Assignment 8

June 24, 2018

0.1 Team Members:

0.1.1 Swaroop Bhandary, Vajra Ganeshkumar, Supriya Vadiraj

```
In [12]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.datasets import make_classification
         from sklearn.model_selection import train_test_split
         import pickle
         import time
         from sklearn.metrics import accuracy_score
         import numpy as np
         import matplotlib.pyplot as plt
         import random
```

0.2 Get the MNIST dataset and train random forest (with different number of estimators) on this data

```
In [2]: custom_data_home = '/home/swaroop/Documents/2nd sem/Learning and Adaptivity/Assignment/D
        from sklearn.datasets import fetch_mldata
        mnist = fetch_mldata('MNIST original', data_home=custom_data_home)
```

```
In [3]: x_train, x_test, y_train, y_test = train_test_split(mnist.data, mnist.target, test_size=
```

```
In [82]: random_forest_list = list()
         for i in range(5,100,5):
             clf_1 = RandomForestClassifier(n_estimators=i, random_state=0)
             clf_1.fit(x_train, y_train)
             random_forest_list.append(clf_1)
         file_name = 'random_forest.pickle'
```

```
In [41]: pickle.dump(random_forest_list, open(file_name, 'wb'))
```

```
In [4]: file_name = 'random_forest.pickle'
        random_forest = open(file_name, 'rb')
        forest_list = pickle.load(random_forest)
```

```
In [5]: accuracy_list = list()
        for value in forest_list:
```

```
            pred = value.predict(x_test)
            accuracy = accuracy_score(pred, y_test)
            accuracy_list.append(accuracy)
            print "No of trees: ",value.n_estimators, " Accuracy: ", accuracy
```

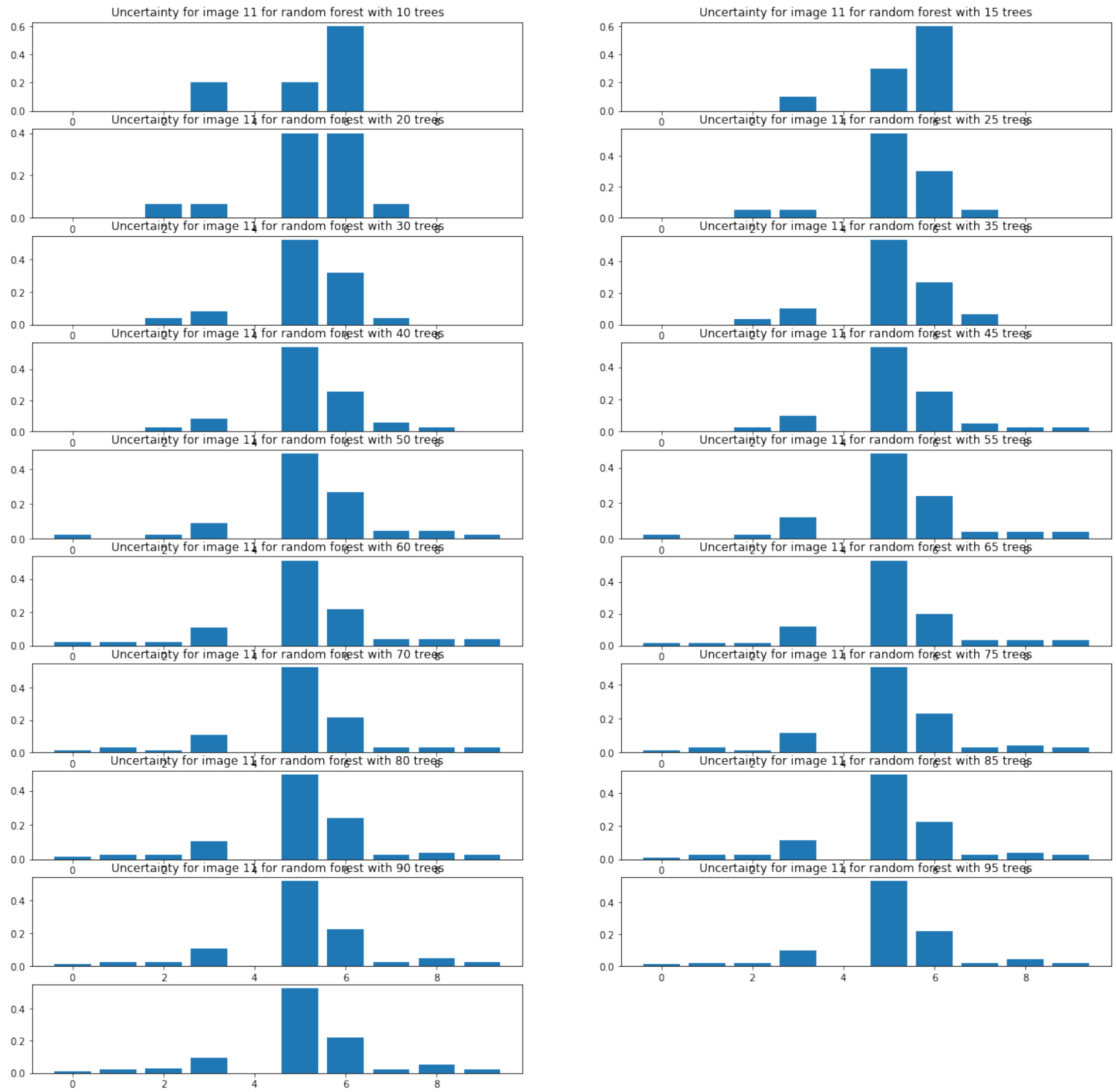
```
No of trees:  5  Accuracy:  0.9202857142857143
No of trees:  10  Accuracy:  0.9475714285714286
No of trees:  15  Accuracy:  0.9567857142857142
No of trees:  20  Accuracy:  0.9603571428571429
No of trees:  25  Accuracy:  0.9633571428571429
No of trees:  30  Accuracy:  0.9644285714285714
No of trees:  35  Accuracy:  0.9657857142857142
No of trees:  40  Accuracy:  0.9662142857142857
No of trees:  45  Accuracy:  0.9671428571428572
No of trees:  50  Accuracy:  0.9672142857142857
No of trees:  55  Accuracy:  0.968
No of trees:  60  Accuracy:  0.9680714285714286
No of trees:  65  Accuracy:  0.9677857142857142
No of trees:  70  Accuracy:  0.9677142857142857
No of trees:  75  Accuracy:  0.9685
No of trees:  80  Accuracy:  0.9688571428571429
No of trees:  85  Accuracy:  0.9685
No of trees:  90  Accuracy:  0.9685
No of trees:  95  Accuracy:  0.9687857142857143
```

0.2.1 Uncertainty estimation

```
In [83]: labels = np.linspace(0,9,10)

         # this image was chosen since the forest with lesser number of trees classified it inco
         # but forests with more trees were classifying it correctly.
         image_index_to_test = [11]
         f = plt.figure(figsize=(20,20))

         print "Actual value:", y_test[11]

         for index,value in enumerate(forest_list):
             y_pred = value.predict(x_test[image_index_to_test])
             print "Label predicted by forest with ", value.n_estimators, " trees",  y_pred
             misclassified_index = [i for i in range(len(y_pred)) if (y_pred[i] != y_test[i]).an
             uncertainty_forest = np.squeeze(value.predict_proba(x_test[image_index_to_test]))
             ax.set_title("Uncertainty for image 11 for random forest with "+str(value.n_estimat
             ax = f.add_subplot(10,2,index+1)
             ax.bar(labels, uncertainty_forest)
```

```
Actual value: 5.0
Label predicted by forest with  5  trees [6.]
```

```
Label predicted by forest with  10  trees [6.]
Label predicted by forest with  15  trees [5.]
Label predicted by forest with  20  trees [5.]
Label predicted by forest with  25  trees [5.]
Label predicted by forest with  30  trees [5.]
Label predicted by forest with  35  trees [5.]
Label predicted by forest with  40  trees [5.]
Label predicted by forest with  45  trees [5.]
Label predicted by forest with  50  trees [5.]
Label predicted by forest with  55  trees [5.]
Label predicted by forest with  60  trees [5.]
Label predicted by forest with  65  trees [5.]
Label predicted by forest with  70  trees [5.]
Label predicted by forest with  75  trees [5.]
Label predicted by forest with  80  trees [5.]
Label predicted by forest with  85  trees [5.]
Label predicted by forest with  90  trees [5.]
Label predicted by forest with  95  trees [5.]
```



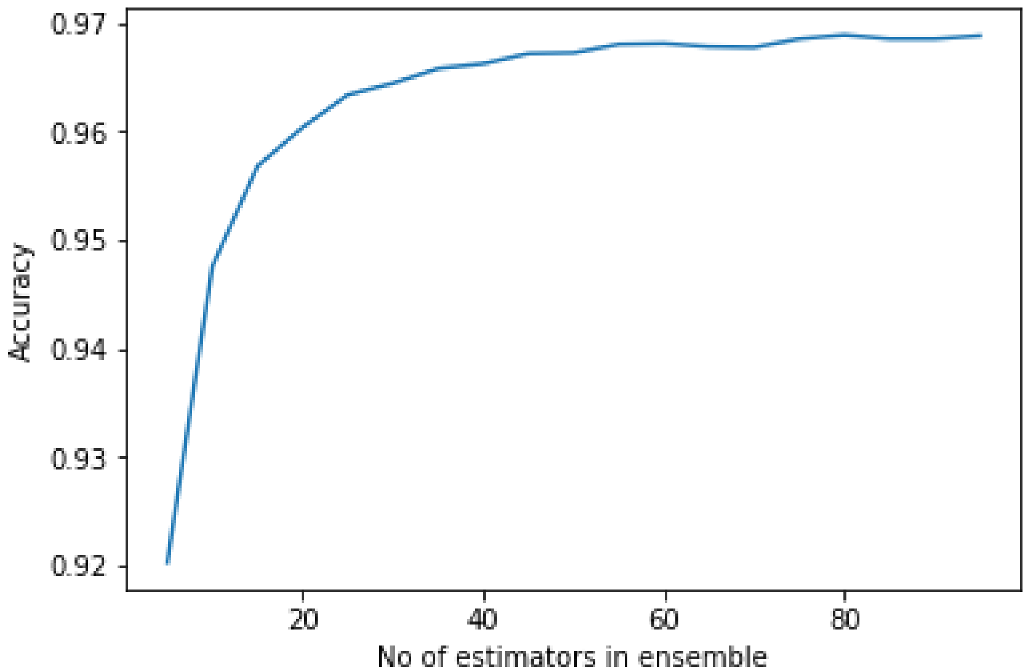
We can see that the forest with lesser number of trees is predicting the wrong value with high confidence. Forest with 5 trees concludes the number is a 6 with 60% certainty.

As the number of the trees increases we can see that the forest is capable of giving better uncertainty estimation as there are more number of trees to decide which label to choose. Also, since random forest use pooling to decide which label to predict as the final output, the more trees the better the prediction is and also better the estimate of uncertainty is.

0.3 3) Does accuracy and uncertainty improve by having more members in each ensemble

```
In [57]: plt.plot(range(5,100,5), accuracy_list)
         plt.xlabel('No of estimators in ensemble')
```

```
plt.ylabel('Accuracy')
plt.show()
```



As per the plot we can see that the accuracy improves with the increase in number of estimators in the ensemble. Also, we get a better estimate of the uncertainty with more number of estimators in the ensemble. If we have just two estimators in the ensemble then we get the output with just 0%, 50% or 100% accuracy values and as the number of estimators increases we get it with more different values. For ex: If number of estimators in 10, the output certainty will vary from 0%., 10%, 20%, 30%, ... 100%.

0.4 4) Using a single ensemble of your choosing (you define the number of members), find the misclassified examples in the test set and analyze the uncertainty of those examples. Can the uncertainty explain why those examples are misclassified? Give examples and a complete analysis.

```
In [6]: clf = RandomForestClassifier(n_estimators=25, random_state=0)
        clf.fit(x_train, y_train)
        y_pred = clf.predict(x_test)
        misclassified_index = [i for i in range(len(y_pred)) if (y_pred[i] != y_test[i]).any()]
```

```
In [11]: uncertainty = clf.predict_proba(x_test)
         misclassified_uncertainty = uncertainty[misclassified_index]
         print "Uncertainty for image 18: ", misclassified_uncertainty[1]
```

```
Uncertainty for image 18:  [0.   0.   0.   0.36 0.   0.4  0.   0.   0.04 0.2 ]
```

We can see that for the misclassified images, the model is very uncertain. This is depicted by the spread of the probability across various labels. For example, if we look at the first misclassified image (Image 18 in test set), we can see that the model is not sure if it is a 2 or a 7. This is mainly because some 2s look very similar to 7s and can sometimes even fool humans.

The model thinks that the image is a 2 with 0.36 certainty which is very slow. But since it is the greatest probability value this has been chosen as the final label.

```
In [87]: print "predicted label: ",y_pred[misclassified_index[0]]
         print "actual label:", y_test[misclassified_index[0]]
         print "Image id:", misclassified_index[0]
         print "Probability value for all label", misclassified_uncertainty[0]
```

```
predicted label:  2.0
actual label: 7.0
Image id: 18
Probability value for all label [0.   0.   0.36 0.   0.08 0.   0.   0.32 0.08 0.16]
```

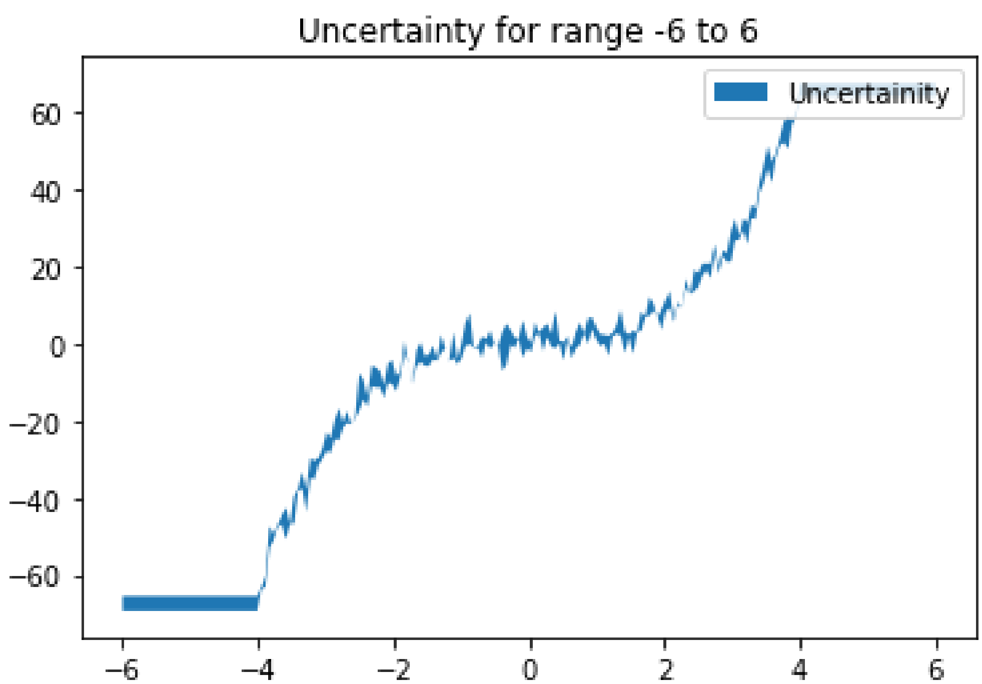
0.5 5) Reproduce Figure one from the "Simple and Scalable Predictive UncertaintyEstimation using Deep Ensembles" paper (attached) using a random forest

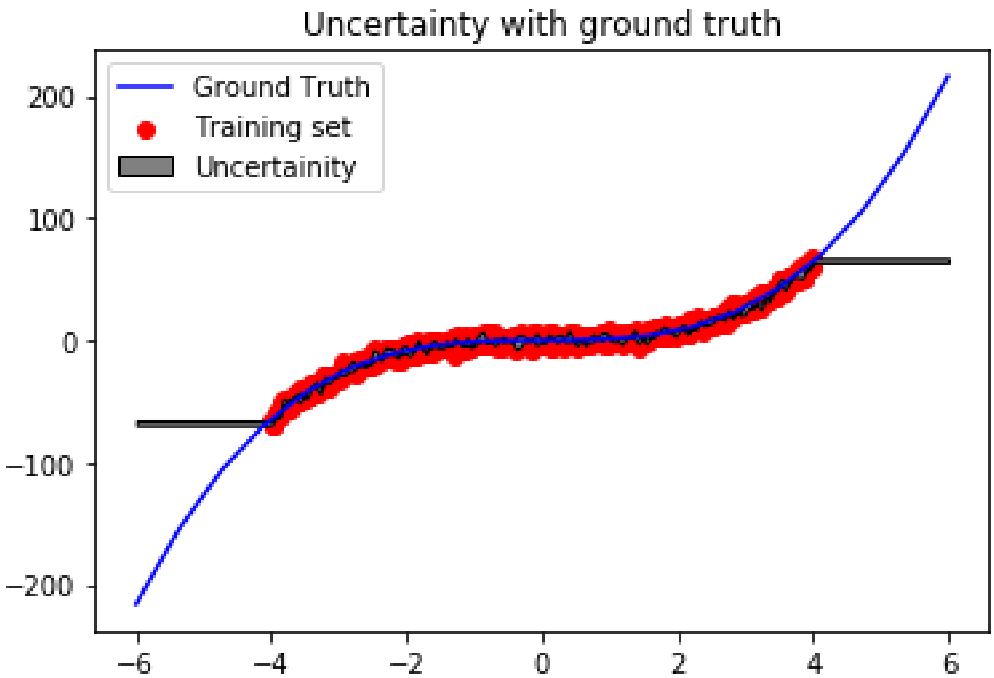
```
In [86]: x_trainingset = np.linspace(-4.0,4.0,1000)
         # print x_trainingset.shape
         noise = np.random.normal(loc=0.0,scale = 3, size = len(x_trainingset))
         y_trainingset = x_trainingset**3 + noise
         x_trainingset = x_trainingset.reshape((-1,1))
         # print y_trainingset.shape
         clf = RandomForestRegressor(n_estimators=5)
         clf.fit(x_trainingset,y_trainingset)
```

```
Out[86]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
In [98]: x_test = np.linspace(-6,6,200)
         noise = np.random.normal(loc=0.0,scale = 3, size = len(x_test))
         x_test = np.reshape(x_test,(-1,1))
         Predictions = []
         Mean = []
         Variance =[]
         for i in range(len(clf.estimators_)):
             predicts = clf.estimators_[i].predict(x_test)
             Predictions.append(predicts)

         mean = np.mean(Predictions, axis =0)
```

```
var = np.var(Predictions, axis =0)

x_test = np.squeeze(x_test)
plt.fill_between(x_test, mean+np.sqrt(var), mean-np.sqrt(var),label = 'Uncertainity')
plt.legend()
plt.title('Uncertainty for range -6 to 6')
plt.show()

#plot with ground truth
X = np.linspace(-6.0,6.0,20)
Y = X**3

plt.plot(X,Y,'blue',label = 'Ground Truth')
plt.scatter(x_trainingset,y_trainingset,color = 'red', label ='Training set')
plt.fill_between(x_test, mean+np.sqrt(var), mean-np.sqrt(var),facecolor='grey',
                 edgecolor = 'black',label ='Uncertainity')
plt.legend()
plt.title("Uncertainty with ground truth")
plt.show()
```





From the plot we can see that the uncetainty is greater between the values from -6 to -4 and from 4 to 6.

Reference Lakshminarayanan, Balaji et al. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." NIPS (2017).