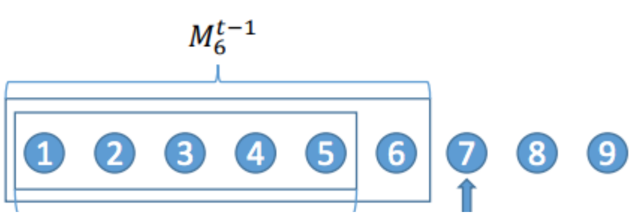
Note for XgBoost, LightGBM, CatBoost

Monday, April 15, 2019 10:18 AM

1. XgBoost
    a. Pre-sorting:
        i. For each node, enumerate over all features
        ii. For each feature, **sort** the instances by feature value
        iii. Use linear scan to decide the best split along that feature basis informatio
        iv. Take the best split solution along all the features
    b. Histogram-based:
        i. Split all data points for a feature into discrete bins
        ii. Uses these bins to find the best split value of histogram
2. LightGBM
    a. Gradient-based One-Side Sampling (GOSS)
        i. Keeps all the instances with large gradients and performs random sampli small gradients
        ii. Training instances with small gradients have smaller training error and it
        iii. To achieve good balance between reducing the number of data instances accuracy for learned decision trees, GOSS introduces a constant multiplie with small gradients
3. *CatBoost*
    a. *Practical implement of orderd boosting*
    b. *Permutating the set of input observations in a random order, multiple random pe generated*
    c. *Converting the label value from a floating point or category to an integer*
    d. *Transform all categorical feature to numeric values using* $avg\_target = \frac{countInC}{totalC}$
        i. ***countInClass*** *is how many times the label value was equal to "1" for object categorical feature value.*
        ii. ***TotalCount*** *is the total number of objects that have a categorical feature v current one*



Algorithm 2: Building a tree in CatB

input : $M, \{y_i\}_{i=1}^{n}, \alpha, L, \{\sigma_i\}_{i=1}^{s},$

$grad \leftarrow CaclGradient(L, M, y);$

$r \leftarrow random(1, s);$

on gain.

ing on the instances with

t is already well-trained.
and keeping the
r for the data instances

ermutations are

$$\frac{Class+prior}{Count+1}$$

s with the current

alue matching the

oost

$Mode$

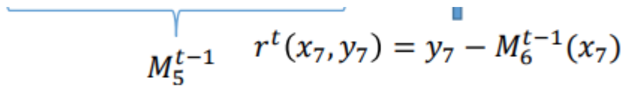


Figure 1: Ordered boosting principle.

Algorithm 1: Ordered boosting

input : $\{(\mathbf{x}_k, y_k)\}_{k=1}^n$, I;

$\sigma \leftarrow$ random permutation of $[1, n]$;
$M_i \leftarrow 0$ for $i = 1..n$;
for $t \leftarrow 1$ **to** I **do**
  for $i \leftarrow 1$ **to** n **do**
    $r_i \leftarrow y_i - M_{\sigma(i)-1}(i)$;
  for $i \leftarrow 1$ **to** n **do**
    $\Delta M \leftarrow$ $LearnModel((\mathbf{x}_j, r_j) : \sigma(j) \leq i)$;
    $M_i \leftarrow M_i + \Delta M$;
return M_n

$r \leftarrow random(1, s)$;
$G \leftarrow (grad_r(1), \ldots, grad_r(n))$ for F
$G \leftarrow (grad_{r,\sigma_r(1)-1}(i)$ for $i = 1$ to n
$T \leftarrow$ empty tree;
foreach *step of top-down procedure* **d**
  foreach *candidate split c* **do**
    $T_c \leftarrow$ add split c to T;
    if $Mode == Plain$ **then**
      $\Delta(i) \leftarrow \text{avg}(grad_r(p)$ for $p : leaf(p) = leaf(i))$
    if $Mode == Ordered$ **then**
      $\Delta(i) \leftarrow \text{avg}(grad_{r,\sigma_r(i)-}$
      $p : leaf(p) = leaf(i), \sigma$
    $loss(T_c) \leftarrow ||\Delta - G||_2$
  $T \leftarrow \text{argmin}_{T_c}(loss(T_c))$
if $Mode == Plain$ **then**
  $M_{r'}(i) \leftarrow M_{r'}(i) - \alpha\, \text{avg}(grad_r$
  $p : leaf(p) = leaf(i))$ for all r'
if $Mode == Ordered$ **then**
  $M_{r',j}(i) \leftarrow M_{r',j}(i) - \alpha\, \text{avg}(gra$
  $p : leaf(p) = leaf(i), \sigma_{r'}(p) \leq$
return T, M

oost

$Mode$

$Plain$;

) for $Ordered$;

lo

.

for all i;

$_1(p)$ for

$_r(p) < \sigma_r(i)) \ \forall i$;

$_{\prime}(p)$ for

$'$, i;

$ad_{r',j}(p)$ for

j for all r', j, i;