Likelihood Machine Learning Seminar II

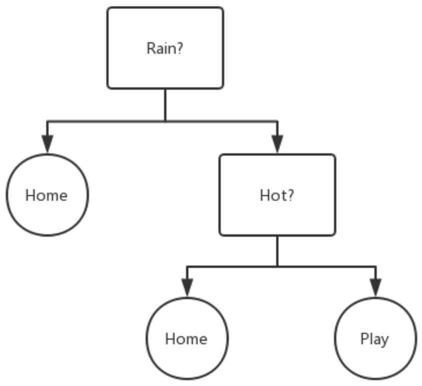
Lecture 5: Decision Tree

Likelihood Lab

XingYu Fu

1. Structure of Decision Tree

Decision tree is a famous machine learning algorithm that is best known for the interpretability of its decision-making process. The following figure is a simple decision tree of homing or playing depending on the current weather.



Each decision tree is comprised of the following parts:

♣ Node:

- Split Node: contains the condition of split (the square in the figure).
- Terminal Node: contains the result of inference (the circle in the figure).

♣ Edge: links to the next stage of the decision-making procedure.

2. Inference Algorithm of Decision Tree

The inference algorithm of decision tree is quite straightforward. Given an unclassified sample $x \in R^n$, starting from the root node of the tree, we let the sample travel in the decision tree until it encounters one terminal node, where we classify x to the result stored in this terminal node. At each split node, we judge if the sample x satisfies the corresponding condition. If yes, then the sample is passed to the left node of the split point. If no, then the sample is passed to the right node of the split point.

In machine learning setup, the condition stored in each split node is usually encoded as a tuple of index and value noted by:

$$(i, v)$$

, which means that if $x_i < v$, then x will be passed to the left node of the split point and vice versa.

2. Construction (training) Algorithm of Decision Tree

The construction of decision tree is a *recursive* procedure. At the root node, we smartly define the split index $i*$ and split value $v*$ of the root node and then split the training data $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ into the left data set :

$$D_l(i*, v*) = \{(x_1^l, y_1^l), \ldots, (x_{m_l}^l, y_{m_l}^l)\}$$

and the right data set:

$$D_r(i*, v*) = \{(x_1^r, y_1^r), \ldots, (x_{m_r}^r, y_{m_r}^r)\}$$

with respect to the condition defined by $(i*, v*)$. Then the construction process carries on to build the left tree and the right tree under the root node using $D_l(i*, v*)$ and $D_r(i*, v*)$ respectively.

Now we have three questions

1) How to find the proper split index and split value of the root node?

2) How to end the recursion?

3) How to determine the inference result of terminal node?

- For question 1, we picks the tuple $(i*, v*)$ that minimizes some criteria, e.g. *Gini index* and *entropy*. To be specific, taking entropy as an example, we choose the best split index and split value according to:

$$(i*, v*) \in \arg\min_{(i,v)} E(D_l(i, v)) \cdot m_l + E(D_r(i, v)) \cdot m_r$$

  where $E(\cdot)$ is the entropy function that measures the randomness of a set.

- For question 2, we end the recursion if the node is too deep or the number of training data to be split is too small. The reason why we end it here is to avoid the over-fitting problems caused by decision trees that are too intricate.

- For question3, we define the inference result of the terminal node, where we end the recursion process, as the category with the maximal frequency in the remaining training data set, i.e. majority voting.