The Bias-Variance Tradeoff

Jack Yang

November 18, 2020

The Regression Setup

Consider the training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where x_i is a random vector of p features. We want to predict y based on the function of x, $f(x)$.

Define the **squared error loss** of estimating y based on $f(x)$ as

$$L(y, f(x)) = (y - f(x))^2.$$

We define the expected loss the **risk** of estimating y based on $f(x)$ as

$$R(y, f(x)) = \mathbb{E}[L(y, f(x))] = \mathbb{E}[(y - f(x))^2].$$

Minimizing Risk

By conditioning on X, we have

$$\mathbb{E}_{X,Y}[(Y - f(X))^2] = \mathbb{E}_X \mathbb{E}_{Y|X}[(Y - f(X))^2 | X = x].$$

Then,

$$\begin{aligned}\mathbb{E}[(Y - f(X))^2] &= \mathbb{E}[(Y - \mathbb{E}[Y|X] + \mathbb{E}[Y|X] - f(X))^2] \\ &= \mathbb{E}[(Y - \mathbb{E}[Y|X])^2] + \mathbb{E}[(\mathbb{E}[Y|X] - f(X))^2] \\ &\quad + 2\mathbb{E}[(Y - \mathbb{E}[Y|X])(\mathbb{E}[Y|X] - f(X))].\end{aligned}$$

The model that minimizes the squared error is

$$f(X) = \mathbb{E}[Y|X].$$

Expected Prediction Error

Given the training data set, our goal is to find some functino $\hat{f}$ that is a good estimate of f.

Assume that we obtain some $\hat{f}$, how well does it estimate f?
The **expected prediction error** of predicting Y using $f(\hat{X})$ is defined as

$$\text{EPE}\left(Y, f(\hat{X})\right) = \mathbb{E}\left[\left(Y - f(\hat{X})\right)^2\right].$$

Reducible and Irreducible Error

$$\begin{aligned}\text{EPE}\left(Y, f(\hat{X})\right) &= \mathbb{E}\left[\left(Y - f(\hat{X})\right)^2 | X = x\right] \\ &= \underbrace{\mathbb{E}\left[\left(f(x) - f(\hat{x})\right)^2\right]}_{\text{reducible error}} + \underbrace{\text{Var}\left(Y \mid X = x\right)}_{\text{irreducible error}}\end{aligned}$$

The **reducible error** is the expected squared error loss of estimation $f(x)$ using $f(\hat{x})$ at a fixed point x.

The **irreducible error** is simply the variance of Y given that $X = x$.

Bias and Variance

The **bias** of an estimator is given by

$$\text{Bias}\left(\hat{\theta}\right) = \mathbb{E}\left[\hat{\theta}\right] - \theta.$$

The **variance** of an estimator is given by

$$\text{Var}\left(\hat{\theta}\right) = \mathbb{E}\left[\left(\hat{\theta} - \mathbb{E}\left[\hat{\theta}\right]\right)^2\right].$$

Decomposition

$$\begin{aligned}\text{MSE}\left(f(x), \hat{f}(x)\right) &= \mathbb{E}\left[\left(f(x) - f(\hat{x})\right)^2\right] \\ &= \underbrace{\left(f(x) - \mathbb{E}\left[f(\hat{x})\right]\right)^2}_{\text{bias}^2\left(f(\hat{x})\right)} + \underbrace{\mathbb{E}\left[\left(f(\hat{x}) - \mathbb{E}\left[f(\hat{x})\right]\right)^2\right]}_{\text{var}\left(f(\hat{x})\right)}\end{aligned}$$

Why is There a Tradeoff

Low variance algorithms tend to be **less complex**, with simple or rigid underlying structure.

- Regression (e.g., Linear Regression)
- Naive Bayes

Low bias algorithms tend to be **more complex**, with flexible underlying structure.

- Decision Trees
- k-Nearest Neighbors
- Non-Linear Algorithms

Overfitting of Decision Trees

Overfitting happens when the learning algorithm continues to develop hypotheses that reduce training set error at the cost of an increased test set error.

There are several approaches to avoiding overfitting in building decision trees.

- Stop growing the tree earlier, before it perfectly classifies the training set. (**Pre-pruning**)
- Allow the tree to perfectly classify the training set, and then post prune the tree. (**Post-pruning**)

Bagging

While an individual tree is overfit to the training data and is likely to have large error, bagging (Bootstrap Aggregating) uses the insight that a suitably large number of uncorrelated errors average out to zero to solve this problem. Bagging is known to reduce the variance of the algorithm.

Given a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ and separate a test set $\{x_{t_1}, x_{t_2}, \ldots, x_{t_T}\}$

1. For $b = 1, 2, \ldots, m$, draw a bootstrap sample D_b and build a decision tree T_b to the bootstrapped sample.
2. For test observation $t = 1, 2, \ldots, T$, for each b, we calculate the fitted value $\hat{y_{tb}}$ according to the tree. Then, combine all fitted value based on a single tree to a single fitted value.

Random Forest: Extension of Bagging

In short, random forest is a collection of unproved decision trees. The purpose is to improve prediction accuracy.

If we can build many small, weak decision trees in parallel, we can then combine the trees to form a single, strong learner by averaging or taking the majority vote.

At each split within each tree, we only consider splitting a randomly-chosen subset of the predictors.