DSO530 Statistical Learning Methods

Lecture 7b : Bagging, Random Forest(s) and Boosting

Dr. Xin Tong
Department of Data Sciences and Operations
Marshall School of Business
University of Southern California
xint@marshall.usc.edu

Bagging

- Deep and bushy decision trees typically suffer from high variance. And pruning can introduce some bias.
- *Bootstrap aggregation*, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- In this approach we generate B different *bootstrapped* training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$ (not pruned), and average all the predictions:

regression

good

n Sample $\to$ bootstrapped sample $1^* \to \hat{f}^{1*}$

Sample $\to$ Sample $B^* \to \hat{f}^{B*}$

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x) \, .$$

side remark: let $x_1 \dots x_n$ be $N(\mu, \sigma^2)$ & $\bar{x} = \frac{x_1 + \dots + x_n}{n}$

(1) $X_1, \dots, X_n$ indep, $Var(\bar{X}) = 1/n^2 \, Var(X_1 + \dots + X_n) = \frac{\sigma^2}{n}$

(2) $X_1 = X_2 = \dots = X_n$, $Var(\bar{X}) = Var(X_1) = \sigma$ In this case averaging doesn't help at all

Bagging

because they are not identical.

- Averaging these B trees reduces the "variance". (Why?)
- B is not a critical parameter with bagging; a very large value of B will not lead to overfitting.
- In practice, use B sufficiently large so that the error has settled down.
- For a given test observation in `classification`, we can record the class predicted by each of the B trees, and take a vote.
  - by default, the final prediction is the most commonly occurring class among the B predictions.
  - For binary classification, we can always change the threshold for $P(Y = 1|X = x)$; for bagging (or random forest), this probability is computed as the fraction of B trees that predict 1.

$\mathbb{1}(P(Y=1|X=x) > \text{threshold})$

Variable Importance Measures

- Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using
  - the RSS (for bagging regression trees) try for DSO 530 project
  - the Gini index (for bagging classification trees)
- In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor
- For bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.
- These variable importance measures are widely-used. However, they should be used with Caution. Why ↓

① These measures are specific to the tree ensemble models.

② These measures do not take into consideration about prediction performance eval metrics

Random Forest(s)

(this is less pronounced in regression since RSS is used in R^2 & RMSE metrics.

p predictors

Sample → Sample 1^* ⇝ $\hat{f}^{1*}$

(m out of p predictors)

→ Sample B^* ⇝ $\hat{f}^{B*}$

{this step makes trees less equivalent to each oth}

{therefore, w/ different trees, fewer required}

- Bagging constructs trees that are too "similar" (why?), so it probably does not reduce the variance as much as we wish to.
- Random forests provide an improvement over bagged trees by a small tweak that *decorrelates* the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, *a random sample of m predictors* is chosen as split candidates from the full set of *p* predictors. The split is allowed to use *only one* of those *m* predictors.
- So bagging is a special case of random forest when $\boxed{m = p}$.

Random Forest(s)

- If one does not want to spend extra efforts on m, one might use $m = \sqrt{p}$ as a canonical choice for classification and $m = p/3$ as a canonical choice for regression.
- As with bagging, random forests will not overfit if we increase B, so in practice people use B sufficiently large for the error rate to have settled down.
- Random forest is a really good `off-the-shelf` algorithm.

Python implementation

- `RandomForestClassifier` and `RandomForestRegressor` in `sklearn` implement random forests in Python for classification and regression problems, respectively
- Our tutorial covers `RandomForestClassifier`
- Parameters:
  - `n_estimators` (default 100) is the number of trees in the forest
  - `max_features` (default `sqrt(n_features)`) is the number of features to consider when looking for the best split. $\longrightarrow m$
- You can learn pick up `RandomForestRegressor` from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- `RandomForestRegressor` in `sklearn` has a default setting of `max_features=n_features`. $\longrightarrow$ bagging $(m=p)$

A definition and some questions

- If you need to communicate a one sentence ad-hoc definition of random forests:
  – *Random forests are bagged decision tree models that split on a random subset of features on each split*.
- Q: In random forest algorithms, we restrict our attention to randomly selected m out of p features in each split. Now we change this procedure to restriction to the first m features (i.e., $X_1, \cdots, X_m$) in every split. Do you expect the new procedure to work well? And why?
- Q: If a decision tree partitions the feature space into regions $R_1 \cdots, R_J$, can any of these regions be a ball?
- Q: Is random forest always a better algorithm compared to decision trees?
- Q: What are the sources of randomness that a random forest model has? Hint: 3.

A definition and some questions

- If you need to communicate a one sentence ad-hoc definition of random forests:
  – *Random forests are bagged decision tree models that split on a random subset of features on each split*.
- Q: In random forest algorithms, we restrict our attention to randomly selected m out of p features in each split. Now we change this procedure to restriction to the first m features (i.e., $X_1, \cdots, X_m$) in every split. Do you expect the new procedure to work well? And why?
- Q: If a decision tree partitions the feature space into regions $R_1 \cdots, R_J$, can any of these regions be a ball? No
- Q: Is random forest always a better algorithm compared to decision trees? No, why? Less interpretability,
- Q: What are the sources of randomness that a random forest model has? Hint: 3. ① randomness due to sampling ② randomness due to bootstrapping ③ randomly select m out of p features in every split. * specific to random forest

No, ① maybe more important features come later. (after m)
② making similar trees makes it more like bagging than random forest.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- Boosting is an ensemble technique where new models are added to correct the errors made by existing models.
- A differentiating characteristic `Random forest: parallel vs. boosting: sequential`

A boosting algorithm for regression (optional)

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

what??? optional

In practice (optional)

- XGboost
  - XGBoost is one popular implementation of boosting algorithms for its model performance.
  - It is more complicated than what we described on the previous slide. For example, subsampling and thrinkage ideas are adopted.
  - `xgboost` available in Python.
  - A tutorial: https://xgboost.readthedocs.io/en/latest/tutorials/model.html.
- LightGBM another popular one.
  - fast speed
  - https://lightgbm.readthedocs.io/en/latest/Python-Intro.html