Boosting

Chris Cornwell

May 8, 2025

Outline

Outline

Boosting - General Idea

Boosting is a technique that trains models which each have high Bias – low Variance – and it creates a particular linear combination of these models which is a better fit for the data (this “boosted” model having increased Variance).

- Ideally, the simple models, that have high Bias, allow for very efficient training, keeping computation times small.
- Two popular algorithms for this general idea are AdaBoost and “Gradient Boosted Decision Trees” – which are used in a tool called XGBoost.

The 0-1 Loss Function and Learning Algorithms

To discuss our assumptions with boosting, we will consider the setting of binary classification. The loss function in this slide clearly generalizes to Classification tasks with more labels. One can also adapt it to a Regression task, though it is not obvious how and requires a choice of another hyperparameter.
In binary classification, we'll write the empirical 0-1 **loss function** as $\mathcal{L}_{\mathcal{S}}^{01}$; it is defined as follows. Given a prediction function $f : \mathbb{R}^d \to \{1, -1\}$ that has been trained on data, the loss function on data $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is simply the probability that $f(\mathbf{x}_i) \neq y_i$. That is,

$$\mathcal{L}_{\mathcal{S}}^{01}(f) = \frac{|\{i\ :\ f(\mathbf{x}_i) \neq y_i\}|}{n}.$$

The population 0-1 loss would be the probability that $f(\mathbf{x}) \neq y$ when $(\mathbf{x}, y)$ is drawn from the population.
A **learner** or **learning algorithm** is a procedure or rule that finds (or trains) $f_{\mathcal{S}}$ from given training data $\mathcal{S}$. In boosting, we begin with a learner that does okay, but may not have a very small value of $\mathcal{L}_{\mathcal{S}}^{01}(f_{\mathcal{S}})$ and build one which will have much better 0-1 loss through linear combinations.
Need to include information about Loss with probability vector **p**.

Weak Learners

For γ, with $0 < \gamma < \frac{1}{2}$, we call a learning algorithm a γ-**weak learner** if, for any $0 < \delta < 1$, there is a positive integer N so that whenever $|\mathcal{S}| \geq N$ then the learning algorithm determines $f_{\mathcal{S}}$ so that $\mathcal{L}^{01}_{\mathcal{S}}(f_{\mathcal{S}}) \leq \frac{1}{2} - \gamma$.

Note that, for a sufficiently large set of training data $\mathcal{S}$, a γ-weak learner will produce a prediction function $f_{\mathcal{S}}$ that is incorrect less than half of the time. That is, for small γ, this is a bit better than using a coin flip.
Decision stumps can be viewed as weak learners. More precisely, for data where the label should have a single sign ($+1$ or -1) for all $\mathbf{x}$ that have some coordinate x_j that is in a fixed interval, decision stumps are γ-weak learners for some $0 < \gamma < \frac{1}{6}$.
Since decision stumps are decision trees that have only depth 1, and they are weak learners for such a classification problem, decision trees are also weak learners.

- Hence, it is common to use decision trees with small max depth as the base model in a boosting algorithm.

Outline

Description of AdaBoost

One chooses a number T of *rounds* for the AdaBoost algorithm. In round t, with $1 \leq t \leq T$, say that f_t is the basic model that is trained in that round. That round will also have a corresponding coefficient w_t. The final (boosted) model is given by

$$f(\mathbf{x}) = \operatorname{sign}(w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_T f_T(\mathbf{x})).$$

The procedure for finding f_t and w_t is the following.
In each round t, there is a probability vector $\mathbf{p}^{(t)} \in \mathbb{R}^n$. In round 1, we give $\mathbf{p}^{(1)}$ uniform weights; that is, $\mathbf{p}^{(1)} = (1/n, 1/n, \ldots, 1/n)$.
Now, in round t of AdaBoost:

1. Use your learner, with weights $\mathbf{p}^{(t)}$, to determine f_t.

Description of AdaBoost

One chooses a number T of *rounds* for the AdaBoost algorithm. In round t, with $1 \leq t \leq T$, say that f_t is the basic model that is trained in that round. That round will also have a corresponding coefficient w_t. The final (boosted) model is given by

$$f(\mathbf{x}) = \operatorname{sign}(w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_T f_T(\mathbf{x})).$$

The procedure for finding f_t and w_t is the following.
In each round t, there is a probability vector $\mathbf{p}^{(t)} \in \mathbb{R}^n$. In round 1, we give $\mathbf{p}^{(1)}$ uniform weights; that is, $\mathbf{p}^{(1)} = (1/n, 1/n, \ldots, 1/n)$.
Now, in round t of AdaBoost:

1. Use your learner, with weights $\mathbf{p}^{(t)}$, to determine f_t.
2. Compute the weighted empirical loss $\varepsilon_t = \mathcal{L}_{\mathcal{S}}(f_t, \mathbf{p}^{(t)})$.
3. Define $w_t = \ln\left(\frac{1}{\varepsilon_t} - 1\right)$.
4. Find $\mathbf{p}^{(t+1)} = (p_1^{(t+1)}, \ldots, p_n^{(t+1)})$ by setting

$$p_i^{(t+1)} = \frac{p_i^{(t)} e^{-w_t y_i f_t(\mathbf{x}_i)}}{\sum_{j=1}^n p_j^{(t)} e^{-w_t y_j f_t(\mathbf{x}_j)}}.$$