

On Boosting Cumulative Distribution Function Estimates: The L_2 View

Sul boosting di stime di funzioni di ripartizione: l'approccio L_2

Marco Di Marzio

Dip. di Met. Quant. e Teoria Ec.
Università di Chieti-Pescara
dimarzio@dmqte.unich.it

Charles C. Taylor

Dipartimento di Statistica
Università di Leeds
charles@maths.leeds.ac.uk

Riassunto: Si esplorano le potenzialità del *boosting* nella stima di funzioni di ripartizione di vv. cc. continue. In particolare viene proposto un algoritmo di L_2 boosting basato sulla regressione polinomiale locale. Sia da considerazioni teoriche sia da prove numeriche emergono potenzialità di riduzione della distorsione a favore del metodo presentato rispetto a tradizionali metodi di stima non parametrici.

Keywords: Bias Reduction; Machine Learning; Nonparametric Regression; Simulation.

1. Introduction

Given *i.i.d.* realizations of a continuous random variable, consider the classical problem of estimating the cumulative distribution function (cdf). Although non-smooth, the empirical distribution function (edf) enjoys nice properties such as first order minimaxity and uniform minimal variance unbiasedness. But is the edf asymptotically optimal even if the underlying distribution is smooth? A good competitor is the kernel cdf estimator.

Boosting (Shapire, 1990; Friedman *et al.*, 2000) is one of the most studied machine learning ideas. While boosting has been mainly applied to classification or, more recently, regression problems (see Bühlmann & Yu, 2003), Di Marzio & Taylor (2004) use it for density estimation. In this paper we extend this work and present a L_2 boost (Friedman, 2001) algorithm, called L_2 boostCDF, for estimating smooth cdfs. L_2 boostCDF is a smooth estimator which is able to beat both the kernel and edf methods due to its bias reduction properties.

Now we briefly describe L_2 boosting. Consider the canonical regression model

$$Y_i = m(x_i) + \varepsilon_i, \quad i = 1, \dots, n$$

where $\{\varepsilon_i\}$ are *i.i.d.* random noise. A generic weak learner $q(\cdot; \gamma)$, $\gamma \in \Gamma$ is a function that maps the estimates $\{\hat{m}(x_i)\}$ to the respective observed responses with an expected error less than “random guessing”. In the regression context this means that $\hat{m}(\cdot)$ is better than $\bar{Y}$. L_2 boost is a stagewise procedure of residual fitting as follows. An initial least squares fit is operated through $q(\cdot; \gamma)$, so obtaining the smoothed response $\{\hat{Q}_0(x_i)\}$. Given M boosting steps, for $m \in [1, M]$, $\hat{Q}_m(x)$ is the sum of $\hat{Q}_{m-1}(x)$ and a least squares fit of the residuals $\{\tilde{Y}_i = Y_i - \hat{Q}_{m-1}(x_i)\}$ evaluated at x . The L_2 boost estimate is $\hat{Q}_M(x)$.

Note that the choice of M is important, in that the larger is M , the more $\hat{Q}_M(x)$ becomes complex and tends to closely reproduce the sample (overfitting). Therefore, a stopping rule indicating the optimal number of iterations is needed. However, not a great deal of theory exists on this matter, and cross-validation based on subsamples is usually advised.

As intuition confirms, if the weak learner is overly complex (technically, not sufficiently ‘weak’), overfitting can appear after a few steps, or even at the first iteration, yielding poor estimates. In statistical terms a complex learner is characterized by small bias and large variance. Recall that the N-W estimator, that locally approximates with a constant term, is the most biased and stable local polynomial estimator. These considerations confirm the choice of the N-W estimator as the weak learner in our algorithm. Indeed, as we will later point out, a natural and direct way for reducing the complexity of whatever kernel method is to *oversmooth*. Our choice of using a fixed smoothing parameter, h , seems appropriate for our algorithm. In fact, if we optimally select h for every estimation task, we encourage the overfitting tendency, because the ‘learning rate’ of every single step is optimized. This subject will be taken up again in Section 3.2.

In Section 2 we introduce L_2 boostCDF. The key idea is to construct the residuals as the difference between the edf and a Nadaraya Watson (N-W) fit of them. In Section 3 we illustrate the method through simulation experiments: the kernel method beats the edf; the first fit seems equivalent to the kernel method, and the second fit (first boosting iteration) appears superior to it.

2. Cdf estimation

Given a random sample $x_1, \dots, x_n$ from the density f , the kernel density estimate at x is $\hat{f}_h(x) = (nh)^{-1} \sum_{i=1}^n K(\frac{x-x_i}{h})$ where K , called the kernel, is usually a symmetric pdf and the scale factor h is the bandwidth or smoothing parameter. We define our kernel cdf estimator as the integral of the kernel density estimate $\hat{F}_h(x) = \int_{-\infty}^x \hat{f}_h(u) du$ and for a normal kernel this is simply: $\hat{F}_h(x) = n^{-1} \sum_{i=1}^n \Phi((x - x_i)/h)$ where $\Phi(\cdot)$ is the standard normal cdf. If $h \rightarrow 0$, we obtain the edf: $\hat{F}_0(x) := n^{-1} \sum_{i=1}^n I[x_i \leq x]$. If f is smooth enough and $K(\cdot)$ is symmetric, the expectation admits the second order expansion $E\hat{F}_h(x) = F(x) + h^2 \mu_2(K) f'(x)/2 + O(h^4)$ which shows that $\hat{F}_0(x)$ is unbiased. We note that, unlike kernel density estimation, we do not get infinite variance as $h \rightarrow 0$ since $\text{var}(\hat{F}_0(x)) = (F(x)(1 - F(x)))/n$. What about $\text{var}(\hat{F}_h(x))$ for $h > 0$? If we use a Normal kernel, we calculate: $\text{var}(\hat{F}_h(x)) = n^{-1} [E\Phi((x - X)/h)^2 - E^2\Phi((x - X)/h)]$ and $E\Phi((x - X)/h)^2 \approx F(x) - hf(x)/(\sqrt{\pi})$ so $\text{var}(\hat{F}_h(x)) \approx (F(x)(1 - F(x)))/n - hf(x)/(n\sqrt{\pi})$.

In what follows we present our boosting algorithm. Since L_2 boost is designed for regression data, we firstly have to construct a regression framework starting from the data $\{x_i\}$. So we assume a random design in which we take the sample observations $\{x_i\}$ as predictors, and the estimates $\{Y_i = \hat{F}_0(x_i)\}$ as responses. Our weak learner is the N-W smoother:

$$r(x, \{Y_i\}) = m_{nw}(x; \{(x_i, Y_i)\}, h) = \frac{\sum_{i=1}^n K(\frac{x-x_i}{h}) Y_i}{\sum_{i=1}^n K(\frac{x-x_i}{h})}.$$

The residuals are defined as $\{\tilde{Y}_i = \hat{F}_0(x_i) - r(x, \{Y_i\})\}$. In the first iteration we calculate the values $r(x, \{Y_i\})$, then fit $\{\tilde{Y}_i\}$, and finally we add $r(x, \{\tilde{Y}_i\})$ to $r(x, \{Y_i\})$. This way we obtain a new estimate that is the starting point of the next iteration.

Algorithm: L_2 boostCDF

1. (Initialization) Given $x_1, \dots, x_n$,
 - (i) define the responses $Y_i = \hat{F}_0(x_i), i = 1, \dots, n$;
 - (ii) select $h > 0$;
 - (iii) calculate $\hat{F}_h(x) = r(x, \{Y_i\})$.
2. (Iteration) For $m = 1, \dots, M$
 - (i) compute the residuals $\tilde{Y}_i = Y_i - \hat{F}_{h,m-1}(x_i), i = 1, \dots, n$;
 - (ii) calculate $r(x, \{\tilde{Y}_i\})$;
 - (iii) update $\hat{F}_h(x) = \hat{F}_{h,m-1}(x) + r(x)$.

L_2 boostCDF lends itself to a simple statistical interpretation. Primarily, note that $\tilde{Y}_i$ is an estimate of the bias of $\hat{F}_{h,m-1}(x_i)$ taken with the *negative* sign. As a consequence, $r(x)$ approximates $-\text{bias}(\hat{F}_h(x))$. Thus, by adding $r(x)$ to $\hat{F}_h(x)$, we should get a bias reduction. We can iterate arbitrarily many times, because the bias-adjusted estimate $\hat{F}_h(x)$ can be subtracted from $\hat{F}_0(x)$ so obtaining new ‘negative’ biases to fit. Every iteration can be viewed as a bias reduction step.

It can be shown that $\text{bias}(\hat{F}_h(x)) = 3h^2\mu_2(K)f'(x)/2 + O(h^4)$ which is three times that of $\hat{F}_h(x)$ so the starting point might be rather poor. However, some algebra proves that the first boosting iteration reduces the bias to $O(h^4)$. Therefore, provided the variance is well-behaved, boosting has the potential to provide better estimates of the cdf.

3. Numerical Examples

3.1 Simulations

We use, as an accuracy criterion, the average (over all 200 samples) of the sum of the MSEs calculated at points x in $[a : .02 : b]$, i.e.

$$\text{ACC}(\hat{\theta}_i(x)) = \frac{1}{200} \sum_{i=1}^{200} \sum_{x \in \{a:.02:b\}} \left(F(x) - \hat{\theta}_i(x) \right)^2$$

where $\hat{\theta}_i(x)$ is one of the cdf estimates based on the i -th sample. We incorporate *a priori* information about $\hat{\theta}_i(x)$, namely since we are estimating a cdf we have that $0 \leq \hat{\theta}_i(x) \leq 1$ for all x , and so we simply truncate any values which lie outside this range. This very simple, somewhat *ad hoc* procedure greatly improves the performance for smaller samples, and has close analogies with density estimation procedures.

The methods were tested on a range of distributions and sample sizes, including bimodal, skewed, and thick-tailed distributions. Only a selection of the results are presented here due to lack of space.

For sample sizes 50, 150 and 300 we draw 200 samples from: a standard normal cdf with $[a, b] = [-3.5, 3.5]$; a Beta(2,2) distribution with $[a, b] = [0, 1]$; and an equal mixture of normals $N(-0.3, 0.1^2), N(0.4, 0.3^2)$ with $[a, b] = [-3.5, 3.5]$ in ACC. In the Tables below we show the optimal values with the corresponding h for each estimator.

Gaussian Mixture								
n	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h
50	.1527	.03	.1497	.06	.1493	.08	.1724	.06
150	.1103	.02	.1089	.05	.1083	.07	.1181	.03
300	.1018	.01	.1010	.03	.1004	.06	.1057	.02
Beta								
n	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h
50	.0940	.05	.0867	.12	.0837	1.7	.0863	.09
150	.0389	.04	.0362	.10	.0351	.16	.0379	.06
300	.0172	.03	.0156	.09	.0151	.15	.0173	.05
Standard Gaussian								
n	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h	$\hat{F}_h(x)$	h
50	.1961	.19	.1806	.47	.1843	.65	.1903	.38
150	.0665	.15	.0635	.37	.0661	.45	.0661	.25
300	.0378	.13	.0368	.32	.0382	.35	.0380	.22

3.2 Comments

$\hat{F}_h(x)$ seems asymptotically equivalent to the kernel density estimator, while $\hat{F}_h(x)$ improves more rapidly as the sample size increases. In general, we can observe a 4-10% of improvement for the highest sizes. As mentioned, variance inflates along the boosting iterations, when the rise in variance dominates the bias reduction, we can observe estimates worse than $\hat{F}_h(x)$. A natural way to face the variance inflation is operating an over-smoothed fit of residuals in step 2(ii). We consider this as an effective implementation of the shrinkage idea (Friedman, 2001). This strategy can greatly improve the performance with small samples in particular. Contrary to our examples, other researchers have often found shrinkage to be necessary to beat the weak learner when L_2 boost is implemented. In conclusion, we note that in common with previous applications of kernel boosting (Di Marzio & Taylor, 2003, 2004), we have found that the optimal number of boosting steps depends on the underlying distribution. In estimation of the cdf it is usually better to take $M = 1$ or 2. As before, larger smoothing parameters are required when more boosting steps are used.

References

- Bühlmann, P. & Yu, B. (2003). Boosting with L_2 -Loss: regression and classification. *Journal of the American Statistical Association* **98**, 324–449.
- Di Marzio, M. & Taylor, C.C. (2004). Boosting kernel density estimates: a bias reduction technique? *Biometrika*, **91**, 226–233.
- Di Marzio, M. & Taylor, C.C. (2003). Kernel density classification and Boosting. *submitted for publication*.
- Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, **29**, 1189–1232.
- Friedman, J.H., Hastie, T. & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics*, **28**, 337–407.
- Shapire, R.E. (1990). The strength of weak learnability. *Machine Learning*, **5**, 313–321.