Decision Tree and Random Forest

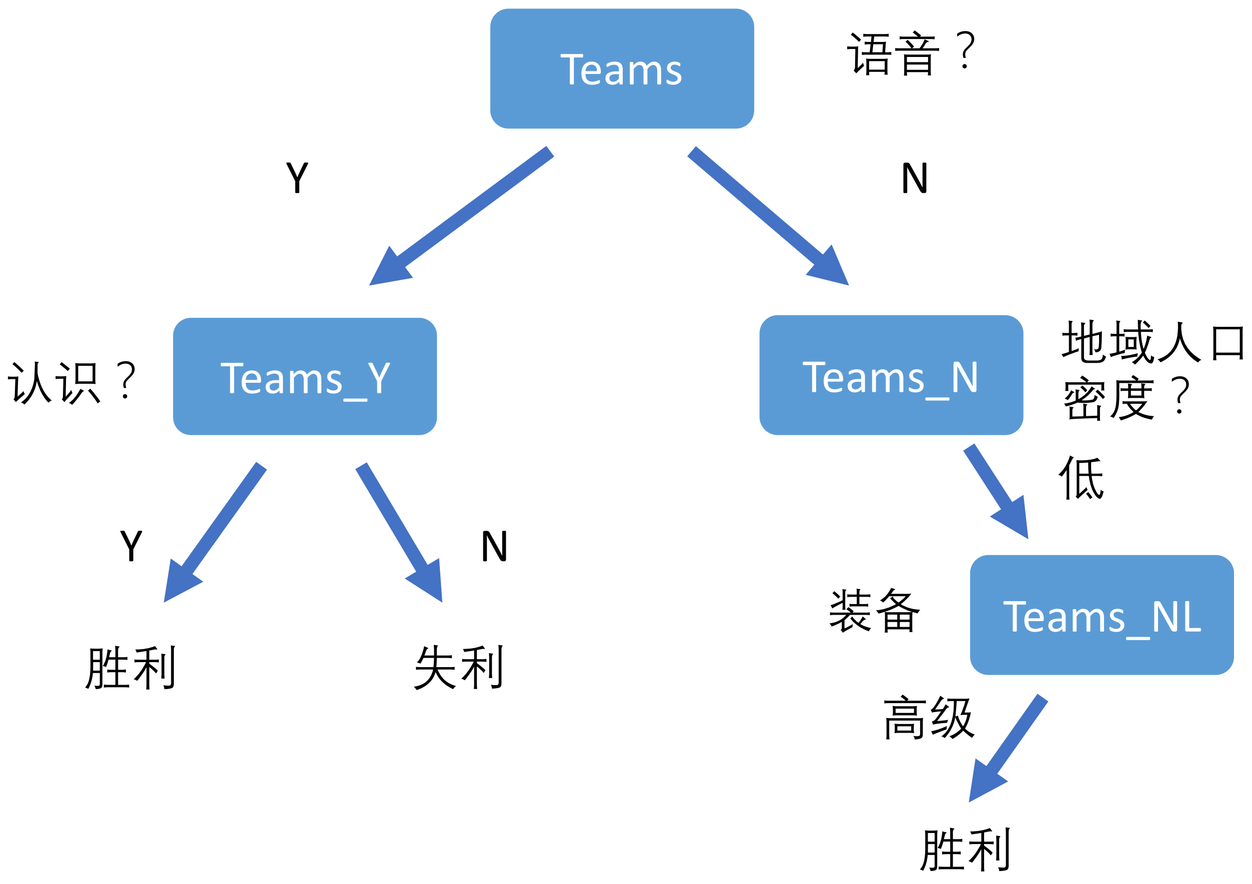
一个神奇的游戏：

我们小时候，可能都玩过这么一个游戏：

问一个问题，回答 是 或者 否 然后来猜你心中想的那个人

http://cn.akinator.com/personnages/jeu?reset=1

What is a Tree?



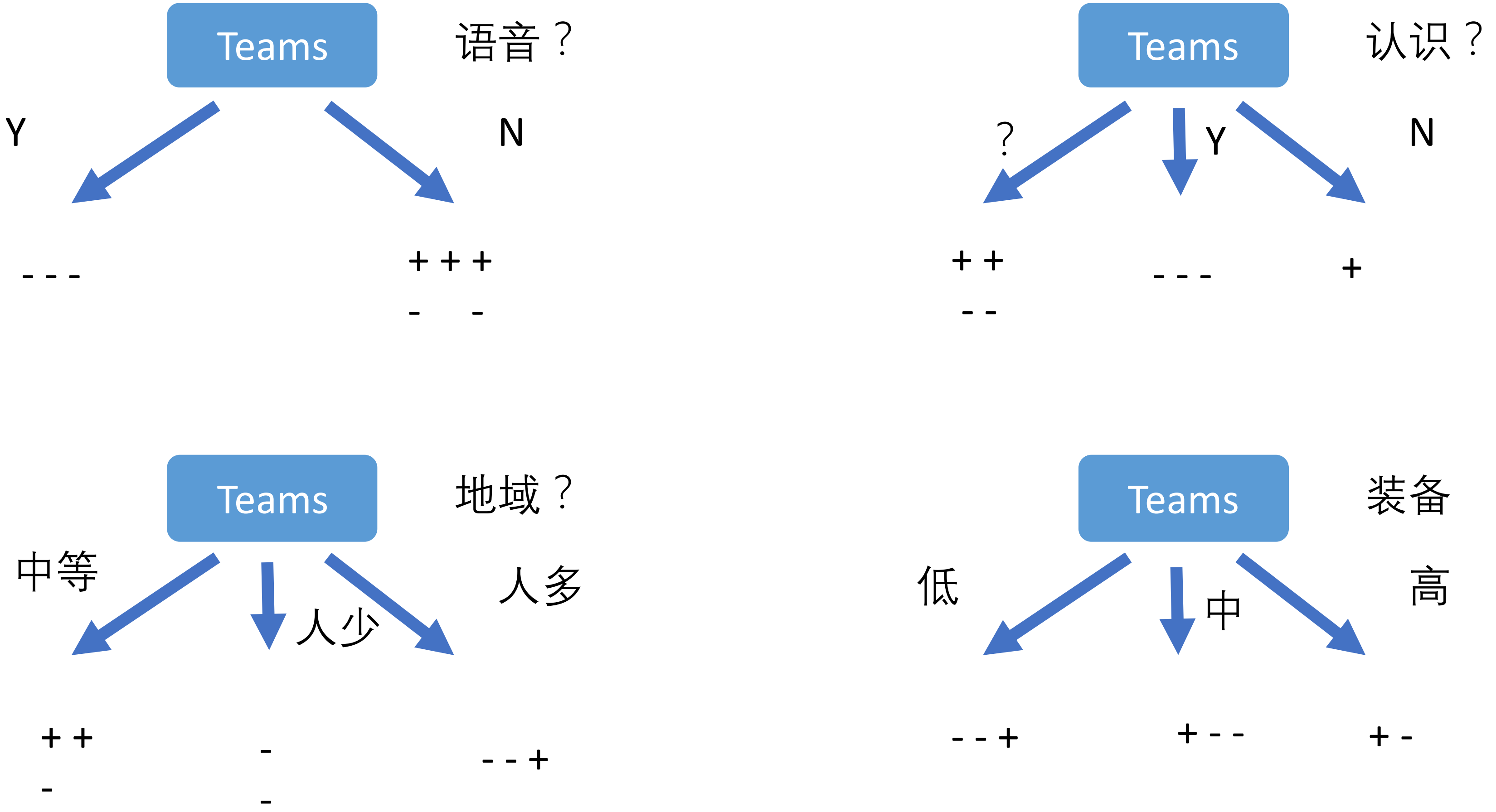
[Picture retrieved from www.dayinhu.com](http://www.dayinhu.com)

Take a look at trees

让我们开始问问题！

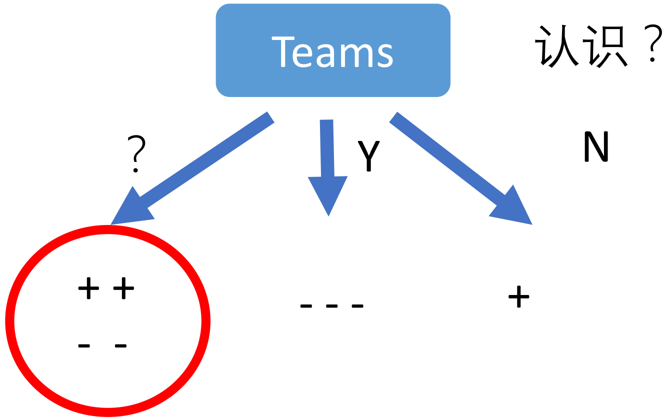
Win_the_game	know_each_other	Talk	Population	equipment
No	Not sure	Yes	few	low_level
No	Yes	Yes	many	low_level
Yes	Not sure	No	many	low_level
Yes	No	No	average	high_lavel
Yes	Not sure	No	average	medium_level
No	Yes	No	few	high_lavel
No	Yes	No	average	high_lavel
No	Not sure	Yes	many	medium_level

语音？

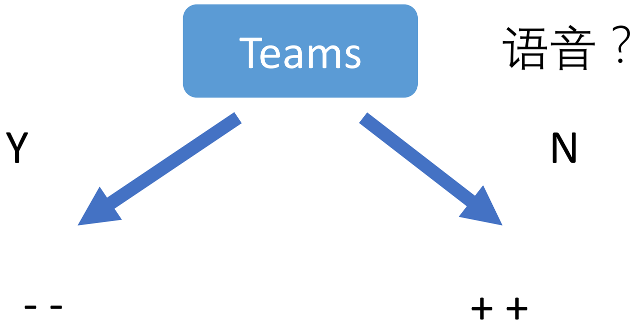


语音 ： 3分
认识 ： 4分
地域 ： 2分
装备 ： 0分

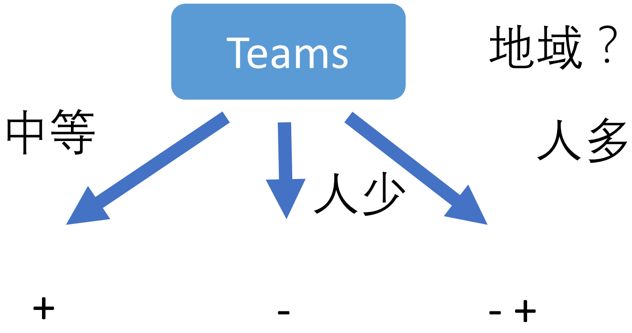
选了认识以后？



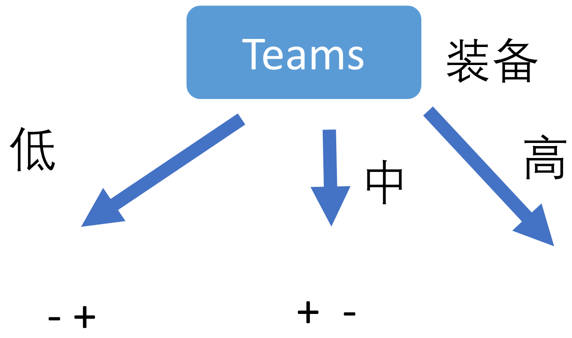
Win_the_game	know_each_other	Talk	Population	equipment
No	Not sure	Yes	few	low_level
No	Yes	Yes	many	low_level
Yes	Not sure	No	many	low_level
Yes	No	No	average	high_lavel
Yes	Not sure	No	average	medium_level
No	Yes	No	few	high_lavel
No	Yes	No	average	high_lavel
No	Not sure	Yes	many	medium_level



4 分



2 分



0 分

到底应该从哪个问题开始分？

到底哪个问题分的最好？

到底哪个问题分的结果最纯洁/纯粹？

什么叫纯粹？

$$Entropy = \sum_j -p_j * log_2(p_j)$$

$$Gini = 1 - \sum_j p_j^2$$

From Information Theory

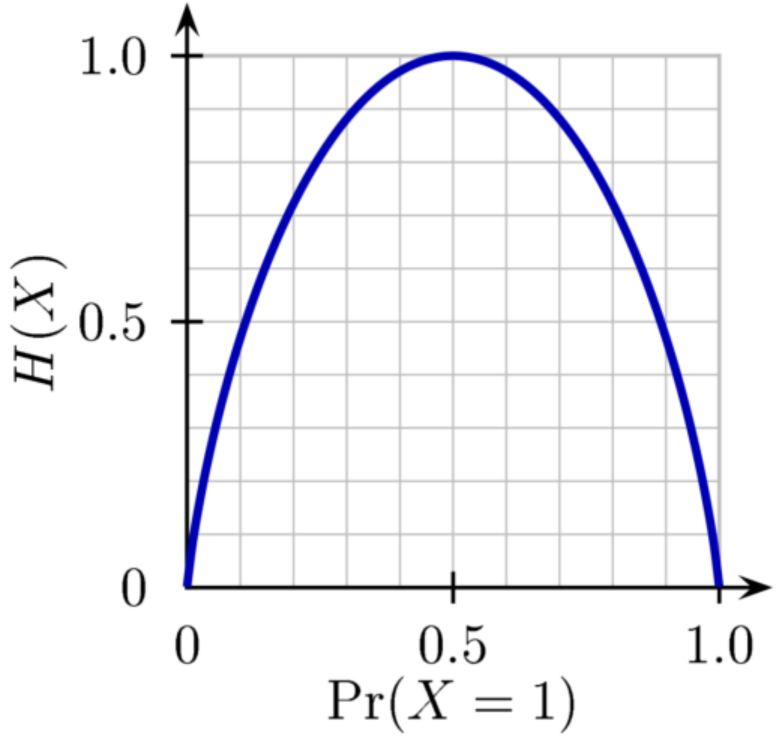
Gini Impurity

越低越好

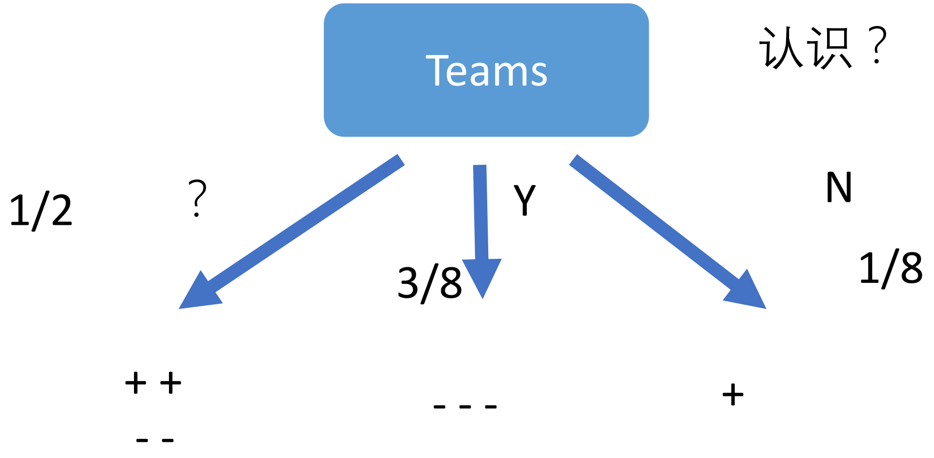
$$Entropy = \sum_j -p_j * log_2(p_j)$$

注意：

这里算的熵是对于每一个叶子，要想评价一个node，必须把每一个叶子按权重比例加起来



If j =2

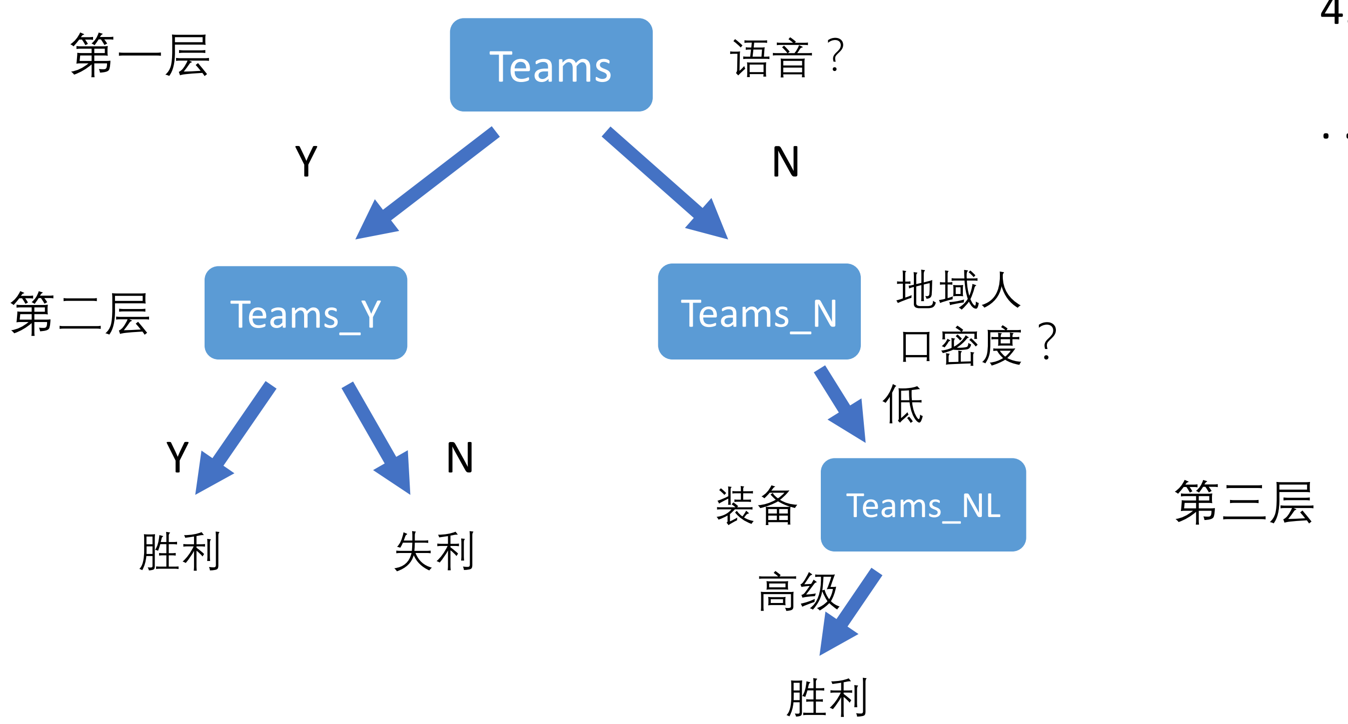


Entropy = 0.5*1 +3/8 * 0 + 1/8 * 0
= 0.5

Picture retrieved from wiki：Binary entropy

How to deal with overfitting?

1. 限制能分几层 （number of split）



2. 最少可以分的数量
3. 每个叶子含有的样本数
4. 至少要增加多少information （information gain）

. . .

```
DTree=DecisionTreeClassifier(max_depth=5,criterion='gini',min_samples_split=2)
```

怎么把 Decision Tree 变的更强大 ？

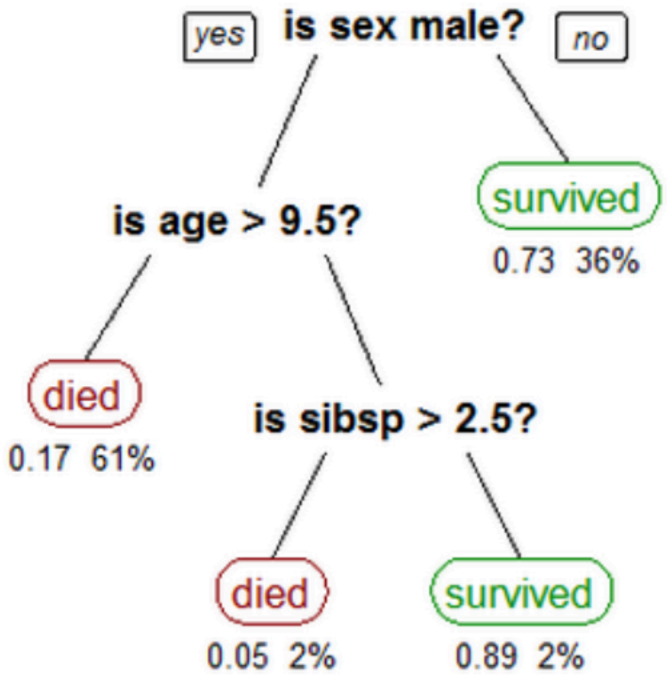
- Bagging (Breiman, 1996) Bagging
- Boosting (Freund & Schapire, 1996) Boosting
- AdaBoost (Freund & Schapire, 1997) Boosting
- Random Forrest (Breiman, 1999) Bagging
- Gradient Boosting (Friedman et al, 2000) Boosting

这些方法就叫做 集成算法 （Ensemble Learning）

Regression Tree 回归树

Outputs a number, 意思就是结果出来的是一个数

对比之前 Classification Tree 出来的是一个类别 （赢了 or 输了）



Titanic Survival Example

Picture source from wikipedia

Bagging (Bootstrap aggregating)

假设有10个样本：

[0,1,2,3,4,5,6,7,8,9]

有放回的取10个样本：

[2,3,1,5,2,5,6,7,7,4]

[1,1,7,0,2,4,5,5,2,9]

...

Bagging (Bootstrap aggregating)

当样本量足够大时，取出的样本数量接近于63.2%

$$P=1-\left(1-\frac{1}{N}\right)^{N}$$

If N =10 p= 65.1%

If N= 100 p=63.2%

好处：

- 模型更稳定
- 防止过拟合
- 提高准确率

Bagging (Bootstrap aggregating)

如何决定结果：少数服从多数

What about Random Forrest?

（随机 森林） = 随机 + 树 +树 +树…

- ➢ 用Bagging 来选取样本
- ➢ 用Bagging 来选取特征值

图片来源：http://csctalkradio.com/podcast/free-4-friday-americans-disheartened-forrest-trees/

Boosting

本意：助推

“

… an efficient algorithm for converting *relatively poor hypotheses* into *very good hypotheses…*”

— Thoughts on Hypothesis Boosting [PDF], 1988

能不能联合一帮弱的变成一个强的 ？

Ada Boosting was invented in 1995

“

Boosting refers to this general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb.

— A decision-theoretic generalization of on-line learning and an application to boosting [PDF], 1995

Reference: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Ada Boosting.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
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (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 $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

依次列出m 个分类器：

- 每次对于错误的样本，增加权重 （起始的时候权重 都是 1/N） w_i
- 对于预测的好的分类器 （err 小）给予更多的权重 α_m
- 以少数服从多数的原则进行分类
- 最初是二分类 { -1, 1 }

Reference from: T. Hastie, R. TibshiRani, J. Friedman The element of statistical learning, 2nd Edition

Ada Boosting 更深层次的理解

其实 Ada Boosting 所做的就是在做 Forward Stage Learning， 然后选取了 exponential 作为Loss Function

$$Y_{\mathrm{i}} = \mathrm{fi}(\mathrm{x}) + \beta * b(xi; r)$$

每一步都找一个模型 $b(xi;r)$ 去 fit 和真实值之间的差距

Exponential Loss Function：

$$L\big(y, f(x)\big) = \exp(-yf(x))$$

Ada Boosting 就是相当于用了 Exponential 作为Loss Function

$$\alpha = 2\beta$$

Reference from: T. Hastie, R. TibshiRani, J. Friedman The element of statistical learning, 2nd Edition

Gradient Boosting

Ada Boosting 的一种通用形式

- Invent Adaboost, the first successful boosting algorithm
  [Freund et al., 1996, Freund and Schapire, 1997]

- Formulate Adaboost as gradient descent with a special loss function
  [Breiman et al., 1998, Breiman, 1999]

- Generalize Adaboost to Gradient Boosting in order to handle a variety of loss functions
  [Friedman et al., 2000, Friedman, 2001]

Gradient Boosting

Consider you have series of predictions of x : $F(x_1)$, $F(x_2)$, $F(x_3)$, $F(x_4)$, … $F(x_N)$,

But they are not quite accurate when comparing to y

For Example:

y_1=1.5 but $F(x_1)$ =1.34

y_2=0.8 but $F(x_2)$ =0.82

y_1=2.0 but $F(x_1)$ =1.91

…

You may choose to ignore it

Or fit the left off with some model

Reference from slice of Cheng Li, A gentle introduction to gradient boosting Northeastern University

Gradient Boosting

Say if I use a model h to fit the difference between y and F(x):

Use $h(x_1)$ to fit $y_1 - F(x_1)$ 这个叫残差

$h(x_2)$ to fit $y_2 - F(x_2)$

…

更新 $F'(x) = F(x) + h(x)$

但是如果 $F'(x)$还是不是完美的预测怎么办？

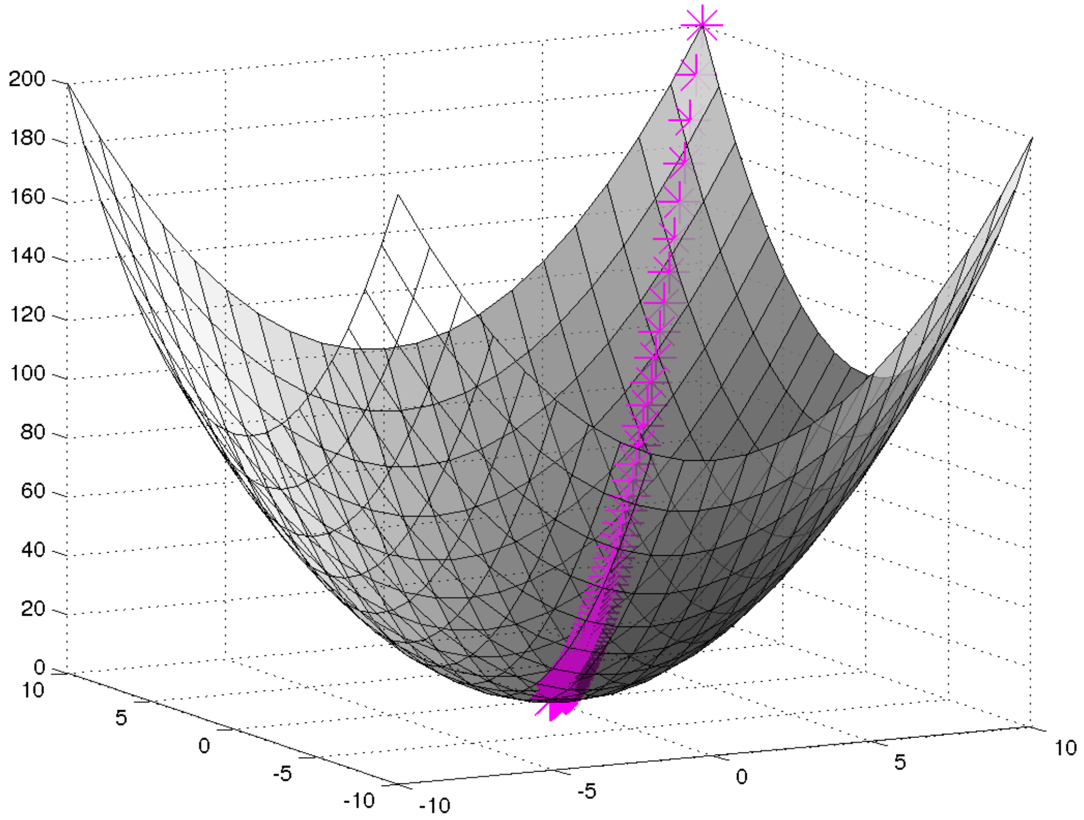
继续做一个模型 b(x) 去 fit 新的残差 $y_1 - F(x_1) - h(x_1)$

这就是Boosting 的哲学

为什么叫 Gradient Boosting？

1. 大家都听说过 梯度下降 （gradient decent）吧：

怎么找到最小的位置：往 gradient 的反方向一小步一小步移动



$$\theta_i = \theta_i - \alpha \frac{\partial J}{\partial \theta_i}$$

J 在这里就是 Loss Function

https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/35389/versions/1/screenshot.png

如果我们用最小方差 作为 Loss Function， 或者说:

$$J = \frac{1}{2}\Sigma\,(\,y_i\text{-}F(x_i)\,)^2$$

$$\frac{\partial J}{\partial F(xi)} = F(x_i) - y_i$$

$$-\frac{\partial J}{\partial F(xi)} = y_i - F(x_i)$$

Negative Gradient 相当于 残差

那么每次模型去 fit 残差   就是   模型去 fit Negative Gradient

每次根据残差去更新 F'(x)   就是   根据 Negative Gradient 去更新 F'(x)

其实是用了 Gradient Descent 的思想 所以叫 Gradient Boosting （Gradient + Boosting）

当然 Gradient Boosting 所用的 Loss Function 不至于最小方差， 比如可以是 $\exp(-yf(x))$


谢谢观看！