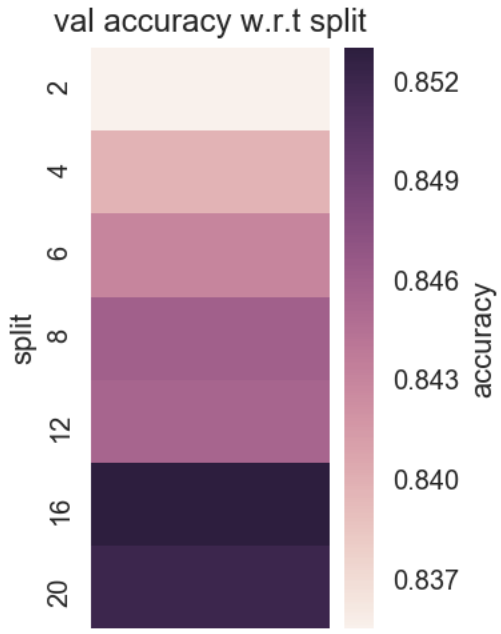





```
In [11]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2)/len(train_acc2))
         print(tot_train2)
         tot_val2 = (sum(val_acc2)/len(val_acc2))
         print(tot_val2)

{'min_samples_split': 8}
0.8444
[ 0.94187119]
[ 0.84205714]
```

```
In [12]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', split_list)

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```






```
In [13]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3)/len(train_acc3))
```

```
        print(tot_train3)
        tot_val3 = (sum(val_acc3)/len(val_acc3))
        print(tot_val3)
```

```
{'min_samples_split': 16}
0.84125
[ 0.94835649]
[ 0.855]
```

```
In [14]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = sum(tot_train + tot_train2 + tot_train3)/3
         avg_val = sum(tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)
```

```
0.84655 0.943670979299 0.84734047619
```

```
In [ ]:
```

RandomForestCOV

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/covtype.data.gz', compres
X_and_Y = df.as_matrix()
X_and_Y = X_and_Y[:5000, :55]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

```
(5000, 55) (5000, 54) (5000,)
```

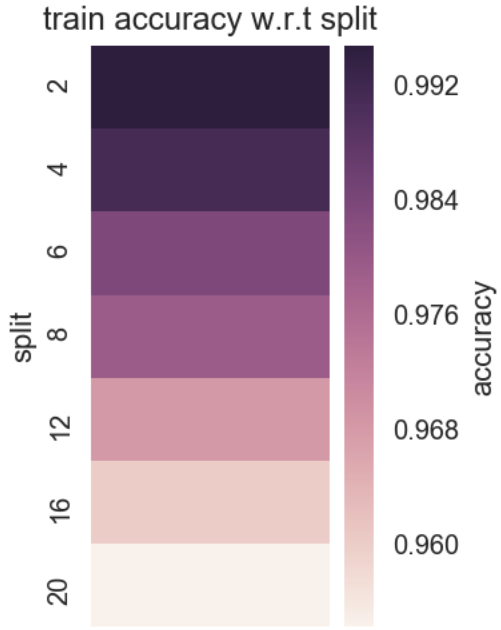
In [3]:
```
#change to binary classification
for i in range(len(Y)):
    if Y[i] == 2:
        Y[i] = 1
    else:
        Y[i] = 0
np.random.shuffle(X_and_Y)
```

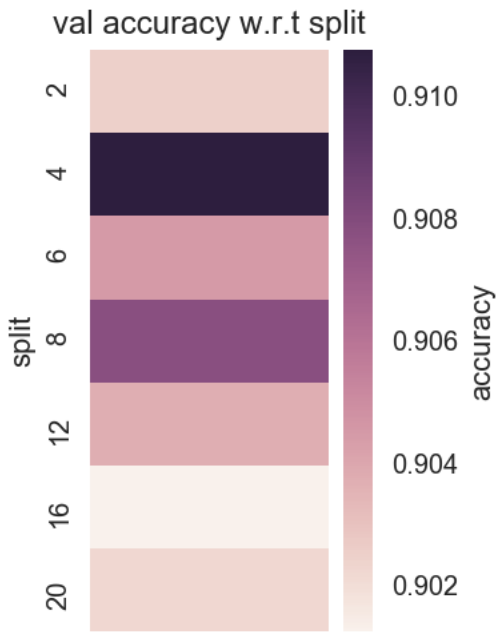
In [4]:
```
classifier = RandomForestClassifier()
split_list = [2,4,6,8,12,16,20]
params = {'min_samples_split':split_list}
grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

In [5]:
```
def draw_heatmap_linear(acc, acc_desc, split_list):
    plt.figure(figsize = (2,4))
    ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=split_list, xt
    ax.collections[0].colorbar.set_label("accuracy")
    ax.set(ylabel='split')
    plt.title(acc_desc + ' w.r.t split')
    sns.set_style("whitegrid", {'axes.grid' : False})
    plt.show()
```

In [6]:
```
X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
X_test      = X[int(0.8*len(X)):] # Get features from test set.
Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
print(X_train_val.shape, X_test.shape, Y_train_val.shape, Y_test.shape)
grid_search.fit(X_train_val, Y_train_val)

train_acc = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
draw_heatmap_linear(train_acc, 'train accuracy', split_list)

val_acc = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```

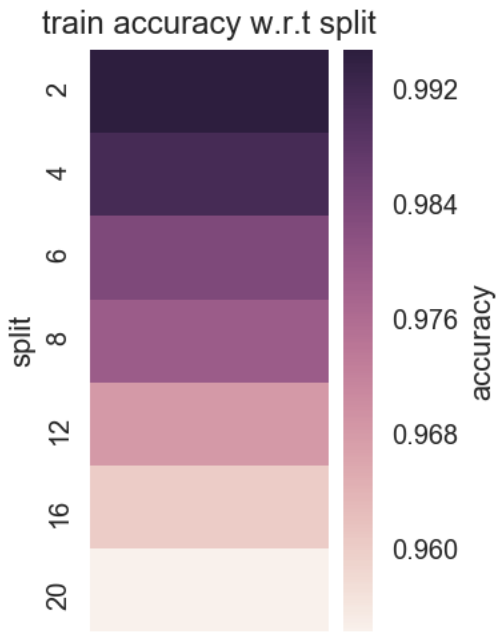
```
(4000, 54) (1000, 54) (4000,) (1000,)
```

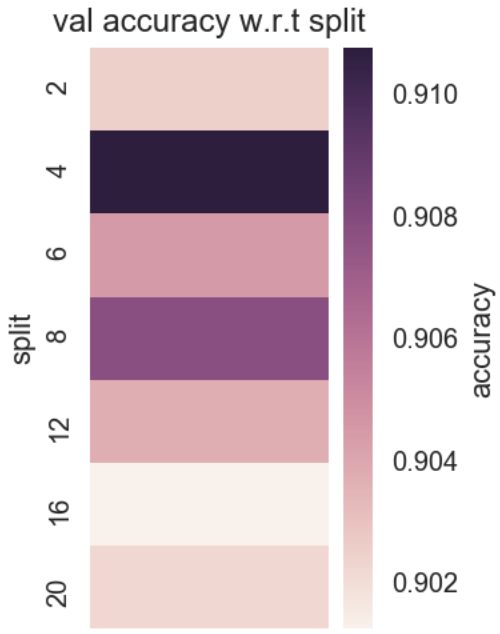





```
In [7]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / len
        print(grid_search.best_params_)
        print(test_acc)
        tot_train = (sum(train_acc)/len(train_acc))
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        print(tot_val)

{'min_samples_split': 4}
0.901
[ 0.97588404]
[ 0.90467857]
```

```
In [8]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
        X_test      = X[int(0.5*len(X)):] # Get features from test set.
        Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
        Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
        grid_search.fit(X_train_val, Y_train_val)
        train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
        draw_heatmap_linear(train_acc, 'train accuracy', split_list)

        val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
        draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```



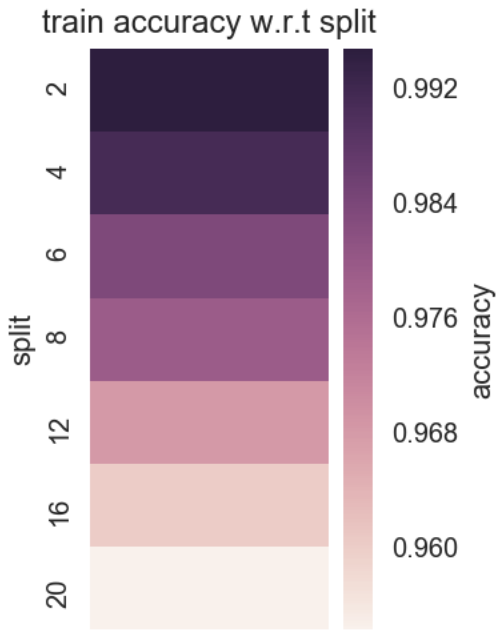


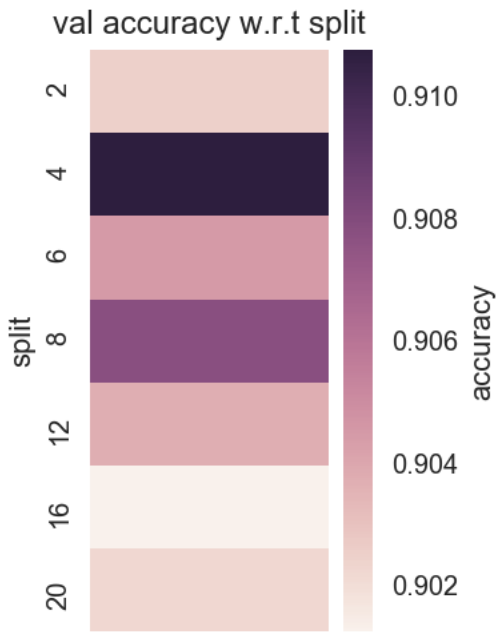

```
In [9]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / le
        print(grid_search.best_params_)
        print(test_acc2)
        tot_train2 = (sum(train_acc2)/len(train_acc2))
```

```
        print(tot_train2)
        tot_val2 = (sum(val_acc2)/len(val_acc2))
        print(tot_val2)
```

```
{'min_samples_split': 2}
0.904
[ 0.97372873]
[ 0.89211429]
```

```
In [10]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', split_list)

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```






```
In [11]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
         tot_train3 = (sum(train_acc3)/len(train_acc3))
         print(tot_train3)
         tot_val3 = (sum(val_acc3)/len(val_acc3))
         print(tot_val3)

{'min_samples_split': 16}
0.8705
[ 0.96846418]
[ 0.87942857]


In [12]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = sum(tot_train + tot_train2 + tot_train3)/3
         avg_val = sum(tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)

0.891833333333 0.972692313174 0.892073809524
```

RandomForestLETTERS

March 24, 2018

In [1]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
df = pd.read_csv('C:/Users/Preston Wong/Downloads/letter-recognition.data',
df2 = df.copy()
df = df.drop([0], axis=1)
df = df.join(df2[0])
```

In [3]:
```
X_and_Y = df.as_matrix()
np.random.shuffle(X_and_Y)
X_and_Y = X_and_Y[:5000, :17]
X = X_and_Y[:, 0:-1]
Y = X_and_Y[:, -1]
print(X_and_Y.shape, X.shape, Y.shape)
```

```
(5000, 17) (5000, 16) (5000,)
```

In [4]:
```
ordlist = []
for i in range(len(Y)):
    ordlist.append(ord(Y[i]))
```

In [5]:
```
for i in range(len(Y)):
    if ((ordlist[i]) >= 65 and (ordlist[i] <= 77)):
        Y[i] = 1
    else:
        Y[i] = 0
```

In [6]:
```
depth_list = []
for i in range(1, 21, 1):
    depth_list.append(i)
print(depth_list)
```

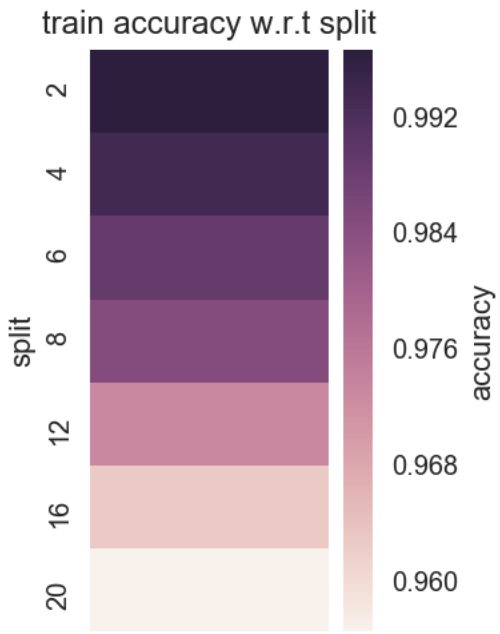
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

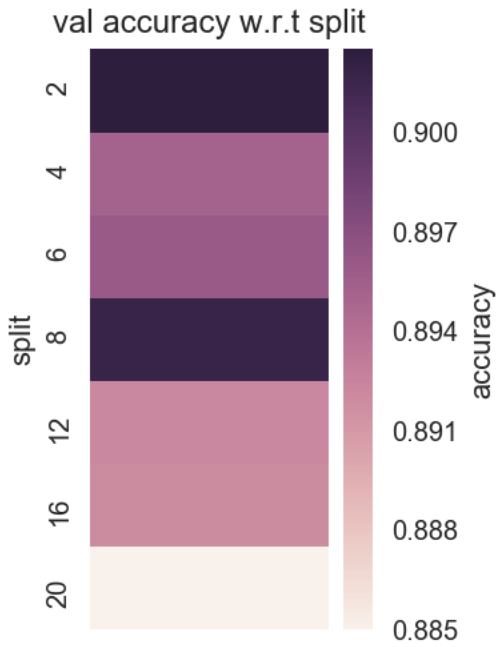
```
In [7]: classifier = RandomForestClassifier()
        split_list = [2,4,6,8,12,16,20]
        params = {'min_samples_split':split_list}
        grid_search = GridSearchCV(classifier, params, return_train_score = True, c
```

```
In [8]: def draw_heatmap_linear(acc, acc_desc, split_list):
            plt.figure(figsize = (2,4))
            ax = sns.heatmap(acc, annot=True, fmt='.3f', yticklabels=split_list, xt
            ax.collections[0].colorbar.set_label("accuracy")
            ax.set(ylabel='split')
            plt.title(acc_desc + ' w.r.t split')
            sns.set_style("whitegrid", {'axes.grid' : False})
            plt.show()
```

```
In [9]: Y = Y.astype('int')
        X_train_val = X[:int(0.8*len(X))] # Get features from train + val set.
        X_test      = X[int(0.8*len(X)):] # Get features from test set.
        Y_train_val = Y[:int(0.8*len(Y))] # Get labels from train + val set.
        Y_test      = Y[int(0.8*len(Y)):] # Get labels from test set.
        print(X_train_val.shape, X_test.shape, Y_train_val.shape, Y_test.shape)
        grid_search.fit(X_train_val, Y_train_val)

        train_acc = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
        draw_heatmap_linear(train_acc, 'train accuracy', split_list)

        val_acc = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
        draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```

```
(4000, 16) (1000, 16) (4000,) (1000,)
```



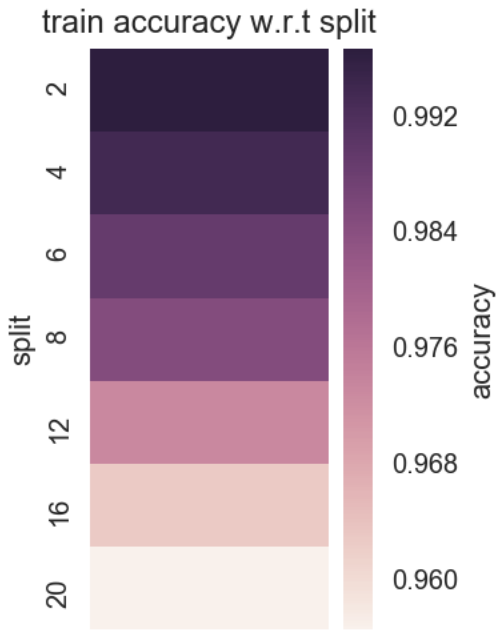


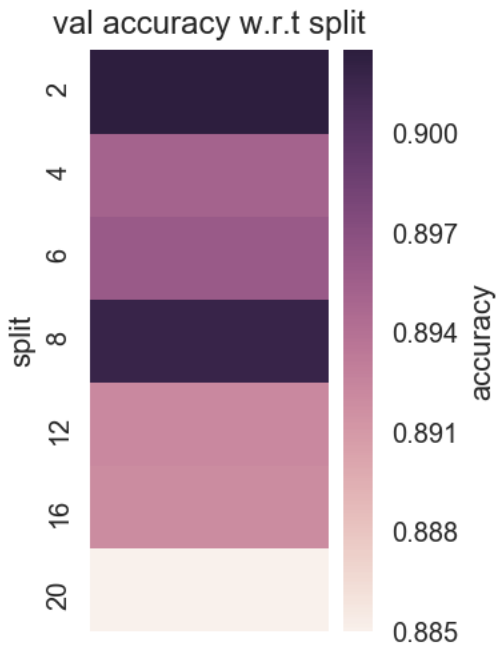

```
In [10]: test_acc = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / le
         print(grid_search.best_params_)
         print(test_acc)
```

```
        tot_train = (sum(train_acc)/len(train_acc))
        print(tot_train)
        tot_val = (sum(val_acc)/len(val_acc))
        print(tot_val)
```

```
{'min_samples_split': 2}
0.919
[ 0.97942867]
[ 0.89496429]
```

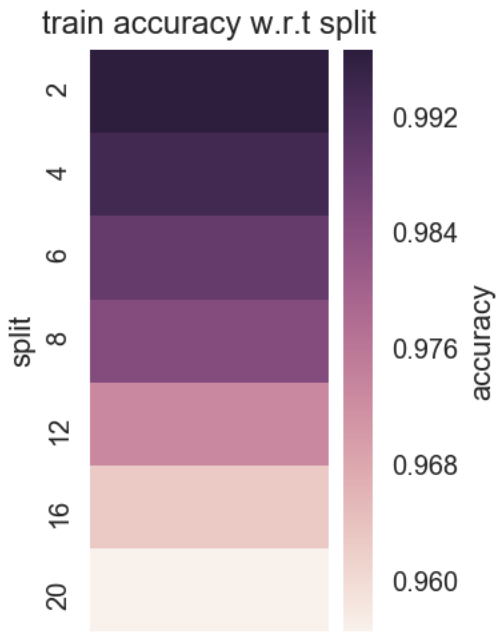
```
In [11]: X_train_val = X[:int(0.5*len(X))] # Get features from train + val set.
         X_test      = X[int(0.5*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.5*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.5*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc2 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', split_list)

         val_acc2 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```

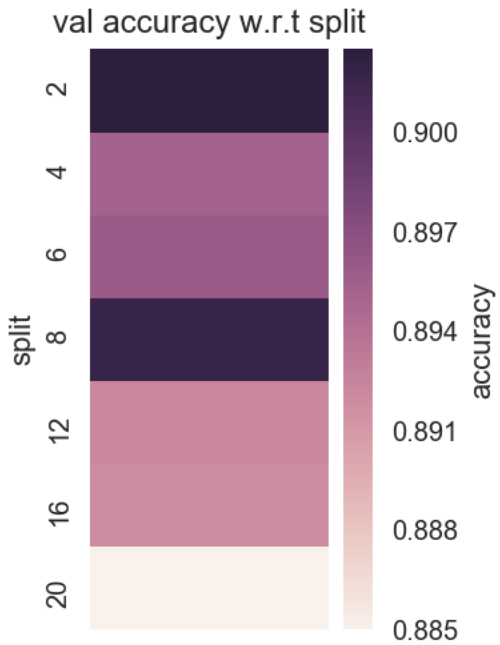





```
In [12]: test_acc2 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc2)
         tot_train2 = (sum(train_acc2)/len(train_acc2))
         print(tot_train2)
         tot_val2 = (sum(val_acc2)/len(val_acc2))
         print(tot_val2)

{'min_samples_split': 2}
0.8924
[ 0.97495714]
[ 0.8812]
```

```
In [13]: X_train_val = X[:int(0.2*len(X))] # Get features from train + val set.
         X_test      = X[int(0.2*len(X)):] # Get features from test set.
         Y_train_val = Y[:int(0.2*len(Y))] # Get labels from train + val set.
         Y_test      = Y[int(0.2*len(Y)):] # Get labels from test set.
         grid_search.fit(X_train_val, Y_train_val)
         train_acc3 = grid_search.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', split_list)

         val_acc3 = grid_search.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', split_list)
```






```
In [14]: test_acc3 = sum(grid_search.best_estimator_.predict(X_test) == Y_test) / l
         print(grid_search.best_params_)
         print(test_acc3)
```

```
        tot_train3 = (sum(train_acc3)/len(train_acc3))
        print(tot_train3)
        tot_val3 = (sum(val_acc3)/len(val_acc3))
        print(tot_val3)
```

```
{'min_samples_split': 6}
0.84825
[ 0.96539286]
[ 0.82657143]
```

```
In [15]: avg_test = (test_acc + test_acc2 + test_acc3)/3
         avg_train = sum(tot_train + tot_train2 + tot_train3)/3
         avg_val = sum(tot_val + tot_val2 + tot_val3)/3
         print(avg_test, avg_train, avg_val)
```

```
0.88655 0.973259556766 0.867578571429
```

```
In [ ]:
```