Homework 3
CS 6375: Machine Learning

Vibhav Gogate

Due: 11:59 PM, November, 12 via E-learning

1 Tree Classifiers (75 points)

In this question, you will evaluate tree-based classifiers and their ensembles that we discussed in class. You should use `scikit-learn` and perform the following experiments.

- Download the 15 datasets available on MS Teams. Each data set is divided into three sub-sets: the *training set*, the *validation set* and the *test set*. Data sets are in CSV format. Each line is a training (or test) example that contains a list of attribute values separated by a comma. The last attribute is the class-variable. Assume that all attributes take values from the domain {0,1}.

  The datasets are generated synthetically by randomly sampling solutions and non-solutions (with solutions having class "1" and non-solutions having class "0") from a Boolean formula in conjunctive normal form (CNF). I randomly generated five formulas having 500 variables and 300, 500, 1000, 1500 and 1800 clauses (where the length of each clause equals 3) respectively and sampled 100, 1000 and 5000 positive and negative examples from each formula. I am using the following naming convention for the files. Filenames $train*$, $test*$ and $valid*$ denote the training, test and validation data respectively. $train\_c[i]\_d[j].csv$ where i and j are integers contains training data having j examples generated from the formula having i clauses. For example, the file with filename $train\_c500\_d100$ contains 100 examples generated from the formula having 500 clauses. **Note:** Do not mix and match the datasets. For example, do not train using $train\_c500\_d100.csv$ and test using $test\_c500\_d5000.csv$.

1. (15 points) Use the `sklearn.tree.DecisionTreeClassifier`

   on the 15 datasets. Use the validation set to tune the parameters (see the documentation for parameters; e.g., criterion, splitter, max_depth, etc.). After tuning the parameters, mix the training and validation sets, relearn the decision tree using the "best parameter settings found via tuning" and report the accuracy and F1 score on the test set. For each dataset, also report the "best parameter settings found via tuning." (Note that we expect that no two students will have the same "best parameter settings"').

2. (15 points) Repeat the experiment described above using:

   `sklearn.ensemble.BaggingClassifier` with "DecisionTreeClassifier" as the base estimator.

   Again, use the validation set to tune the parameters, mix training and validation after tuning to learn a new classifier and report (a) Best parameter settings after tuning and (b) Classification accuracy and F1 score.

3. (15 points) Repeat the experiment described above using:

   `sklearn.ensemble.RandomForestClassifier`.

4. (15 points) Repeat the experiment described above using:

   `sklearn.ensemble.GradientBoostingClassifier`.

5. (15 points) Record the classification accuracy and F1 score for each dataset and classifier (recall that we have four classifiers and we are using the best/tuned parameter settings for the classifier) in a table and then answer the following questions using the table:

   - Which classifier (among the four) yields the best overall generalization accuracy/F1 score? Based on your ML knowledge, why do you think the "classifier" achieved the highest overall accuracy/F1 score.
   - What is the impact of increasing the amount of training data on the accuracy/F1 scores of each of the four classifiers.
   - What is the impact of increasing the number of features on the accuracy/F1 scores of each of the four classifiers.

6. (Extra credit, 5 points) Evaluate the four tree classifiers you used above on the MNIST dataset from Homework 2 (do not compute F1 scores on MNIST, just classification accuracy). Which classifier among the four yields the best generalization accuracy on the MNIST dataset and why?

What to Turn in for Part 1

- Your code which demonstrates how you set up the experiments and a README file for using/running the code. Your results should be repeatable.
- Your report describing your results and answers to the questions

2 K-means clustering on images [25 points]

You can use either Java or Python to implement this part.

In this problem, you will use K-means clustering for image compression. We have provided you with two images.

- Display the images after data compression using K-means clustering for different values of K (2, 5, 10, 15, 20).
- What are the compression ratios for different values of K? Note that you have to repeat the experiment multiple times with different initializations and report the average as well as variance in the compression ratio.
- Is there a tradeoff between image quality and degree of compression. What would be a good value of K for each of the two images?

We have provided you a java template "KMeans.java" which implements various image input/output operations. You have to implement the function kmeans in the template. If you want to use python, you will have to replicate the code in KMeans.java in python (will take roughly 10-15 minutes). Note that you cannot use scikit learn implementation of k-means for this part.

What to Turn in for Part 2

In a single zip file:

- Your source code for the kmeans algorithm.
- A PDF report containing your write up.

Note that your program must compile and we should be able to replicate your results. Otherwise no credit will be given.