Lab #10: Demonstration of AdaBoost

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader, and Chris Tanner

Our Data

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Bagging

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

1. Shuffle (i.e., bootstrap the data)
2. Train a new decision tree $\mathbf{T_i}$

Bagging

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Do **N** times

1. Shuffle (i.e., bootstrap the data)
2. Train a new decision tree T_i

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

Bagging

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Do **N** times

1. Shuffle (i.e., bootstrap the data)
2. Train a new decision tree T_i

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

Testing Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	158	?

Bagging

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Do **N** times

1. Shuffle (i.e., bootstrap the data)
2. Train a new decision tree T_i

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

Testing Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	158	?

Take a majority vote from $\{T_1, T_2, T_3, \ldots, T_N\}$

Boosting

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

1. Shuffle (i.e., bootstrap the data)
2. Select a random subset of P_i features
3. Train a new decision tree $\mathbf{T_i}$

Boosting

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Do **N** times

1. Shuffle (i.e., bootstrap the data)
2. Select a random subset of P_i features
3. Train a new decision tree $\mathbf{T_i}$

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

Boosting

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Do **N** times

1. Shuffle (i.e., bootstrap the data)
2. Select a random subset of P_i features
3. Train a new decision tree $\mathbf{T_i}$

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

Testing Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	158	?

Take a majority vote from $\{T_1, T_2, T_3, \ldots, T_N\}$

Ideas

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

"Fool me once, shame on … shame on you. Fool me – you can't get fooled again" –George W. Bush

"Fool me once, shame on you; fool me twice, shame on me" –Proverb

Ideas

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

"Fool me once, shame on … shame on you. Fool me – you can't get fooled again" –George W. Bush

"Fool me once, shame on you; fool me twice, shame on me" –Proverb

Let's learn from our mistakes!

Gradient Boosting

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

$$T = \sum_h \lambda_h T_H$$

Gradient Boosting

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

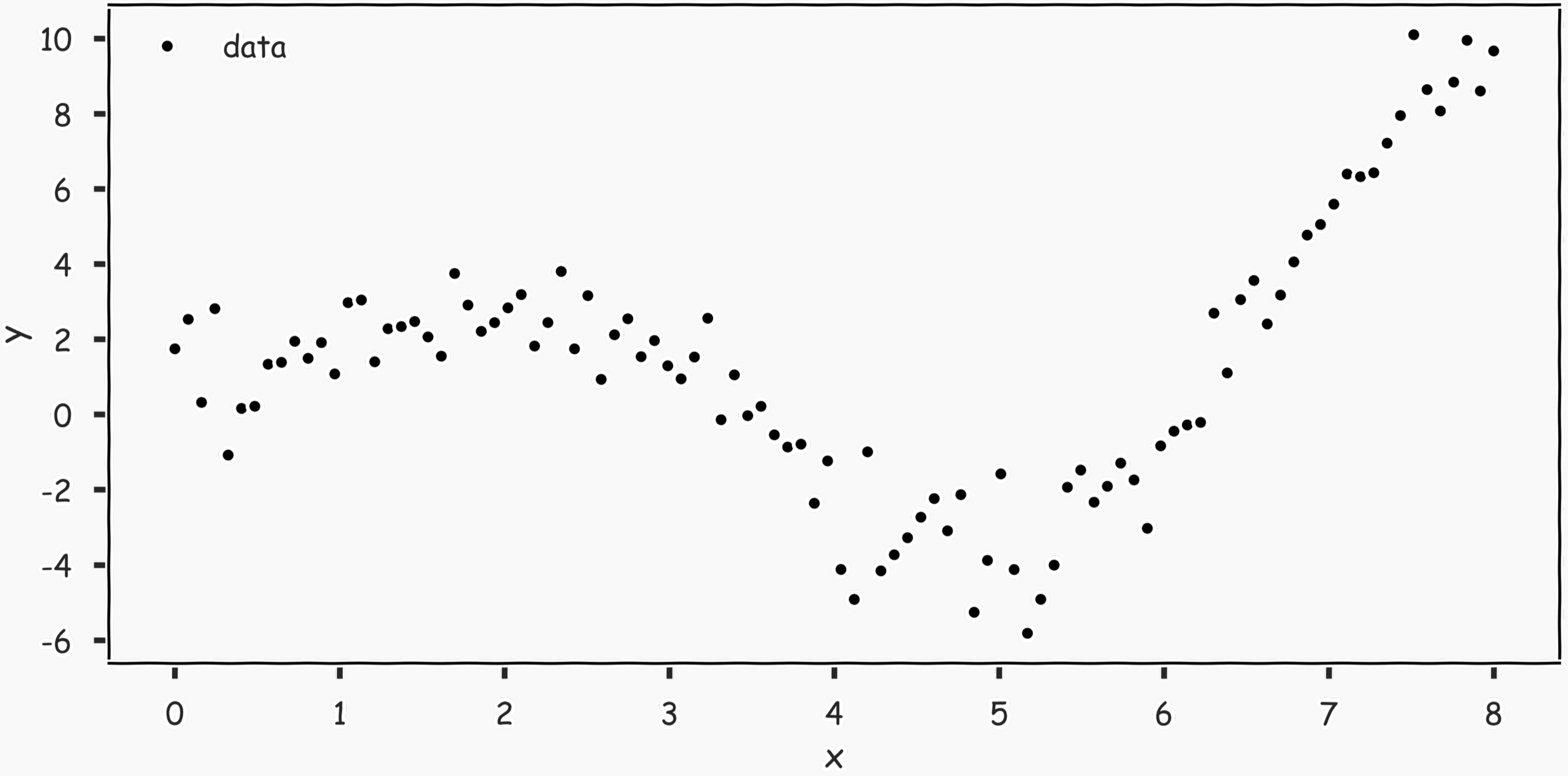
We have $\{T_1, T_2, T_3, \ldots, T_N\}$

$$T = \sum_h \lambda_h T_H$$

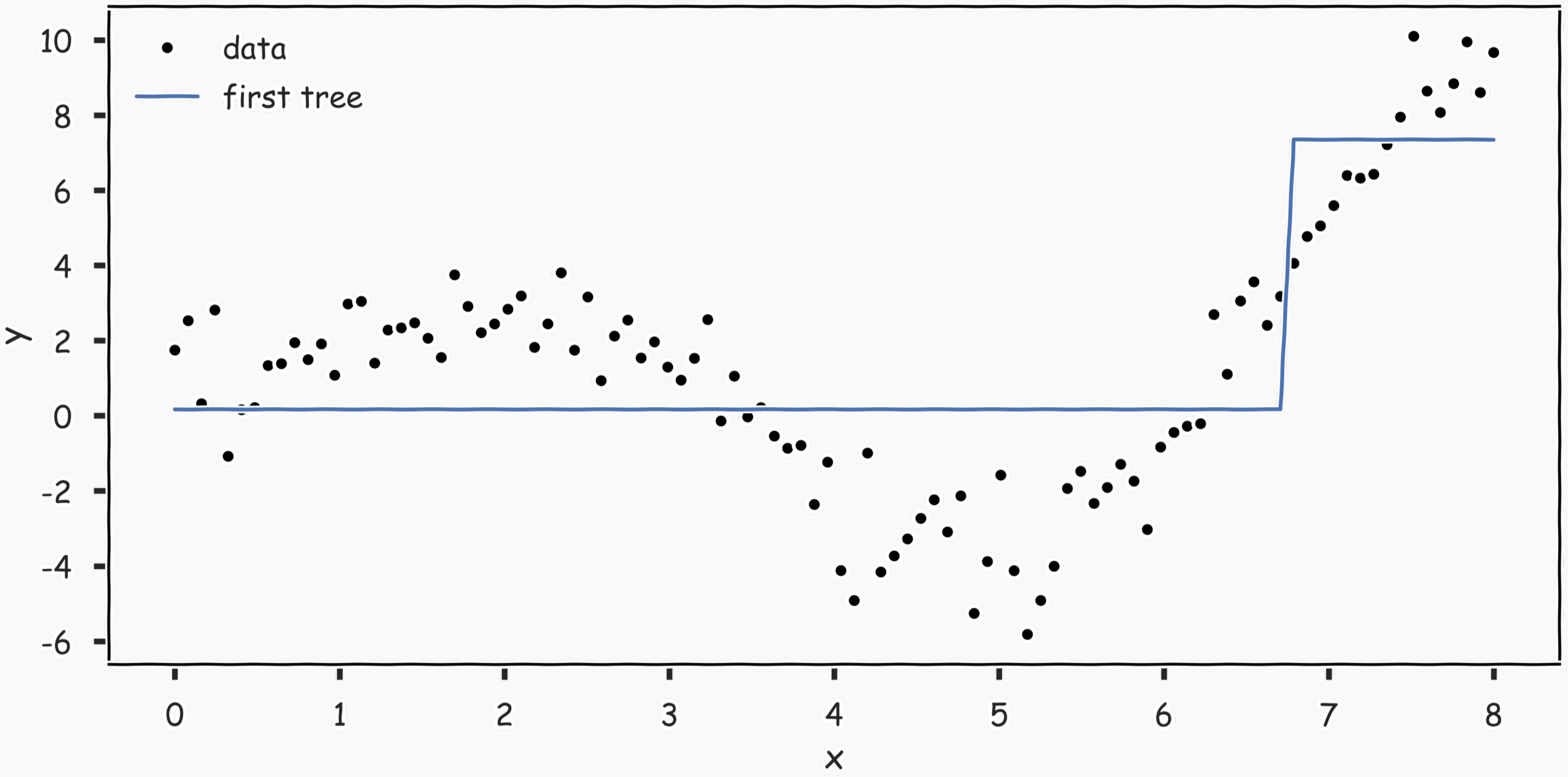
Each T_h is:

- a "weak"/simple decision tree
- built after the previous tree
- tries to learn the shortcomings (the errors/residuals) from the previous tree's predictions

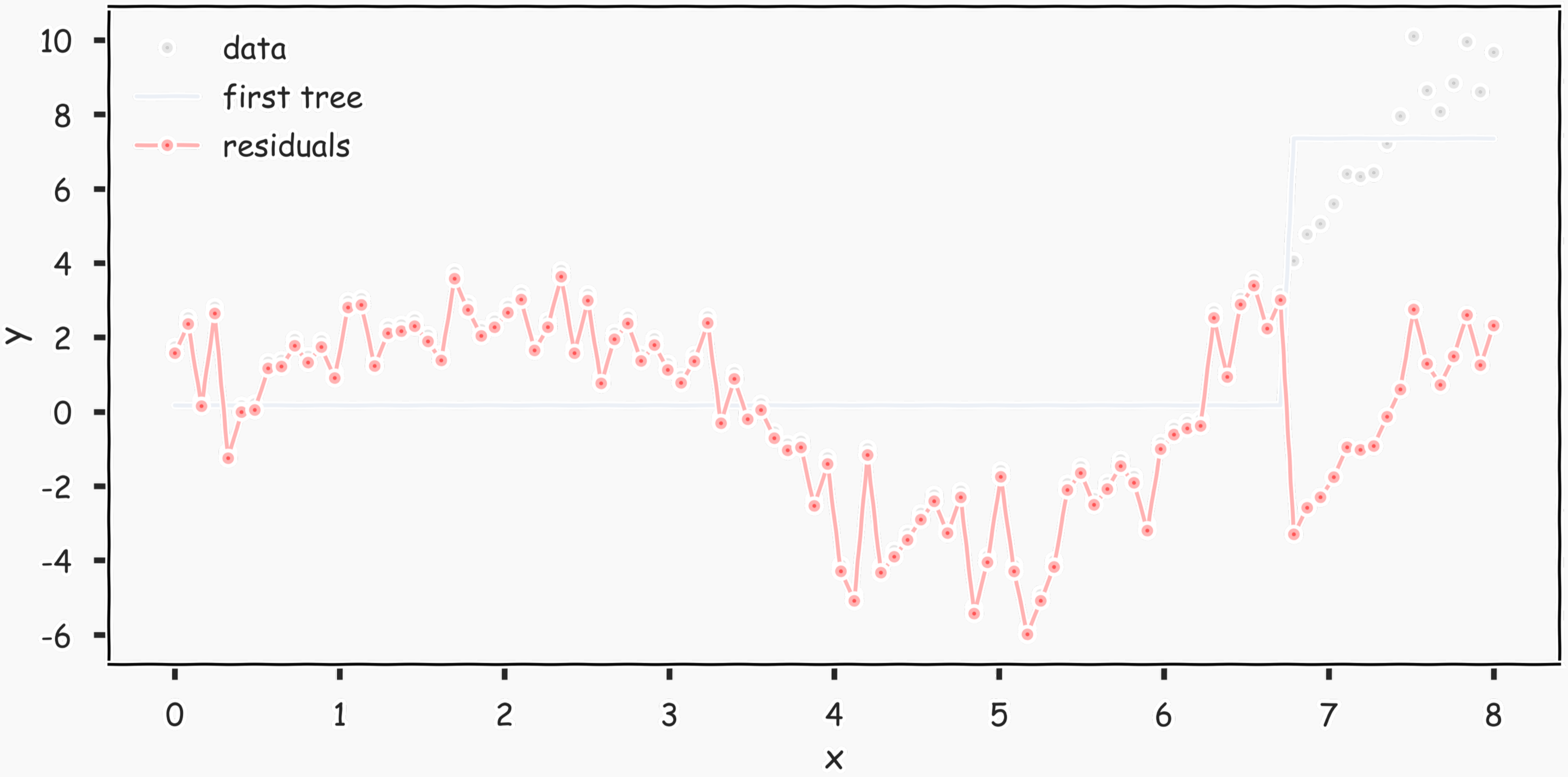
Gradient Boosting: illustration



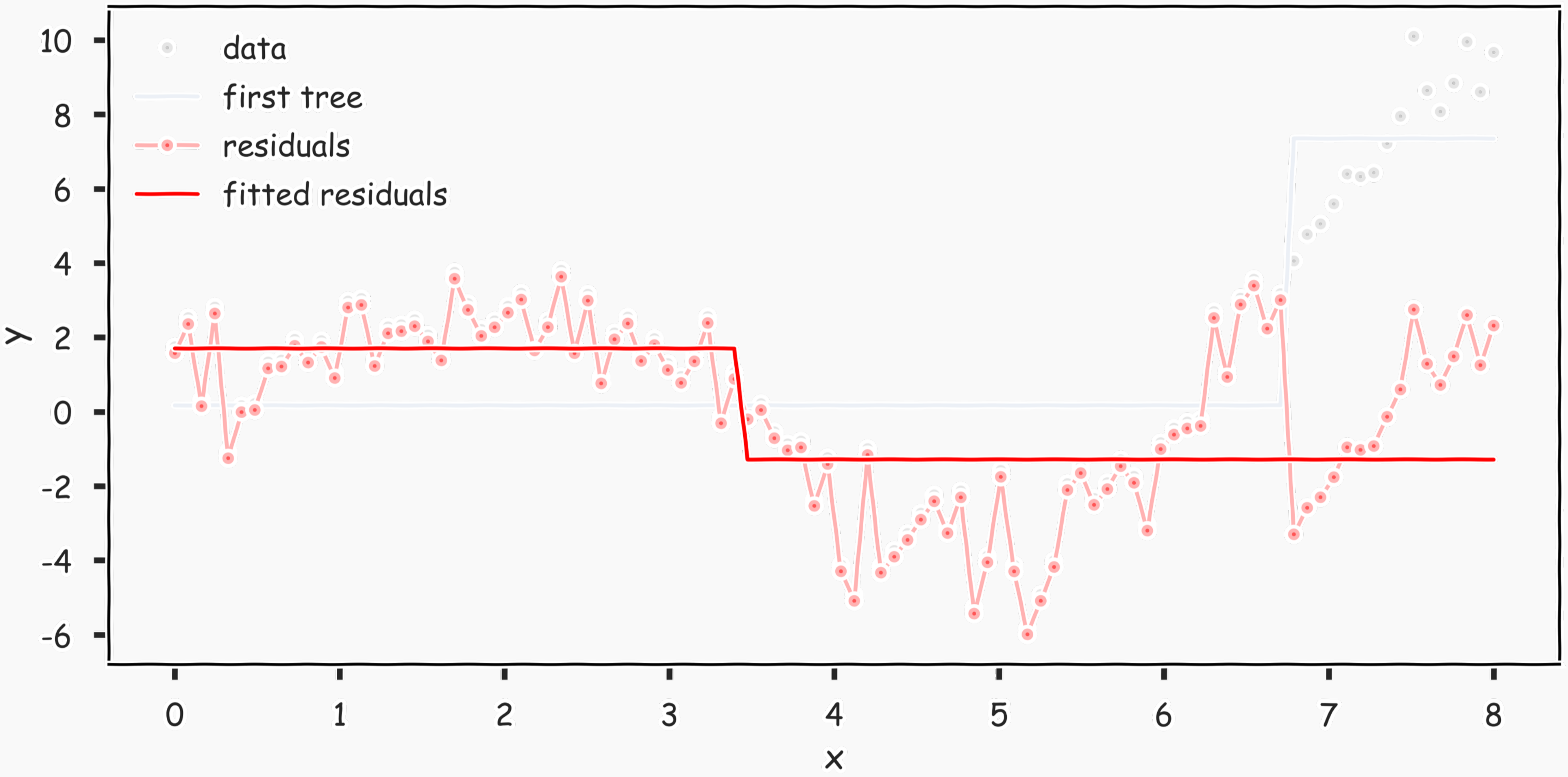
Gradient Boosting: illustration



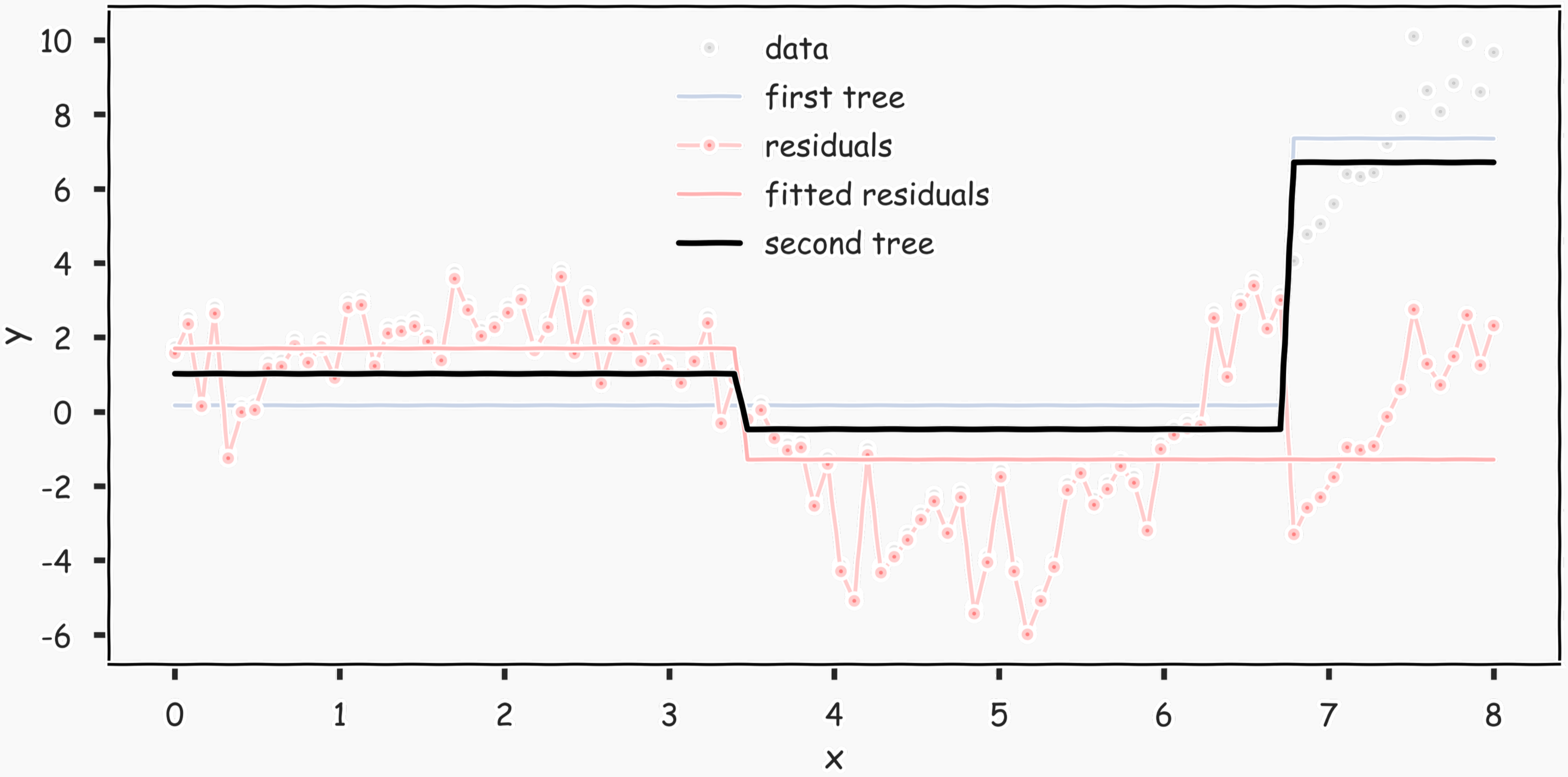
Gradient Boosting: illustration



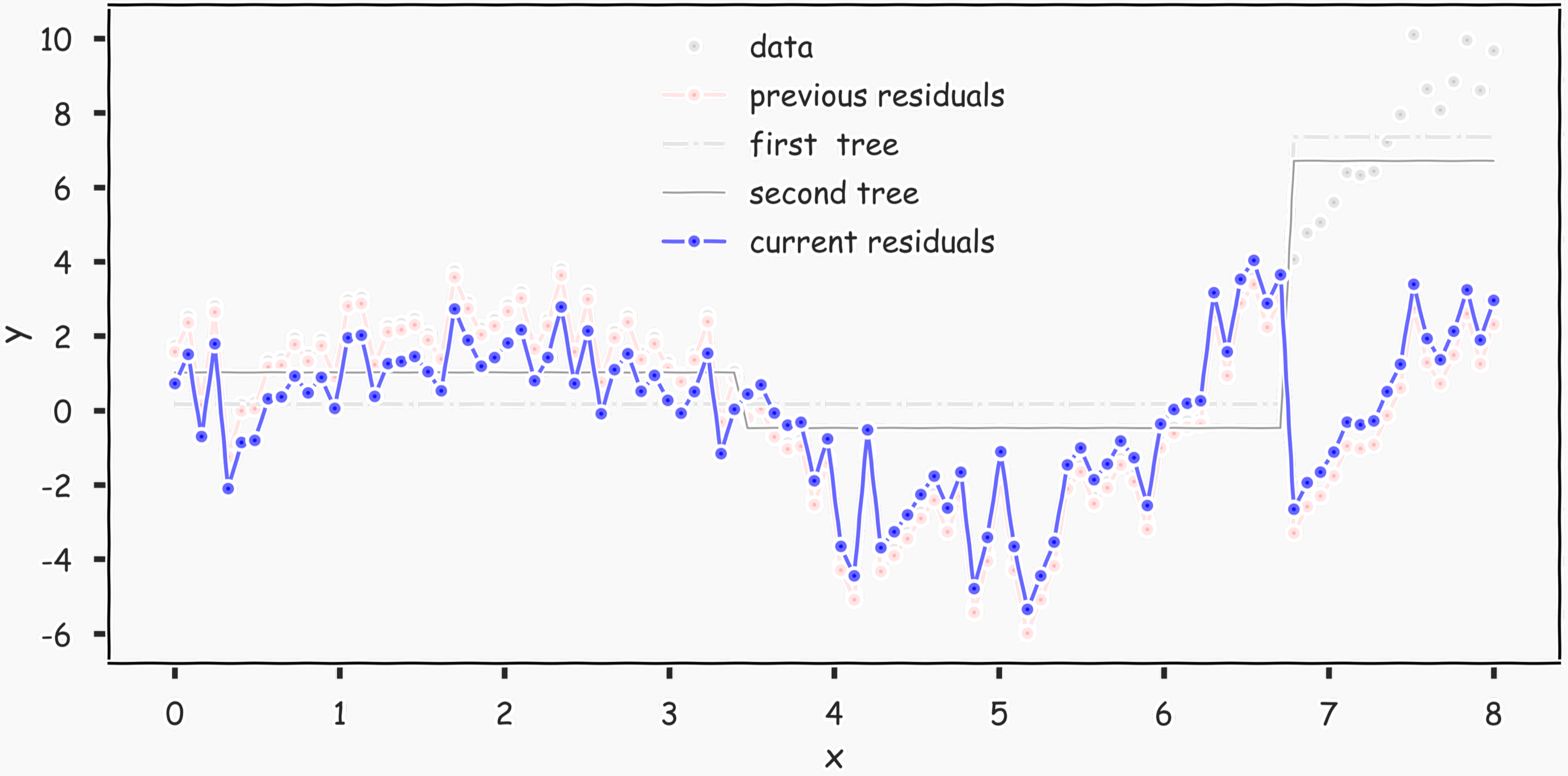
Gradient Boosting: illustration



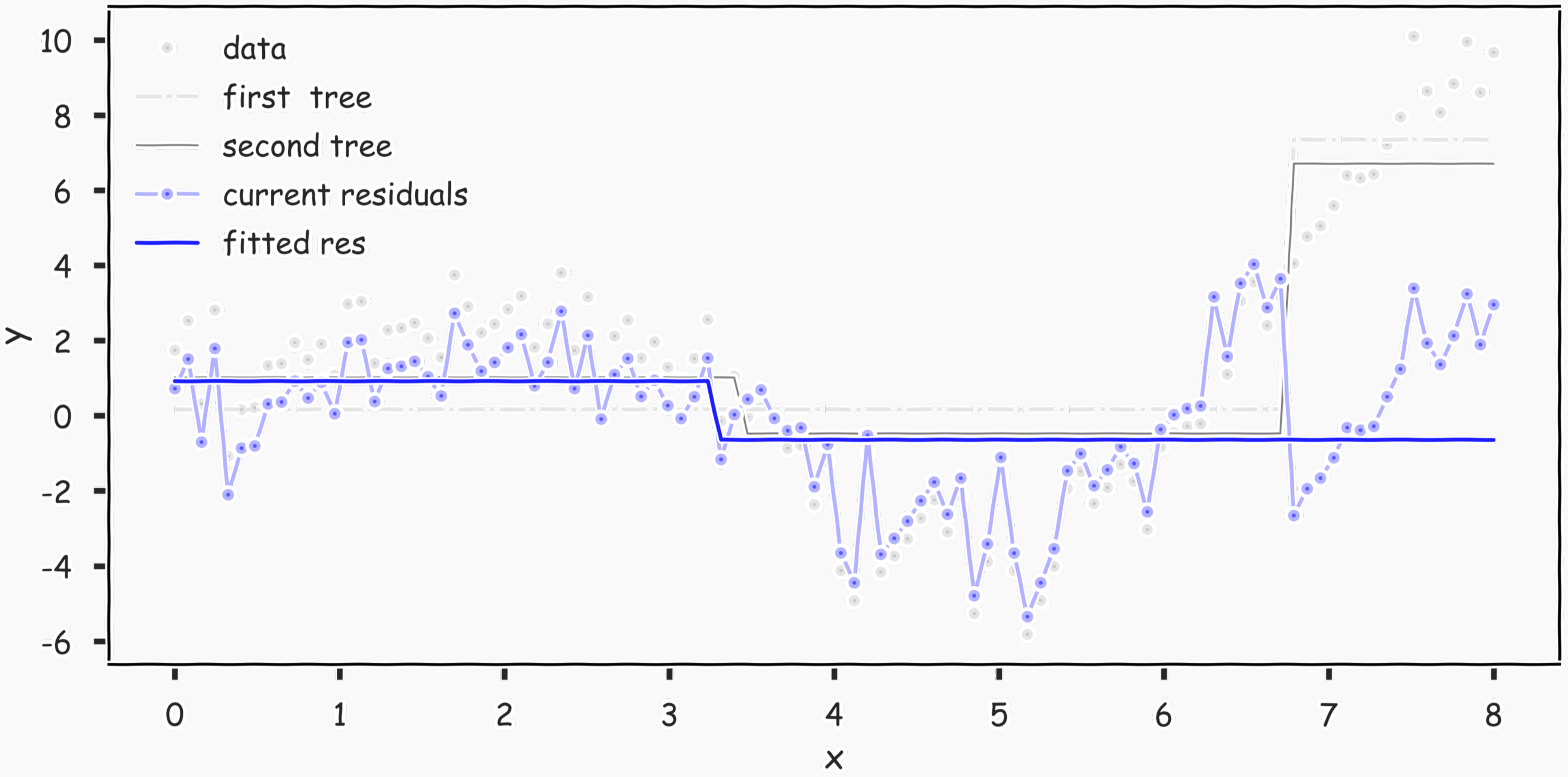
Gradient Boosting: illustration



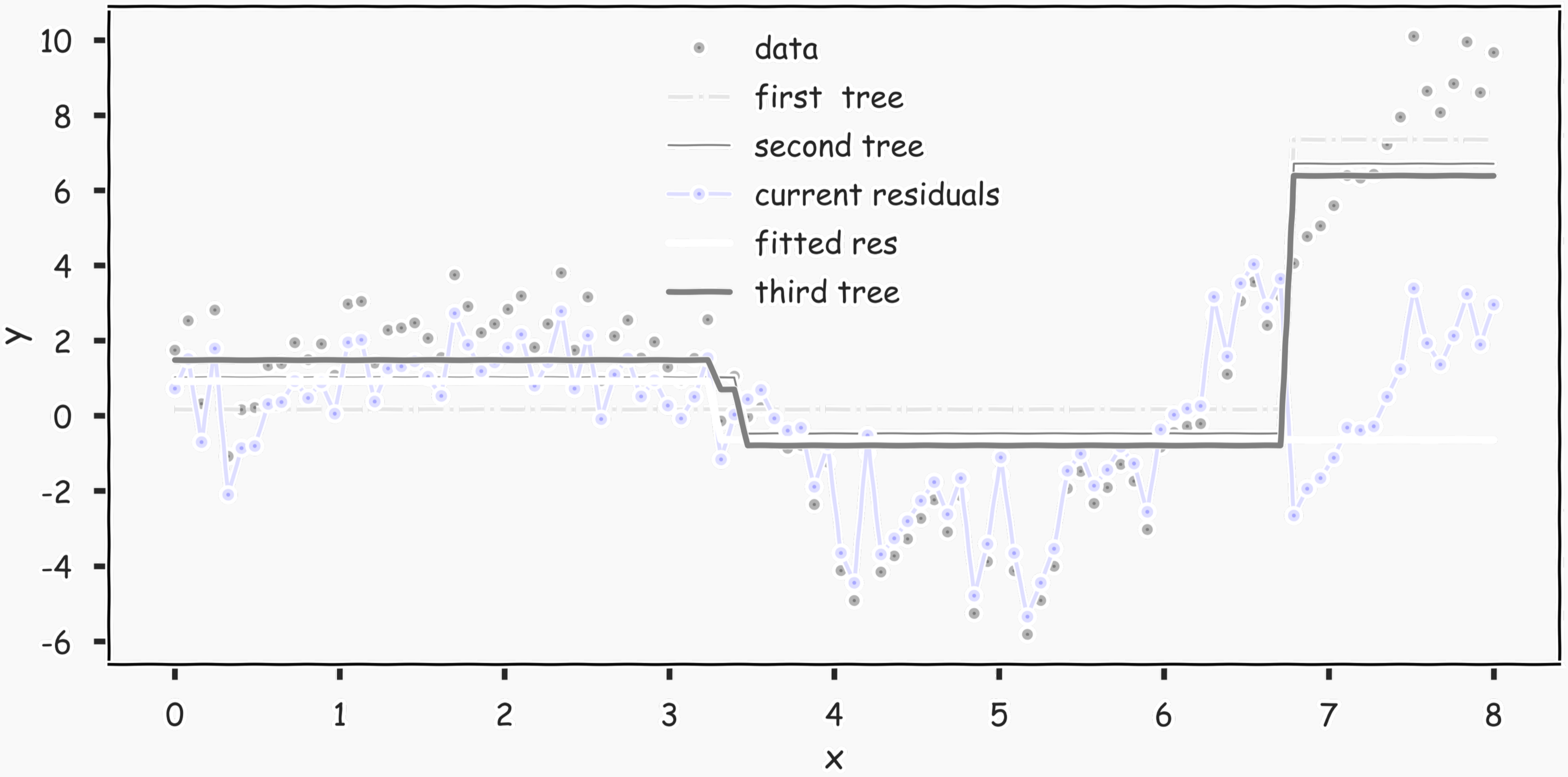
Gradient Boosting: illustration



Gradient Boosting: illustration



Gradient Boosting: illustration



Gradient Boosting

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

We have $\{T_1, T_2, T_3, \ldots, T_N\}$

$$T = \sum_h \lambda_h T_H$$

We can determine each λ_h

by using gradient descent.

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda r_n, \quad n = 1, \ldots, N$$

Idea

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

If we have **categorical** data (not a regression task), we can use AdaBoost

Idea

Training Data

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

If we have **categorical** data (not a regression task), we can use AdaBoost

1. Train a single weak (stump) Decision Tree T_i
2. Calculate the total error of your predictions
3. Use this error (λ_i) to determine how much stock to place in that Tree
4. Update the weights of each observation
5. Update our running model T

AdaBoost

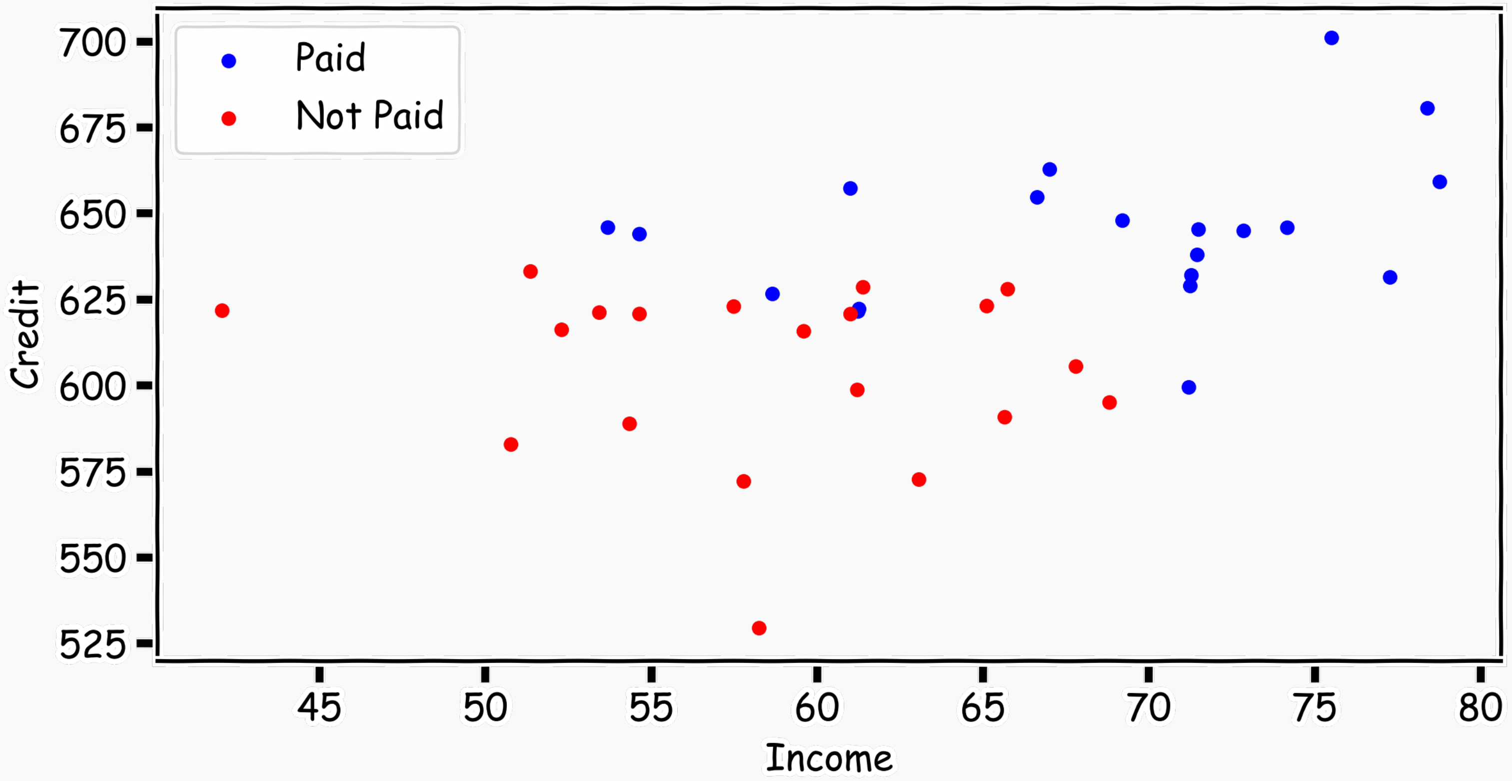
With a minor adjustment to the exponential loss function, we have the algorithm for gradient descent:

1. Choose an initial distribution over the training data, $w_n = 1/N$.
2. At the i^{th} step, fit a simple classifier $T^{(i)}$ on weighted training data
$$\{(x_1, w_1 y_1), \dots, (x_N, w_N y_N)\}$$
3. Update the weights:
$$w_n \leftarrow \frac{w_n \exp(-\lambda^{(i)} y_n T^{(i)}(x_n))}{Z}$$
where Z is the normalizing constant for the collection of updated weights

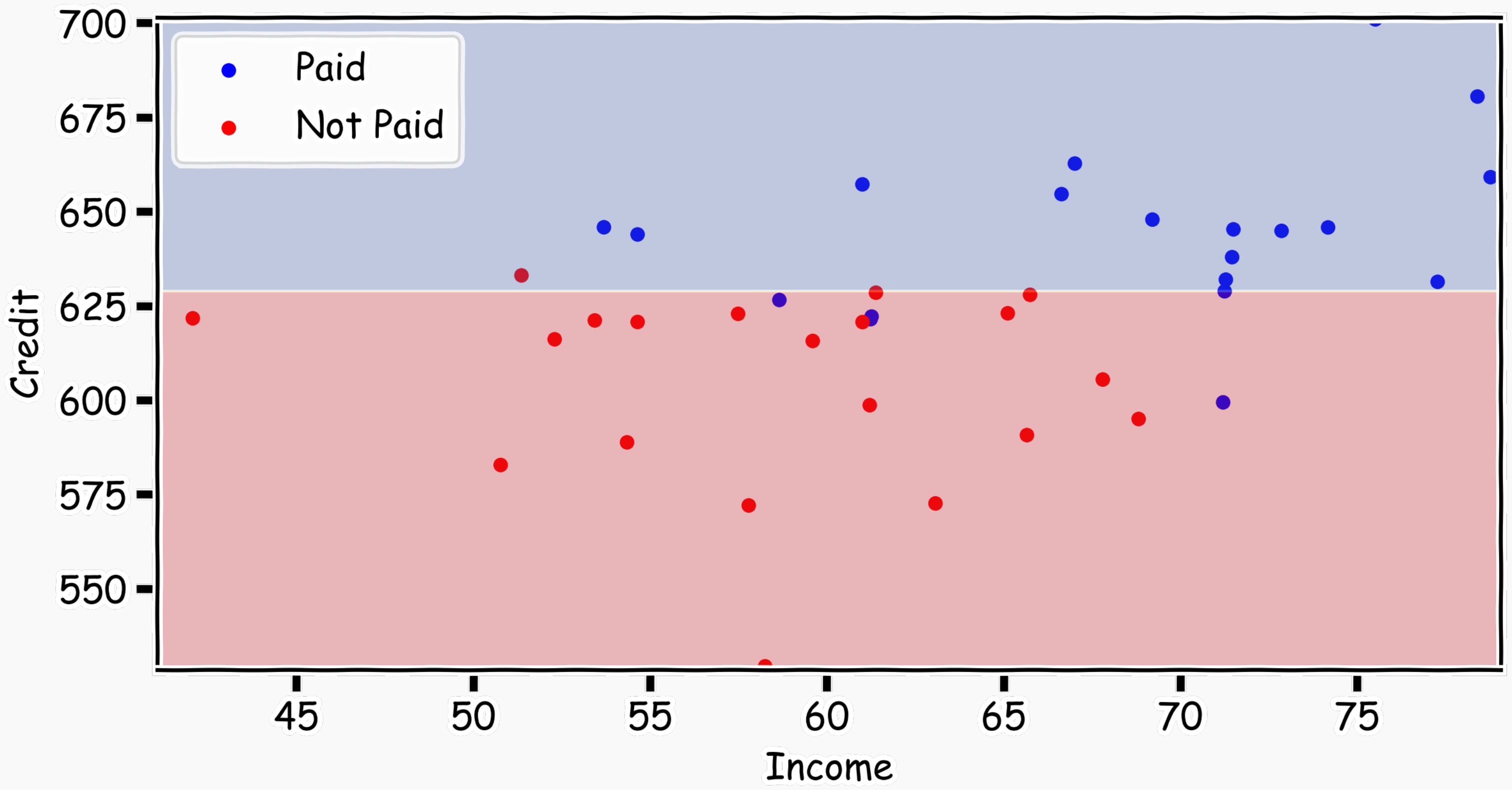
4. Update T: $T \leftarrow T + \lambda^{(i)} T^{(i)}$

where λ is the learning rate.

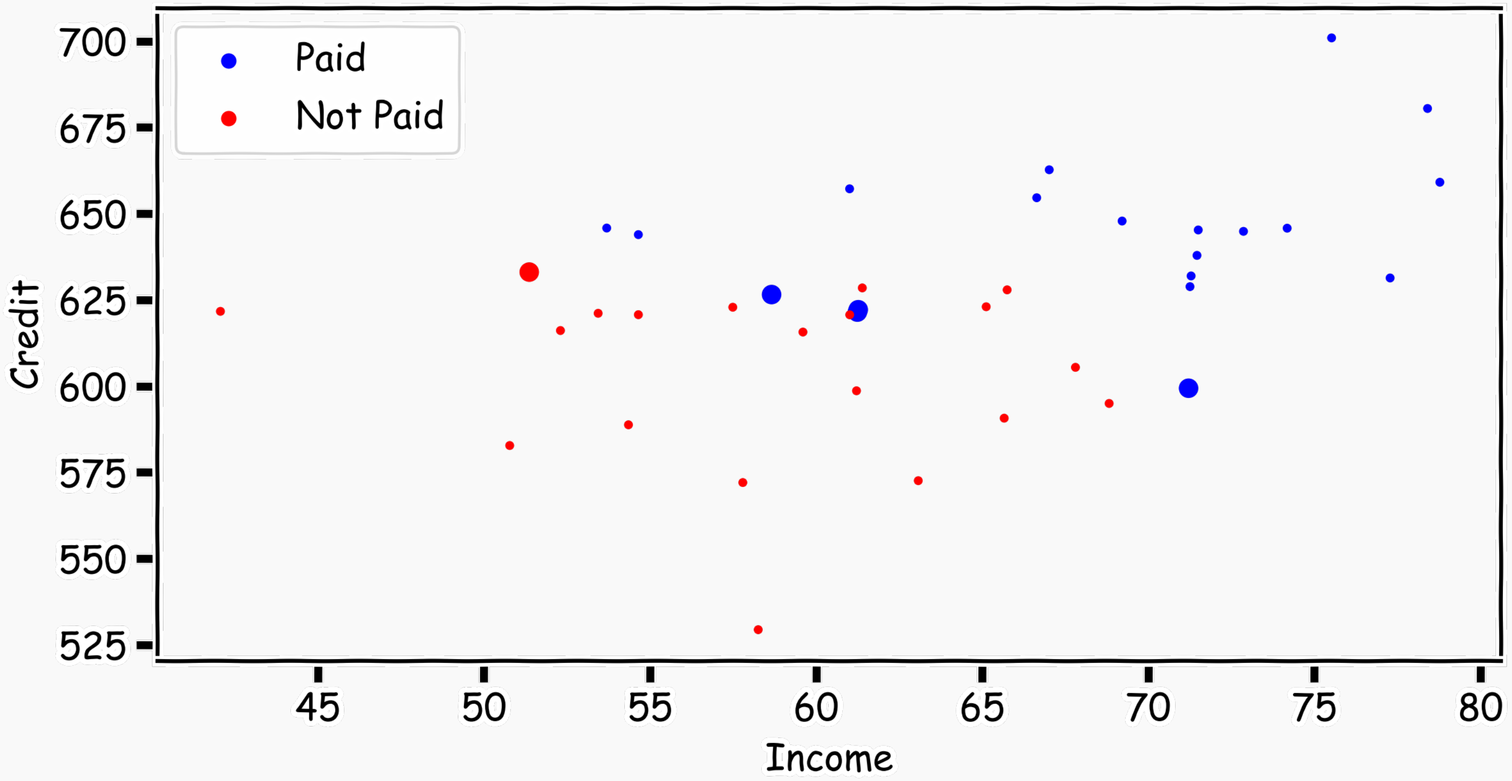
AdaBoost: start with equal weights



AdaBoost: fit a simple decision tree

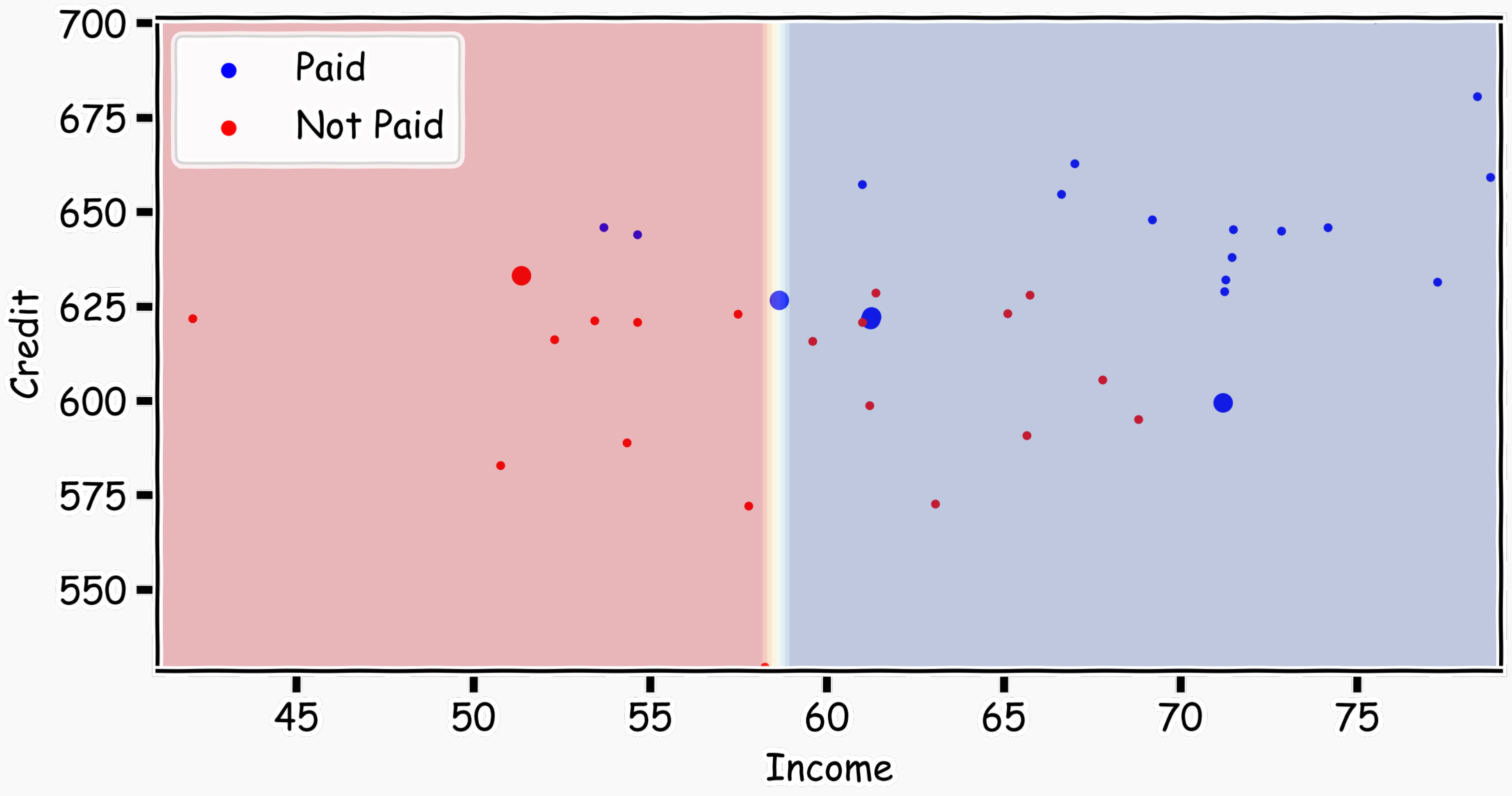


AdaBoost: update the weights

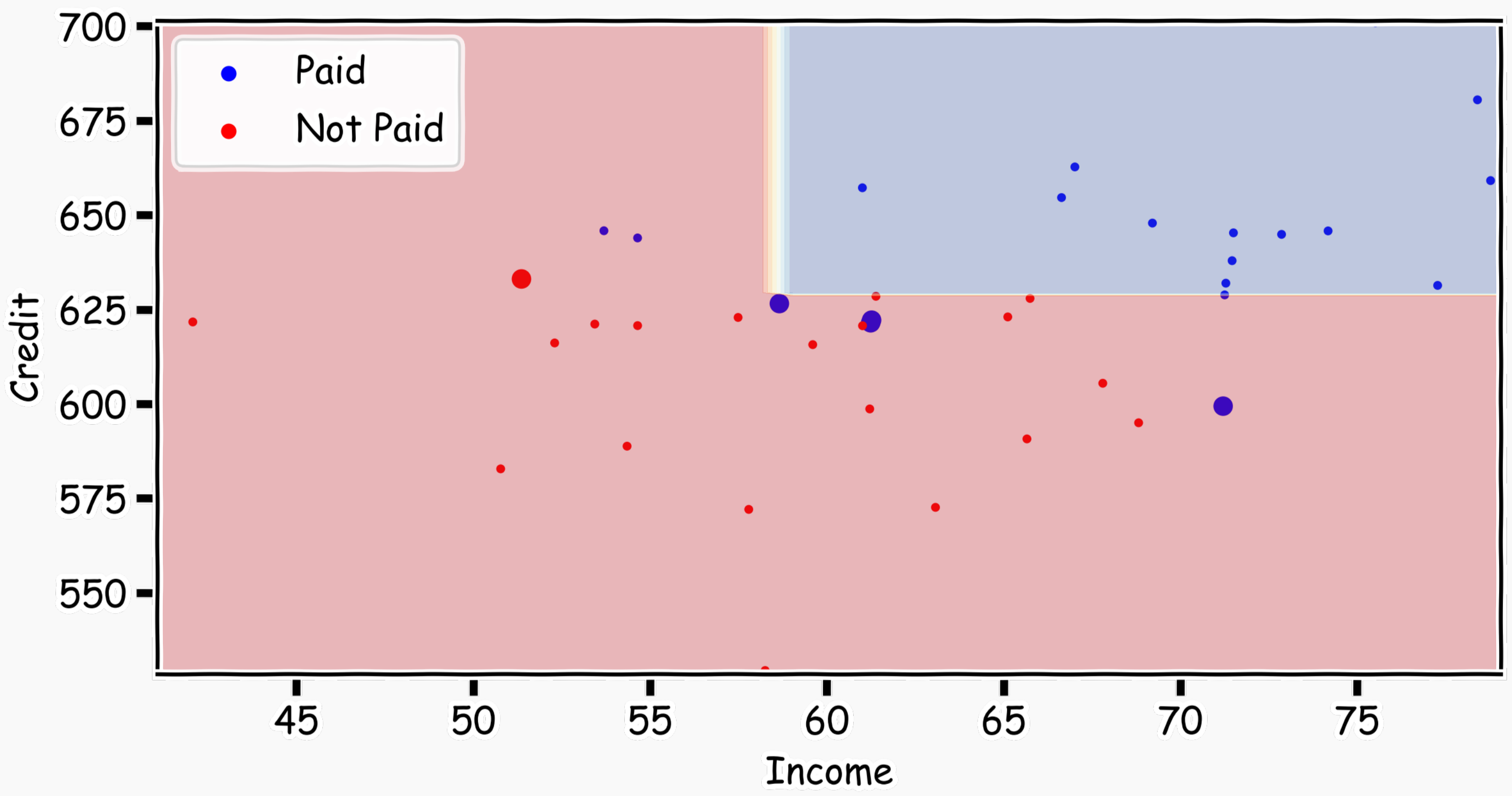


$$w_n \leftarrow \frac{w_n \exp(-\lambda^{(i)} y_n T^{(i)}(x_n))}{Z}$$

AdaBoost: fit another simple decision tree on re-weighted data

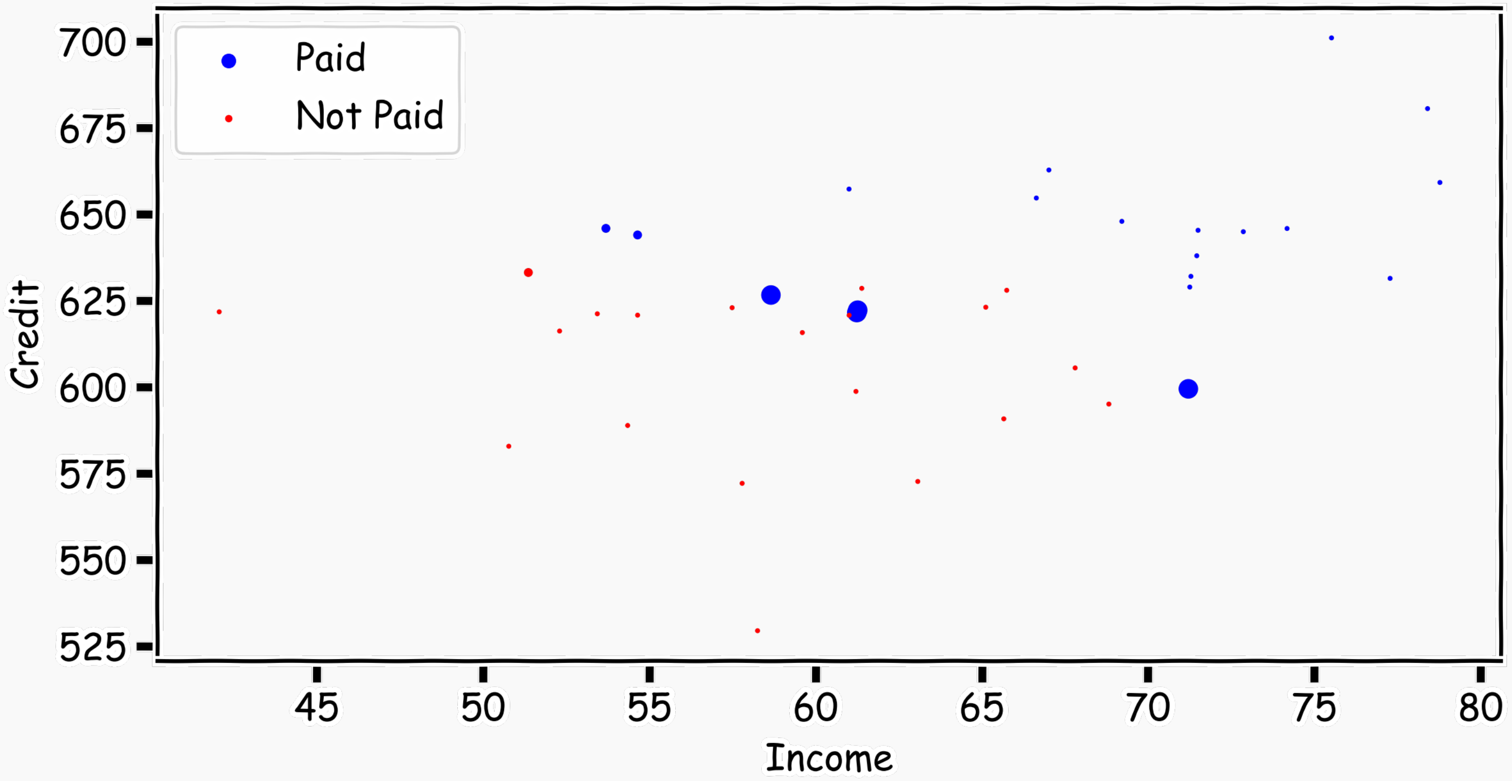


AdaBoost: add the new model to the ensemble



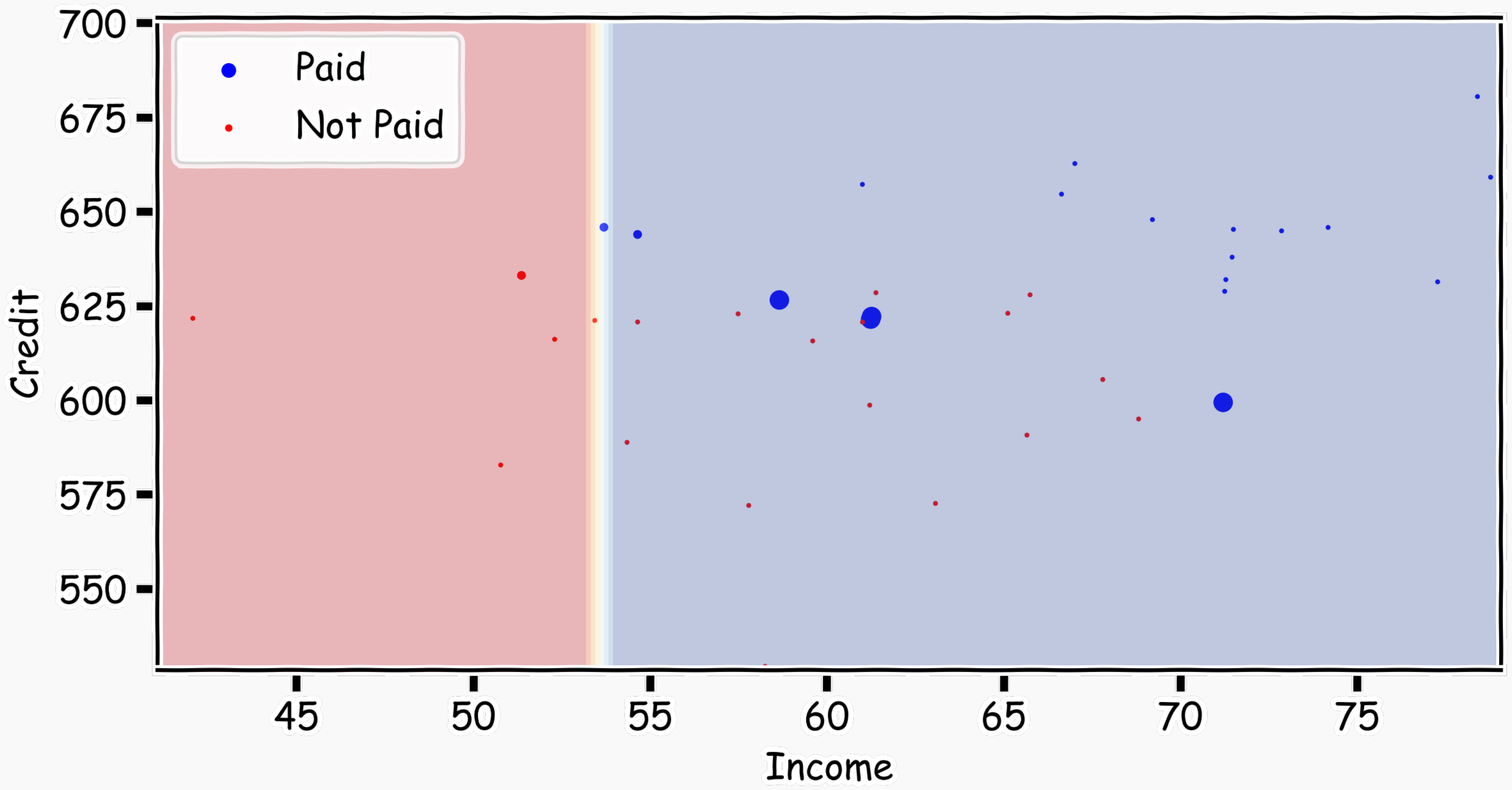
$$T \leftarrow T + \lambda^{(i)} T^{(i)}$$

AdaBoost: update the weights

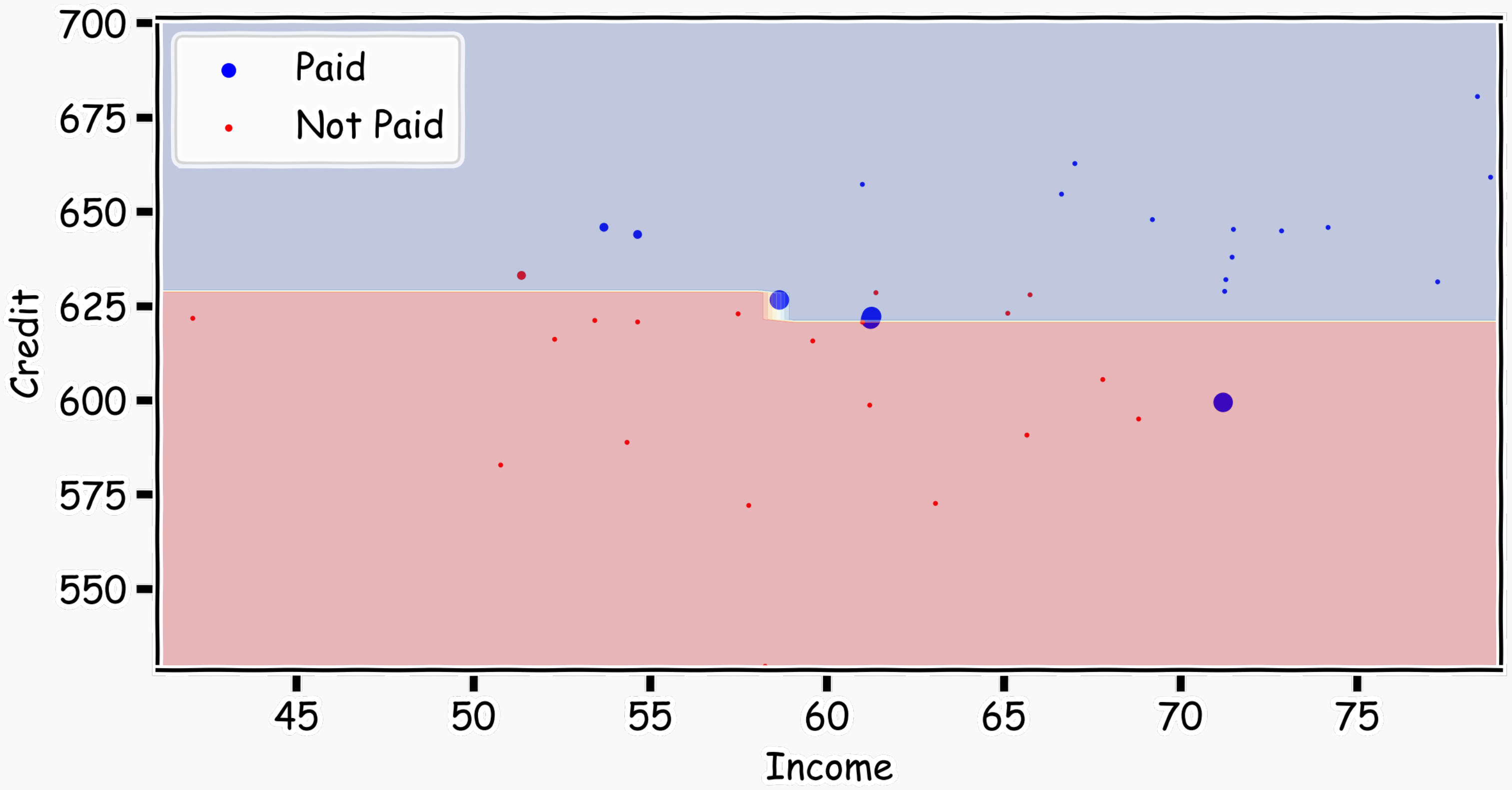


$$w_n \leftarrow \frac{w_n \exp(-\lambda^{(i)} y_n T^{(i)}(x_n))}{Z}$$

AdaBoost: fit a third, simple decision tree on re-weighted data



AdaBoost: add the new model to the ensemble, repeat...



$$T \leftarrow T + \lambda^{(i)} T^{(i)}$$

Choosing the Learning Rate

Unlike in the case of gradient boosting for regression, we can analytically solve for the optimal learning rate for AdaBoost, by optimizing:

$$\underset{\lambda}{\text{argmin}} \frac{1}{N} \sum_{n=1}^{N} \exp\left[-y_n(T + \lambda^{(i)} T^{(i)}(x_n))\right]$$

Doing so, we get that

$$\lambda^{(i)} = \frac{1}{2} \ln \frac{1-\epsilon}{\epsilon}, \quad \epsilon = \sum_{n=1}^{N} w_n \mathbb{1}(y_n \neq T^{(i)}(x_n))$$