

03/23/18 02:15:25

/Users/caiglencross/Documents/MachineLearning/ps2/ps7/source/digits.py

```

1  """
2  Author      : Cai Glencross & Katie Li
3  Class      : HMC CS 158
4  Date       : 2018 Mar 23
5  Description : Bagging with Digits Dataset
6              This code was adapted from course material by Jenna
7  Wiens (UMich)
8  """
9  # python libraries
10 import collections
11
12 # numpy libraries
13 import numpy as np
14
15 # matplotlib libraries
16 import matplotlib as mpl
17 mpl.use('TkAgg')
18
19 import matplotlib.pyplot as plt
20
21 # scikit-learn libraries
22 from sklearn.datasets import load_digits
23 from sklearn.tree import DecisionTreeClassifier
24 from sklearn.ensemble import BaggingClassifier
25 from sklearn.ensemble import RandomForestClassifier
26 from sklearn.model_selection import train_test_split
27 from sklearn import metrics
28
29
30 #####
31 ##
32 # bagging functions
33 #####
34 ##
35
36 def bagging_ensemble(X_train, y_train, X_test, y_test,
37                     max_features=None, num_clf=10) :
38     """
39     Compute performance of bagging ensemble classifier.
40
41     Parameters
42     -----
43     X_train      -- numpy array of shape (n_train,d), training
44     features
45     y_train      -- numpy array of shape (n_train,), training

```

```

targets
42         X_test          -- numpy array of shape (n_test,d),  test
features
43         y_test          -- numpy array of shape (n_test,),  test
targets
44         max_features -- int, number of features to consider when
looking for best split
45         num_clf       -- int, number of decision tree classifiers in
bagging ensemble
46
47     Returns
48     -----
49         accuracy      -- float, accuracy of bagging ensemble
classifier on test data
50     """
51     base_clf = DecisionTreeClassifier(criterion='entropy',
max_features=max_features)
52     clf = BaggingClassifier(base_clf, n_estimators=num_clf)
53     clf.fit(X_train, y_train)
54     y_pred = clf.predict(X_test)
55     return metrics.accuracy_score(y_test, y_pred)
56
57
58 def random_forest(X_train, y_train, X_test, y_test, max_features,
num_clf=10,
59                  bagging=bagging_ensemble) :
60     """
61     Wrapper around bagging_ensemble to use feature-limited decision
trees.
62
63     Additional Parameters
64     -----
65         bagging      -- bagging_ensemble or bagging_ensemble2
66     """
67     return bagging(X_train, y_train, X_test, y_test,
68                  max_features=max_features, num_clf=num_clf)
69
70
71 def bagging_ensemble2(X_train, y_train, X_test, y_test,
max_features=None, num_clf=10) :
72     """
73     Compute performance of bagging ensemble classifier.
74
75     You are allowed to use DecisionTreeClassifier but NOT
BaggingClassifier.
76
77     Details
78     - Train num_clf base classifiers using bootstrap samples from
X_train and y_train.
79     Use DecisionTreeClassifier with information gain as base

```

```

classifier.
80     Hints: Use np.random.choice(...) for bootstrap samples.
81     Make sure to use same indices from X_train and y_train.
82     - Predict using X_test and y_test.
83     For each base classifier, track predictions on X_test.
84     Make ensemble prediction using using majority vote.
85     - Return accuracy compared to y_test.
86
87     Same parameters and return values as bagging_ensemble(...)
88     """
89
90     n_train, d = X_train.shape
91
92     ### ===== TODO : START ===== ###
93     # extra credit: implement bagging ensemble (see details above)
94
95     return 0.0
96     ### ===== TODO : START ===== ###
97
98
99     #####
100    ##
101    # plotting functions
102    #####
103    ##
104
105    def plot_scores(max_features, bagging_scores, random_forest_scores)
106    :
107        """
108        Plot values in random_forest_scores and bagging_scores.
109        (The scores should use the same set of 100 different train and
110        test set splits.)
111
112        Parameters
113        -----
114            max_features          -- list, number of features considered
115            when looking for best split
116            bagging_scores        -- list, accuracies for bagging
117            ensemble classifier using DTs
118            random_forest_scores -- list, accuracies for random forest
119            classifier
120        """
121
122        plt.figure()
123        plt.plot(max_features, bagging_scores, '--', label='bagging')
124        plt.plot(max_features, random_forest_scores, '--', label='random
125        forest')
126        plt.xlabel('max features considered per split')
127        plt.ylabel('accuracy')
128        plt.legend(loc='upper right')

```

```

121     plt.show()
122
123
124 def plot_histograms(bagging_scores, random_forest_scores):
125     """
126     Plot histograms of values in random_forest_scores and
127     bagging_scores.
128     (The scores should use the same set of 100 different train and
129     test set splits.)
130
131     Parameters
132     -----
133         bagging_scores      -- list, accuracies for bagging
134         ensemble classifier using DTs
135         random_forest_scores -- list, accuracies for random forest
136         classifier
137     """
138
139     bins = np.linspace(0.8, 1.0, 100)
140     plt.figure()
141     plt.hist(bagging_scores, bins, alpha=0.5, label='bagging')
142     plt.hist(random_forest_scores, bins, alpha=0.5, label='random
143     forest')
144     plt.xlabel('accuracy')
145     plt.ylabel('frequency')
146     plt.legend(loc='upper left')
147     plt.show()
148
149 #####
150 ##
151 # main
152 #####
153 ##
154
155 def main():
156     np.random.seed(1234)
157
158     # load digits dataset
159     digits = load_digits(4)
160     X = digits.data
161     y = digits.target
162
163     # evaluation parameters
164     num_trials = 100
165
166     # sklearn or home-grown bagging ensemble
167     bagging = bagging_ensemble
168
169     #=====

```

```

164     # vary number of features
165
166     # calculate accuracy of bagging ensemble and random forest
167     #   for 100 random training and test set splits
168     # make sure to use same splits to enable proper comparison
169     max_features_vector = range(1,65,2)
170     bagging_scores = []
171     random_forest_scores = collections.defaultdict(list)
172     for i in range(num_trials):
173         print i
174         X_train, X_test, y_train, y_test = train_test_split(X, y,
175 test_size=0.2)
176         bagging_scores.append(bagging(X_train, y_train, X_test,
177 y_test))
178         for m in max_features_vector :
179             random_forest_scores[m].append(random_forest(X_train,
180 y_train, X_test, y_test, m,
181 bagging=bagging))
182
183     # analyze how performance of bagging and random forest changes
184     with m
185     bagging_results = []
186     random_forest_results = []
187     for m in max_features_vector :
188         bagging_results.append(np.median(np.array(bagging_scores)))
189     #     print m, np.median(np.array(random_forest_scores[m]))
190
191     random_forest_results.append(np.median(np.array(random_forest_scores
192 [m])))
193
194     plot_scores(max_features_vector, bagging_results,
195 random_forest_results)
196
197     #=====
198     # plot histograms of performances for max_features=8
199     bagging_scores = []
200     random_forest_scores = []
201     for i in range(num_trials) :
202         X_train, X_test, y_train, y_test = train_test_split(X, y,
203 test_size=0.2)
204         bagging_scores.append(bagging(X_train, y_train, X_test,
205 y_test))
206         random_forest_scores.append(random_forest(X_train, y_train,
207 X_test, y_test, 8,
208 bagging=bagging))
209     plot_histograms(bagging_scores, random_forest_scores)
210
211     ### ===== TODO : START ===== ###
212     # part d: determine pixel importance
213
214

```

```
203     print(X)
204
205     randomForest = RandomForestClassifier()
206     randomForest.fit(X, y)
207     importances = randomForest.feature_importances_
208     importances = importances.reshape(digits.images[0].shape)
209
210     # Plot pixel importances
211     plt.matshow(importances, cmap=plt.cm.hot)
212     plt.title("Pixel importances with forests of trees")
213     plt.colorbar()
214     plt.show()
215
216
217
218
219
220     ### ===== TODO : END ===== ###
221
222
223     if __name__ == "__main__":
224         main()
```