Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

GBDT
梯度提升决策树

博士研究生　刘晓双

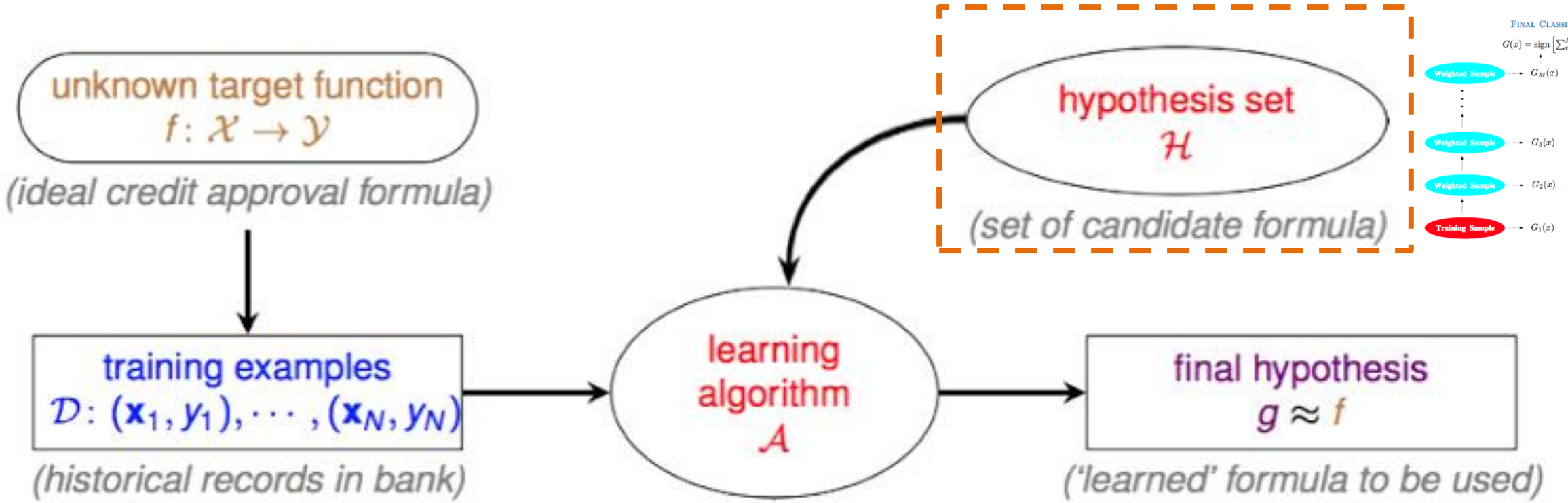
2018年05月06日

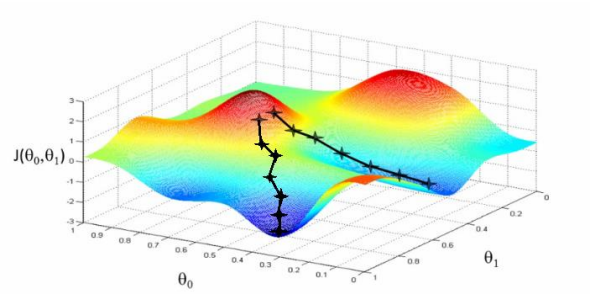
内容提要

- 背景简介
- 算法原理
  - GBDT
  - XGBoost
- 优劣分析
- 应用总结
- 参考文献

背景介绍

- 机器学习基础架构[1]





[1] 林轩田，机器学习基石

背景介绍

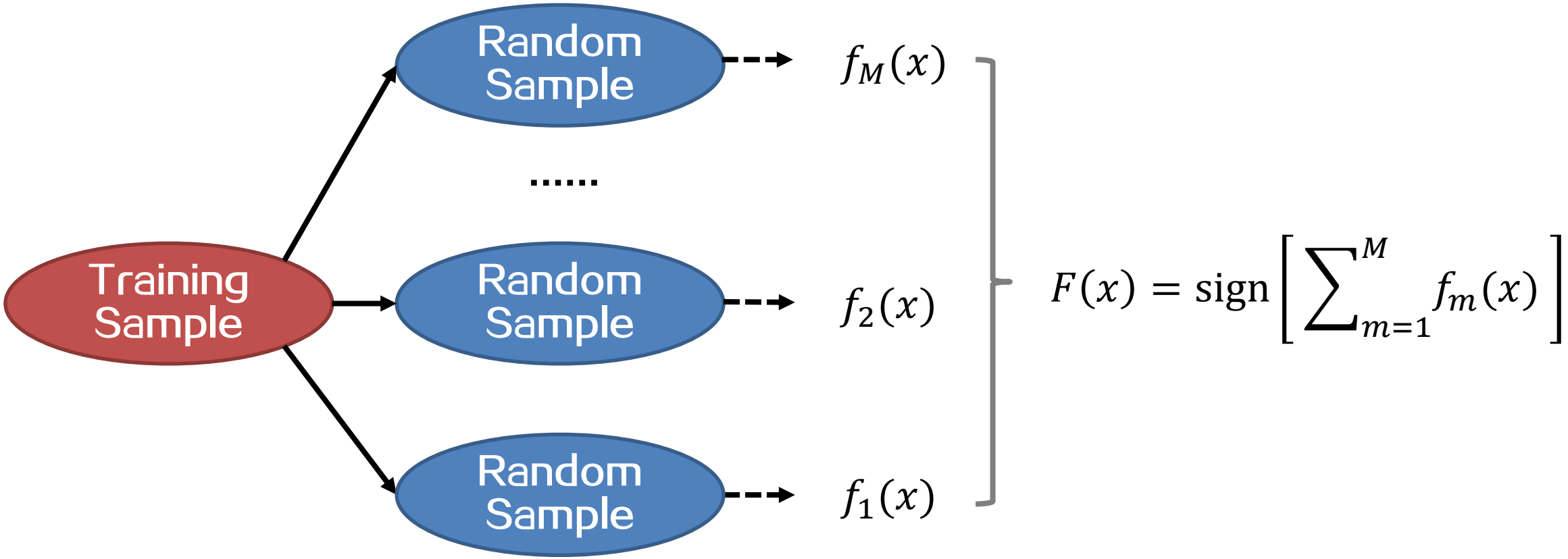
- 集成学习 (Ensemble Learning)
  - 在统计学和机器学习领域中，集成学习通过组合多个（种）学习算法以提高模型性能（相比于单一学习算法）

  - 代表算法
    - Bayesian家族（搜索最佳组合）
    - Bucket of models（模型库）
    - Bagging
    - Boosting

Wikipedia，Ensemble learning

背景介绍

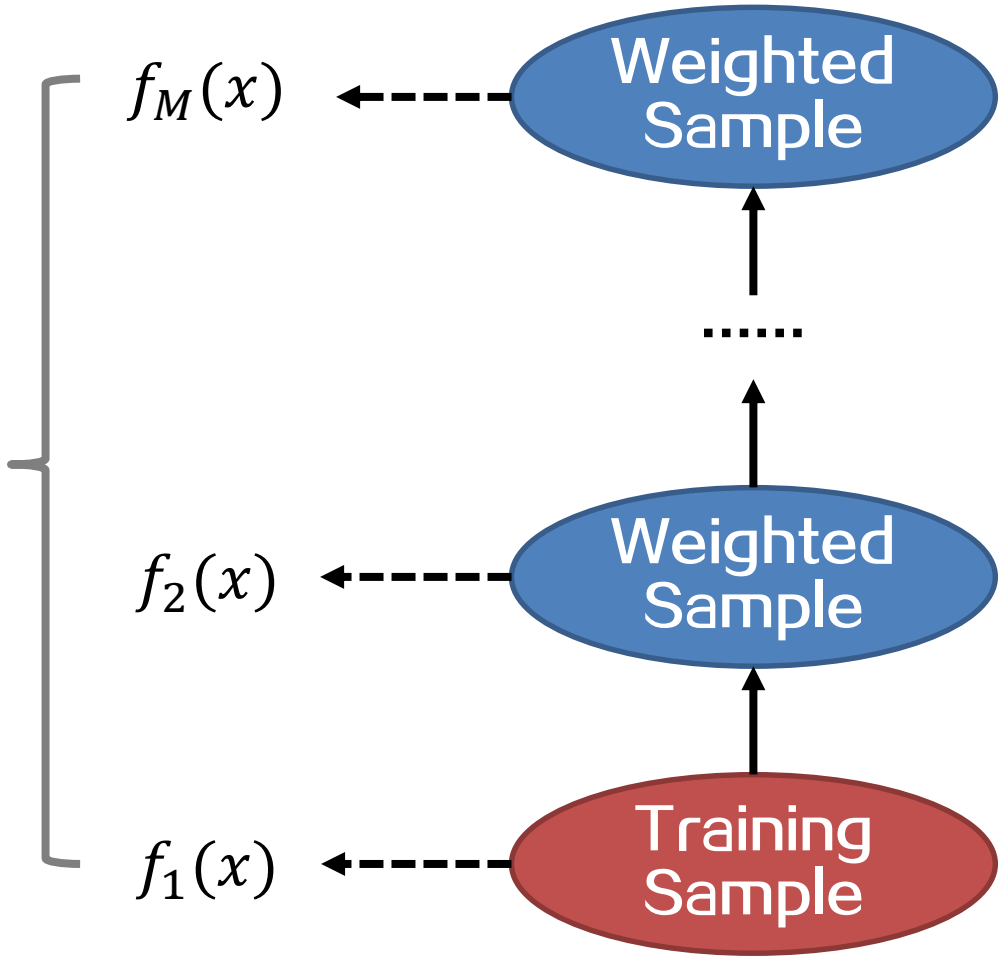
- Bagging



背景介绍

- Boosting
  - 增量训练新模型，每个新模型更加关注于目前为止容易被误判的样本，即提高样本在损失函数中的权重

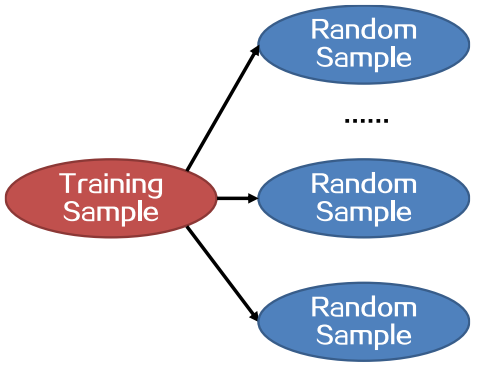
$$F(x) = \text{sign}\left[\sum_{m=1}^{M} \beta_m f_m(x)\right]$$

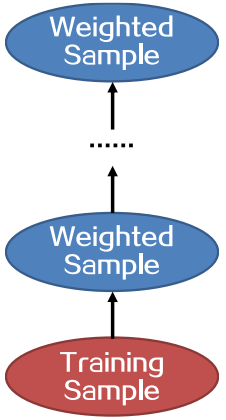


背景介绍

- 简要对比

	Bagging	Boosting
训练过程	可并行	不可并行
样本选择	有放回抽样	不变化
样本权重	相等	变化
子模型权重	一般相等	一般不等
测试过程	可并行	可并行
核心原理	减少方差	减少偏差
其他特性	异常值不敏感	异常值敏感

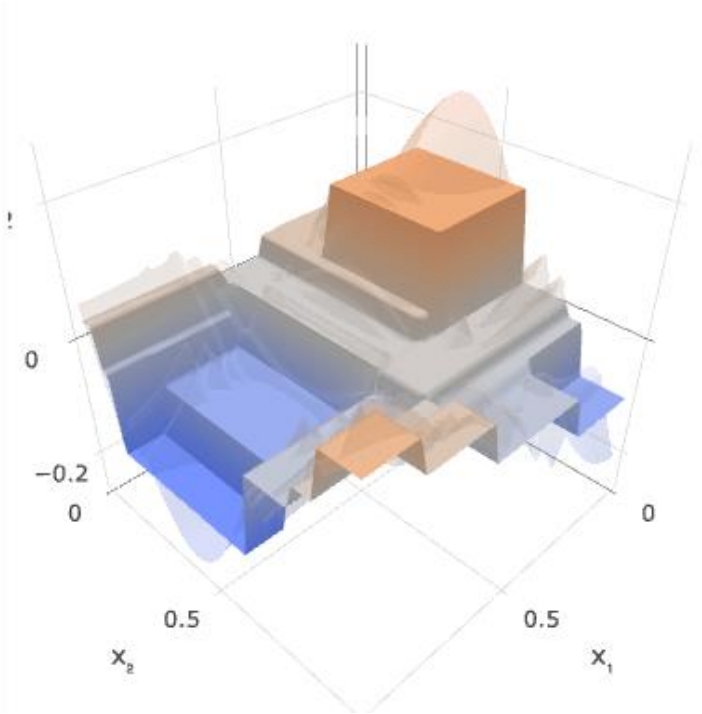




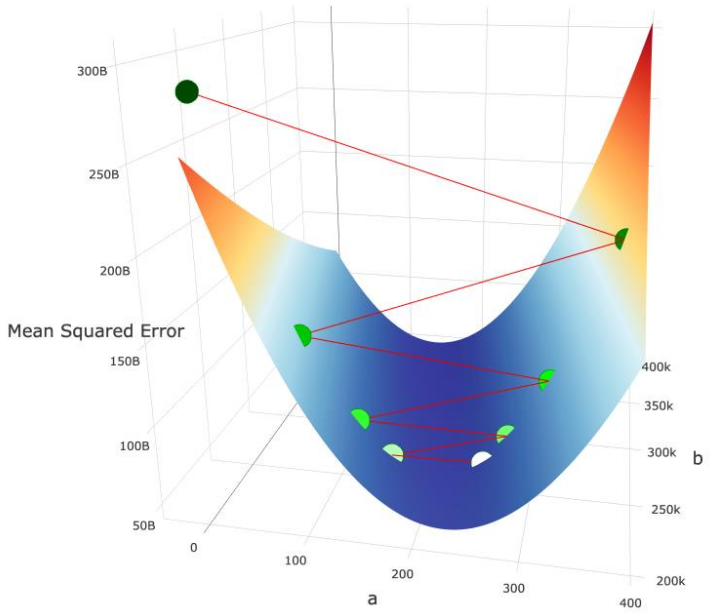
算法原理

算法原理 GBDT

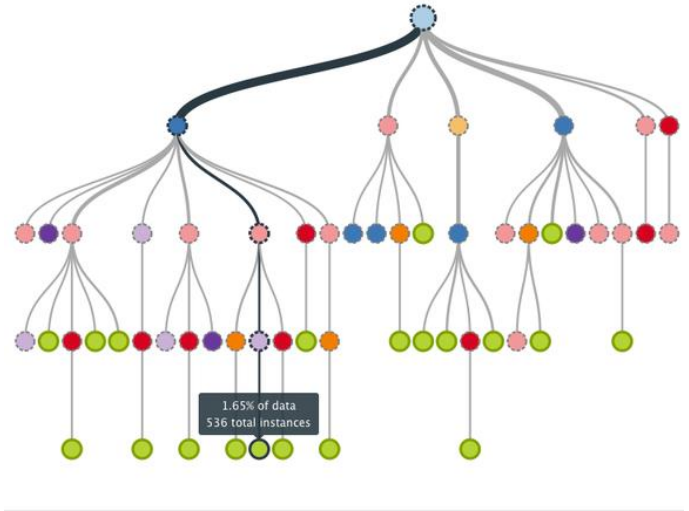
- GBDT
  - Gradient Boost (梯度提升) + Decision Tree (决策树)



=



+

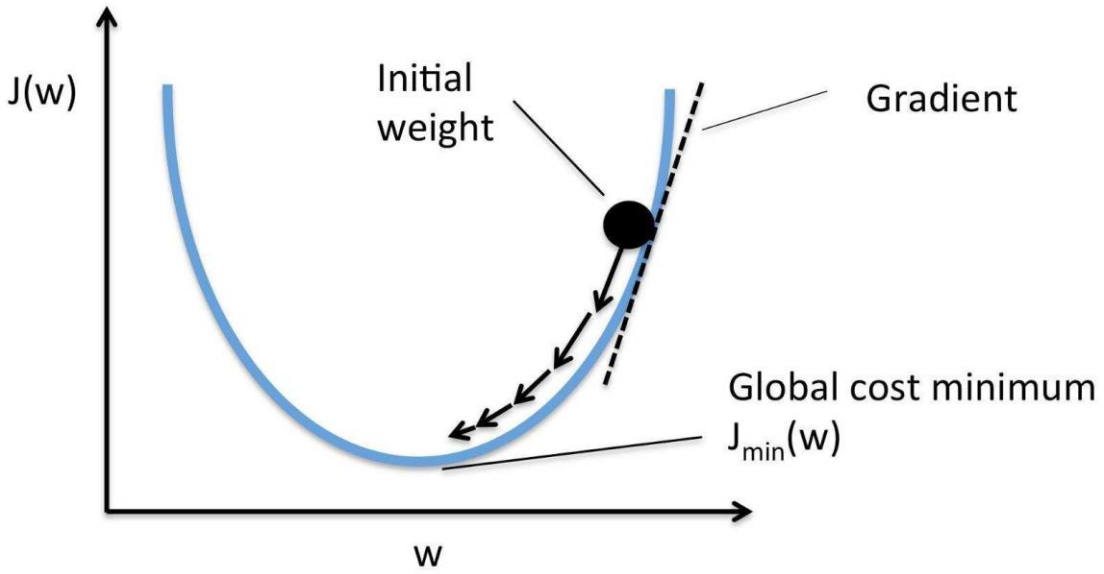


算法原理 Gradient Boost

- 梯度下降
  - 数据

特征	标签
$x_1 = 1$	$y_1 = 1$
$x_2 = 2$	$y_2 = 2$

  - 模型
    - $y = \boldsymbol{w}^T \boldsymbol{x}$ $\quad \boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \ \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$
    - d = 1，即 $y = wx$
  - 目标函数 / 损失函数
    - 均方误差 $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
    - 损失函数 $J = \frac{1}{n}\sum_{i=1}^{n}(y_i - wx_i)^2$



算法原理 Gradient Boost

- 优化过程
  - 随机初始化 $w^0 = 2$
  - 设置学习率 $\alpha = 0.1$
  - 迭代更新至收敛
    - 例如 w 变化小于 0.1

特征	标签
$x_1 = 1$	$y_1 = 1$
$x_2 = 2$	$y_2 = 2$

$$y = wx \qquad w\ ?$$

$$w^{t+1} = w^t - \alpha\left(\frac{\partial J}{\partial w}\right)^t \qquad \frac{\partial J}{\partial w} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - wx_i)x_i$$

$$w^* = \underset{w}{\operatorname{argmin}} J = w^0 + [-\alpha(\partial J/\partial w)]^1 + \cdots + [-\alpha(\partial J/\partial w)]^T$$

算法原理 Gradient Boost

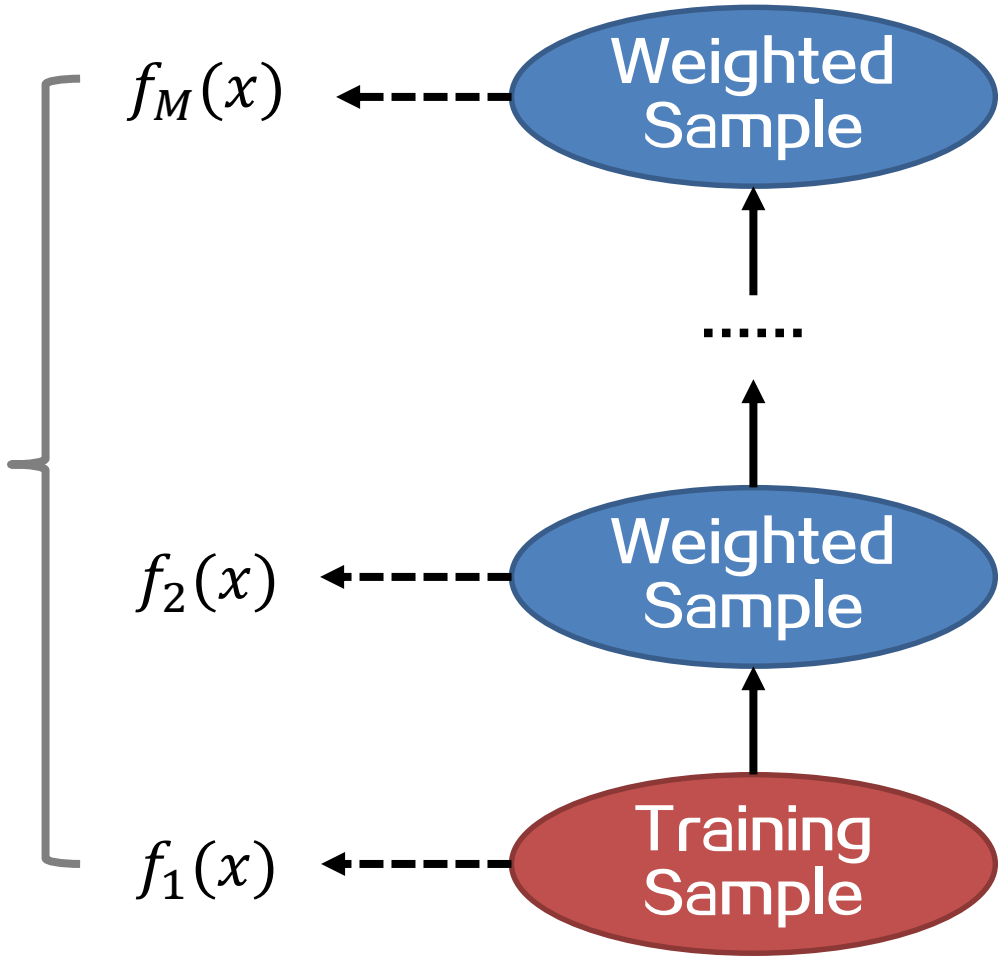
【核心页】

- 梯度提升（Gradient Boost）
  - 单模型损失：$J = \frac{1}{n}\sum_{i=1}^{n}(y_i - wx_i)^2$ → $E_{y,x}\theta[y, f(x; w)]$
  - Boosting：$E_{y,x}\theta[y, F(x; \boldsymbol{P})]$ 其中，$F(x) = \sum_{m=1}^{M}\beta_m f_m(x)$， $\boldsymbol{P} = \{\beta_m, w_m\}_0^M$

算法原理 Gradient Boost

- 回顾 Boosting
  - 增量训练新模型，每个新模型更加关注于目前为止容易被误判的样本，即提高样本在损失函数中的权重

$$F(x) = \text{sign}\left[\sum_{m=1}^{M} \beta_m f_m(x)\right]$$



算法原理 Gradient Boost

【核心页】

- 梯度提升（Gradient Boost）
  - 单模型损失：$J = \frac{1}{n}\sum_{i=1}^{n}(y_i - wx_i)^2$ → $E_{y,x}\theta[y, f(x;w)]$
  - Boosting：$E_{y,x}\theta[y, F(x;\boldsymbol{P})]$ 其中，$F(x) = \sum_{m=1}^{M}\alpha_m f_m(x)$，$\boldsymbol{P} = \{\alpha_m, w_m\}_0^M$
  - 此时，若将 $F(x;\boldsymbol{P})$ 看做一个整体 → 因吹斯听的事情就来了
  - $w^* = \underset{w}{\operatorname{argmin}} J = w^0 + [-\alpha(\partial J/\partial w)]^1 + \cdots + [-\alpha(\partial J/\partial w)]^T$
  - $F^* = \underset{F}{\operatorname{argmin}} J = F^0 + [-\alpha(\partial J/\partial F)]^1 + \cdots + [-\alpha(\partial J/\partial F)]^T$
  - $F^* = F^0 + \alpha g_1 + \cdots + \alpha g_M$，$g_m = -\left[\frac{\partial E_{y,x}\theta[y,F(x)]}{\partial F(x)}\right]_{F(x)=F_{m-1}(x)}$

算法原理 Gradient Boost

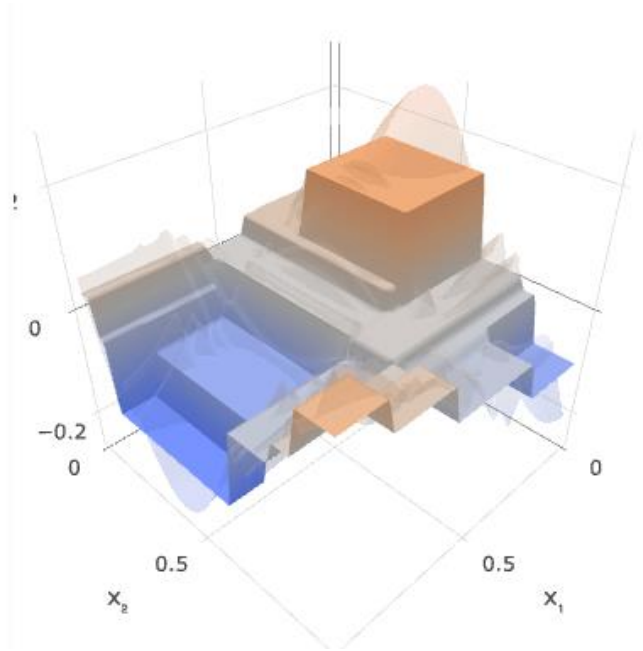
- 梯度提升（Gradient Boost）
  - $F^* = F^0 + \alpha g_1 + \cdots + \alpha g_M$，　$g_m = -\left[\frac{\partial E_{y,x}\theta[y,F(x)]}{\partial F(x)}\right]_{F(x)=F_{m-1}(x)}$
  - 迭代过程
    1. 拟合 $f_m \approx g_m$，样本视角：样本标签变为当前预测误差的某种形式
    2. $\beta_m = \underset{\beta}{\operatorname{argmin}} E_{y,x}\theta\ [y,\ F_{m-1}(x) + \beta_m f_m]$

算法原理 GBDT

- 梯度提升（Gradient Boost）

Algorithm: Gradient Boost	
1	$F_0(\boldsymbol{x}) = argmin_\beta \sum_{i=1}^{N} \theta(y_i, \beta)$
2	For m = 1 to M do:
3	$\tilde{y}_i = -\left[\frac{\partial \theta[y_i, F(x_i)]}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, \ i = 1, \ldots, N$
4	$\mathbf{w}_m = argmin_{\mathbf{w}} \sum_{i=1}^{N} [\tilde{y}_i - \alpha f(\mathbf{x}_i; \mathbf{w})]^2$
5	$\beta_m = argmin_\beta \sum_{i=1}^{N} \theta[y_i, F_{m-1}(\mathbf{x}_i) + \beta f(\mathbf{x}_i; \mathbf{w}_m)]$
6	$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m f(\mathbf{x}; \mathbf{w}_m)$
7	endFor
	end Algorithm

- 决策树（Decision Tree）
  - $f(\mathbf{x}_i; \mathbf{w})$
  - 回归树！
  - Just use CART



算法实现 GBDT

- Scikit-learn

```
1  from sklearn import ensemble
2
3  # 配置超参数
4  params = {'n_estimators': 1000, 'max_leaf_nodes': 4,
5            'max_depth': None, 'random_state': 2,
6            'min_samples_split': 5}
7  # 初始化模型
8  clf = ensemble.GradientBoostingClassifier(**params)
9
10 # 训练模型, X_train为矩阵, y_train为标签向量
11 clf.fit(X_train, y_train)
12
13 # 使用模型预测新样本
14 clf.predict(X_test)
```

算法原理 XGBoost

- XGBoost（eXtreme Gradient Boosting，陈天奇，华盛顿大学）
  - 可防止过拟合
    - 正则项（Regularization）
      - $Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$
    - 缩减（Shrinkage）
      - 类似 learning rate，随着迭代的持续减少新增树及叶子空间的影响
    - 列特征子抽样(Column feature subsampling)
      - 与随机森林算法一致，新增树只能使用部分特征进行训练，同时还可提高速度

算法原理 XGBoost

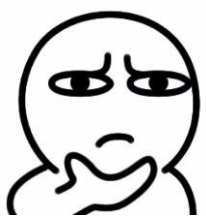
- XGBoost（eXtreme Gradient Boosting，陈天奇，华盛顿大学）
  - 考虑二阶导数信息
    - GBDT只利用了一阶的导数，XGBoost对损失函数做了二阶的泰勒展开

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

    - 可理解为，类似梯度下降和牛顿法的区别
  - 贪心算法构建树（也可使用近似算法）
    - 在树分裂时，遍历特征并对每个特征的全部取值进行排序，寻找最佳特征及最佳分裂点，最后减枝
  - 弱分类器选择
    - 除 GBDT 的树模型外，还可使用其他线性模型作为弱分类器

算法原理 XGBoost

- XGBoost（eXtreme Gradient Boosting，陈天奇，华盛顿大学）
  - 缺失值处理
    - 可为缺失值设置默认的划分方向（模拟左、右，选择增益大的方向）
  - 并行及大数据支持
    - 注意：并未改变 boosting 串行构建模型的结构，而是在特征层面实现并行，针对最大的资源消耗点：特征值排序确定分裂点
    - 实现
      - 特征预排序，以column block的结构存于内存中
      - block中采用CSC稀疏格式（Compressed Column format）
      - 当数据量过大时，将block存储于硬盘，按列压缩，设置相对索引
      - 优化CPU cache命中率（预取数据到buffer中，以及调节block大小）



算法实现 XGBoost

- LightGBM

```
1  import lightgbm as lgb
2
3  # 配置超参数
4  param = {'num_leaves':31, 'num_trees':100, 'objective':'binary'}
5  param['metric'] = 'auc'
6  num_round = 10
7
8  # 训练模型
9  bst = lgb.train(param, train_data, num_round, valid_sets=[test_data])
10
11 # 使用模型预测新样本
12 ypred = bst.predict(data)
```

优劣分析 应用总结

优劣分析

- RF
  - 训练速度快、预测准确度高，能够处理高维数据，无需特征选择，可给出特征重要度，容易并行化，不易过拟合
- GBDT
  - 预测准确度相比 RF 所有提升
  - 难以并行训练
- XGBoost
  - 预测准确度相比 GBDT 进一步提升，相对 GBDT 训练速度快，不易过拟合

应用总结

- 在 Kaggle 各竞赛中使用频率独占鳌头，被戏称为“R语言三马车”之一
  - randomForest（RF）, gbm（GB）, glmnet（特征选择）
- 阿里云大数据计算服务ODPS
- 腾讯微信内的购买点击预测
- 汽车之家展示广告的点击率预测

应用总结

- 2018年春季算法工程师面试真题
  - 360 一面
    - XGBoost 的原理
  - 美团 一面
    - GBDT 原理及常用的超参数
    - XGBoost 比 GBDT 好在哪
  - 美团 二面
    - XGBoost 中行抽样的作用
  - GrowingIO 一面
    - XGBoost 原理及步长如何设定
  - 数旦科技
    - Bagging 和 Boosting 的区别
    - 哪一个可以让方差更小
  - 阿里 一面
    - RF、GBDT、Adaboost
  - 阿里 二面
    - RF、GBDT、XGBoost
  - 阿里 三面
    - GBDT 和 XGBoost 的异同

参考文献

- Additive logistic regression a statistical view of boosting. Friedman(2000).
- XGBoost: A Scalable Tree Boosting System. T. Chen, C. Guestrin (2016).
- Complete Guide to Parameter Tuning in XGBoost (with codes in Python)
- 知乎：机器学习算法中GBDT和XGBOOST的区别有哪些?

知人者智，自知者明。

胜人者有力，自胜者强。

知足者富，强行者有志。

不失其所者久，死而不亡者，寿。

谢谢！