Multiple Scale Boosting

Charles A. Pehlivanian

Abstract

We propose a bottom-up, inductive approach to tree selection within the popular Gradient Boosting algorithm and its derivatives (LightFBM, XGBoost, etc.) based on exact solution of the combinatorial optimization program at each iterative step. A recursive, multiscale inductive can then be defined to iteratively approximate the exact solution, leading to classifiers, regressors that compare favorably out of sample to state-of-the art methods for tabular data.

1 Preliminaries

Let $n \in \mathbb{N}$ be positive and set $\mathcal{V} = \{1, \ldots, n\}$. Let $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$ be finite real sequences with $y_i > 0$, for all i. It is assumed that X, Y are orderd to satisfy $\frac{x_1}{y_1} \leq \frac{x_2}{y_2} \leq \cdots \leq \frac{x_n}{y_n}$. In spatial scan statistics applications, for example, the tuple (x_i, y_i) corresponds to realized occurence and estimated baseline attributes for i, possibly associated with a spatial location. Denote by $\mathcal{D} = \mathcal{D}_{X,Y}$ the set of tuples $\{(x_i, y_i)\}$ associated with X, Y. $\mathcal{D}$ is assumed to have an order induced by a *priority* function on the product $X \times Y$.

Definition 1. *A priority function is a function $g\colon \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$ that induces an ordering on the dataset $\mathcal{D}$. We refer to $g(x, y) = \frac{x}{y}$ as the standard priority function.*

A partition $\mathcal{P} = \{S_1, \ldots, S_t\}$ of size t of $\mathcal{V}$ can be identified with a pointset in $\mathbf{R}^2$ by associating $S \subseteq \mathcal{V}$ with the point $(\sum_{i \in S} x_i, \sum_{i \in S} y_i)$, called the *partition point* of S. Set $p_\emptyset = (0, 0)$. In this way the sequences X, Y induce an embedding of $\mathcal{P}$ into $\mathbf{R}^2$. The notion of score function comes from the spatial scan statistics literature.

Definition 2. *A score function is a continuous $f(x, y)\colon \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$, nondecreasing in x, with continuous extension to the origin in any wedge $\mathcal{W}(\mu_1, \mu_2) = \left\{(x, y) : y > 0, \mu_1 \leq \frac{x}{y} \leq \mu_2\right\}$, for $-\infty < \mu_1 \leq \mu_2 < \infty$, with the extension satisfying $f(0, 0) = 0$. If f is of the form $f(x, y) = x^\alpha y^{-\beta}$ for some $\alpha, \beta > 0$, then f is a rational score function.*

Rational score functions will play a central role in this paper. The regularity condition on $\mathcal{W}$ in wedges simply guarantees a continuous extension to the origin on any positive cone in $\mathbf{R}^+$, for rational score functions the constraint corresponds to the constraint $\alpha > \beta$. We do not assume smoothness beyond continuity, nor (quasi)convexity, etc., unless explicitly stated. A score function f induces a real-valued set function on $2^{\mathcal{V}}$ by defining $F(S) = f(\sum_{i \in S} x_i, \sum_{i \in S} y_i)$, for $S \subseteq \mathcal{V}$. For a given f we call the F so defined as the *discrete scoring* of f on (X, Y)

Unless otherwise stated, the sets X, Y will be assumed to be already indexed in (standard) priority order, i.e., $g(x_1, y_1) \leq \cdots \leq g(x_n, y_n)$, where g is the (standard) priority function.

For a given f, $t \leq n$ we are interested in maximal partitions for F, i.e., solutions to the program

$$\mathcal{P}^* = \underset{\mathcal{P}=\{S_1 \ldots, S_t\}}{\operatorname{argmax}} \sum_{j=1}^{t} F(S_j) = \underset{\mathcal{P}=\{S_1, \ldots, S_t\}}{\operatorname{argmax}} \sum_{j=1}^{t} f(\sum_{i \in S_j} x_i, \sum_{i \in S_j} y_i) \tag{1}$$

which for a rational score function becomes

$$\mathcal{P}^* = \underset{\mathcal{P}=\{\pi_1 \ldots, \pi_T\}}{\operatorname{argmax}} \sum_{j=1}^{t} \frac{(\sum_{i \in P_j} x_i)^\alpha}{(\sum_{i \in P_j} y_i)^\beta} \tag{2}$$

Definition 3. *A consecutive subset of $\mathcal{V}$ is a subset of the form $\{j, j+1, \ldots, k\}$ for some $1 \leq j \leq k \leq n$. Consecutive subsets of the form $\{1, \ldots, j\}$, $\{k, \ldots, n\}$ are called ascending consecutive, descending consecutive, respectively. A consecutive partition $\mathcal{P} = \{S_1, \ldots, S_t\}$ is a partition of $\mathcal{V}$ such that each S_i is a consecutive subset.*

Letting $\mathcal{S}_c$ be the set of consecutive partitions of $\mathcal{V}$, it is easy to see that $|\mathcal{S}_c| = \frac{n(n+1)}{2} + 1$.

Definition 4. *The score function F satisfies the Consecutive Partitions Property (CPP) if the solution*

$$\mathcal{P}^* = \underset{\substack{\mathcal{P} \\ |\mathcal{P}|=t}}{\operatorname{argmax}} \sum_{j=1}^{T} F(\sum_{i \in P_j} x_i, \sum_{i \in P_j} y_i)$$

is a consecutive partition, for all X, Y. F satisfies the Weak Consecutive Partitions Property (WCPP) if the solution

$$\mathcal{P}^* = \underset{\substack{\mathcal{P} \\ |\mathcal{P}|\leq t}}{\operatorname{argmax}} \sum_{j=1}^{T} F(\sum_{i \in P_j} x_i, \sum_{i \in P_j} y_i)$$

is a consecutive partition, for any X, Y. F satisfies CPP($\mathbf{R}^+$), WCPP($\mathbf{R}^+$) if it satisfies CPP, WCPP, respectively, for $X \subseteq \mathbf{R}^+$, Y.

It was shown in [7] that if f satisfies some simple properties, that the solution to 1 is realized at a consecutive partition, namely

Theorem 1. *Let $f(x, y)$ be a convex, subadditive score function. Then f satisfies CPP. If f is convex, then it satisfies $WCPP$.*

We can also easily characterize the partition scan statistics corresponding to the rational score functions:

Corollary 1. *Let F be the partition scan statistic with associated score function $f(x, y) = x^{\alpha} y^{-\beta}$, for constants $\alpha, \beta > 0$, and priority function $g(x, y) = \frac{x}{y}$.*

- *If $\alpha - \beta = 1$, and α is even then F satisfies CPP. If $\alpha - \beta \geq 1$, and α is even then F satisfies $WCPP$*
- *If $\alpha - \beta = 1$ then F satisfies $CPP(\mathbf{R}^+)$. If $\alpha - \beta \geq 1$ then F satisfies $WCPP(\mathbf{R}^+)$*

Proof. We note that the Hessian of f has principal minors

$$M_1 = \alpha(\alpha - 1) x^{\alpha - 2} y^{-\beta},$$
$$M_2 = \alpha\beta(\alpha - \beta - 1) x^{2(\alpha-1)} y^{-2(\beta+1)},$$

so that for $x \in \mathbf{R}^+$, f is convex iff $\alpha - \beta \geq 1$, and f is subadditive iff $\alpha - \beta \leq 1$. The result then follows from Theorem (1) above. □

In addition a dynamic programming approach provides an order $\mathcal{O}(n^2 t)$ solution to 2, with storage requirements proportional to nt as outlined in [7].

1.1 Applications to gradient boosting

Given a set of quadratic polynomials $p_i(x) = \frac{1}{2} h_i x^2 + g_i x + c_i$, with $h_i > 0$ for all $i \in \mathcal{V}$, the minimum individual values attained are $\frac{-g_i^2}{2h_i}$ at the values $x_i^* = \frac{-g_i}{h_i}$. Assume now that the polynomials are

ordered by the x_i^*, and for a subset of indices $S \subseteq \mathcal{V}$ define the aggregate polynomials

$$\begin{aligned} p_S(x) = \sum_{i \in S} p_i(x) &= \frac{1}{2} \sum_{i \in S} h_i x^2 + \sum_{i \in S} g_i x + \sum_{i \in S} c_i \\ &= \frac{1}{2} \left(\sum_{i \in S} h_i \right) x^2 + \left(\sum_{i \in S} g_i \right) x + \left(\sum_{i \in S} c_i \right) \\ &= \frac{1}{2} H_S x^2 + G_S x + C_S \end{aligned}$$

So given a set $\{g_i\}$, $\{h_i\}$ with all $h_i > 0$, and $T < n$, the program

$$\begin{aligned} \min_{\mathcal{P}=\{S_1 \ldots, S_t\}} p_{S_j}\left(x_{S_j}, y_{S_j}\right) &= \min_{\mathcal{P}=\{S_1 \ldots, S_t\}} \sum_{j=1}^{T} p_{S_j} \left(\sum_{i \in S_j} x_i, \sum_{i \in S_j} y_i \right) \\ &= \min_{\mathcal{P}=\{S_1 \ldots, S_t\}} \sum_{j=1}^{T} \left(\frac{1}{2} H_S x^2 + G_S x + C_S \right) \\ &= - \max_{\mathcal{P}=\{S_1 \ldots, S_t\}} \sum_{j=1}^{T} \frac{G_j^2}{H_j} \\ &= - \max_{\mathcal{P}=\{S_1 \ldots, S_t\}} F\left(S\right), \end{aligned}$$

where F is the discrete scoring of the function $f\left(x, y\right) = \frac{x^2}{y}$ on (X, Y).

By the results of the previous section, F attains its maximum on a consecutive partition under the priority function $g\left(x, y\right) = \frac{x}{y}$. We take this approach to the split-finding step in the Gradient Boosting algorithm. In particular, the iterative boosting step attempts to find a classifier update f_t on the dataset $\mathcal{D} = (X, Y)$ which minimizes the loss expression

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{3}$$

where $l(y_i, \hat{y}_i)$ is an arbitrary convex loss function, and $\Omega(f_t)$ is an l^2-regularization term. In most cases the f_t is assumed to come from a family of classifiers under suitable parameterization or calibration. In the classical gradient boosting approach, the loss in Equation 3 is approximated by a quadratic polynomial, and a decision tree classifier is chosen as f_t. Denoting by $\{(S_j, w_j)\}_{j=1}^{t}$ the leaf sets (sets of constant leaf value for f_t) and w_j the leaf value, equation 3 becomes

$$\mathcal{L}^{(t)} = \sum_{j=1}^{T} [(\sum_{i \in S_j} \beta_i) w_j + \frac{1}{2} (\sum_{i \in S_j} \alpha_i + \lambda) w_j^2] + \gamma t \tag{4}$$

where $\beta_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}$, $\alpha_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial^2 \hat{y}^{(t-1)}}$. Obtaining the optimal leaf sets and values is equivalent to solving

$$\mathcal{L}^{(t)} = \max_{\mathcal{P}=\{S_1 \ldots, S_t\}} \sum_{j=1}^{T} \frac{(\sum_{i \in S_j} \beta_i)^2}{\sum_{i \in S_j} \alpha_i + \lambda} + \gamma t \tag{5}$$

and setting $w_j = -\frac{\sum_{i \in S_j} \alpha_i}{\sum_{i \in S_j} \beta_i + \lambda}$ (for details see [4]). By our Theorem 1, this optimization can be solved with time requirement no more than $\mathcal{O}(n^2 t)$.

The solution to [5] is exact, while the Taylor series approximation to the loss function is present in the gradient boosting approach. The association of 5 to the original loss minimization requires a specification of $T < n$, which we call the *T-loss resolution*.

1.2 Multiscale boosting

The program in (1) plays a fundamental role in spatial scan statistics problems, in which a single outlier aggregate cluster is identified among regions in which empirical observations and historical baselines are observed. In that context, the observations correspond to our X vector, the baselines to Y and the solution to (1) for $T = 1$ represents the single cluster with maximum excedence as measured by F. Parametric or non-parametric techniques can then be used to assign a p-value to this cluster. The above results extend exact solution methods to $T \geq 2$. The authors also provide a heuristic for determining a single optimal T. We will not take that approach here, instead, along the lines of iterative multiple-resolution solvers (multigrid, algebraic multigrid, etc.) we propose a novel multiple-resolution approach to the traditional gradient boosting algorithm. Given a dataset of size n, we will successively solve T-loss resolution programs, for $n \geq T_1 \geq T_2 \geq \cdots \geq T_m \geq 1$. The approach is similar to the iterative solvers used in discretizations resulting from finite-difference methods. The classic multigrid approach relies on one or more relaxation steps on a fine discretization of the domain, after which the resulting residual (error) is restricted to a coarser grid and the process is repeated there. Once relaxation of the residual on the coarsest grid is complete, the solution is interpolated 'upward' to a finer grid, and the process is repeated. The recursion can take several forms, usually refrerred to as V-cycles, W-cycles, etc. Our approach is similar in nature. Having specified a sequence $\{T_i\}_{i=1}^m$, we define the basic recursive step as finding an exact solution, in $\mathcal{O}n^2T_i$ time, to the T_i-loss resolution problem. The current prediction $\hat{y}$ is the last prediction for the level T_1 model, initially the zero vector. The error vector $y - \hat{y}$ is then passed to a the subsequent T_{i+1} discretization, and the process is repeated. We will concentrate on the so-called W-cycle formulation here, in which a stepsize S_i is specified at each level, a completion of which is defined as a completion of S_j steps of the T_j discretization, for each $j > i$. In effect, the W-cycle process starts by setting $\hat{y}$ to be the zero vector at the finest (T_1) level, passing the residual y to the $T_2, \ldots$, respectively, until T_1 is reached. At that level a base classifier (decision tree, random forest) is fit S_1 times, each time updating $\hat{y}$ and fitting the residual $y - \hat{y}$. The subsequent accumulated $\hat{y}$ is then passed upward to level T_2. In particular, a base classifier is only fit on residual data on the last (S_i) step at level T_i, for all i. The composite prediction is the sum of all the base classifier predictions at each step. An algorithm is specified below. There are fundamental differences between the classic multigrid approach and our proposed approach. The classic approach relies on a discretization of space-time dimensions over which the desired solution is viewed as a projection of a continuous ambient function to the discrete grid. The resulting continuous function and its approximation are thus amenable to frequency or Von Neumann analysis. In our case there is no temporal aspect to the data, the solution is not viewed as a restriction of a continuous $\mathbf{R}^n$-defined function. The notions of low-frequency and high-frequency components of the solutions which are so effectively addressed by the multigrid method are analagous to low-level and high-level features in the classification case, aspects which are difficult to quantify.

Algorithm 1 Consecutive Partition Solution

function DISCOPTSOLUTION(x, y, t)
    # x: X vector
    # y: Y vector
    # t: size of partition
    # Base case ($t' = 1$)
    $(C_x, C_y) = (0, 0)$
    for $j \in \{n, n-1, \ldots, 1\}$ **do**
        $(C_x, C_y) = (C_x, C_y) + (x_j, y_j)$
        $F^*(j, 1) = f(C_x, C_y)$
        $\rho(j, 1) = n + 1$

    # Iterative step
    for $t' \in \{2, \ldots, t\}$ **do**
        for $j \in \{1, \ldots, (n + 1 - t')\}$ **do**
            $F^*(j, t') = -\infty$
            $(C_x, C_y) = (0, 0)$
            for $k \in \{j, \ldots, (n + 1 - t')\}$ **do**
                $(C_x, C_y) = (C_x, C_y) + (x_k, y_k)$
                $F^{j,k} = f(C_x, C_y) + F^*(k + 1, t' - 1)$
                if $F^{j,k} > F^*(j, t')$ **then**
                    $F^*(j, t') = F^{j,k}$
                    $\rho(j, t') = k + 1$

    # Recover the highest-scoring consecutive partitioning for each t'
    for $t' \in \{1, \ldots, t\}$ **do**
        $scores[t'] = F^*(1, t')$.
        $partitions[t'] = []$
        $j = 1$;
        for $t'' \in \{t', (t' - 1), \ldots, 1\}$ **do**
            $j_{next} = \rho(j, t'')$
            Append $[j, j_{next} - 1]$ to $partitions[t']$
            $j = j_{next}$
    return *scores*, *partitions*

Algorithm 2 Multiscale Gradient Boosting Fit

function GH($y, \hat{y}$)
    # y: labels,
    # $\hat{y}$: prediction
    for $i \in \{1, \ldots, n\}$ **do**
        $g_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}} (y_i, \hat{y}_i)$
        $h_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial^2 \hat{y}^{(t-1)}} (y_i, \hat{y}_i)$
    return g, h
function LEAVES(g, h, P, lr)
    # g: first partial vector of loss,
    # h: second partial vector of loss
    # P: partition $\{S_1, \ldots, S_T\}$ of $\{1, \ldots, n\}$,
    # lr: learning rate
    for $j \in \{1, \ldots, T\}$ **do**
        $G_j = 0$, $H_j = 0$
        for $i \in S_j$ **do**
            $G_j = G_j + g_i$
            $H_j = H_j + h_i$
        $l_j = -lr \cdot \frac{G_j}{H_j}$
    return l
function AGGREGATECLASSIFIER(c)
    # c: classifier (weak or composite) to aggregate
    append c to $classifiers$ vector
    return $classifiers$

Algorithm 3 Multiscale Gradient Boosting Fit (cont.)

function CLASSIFYMULTI(c, d)
    # c: composite classifier,
    # d: dataset
    $p = 0$
    for $cls \in c$ **do**
        $p = p + c.classify\,(d)$
    return p
function FITMULTI($d, l, p, Tvec, i, lr$)
    # d: dataset, l: labels, p: current prediction
    # Tvec: vector of partition sizes, i: current index into Tvec
    # lr: learning rate
    #
    # Assume the weak classifier has the methods .fit() .classifiy()
    #
    $g, h = GH\,(l, p)$
    $Scores, P = DISCOPTSOLUTION\,(g, h, Tvec[i])$
    $leaves = LEAVES\,(g, h, P, lr)$
    $i = i + 1$
    if $i < size\,(Tvec)$ **then**
        $csf = fitmulti\,(d, leaves, p, Tvec, i, lr)$
    else
        $csf = fitweak\,(d, leaves)$
    $c = aggregateclassifiers\,(csf)$
    return c

1.3 Loss function specification

In this section we attempt to answer the question: What choice of loss function $\mathcal{L}$ is best for a given problem? We first assume a normalization, that the values y_i all lie in $\{\pm 1\}$. Outside of typical assumptions ($\mathcal{L}\,(y, y) = 0$, $\mathcal{L} \geq 0$, etc) the only requirement assumed in our approach [[4]] is that the $\mathcal{L}$ be C^2 smooth. This is in order to write the approximation

$$\begin{aligned}\mathcal{L}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \\ &\approx \sum_{j=1}^{T} [(\sum_{i \in S_j} \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}) f_t\,(x_j) + \frac{1}{2} \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial^2 \hat{y}^{(t-1)}} f_t\,(x_j)^2] + \gamma t,\end{aligned}$$

where $f_t\,(x_j)$ denotes the constant value of our next-step classifier f_t on the set Y indexed by S_j. The approach outlined so far attempts to minimize the quadratic above by recasting it as a maximization, equivalent to finding the vertex of a convex quadratic function. But the approximation, a Taylor series, is only valid locally at $y_i = \hat{y}^{(t-1)}$, and the global minimum may not have a direct relationship to the global minimum of $l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$. A learning rate parameter in $(0, 1)$ is usually used to partially address this problem, although not as directly as a per-sample point weight as in AdaBoost. In addition, for some loss functions with no global minimums (such as exp loss $l(y, \hat{y}) = \exp\,(-y \cdot \hat{y})$) the loss function is never 0 and additional work is expended by a classifier in minimizing an error for a point which is already correctly classified. In this case forcing the base classifier to fit the residual on those points may sacrifice quality on misclassified points whose residual represents a bona fide error. In order to address these shortcomings we first note that the expression in (5) only depends

on the expressions $\sum_{j=1}^{T} \frac{\sum_{i \in S_j} \beta_i}{\sum_{i \in S_j} \alpha_i}$, for all i. So only the ratios $\Psi_i = -\frac{\frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}}{\frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial^2 \hat{y}^{(t-1)}}}$ are relevant. This expression is related to the *loss margin* of $\mathcal{L}$ which has shown to be of importance in estimating the quality of the *a posteriori* distribution estimation. In the multiscale boosting case, the desired leaf values specified in *LEAVES*, modulo multiplication by the learning rate, are the Ψ_i. So instead of speciying a loss function l then differentiating to obtain the ratios, we specify the functional form of $\Psi(y, \hat{y})$ directly. Since we are solving an error propagation problem at each level we need Ψ to act as a 'guidance system'. Although each $y_i \in \{\pm 1\}$, we will be solving an error propagation problem at each step for which the propagated y_i may be any real number. We propose the following desirable properties of y

- For $y < 0$, $l(y, \hat{y}) < 0$ for $\hat{y} > \alpha$, for some $\alpha \leq 0$, while $l(y, \hat{y}) = 0$ for $\hat{y} < \alpha$.
- Similarly, for $y >= 0$, $l(y, \hat{y}) \geq 0$ for $\hat{y}$ for $y \leq \beta$, for some $\beta \geq 0$, while $l(y, \hat{y}) = 0$ for $\hat{y} > \beta$.
- For $y < 0$, $l(y, \hat{y})$ is nonincreasing for $\hat{y} > \alpha$.
- For $y > 0$, $l(y, \hat{y})$ is nondecreasing for $\hat{y} \leq \beta$.

The functional form of Ψ is given, for some well-known classifier loss functions.

We then have the PDE

$$-\frac{\frac{\partial l(y,\hat{y})}{\partial \hat{y}}}{\frac{\partial^2 l(y,\hat{y})}{\partial \hat{y}^2}} = \Psi \tag{6}$$

$$l(y, y) = 0 \tag{7}$$

$$\Psi(-1, \hat{y}) = a(\hat{y}) \tag{8}$$

$$\Psi(1, \hat{y}) = b(\hat{y}) \tag{9}$$

It is left to specify a, b.

We can study the forms of a, b for some popular classification loss functions. These are shown in the table

Name	**Loss Specification**	Ψ
Square Loss	$\mathrm{l}_{sq}(y, \hat{y}) = (1 - y \cdot \hat{y})^2$	$\Psi_{sq}(y, \hat{y}) = \frac{1}{y} - \hat{y}$
Exp Loss	$\mathrm{l}_{exp}(y, \hat{y}) = \exp(-y \cdot \hat{y})$	$\Psi_{exp}(y, \hat{y}) = \frac{1}{y}$
Arctan Loss	$\mathrm{l}_{arc}(y, \hat{y}) = (2 \arctan(y \cdot \hat{y}) - 1)^2$	$\Psi_{arc}(y, \hat{y}) = -\frac{\left(y^2\hat{y}^2+1\right)(2\arctan(y-\hat{y})-1)}{2y(-y\hat{y}+2y\hat{y}\arctan(y\hat{y})-1)}$
Logistic Loss	$\mathrm{l}_{log}(y, \hat{y}) = \log(1 + \exp(-y \cdot \hat{y}))$	$\Psi_{log}(y, \hat{y}) = \frac{\exp(-y \cdot \hat{y})}{y} + \frac{1}{y}$
Savage Loss	$\mathrm{l}_{sav}(y, \hat{y}) = \frac{1}{(1+\exp(y \cdot \hat{y}))^2}$	$\Psi_{sav}(y, \hat{y}) = \frac{1+\exp(y \cdot \hat{y})}{y(2\exp(y \cdot \hat{y})-1)}$

We first note that all of the loss functions except l_{exp} are unbounded near $y = 0$ except for l_{exp}.

We can solve (?) above by writing it as $\frac{\partial}{\partial \hat{y}} \log f_{\hat{y}}(y, \hat{y}) = -\frac{1}{\Psi}$, obtaining

$$l(y, \hat{y}) = l(y, 0) + \int_0^y l_{\hat{y}}(y, 0)\, e^{-\int_0^w \frac{dz}{\Psi(y,w)}} dw \tag{10}$$

Our intention is to specify a synthetic loss function Ψ without regard to the initial loss function l, although (10) is instructive.

An examination of Ψ for some popular loss function specifications can help determine which loss functions are suitable for use in Gradient Boosting. What are desirable qualities in Ψ? We are only concerned with the behavior of the restrictions $\Psi(-1, \hat{y})$ and $a(\hat{y}) = \Psi(1, \hat{y})$. For the first restriction to $y = -1$, we require that a be nonpositive for $\hat{y} > 0$. It can be argued that a can be identically 0 for $\hat{y} \leq 0$, although we found that this is not condusive to optimal out-of-sample prediction. The reason is that once the loss is 0 for a particular set of features,

1.4 Empirical results

[Forthcoming]

References

[1] Francis Bach, *Learning with Submodular Functions: A Convex Optimization Perspective*, Carnegie Mellon University, Pittsburgh, USA, 2011

[2] Charles A. Pehlivanian, Daniel B. Neill, *Efficient Optimization of Partition Scan Statistics via the Consecutive Partitions Property*, preprint, 2021

[3] J.Friedman, *Greedy function approximation: a gradient boosting machine*, Annals of Statistics, 29(5):1189–1232, 2001.

[4] Chen, Tianqi and Guestrin, Carlos, *XGBoost*, Proceedings of the 22nd ACM SIGKDD International Confernce of Knowledge Discovery and Data Mining, August, 2016.

[5] LightGBM documentation, https://lightgbm.readthedocs.io/en/latest/

[6] Kulldorff, Martin, *A spatial scan statistic*, Communications in Statistics: Theory and Methods, 26(6):1481-1496, 1997.

[7] Pehlivanian, Charles A. and Neill, Daniel B., *Efficient optimization of partition scan statistics via the Consecutive Partitions*, Property. Journal of Computational and Graphical Statistics 32(2): 712-729, 2023