Gradient Boosting

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner

Comparison of Models:

Choosing the right model isn't just about minimizing the test errors.
We want extra insights from our models:

	Has a fixed form f(x) parametric	easy to interpret	computational complexity
Linear Regression	YES	YES	LOW
Polynomial Regression	YES	NO	LOW
Regression Trees	NO	YES	LOW
Bagging and RF	NO	YES/NO	MEDIUM
K-nearest Neighbors	NO	YES	HIGH

"Can a set of weak learners create a single strong learner?"

Leslie Gabriel Valiant

How many jelly beans do you see?

I VOTED.

Motivation for Boosting

Question: Could we address the shortcomings of single decision trees models in some other way?

For example, rather than performing variance reduction on complex trees, can we decrease the bias of simple trees - make them more expressive?

Can we learn from our mistakes?

A solution to this problem, making an expressive model from simple trees, is another class of ensemble methods called ***boosting***.

Gradient Boosting

The key intuition behind boosting is that one can take an ensemble of simple models $\{T_h\}_{h \in H}$ and additively combine them into a single, more complex model.

Each model T_h might be a poor fit for the data, but a linear combination of the ensemble:

$$\boxed{T = \sum_h \lambda_h T_H}$$

can be expressive/flexible.

Question: But which models should we include in our ensemble? What should the coefficients or weights in the linear combination be?

Gradient Boosting: the algorithm

Gradient boosting is a method for iteratively building a complex regression model T by adding simple models.

Each new simple model added to the ensemble compensates for the weaknesses of the current ensemble.

Gradient Boosting: the algorithm

1. Fit a simple model $T^{(0)}$ on the training data

$$\{(x_1, y_1), \dots, (x_N, y_N)\}$$

   Set $T \leftarrow T^{(0)}$.

   Compute the residuals $\{r_1, \dots, r_N\}$ for T.

2. Fit a simple model, $T^{(1)}$, to the current ***residuals***, i.e. train using

$$\{(x_1, r_1), \dots, (x_N, r_N)\}$$

3. Set $T \leftarrow T + \lambda T^{(1)}$

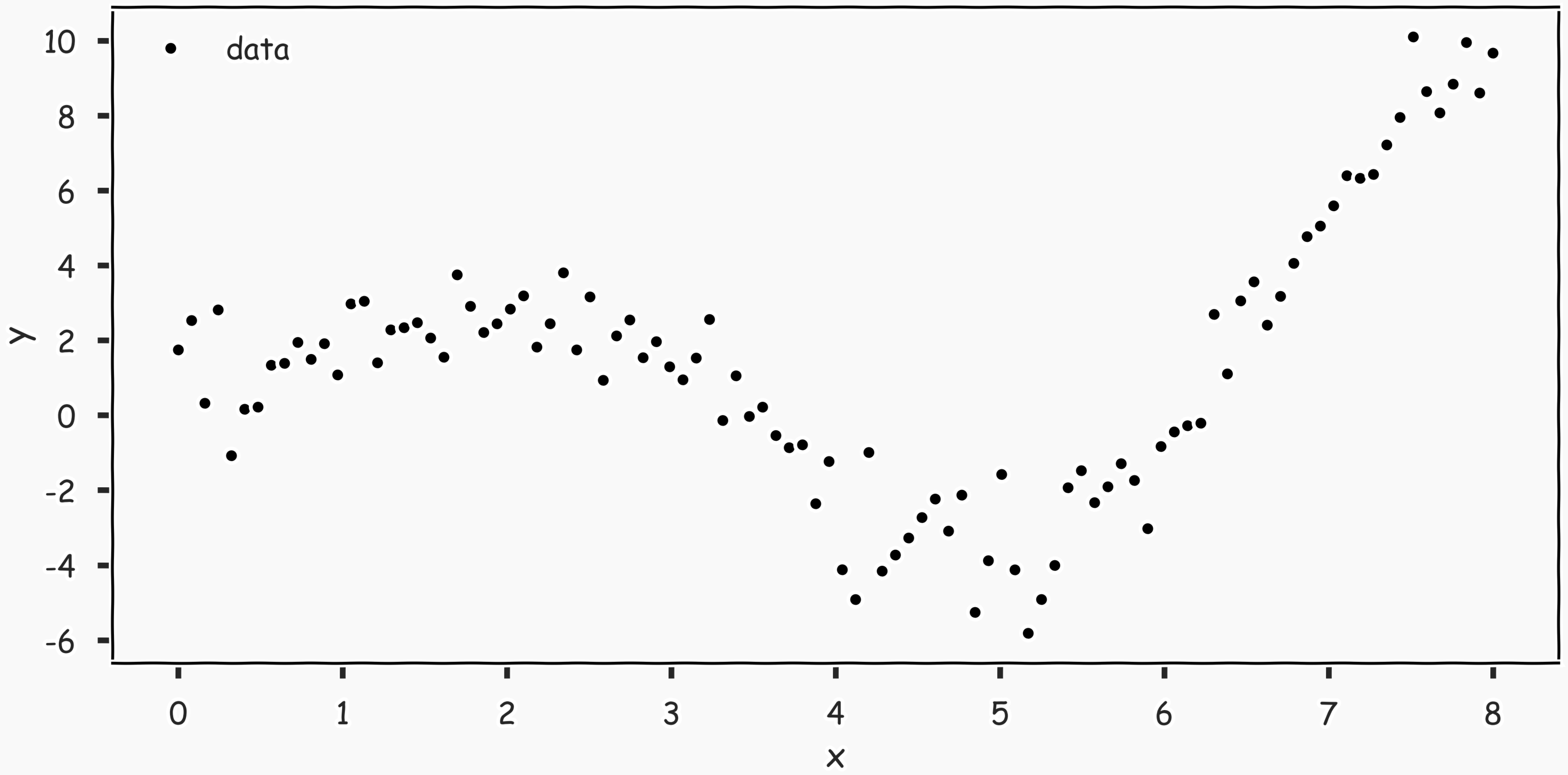
4. Compute residuals, set $r_n \leftarrow r_n - \lambda T^i(x_n),\ n = 1, \dots, N$

5. Repeat steps 2-4 until ***stopping*** condition met.

   where λ is a constant called the ***learning rate***.

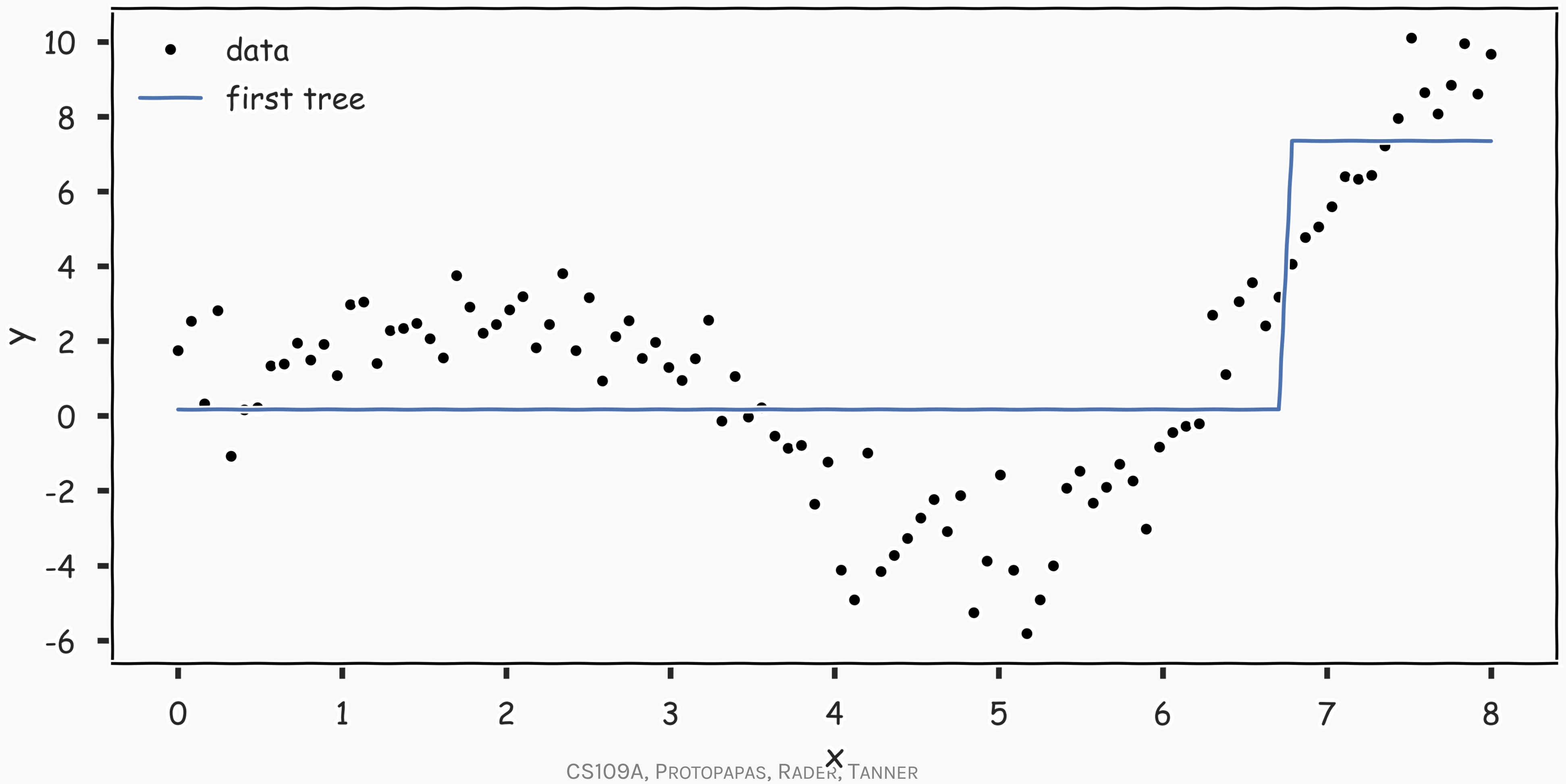
Gradient Boosting: illustration

training data: $\{(x_1, y_1), \ldots, (x_N, y_N)\}$



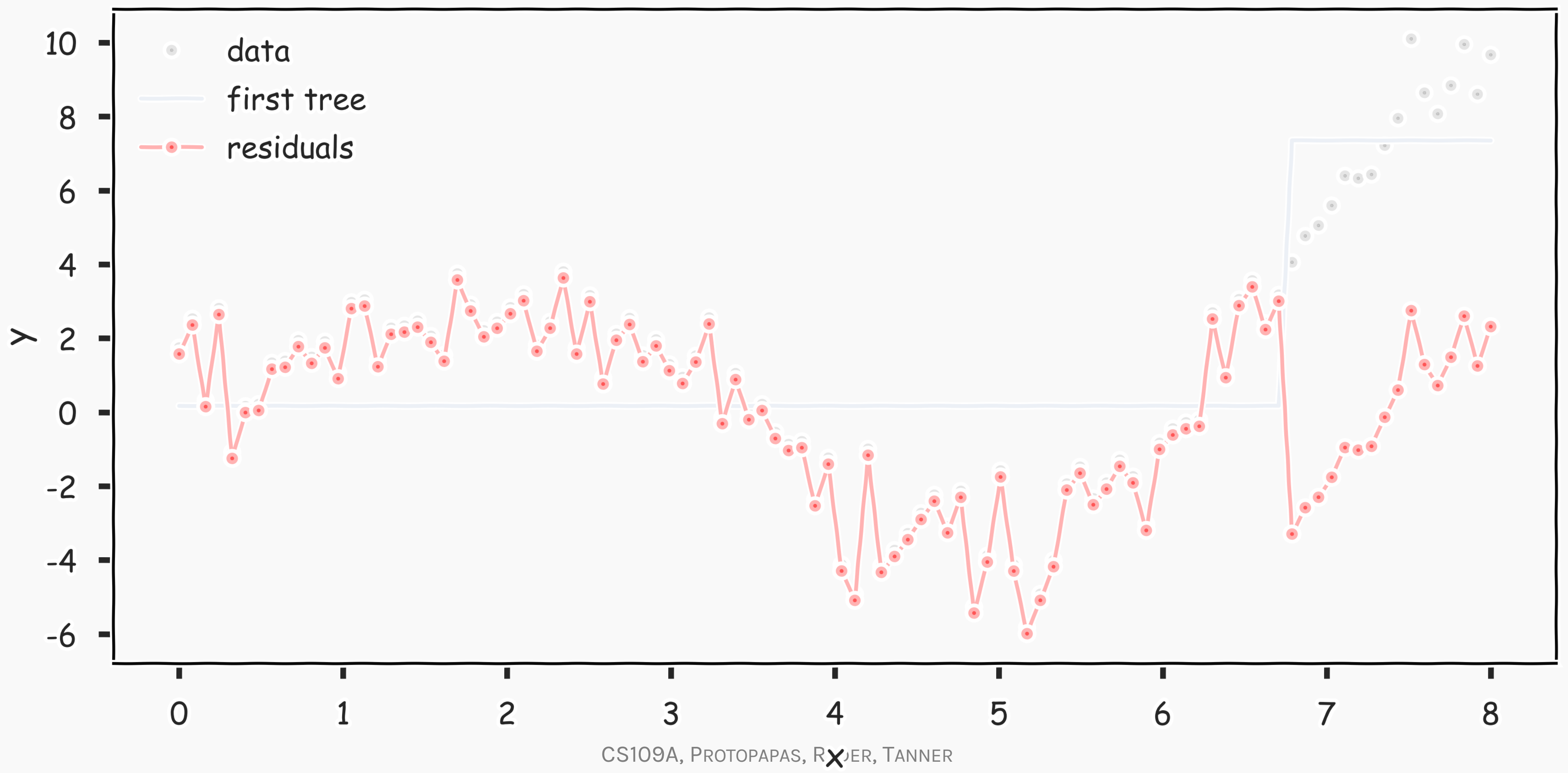
Gradient Boosting: illustration

Fit a simple model $T^{(0)}$



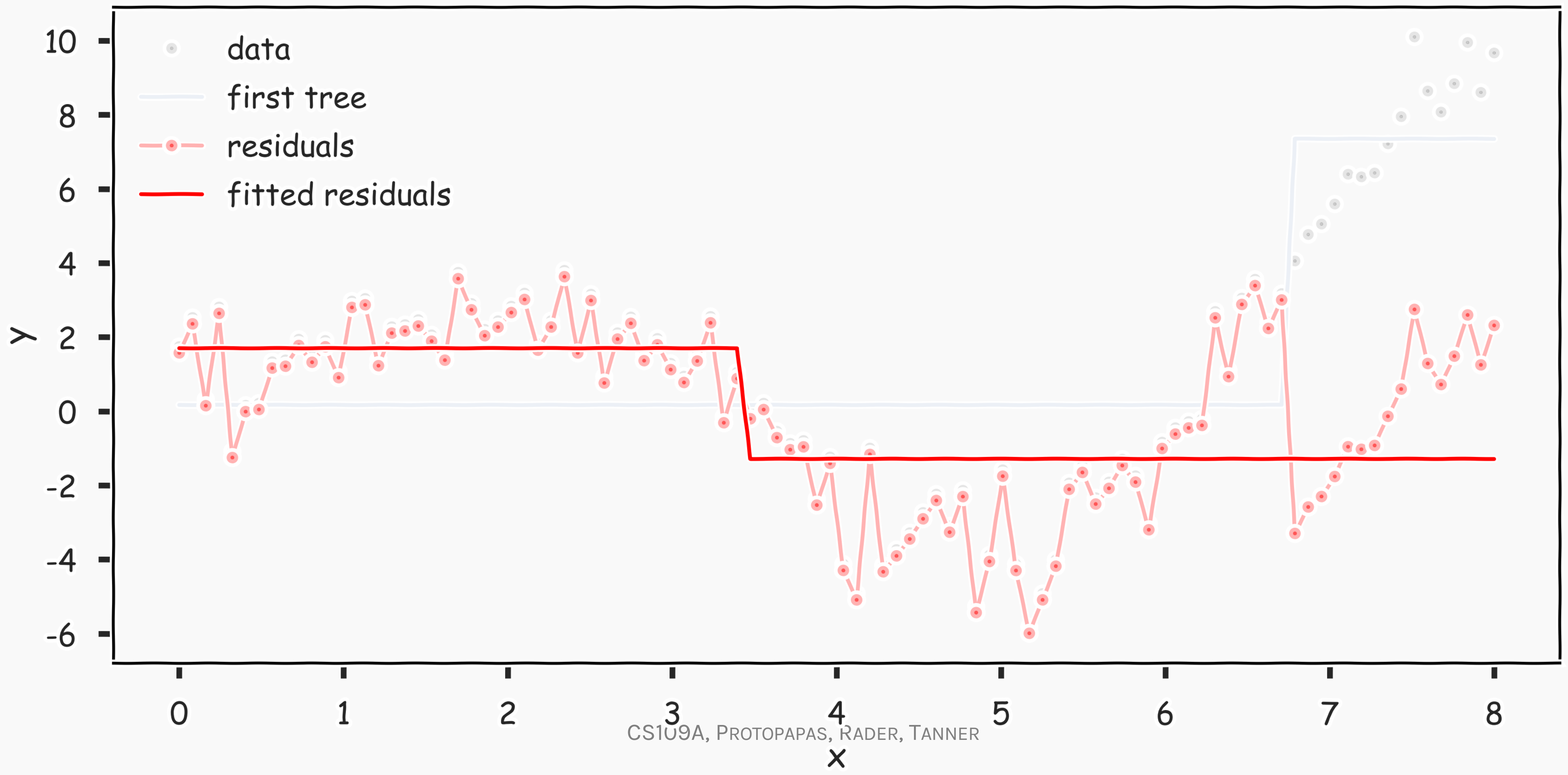
Gradient Boosting: illustration

Compute the residuals $\{r_1, \ldots, r_N\}$ for T.



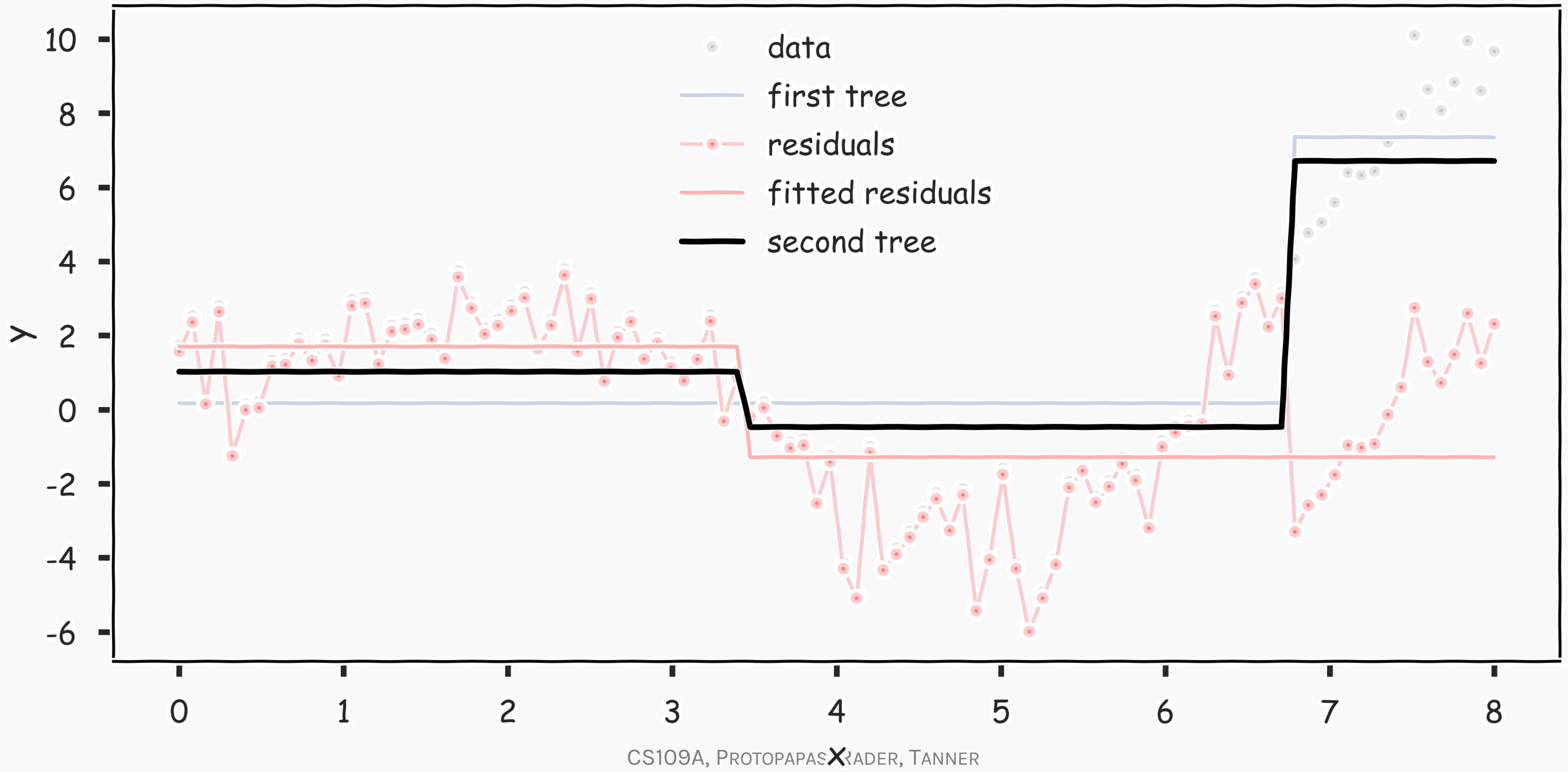
Gradient Boosting: illustration

train usin: $\{(x_1, r_1), \ldots, (x_N, r_N)\}$



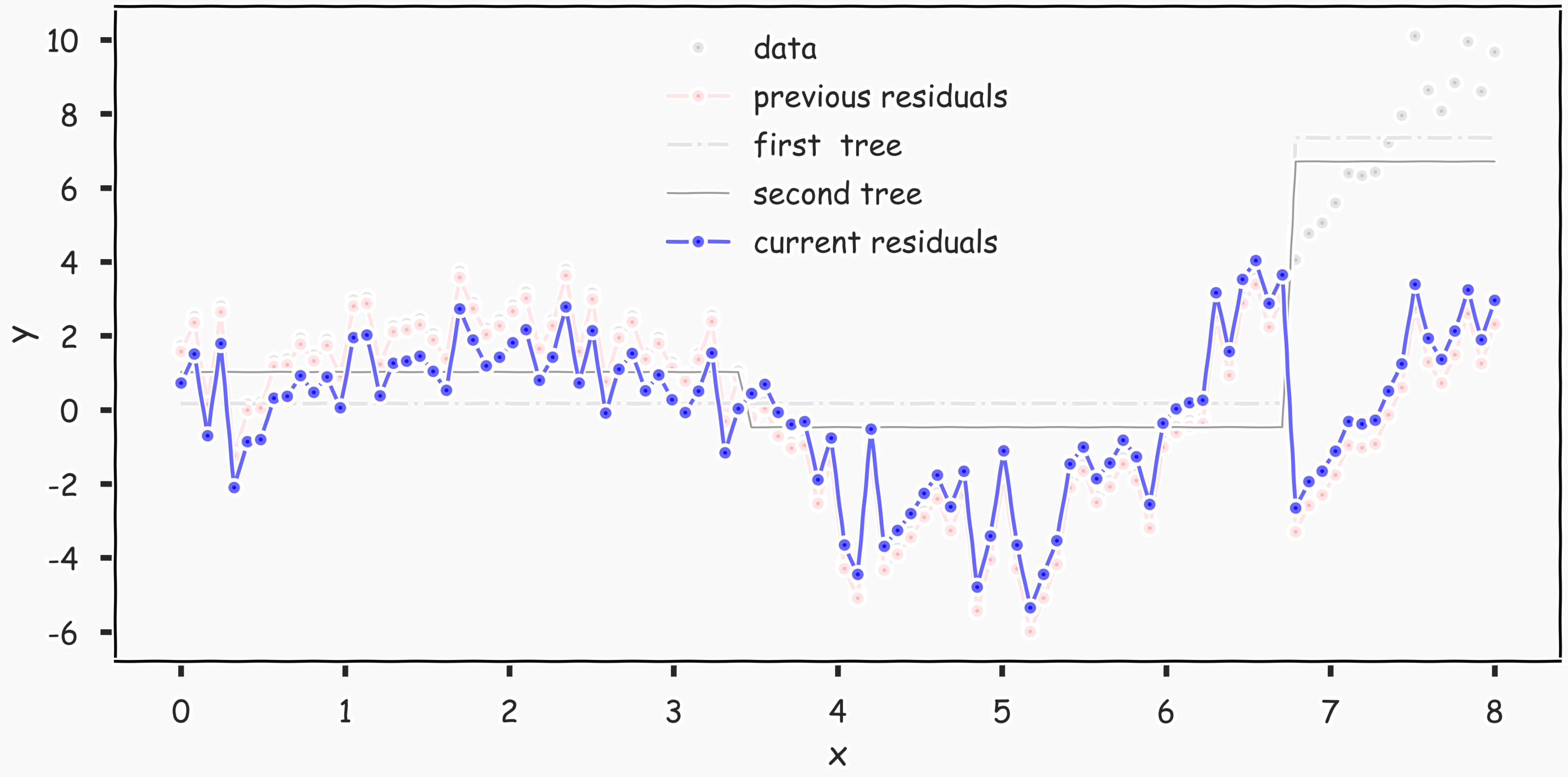
Gradient Boosting: illustration

$$\boxed{\text{Set } T \leftarrow T + \lambda T^{(1)}}$$

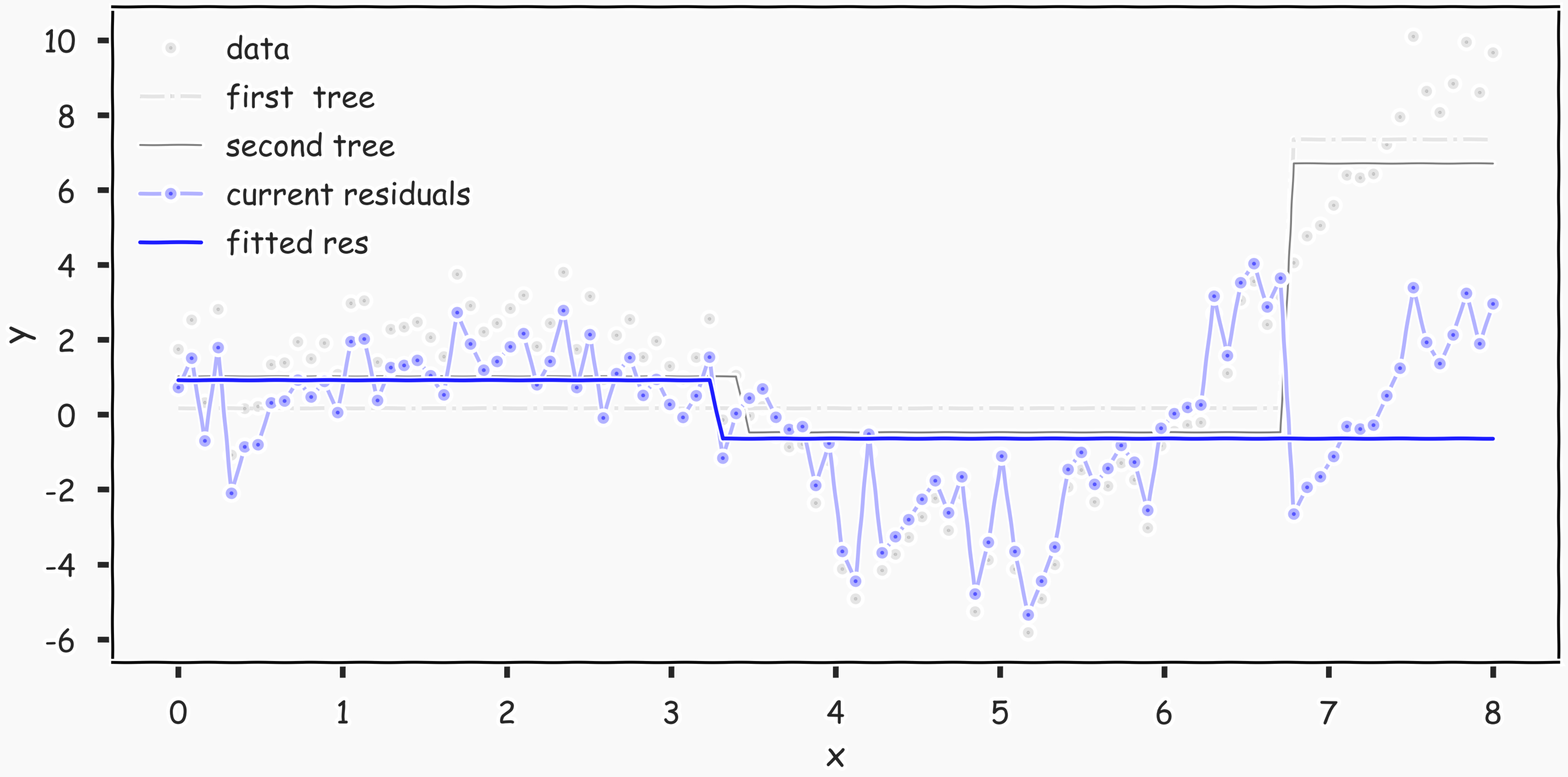


Gradient Boosting: illustration

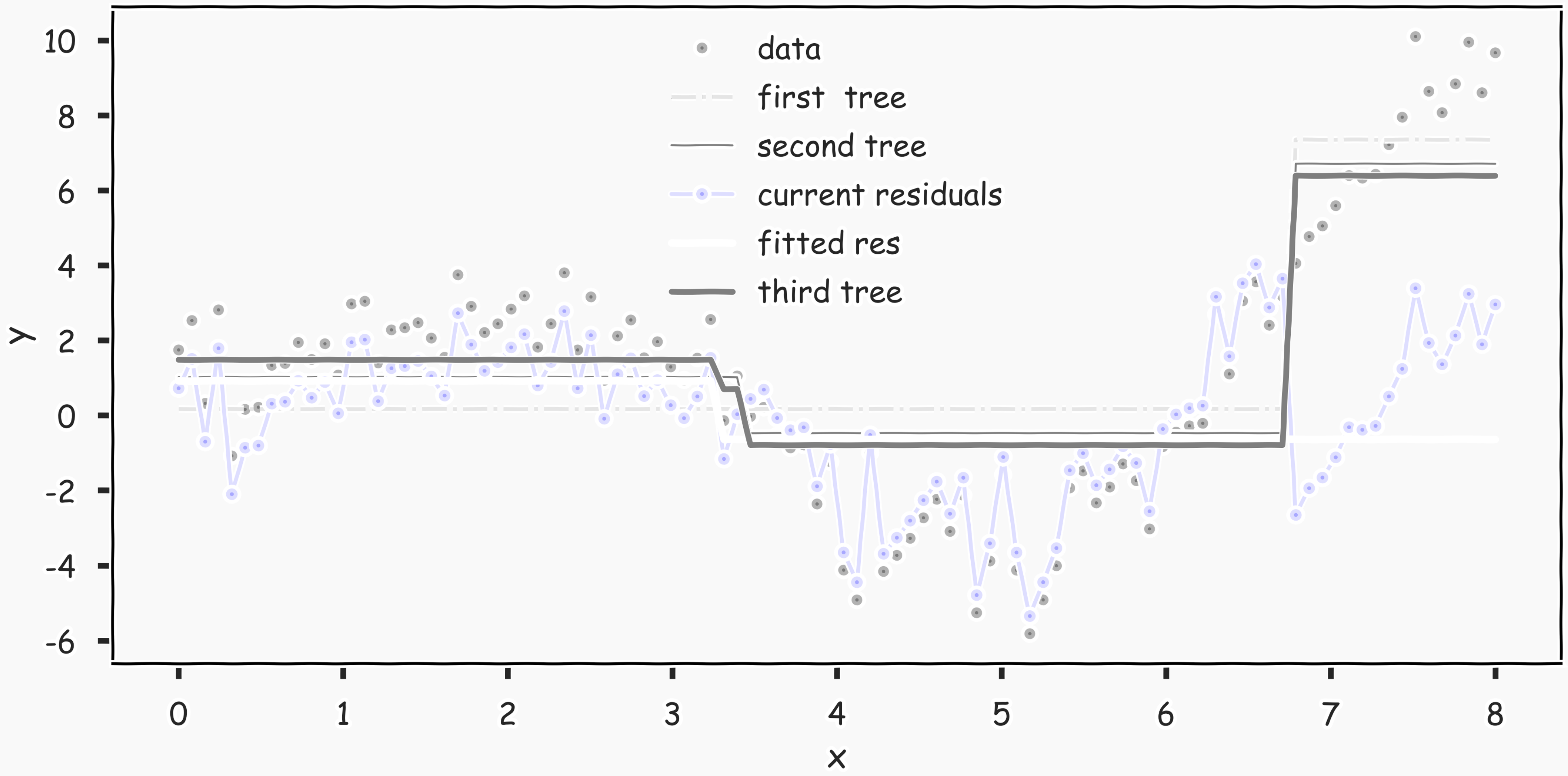
$$\boxed{r_n \leftarrow r_n - \lambda T^i(x_n)}$$



Gradient Boosting: illustration



Gradient Boosting: illustration



Why Does Gradient Boosting Work?

Intuitively, each simple model $T^{(i)}$ we add to our ensemble model T, models the errors of T.

Thus, with each addition of $T^{(i)}$, the residual is reduced

$$\boxed{r_n - \lambda T^{(i)}(x_n)}$$

Note that gradient boosting has a tuning parameter, λ.

If we want to easily reason about how to choose λ and investigate the effect of λ on the model T, we need a bit more mathematical formalism.

In particular, how can we effectively descend through this optimization via an iterative algorithm?

We need to formulate gradient boosting as a type of ***gradient descent***.

Gradient Boosting as Gradient Descent

Often in regression, our objective is to minimize the MSE

$$\boxed{\text{MSE}(\hat{y}_1, \ldots, \hat{y}_N) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$$

Treating this as an optimization problem, we can try to directly minimize the MSE with respect to the predictions

$$\begin{aligned} \nabla \text{MSE} &= \left[\frac{\partial \text{MSE}}{\partial \hat{y}_1}, \ldots, \frac{\partial \text{MSE}}{\partial \hat{y}_N} \right] \\ &= -2 \left[y_1 - \hat{y}_1, \ldots, y_N - \hat{y}_N \right] \\ &= -2 \left[r_1, \ldots, r_N \right] \end{aligned}$$

The update step for gradient descent would look like

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

Gradient Boosting as Gradient Descent (cont.)

There are two reasons why minimizing the MSE with respect to $\hat{y}_n$'s is not interesting:

- We know where the minimum MSE occurs: $\hat{y}_n = y_n$, for every *n*.
- Learning sequences of predictions, $\hat{y}_n^1, \dots, \hat{y}_n^i$, ..., does not produce a model. The predictions in the sequences do not depend on the predictors!

Gradient Boosting as Gradient Descent (cont.)

$\boxed{\hat{y}_n \leftarrow \hat{y}_n + \lambda r_n}$

The solution is to change the update step in gradient descent. Instead of using the gradient - the residuals - we use an ***approximation*** of the gradient that depends on the predictors:

$$\boxed{\hat{y} \leftarrow \hat{y}_n + \lambda\, \hat{r}_n(x_n), \qquad n = 1, \ldots, N}$$

In gradient boosting, we use a simple model to approximate the residuals, $\hat{r}_n(x_n)$, in each iteration.

Motto: gradient boosting is a form of gradient descent with the MSE as the loss (objective) function.

Technical note: note that gradient boosting is descending in a space of models or functions relating x_n to y_n!

Gradient Boosting as Gradient Descent (cont.)

But why do we care that gradient boosting is gradient descent?

By making this connection, we can import the massive amount of techniques for studying gradient descent to analyze gradient boosting.

For example, we can easily reason about how to choose the learning rate λ in gradient boosting.

Choosing a Learning Rate

Under ideal conditions, gradient descent iteratively approximates and converges to the optimum.

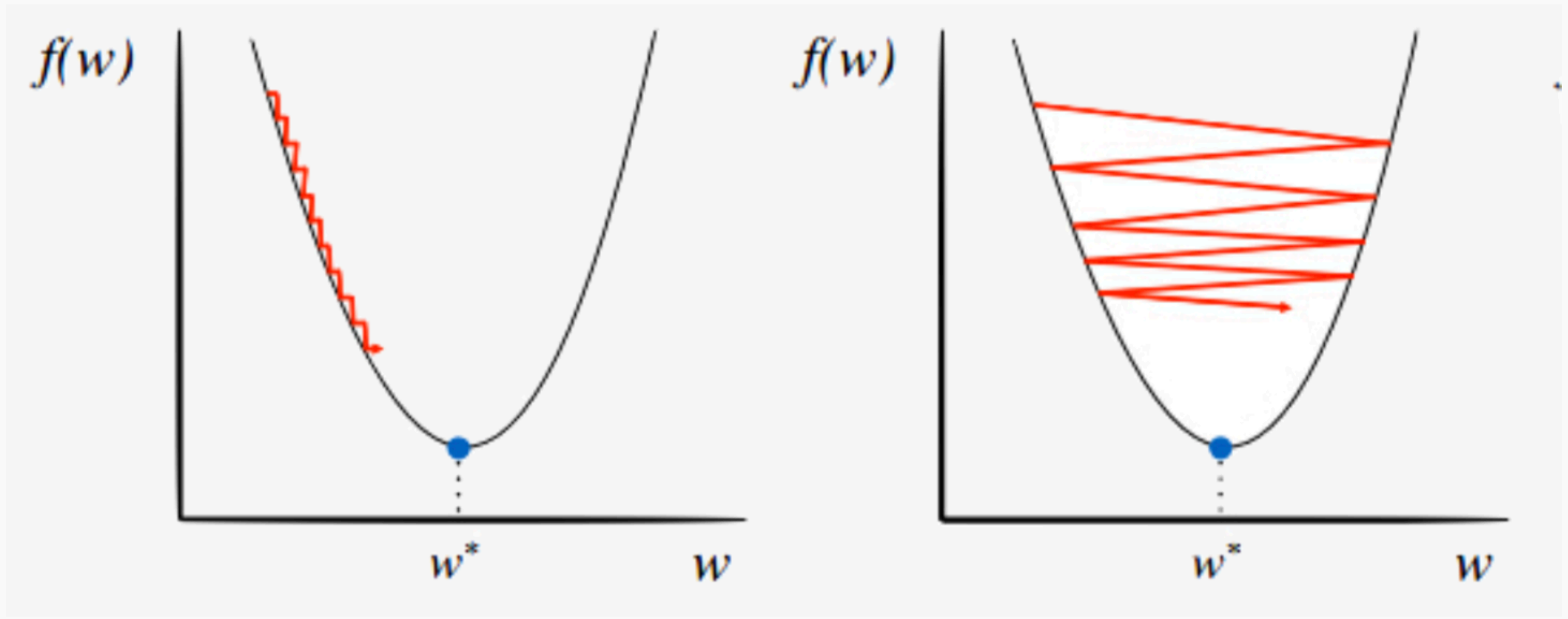
When do we terminate gradient descent?

- We can limit the number of iterations in the descent. But for an arbitrary choice of maximum iterations, we cannot guarantee that we are sufficiently close to the optimum in the end.
- If the descent is stopped when the updates are sufficiently small (e.g. the residuals of T are small), we encounter a new problem: the algorithm may never terminate!

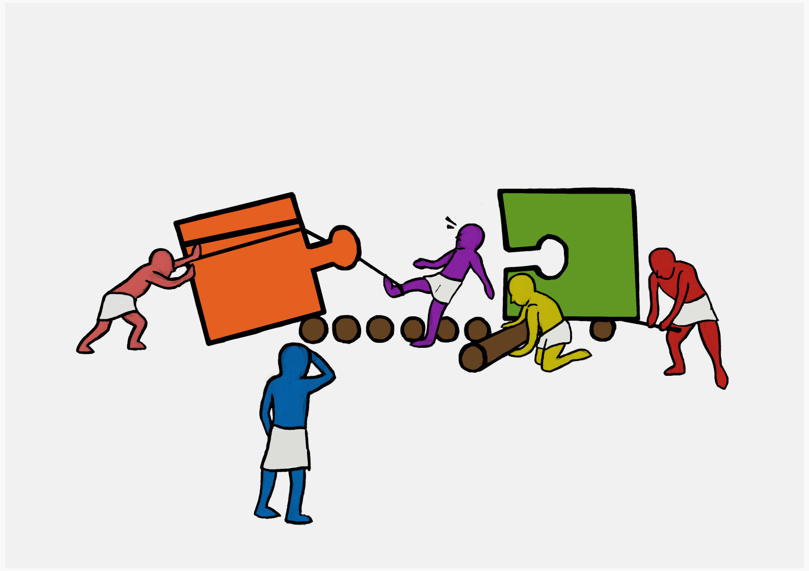
Both problems have to do with the magnitude of the learning rate, λ.

Choosing a Learning Rate

For a constant learning rate, λ, if λ is too small, it takes too many iterations to reach the optimum.



If λ is too large, the algorithm may ‘bounce’ around the optimum and never get sufficiently close.



Today's lucky student: Anyone awake!

Exercise goal

Regression with Boosting