UNIVERSITY OF WASHINGTON

Introduction to Boosted Trees

Tianqi Chen
Oct. 22 2014

Outline

- **Review of key concepts of supervised learning**
  监督学习的关键概念回顾
- Regression Tree and Ensemble (What are we Learning)
  回归树与集成算法（我们在学习什么）
- Gradient Boosting (How do we Learn)
  梯度提升（我们怎么学习）
- Summary
  总结

监督学习的元素

Elements in Supervised Learning

- Notations: $x_i \in \mathbf{R}^d$ i-th training example
  记号
- **Model**: how to make prediction $\hat{y}_i$ given x_i
  - Linear model: $\hat{y}_i = \sum_j w_j x_{ij}$ (include linear/logistic regression)
  - The prediction score $\hat{y}_i$ can have different interpretations depending on the task
    - Linear regression: $\hat{y}_i$ is the predicted score
    - Logistic regression: $1/(1+exp(-\hat{y}_i))$ is predicted the probability of the instance being positive
    - Others... for example in ranking $\hat{y}_i$ can be the rank score
- **Parameters**: the things we need to learn from data
  - Linear model: $\Theta = \{w_j | j = 1, \cdots, d\}$

目标函数

Elements continued: Objective Function

- Objective function that is everywhere

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

Regularization, measures complexity of model

- Loss on training data: $L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$
  - Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$
- Regularization: how complicated the model is?
  - L2 norm: $\Omega(w) = \lambda \|w\|^2$
  - L1 norm (lasso): $\Omega(w) = \lambda \|w\|_1$

具体应用

Putting known knowledge into context

- Ridge regression: $\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|^2$
  - Linear model, square loss, L2 regularization
- Lasso: $\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|_1$
  - Linear model, square loss, L1 regularization
- Logistic regression:

$$\sum_{i=1}^{n}[y_i \ln(1 + e^{-w^T x_i}) + (1 - y_i)\ln(1 + e^{w^T x_i})] + \lambda\|w\|^2$$

  - Linear model, logistic loss, L2 regularization
- The conceptual separation between model, parameter, objective also gives you **engineering benefits**.
  - Think of how you can implement SGD for both ridge regression and logistic regression 解耦，代码复用性高

模型、参数、目标之间的概念分离带来了工程效益

目标函数和偏差方差平衡

Objective and Bias Variance Trade-off

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

Regularization, measures complexity of model

为什么我们要在目标中包含两个成分？

- Why do we want to contain two component in the objective?

优化训练损失鼓励模型拟合程度

- Optimizing training loss encourages **predictive** models
  - Fitting well in training data at least get you close to training data which is hopefully close to the underlying distribution

优化正则化鼓励简单模型

- Optimizing regularization encourages **simple** models
  - Simpler models tends to have smaller variance in future predictions, making prediction **stable**

Outline

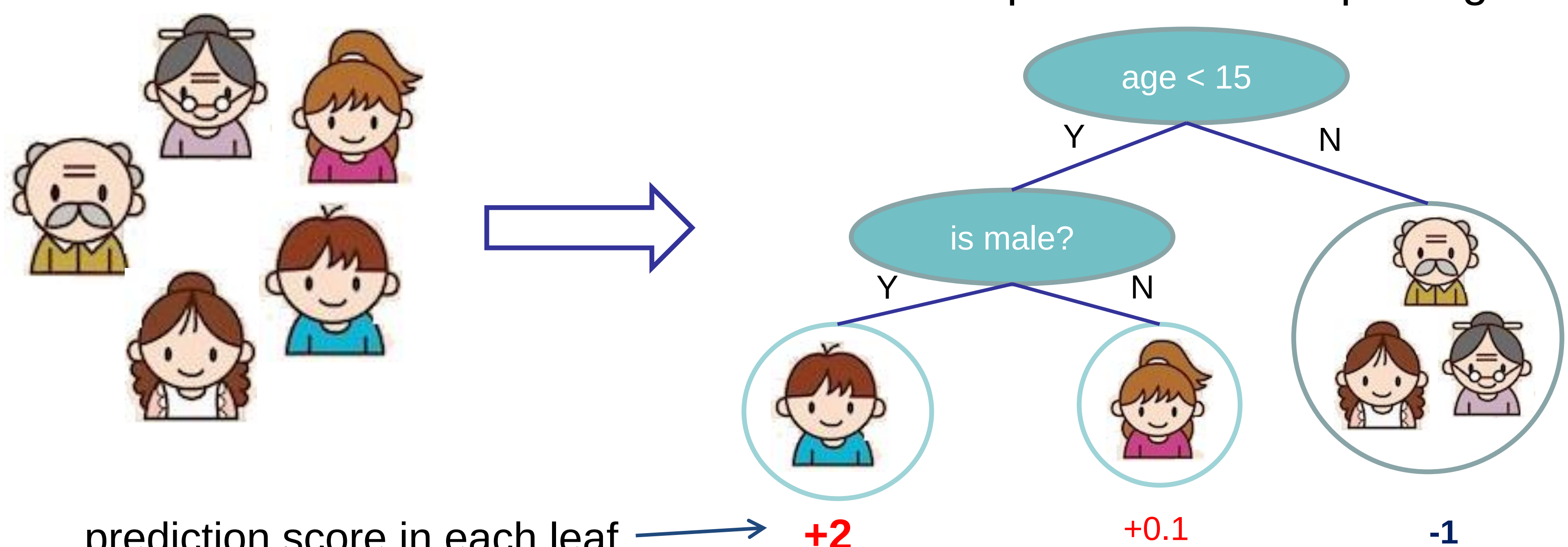
- Review of key concepts of supervised learning
- **Regression Tree and Ensemble (What are we Learning)**
  回归树与集成算法（我们在学习什么）
- Gradient Boosting (How do we Learn)
  用了一套通用的解决方案——梯度提升
- Summary

Regression Tree (CART)

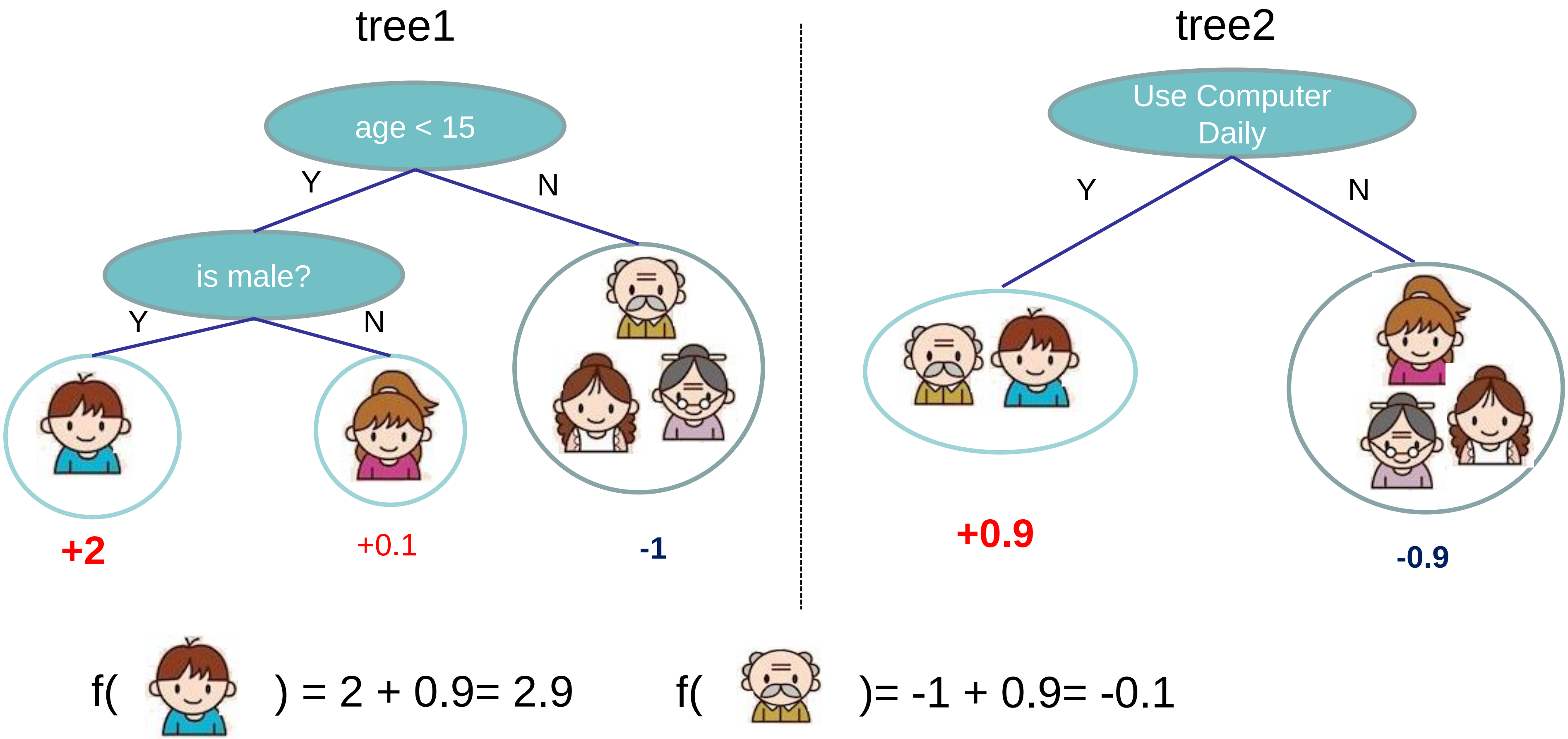
- CART
  regression tree (also known as classification and regression tree):
  - Decision rules same as in decision tree
  - Contains one score in each leaf value

Input: age, gender, occupation, …

Does the person like computer games



Regression Tree Ensemble



Prediction of is sum of scores predicted by each of the tree
预测值是每棵树分数的和

Tree Ensemble methods

- Very widely used, look for GBM, random forest…
  - Almost half of data mining competition are won by using some variants of tree ensemble methods

  使用广泛，例如梯度提升模型，随机森林

- Invariant to scaling of inputs, so you do not need to do careful features normalization.

  不需特征缩放，所以你不需要做仔细的特征归一化

- Learn higher order interaction between features.

  学习特征之间的高阶交互

- Can be scalable, and are used in Industry

  可扩展性，用于工业

集成算法实际应用：模型与参数

Put into context: Model and Parameters

- Model: assuming we have K trees

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

Space of functions containing all Regression trees

F：假设空间，包含了所有回归树

Think: regression tree is a function that maps the attributes to the score

思想：回归树是一个函数，他将特征、属性向量映射为一个分数

- Parameters
  - Including structure of each tree, and the score in the leaf

    每棵树的结构，以及叶子上的分数
  - Or simply use function as parameters

$$\Theta = \{f_1, f_2, \cdots, f_K\}$$

  - Instead learning weights in $\mathbf{R}^d$, we are learning functions(trees)

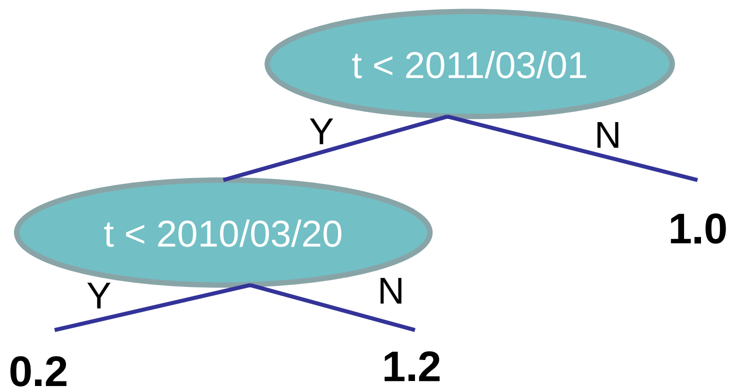
也可以换个角度，认为不是学习这些参数，而是学习多个函数

单变量时，树的学习

Learning a tree on single variable

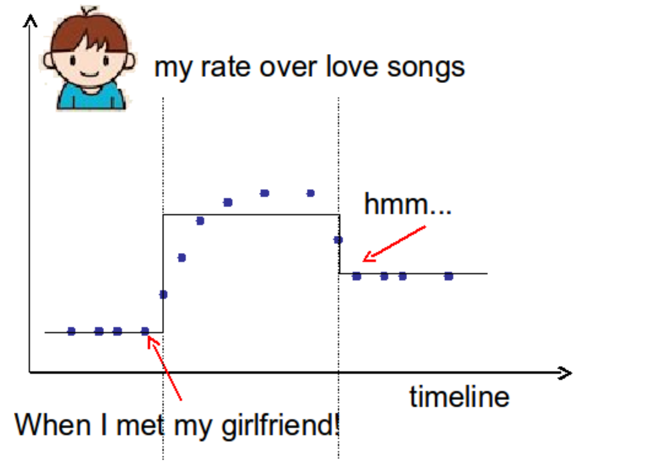
- How can we learn functions?
- Define objective (loss, regularization), and optimize it!!
- Example:
  - Consider regression tree on single input t (time)
  - I want to predict whether I like romantic music at time t

The model is regression tree that splits on time



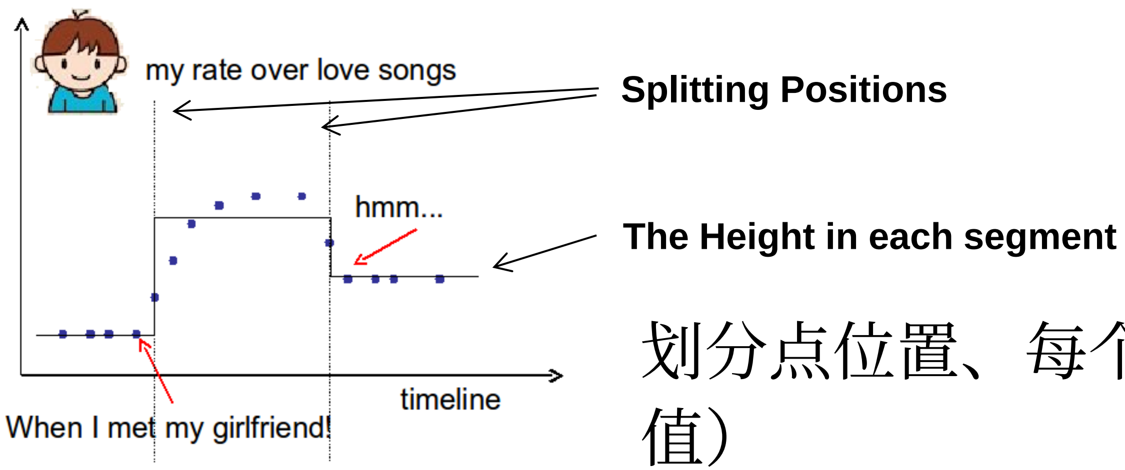
Equivalently

Piecewise step function over time



Learning a step function 学习阶跃函数

- Things we need to learn

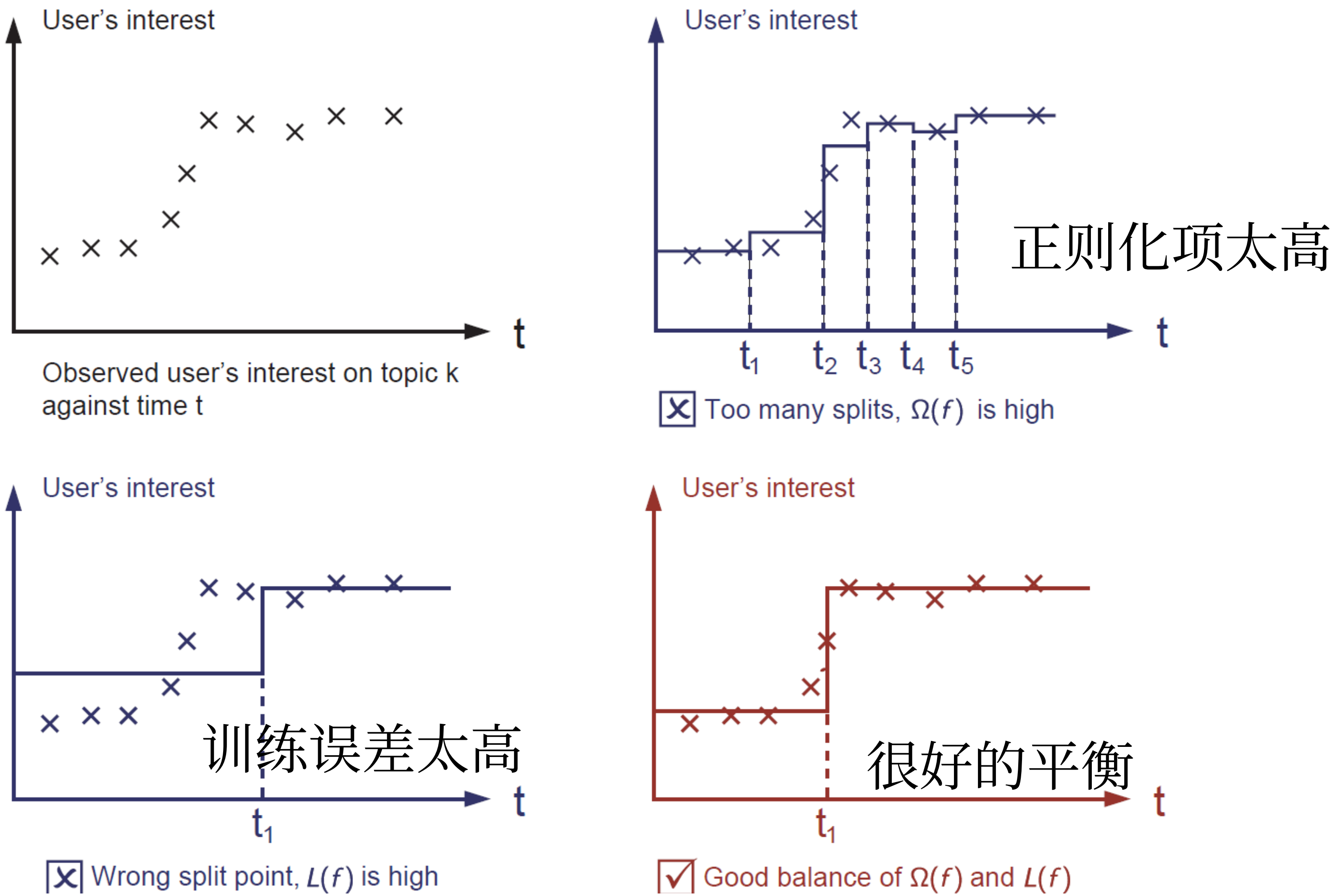


划分点位置、每个分段上的高度（取值）

- Objective for single variable regression tree(step functions)
  - Training Loss: How will the function fit on the points?
  - Regularization: How do we define complexity of the function?
    - Number of splitting points, l2 norm of the height in each segment?

视觉上学习阶跃函数

Learning step function (visually)



回到：集成树模型的目标函数

Coming back: Objective for Tree Ensemble

- Model: assuming we have K trees

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

- Objective

$$Obj = \underbrace{\sum_{i=1}^{n} l(y_i, \hat{y}_i)}_{\text{Training loss}} + \underbrace{\sum_{k=1}^{K} \Omega(f_k)}_{\text{Complexity of the Trees}}$$

- Possible ways to define Ω ?
  - Number of nodes in the tree, depth
  - L2 norm of the leaf weights
  - … detailed later

Objective vs Heuristic 目标函数 VS 启发式学习

- When you talk about (decision) trees, it is usually heuristics
  - Split by information gain
  - Prune the tree
  - Maximum depth
  - Smooth the leaf values
- Most heuristics maps well to objectives, taking the formal (objective) view let us know what we are learning
  - Information gain -> training loss
  - Pruning -> regularization defined by #nodes
  - Max depth -> constraint on the function space
  - Smoothing leaf values -> L2 regularization on leaf weights

不需要了解

回归树不仅仅是用来做回归的

Regression Tree is not just for regression!

- Regression tree ensemble defines how you make the prediction score, it can be used for
  - Classification, Regression, Ranking….
  - ….
- It all depends on how you define the objective function!
- So far we have learned:
  - Using Square loss $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$ 回归
    - Will results in common gradient boosted machine
  - Using Logistic loss $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i)\ln(1 + e^{\hat{y}_i})$
    - Will results in LogitBoost 分类

小结

Take Home Message for this section

- Bias-variance tradeoff is everywhere 偏差和方差的平衡无处不在
- The loss + regularization objective pattern applies for regression tree learning (function learning)
 损失+正则化目标模式适用于回归树学习（函数学习）
- We want **predictive** and **simple** functions
 我们需要训练误差小并且简单的函数。
- This defines what we want to learn (objective, model).
 这就定义了我们想要学习的东西（目标，模型）
- But how do we learn it?
  - Next section

Outline

- Review of key concepts of supervised learning
- Regression Tree and Ensemble (What are we Learning)
- **Gradient Boosting (How do we Learn)**
  梯度提升（我们怎么学习）
- Summary

So How do we Learn?

- Objective: $\sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), f_k \in \mathcal{F}$
- We can not use methods such as SGD, to find f (since they are trees, instead of just numerical vectors)

我们不能使用诸如SGD之类的方法来找到f

- Solution: **Additive Training (Boosting)**
  - Start from constant prediction, add a new function each time

从常数预测开始，每次添加一个新函数

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

New function

Model at training round t

Keep functions added in previous round

保留前面轮次中添加的函数

Additive Training加法模型训练

- How do we decide which f to add?
  - Optimize the objective!!
- The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

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

这里和GBDT不一样

Goal: find f_t to minimize this

- Consider square loss

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

This is usually called residual from previous round

损失的泰勒展开逼近

Taylor Expansion Approximation of Loss

- Goal $Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$
  - Seems still complicated except for the case of square loss
- Take Taylor expansion of the objective 对目标函数泰勒展开
  - Recall $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$
  - Define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

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

令x-x0=△x，将x视为静态的，将x+△x视为动态的

- *If you are not comfortable with this, think of square loss*

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

- Compare what we get to previous slide

思考：为什么要用泰勒展开？

Our New Goal 新目标函数

- Objective, with constants removed

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

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

- Why spending s much efforts to derive the objective, why not just grow trees … 为什么花这么多的精力来推导目标函数，为什么不直接建树呢？
  - Theoretical benefit: know what we are learning, convergence
    理论上的好处：知道我们在学习什么，知道什么时候收敛
  - **Engineering** benefit, recall the elements of supervised learning
    - g_i and h_i comes from definition of loss function
      g和h来自损失函数的定义
    - The learning of function only depend on the objective via g_i and h_i
      函数的学习只依赖于与g和h有关的目标
    - Think of how you can separate modules of your code when you are asked to implement boosted tree for both square loss and logistic loss
      当你被要求使用平方损失和逻辑损失的提升树时，请考虑如何分离代码的模块

Refine the definition of tree 精炼树的定义

- We define tree by a vector of scores in leafs, and a leaf index mapping function that maps an instance to a leaf

我们通过叶子的分数向量w和叶子索引映射函数q(x)来定义树，叶子索引映射函数可以将实例映射到叶节点

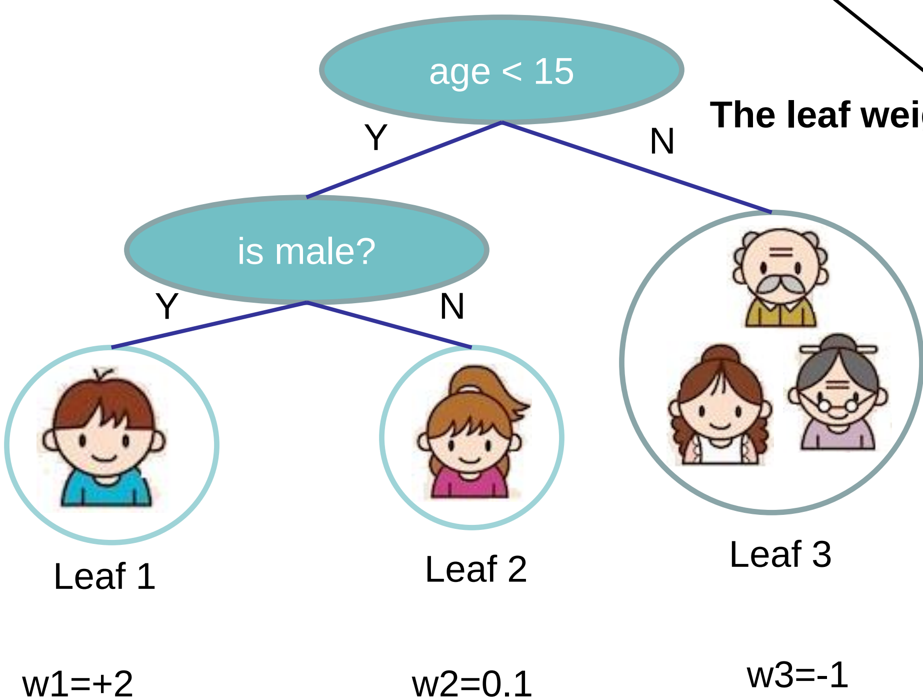
$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \rightarrow \{1, 2, \cdots, T\}$$

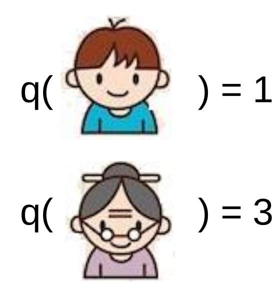
T：叶子编号

The structure of the tree

The leaf weight of the tree

w：叶子节点的分数向量





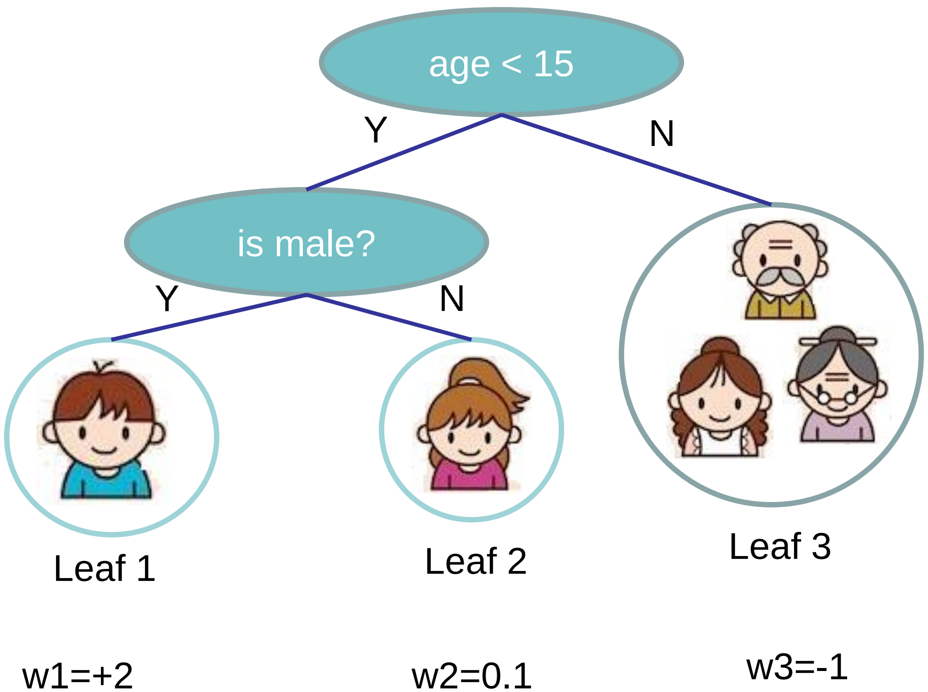
q(x)是叶子索引映射函数（其实就是树结构），输入一个样本，返回样本所在叶子索引

Define the Complexity of Tree 定义树的复杂度

- Define complexity as (this is not the only possible definition)

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

Number of leaves **L2 norm of leaf scores**



$$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

Revisit the Objectives 重新审视目标函数

- Define the instance set in leaf j as $I_j = \{i | q(x_i) = j\}$
  定义叶子j上的样本集合为Ij .
- Regroup the objective by each leaf 用每一个叶子重新组合目标函数

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

- This is sum of T independent quadratic functions

  这是T个独立的二次函数的求和

q(x)是叶子索引映射函数，输入一个样本，返回样本所在叶子索引。Ij是叶子索引为j的所有样本的序号的集合。

The Structure Score 结构分数

- Two facts about single variable quadratic function
  单变量二次函数的两个事实

$$argmin_x \; Gx + \frac{1}{2}Hx^2 = -\frac{G}{H},\; H > 0 \qquad \min_x \; Gx + \frac{1}{2}Hx^2 = -\frac{1}{2}\frac{G^2}{H}$$

- Let us define $G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$

$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \\ &= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

- Assume the structure of tree (q(x)) is fixed, the optimal weight in each leaf, and the resulting objective value are

$$w_j^* = -\frac{G_j}{H_j + \lambda} \qquad Obj = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

This measures how good a tree structure is!

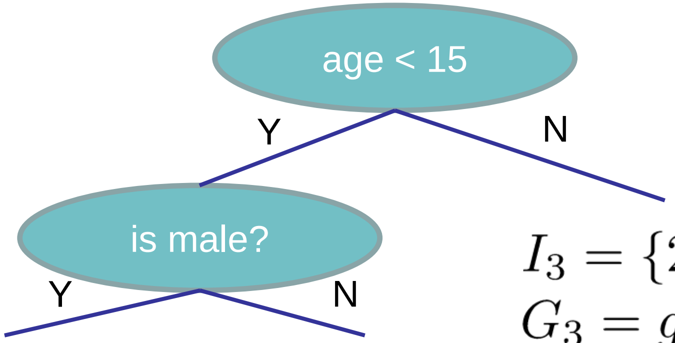
假设树的结构（qx）确定了，每个叶子最优的权重和最终的目标函数值就确定了

！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！

The Structure Score Calculation 结构分数计算

定义叶子j上的样本集合为I j

Instance index		gradient statistics
1		g1, h1
2		g2, h2
3		g3, h3
4		g4, h4
5		g5, h5



$$I_1 = \{1\} \quad G_1 = g_1 \quad H_1 = h_1$$

$$I_2 = \{4\} \quad G_2 = g_4 \quad H_4 = h_4$$

$$I_3 = \{2, 3, 5\} \quad G_3 = g_2 + g_3 + g_5 \quad H_3 = h_2 + h_3 + h_5$$

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

The smaller the score is, the better the structure is

Obj分数越小，结构越好

单颗树搜索算法

Searching Algorithm for Single Tree

- Enumerate the possible tree structures q 枚举可能的树结构q
- Calculate the structure score for the q, using the scoring eq.

$$Obj = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j+\lambda} + \gamma T$$

使用评分方程计算q的结构得分

- Find the best tree structure, and use the optimal leaf weight

$$w_j^* = -\frac{G_j}{H_j+\lambda}$$

寻找到最优树结构后，就能求最优叶子权重

- But... there can be infinite possible tree structures..

但是……可以有无限的可能的树结构

Greedy Learning of the Tree 树的贪婪学习

- In practice, we grow the tree greedily 在实践中，我们贪婪地建树
  - Start from tree with depth 0

  对于树的每个叶节点，尝试添加一个划分。加入划分后目标函数的变化为
  - For each leaf node of the tree, try to add a split. The change of objective after adding the split is

The complexity cost by introducing additional leaf

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

额外增加的代价

the score of left child

the score of if we do not split

the score of right child

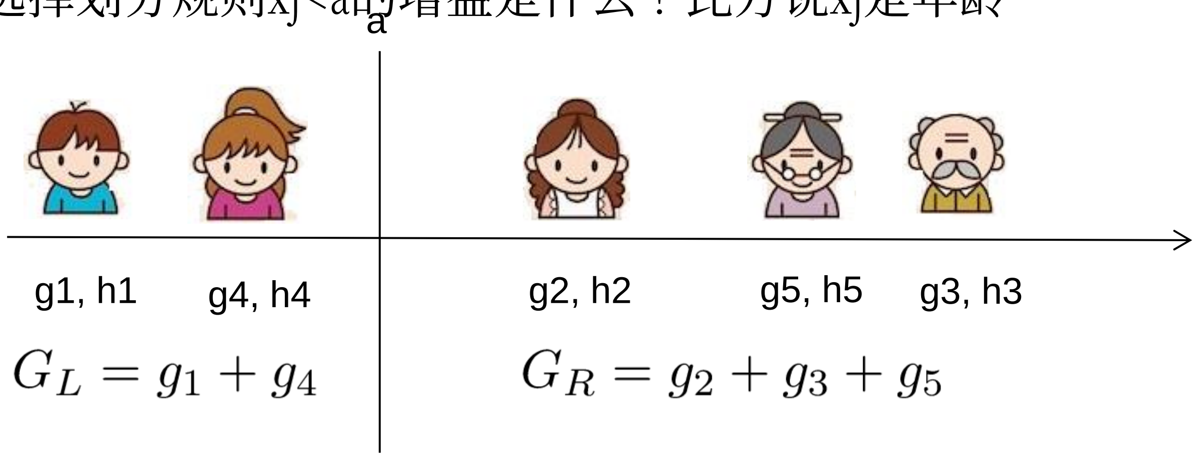
  - Remaining question: how do we find the best split?

疑问：如何找到最优划分点呢？

Efficient Finding of the Best Split 高效地寻找最优化分点

- What is the gain of a split rule $x_j < a$? Say x_j is age

选择划分规则xj<a的增益是什么？比方说xj是年龄



$$G_L = g_1 + g_4 \qquad G_R = g_2 + g_3 + g_5$$

- All we need is sum of g and h in each side, and calculate

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

- Left to right linear scan over sorted instance is enough to decide the best split along the feature

我们所需要的是两边的g和h的总和，并计算

在排序实例上的进行从左到右线性扫描，足以决定在该特征上的最佳划分。

An Algorithm for Split Finding 寻找划分点算法

- For each node, enumerate over all features
  - For each feature, sorted the instances by feature value
  - Use a linear scan to decide the best split along that feature
  - Take the best split solution along all the features

对于每个节点，枚举所有特征
对于每个特征，按特征值对实例进行排序。
使用线性扫描来确定该特征的最佳分割。
采取所有特征中最佳分割方案

时间复杂性分析：深度K树
O（n d K log n）：每个级别，需要O（n log n）排序。有D特征，我们需要做K层
这可以进一步优化（例如使用近似或缓存排序的特征）
可以扩展到非常大的数据集

- Time Complexity growing a tree of depth K
  - It is O(n d K log n): or each level, need O(n log n) time to sort There are d features, and we need to do it for K level
  - This can be further optimized (e.g. use approximation or caching the sorted features)
  - Can scale to very large dataset

那类别型变量呢

What about Categorical Variables?

- Some tree learning algorithm handles categorical variable and continuous variable separately

  一些树学习算法分别处理类别变量和连续变量

  - We can easily use the scoring formula we derived to score split based on categorical variables.

- Actually it is not necessary to handle categorical separately.
  - We can encode the categorical variables into numerical vector using one-hot encoding. Allocate a #categorical length vector

$$z_j = \begin{cases} 1 & \text{if } x \text{ is in category } j \\ 0 & otherwise \end{cases}$$

  - The vector will be sparse if there are lots of categories, the learning algorithm is preferred to handle sparse data

Pruning and Regularization 剪枝和正则化

- Recall the gain of split, it can be negative! 回顾划分后的信息增益，他不能是负的

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

  - When **the training loss reduction** is smaller than **regularization**
  - Trade-off between simplicity and predictivness
- Pre-stopping
  - Stop split if the best split have negative gain如果最佳分裂具有负增益，则停止分裂
  - But maybe a split can benefit future splits.. 但是分裂可能会有利于未来的分裂
- Post-Prunning 将一棵树生长到最大深度，递归地修剪所有负增益的叶子划分点。
  - Grow a tree to maximum depth, recursively prune all the leaf splits with negative gain

Recap: Boosted Tree Algorithm 概括

- Add a new tree in each iteration 每一轮增加一棵新树
- Beginning of each iteration, calculate 每一轮的开始，计算

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

- Use the statistics to greedily grow a tree $f_t(x)$ 使用贪婪算法构建树

$$Obj = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j+\lambda} + \gamma T$$

- Add $f_t(x)$ to the model $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$ 将构建好的模型加入加法模型
  - Usually, instead we do $y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$
  - ϵ is called step-size or shrinkage, usually set around 0.1
  - This means we do not do full optimization in each step and reserve chance for future rounds, it helps prevent overfitting

这意味着我们不会在每一步都做充分的优化，并为未来的回合保留机会，这有助于防止过拟合

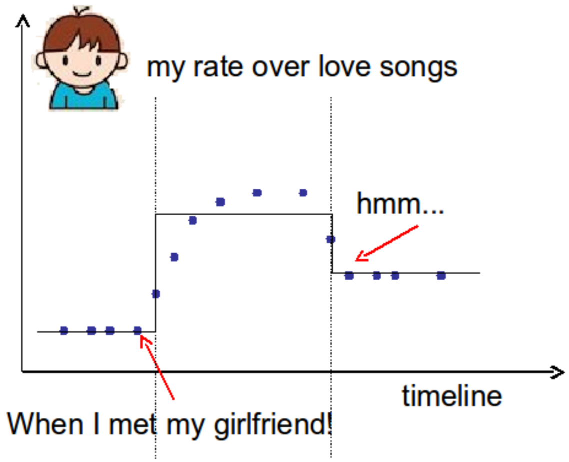
！！！！！！！！！！！！！！！！！！！！！！！！

Outline

- Review of key concepts of supervised learning
- Regression Tree and Ensemble (What are we Learning)
- Gradient Boosting (How do we Learn)
- **Summary**

Questions to check if you really get it

- How can we build a boosted tree classifier to do weighted regression problem, such that each instance have a importance weight?
- Back to the time series problem, if I want to learn step functions over time. Is there other ways to learn the time splits, other than the top down split approach?



Questions to check if you really get it

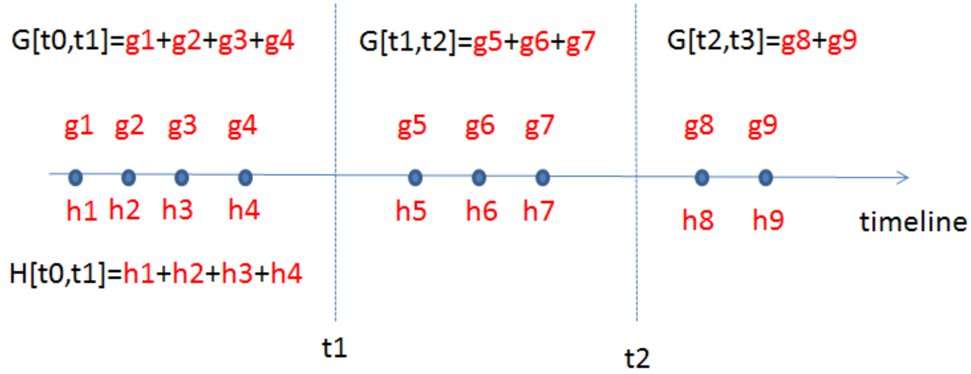
- How can we build a boosted tree classifier to do weighted regression problem, such that each instance have a importance weight?
  - Define objective, calculate g_i, h_i, feed it to the old tree learning algorithm we have for un-weighted version

$$l(y_i, \hat{y}_i) = \frac{1}{2} a_i (\hat{y}_i - y_i)^2 \qquad g_i = a_i(\hat{y}_i - y_i) \quad h_i = a_i$$

  - Again think of separation of model and objective, how does the theory can help better organizing the machine learning toolkit

Questions to check if you really get it

- Time series problem



- All that is important is the structure score of the splits

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

  - Top-down greedy, same as trees
  - Bottom-up greedy, start from individual points as each group, greedily merge neighbors
  - Dynamic programming, can find optimal solution for this case

Summary

- The separation between model, objective, parameters can be helpful for us to understand and customize learning models
- The bias-variance trade-off applies everywhere, including learning in functional space

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

- We can be formal about what we learn and how we learn. Clear understanding of theory can be used to guide cleaner implementation.

Reference

- Greedy function approximation a gradient boosting machine. *J.H. Friedman*
  - *First paper about gradient boosting*
- *Stochastic Gradient Boosting. J.H. Friedman*
  - *Introducing bagging trick to gradient boosting*
- *Elements of Statistical Learning. T. Hastie, R. Tibshirani and J.H. Friedman*
  - *Contains a chapter about gradient boosted boosting*
- Additive logistic regression a statistical view of boosting. *J.H. Friedman T. Hastie R. Tibshirani*
  - *Uses second-order statistics for tree splitting, which is closer to the view presented in this slide*
- Learning Nonlinear Functions Using Regularized Greedy Forest*. R. Johnson and T. Zhang*
  - *Proposes to do fully corrective step, as well as regularizing the tree complexity. The regularizing trick is closed related to the view present in this slide*

- Software implementing the model described in this slide: https://github.com/tqchen/xgboost