Gradient boosting

GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE
BY JEROME H. FRIEDMAN
Stanford University

Slides from Dr. Cheng Li Northeastern University

ESLL, CH 10

Why and how did researchers invent Gradient Boosting?

- 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]

- Breiman, L. (1999). Prediction games and arcing algorithms.

Neural computation, 11(7):1493–1517.

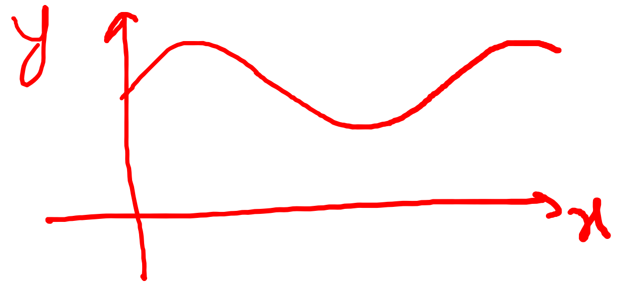
- Breiman, L. et al. (1998). Arcing classifier (with discussion and a rejoinder by the author).

The annals of statistics, 26(3):801–849.

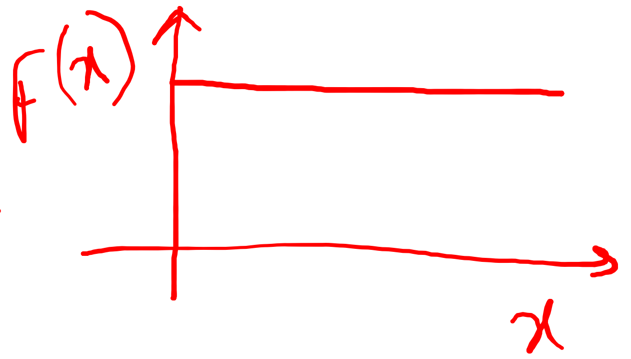
Gradient Boosting for Regression

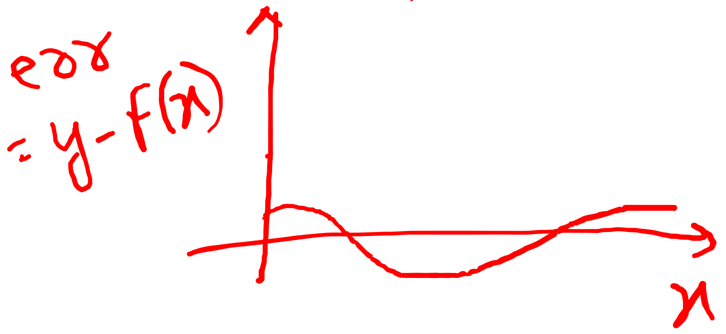
Let's play a game...
You are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.
Suppose your friend wants to help you and gives you a model F.
You check his model and find the model is good but not perfect.
There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$... How can you improve this model?
Rule of the game:

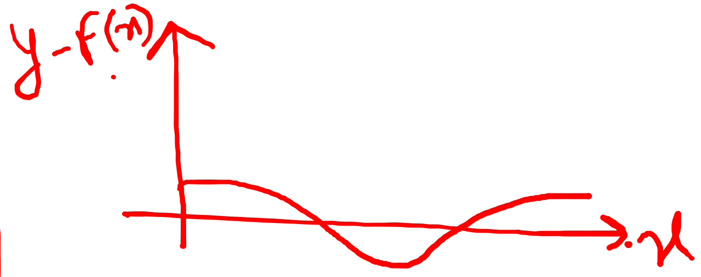
- You are not allowed to remove anything from F or change any parameter in F.



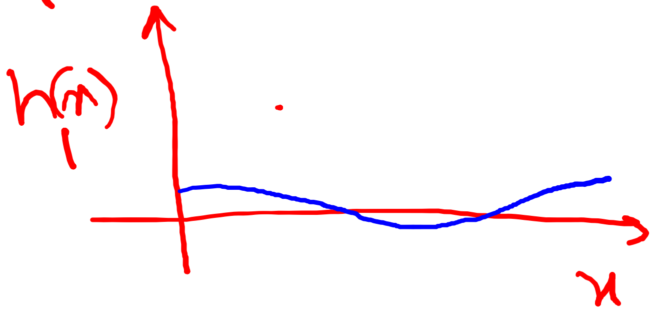
Fixed

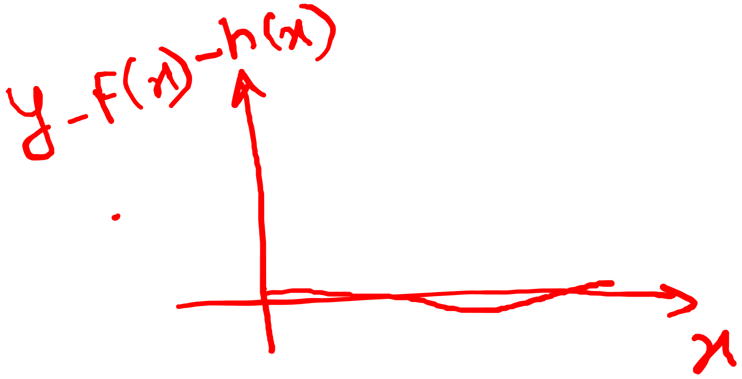


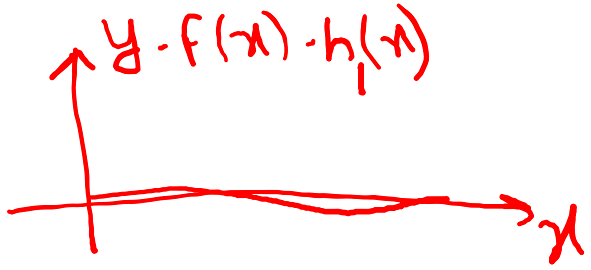


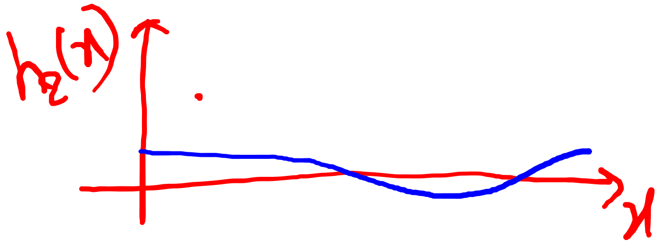


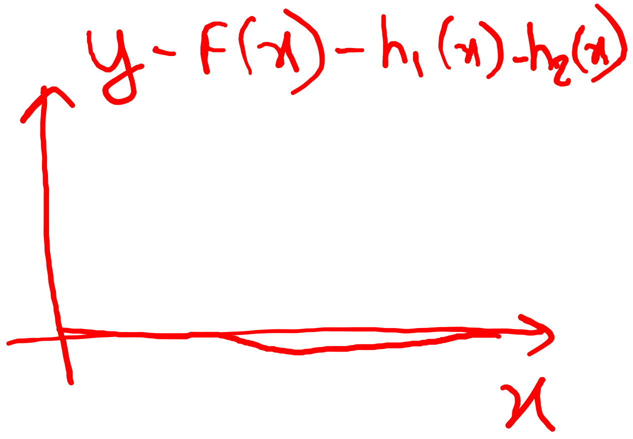
Fixed











Final model: $F(x) + h_1(x) + h_2(x) + \ldots\ldots$

Gradient Boosting for Regression

Let's play a game...
You are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.
Suppose your friend wants to help you and gives you a model F.
You check his model and find the model is good but not perfect.
There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$... How can you improve this model?

Gradient Boosting for Regression

Let's play a game...

You are given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.

Suppose your friend wants to help you and gives you a model F.

You check his model and find the model is good but not perfect.

There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$... How can you improve this model?

Rule of the game:

- You are not allowed to remove anything from F or change any parameter in F.
- You can add an additional model (regression tree) h to F, so the new prediction will be $F(x) + h(x)$.

- We can further add another model by learning it from

(X, ?)

Gradient Boosting for Regression

Simple solution:

You wish to improve the model such that

$$\underline{F}(x_1) + \underline{h}(x_1) = y_1$$
$$F(x_2) + h(x_2) = y_2$$
$$\dots$$
$$F(x_n) + h(x_n) = y_n$$

we learn F using dataset $D = \{x, y\}$

we add another mode h

such that $F(x_1) + h(x_1)$ very close to G.T

$F(x_1) + h(x_1) \simeq y_1$

$h(x_1)$ should.

$h(x_1) = y_1 - F(x_1)$

$$D' = \{ [x_1, y_1 - F(x_1)] \\ [x_2, y_2 - F(x_2)] \\ \vdots \\ [x_n, y_n - F(x_n)]$$

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$
\begin{aligned}
h(x_1) &= y_1 - F(x_1) \\
h(x_2) &= y_2 - F(x_2) \\
&\dots \\
h(x_n) &= y_n - F(x_n)
\end{aligned}
$$

$$\frac{1}{2}\sum_i \left(y_i - f(x_i)\right)^2$$

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$\begin{aligned}
h(x_1) &= y_1 - F(x_1)\\
h(x_2) &= y_2 - F(x_2)\\
&\dots\\
h(x_n) &= y_n - F(x_n)
\end{aligned}$$

Can any regression tree h achieve this goal perfectly?

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$\begin{aligned} h(x_1) &= y_1 - F(x_1) \\ h(x_2) &= y_2 - F(x_2) \\ &\ldots \\ h(x_n) &= y_n - F(x_n) \end{aligned}$$

Can any regression tree h achieve this goal perfectly?
Maybe not....

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$\begin{aligned} h(x_1) &= y_1 - F(x_1) \\ h(x_2) &= y_2 - F(x_2) \\ &\dots \\ h(x_n) &= y_n - F(x_n) \end{aligned}$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$
\begin{aligned}
h(x_1) &= y_1 - F(x_1) \\
h(x_2) &= y_2 - F(x_2) \\
&\dots \\
h(x_n) &= y_n - F(x_n)
\end{aligned}
$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\dots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?
Just fit a regression tree h to data
✓ $(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \dots, (x_n, y_n - F(x_n))$

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?
Just fit a regression tree h to data
$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \ldots, (x_n, y_n - F(x_n))$
Congratulations, you get a better model!

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.
The role of h is to compensate the shortcoming of existing model F.

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory,

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

We are improving the predictions of training data, is the procedure also useful for test data?

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

We are improving the predictions of training data, is the procedure also useful for test data?

Yes! Because we are building a model, and the model can be applied to test data as well.

We want to predict a person's age based on whether they play video games, enjoy gardening, and their preference on wearing hats. Our objective is to minimize squared error. We have these nine training samples to build our model.

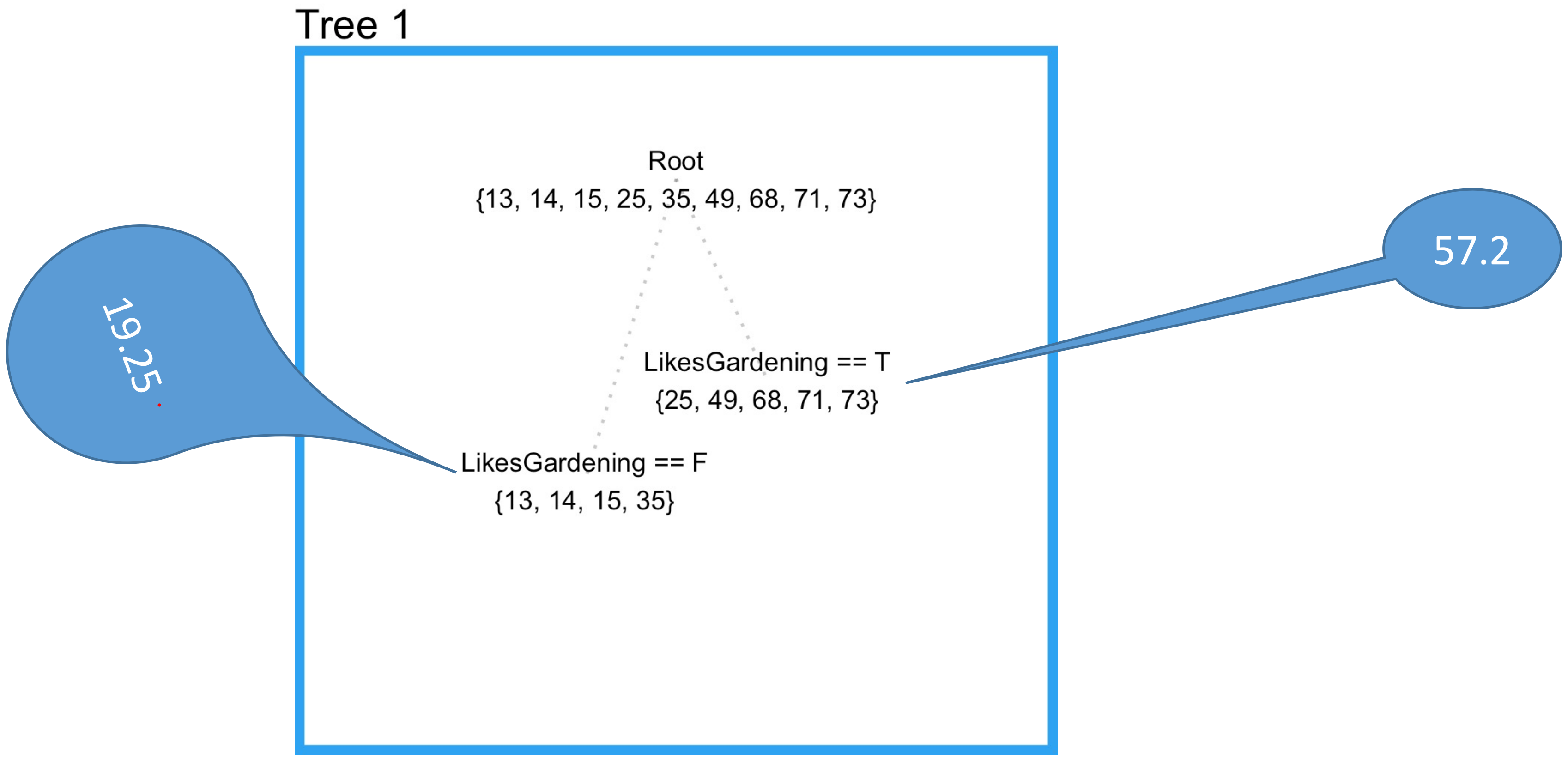
PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE
	?	Fals	Fal	Fal

$\rightarrow$ Test.

Intuitively, we might expect
– The people who like gardening are probably older
– The people who like video games are probably younger
– *LikesHats* is probably just random noise

Decision tree

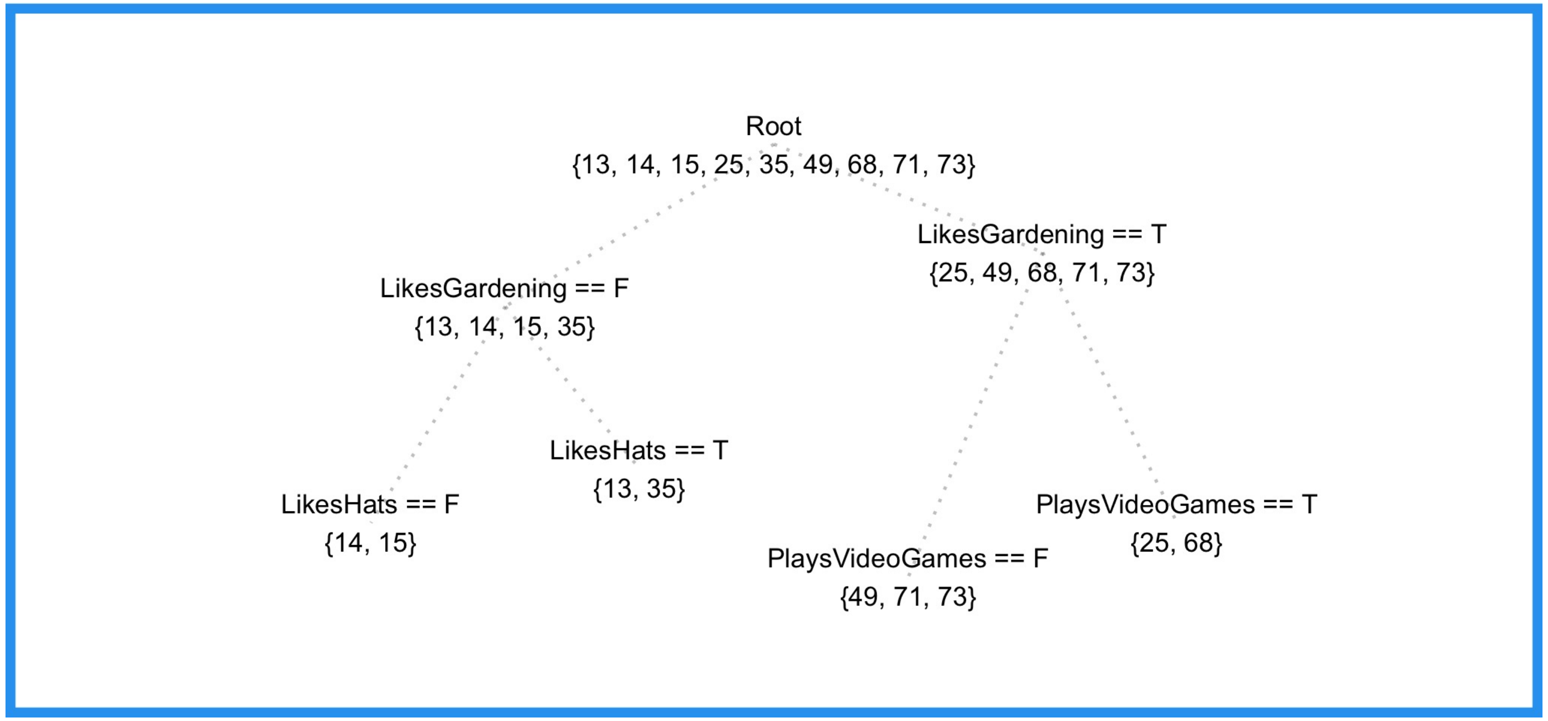
- This is nice, but it's missing valuable information from the feature *LikesVideoGames.*



PersonID	Age	height	weight	Color of cloth
1	13	4	30	R
2	14	4	30	G
3	15	4	40	B
4	25	5	40	Y
5	35	4	40	B
6	49	5	70	R
7	68	5	40	R
8	71	5	80	B
9	73	5	90	G

- Let’s try letting terminal nodes have 2 samples.

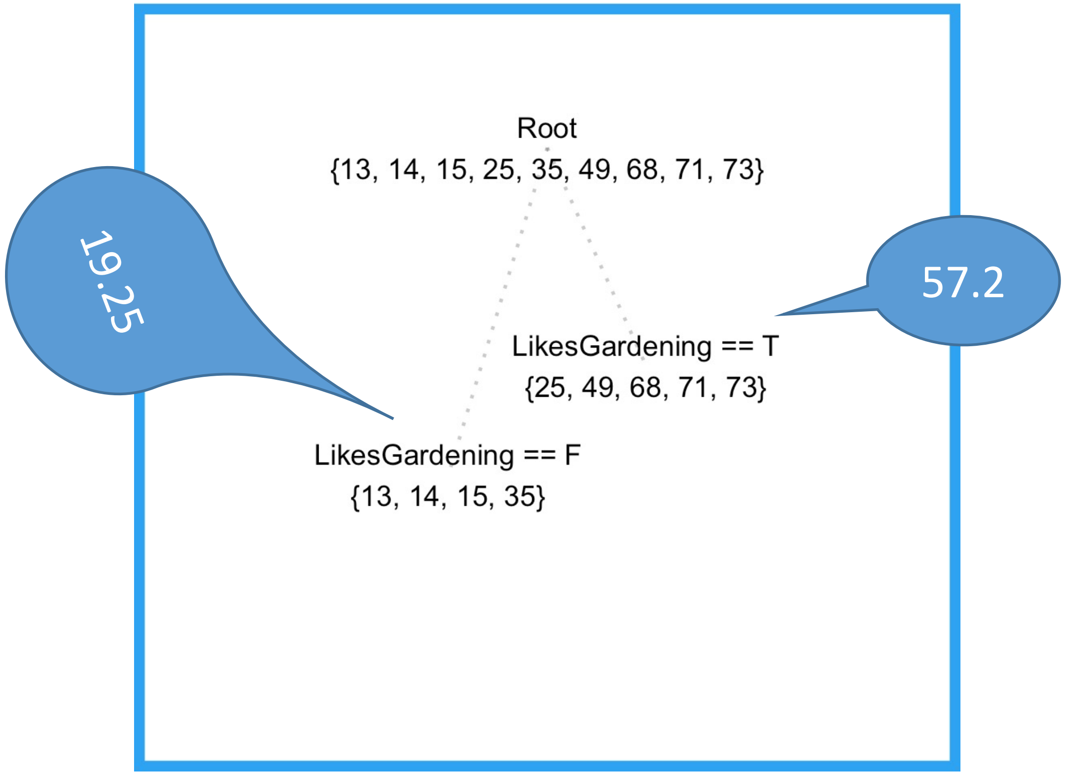
Overfit Tree



- Here we pick up some information from *PlaysVideoGames* but we also pick up information from *LikesHats* – a good indication that we’re overfitting and our tree is splitting random noise.

Suppose we measure the training errors from our first tree.

Tree 1



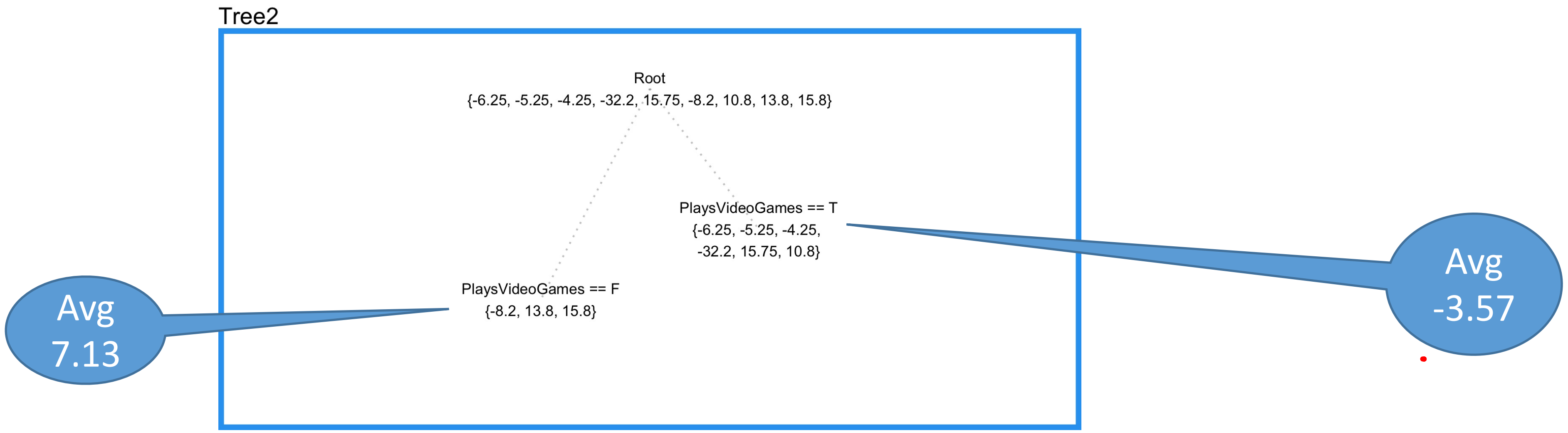
	Y	$F(x)$	Y - F
PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.20	-32.20
5	35	19.25	15.75
6	49	57.20	-8.20
7	68	57.20	10.80
8	71	57.20	13.80
9	73	57.20	15.80

- We have computed F, and the residue Y-F
- Now fit a second tree using {X, Y-F}

$\checkmark$
$Y-F-H$

LikesHats	PlaysVideoGames	LikesGardening	Y-F
TRUE	TRUE	FALSE	-6.25
FALSE	TRUE	FALSE	-5.25
FALSE	TRUE	FALSE	-4.25
TRUE	TRUE	TRUE	-32.20
TRUE	TRUE	FALSE	15.75
FALSE	FALSE ✗	TRUE	-8.20
TRUE	TRUE	TRUE	10.80
FALSE	FALSE ✗	TRUE	13.80
TRUE	FALSE ✗	TRUE	15.80

- Now we can fit a second regression tree to the residuals of the first tree.



- Notice that this tree does **not** include *LikesHats* even though our overfitted regression tree above did.
- The reason is because this regression tree is able to consider *LikesHats* and *PlaysVideoGames* with respect to all the training samples, contrary to our overfit regression tree which only considered each feature inside a small region of the input space, thus allowing random noise to select *LikesHats* as a splitting feature.

- Now we can improve the predictions from our first tree by adding the “error-correcting” predictions from this tree.
-

	Y	F	Y-F	H	F+H	Y-F-H
ID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.57	15.68	-2.68
2	14	19.25	-5.25	-3.57	15.68	-1.68
3	15	19.25	-4.25	-3.57	15.68	-0.68
4	25	57.20	-32.20	-3.57	53.63	-28.63
5	35	19.25	15.75	-3.57	15.68	+19.32
6	49	57.20	-8.20	7.13	64.33	-15.33
7	68	57.20	10.80	-3.57	53.63	+14.37
8	71	57.20	13.80	7.13	64.33	+6.67
9	73	57.20	15.80	7.13	64.33	+8.67

MSE Tree 1
1994 ✓
MSE combined
1764 ✓

Final prediction: $F(x) + h_1(x) + h_2(x) + \cdots + h_n(x)$

Gradient Boosting for Regression

Gradient Descent

Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

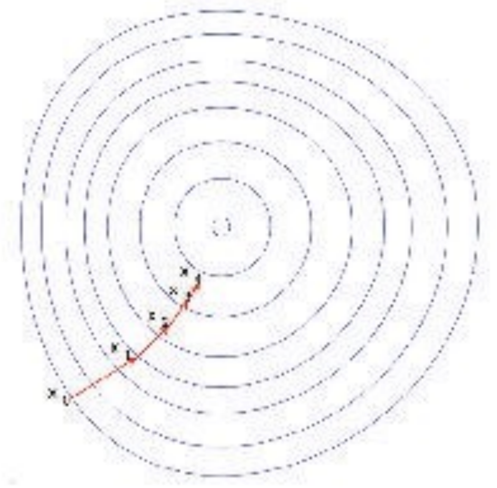


Figure: Gradient Descent. Source: `http://en.wikipedia.org/wiki/Gradient_descent`

Gradient Boosting for Regression

How is this related to gradient descent?

Why GD? Later.

Loss function $L(y, F(x)) = (y - F(x))^2/2$
We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), ..., F(x_n)$.
Notice that $F(x_1), F(x_2), ..., F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

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

Gradient Boosting for Regression

How is this related to gradient descent?

$$F(x_i) := F(x_i) + h(x_i)$$
$$F(x_i) := F(x_i) + y_i - F(x_i)$$
$$F(x_i) := F(x_i) - 1\frac{\partial J}{\partial F(x_i)}$$
$$\theta_i := \theta_i - \rho\frac{\partial J}{\partial \theta_i}$$

Note that
h() is approximating
Y-F

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$
\begin{aligned}
h(x_1) &= y_1 - F(x_1) \\
h(x_2) &= y_2 - F(x_2) \\
&\ldots \\
h(x_n) &= y_n - F(x_n)
\end{aligned}
$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient} \qquad y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient}$$

$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

So we are actually updating our model using **gradient descent**!

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient}$$

$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

So we are actually updating our model using **gradient descent**!
It turns out that the concept of **gradients** is more general and useful than the concept of **residuals**. So from now on, let's stick with gradients. The reason will be explained later.

Gradient Boosting for Regression

Regression with square Loss

Let us summarize the algorithm we just derived using the concept of gradients. Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

```
start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:
    calculate negative gradients $-g(x_i)$
    fit a regression tree $h$ to negative gradients $-g(x_i)$
    $F := F + \rho h$, where $\rho = 1$
```

Gradient Boosting for Regression

Regression with square Loss

Let us summarize the algorithm we just derived using the concept of gradients. Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

```
start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:
    calculate negative gradients $-g(x_i)$
    fit a regression tree $h$ to negative gradients $-g(x_i)$
    $F := F + \rho h$, where $\rho = 1$
```

The benefit of formulating this algorithm using gradients is that it allows us to consider other loss functions and derive the corresponding algorithms in the same way.

Gradient Boosting for Regression

Loss Functions for Regression Problem

Why do we need to consider other loss functions? Isn't square loss good enough?

Gradient Boosting for Regression

Loss Functions for Regression Problem

Square loss is:

✓ Easy to deal with mathematically

× Not robust to outliers

Outliers are heavily punished because the error is squared.

Example:

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

Consequence?

Gradient Boosting for Regression

Loss Functions for Regression Problem

Square loss is:

✓ Easy to deal with mathematically

× Not robust to outliers
Outliers are heavily punished because the error is squared.
Example:

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

Consequence?
Pay too much attention to outliers. Try hard to incorporate outliers into the model. Degrade the overall performance.

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

- Huber loss (more robust to outliers)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & |y - F| \leq \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

Gradient Boosting for Regression

Loss Functions for Regression Problem

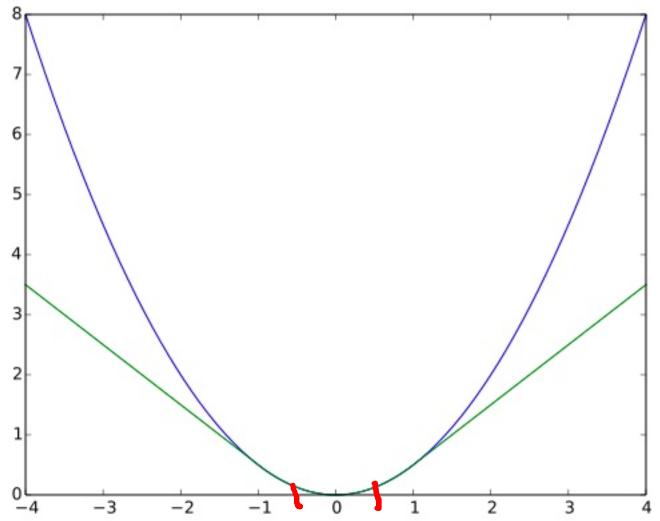
- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

- Huber loss (more robust to outliers)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & |y - F| \le \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
Square loss	0.005	0.02	0.125	5.445
Absolute loss	0.1	0.2	0.5	3.3
Huber loss($\delta = 0.5$)	0.005	0.02	0.125	1.525



Huber loss (green) and squared error loss (blue) as a function of y - f(x). Delta = 1

Gradient Boosting for Regression

Regression with Absolute Loss

Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = sign(y_i - F(x_i))$$

start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$

iterate until converge:

```
    calculate gradients $-g(x_i)$
    fit a regression tree $h$ to negative gradients $-g(x_i)$
    $F := F + \rho h$
```

$|y - F(x)|$

$(y - F(x)), y - F(x)$

Gradient Boosting for Regression

Regression with Huber Loss

Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$$

$$= \begin{cases} y_i - F(x_i) & |y_i - F(x_i)| \leq \delta \\ \delta sign(y_i - F(x_i)) & |y_i - F(x_i)| > \delta \end{cases}$$

```
start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:
    calculate negative gradients $-g(x_i)$
    fit a regression tree $h$ to negative gradients $-g(x_i)$
    $F := F + \rho h$
```

Gradient Boosting for Regression

Regression with loss function L: general procedure

Give any differentiable loss function L

```
start with an initial model, say $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:
    calculate negative gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$
    fit a regression tree $h$ to negative gradients $-g(x_i)$
    $F := F + \rho h$
```

Gradient Boosting for Regression

Regression with loss function L: general procedure

Give any differentiable loss function L

```
start with an initial model, say $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:
    calculate negative gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$
    fit a regression tree $h$ to negative gradients $-g(x_i)$
    $F := F + \rho h$
```

In general,

$$\textit{negative gradients} \not\Leftrightarrow \textit{residuals}$$

We should follow negative gradients rather than residuals. Why?

Gradient Boosting for Regression

Negative Gradient vs Residual: An Example

Huber loss

$$L(y,F) = \begin{cases} \frac{1}{2}(y-F)^2 & |y-F| \le \delta \\ \delta(|y-F| - \delta/2) & |y-F| > \delta \end{cases}$$

Update by Negative Gradient:

$$h(x_i) = -g(x_i) = \begin{cases} y_i - F(x_i) & |y_i - F(x_i)| \le \delta \\ \delta sign(y_i - F(x_i)) & |y_i - F(x_i)| > \delta \end{cases}$$

Update by Residual:

$$h(x_i) = y_i - F(x_i)$$

Difference: negative gradient pays less attention to outliers.

Gradient Boosting for Regression

Summary of the Section

- Fit an additive model $F = \sum_t \rho_t h_t$ in a forward stage-wise manner.
- In each stage, introduce a new regression tree h to compensate the shortcomings of existing model.
- The "shortcomings" are identified by negative gradients.
- For any loss function, we can derive a gradient boosting algorithm.
- Absolute loss and Huber loss are more robust to outliers than square loss.

Questions

- Which loss – squared error, absolute error or Huber, will you prefer?
- Can AdaBoost be parallelized i.e. learning all different classifiers parallely?
- Can bagging and boosting be combined?