Supervised learning, uses training data to predict a target variable.

Model and Parameters

- Parameters are the coefficients (theta)
- The task of training the model is finding the best parameters that best fit the training data xi and labels yi.

Supervised learning, uses training data to predict a target variable.

Model and Parameters

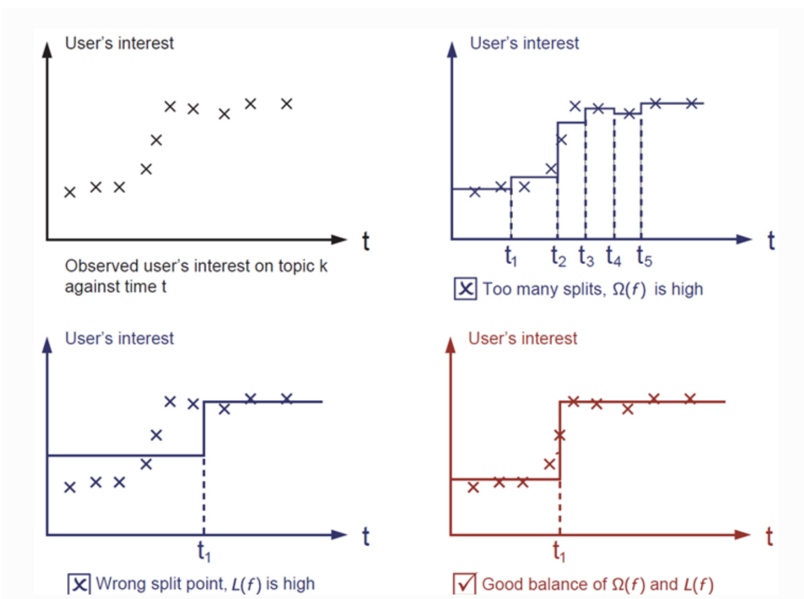
- Parameters are the coefficients (theta)
- The task of training the model is finding the best parameters that best fit the training data xi and labels yi.

Training Loss and Regularization

- Objective functions consist of two parts, training loss and regularization term.

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

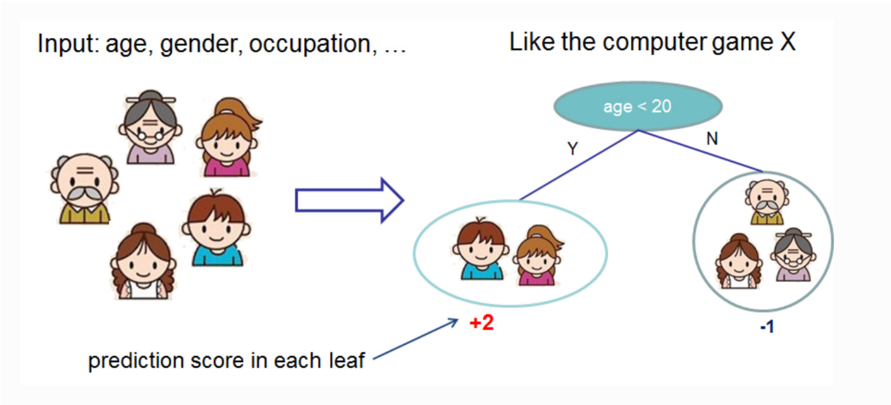
- L is the training loss function, omega is the regularization terms
- A common loss function is mean squared error, others include logistic loss and r squared.
- The regularization is often what people forget to add
  - It controls the complexity of the model and helps to avoid overfitting



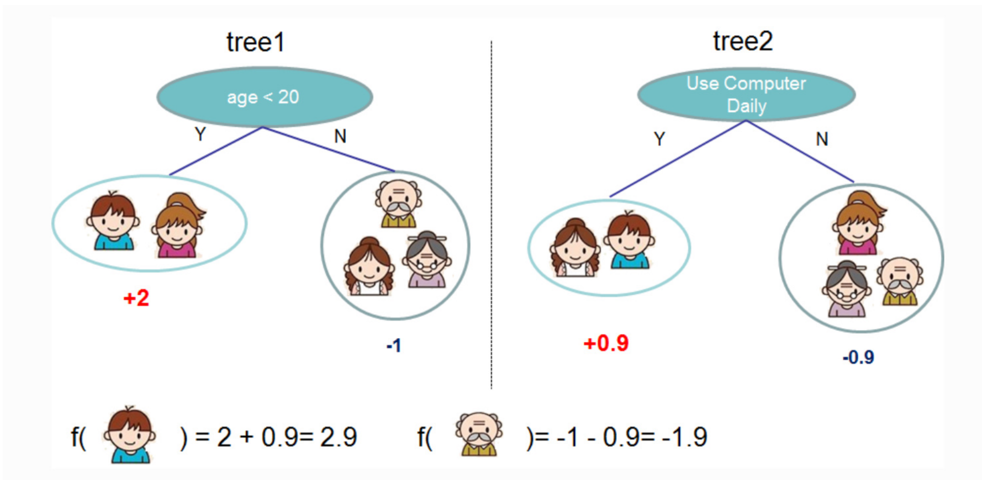
- The general principle is that we want both a simple and predictive model.
  - This trade off between simplicity and predictability is called the bias-variance tradeoff

Decision Tree Ensembles

- The tree ensemble model consists of a set of classification and regression trees (CART).
  - The example below shows a classification of whether someone will like a comput er game



- Each leaf only contains decision values and a real score is associated with each leaf
  - This gives better interpre tations that go beyond classification
  - Anothe r example is shown below:



Mathmatically:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- This is an example of an ensemble of two Rees

K = # of trees

- The scores are summed up to get the final score

f_k = function in space $\mathcal{F}$, $\mathcal{F}$: set of all possible CART;

Objective function:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} w(f_k)$$

w_{f_k} = complexity of the tree

- Random forest also uses tree ensembles

Tree Boosting

- We should train by defining and objective function and optimizing it
- What are the parameters of trees?
  - We need to learn the functions f, which each contain the tree structure and leaf scores caves.
  - Learning all the trees at once would be very difficult, so instead we use an additive strategy as seen below

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0\\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)\\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)\\
&\dots\\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

It remains to ask: which tree do we want at each step? A natural thing is to add the one that optimizes our objective.

$$\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i)\\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \omega(f_t) + \text{constant}
\end{aligned}$$

If we consider using mean squared error (MSE) as our loss function, the objective becomes

$$\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \omega(f_i)\\
&= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \omega(f_t) + \text{constant}
\end{aligned}$$

- In more complicated forms, the Taylor Series representation can be used for simplification
- XGB also supports custom loss functions

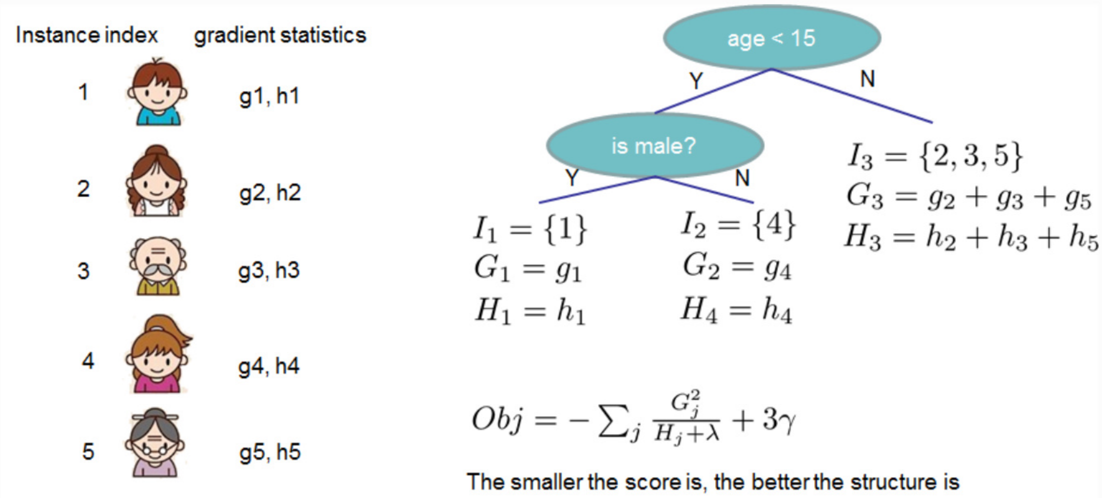
Model Complexity

- In XGB the regularization term is defined formally so that users can get a better idea of what is being learned and obtain models that perform well.

Here w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves. In XGBoost, we define the complexity as

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

The Structure Score



If all this sounds a bit complicated, let's take a look at the picture, and see how the scores can be calculated. Basically, for a given tree structure, we push the statistics g_i and h_i to the leaves they belong to, sum the statistics together, and use the formula to calculate how good the tree is. This score is like the impurity measure in a decision tree, except that it also takes the model complexity into account.

Learn the Tree Structure

- Trees can be optimized by looking at leafs from other trees, then using the structure score to tell if the adding the leaf would be beneficial or not
- This is called pruning
- There are some edge cases where this additive method fail