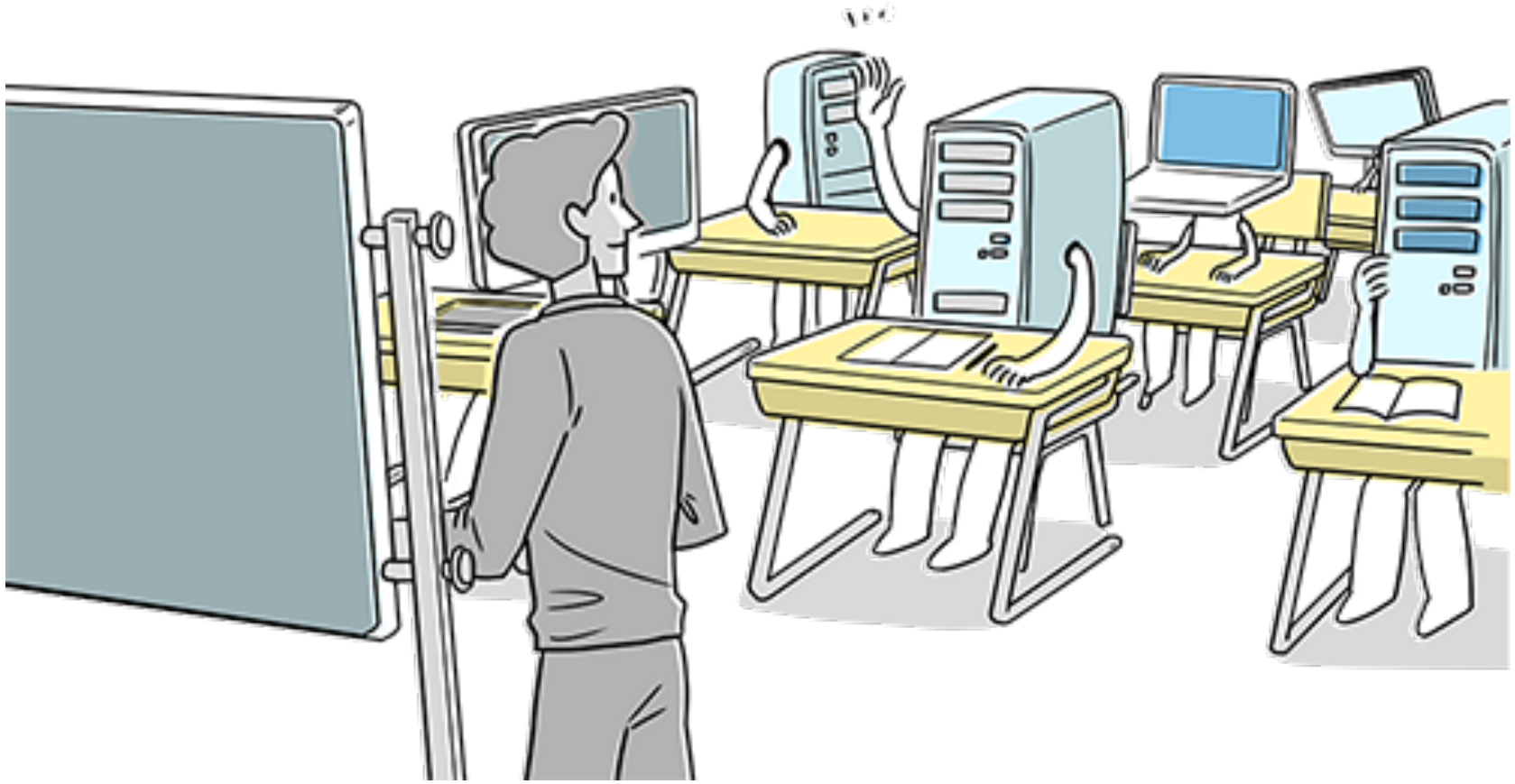
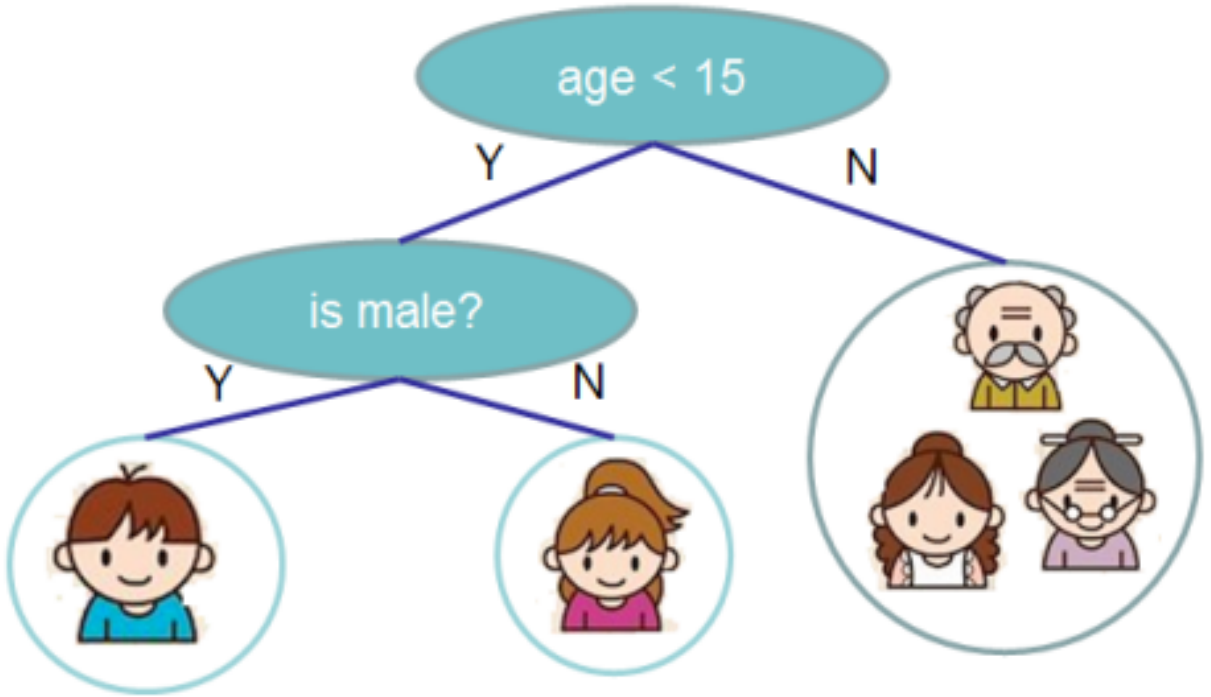


ENSEMBLE MODELS

Recap: Decision trees

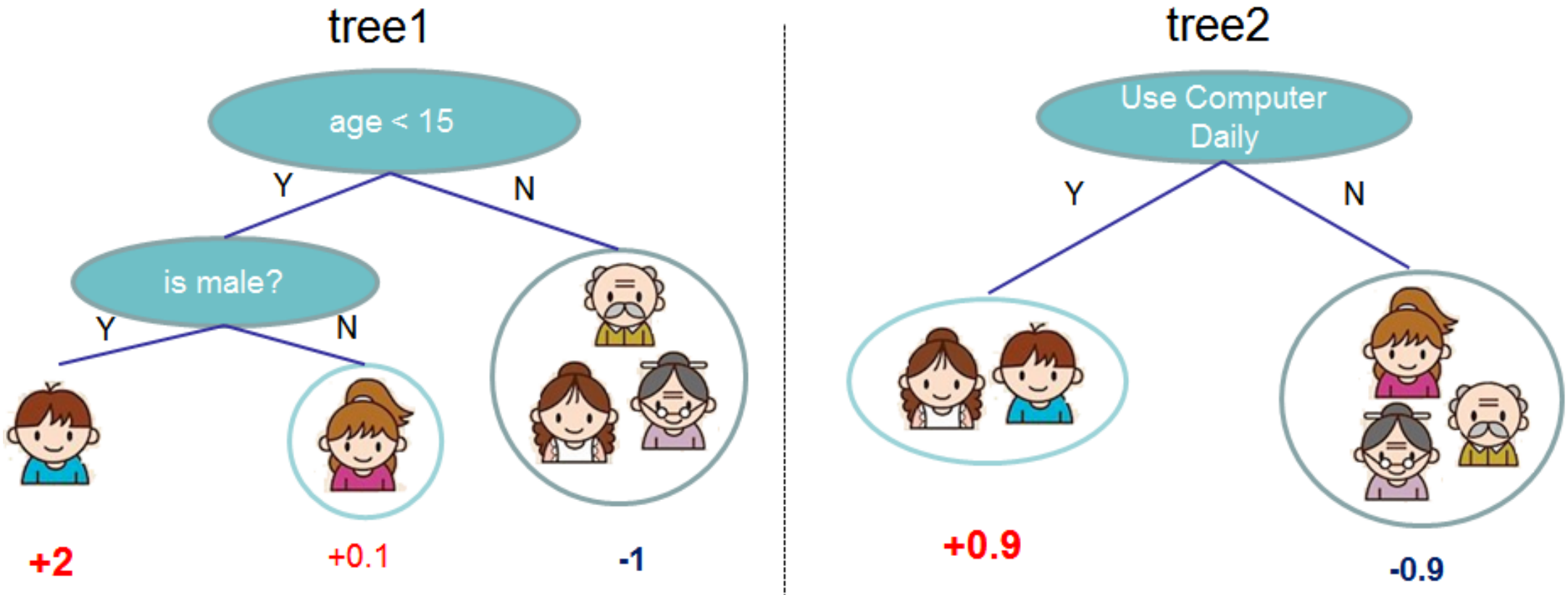
Input: age, gender, occupation, …

Does the person like computer games



prediction score in each leaf → **+2** +0.1 **-1**

Tree Ensemble



f() = 2 + 0.9= 2.9 f()= -1 - 0.9 = -1.9

Prediction is sum of scores predicted by each of the tree

Tree Ensemble methods

- Produce better performance than a single tree while keeping all the advantage of decision trees
  - **Require almost no input preparation:** they can handle binary features, categorical features, numerical features without any need for scaling.
  - **Perform implicit feature selection** and provide a pretty good indicator of feature importance.
  - **Learn higher order interaction between features**
  - **Can be scalable** and are used in Industry

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

Random Forest

- Combine multiple trees, each fit a random sample and features of the original data
- Repeat n times (grow n trees):
  1. Draw a sample with~~out~~ replacement (bootstrap) from the data set
  2. Train a decision tree

     Until the tree is maximum size

     Choose next leaf node

     Select ***m features at random*** from all p features where **m << p**

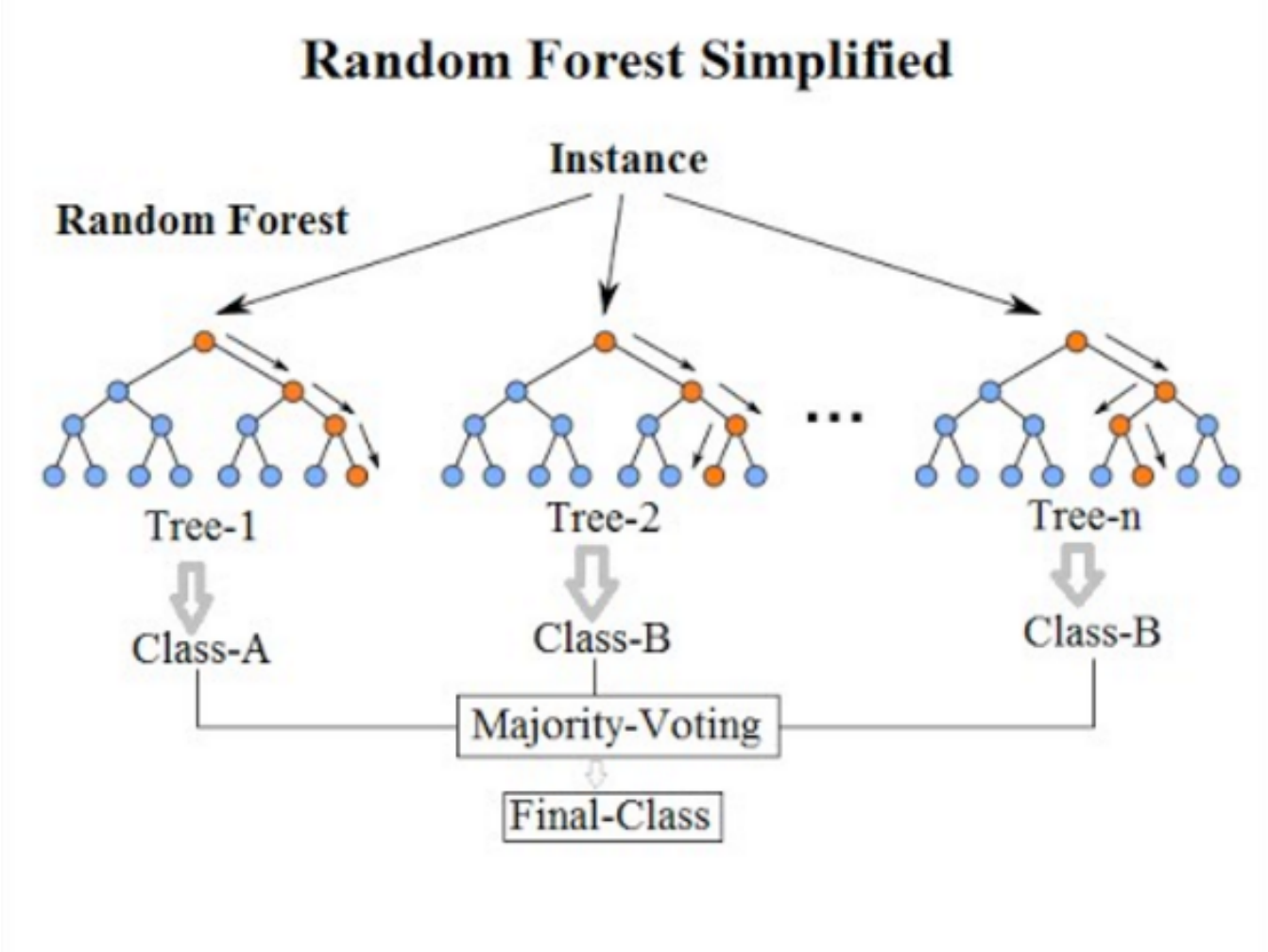
     Pick the best splitting as usual
- Make a prediction by majority vote among the n trees

Random Forest

- Random forests tries to

"de-correlating" the trees

- The goal is to reduce variance



- Each tree has the same expectation and is independent with each other, thus the average prediction produces lower variance than a single prediction

Random Forest: Key Hyper-parameters

- n_estimators: number of trees to use in ensemble
  Should be larger for larger datasets to reduce overfitting

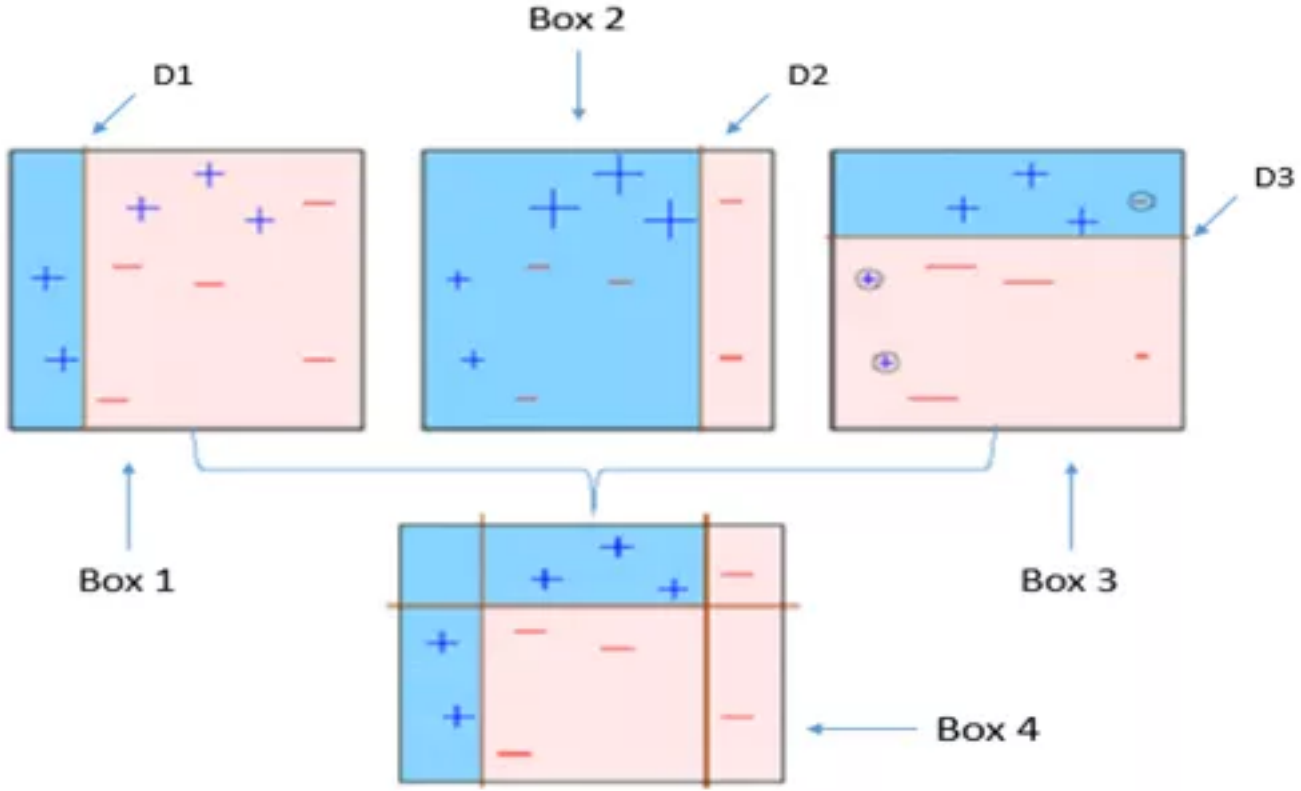
- max_features: number of “m” features sampled in each split, influences the diversity of tress in the forest
  Default works well in practice, but adjusting may lead to some further gains.

- max_depth: controls the depth of each tree

Gradient Boosting Trees(GBT)

- Boosting:

Ensemble of weak leaners



- Fits consecutive trees where each solves for the net error of the prior trees

Understanding GBT

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.
- You can add an additional model (regression tree) h to F, so the new prediction will be $F(x) + h(x)$.

Understanding GBT

Simple solution:

You wish to improve the model such that

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

Or, equivalently, you wish

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

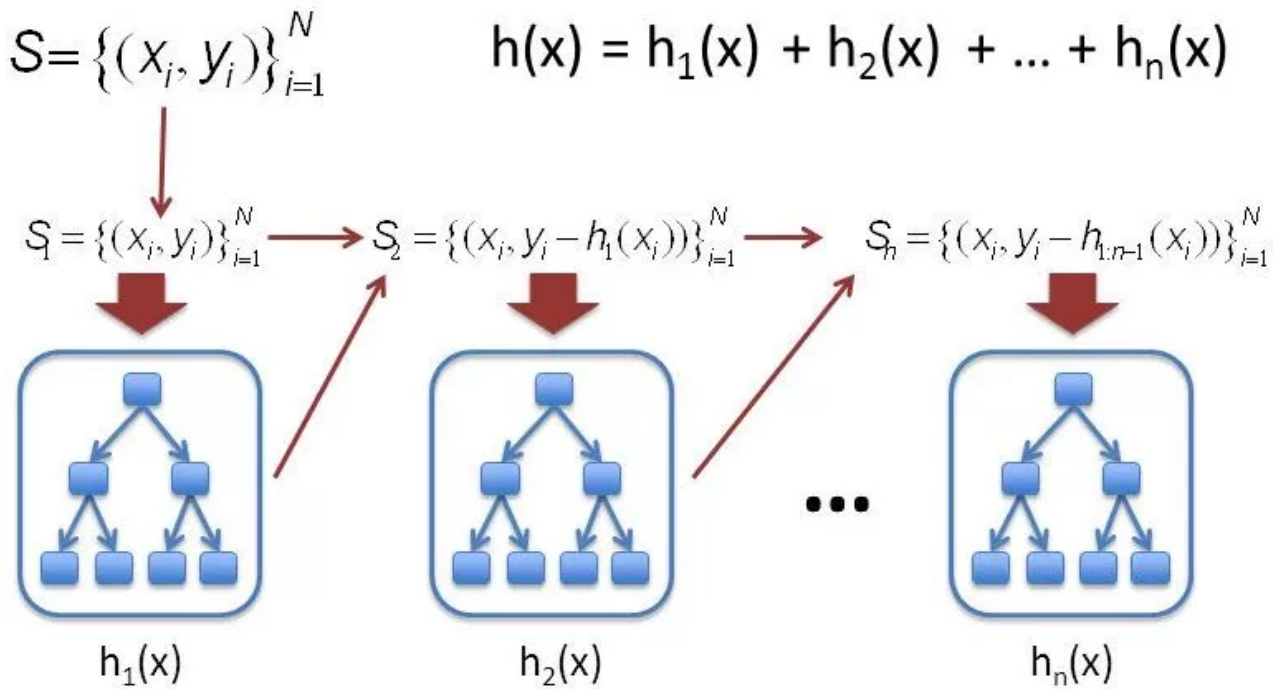
Just fit a regression tree h to data
$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \dots, (x_n, y_n - F(x_n))$
Congratulations, you get a better model!

Understanding GBT

$y_i - F(x_i)$ 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...



Understanding GBT

How is this related to gradient descent?

Loss function $L(y, F(x)) = (y - F(x))^2/2$

We want to minimize $J = \sum_i L(y_i, F(x_i))$

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

So we can interpret residuals as negative gradients.

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

Understanding GBT

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**!

GBT: Key Hyper-parameters

- n_estimators: number of trees to use in ensemble
  Should be larger for larger datasets to reduce overfitting

- learning_rate: controls emphasis on fixing errors from previous iteration
  Default works well in practice, but adjusting may lead to some further gains.

The above two should be tuned together (low learning rate -> more trees)

- max_depth: controls the depth of each tree

Tree Ensemble methods

Random Forest (RF)

- Much easier to tune hyper-parameters
- More robust to overfitting

Gradient Boosting Trees (GBT)

- Very sensitive to hyper-parameters tuning
- Prone to overfitting if not carefully tune the parameters

RF is is more "plug'n'play" than GBT.
However, it is generally true that a well-tuned GBT will outperform a RF.

Very Simplified Cheat Sheet