Project 8 Report : Ensemble approaches

Sajal Kumar

Implementation details on parameters

My implementation uses all 4 methods : 1) Decision Tree classifier, 2) Ada boost classifier using Decision Tree, 3) Random Forest and 4) Bagging classifier using Decision Tree on their default setting except the following general parameters that could be changed:

- `random_state` : Random seed (set to 1 by default).
- `n_jobs` : Number of parallel threads allowed (set to 4 by default).
- `min_samples_leaf` : The minimum number of samples required to be at a leaf node (set to 10 by default).
- `n_estimators` : The maximum number of estimators for the ensemble methods (set to 10 by default).

In my implementation, the object I created for the Decision Tree classifier (1) was used as the base estimators for both Ada boost and Bagging classifiers for consistency. The parameters of Random Forest classifiers were set similar to that of the Decision Tree classifier.

The above represents the 'standard' setting for all classification methods when no parameter is changed.

Performance evaluation

I computed the run-time, prediction accuracy on training data, prediction accuracy on testing data for evaluating the performance of the 4 methods.

Performance on Digits Data-set

Table 1 shows the runtime (in seconds), predictive accuracy on training data and predictive accuracy on testing data for 'Digits' data-set using the 'standard' configuration of Base Decision Tree classifier (Base DT), Ada boost classifier using Decision Tree (ADAB DT), Random Forest (RF) and Bagging classifier using Decision Tree (BAG DT). The data-set was

Info	Base DT	ADAB DT	RF	BAG DT
runtime	0.01	0.16	0.11	2
pred on train	0.88	1	0.96	0.94
pred on test	0.83	0.94	0.92	0.90

Table 1: Result on 'Digits' data-set with 'standard' configuration of the 4 classification methods.

scaled using the 'StandardScaler' method from sklearn. The data was split into 70% training and 30% testing using stratified partition. Bagging classifier using Decision Tree takes the most time to build. I would've expected Random Forest and Bagging Classifier to have similar run-time because they do not differ too much in the core principle but the Bagging classifier seems to have some additional overheads. Accuracy wise, all ensemble approaches do better than the base Decision Tree classifier which was one of the best performers in project 3 against multiple classifiers. Ada boost performs the best overall.

Performance on Mammographic masses Data-set

Info	Base DT	ADAB DT	RF	BAG DT
runtime	0.002	0.019	0.115	1.94
pred on train	0.85	0.94	0.85	0.84
pred on test	0.82	0.76	0.82	0.83

Table 2: Result on 'Mammographic masses' data-set with 'standard' configuration of the 4 classification methods.

Table 2 shows the runtime (in seconds), predictive accuracy on training data and predictive accuracy on testing data for 'Digits' data-set using the 'standard' configuration of Base Decision Tree classifier (Base DT), Ada boost classifier using Decision Tree (ADAB DT), Random Forest (RF) and Bagging classifier using Decision Tree (BAG DT). The data-set was scaled using the 'StandardScaler' method from sklearn. The missing values were replaced with the feature median. The data was split into 70% training and 30% testing using stratified partition. Similar to 'Digits' Bagging classifier using Decision Tree takes the most time to build. Accuracy wise, this time around, Random Forest and Bagging Classifier are similar to the base Decision Tree classifier. While Ada boost does significantly better on training data, it is clearly over-fitting as its accuracy on testing data is the worst and drops significantly.

A more comprehensive evaluation

Here I present results on 'Mammographic masses' dataset for different values of `n_estimators` and `min_samples_leaf`. I chose 'Mammographic masses' because all classifiers perform ex-

pectedly on the 'Digits' data set without much room for improvement. In 'Mammographic masses', we would like to see the effect `n_estimators` and/or `min_samples_leaf` on Ada boost as it was clearly overfitting the data.

n_estimators is not a strong regularization parameter

Info	n_est = 10 (Og)	n_est = 2	n_est = 5	n_est = 2
runtime	0.02	0.003	0.009	0.04
pred on train	0.94	0.88	0.91	0.94
pred on test	0.76	0.78	0.77	0.77

Table 3: Result on 'Mammographic masses' data-set with different values of `n_estimators` of the Ada boost classifier.

Table 3 shows difference in performance for Ada boost classifier with changing values of `n_estimators` (n_est) on 'Mammographic masses' data-set. I do not present results on other classifier as the change in performance was found to be relatively small. Which is expected as the results on 'standard' configuration show slight difference between a single Decision tree versus multiple Decision trees in Random Forest and Bagging classifier. It can be seen that the performance on testing improves slightly with smaller number of estimators. Interestingly, doubling the `n_estimators` also seems to improve the performance slightly on testing data.

min_samples_leaf is a good regularization parameter and has profound effect on ensemble methods

Table 4 shows difference in performance for all 4 classifiers with changing values of `min_samples_leaf` on 'Mammographic masses' data-set. Unlike, `n_estimators`, other classifiers do change with changing `min_samples_leaf` (m_s_l). The predictive accuracy of Ada boost on testing data suddenly improves by 6% with `min_samples_leaf = 50`. Other classifiers experience drop in predictive accuracy on training data-set. At `min_samples_leaf = 100` Random Forest does better than base Decision tree and Bagging classifier on both training and testing data. Ada-boost remains relatively unchanged. At `min_samples_leaf = 200` something interesting happens with all ensemble classifiers. Random forest experiences a big drop in performance, followed by Bagging classifier. Meanwhile, Decision tree and Ada-boost classifiers experience no significant changes. By design both Random Forest and Bagging classifier train multiple 'weak' decision trees, while increasing the `min_samples_leaf` does not affect a single 'strong' Decision tree such as base Decision tree and even Ada-boost (because it indeed has a 'strong' tree and multiple smaller 'weak' trees). It significantly affects the 'weak' trees that are now under-fitting. What is interesting here is that Bagging classifier suffers a relatively smaller drop than Random Forest. In conclusion `min_samples_leaf` has profound effect on the ensemble classifiers but maybe not much on the base Decision tree classifier.

Info	**m_s_l = 10 (Og)**	**m_s_l = 50**	**m_s_l = 100**	**m_s_l = 200**
ADAB DT runtime	0.02	0.01	0.014	0.014
ADAB DT pred on train	0.94	0.85	0.84	0.84
ADAB DT pred on test	0.76	0.82	0.82	0.81
RF runtime	0.11	0.11	0.11	0.11
RF pred on train	0.85	0.83	0.84	0.73
RF pred on test	0.82	0.83	0.82	0.72
Base DT runtime	0.001	0.0001	0.0009	0.001
Base DT pred on train	0.85	0.83	0.82	0.82
Base DT pred on test	0.82	0.82	0.81	0.81
Bag DT runtime	1.94	1.85	1.85	1.81
Bag DT pred on train	0.85	0.84	0.82	0.76
Bag DT pred on test	0.83	0.83	0.81	0.74

Table 4: Result on 'Mammographic masses' data-set with different values of `min_samples_leaf` of all 4 classifiers.