Notes on Statistical and Machine Learning

Ensemble Learning

Chapter: *21* **Prepared by:** *Chenxi Zhou*

This note is prepared based on *Chapter 16, Ensemble Learning* in Hastie, Tibshirani, and Friedman (2009).

I. Introduction

1. **Examples of Ensemble Learning:**
   (a) Bagging;
   (b) Random forests;
   (c) Boosting;
   (d) Stacking;
   (e) $\cdots$
2. **Two Tasks in Ensemble Learning:** Ensemble learning can be broken down into two tasks:
   (a) developing a population of base learners from the training data, and
   (b) combining them to form the composite predictor.

II. Boosting and Regularized Paths

1. **Setup:** Consider a regression problem using the squared error loss function.
2. **Basis Functions:** Consider the dictionary of all possible J-terminal node regression trees

$$\mathcal{T} := \{T_1, T_2, \cdots, T_K\} \tag{1}$$

   that could be realized on the training data as basis functions in $\mathbb{R}^p$, where K is a large number and denotes the number of trees.
3. **Model Specification:** The linear model is

$$f(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k T_k(\mathbf{x}), \qquad \text{for all } \mathbf{x} \in \mathbb{R}^p. \tag{2}$$

4. **Objective Function:** To avoid the overfitting issue, we impose regularization by adding a penalty term and solve the following minimization problem

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{K} \alpha_k T_k(\mathbf{x}) \right)^2 + \lambda \cdot P(\boldsymbol{\alpha}) \right\}, \tag{3}$$

where $\boldsymbol{\alpha} := (\alpha_1, \alpha_2, \cdots, \alpha_K)^\top$ and $P : \mathbb{R}^K \to [0, \infty)$ is a function of the coefficients that penalizes larger values.

5. **Examples of P:** Examples of P include

   (a) ridge

$$P_{\text{ridge}}(\boldsymbol{\alpha}) := \sum_{k=1}^{K} \alpha_k^2;$$

   (b) lasso

$$P_{\text{lasso}}(\boldsymbol{\alpha}) := \sum_{k=1}^{K} |\alpha_k|.$$

   Remark. As we increase the value of λ, non-zero coefficients are shrunk by the lasso penalty function toward 0.

   If we use a sufficiently large value for λ, many of the $\widehat{\alpha}_k(\lambda)$ will be equal to 0, where $\widehat{\alpha}_k(\lambda)$ is the k-th element of the minimizer of (3). That is, only a small fraction of all possible trees enter the model (2). In particular, we have

$$|\widehat{\alpha}_k(\lambda)| < |\widehat{\alpha}_k(0)|, \qquad \text{for all } k = 1, 2, \cdots, K \text{ and all } \lambda > 0.$$

6. **Algorithm:** Due to the very large number of basis functions in (1), directly solving (3) with the lasso penalty P_{lasso} is *not* possible. Instead, we use a forward stagewise strategy to closely *approximate* the effect of the lasso, which is very similar to boosting.

Algorithm 1 Forward Stagewise Linear Regression

Require: Initialize $\check{\alpha}_k = 0$, for all $k = 1, \cdots, K$;
Require: Set $\varepsilon > 0$ to some small constant;
Require: A large integer M.
1: **for** $m = 1$ to M **do**
2: Compute

$$(\beta^*, k^*) := \arg\min_{\beta, k} \sum_{i=1}^{n} \left(y_i - \sum_{\ell=1}^{K} \check{\alpha}_\ell T_\ell(\mathbf{x}_i) - \beta T_k(\mathbf{x}_i) \right)^2;$$

3: Update

$$\check{\alpha}_{k^*} \quad \leftarrow \quad \check{\alpha}_{k^*} + \varepsilon \cdot \text{sign}(\beta^*).$$

4: **end for**
5: **return** $f_M(\mathbf{x}) = \sum_{k=1}^{K} \check{\alpha}_k T_k(\mathbf{x})$.

Remark 1. Although phrased in terms of tree basis functions T_k, Algorithm 1 can be used with *any* set of basis functions.

Remark 2. In Algorithm 1, initially all coefficients are zero; this corresponds to $\lambda = \infty$ in (3). At each successive step, the tree T_{k^*} is selected that best fits the current residuals. Its corresponding coefficient $\check{\alpha}_{k^*}$ is then incremented or decremented by an infinitesimal amount, while all other coefficients $\check{\alpha}_k$, where $k \neq k^*$, are left unchanged.

III. Learning Ensembles — Importance Sampled Learning Ensemble

1. **Setup:** We consider functions of the form

$$f(\mathbf{x}) = \alpha_0 + \sum_{T_k \in \mathcal{T}} \alpha_k T_k(\mathbf{x}),$$

where $\mathcal{T}$ is a dictionary of basis functions, typically trees.

Remark. For gradient boosting machine and random forests, the cardinality of $\mathcal{T}$ is very large, and it is quite typical for the final model to involve many thousands of trees. We would like to shrink some coefficients α_k to be 0.

2. **General Procedure of Importance Sampled Learning Ensemble:** The general procedure of *importance sampled learning ensemble* involves the following two steps:

   (a) Induce a finite dictionary

$$\mathcal{T}_L := \{T_1(\mathbf{x}), T_2(\mathbf{x}), \cdots, T_M(\mathbf{x})\}$$

   of basis functions from the training data;

(b) Build a family of functions f_λ by fitting a lasso path in $\mathcal{T}_L$:

$$\boldsymbol{\alpha}(\lambda) := \arg\min_{\boldsymbol{\alpha}} \left\{ L\left(Y_i, \alpha_0 + \sum_{m=1}^{M} \alpha_m T_m(\mathbf{x}_i)\right) + \lambda \sum_{m=1}^{M} |\alpha_m| \right\},$$

where $\boldsymbol{\alpha} := (\alpha_0, \alpha_1, \cdots, \alpha_M)^\top$.

Remark. The procedure outlined above can be viewed as a way of post-processing gradient boosting machine or random forests, where $\mathcal{T}_L$ is the collection of trees produced by the corresponding algorithm.

By fitting the lasso path to these trees, we would typically use a *much reduced* set, which would save in computations and storage for future predictions.

3. How to Learn $\mathcal{T}_L$:

(a) *General Philosophy:* A good choice of basis functions in $\mathcal{T}_L$ should satisfy the following criteria:

i. basis functions in $\mathcal{T}_L$ should cover the space well in places where they are needed, and

ii. basis functions in $\mathcal{T}_L$ should be sufficiently different from each other for the post-processor to be effective.

(b) *Linking to Quadrature:* We write the unknown regression function f as

$$f(\mathbf{x}) = \alpha_0 + \int \beta(\gamma) b(\mathbf{x}; \gamma) \mathrm{d}\gamma, \tag{4}$$

where $\gamma \in \Gamma$ indexes the basis functions $b(\,\cdot\,; \gamma)$. Then, numerical quadrature approximates f in (4) by

$$\begin{aligned} f(\mathbf{x}) &\approx \alpha_0 + \sum_{m=1}^{M} w_m \beta(\gamma_m) b(\mathbf{x}; \gamma_m) \\ &= c_0 + \sum_{m=1}^{M} c_m b(\mathbf{x}; \gamma_m), \end{aligned}$$

where $\{\gamma_1, \gamma_2, \cdots, \gamma_m\}$ is a set of evaluation points, w_m is the weight for the m-th evaluation point γ_m, $c_0 = \alpha_0$ and

$$c_m := w_m \beta(\gamma_m), \qquad \text{for all } m = 1, 2, \cdots, M.$$

We consider c_m, instead of w_m and $\beta(\gamma_m)$ separately, because w_m and $\beta(\gamma_m)$ are *not* separately identifiable.

With this setup, for a given set of evaluation points $\gamma_1, \gamma_2, \cdots, \gamma_M$, we can estimate $c_0, c_1, \cdots, c_M$ by minimizing

$$\sum_{i=1}^{n} L\left(Y_i, c_0 + \sum_{m=1}^{M} c_m b(\mathbf{x}_i; \gamma_m)\right).$$

Remark. The remaining question becomes how to choose the evaluation points $\gamma_1, \gamma_2, \cdots, \gamma_M$.

(c) *Example:* If the basis functions are trees, then γ indexes the splitting variables, the split-points and the values in the terminal nodes. Then, numerical quadrature amounts to finding a set of M evaluation points $\gamma_m \in \Gamma$ and corresponding weights c_m so that $f_M(\mathbf{x}) = c_0 + \sum_{m=1}^{M} c_m b(\mathbf{x}; \gamma_m)$ approximates f well over the domain of $\mathbf{x}$.

(d) *Measurement of Lack of Relevance:* Given only a single potential evaluation point $\gamma \in \Gamma$, without the knowledge of other points that will be used with it in the integration rule, we measure its *lack of relevance* by

$$Q(\gamma) := \min_{c_0, c_1} \left\{ \sum_{i=1}^{n} L\big(Y_i, c_0 + c_1 b(\mathbf{x}_i; \gamma)\big) \right\},$$

which is the prediction risk of using $\gamma \in \Gamma$ alone.

The optimal single point rule is obtained by

$$\gamma^* := \arg\min_{\gamma \in \Gamma} Q(\gamma).$$

(e) *ISLE Ensemble Generation:* We use sub-sampling to introduce the randomness and to generate an ensemble of basis functions. The reason why we introduce randomness is to let the resulting basis functions be as different as possible.

The algorithm is outlined in Algorithm 2.

Algorithm 2 Importance Sampled Learning Ensemble

1: Set $f_0 = \arg\min_c \sum_{i=1}^{n} L(Y_i, c)$;
2: **for** $m = 1$ to M **do**
3: Compute

$$\gamma_m := \arg\min_{\gamma} \left\{ \sum_{i \in S_m(\eta)} L(Y_i, f_{m-1}(\mathbf{x}_i) + b(\mathbf{x}_i; \gamma)) \right\};$$

4: Set

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu b(\mathbf{x}; \gamma_m).$$

5: **end for**
6: **return** $\mathcal{T}_{\text{ISLE}} := \{b(\,\cdot\,; \gamma_1), b(\,\cdot\,; \gamma_2), \cdots, b(\,\cdot\,; \gamma_M)\}$.

In Algorithm 2, $S_m(\eta)$ refers to a subsample of $n \cdot \eta$, where $\eta \in (0, 1]$, of the training observations, typically *without* replacement. Suggested values of η is $\eta \leq \frac{1}{2}$, and for large n pick $\eta \sim n^{-\frac{1}{2}}$.

Remark 1. Reducing η increases the randomness.

Remark 2. The parameter $\nu \in [0, 1]$ introduces memory into the randomization process; the larger ν, the more the procedure avoids the newly learned basis function $b(\,\cdot\,;\gamma)$ similar to those found before.

References

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Vol. 1. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.