Dive into XGBoost

张海鹏

JiangNan University

2017-10-20

目录

- 数学理论
- 系统设计
- 模型应用
- 参考文献

数学理论

$$f(x_{k+1}) = f(x_k + x_{k+1} - x_k) \approx f(x_k) + \nabla f(x_k)(x_{k+1} - x_k)$$

$$f(x_{k+1}) < f(x_k) \Rightarrow \nabla f(x_k)(x_{k+1} - x_k) < 0$$

$$x_{k+1} = x_k - \gamma \nabla f(x_k)$$

$$\nabla f(x_k) = 0, x_{k+1} = x_k$$

数学理论

$$f(x) = l(h(x, D), Y) \qquad h(x, d) = \frac{1}{1 + e^{-xd}}$$

$$l(h(x, D), Y) = \prod_{d \in D, y \in Y} (h(x, d)^{y} (1 - h(x, d)^{1-y}))$$

$$\begin{aligned} l(H(x_{k+1})) &= l(H(x_t) + h(x_{t+1})) \qquad H(x_t) = \sum_{i=1}^{t} h(x_i) \\ &\approx l(H(x_t)) + \nabla l(H(x_t)) h(x_{t+1}) \end{aligned}$$

数学理论

$$H(x_{t+1}) = H(x_t) - \gamma \nabla l(H(x_t)) \quad H(x_t) = \sum_{i=1}^{t} h(x_i)$$

$$h(x_{t+1}) = -\gamma \nabla l(H(x_t))$$

$$h : D \xrightarrow{x} \nabla l(H(x_t, D))$$

数学理论

Model Formalization:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

Training Objective:

$$L(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

数学理论

$$
\begin{aligned}
L^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i{}^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i{}^{(t-1)} + \boxed{f_t(x_i)}) + \sum_{i=1}^{t} \Omega(f_i)
\end{aligned}
$$

数学理论

$$L^{(t)} \simeq \sum_{i=1}^{n}[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \sum_{i=1}^{t} \Omega(f_i)$$

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \qquad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

remove constant terms!

$$\widetilde{L}^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

数学理论

$$q : \mathbb{R}^d \to \{1, 2, ..., T\}$$

$$f_t(x) = w_{q(x)} \longrightarrow \Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$\widetilde{L}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T$$

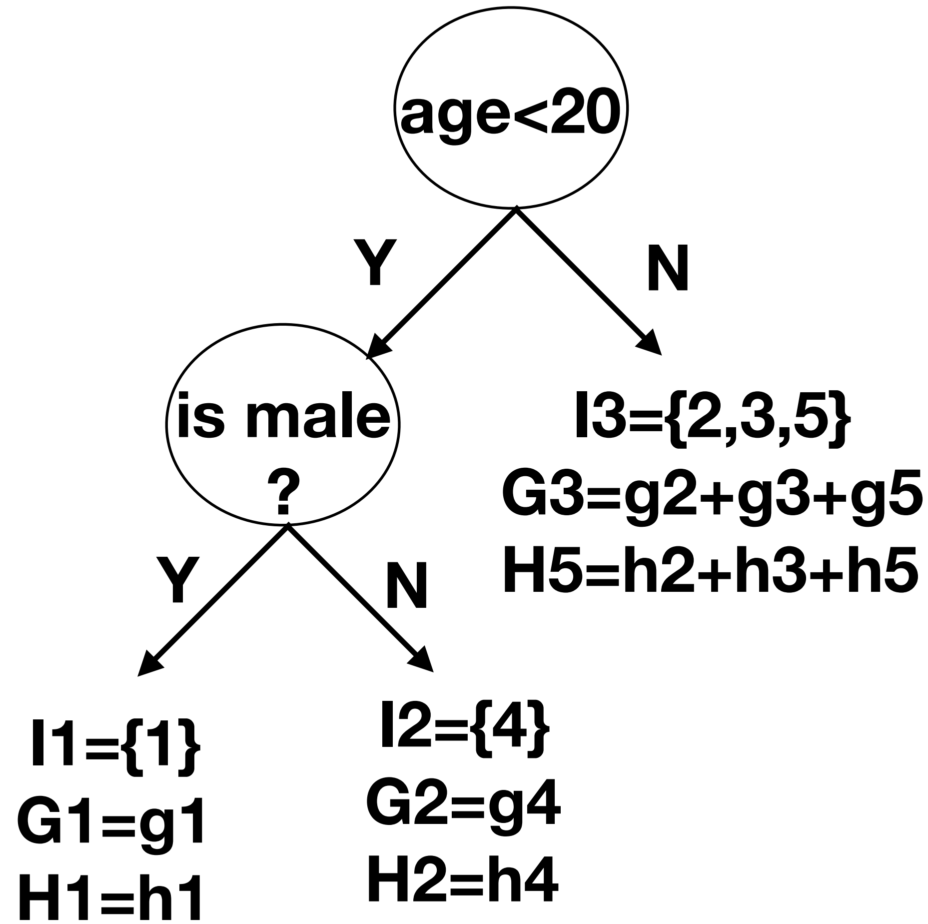
数学理论

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

$$\widetilde{L}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T$$

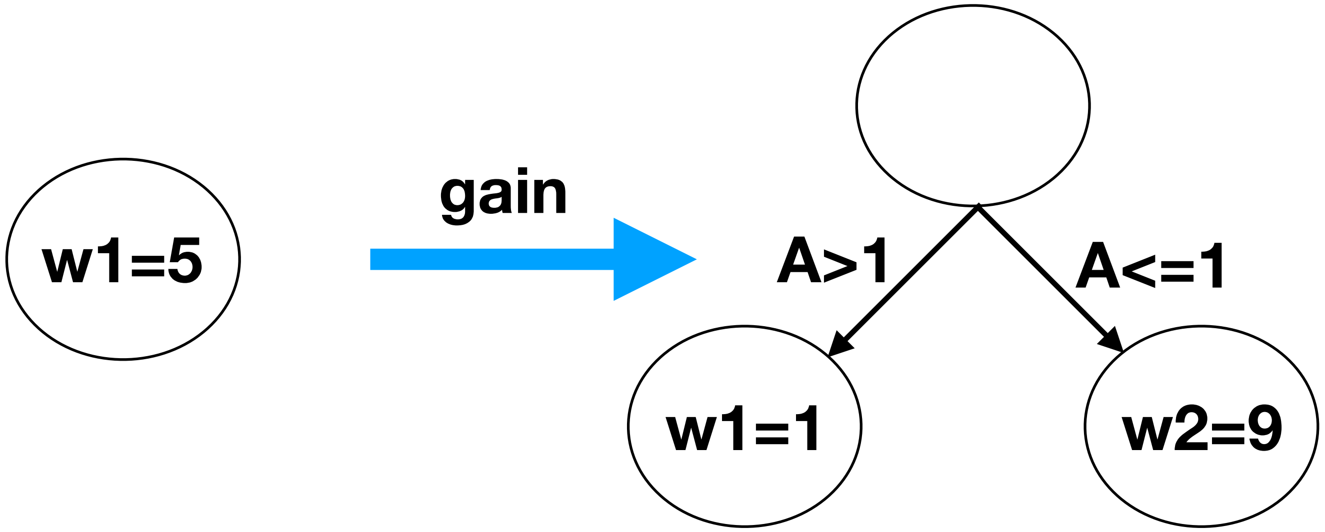
Example

Index	G	H
1	g1	h1
2	g2	h2
3	g3	h3
4	g4	h4
5	g5	h5



$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

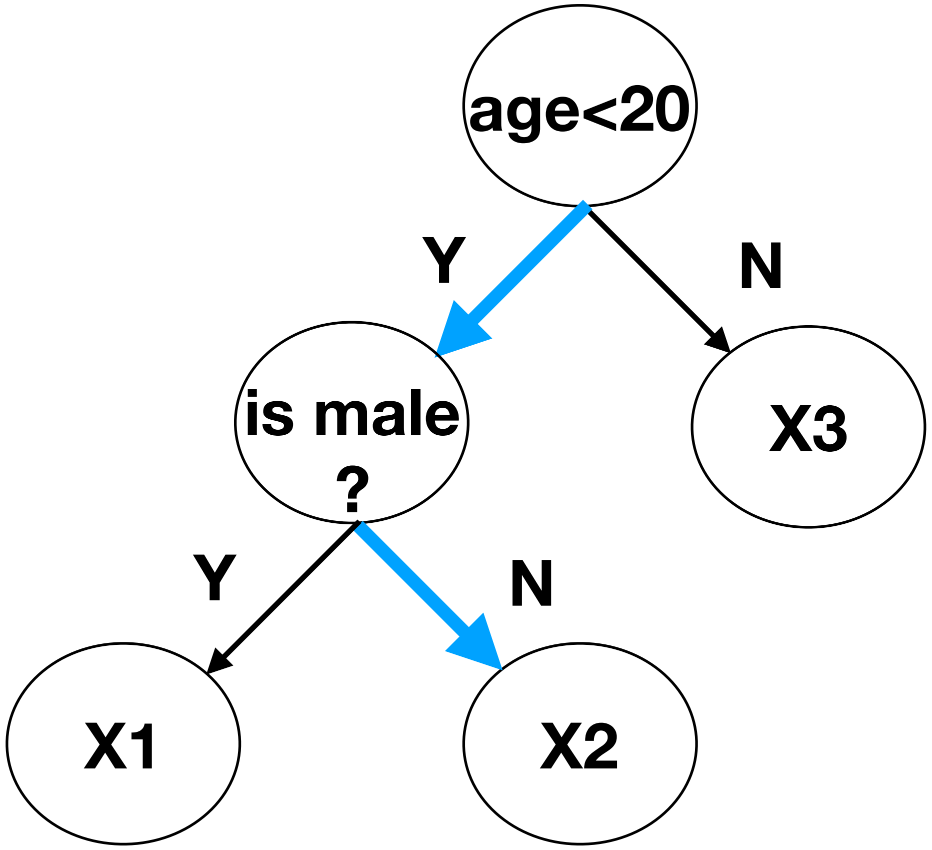
Tree Growing Algorithm



$$gain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

Missing Values Handling

example	Age	Gender
X1	?	male
X2	15	?
X3	25	female



Time Complexity:

$$O(ndKlogn) \xrightarrow{\text{pre-sorted features}} O(ndK)$$

Alg-1

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Alg-2

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ *to* m **do**

  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).

end

for $k = 1$ *to* m **do**

  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$

end

Follow same step as in previous section to find max score only among proposed splits.

Alg-3

Algorithm 3: Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ *to* m **do**
  // *enumerate missing value goto right*
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*(I_k, *ascent order by* $\mathbf{x}_{jk}$) **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // *enumerate missing value goto left*
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*(I_k, *descent order by* $\mathbf{x}_{jk}$) **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

训练过程

初始化模型参数；

设置training sample默认预测值为相同类别，计算grad和hess；

从第一棵树直到设定的树的数目：

grad’, hess’=(grad, hess)*sample_weight；

原始数据集行列采样得到采样后数据集D；

对D进行划分，X=(属性集合)，Y=(真实值，预测值，grad’，hess’)；

建树，返回树模型curTree；

更新预测值(学习率*新的预测值)，grad，hess；

子树添加到根树；打印训练信息；

建树过程

初始化参数，获取训练数据X,Y；

判断当前训练样本是否构成叶子节点，如果是，计算叶子节点score并返回当前TreeNode，内含信息：叶子标志，叶子得分；否则，执行下一步；

对X列采样，得到X_selected;

找到X_selected的最佳属性，最佳属性分割阈值，最佳gain，空值默认分割方向；

如果**最佳gain为负**，构成叶子节点，返回当前TreeNode；

分割X；分别对X的左右子树建树；

属性重要性计数；

返回TreeNode，内含信息：左右子树，分割属性，分割阈值，分割方向；

预测过程

初始化预测值pred为0；

对根树中的每一棵子树：//RandomForest的区别

    //并行加速

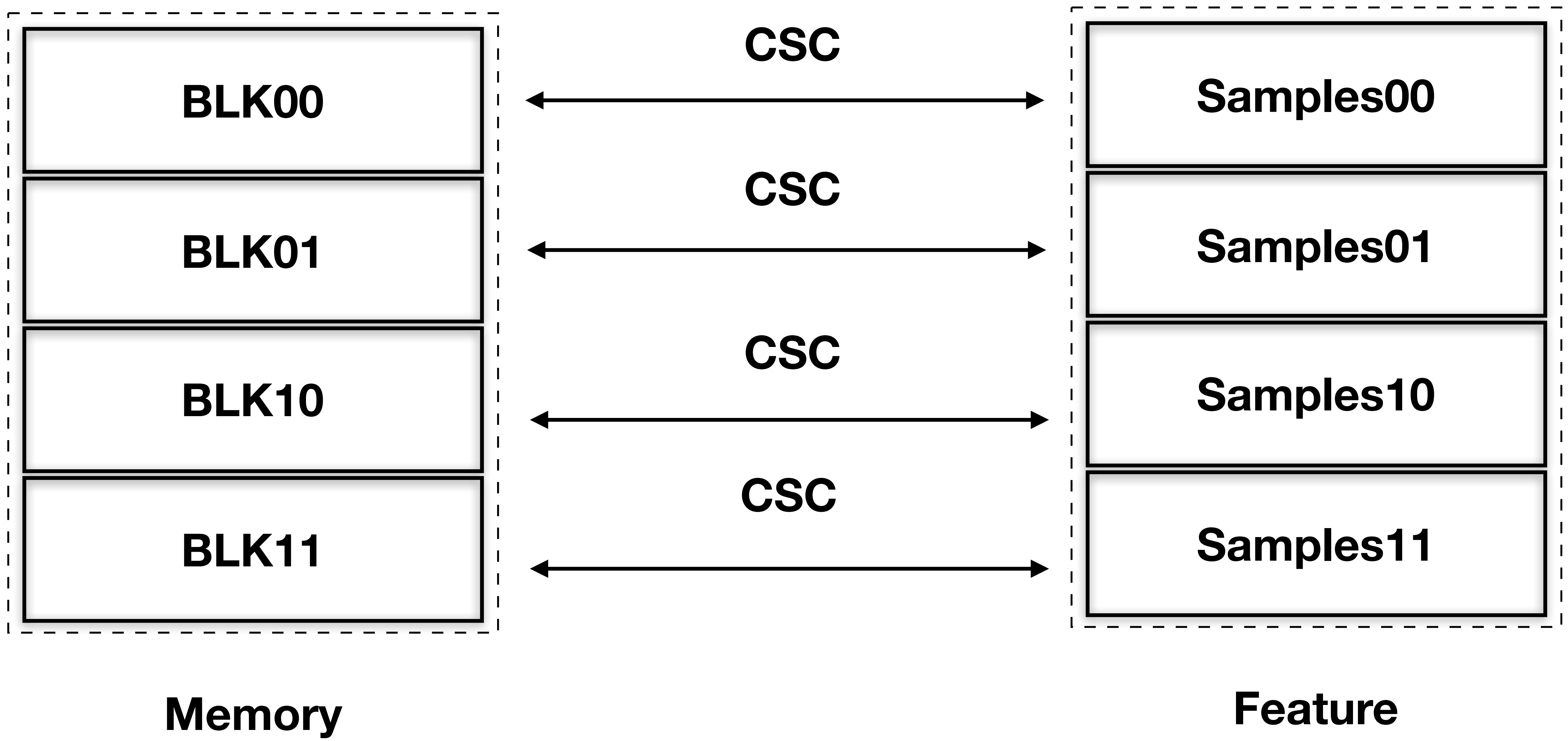
    pred += 学习率*每棵树的预测结果；

返回预测结果pred；

系统设计

- ☐ Column Block for Parallel Learning
  - ☑ exact greedy, approximate
- ☐ Cache-aware Access
  - ☑ exact greedy: cache-aware prefetching
  - ☑ approximate: choose better block size to balance cache property and parallelization
- ☐ Blocks for Out-of-core Computation
  - ☑ block compression, block sharding

Column-Block



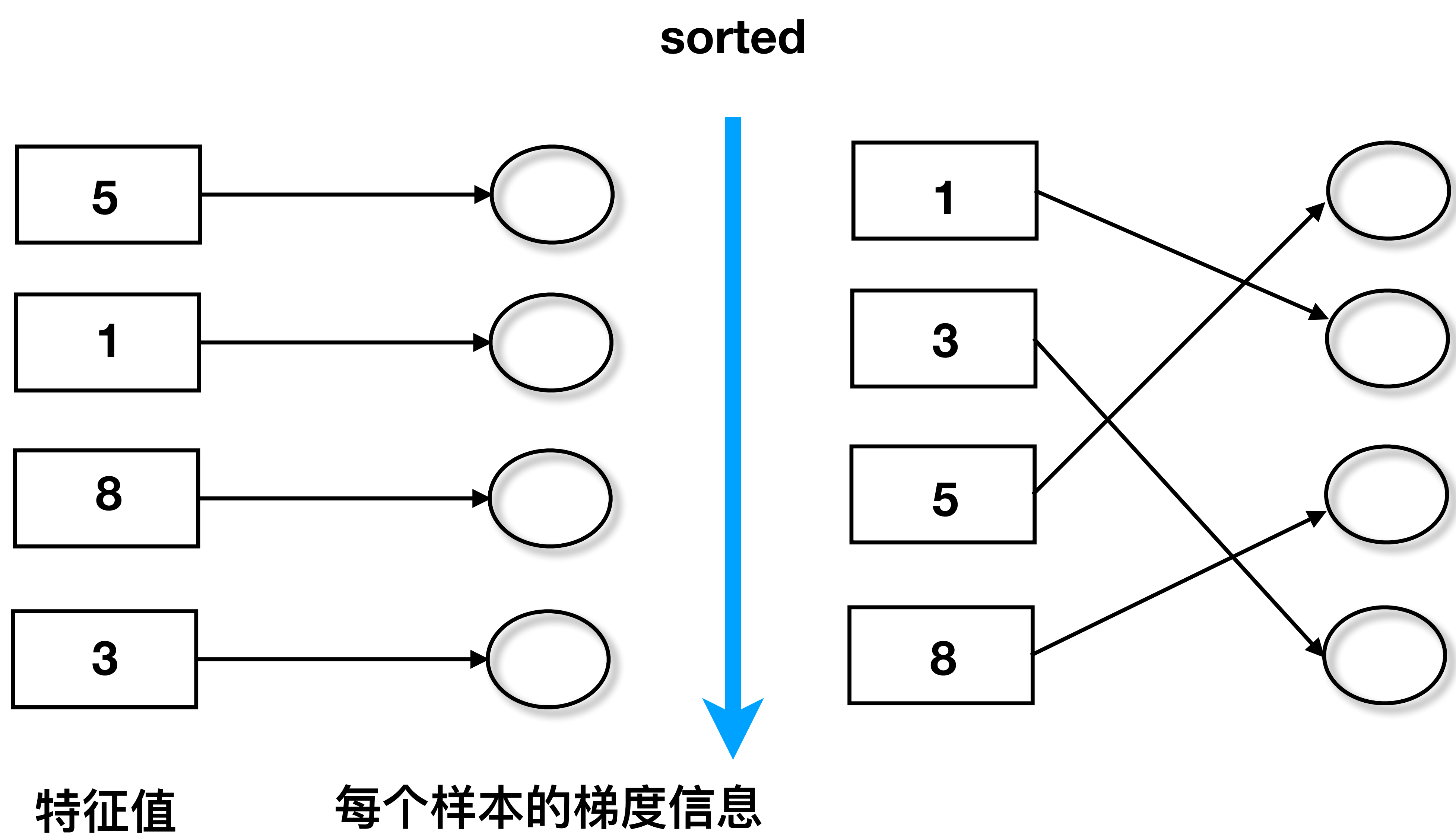
Compressed Row Storage

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

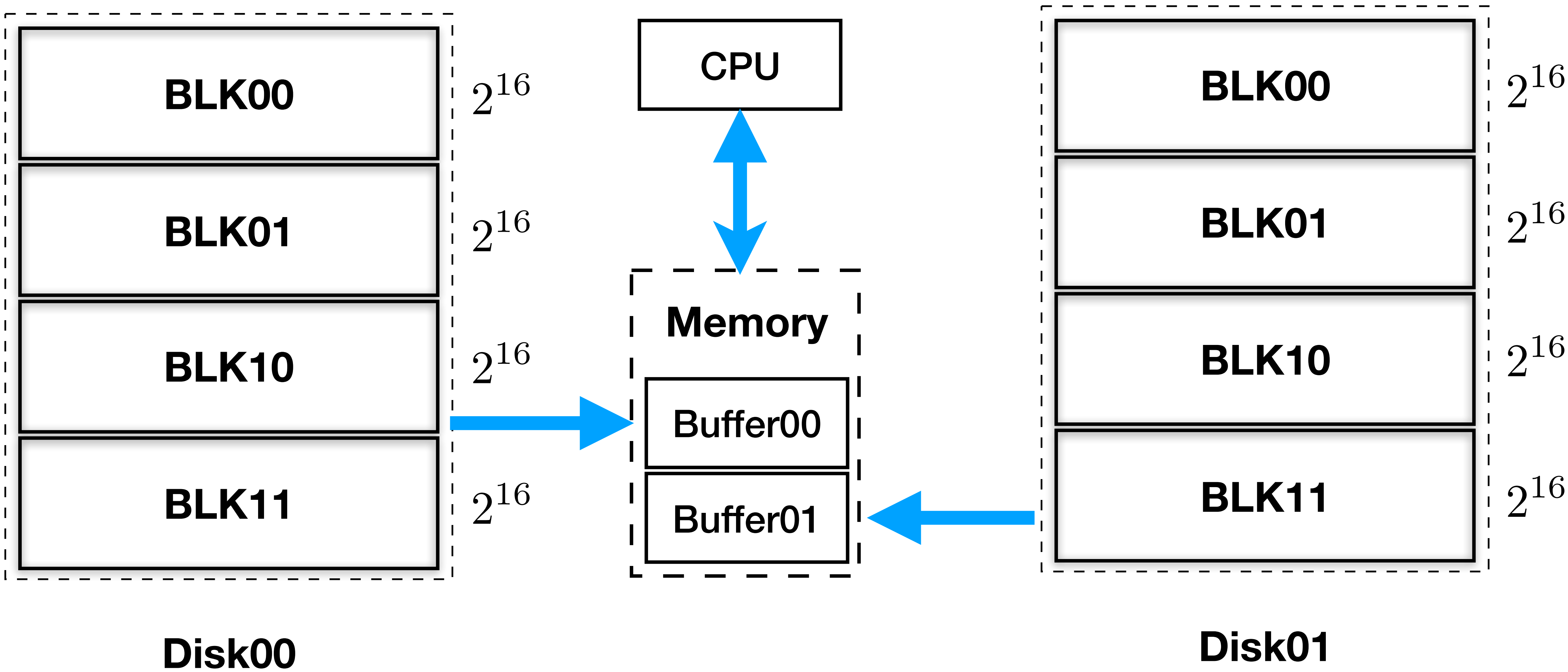
row_ptr	1	3	6	9	13	17	20

val	10	-2	3	9	3	7	8	7	3 · · · 9	13	4	2	-1
col_ind	1	5	1	2	6	2	3	4	1 · · · 5	6	2	5	6

Cache-Aware



Out-of-core Computation



模型应用

- ☑ Feature Importance
- ☑ Customized Metric/Loss Function
- ☑ Combination Features(stacking)

参考文献

1.XGBoost的Github地址: https://github.com/dmlc/xgboost

2.XGBoost的tiny实现：https://github.com/zhpmatrix/groot

3.GBDT的实现：https://github.com/liuzhiqiangruc/dml/tree/master/gbdt

4.《Higgs Boson Discovery with Boosted Trees》, JMLR Workshop，Tianqi Chen, Tong He, 2015

5.《XGBoost: A Scalable Tree Boosting System》, KDD, Tianqi Chen, Carlos Guestrin, 2016

6.《Practical Lessons from Predicting Clicks on Ads at Facebook》, Xinran He

7.XGBoost源码阅读：https://zhpmatrix.github.io/2017/03/15/xgboost-src-reading-2

TKS

有啥问题需要探讨的吗?