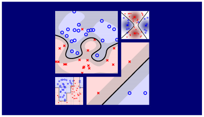
Machine Learning Techniques
(機器學習技法)



Lecture 11: Gradient Boosted Decision Tree

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering
National Taiwan University
(國立台灣大學資訊工程系)

Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 10: Random Forest

bagging of **randomized C&RT** trees with **automatic validation** and **feature selection**

Lecture 11: Gradient Boosted Decision Tree

- Adaptive Boosted Decision Tree
- Optimization View of AdaBoost
- Gradient Boosting
- Summary of Aggregation Models

3. Distilling Implicit Features: Extraction Models

From Random Forest to AdaBoost-DTree

需要让决策树处理带有不同权重的样本。
得到加权错误率最小的gt

function RandomForest($\mathcal{D}$)
For $t = 1, 2, \ldots, T$
1. request size-N' data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree g_t by Randomized-DTree($\tilde{\mathcal{D}}_t$)

return $G = \text{Uniform}(\{g_t\})$

function AdaBoost-DTree($\mathcal{D}$)
For $t = 1, 2, \ldots, T$
1. reweight data by $\mathbf{u}^{(t)}$
2. obtain tree g_t by DTree($\mathcal{D}, \mathbf{u}^{(t)}$)
3. calculate 'vote' α_t of g_t

return $G = \text{LinearHypo}(\{(g_t, \alpha_t)\})$

need: weighted DTree($\mathcal{D}, \mathbf{u}^{(t)}$)

随机森林：
把决策树bagging起来。
不同gt不同的随机化方式：bootstrap sample/随机特征子空间

adaboost提升树：
1每个gt也是决策树
2每个gt接收的数据带有不同的权重ut.根据权重选择最优的gt
并根据gt结果，得到下一次ut+1的权重，用来算gt+1
3 算gt同时得到gt的权重at
4 最后按权重合并所有gt的预测结果，得到G

Weighted Decision Tree Algorithm

Weighted Algorithm

最小化
加权错误率

minimize (regularized) $E_{\text{in}}^{\mathbf{u}}(h) = \frac{1}{N}\sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(\mathbf{x}_n))$

本来应该是在看算法计算Ein时，哪里能用到权重，做相应改变。但决策树改算法很麻烦。
因此根据ut抽样，来得到对应这些权重的样本。

if using existing algorithm as **black box** (no modifications),
to get $E_{\text{in}}^{\mathbf{u}}$ approximately optimized......

'Weighted' Algorithm in Bagging

weights $\mathbf{u}$ expressed by
bootstrap-sampled copies
—request size-N' data $\tilde{\mathcal{D}}_t$
by bootstrapping with $\mathcal{D}$

A General Randomized Base Algorithm

weights $\mathbf{u}$ expressed by
sampling proportional to u_n
—request size-N' data $\tilde{\mathcal{D}}_t$
by sampling $\propto \mathbf{u}$ on $\mathcal{D}$

bagging里，
样本被重复抽到Dt~以后，共有几份，就可以看作该样本权重是多少

adaboost-Tree:进入gt前，先通过权重un抽样。得到的数据Dt~就和un希望的权重一样。

AdaBoost-DTree: often via
AdaBoost + **sampling** $\propto \mathbf{u}^{(t)}$ + DTree($\tilde{\mathcal{D}}_t$)
without modifying DTree

之后直接用决策树拟合加权的数据Dt~
算error + gt权重at
其中error用D，un算而不是用Dt~算。
否则容易是0

Weak Decision Tree Algorithm

AdaBoost: **votes** $\alpha_t = \ln \blacklozenge_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ with weighted error rate ϵ_t

if fully grown tree trained on all $\mathbf{x}_n$

$\Longrightarrow E_{\text{in}}(g_t) = 0$ if all $\mathbf{x}_n$ different

$\Longrightarrow E^{\mathbf{u}}_{\text{in}}(g_t) = 0$

$\Longrightarrow \epsilon_t = 0$

$\Longrightarrow \alpha_t = \infty$ (**autocracy**!!)

如果一棵树gt能完全拟合输入数据：

gt错误率是0。该模型at会是无穷大。只有这个树发挥作用，别的gt没有用了。

希望这个树犯一点错，Ein(gt)!=0。不要完全拟合。

need: **pruned** tree trained on **some** $\mathbf{x}_n$ to be **weak**

- **pruned**: usual pruning, or just **limiting tree height** 剪枝/限制树高
- **some**: **sampling** $\propto \mathbf{u}^{(t)}$ 用un抽样，总有抽不到的数据

使gt不能在全部数据上完全拟合

但算error还是在整体D上用un算

AdaBoost-DTree: often via AdaBoost + **sampling** $\propto \mathbf{u}^{(t)}$ + **pruned** DTree($\tilde{\mathcal{D}}$)

符合un分布   非完全的剪枝

AdaBoost with Extremely-Pruned Tree

what if DTree with **height** ≤ 1 (extremely pruned)?

DTree (C&RT) with **height** ≤ 1

learn branching criteria 最弱的树：树高只有一层。只切一次

$$b(\mathbf{x}) = \underset{\text{decision stumps } h(\mathbf{x})}{\operatorname{argmin}} \sum_{c=1}^{2} |\mathcal{D}_c \text{ with } h| \cdot \text{impurity}(\mathcal{D}_c \text{ with } h)$$

—if impurity = binary classification error,

just a decision stump, remember? :-)

如果不纯净度用分类误差算，每个树就是决策桩。这时候树比较简单，再用adaboost时,un可以直接算误差时算进去，不用在进入gt前抽样得Dt~

AdaBoost-Stump
= **special case** of AdaBoost-DTree

Fun Time

When running AdaBoost-DTree with sampling and getting a decision tree g_t such that g_t achieves zero error on the sampled data set $\tilde{D}_t$. Which of the following is possible?

1. $\alpha_t < 0$
2. $\alpha_t = 0$
3. $\alpha_t > 0$
4. all of the above

Fun Time

When running AdaBoost-DTree with sampling and getting a decision tree g_t such that g_t achieves zero error on the sampled data set $\tilde{D}_t$. Which of the following is possible?

1. $\alpha_t < 0$
2. $\alpha_t = 0$
3. $\alpha_t > 0$
4. all of the above

Reference Answer: 4

While g_t achieves zero error on $\tilde{\mathcal{D}}_t$, g_t may not achieve zero weighted error on $(\mathcal{D}, \mathbf{u}^{(t)})$ and hence ϵ_t can be anything, even $\geq \frac{1}{2}$. Then, α_t can be ≤ 0.

Example Weights of AdaBoost

$$u_n^{(t+1)} = \begin{cases} u_n^{(t)} \cdot \blacklozenge_t & \textbf{if incorrect} \\ u_n^{(t)} / \blacklozenge_t & \textbf{if correct} \end{cases}$$

adaboost:根据gt预测结果更新ut。使ut+1在gt上表现很差，gt+1,gt diverse

$$= u_n^{(t)} \cdot \blacklozenge_t^{\,-y_n g_t(\mathbf{x}_n)} = u_n^{(t)} \cdot \exp\left(-y_n \alpha_t g_t(\mathbf{x}_n)\right)$$

简化写法 at=ln(t)

at:gt的权重

$$u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} \exp(-y_n \alpha_t g_t(\mathbf{x}_n)) = \frac{1}{N} \cdot \exp\left(-y_n \boxed{\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)}\right)$$

yn是固定的

- recall: $G(\mathbf{x}) = \text{sign}\left(\boxed{\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})}\right)$

可以看作是T个模型对xn的加权投票分数

- $\boxed{\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})}$: **voting score** of $\{g_t\}$ on $\mathbf{x}$

adaboost每轮的新权重和加权score有关

AdaBoost: $u_n^{(T+1)} \propto \exp(-y_n(\boxed{\text{voting score on } \mathbf{x}_n}))$

Voting Score and Margin

linear blending = LinModel + hypotheses as transform + ~~constraints~~

$$G(\mathbf{x}_n) = \text{sign}\left(\overbrace{\sum_{t=1}^{T} \underbrace{\alpha_t}_{w_i} \underbrace{g_t(\mathbf{x}_n)}_{\phi_i(\mathbf{x}_n)}}^{\text{voting score}}\right)$$

voting score各项
相当于linear model里的wi,fai(x)

voting score可以看作是一种没有正规化的距离（类比SVM）

and hard-margin SVM **margin** = $\frac{y_n \cdot (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$, **remember? :-)**

y_n(voting score) = signed & unnormalized **margin**

want y_n(voting score) **positive & large**

$\Leftrightarrow$ $\exp(-y_n(\text{voting score}))$ **small**

$\Leftrightarrow$ $u_n^{(T+1)}$ **small**

预测结果如果正确，yn*voting_score会很大
yn=1/-1，得分相差很大。
因此希望margin：yn*voting_score较大

对应到adaboost,最后是希望权重的值un较小(都学会了)

claim: AdaBoost **decreases** $\sum_{n=1}^{N} u_n^{(t)}$ 所有点的总权重

AdaBoost Error Function

claim: AdaBoost decreases $\sum_{n=1}^{N} u_n^{(t)}$ and thus somewhat **minimizes**

$$\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

voting score: 不同类别的margin变大

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)$

- $\mathrm{err}_{0/1}(s, y) = [\![ys \le 0]\!]$
- $\widehat{\mathrm{err}}_{\mathrm{ADA}}(s, y) = \exp(-ys)$: upper bound of $\mathrm{err}_{0/1}$ —called **exponential error measure**

$\widehat{\mathrm{err}}_{\mathrm{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\mathrm{err}_{0/1}$

AdaBoost Error Function

claim: AdaBoost decreases $\sum_{n=1}^{N} u_n^{(t)}$ and thus somewhat **minimizes**

$$\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)$

- $\mathrm{err}_{0/1}(s, y) = [\![ys \leq 0]\!]$
- $\widehat{\mathrm{err}}_{\mathrm{ADA}}(s, y) = \exp(-ys)$:
  upper bound of $\mathrm{err}_{0/1}$
  —called **exponential error measure**

$\widehat{\mathrm{err}}_{\mathrm{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\mathrm{err}_{0/1}$

AdaBoost Error Function

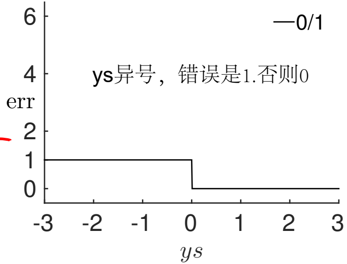
claim: AdaBoost **decreases** $\sum_{n=1}^{N} u_n^{(t)}$ and thus somewhat **minimizes**

$$\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

voting score: 不同类别的margin变大

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)$

- $\mathrm{err}_{0/1}(s, y) = [\![ys \le 0]\!]$
- $\widehat{\mathrm{err}}_{\mathrm{ADA}}(s, y) = \exp(-ys)$: upper bound of $\mathrm{err}_{0/1}$ —called **exponential error measure**



$\widehat{\mathrm{err}}_{\mathrm{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\mathrm{err}_{0/1}$

AdaBoost Error Function

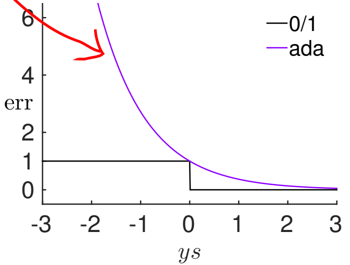
claim: AdaBoost **decreases** $\sum_{n=1}^{N} u_n^{(t)}$ and thus somewhat **minimizes**

$$\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

adaboost实际降低的是这个分类错误函数

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)$

- $\mathrm{err}_{0/1}(s, y) = [\![ys \le 0]\!]$
- $\widehat{\mathrm{err}}_{\mathrm{ADA}}(s, y) = \exp(-ys)$: upper bound of $\mathrm{err}_{0/1}$ 是01错误的上限
  —called **exponential error measure**



所以降低权重之和，相当于降低分类错误的凸上限。

$\widehat{\mathrm{err}}_{\mathrm{ADA}}$: **algorithmic error measure**
by **convex upper bound** of $\mathrm{err}_{0/1}$

Gradient Descent on AdaBoost Error Function

希望选一个更新方向v，使f(wt+1)=f(wt+nv)最小。wt+1=wt+nv按照泰勒展开 单位向量v和梯度反向(夹角180度)结果最小。因此v选择负梯度方向
更新wt: wt+1=wt+nv

recall: gradient descent (**remember? :-)**), at iteration t

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\overset{\text{步长}}{\eta}}_{\text{given positive}} \underset{\text{单位向量}}{\mathbf{v}^T} \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

复习一下梯度下降

Wt处的梯度方向

以前是找一个使表现最好的向量w。通过更新向量，使表现更好。现在是找一个使表现最好的gt。通过更新gt,使整体表现更好
向量是给定index,输出对应值。函数gt输入是实数

at iteration t, to find g_t, solve

$$\min_h \widehat{E}_{\text{ADA}} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n\left(\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n) + \eta h(\mathbf{x}_n)\right)\right)$$

当前的整体的函数

我们希望通过加一个新的at+1,gt+1更新G.使Eada向减小的方向移动

$$= \sum_{n=1}^{N} u_n^{(t)} \exp\left(-y_n \eta h(\mathbf{x}_n)\right)$$

$$\overset{\text{taylor}}{\approx} \sum_{n=1}^{N} u_n^{(t)}\left(1 - y_n \eta h(\mathbf{x}_n)\right) = \sum_{n=1}^{N} u_n^{(t)} - \eta \sum_{n=1}^{N} u_n^{(t)} y_n h(\mathbf{x}_n)$$

对exp(x)在原点处展开

固定

adaboost相当于通过选择合适的n, h(X),最小化这部分

选择合适的h
最小化error

good h: minimize $\sum_{n=1}^{N} u_n^{(t)}\left(-y_n h(\mathbf{x}_n)\right)$

Learning Hypothesis as Optimization

finding good h (function direction) $\Leftrightarrow$ minimize $\sum_{n=1}^{N} u_n^{(t)} \left(-y_n h(\mathbf{x}_n)\right)$

for binary classification, where y_n and $h(\mathbf{x}_n)$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u_n^{(t)} \left(-y_n h(\mathbf{x}_n)\right) = \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(\mathbf{x}_n) \\ +1 & \text{if } y_n \neq h(\mathbf{x}_n) \end{cases}$$

抽出一个-un

$$= -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(\mathbf{x}_n) \\ 2 & \text{if } y_n \neq h(\mathbf{x}_n) \end{cases}$$

如果是二分类
相当于选一个h,
最小化h在N个训练数据上的加权错误率

$$= -\sum_{n=1}^{N} u_n^{(t)} + 2 E_{\text{in}}^{\mathbf{u}^{(t)}}(h) \cdot N$$

固定的 h(x)上的错误率

而adaboost里
gt就是使得ut加权下的Ein最小的算法
ht=gt

—who minimizes $E_{\text{in}}^{\mathbf{u}^{(t)}}(h)$? $\mathcal{A}$ **in AdaBoost! :-)**

$\mathcal{A}$: **good** $g_t = h$ for 'gradient descent'

adaboost选的是一个好的，G的更新方向。使G朝着nh的方向更新

Deciding Blending Weight as Optimization

AdaBoost finds g_t by approximately $\min_h \widehat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp\left(-y_n \eta h(\mathbf{x}_n)\right)$

adaboost每次找一个最好的更新方向h, 更新G

after finding g_t, how about $\min_\eta \widehat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp\left(-y_n \eta g_t(\mathbf{x}_n)\right)$

如何选步长呢？在最优方向基础上，选一个最优的步长

在当前gt情况下，选最优的方向和步长，贪婪的更新G（尽管未必全局最优）

- optimal η_t somewhat **'greedily faster'** than fixed (small) η
  —called **steepest** decent for optimization
- two cases inside summation: 考虑2分类，可以把Eada错误写成：
  - $y_n = g_t(\mathbf{x}_n)$: $u_n^{(t)} \exp(-\eta)$ (correct)
  - $y_n \neq g_t(\mathbf{x}_n)$: $u_n^{(t)} \exp(+\eta)$ (incorrect)

  加权错误率/归一化
- $\widehat{E}_{\text{ADA}} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left((1-\epsilon_t)\exp(-\eta) + \epsilon_t \exp(+\eta)\right)$

最优化n

by solving $\frac{\partial \widehat{E}_{\text{ADA}}}{\partial \eta} = 0$, **steepest** $\eta_t = \ln\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \alpha_t$, **remember? :-)**

—AdaBoost: steepest decent with approximate functional gradient

Fun Time

With $\widehat{E}_{\text{ADA}} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left((1-\epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$, which of the following is $\frac{\partial \widehat{E}_{\text{ADA}}}{\partial \eta}$ that can be used for solving the optimal η_t?

1. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(+ (1-\epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$
2. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(+ (1-\epsilon_t) \exp(-\eta) - \epsilon_t \exp(+\eta) \right)$
3. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(- (1-\epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$
4. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(- (1-\epsilon_t) \exp(-\eta) - \epsilon_t \exp(+\eta) \right)$

Fun Time

With $\widehat{E}_{\text{ADA}} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left((1-\epsilon_t)\exp(-\eta) + \epsilon_t \exp(+\eta)\right)$, which of the following is $\frac{\partial \widehat{E}_{\text{ADA}}}{\partial \eta}$ that can be used for solving the optimal η_t?

1. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(+(1-\epsilon_t)\exp(-\eta) + \epsilon_t \exp(+\eta)\right)$
2. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(+(1-\epsilon_t)\exp(-\eta) - \epsilon_t \exp(+\eta)\right)$
3. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(-(1-\epsilon_t)\exp(-\eta) + \epsilon_t \exp(+\eta)\right)$
4. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left(-(1-\epsilon_t)\exp(-\eta) - \epsilon_t \exp(+\eta)\right)$

Reference Answer: 3

Differentiate $\exp(-\eta)$ and $\exp(+\eta)$ with respect to η and you should easily get the result.

Gradient Boosting for Arbitrary Error Function

AdaBoost

$$\min_{\eta} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \left(\underbrace{\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(\mathbf{x}_n)}_{s} + \eta h(\mathbf{x}_n)\right)\right)$$

s是vote function

adaboost相当于最小化二分类误差上限exp(-yn*s)
对于固定步长n，选一个最佳更新方向gt，
对于固定更新方向，选一个最佳更新步长n
组合起来更新G，使G(x)对应的error function降低

with binary-output hypothesis h

GradientBoost

同样
先最优化方向h(x)
再优化步长n
作为我新的gt+1
at+1

$$\min_{\eta} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \underbrace{\mathrm{err}\left(\underbrace{\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(\mathbf{x}_n)}_{\text{已有的voting score}} + \eta h(\mathbf{x}_n), y_n\right)}_{\text{换成任意感兴趣的error function}}$$

沿着使error function下降最快的步长n和方向h(x)更新G

with any hypothesis h (usually real-output hypothesis)

GradientBoost: allows **extension to different** err for regression/soft classification/etc.

GradientBoost for Regression

误差是回归误差。G(x)输出是实数

$$\min_{\eta} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \mathrm{err}\left(\underbrace{\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)}_{s_n} + \eta h(\mathbf{x}_n), y_n\right) \text{ with } \mathrm{err}(s, y) = (s-y)^2$$

已有的voting score
可能是实数

先考虑回归：平方误差函数

$$\min_{h} \ldots \overset{\text{taylor}}{\approx} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \underbrace{\mathrm{err}(s_n, y_n)}_{\text{constant}} + \frac{1}{N}\sum_{n=1}^{N} \eta h(\mathbf{x}_n) \left.\frac{\partial \mathrm{err}(s, y_n)}{\partial s}\right|_{s=s_n}$$

原来的错误率，常数

会展出一个一阶梯度的项
是error函数在sn点处的梯度

error function在sn处一阶泰勒展开。对每个点(xnyn）都这么做，最后加起来求平均。

$$= \min_{h} \text{constants} + \frac{\eta}{N}\sum_{n=1}^{N} h(\mathbf{x}_n) \cdot 2(s_n - y_n)$$

最小化这部分。让h(x)与（sn-yn）反向

naïve solution $h(\mathbf{x}_n) = -\infty \cdot (s_n - y_n)$
if no constraint on h

只需要考虑h的方向。大小由n控制。但限制|h|==1不好优化。用正则项制约下

Learning Hypothesis as Optimization

$$\min_{h} \quad \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} 2h(\mathbf{x}_n)(s_n - y_n)$$

- magnitude of h does not matter: because η will be optimized next
- penalize large magnitude to avoid naïve solution

$$\min_{h} \quad \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left(2h(\mathbf{x}_n)(s_n - y_n) + (h(\mathbf{x}_n))^2\right)$$

防止h(xn)的绝对值太大。
我们只在意h方向。大小主要由n控制

$$= \quad \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left(\text{constant} + (h(\mathbf{x}_n) - (y_n - s_n))^2\right)$$

(sn-yn)**2 常数　　可以凑出这样一个形式

- solution of penalized approximate functional gradient: squared-error regression on $\{(\mathbf{x}_n, \underbrace{y_n - s_n}_{\text{residual}})\}$

希望学到的gt,平方误差上逼近yn-sn
给gt的数据是（xn,residual）
希望拟合的数据是残差，
是yn-sn(xn对应的当前的voting score)

这是一个回归问题。
如果用决策树作为gt，选择regression error作为不纯净度
gt最终的训练数据不是(xn,yn),而是（xn,yn-sn）

GradientBoost for regression:
find $g_t = h$ by regression with **residuals**

Deciding Blending Weight as Optimization

after finding $g_t = h$,

找到好的gt/h后，需要找到使error最小的最优步长n。即新gt的权重at

$$\min_{\eta} \cancel{\min_{h}} \frac{1}{N}\sum_{n=1}^{N} \text{err}\Big(\underbrace{\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)}_{s_n} + \eta g_t(\mathbf{x}_n), y_n\Big) \text{ with } \text{err}(s,y) = (s-y)^2$$

重写一下

$$\min_{\eta} \quad \frac{1}{N}\sum_{n=1}^{N}(s_n + \eta g_t(\mathbf{x}_n) - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{(y_n - s_n)}_{\text{还是xn在这点的残差}} - \eta \underbrace{g_t(\mathbf{x}_n)}_{\text{固定的实数}})^2$$

—one-variable linear regression on $\{(g_t\text{-transformed input, residual})\}$

X:N个样本，每个样本只有一维特征gt(xn)　只需要一个变量wi　error=mean（yn-sum_i(wi*xn,i)）**2
Y:N个样本，每个样本的输出是（yn-sn）

相当于单维度的线性回归问题。求出单变量n=wi即gt的权重

GradientBoost for regression: α_t = optimal η by g_t-transformed linear regression

Putting Everything Together

Gradient Boosted Decision Tree (GBDT)

$s_1 = s_2 = \ldots = s_N = 0$ T=0所有点sn=0。因为还没有voting score.

for $t = 1, 2, \ldots, T$

1. obtain g_t by $\mathcal{A}(\{(\mathbf{x}_n, y_n - s_n)\})$ where $\mathcal{A}$ is a (squared-error) regression algorithm 1 先解一个regression问题：xn-->yn-sn 得到最优的gt
   —**how about sampled and pruned C&RT?** 这里用决策树cart做regression
2. compute $\alpha_t = \text{OneVarLinearRegression}(\{(g_t(\mathbf{x}_n), y_n - s_n)\})$
3. update $s_n \leftarrow s_n + \alpha_t g_t(\mathbf{x}_n)$

2 之后解一个一维的线性regression问题 gt -->yn-sn 得到at

3 根据gt,at, 更新所有样本的voting score sn

return $G(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})$ 4 T棵树后，输出总的G(x) 因为是regression,直接输出加权实值

GBDT: 'regression sibling' of AdaBoost-DTree
—**popular in practice**

是Adaboost树的回归版本。降低的是不同的error function
adaboost-决策树是最小化二分类损失的上限。GBDT是最小化回归问题的平方损失
基本函数A都是决策树CART。用不同的加权函数组合起来，得到G

Fun Time

Which of the following is the optimal η for

$$\min_{\eta} \quad \frac{1}{N}\sum_{n=1}^{N}\left((y_n - s_n) - \eta g_t(\mathbf{x}_n)\right)^2$$

1. $(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)) \cdot (\sum_{n=1}^{N} g_t^2(\mathbf{x}_n))$
2. $(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)) \;/\; (\sum_{n=1}^{N} g_t^2(\mathbf{x}_n))$
3. $(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)) + (\sum_{n=1}^{N} g_t^2(\mathbf{x}_n))$
4. $(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)) - (\sum_{n=1}^{N} g_t^2(\mathbf{x}_n))$

第二步，算最优at
可以得到公式解
n=@2

Fun Time

Which of the following is the optimal η for

$$\min_{\eta} \quad \frac{1}{N}\sum_{n=1}^{N}\left((y_n - s_n) - \eta g_t(\mathbf{x}_n)\right)^2$$

1. $\left(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)\right) \cdot \left(\sum_{n=1}^{N} g_t^2(\mathbf{x}_n)\right)$
2. $\left(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)\right) / \left(\sum_{n=1}^{N} g_t^2(\mathbf{x}_n)\right)$
3. $\left(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)\right) + \left(\sum_{n=1}^{N} g_t^2(\mathbf{x}_n)\right)$
4. $\left(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)\right) - \left(\sum_{n=1}^{N} g_t^2(\mathbf{x}_n)\right)$

Reference Answer: 2

Derived within Lecture 9 of ML Foundations, **remember? :-)**

Map of Blending Models

blending: aggregate after getting **diverse** g_t 已经有gt了

uniform	non-uniform	conditional
simple voting/averaging of g_t	linear model on g_t-transformed inputs	nonlinear model on g_t-transformed inputs
降低方差，结果稳定	线性组合，linear model学习权重at	gt的非linear 组合

uniform for 'stability';
non-uniform/conditional **carefully** for 'complexity'

相对容易过拟合

Map of Aggregation-Learning Models

learning: aggregate as well as getting **diverse** g_t

Bagging

diverse g_t by bootstrapping;
uniform vote
by nothing :-)

Dt~
+决策树（不同维度特征）
（完全长成 差异大）

AdaBoost

diverse g_t
by reweighting;
linear vote
by steepest search

数据D
gt降低
加权分
类误差

不断改变ut/贪心的学习最优的gt,at，降低分类误差 exp error

Decision Tree

diverse g_t
by data splitting;
conditional vote
by branching

可以用surrogate解决缺失值问题
可以回归/分类
可以处理category类型的特征

GradientBoost

diverse g_t
by residual fitting;
linear vote
by steepest search

输出是实数

与adaboost类似。但最小化的是regression error
决策树gt拟合的是残差。at最小化的也是残差

boosting-like algorithms most popular

Map of Aggregation of Aggregation Models

Bagging

AdaBoost

Decision Tree

Random Forest

randomized bagging
+ ‘strong’ DTree

完全长成的gt: 对输入更加敏感

加上bootstrap+ random d’维特征

diversity

可以用OOV来做自我验证+特征选择

特征选择时，用对应维度的特征取值
permuation得到Dp。和原始输入D结果
作对比。
可以G不变，只在Eval时做。
差距大，该特征重要

AdaBoost-DTree

AdaBoost
+ ‘weak’ DTree

一开始弱一点。否则et==0,at无穷

如果是决策树， ut不好直接搞到
通过ut sample 产生数据D，再送给gt

最后在D，ut上，计算et, at

一般是非完全树，提前剪枝。
限制阈值、树高

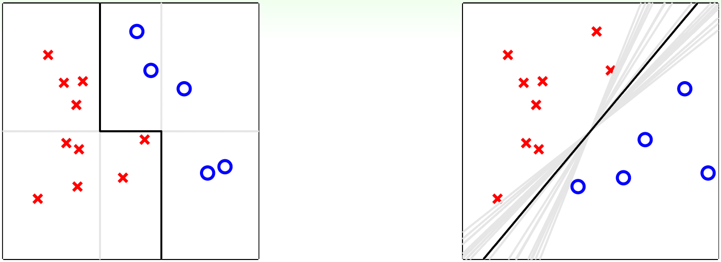
GradientBoost

GBDT

GradientBoost
+ ‘weak’ DTree

all three frequently used in practice

Specialty of Aggregation Models



cure underfitting

- $G(\mathbf{x})$ 'strong' 类似于特征变换
- aggregation 通过特征组合使模型变强 有一定的解决欠拟合的效果
  $\Longrightarrow$ **feature transform**

cure overfitting

- $G(\mathbf{x})$ 'moderate'
- aggregation 有一定的解决过拟合的效果 比如降低方差/增大margin
  $\Longrightarrow$ **regularization**

proper aggregation (a.k.a. 'ensemble')
$\Longrightarrow$ **better performance**

Fun Time

Which of the following aggregation model learns diverse g_t by reweighting and calculates linear vote by steepest search?

1. AdaBoost
2. Random Forest
3. Decision Tree
4. Linear Blending

Fun Time

Which of the following aggregation model learns diverse g_t by reweighting and calculates linear vote by steepest search?

1. AdaBoost
2. Random Forest
3. Decision Tree
4. Linear Blending

Reference Answer: 1

Congratulations on being an **expert** in aggregation models! **:-)**

Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 11: Gradient Boosted Decision Tree

- Adaptive Boosted Decision Tree
  sampling and pruning for 'weak' trees
- Optimization View of AdaBoost
  functional gradient descent on exponential error
- Gradient Boosting
  iterative steepest residual fitting
- Summary of Aggregation Models
  some cure underfitting; some cure overfitting

3. Distilling Implicit Features: Extraction Models
   - **next: extract features other than hypotheses**