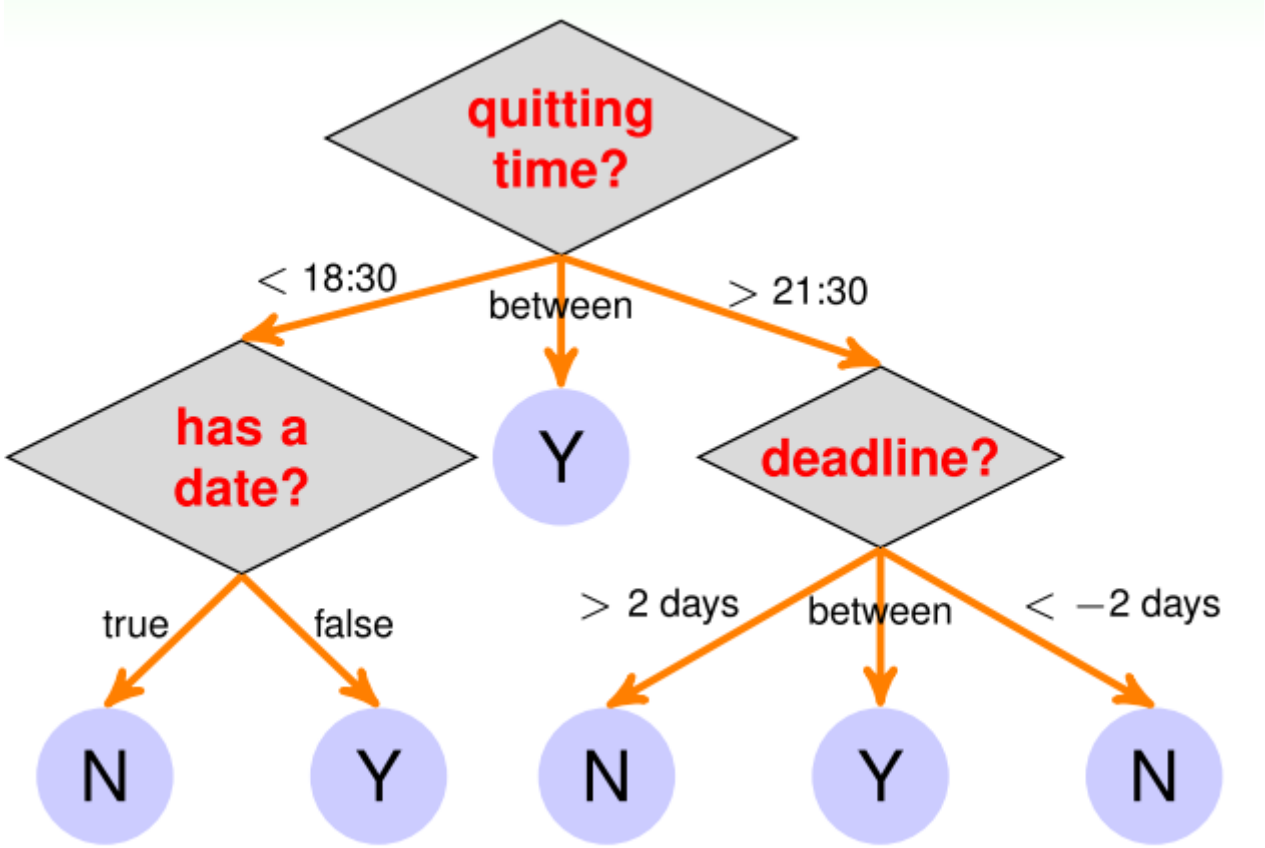
Decision tree and Random forest

JrPhy

Introduction

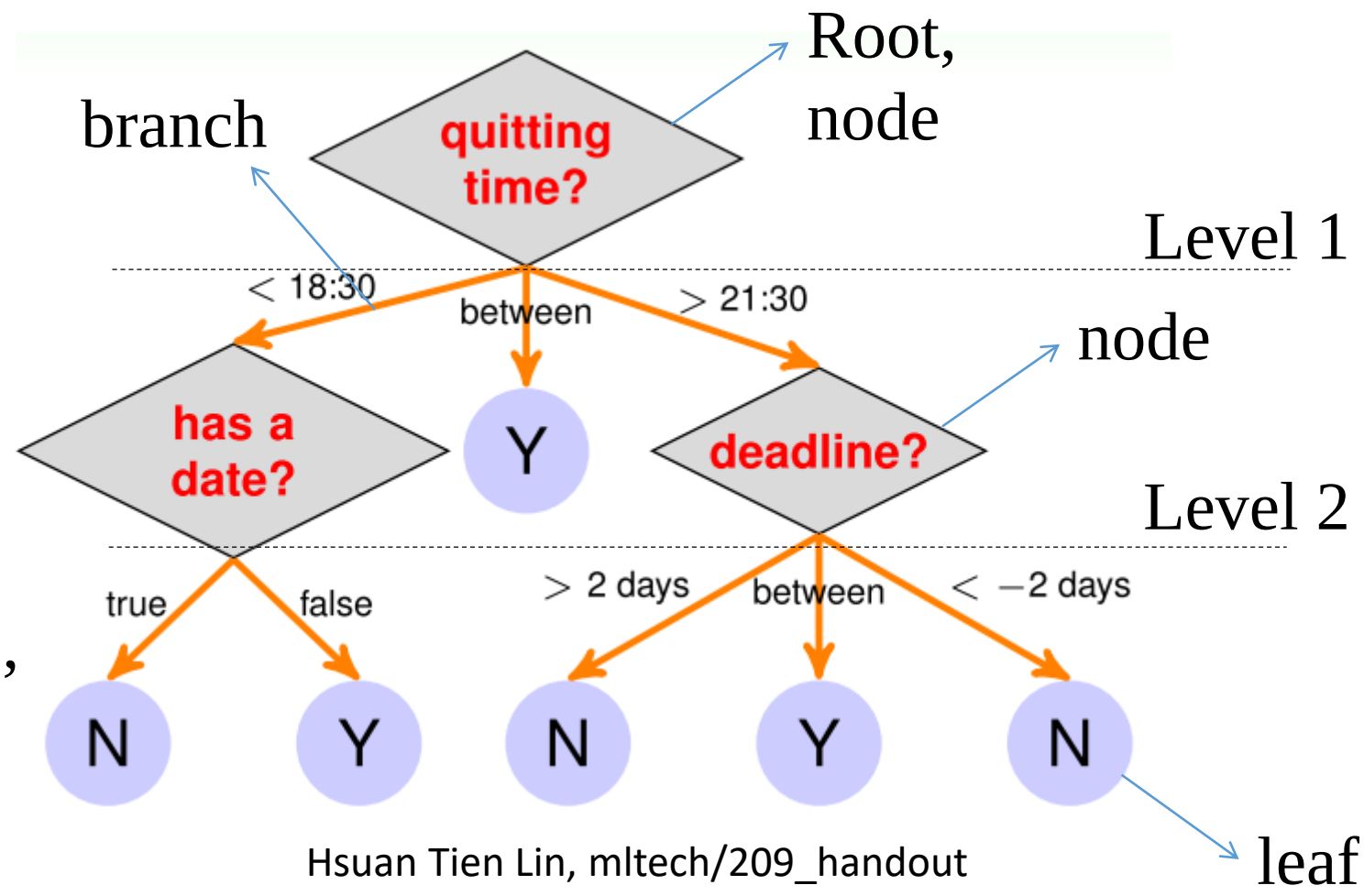
- Decision tree is a method and it mimics the process of a human making decision, it's easy to interpret the criteria, but the theory is hard to prove.
- In the data structure, decision tree is a tree with the conditions.
- It's realized by linked list in C/C++



Hsuan Tien Lin, mltech/209_handout

Language in tree

- A node splits some branches, then it's called child. The branch are split by a node, then it's called parent.
- There are some nouns the same as family tree, like parent, child,



Hsuan Tien Lin, mltech/209_handout

A Basic Algorithm for Decision tree

- $G(x) = \sum_{i=1}^{N} q_i(x) g_i(x)$, $q_i(x)$ is the condition, $g_i(x)$ is the hypothesis.
- By the number of conditions, the tree can be binary or multi.

function DecisionTree(data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$)
if termination criteria met
    return base hypothesis $g_t(\mathbf{x})$
else
    1. learn branching criteria $b(\mathbf{x})$
    2. split $\mathcal{D}$ to C parts $\mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\}$
    3. build sub-tree $G_c \leftarrow$ DecisionTree($\mathcal{D}_c$)
    4. return $G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!] G_c(\mathbf{x})$

Hsuan Tien Lin, mltech/209_handout

Classification **A**nd **R**egression **T**ree

- A famous algorithm of decision tree is CART, from California Statistical Software.
- It uses binary tree in the algorithm, and returns a constant
- For classification, it returns the majority of the constant.
- For regression with square error, it returns the average of the constant.
- CART uses decision stump to split into 2 parts D_1 and D_2, and calculate the purity, that is, minimize the impurity.

$$b(x) = \underset{\text{decision stump } h(x)}{\arg\min} \sum_{i=1}^{2} \left| D_c \text{ with } h(x) \right| \bullet impurity(D_c \text{ with } h(x))$$

Impurity function

by E_{in} of optimal constant

- regression error:

$$\text{impurity}(\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \bar{y})^2$$

with $\bar{y}$ = average of $\{y_n\}$

- classification error:

$$\text{impurity}(\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}[\![y_n \neq y^*]\!]$$

with y^* = majority of $\{y_n\}$

for classification

- Gini index:

$$1 - \sum_{k=1}^{K}\left(\frac{\sum_{n=1}^{N}[\![y_n = k]\!]}{N}\right)^2$$

—all k considered together

- classification error:

$$1 - \max_{1\le k\le K}\frac{\sum_{n=1}^{N}[\![y_n = k]\!]}{N}$$

—optimal $k = y^*$ only

Hsuan Tien Lin, mltech/209_handout

Terminate condition

- all y_i are the same: impurity = 0 →$g_i(x) = y_i$
- all x_i are the same: no decision stumps
  - CART: **fully-grown** tree with constant leaves
- But a fully-grown tree means $E_{in}(G) = 0$ if all x_i are different, so it needs to regularize.
- A method uses the NumberOfLeaves to regularize $\Omega(G)$

$$\underset{\text{all possible } G(x)}{\arg\min} \; E_{in}(\mathrm{G}) + \lambda\Omega(\mathrm{G})$$

- $\mathrm{G}^{(0)}$ is fully-grown tree
- $\mathrm{G}^{(i)}$ = argmin $E_{in}(G)$ such that G is one-leaf removed from $\mathrm{G}^{(i\text{-}1)}$

The features

- Some features are numerical, like weight, height, etc.. The decision stump is
  - $b(x) = [x_i \leq n]+1$, n in $\mathbb{R}$
- The others are categorical, like fever, pain, tire, etc.. The decision stump is
  - $b(x) = [x_i \leq n]+1$, S in {fever, pain, tire……}
- If there is no feature, it may use another feature to guess the missing feature.
- For example, if there is no weight data, then we can use the height data to guess its weight
- So CART handle the missing feature easily.

Advantages of CART

- Human-explainable
- Multiclass easily
- Categorical features easily
- Missing features easily
- Efficient non-linear training (and testing)
- Almost no other learning model share all such specialties, except for other decision trees

Random forest

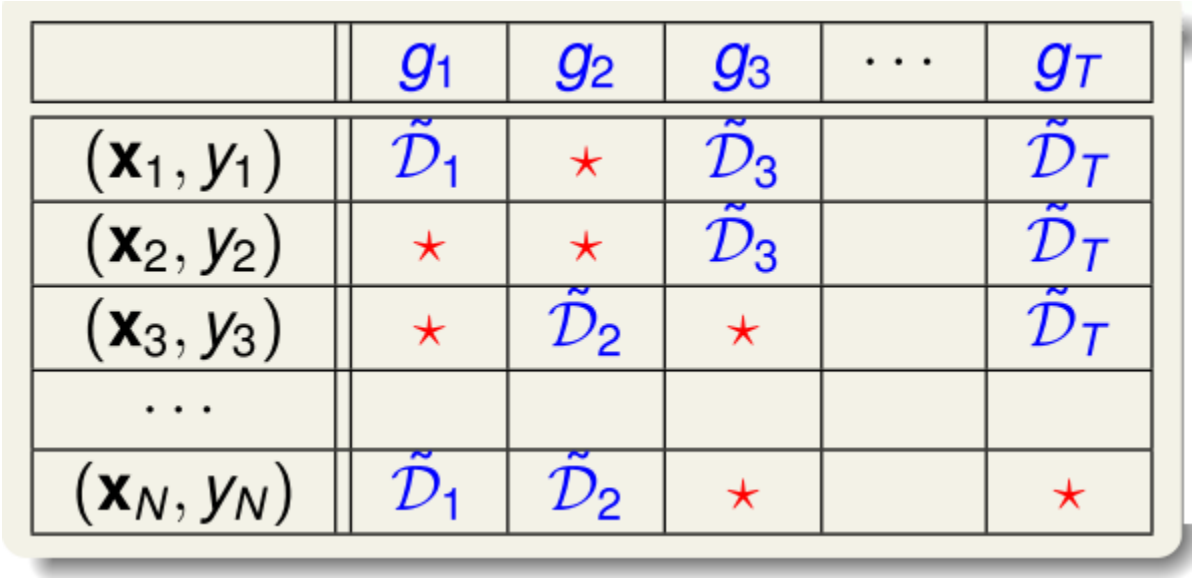
- In BAGging, it separates data into *n* parts, then gets a model from each part, combine the model and reduce the variance by voting or averaging.
- In decision tree, it classifies the data by the conditions, so the variance is very large.
- Now we've learned the BAGging and decision tree algorithm, we want to combine both advantages and make a new algorithm, this algorithm is called **Random forest**.
- **Random forest** = **BAGging** + **fully-grown CART decision tree**

Random forest

- First, it choose a sub-dataset from the sample space, then each dataset is thrown into the CART, so you'll get many trees, then aggregation them and get the model.
- By bootstrap, it's easy to parallel, and may get more diversity model.
- By CART, it inherit the pros of CART
- By aggregation, it eliminates the cons of CART to reduce the variance.
- So it's recommended using the random-subspace in CART.
- There is full of randomness, so it's called Random forest

Out-of-bagging

- If the sample is chosen randomly, then some data may not be chosen
- The star mark means the data is not chosen in i^{th} choice.
- The probability is $p = \left(1 - \frac{1}{N}\right)^N$



	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	$\star$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	$\star$	$\tilde{\mathcal{D}}_2$	$\star$		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\tilde{\mathcal{D}}_2$	$\star$		$\star$

Hsuan Tien Lin, mltech/209_handout

- If N is big enough, then

$$\lim_{N\to\infty} p = \lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N = \lim_{N\to\infty}\left(1+\left(\frac{-1}{N}\right)\right)^N = e^{-1} \approx 0.37$$

Out-of-bagging and Validation

- If the data is not chosen at the i^{th}, then it can be used for validation. But random forest uses aggregation, so it doesn't need to validate each model, but validate the model G^- after aggregating.

	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	$\star$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	$\star$	$\tilde{\mathcal{D}}_2$	$\star$		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\tilde{\mathcal{D}}_2$	$\star$		$\star$

Hsuan Tien Lin, mltech/209_handout

- For example, (x_2, y_2) is not in the training data of g_3, g_T, so if G^- is aggregate(g_3, g_T), then (x_2, y_2) can be used for validate G^-.
- So that you can define your own error of OOB, and it will validate itself by those dataset.

Feature selection

- The data may exist many features, but do we need all feature?
- Some features are similar that get the same information, and some are irrelevant of the prediction, we don’t want those features that waste our computational source.
- The linear model can help us to choose the feature, if you train a linear model, then you will get the vector $\boldsymbol{w}$, a method is using its norm to estimate the importance.
- For nonlinear model, decision tree is a algorithm with feature selection.

Feature selection

- Idea: if the data is important, then the result will be very different when the data is polluted.
- A method is using the noise follows some distributions, but it may change the distribution of original data.
- Here we use permutation test, it disorder the data. For example, the i^{th} data may be j^{th} after permutation, the importance is
    - Importance(i) = performance(D) – performance($D^{(p)}$)
- It suggested by the author of random forest.

Feature selection

- Then how to estimate performance? In the random forest, the author suggested using E_{oob} to estimate. So that the importance can be written as
    - Importance(i) = $E_{oob}(G) - E_{oob}^{(p)}(G)$
- It just needs the OOB data after permutation, so if you need to nonlinear transformation, Random forest is a proper method.
- There are more examples in the Hsuan Tien Lin, mltech/210_handout