Boosting

CS534 - Machine Learning

Yubin Park, PhD

Recall our original problem:

$$\min_{f} \mathcal{L}(\mathbf{y}, f(\mathbf{X}))$$

As the original problem is difficult to solve,

we limited our search as follows:

$$\min_{\beta} \mathcal{L}(\mathbf{y}, f(\mathbf{X}; \beta))$$

(Remember the linear models)

This time, let's try to solve

the difficult one directly.

Perhaps, we can try Gradient Descent

on the function itself?

$$\mathbf{f}_k = \mathbf{f}_{k-1} - \rho_k \frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{f})}{\partial \mathbf{f}}\Big|_{\mathbf{f}=\mathbf{f}_{k-1}}$$

where $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^T$

Note that we are looking at

$$\frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{f})}{\partial \mathbf{f}}$$

not

$$\frac{\partial \mathcal{L}(\mathbf{y}, f(\mathbf{X}; \beta))}{\partial \beta}$$

Meaning of the Functional Gradient

Consider Squared Loss for example.

$$\frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{f})}{\partial \mathbf{f}} = \frac{1}{2} \frac{\partial (\mathbf{y} - \mathbf{f})^T (\mathbf{y} - \mathbf{f})}{\partial \mathbf{f}} = -(\mathbf{y} - \mathbf{f})$$

Plugging the above into the update form:

$$\mathbf{f}_k = \mathbf{f}_{k-1} + \rho_k (\mathbf{y} - \mathbf{f}_{k-1})$$

If we initialize $\mathbf{f}_0 = \mathbf{0}$, then

$$\mathbf{f}_1 = \mathbf{f}_0 + \rho_1 (\mathbf{y} - \mathbf{f}_0) = \rho_1 \mathbf{y}$$

$$\mathbf{f}_2 = \mathbf{f}_1 + \rho_2 (\mathbf{y} - \mathbf{f}_1) = (\rho_1 + \rho_2 - \rho_1 \rho_2) \mathbf{y}$$

$$\vdots$$

$$\lim_{k \to \infty} \mathbf{f}_k = \mathbf{y}$$

Approximating the Functional Gradient (1)

This seems too obvious and not that useful.

In the end, we need a function that can process **unseen** data.

One idea is to **approximate** the gradient with another function:

$$\left.\frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{f})}{\partial \mathbf{f}}\right|_{\mathbf{f}=\mathbf{f}_{k-1}} \approx h_k(\mathbf{X})$$

where $h_k(\mathbf{X})$ is a function that maps $\mathbb{R}^{n \times m}$ to $\mathbb{R}^n$.

For example, if $h(\mathbf{X}) = \mathbf{X}\beta$ (a linear regression),

$$\mathbf{y} - \mathbf{f}_{k-1} \approx \mathbf{X}\beta_k$$

$$\mathbf{f}_k = \mathbf{f}_{k-1} + \rho_k \mathbf{X}\hat{\beta}_k$$

Similar to **Forward Stagewise Regression**?

Approximating the Functional Gradient (2)

After K iterations, you have

$$\mathbf{f}_K = \mathbf{f}_0 + \sum_{k=1}^{K} \rho_k h_k(\mathbf{X})$$

For a new dataset, $\mathbf{X}'$, you can use this collection of functions to predict the target variable:

$$\hat{\mathbf{y}} = \mathbf{f}_0 + \sum_{k=1}^{K} \rho_k h_k(\mathbf{X}')$$

Gradient Boosting Machine

1. Initialize $\mathbf{f}_0 = \arg\min_\gamma \mathcal{L}(\mathbf{y}, \gamma)$
2. For $k = 1$ to K:
    1. Set $\mathbf{r}_k = -\frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{f})}{\partial \mathbf{f}}\Big|_{\mathbf{f}=\mathbf{f}_{k-1}}$
    2. Fit $h_k(\mathbf{X})$ to $\mathbf{r}_k$
    3. Find ρ_k that minimizes the loss function
    4. Update $\mathbf{f}_k = \mathbf{f}_{k-1} + \rho_k h_k(\mathbf{X})$

$h(\mathbf{X})$ can be any parametric or non-parametric function.

Decision Tree is a popular choice for $h(\mathbf{X})$.

When Decision Tree is used as $h(\mathbf{X})$, then you can estimate ρ_k for each region (or node) of the estimated decision tree. This variant of Gradient Boosting Machine (GBM) is called **TreeBoost**.

GBM for Classification

$$\mathcal{L}(\mathbf{y}, f(\mathbf{X})) = -\sum_{i=1}^{n} [y_i f(\mathbf{x}_i) - \log(1 + \exp(f(\mathbf{x}_i)))]$$

Note that, for Logistic Regression, we have $f(\mathbf{x}_i) = \mathbf{x}_i \beta$.

Taking the first derivative w.r.t. the function:

$$-\frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{f})}{\partial \mathbf{f}} = [y_1 - p_1, y_2 - p_2, \cdots, y_n - p_n]^T$$

where $p_i = \frac{1}{1+\exp(-f(\mathbf{x}_i))}$.

Stochastic Gradient TreeBoost

GBM can fit the training data very well, sometimes too well. To mitigate overfitting to the training data, Stochastic Gradient TreeBoost (SGTB) adds two regularization mechanisms:

- **Shrinkage**: when updating the function, we reduce the contribution of each model by a factor $0 < \nu < 1$. Thus, the update becomes:

$$\mathbf{f}_k = \mathbf{f}_{k-1} + \nu \rho_k h_k(\mathbf{X})$$

- **Subsampling**: to reduce the variance (in the bias-variance trade-off), at each iteration, we fit a model on randomly subsampled data points. Typically 50% to 70%.

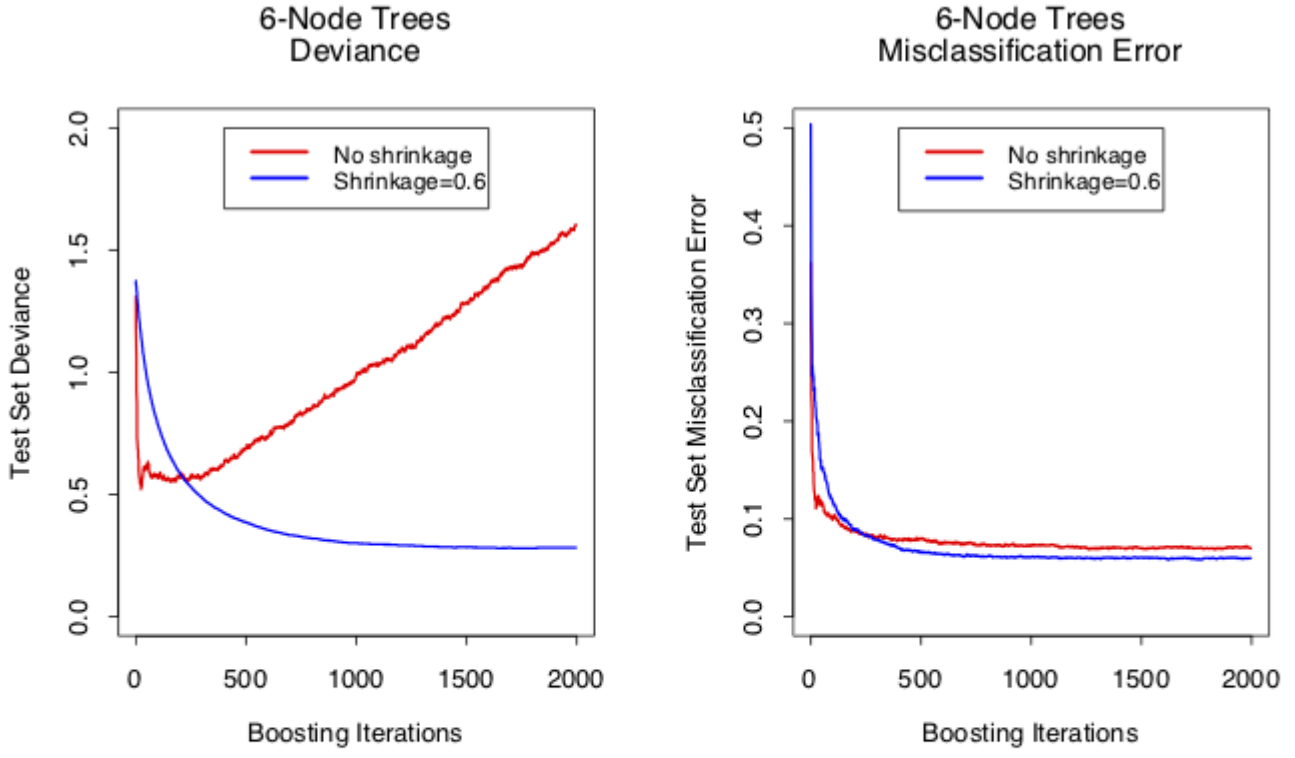

FIGURE 10.11. *Test error curves for simulated example (10.2) of Figure 10.9, using gradient boosting (MART). The models were trained using binomial deviance, either stumps or six terminal-node trees, and with or without shrinkage. The left panels report test deviance, while the right panels show misclassification error. The beneficial effect of shrinkage can be seen in all cases, especially for deviance in the left panels.*

IMPORTANT

Please read the Chapter 10.14 of ESLII.

Questions?