

1 Boosting

Let \mathcal{X} be an input space and let \mathcal{W} be a set of classification rules such that for $f \in \mathcal{W}$ and $x \in \mathcal{X}$ we have $f(x) \in \{-1, 1\}$. We will call \mathcal{W} the set of weak learners. For example, we can take \mathcal{W} to be the set of decision trees f with $|f| \leq \lambda$ for a fairly small value of λ .

We now consider linear combinations of weak learners. In particular let F be a sequence of N weak learners f_1, \dots, f_N , and let β be a sequence of N real numbers β_1, \dots, β_N . We write $\beta \cdot F(x)$ for the following sum.

$$\beta \cdot F(x) = \beta_1 f_1(x) + \dots + \beta_n f_N(x)$$

Now are interested in finding β and F minimizing the following where $L(m_t)$ is a loss function.

$$\langle \beta^*, F^* \rangle = \operatorname{argmin}_{\beta, F} \sum_{t=1}^T L(m_t) \quad (1)$$

$$m_t = y_t \beta \cdot F(x_t) \quad (2)$$

We start with β_0 and f_0 defined as follows.

$$\beta_0 = 1 \quad (3)$$

$$f_0 = \operatorname{argmin}_{f \in \mathcal{W}} \sum_{t=1}^T I[y_t \neq f(x_t)] \quad (4)$$

In practice the optimization problem in (4) may not be solvable exactly. For example, if \mathcal{W} is the set of all four node decision trees over a large test set \mathcal{Q} (to be used at nodes of the trees) then finding $f_0 \in \mathcal{W}$ of minimum error rate is prohibitively expensive. However, a greedy tree growth algorithm (which greedily minimizes log loss) can be used to heuristically minimize the error rate of f_0 . In general f_0 is some heuristic approximation to the solution of (4).

Now we can assume that we have weights β_0, \dots, β_n and classifiers f_0, \dots, f_n and we want to extend these sequences with β_{n+1} and f_{n+1} . We will do this so as to greedily improve the objective function (1). In particular, we can write the loss that would result from selecting β_{n+1} and f_{n+1} as follows.

$$L = \sum_{t=1}^T L(m_t) \quad (5)$$

$$m_t = y_t (\beta_0 f_0(x_t) + \dots + \beta_n f_n + \beta_{n+1} f_{n+1}(x_t)) \quad (6)$$

For a particular choice of f_{n+1} we are interested in the following quantity.

$$\begin{aligned} \frac{\partial L}{\partial \beta_{n+1}} &= \sum_{t=1}^T \frac{\partial L(m_t)}{\partial m_t} \frac{\partial m_t}{\beta_{n+1}} \\ &= \sum_{t=1}^T \frac{\partial L(m_t)}{\partial m_t} y_t f_{n+1}(x_t) \end{aligned} \quad (7)$$

We will greedily select f_{n+1} so that it contributes as much as possible to the reduction in loss. We interpret this as saying that we want the derivative of loss with respect to β_{n+1} to be as negative as possible (so that loss is reduced as β_{n+1} increases away from zero). This leads to the following criterion for selecting f_{n+1} .

$$\begin{aligned} f_{n+1} &= \operatorname{argmin}_{f \in \mathcal{W}} \sum_{t=1}^T \frac{\partial L(m_t)}{\partial m_t} y_t f_{n+1}(x_t) \\ &= \operatorname{argmin}_{f \in \mathcal{W}} \sum_{t=1}^T \left(-\frac{\partial L(m_t)}{\partial m_t} \right) (-y_t f_{n+1}(x_t)) \\ &= \operatorname{argmin}_{f \in \mathcal{W}} \sum_{t=1}^T \alpha_t I[y_t \neq f(x_t)] \\ \alpha_t &= -\frac{\partial L(m_t)}{\partial m_t} \end{aligned} \quad (8)$$

Loss generally declines as the margin increases (the one exception being quadratic loss $(1 - m_t)^2$). So usually we have $\alpha_t \geq 0$. We then have that (8) is minimizing a training error on a reweighting of the training data. So to use boosting we need to be able to solve, or heuristically approximate a solution to, equation (8). This means that we need a learning algorithm that can take as input a weighted sample. We consider this in more detail below. Once f_{n+1} is selected according to (8), we can select β_{n+1} so as to minimize the loss in (5) as follows.

$$\beta_{n+1} = \operatorname{argmin}_{\beta} \sum_{t=1}^T L(y_t (\beta_0 f_0(x_t) + \dots + \beta_n f_n + \beta f_{n+1}(x_t))) \quad (9)$$

Boosting (AdaBoost for a general loss function) is then defined by equations (3), (4), (8), and (9).

2 Learning Trees with Weighted Training Data

Now consider the case when \mathcal{W} is the set of decision trees f satisfying $|f| \leq \lambda$. For example \mathcal{W} might be the set of all five node decision trees or the set of all decision trees consisting of only a single branch (a stump). To (heuristically) solve (8) we need to handle a sample weighted with weights $\alpha_1, \dots, \alpha_T$. If the tree contains more than a single branch we can do this with the same kind of greedy search as is used for normal decision tree learning using the log loss to facilitate the effectiveness of greedy search.

$$f^* = \operatorname{argmin}_{f:|f|\leq\lambda} \sum_{t=1}^T \alpha_t \ln \frac{1}{P(y_t|x_t, f)} \quad (10)$$

3 The Boosting Theorem

The boosting theorem states that weak learning implies strong learning. Weak learning is the assumption that \mathcal{W} has the property that there exists a constant $\delta > 0$ such that for any distribution D on $\mathcal{X} \times \{-1, 1\}$, the set \mathcal{W} contains, and we can efficiently find, a classifier f whose error rate on D is at most $1/2 - \delta$. To use the weak learning hypothesis we renormalize the weights α_t to be a convex combination. Once this is done the weighted sample is a probability distribution on $\mathcal{X} \times \{-1, 1\}$. The weak learning hypothesis implies that f_{n+1} always has an error rate of no more than $1/2 - \delta$ on this distribution where δ is a fixed constant independent of n . The boosting theorem states that if one assumes the weak learning hypothesis, and uses exponential loss $\exp(-m_t)$, then Boosting will achieve a combined rule whose training error rate is no more than $1/\epsilon$ in time polynomial in $1/\epsilon$. In fact, by selecting fresh training data in each iteration of boosting, one can prove that the generalization error can also be driven to $1/\epsilon$ in time polynomial in $1/\epsilon$. However, exponential loss is empirically a bad choice of loss function. Furthermore, the weak learning hypothesis is clearly not true in practice. Still, boosting is empirically powerful when used with standard loss functions. The power of boosting may simply result from the power of greedy search.