CS 3430: SciComp with Py
Assignment 12

Vladimir Kulyukin
Department of Computer Science
Utah State University

April 14, 2018

1 Introduction

You may have heard that many honeybee colonies are failing. In the U.S., the number of hives that do not survive the winter months has averaged 28.7% since 2006-2007. During the 2014 California almond bloom, between 15% and 25% of beehives experienced failures ranging from complete hive collapse to dead and deformed brood. Typical culprits of bee colony failures are bacteria and parasites (e.g., Varroa mites, foulbrood, chalkbrood) and bad weather. Other factors contributing to bee colony failures include pesticides (e.g., neonicotinoids), monoculture (increasing emphasis on growing large quantities of the same crop), and transportation stress caused by long hauls of large quantities of commercial beehives on semitrucks. Since 1 out of every 3 crops have the honeybee as their sole pollinator, the nutritional diversity of our food supply is in danger.

It's not only the bees that are suffering, beekeepers are suffering, too. The average loss for beekeepers annually is 1 out of every 3 hives. The loss range varies from 25% to 50% per year. Nationally we've lost more than half of our beekeepers since the arrival of the Varroa mite, a parasite that now afflicts many beehives. The cost of equipment, bee packages, maintenance, and transportation is so high that profit margins for beekeepers are very small.

I believe that in the future significant practical and scientific benefits will come from transforming our bee yards into smart worlds. Think of a beehive as an immobile robot that monitors the health of the bee colony inside, analyzes the data from its sensors, and alerts the beekeeper of any deviations from the norm through the Internet of Things (IoT).

Why is this useful? It's useful, because human beekeepers cannot monitor their hives continuously due to obvious problems with logistics and fatigue. It will also reduce the number of invasive hive inspections that disrupt the life cycle of bee colonies and put stress on the bees. The transportation costs will likely be lower, because many beekeepers now drive long distances to their far flung bee yards.

In this assignment, we'll use decision trees and random forests to classify audio data obtained from a deployed electronic beehive monitor. I included three sample audio files: `bee.wav`, `cricket.wav`, `noise.wav`. You can quickly listen to them to get an idea of what the original audio data is like. However, we won't work with the raw data, because audio signal processing is beyond the scope of this class. We'll work with the processed data instead, i.e., feature vectors. The zip archive for this assignment contains three directories: `beefv`, `cricketfv`, and `noisefv`. These directories contain feature vectors obtained from wav audio files with bee buzzing, cricket chirping, and ambient noise, respectively.

Each audio file is turned into a feature vector by applying to it various audio filters (e.g., FFT, SFFT, MFCC, etc.) to extract 193 features from it. Each feature is a float. After the features are extracted, each feature vector is turned into a numpy array and persisted. Thus, the directory `beefv` contains 3,000 persisted numpy arrays with bee buzzing features; the directory `cricketfv` contains 3,000 persisted numpy arrays of cricket chirping features; the directory `noisefv` contains 3,110 persisted numpy arrays with ambient noise features.

This ground truth classification was obtained from four human evaluators (three of my graduate students and myself) who manually labeled 9,110 2-second audio samples from May and June 2017 by listening to each sample and placing it into one of the three non-overlapping categories: bee buzzing, cricket chirping, and ambient noise. The bee buzzing category consisted of the samples where the evaluators could hear the buzzing of bees. The cricket churping category consisted of the files, captured at night, where the evaluators could hear the chirping of crickets. The noise category included all samples where the evaluators could not clearly hear either bee buzzing or cricket churping. These were the samples with sounds of static microphone noise, thunder, wind, rain, auto vehicles, human conversation, sprinklers, and relative silence, i.e., absence of any sounds discernable to a human ear.

2 Loading Audio Data

Let's see how we can read the data in. The file `random_audio_forests.py` with the starter code has the function `read_data_audio` that takes the base directory where `beefv`, `cricketfv`, and `noisefv` are placed and returns two numpy arrays: `data` and `target`.

```
>>> base_dir = '/home/vladimir/audio_data/'
>>> data, target = read_audio_data(base_dir)
>>> data.shape
(9110, 193)
>>> target.shape
(9110,)
```

The `data` array is a 9110 x 193 numpy matrix where each row is a 193-element audio feature vector. The `target` array is a 1D numpy array with 9110 labels. The first 3,000 elements of this array are 0's, the second 3,000 elements are 1's, and the rest are 2's, where 0 denotes `bee buzzing`, 1 - `cricket chirping`, 2 - `ambient noise`.

3 Growing Decision Trees

Let's see how to classify the processed audio data with decision trees. Implement the function `train_test_split_eval_dtr()`.

```
def train_test_split_eval_dtr(dtr, data, target, test_size=0.4):
    pass
```

This function takes a decision tree and the data and target arrays, uses `train_test_split` to split the data and target arrays into the train and test data and train and test target arrays using the value of the keyword parameter `test_size`, fits the decision tree to the train data and target, predicts the targets of the test data, computes the decision tree's accuracy (DTR accuracy), the classification report, and the confusion matrix, and prints all of them. Here is a test run.

```
>>> from sklearn import tree
>>> data, target = read_audio_data(base_dir)
>>> dtr = tree.DecisionTreeClassifier(random_state=random.randint(0, 100))
>>> train_test_split_eval_dtr(dtr, data, target)

DTR accuracy: 0.995334796926

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=47, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      0.99      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1171    1    6]
 [   0 1198    0]
 [  10    0 1258]]
```

You may see and safely ignore a deprecation warning about `rank` in `numpy.linalg.matrix`.

4 Growing Random Forests

A random forest is a set of decision trees. Let's generalize `train_test_split_eval_dtr` to random forests and implement the function `train_test_split_eval_rf()`.

```
def train_test_split_eval_rf(rf, data, target, test_size=0.4):
    pass
```

This function takes a random forest `rf` and the `data`, `target` arrays, and the percent of of the data that should be used for testing specified by `test_size`. The default value of `test_size` is 0.4, i.e., 40%. This function

1. uses `train_test_split` to split the `data` and `target` arrays into the train and test data using the value specified by `test_size`;

2. fits the random forest to the train data and train target arrays;
3. predicts the targets of the test data;
4. computes the random forest's accuracy (RF accuracy) and the confusion matrix;
5. prints the number of trees in the random forest, the random forest's accuracy, the classification report, and the confusion matrix.

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> data, target = read_audio_data('/home/vladimir/audio_data/')
>>> rf = RandomForestClassifier(n_estimators=10, random_state=random.randint(0, 1000))
>>> train_test_split_eval_rf(rf, data, target)
# of trees in random forest: 10
RF accuracy: 0.997804610318

Classification report for decision tree RandomForestClassifier(bootstrap=True, class_weight=None,
            criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=1, oob_score=False, random_state=476,
            verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       0.99      1.00      1.00      1178
          1       1.00      1.00      1.00      1198
          2       1.00      1.00      1.00      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1176    0    2]
 [   0 1198    0]
 [   6    0 1262]]
```

5 Testing Trees and Forests

Are there any differences between decision trees and random forests on this data set? Let's further generalize testing trees and random forests. Implement the function `train_test_split_dtr_range_eval()`.

```
def train_test_split_dtr_range_eval(n, data, target):
    pass
```

This function takes the number of experiments `n` and the `data` and `target` arrays, creates `n` decision trees and calls each `train_test_split_eval_dtr`, defined in Section 3, on each decision tree and the `data` and `target` arrays. Here is a test run.

```
>>> from sklearn import tree
>>> data, target = read_audio_data(base_dir)
>>> train_test_split_dtr_range_eval(5, data, target)
Starting with train_test_split procedure
DTR accuracy: 0.995334796926

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=47, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      0.99      1268
```

```
avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1171    1    6]
 [   0 1198    0]
 [  10    0 1258]]
Starting with train_test_split procedure
DTR accuracy: 0.995883644347

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=50, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      0.99      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1172    1    5]
 [   0 1198    0]
 [   9    0 1259]]
Starting with train_test_split procedure
DTR accuracy: 0.995060373216

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=72, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      0.99      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1172    1    5]
 [   0 1198    0]
 [  11    1 1256]]
Starting with train_test_split procedure
DTR accuracy: 0.995609220637

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=25, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      0.99      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1172    1    5]
 [   0 1198    0]
 [   8    2 1258]]
Starting with train_test_split procedure
DTR accuracy: 0.993962678375

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=3, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      0.99      0.99      1178
          1       0.99      1.00      1.00      1198
          2       1.00      0.99      0.99      1268

avg / total       0.99      0.99      0.99      3644


Confusion matrix:
[[1171    1    6]
 [   0 1198    0]
 [   8    7 1253]]
Starting with train_test_split procedure
DTR accuracy: 0.996432491767

Classification report for decision tree DecisionTreeClassifier(class_weight=None,
            criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=83, splitter='best'):
             precision    recall  f1-score   support

          0       0.99      1.00      0.99      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      1.00      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1174    1    3]
 [   0 1198    0]
 [   9    0 1259]]
```

Finally, let's implement the function `train_test_split_rf_range_eval`.

```
def train_test_split_rf_range_eval(nt_lower_bound, nt_upper_bound, data, target):
    pass
```

This function generates random forests with the number of trees in `[nt_lower_bound, nt_upper_bound]` in increments of 10. For example, `[10, 50]`, will create random forests with 10 trees, 20 trees, 30 trees, 40 trees, and 50 trees. For each random forest `rf`, this function calls `train_test_split_eval_rf` on `rf` and the data and target arrays.

```
>>> train_test_split_rf_range_eval(10, 50, data, target)
```

```
Starting with train_test_split procedure
# of trees in random forest: 10
RF accuracy: 0.998353457739

Classification report for decision tree RandomForestClassifier(bootstrap=True,
            class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=1, oob_score=False, random_state=780,
            verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       1.00      1.00      1.00      1178
          1       1.00      1.00      1.00      1198
          2       1.00      1.00      1.00      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1174    0    4]
 [   0 1198    0]
 [   2    0 1266]]
Starting with train_test_split procedure
# of trees in random forest: 20
RF accuracy: 0.997255762898

Classification report for decision tree RandomForestClassifier(bootstrap=True,
            class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=20, n_jobs=1, oob_score=False, random_state=448,
            verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       0.99      1.00      1.00      1178
          1       1.00      1.00      1.00      1198
          2       1.00      0.99      1.00      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1177    0    1]
 [   0 1198    0]
 [   9    0 1259]]
Starting with train_test_split procedure
# of trees in random forest: 30
RF accuracy: 0.997530186608

Classification report for decision tree RandomForestClassifier(bootstrap=True,
            class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=30, n_jobs=1, oob_score=False, random_state=849,
            verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       0.99      1.00      1.00      1178
          1       1.00      1.00      1.00      1198
          2       1.00      1.00      1.00      1268
```

```
avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1175    0    3]
 [   0 1198    0]
 [   6    0 1262]]
Starting with train_test_split procedure
# of trees in random forest: 40
RF accuracy: 0.997804610318

Classification report for decision tree RandomForestClassifier(bootstrap=True,
            class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=40, n_jobs=1, oob_score=False, random_state=569,
            verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       1.00      1.00      1.00      1178
          1       1.00      1.00      1.00      1198
          2       1.00      1.00      1.00      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1175    0    3]
 [   0 1198    0]
 [   5    0 1263]]
Starting with train_test_split procedure
# of trees in random forest: 50
RF accuracy: 0.998627881449

Classification report for decision tree RandomForestClassifier(bootstrap=True,
            class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=50, n_jobs=1, oob_score=False, random_state=609,
            verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       1.00      1.00      1.00      1178
          1       1.00      1.00      1.00      1198
          2       1.00      1.00      1.00      1268

avg / total       1.00      1.00      1.00      3644


Confusion matrix:
[[1177    0    1]
 [   0 1198    0]
 [   4    0 1264]]
```

What To Submit

Implement these functions in `random_audio_forests.py` and submit this file via Canvas. In a comment at the head of the file, briefly answer the following question: Do you think that random forests buy us better performance on this data set?

References

1. V. Kulyukin & S. Mukherjee. "Computer Vision in Low-Power Electronic Beehive Monitoring: In Situ Vision-Based Bee Counting on Langstroth Hive Landing Pads." *Graphics, Vision, and Image Processing* (GVIP), vol. 17, issue 1, pp. 25 - 37, June 2017, ISSN: 1687-398X.

2. V. Kulyukin & S. Reka. "Toward Sustainable Electronic Beehive Monitoring: Algorithms for Omnidirectional Bee Counting from Images & Harmonic Analysis of Buzzing Signals." *Engineering Letters*, vol. 24, no. 3, pp. 317-327, Aug. 2016.