Review on boosting algorithms

Vu Tuan Hung and Do Quoc Khanh

9 mai 2012

1 Introduction

2 Two-class classification

In this first part, we present an overview on boosting methods in the two-class classification framework. From a *training* set $(\mathbf{x}_i, y_i)_{i=1,...,N}$ in which $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, we try to construct a function $F : \mathcal{X} \to \mathcal{Y}$ so that when a new value $\mathbf{x}$ is randomly introduced, we have the highest probability to predict correctly the value y corresponding to this value of $\mathbf{x}$. Formally, we want to minimize the probability :

$$\mathbb{P}_{(\mathbf{x},y)}\left(y \neq F(\mathbf{x})\right)$$

The variable $\mathbf{x}$ is called explanatory variables ($\mathbf{x}$ may be multi-variational) and y is called response variable. In the two-class classification framework, $\mathcal{Y} = \{-1, 1\}$.

2.1 Boosting and optimization in function space

We exploit the point of view presented in [2] by considering this problem as an estimation and optimization in function space. Indeed, if there exists a function F^* which minimizes the above error :

$$\begin{aligned} F^* &= arg\min_F \mathbb{P}_{(\mathbf{x},y)}(y \neq F(\mathbf{x})) \\ &= arg\min_F \mathbb{E}_{(\mathbf{x},y)}\left[1(y \neq F(\mathbf{x}))\right] \end{aligned}$$

then we are trying to estimate F^* by a function $\hat{F}$ through the training set $(\mathbf{x}_i, y_i)_{i=1,...,N}$.

Base classifiers. An approach frequently employed by classification algorithms is to suppose F^* belongs to a function class parameterized by $\theta \in \Theta$:

$$F^* \in \mathcal{Q} = \{F(., \theta) | \theta \in \Theta\}$$

so that the problem of estimating F^* becomes an optimization of the parameters on Θ :

$$\hat{\theta} = arg\min_{\theta \in \Theta} \mathbb{E}_{(\mathbf{x},y)}\left[1(y \neq F(\mathbf{x}, \theta))\right]$$

and then we will take $\hat{F} = F(., \hat{\theta}) \in \mathcal{Q}$. For example, with regression tree algorithms, we have :

$$\mathcal{Q} = \left\{ F(x, \theta) = \sum_{k=1}^{K} \lambda_k 1(\mathbf{x} \in R_k) | (\lambda_1,...,\lambda_K) \in \mathbb{R}^K, (R_1,...,R_K) \in \mathcal{P}_{\mathcal{X}} \right\}$$

in which $\theta = (\lambda_{1:K}, R_{1:K})$ and $\mathcal{P}_{\mathcal{X}}$ is the set of all partitions of $\mathcal{X}$ into K disjoint subsets by hyperplans which are orthogonal to axes. Similarly for support vector machines, K disjoint subsets R_1,...,R_K are divided by hyperplans in the reproducing kernel Hilbert space of $\mathcal{X}$ corresponding to some kernel.

We can see that a classifier is characterized by its function sub-space $\mathcal{Q}$ and the corresponding parameter space. Having the base classifiers $\mathcal{Q}_{1:M}$ with parameter spaces $\Theta_{1:M}$, instead of considering each of these classifiers separately, boosting methods consider functions of the following additive form :

$$\hat{F} \in \mathcal{F}_{\mathcal{Q}_1,...,\mathcal{Q}_M} = \left\{ \sum_{m=1}^{M} \beta_m F(.,\theta_m) | \theta_m \in \Theta_m, \forall m = 1,...,M \right\}$$

so that the optimization problem becomes :

$$\left\{ \hat{\beta}_{1:M}, \hat{\theta}_{1:M} \right\} = arg \min_{\beta \in \mathbb{R}^M, \theta_{1:M} \in \Theta_{1:M}} \mathbb{E}_{(\mathbf{x},y)} \left[1(y \neq F(\mathbf{x}; \beta_{1:M}, \theta_{1:M})) \right] \quad (1)$$

Friedman, J. and Hastie, T. in [1] explained boosting as a forward stepwise algorithm for resolve the optimization problem (1). Friedman, J. in [2] considered boosting like optimization algorithm in function space. We will try to adopt the latter to explain all mentioned boosting algorithms. Before going into greater details, we remark that as an classification algorithm, the boosting algorithm has its corresponding function subset which is $\mathcal{F} = \mathcal{F}_{\mathcal{Q}_1,...,\mathcal{Q}_M} = \sum_{m=1}^{M} \mathcal{Q}_m$ so much larger than function subset of all base classifiers, explaining the dominating performance of boosting compared to its base classifiers.

Loss function. We remark that the binary loss function $1(y \neq F(\mathbf{x}))$ is not the only function that reflects the difference between y and $F(\mathbf{x})$. In machine learning, one has other loss functions that are continuous, convex and then easier to do the optimization. For the rest of the report, we use in general $L(y, F(\mathbf{x}))$ to indicate this function.

Optimization on training set. One difficulty is that we can not evaluate the distribution of $(\mathbf{x}, y)$ and calculate the expectation in the right hand side formula of (1). Instead, we only want to optimize on the training data, which means the following optimization problem :

$$F^* = arg \min_F \sum_{i=1}^{N} L(y_i, F(x_i))$$

Put $Q(F) = \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i))$, we remark that $Q(F)$ depends only on N values of the function F at ($\mathbf{x}_1$,...,$\mathbf{x}_N$). We denote for each function F in the function space, a corresponding vector $\overline{F} \in \mathbb{R}^N$ so that $\overline{F} = (F(\mathbf{x}_1),...,F(\mathbf{x}_N))$, and we consider the relaxation problem on vector space $\mathbb{R}^N$: $\overline{F^*} = arg \min_{\overline{F} \in \mathbb{R}^N} Q(\overline{F})$. We try to resolve this problem by recursive numerical methods. Suppose that at $m-1^{th}$ step we obtain a value $\overline{F}_{m-1}$. By numerical methods (Newton-Raphson, algorithm of gradient descent etc.) we find a direction of descent $d_m \in \mathbb{R}^N$ and a coefficient c_m so that if we put $\overline{F}_m = \overline{F}_{m-1} + c_m.d_m$, then $Q(\overline{F}_m) \leq Q(\overline{F}_{m-1})$. But we can not use the direction d_m directly in the original problem with functions F_m because $\overline{F}_m$ identifies the values of functions only at N points. Instead, we have to find a regression function near to the direction d_m ; it means if we use a function subspace $\mathcal{Q}_m$ at m^{th} step, then we have to solve :

$$\{f_m, c\} = arg \min_{f_m \in \mathcal{Q}_m, c \in \mathbb{R}} \|d_m - c.\overline{f_m}\|^2$$

in which $\overline{f_m}$ is the vector of values of f_m at ($\mathbf{x}_1$,...,$\mathbf{x}_N$). After that, we have to look for a coefficient β_m so that, if F_{m-1} is the function obtained at the precedent step, then $Q(F_m) \leq Q(F_{m-1})$

with $F_m = F_{m-1} + \beta_m f_m$. If we start with $F_0 = 0$ then we obtain at M^{th} the additive form $F_M = \sum_{m=1}^{M} \beta_m f_m \in \mathcal{F} = \sum_{m=1}^{M} \mathcal{Q}_m$. We summary this generic algorithm in the following table.

Algorithm 1 : Generic algorithm of boosting
1. Start with $F_0(\mathbf{x}) = 0$.
2. Repeat for $m = 1,2,...,M$:
(a) Search for a direction descent $d_m \in \mathbb{R}^N$ by some Newton-like numerical algorithm of optimization in $\mathbb{R}^N$.
(b) Solve $\{f_m, c\} = arg \min_{f_m \in \mathcal{Q}_m, c \in \mathbb{R}} \|d_m - c.\overline{f_m}\|^2$.
(c) Search for a coefficient β_m so that $Q(F_{m-1} + \beta_m f_m) \leq Q(F_{m-1})$, a line-search strategy can be used.
(d) $F_m = F_{m-1} + \beta_m f_m$.
3. Conclude with $F(\mathbf{x})$.

2.2 One-degree optimization

2.3 Two-degree optimization

3 Multi-class classification and some generalizations

3.1 A traditional approach

3.2 Some generalization of two-class algorithms

3.3 Other generalizations

4 Experiments

4.1 Experiments with simulated data

4.2 Experiments with real data

5 Conclusion

Références

[1] Friedman, J., Hastie, T. & Tibshirani, R. *Additive Logistic Regression : a Statistical View of Boosting*, 2000.

[2] Friedman, J. *Greedy Function Approximation : A Gradient Boosting Machine*, IMS 1999 Reitz Lecture, 2001.

[3] Schapire, R.E. & Singer, Y. *Improved Boosting Algorithms : Using Confidence-rated Predictions*, 1998.