An Introduction to GBDT & XGBoost

wangfei 2015-07-17

- Gradient Boosting Decision Tree (GBDT)
- Gradient Boosting Model (GBM)
- Multiple Additive Regression Tree (MART)
- TreeNet

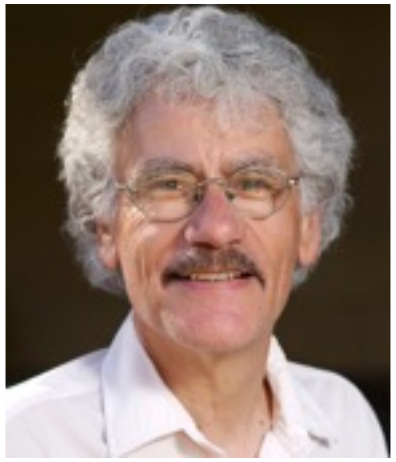
Summary

- What's the problem?
- How to solve it
- An user gender prediction example
- Xgboost's solution

A Little History

- 1984, Breiman et al. CART


- 1996, Freund and Schapire
  AdaBoost
- 2000, Friedman et al.
  boosting as minimization exponential error



- 2001, Friedman et al. gradient boosting machine

What’s the Problem?

- We know how to growth trees (1984, CART)
- Trees can be combined to solve classification problem well (1996, 2000, Adaboost)
- To solve general supervised problem well: boosting + tree (2001, GBM)

Tree Model

- We love to growth trees:
  - somewhat interpretable
  - feature selection builtin
  - invariant under (strictly monotone) transformations
  - fast to train
- But it’s inaccurate…
- Let’s fix it.

A Peek at AdaBoost

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Forward Stagewise Additive Modeling

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute
   $$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$
   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

$$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} \exp[-y_i(f_{m-1}(x_i) + \beta\, G(x_i))]$$

Examples

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left[y_i - \left(\hat{y}_i^{(t-1)} + f_t(x_i) \right) \right]^2 \\
&= \sum_{i=1}^{n} \left[\left(y_i - \hat{y}_i^{(t-1)} \right) - f_t(x_i) \right]^2
\end{aligned}
$$

Steepest Gradient Descent

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i) = f_{m-1}(x_i)}$$

$$\rho_m = \arg\min_{\rho} L(\mathbf{f}_{m-1} - \rho \mathbf{g}_m)$$

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_m \mathbf{g}_m$$

Gradient Boosting

$$\tilde{\Theta}_m = \arg\min_{\Theta} \sum_{i=1}^{N} \left(-g_{im} - T(x_i; \Theta)\right)^2$$

Gradient Boosting Tree Algo

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

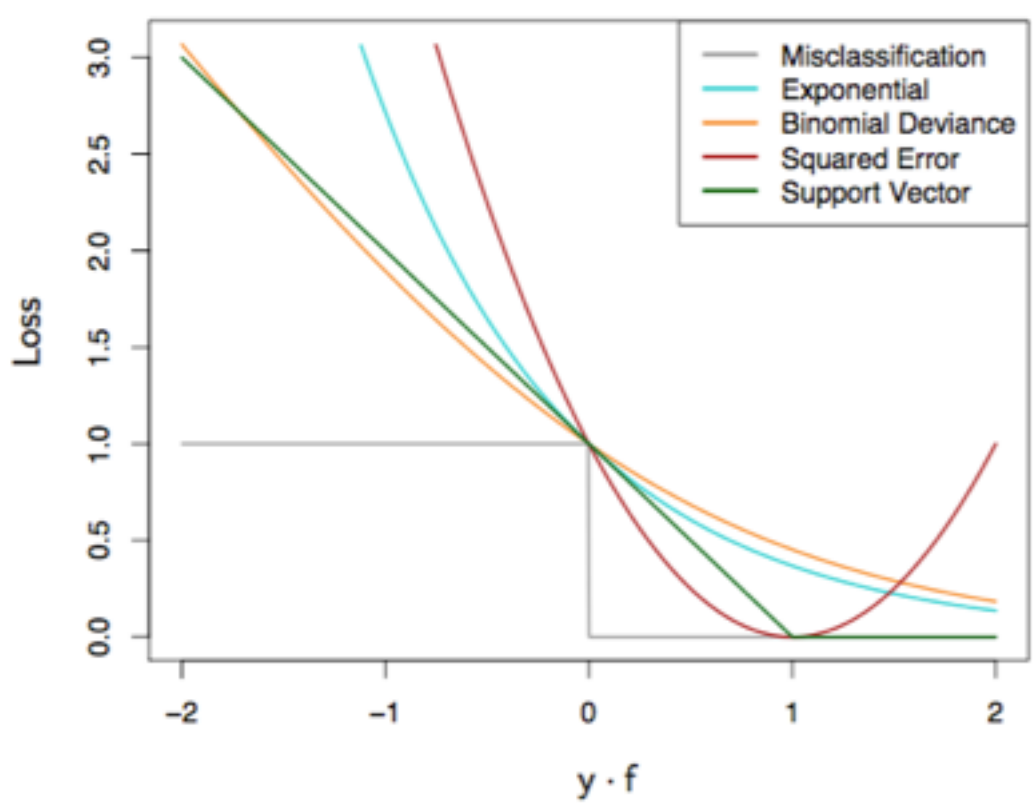
   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

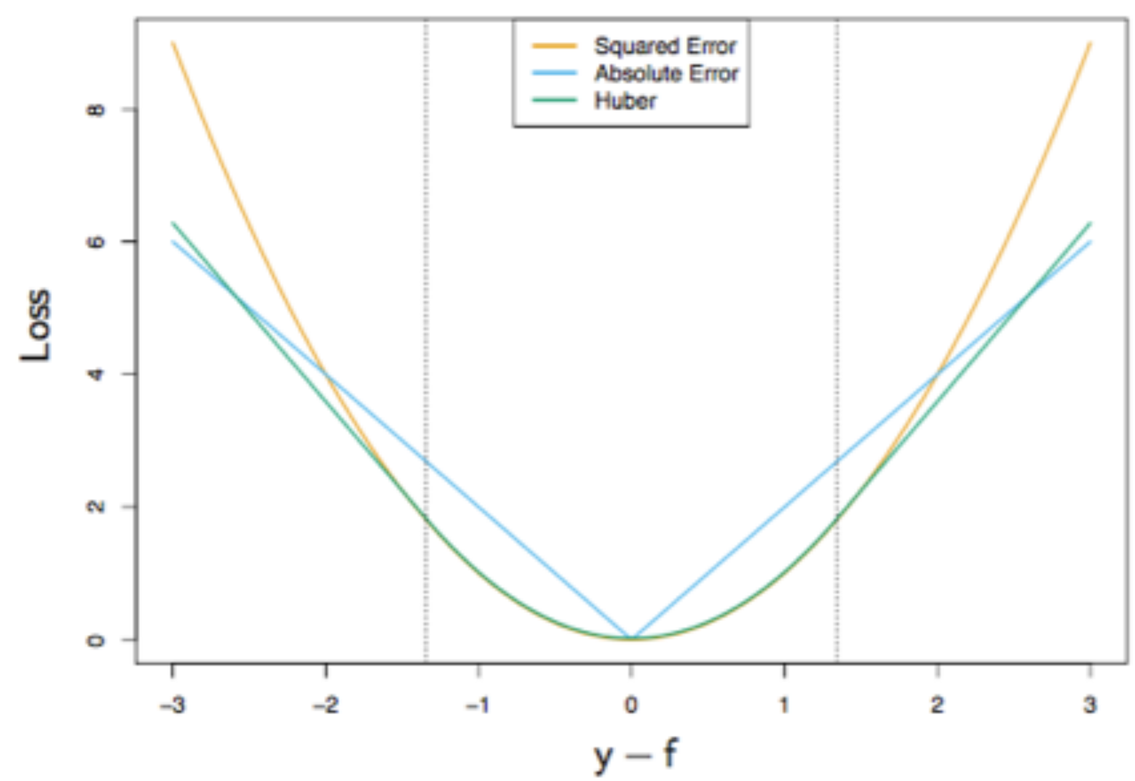
   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Then We Can Do





Why Tree?

- high order interaction

Regularization

- Tree size
- Shrinkage $$f_m(x) = f_{m-1}(x) + \nu \cdot \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm}).$$
- Subsampling

Squared Relevance

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 \, I(v(t) = \ell)$$

$$\mathcal{I}_\ell^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_\ell^2(T_m)$$

Let’s Predict User’s Gender

- Candidate user: 205W
  - at least “rated” 10 movies
- User with gender: 5.5W (2.7%)

Features

```
0 24138:1 7892:1 59141:1 77344:1 38242:1 70871:1 50898:1 36490:1 17623:1 6224:1 6204:1 24664:1 117252:1 35247:1 102030:1 87637:1 86
841:1 64991:1 22362:1 71879:1 23608:1 99369:1 114976:1 115829:1 96857:1 118737:1 8635:1 99861:1 3506:1 95318:1 43771:1 119943:1 538
28:1 32473:1 64490:1 13827:1 109631:1 115714:1 45390:1 61540:1 1958:1 113562:1 78255:1 61437:1 42807:1 78066:1 106786:1 25779:1 662
93:1 78234:1 43263:1 121721:1 50312:1 58719:1 13708:1 83253:1 80470:1 32124:1 26356:1 121865:1 46113:1 94926:1 40224:1 30640:1 2713
6:1 63234:1 37090:1 113451:1 79344:1 6916:1 90938:1 77534:1 47910:1 2265:1 82032:1 63515:1 30828:1 35914:1 69587:1 78513:1 122346:1
 21415:1 82813:1 104133:1 63564:1 115264:1 50637:1 62922:1 118183:1 31384:1 120032:1 101359:1 90475:1 87534:1 56343:1 66132:1 12346
1:1 81811:1 9583:1 112810:1 69429:1 42944:1 85142:1 80949:1 117440:1 32594:1 9277:1 38325:1 32964:1 107414:1 57717:1 28797:1 110965
:1 112013:1 6271:1 43772:1 18861:1 10200:1 55187:1 45275:1 64757:1 26218:1 32050:1 11998:1 113443:1 59937:1 49295:1 56215:1 60514:1
 100767:1 3880:1 53643:1 123338:1 58356:1 50260:1 29561:1 6207:1 40139:1 105922:1 5910:1 25004:1 107624:1 113891:1 2859:1 36623:1 2
5720:1 87993:1 100596:1 77419:1 87523:1 8847:1
```

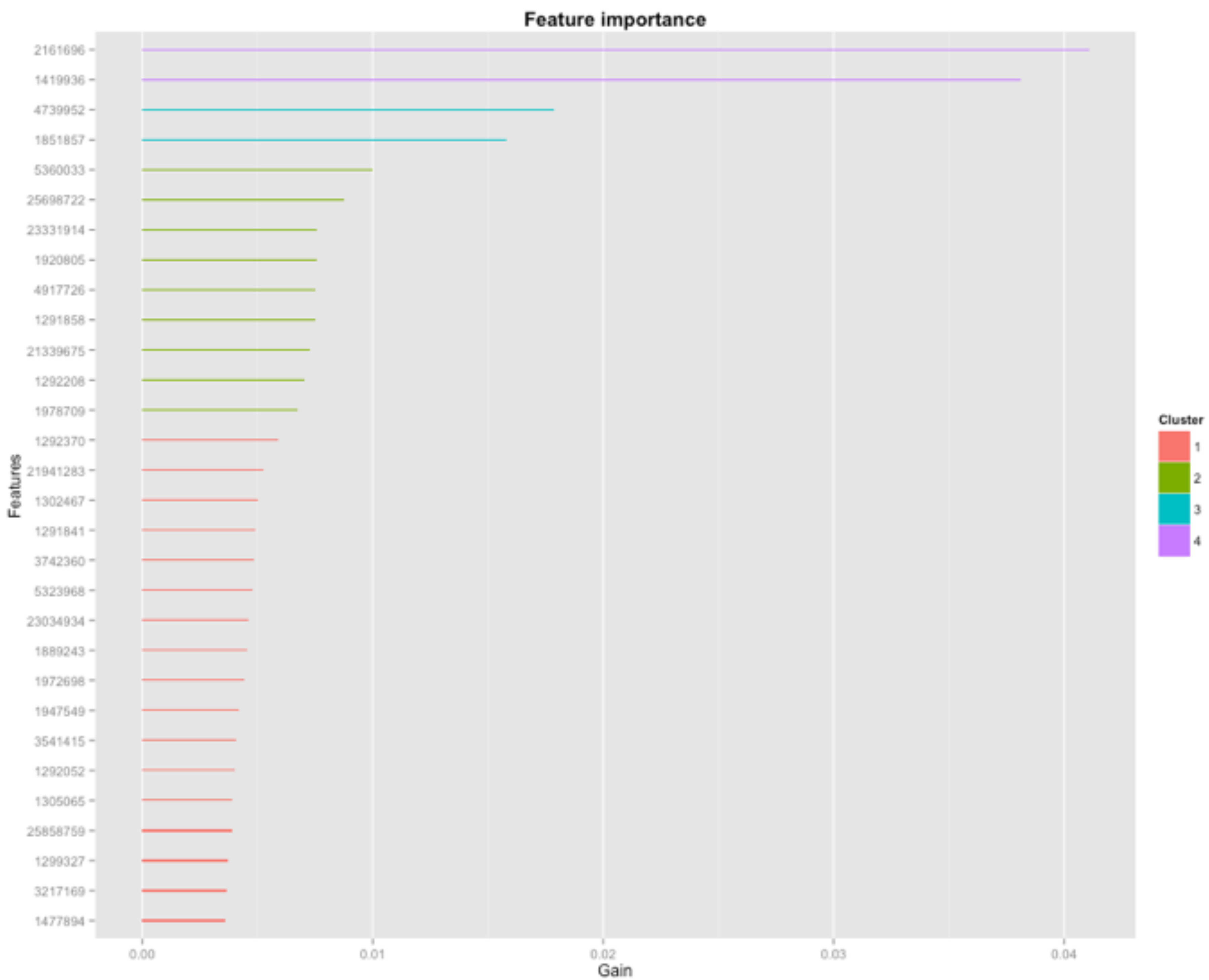
- #feature: 12W

Results

```
49899x123785 matrix with 15350150 entries is loaded from /home2/alg/gender/data/movie/train.csv
5545x123784 matrix with 1677481 entries is loaded from /home2/alg/gender/data/movie/valid.csv
boosting round 0, 0 sec elapsed
tree prunning end, 1 roots, 276 extra nodes, 0 pruned nodes ,max_depth=8
[0]     valid_data-auc:0.760207 valid_data-error:0.313616       train-auc:0.765531      train-error:0.306118
boosting round 1, 0 sec elapsed
tree prunning end, 1 roots, 202 extra nodes, 0 pruned nodes ,max_depth=8
[1]     valid_data-auc:0.802058 valid_data-error:0.283318       train-auc:0.812636      train-error:0.277200
boosting round 2, 0 sec elapsed
tree prunning end, 1 roots, 172 extra nodes, 0 pruned nodes ,max_depth=8
[2]     valid_data-auc:0.822431 valid_data-error:0.270875       train-auc:0.836732      train-error:0.259043
boosting round 3, 0 sec elapsed
tree prunning end, 1 roots, 132 extra nodes, 0 pruned nodes ,max_depth=8
[3]     valid_data-auc:0.840676 valid_data-error:0.253201       train-auc:0.853706      train-error:0.239323
```

```
[997]   valid_data-auc:0.991235 valid_data-error:0.022904       train-auc:0.999725      train-error:0.006954
boosting round 998, 172 sec elapsed
tree prunning end, 1 roots, 48 extra nodes, 0 pruned nodes ,max_depth=8
[998]   valid_data-auc:0.991237 valid_data-error:0.022723       train-auc:0.999728      train-error:0.006894
boosting round 999, 172 sec elapsed
tree prunning end, 1 roots, 28 extra nodes, 0 pruned nodes ,max_depth=8
[999]   valid_data-auc:0.991203 valid_data-error:0.022904       train-auc:0.999729      train-error:0.006834
```

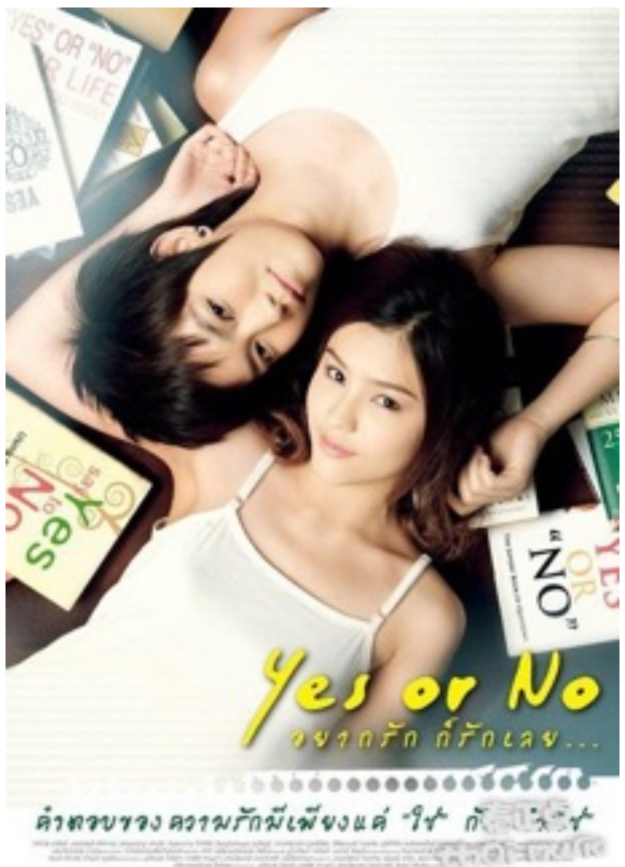
Interpretation



Top - 3

#M	#F	#M / SUM
1299	2741	0.3215
1970	834	0.7026
1538	2643	0.3679

Female Movies



#M	#F	#M / SUM
242	1031	0.1901
368	1210	0.2332
774	1797	0.3011

Male Movies

#M	#F	#M / SUM
2569	1470	0.6360
1170	538	0.6850
1755	680	0.7207

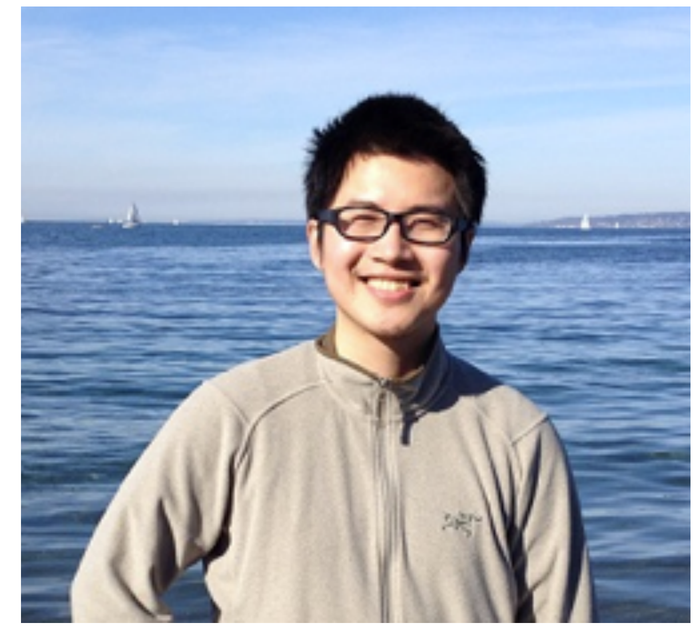
Not So Obvious

#M	#F	#M / #F
3923	3808	0.5074

Gender Project

- repo: http://code.dapps.douban.com/gender
- prediction: /home2/alg/gender/gender.csv
  - 550W
  - movie, music, book, fm, group

XGBoost



- https://github.com/dmlc/xgboost
- Win a lot of kaggle competitions
- Features:
  - With python wrapper (also R, Julia)
  - Support external memory
  - Distributed with Hadoop (YARN), MPI…
  - Dump & load model (plain txt or binary)
  - …

Additive Training: A Second Look

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right)$$

$$Obj^{(t)} = \sum_{i=1}^{n} \left[y_i - \left(\hat{y}_i^{(t-1)} + f_t(x_i) \right) \right]^2$$

$$= \sum_{i=1}^{n} \left[\left(y_i - \hat{y}_i^{(t-1)} \right) - f_t(x_i) \right]^2$$

$$= \sum_{i=1}^{n} \left[\left(y_i - \hat{y}_i^{(t-1)} \right)^2 - 2 \cdot \left(y_i - \hat{y}^{(t-1)} \right) \cdot f_t(x_i) \right.$$

$$\left. + f_t^2(x_i) \right]$$

Taylor Expansion Approximation

$$f(x+\Delta x) \simeq f(x) + f'(x) \cdot \Delta x + \frac{1}{2} f''(x) \Delta x^2$$

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

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

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

New Objective

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right]$$

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

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

Revisit the Objective

$$f_t(x) = w_{q(x)}, \quad w \in R^T, \quad q : R^d \rightarrow \{1, 2, \cdots T\}$$

$$I_j = \{ i \mid q(x_i) = j \}$$

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n} \left[g_i \cdot f_t(x_i) + \frac{1}{2} h_i \cdot f_t^2(x_i) \right] \\ &= \sum_{i=1}^{n} \left[g_i \cdot w_{q(x_i)} + \frac{1}{2} h_i \cdot w_{q(x_i)}^2 \right] \\ &= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) \cdot w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i \right) \cdot w_j^2 \right] \end{aligned}$$

The Structure Score

$$G_j = \sum_{i \in I_j} g_i$$

$$H_j = \sum_{i \in I_j} h_i$$

$$Obj^{(t)} = \sum_{j=1}^{T} \left[G_j \cdot w_j + \frac{1}{2} H_j \cdot w_j^2 \right]$$

$$\arg\min_x \; Gx + \frac{1}{2} H x^2, \quad H > 0$$

$$x = -\frac{G}{H}$$

$$\min_x \; Gx + \frac{1}{2} H x^2 = -\frac{1}{2} \cdot \frac{G^2}{H}$$

Let's Growth Trees

$$\text{Gain} = \frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{(G_L + G_R)^2}{H_L + H_R}$$

$$O(\underset{\text{level}}{\underset{\downarrow}{K}} \cdot \underset{\text{feature}}{\underset{\downarrow}{d}} \cdot \underset{\text{Sorting}}{\underset{\downarrow}{n \log n}})$$

See Some Code

```
inline void UpdateOneIter(int iter, const DMatrix &train) {
  if (seed_per_iteration != 0 || rabit::IsDistributed()) {
    random::Seed(this->seed * kRandSeedMagic + iter);
  }
  this->PredictRaw(train, &preds_);
  obj_->GetGradient(preds_, train.info, iter, &gpair_);
  gbm_->DoBoost(train.fmat(), this->FindBufferOffset(train), train.info.info, &gpair_);
}
```

```
virtual void Update(const std::vector<bst_gpair> &gpair,
                    IFMatrix *p_fmat,
                    const BoosterInfo &info,
                    RegTree *p_tree) {
  this->InitData(gpair, *p_fmat, info.root_index, *p_tree);
  this->InitNewNode(qexpand_, gpair, *p_fmat, info, *p_tree);
  for (int depth = 0; depth < param.max_depth; ++depth) {
    this->FindSplit(depth, qexpand_, gpair, p_fmat, info, p_tree);
    this->ResetPosition(qexpand_, p_fmat, *p_tree);
    this->UpdateQueueExpand(*p_tree, &qexpand_);
    this->InitNewNode(qexpand_, gpair, *p_fmat, info, *p_tree);
    // if nothing left to be expand, break
    if (qexpand_.size() == 0) break;
  }
```

Gradient Boosting
is
Gradient Descent in Function Space

Thank you + Q&A

Reference

- Introduction to Boosted Trees by tianqi chen
- The Elements of Statistical Learning, Chapter 9
- Greedy function approximation a gradient boosting machine. J.H. Friedman 1999
- Boosting Algorithms as Gradient Descent in Function Space. Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (May 1999).