Python Machine Learning: The 3rd Book Circle

Classification Algorithm III: Decision Tree, Random Forest and kNN

Jianan Liu

Gothenburg

Dec, 7, 2016

Disclaimer

All opinions and statements in this presentation are mine and do not in any way represent the company
Any comment or correction of error is welcome, please contact me
chisyliu@hotmail.com

The Third Book Circle of Python Machine Learning

In this presentation belongs to **algorithm** part of the book

- If you have time, please read the book first. This slide could be used as complementary resource for the book
- We will try to go through every algorithm in the second half of chapter 3 of book Python Machine Learning, also show the mathematics behind each algorithm. But we only use the mathematics conclusion to explain algorithm rather than showing mathematical derivation
- All of us need to debug the python code, in order to get practice of implementing machine learning algorithm

Overview

Data in Chapter 3

From this slide we go through the classification algorithms decision tree, random forest and kNN in chapter 3

Training data set ($\mathbf{y}$, $\mathbf{X}$)

Suppose we have L samples, each sample has D features/dimensions(So the input $\mathbf{X}$ is L by D matrix, label $\mathbf{y}$ is L by 1 vector). If we only consider the I^{th} data sample pair, $\mathbf{x}_I$ is 1 by D vector which represents D features/dimension for I^{th} training data sample, $\mathbf{w}$ is D by 1 vector which represents weight and y_I represents the label of I^{th} data sample

Outline for Section 1

Scikit Learn Decision Tree

- This page introduces where to find the corresponding methods in scikit learn package, please come back current page and check the link after finishing up all the contents in this section
- The description of decision tree algorithm in Scikit Learn package regarding how to setup the parameters and so on could be seen here:
  - Link: Decision tree
- Decision tree classifier:
  - Link: DecisionTreeClassifier
- Decision tree regressor:
  - Link: DecisionTreeRegressor

Basical Understanding of Decision Tree Algorithm

- The decision tree algorithm grows the tree from root, starts from top to down, until all the training data are classified perfectly.
- Basically the logic of algorithm looks like the 'tree G' calls 'tree G' recursively by following different 'branch B'. It is similar to 'call the function in C language recursively'(The function f(x) calls itself inside itself)
- So the question comes. Among all the features, which one will be used first to branch the tree into two (internal) nodes?
- We will answer this question by figuring out basic logic in decision tree
- But before that we start from checking what is entropy. Cause this definition is very important in decision tree

Basical Understanding of Decision Tree Algorithm

What is entropy?

- I think most of us learned entropy in **information theory** course
- Entropy of random variable X is defined:
  $H(X) = -\sum_{l=1}^{L} P(X = l) \log_2 P(X = l)$
- Where $H(X)$ stands for 'expected number of bits needed to encode randomly drawn value of X(under most efficient code)'

Basical Understanding of Decision Tree Algorithm

Let's define more entropies, and most importantly mutual information(Information Gain, IG)

- Conditional entropy $H(X|Y=a)$ of X given $Y=a$ is: $H(X|Y=a) = -\sum_{l=1}^{L} P(X=l|Y=a)\log_2 P(X=l|Y=a)$
- And conditional entropy $H(X|Y)$ of X given Y: $H(X|Y) = \sum_{a\in A} P(Y=a)H(X|Y=a)$
- Eventually mutual information(Information Gain) $I(X,Y)$ of X and Y: $I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Basical Understanding of Decision Tree Algorithm

How to use entropies and mutual information(Information Gain, IG) on decision tree? Let's see the good explanation from Tom M. Mitchell's machine learning course in Carnegie Mellon University

- We check what is the physical meaning behind mutual information(Information Gain, IG) first by rewriting it as: $Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$
- This means information gain is the mutual information between input attribute A and target variable Y, on the other word, information gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A

Basical Understanding of Decision Tree Algorithm

- This means if we use feature gives the higher information gain over all L training data samples to be as criteria for branching, the expected reduction in entropy will be bigger, then the remaining information is smaller which can be simpler to be classified(lower complexity to classify)
- As we mentioned, the decision tree is somehow 'generate the tree recursively'. For each loop, we try to find which feature(among all non-examined feature) gives the maximum information gain over all L training data samples, and use this feature to branch for current loop. Then repeat this procedure until all data samples are classified perfectly

Basical Understanding of Decision Tree Algorithm

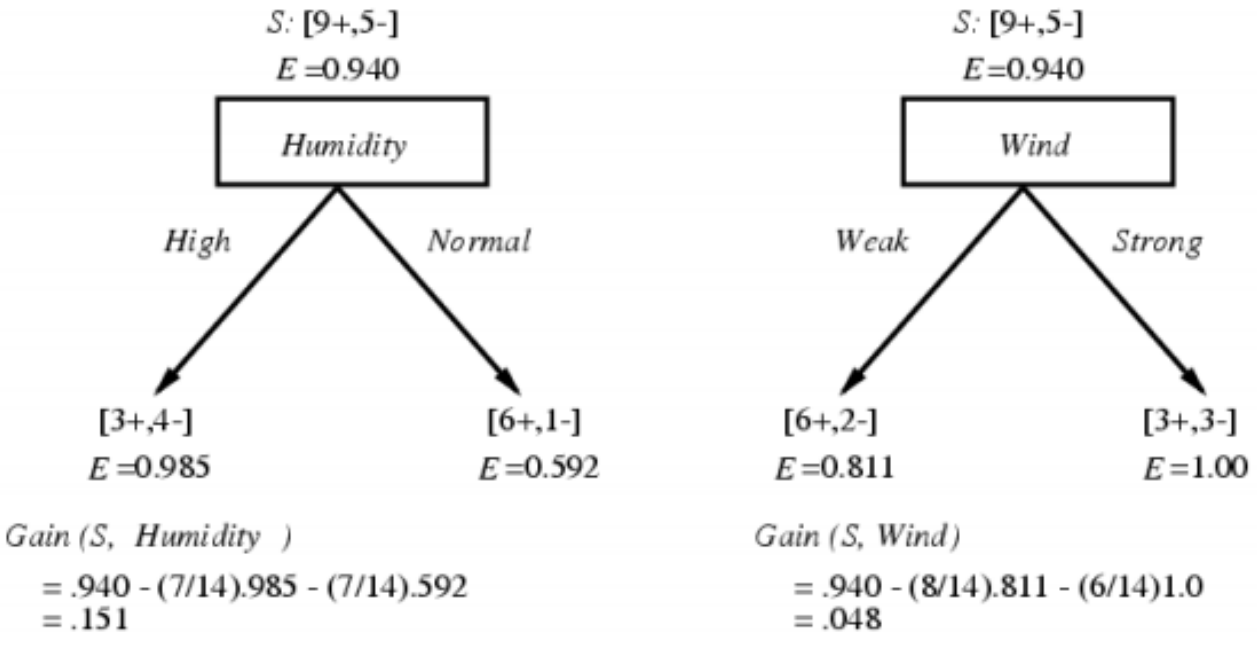
Now let have a look at example from Tom M. Mitchell's machine learning course, which explains the rule of choosing which feature to be used fro branching by searching the feature has maximum mutual information clearly

- Here is the training data samples

Day	Outlook	Temperature	Humidity	Wind	PlayTeni
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Basical Understanding of Decision Tree Algorithm

- If we only select what could be next feature for branching between humidity and wind, which one is the better feature for branching?



- Clearly humidity according to max mutual information criteria

Decision Tree Algorithm: ID3

By using criteria of choosing the feature who has maximum mutual information(Information Gain) to be as prioritized (internal)node to branch tree, we will get ID3 decision tree algorithm

Decision Tree Algorithm: ID3

Method: DesicionTreeClassifier()

```
1: if Termination criteria met then
2:     //e.g. all the records in the dataset have the same classification, return that classification.
3:     Return that classification
4: else
5:     for node do in current tree
6:         for available feature do in training data set features
7:             Calculate the selected feature for branching by picking the feature with maximum information gain
8:         end for
9:         DesicionTreeClassifier(selected feature)
10:    end for
11: end if
```

Decision Tree Algorithm: CART

Decision Tree Algorithm: C4.5

Advantage of Decision Tree

- Decision tree explains all the way decisions have been made, it is easily explainable to human.
- Simple to be understood by NOT only engineers but human being(e.g. support service unit)
- Efficient in training and prediction

Drawback of Decision Tree

- Less theoretical explanation(It is just 'looks good and follow the human being's logic but not guarantee the result theoretically)
- Lots of heuristic/parameters to setup by trick:)

Outline for Section 2

Aggregation Method

Sometime, we want to get more than one hypothesis/models for the same training data set in order to find the best one, or to combine all models together for fetching the better performance. **This is called aggregation/combination**.

- We still consider classification problem as example, but aggregation works for regression as well
- Let us suppose we have T different hypothesis/models $y = g_t(\mathbf{x})$, t = 1, 2, ..., T, for same observed training data samples $\mathbf{X}$ which has L vector $\mathbf{x}$
- How many ways to combine the T hypothesis/models?

Aggregation Method

According to the machine learning technique course by Hsuan-Tien Lin, some of ways to define aggregation model $G(\mathbf{x})$ by maxing/combination of T different models $y = g_t(\mathbf{x})$:

- Mix the predictions from all your models uniformly, it means let all the model added linearly
- Mix the predictions from all your models non-uniformly, it means let all the model added linearly but with weights
- Combine the predictions conditionally, it means if [t satisfies some condition] collect the model t

By doing aggregation/combination of different models, we have chance to enhance the performance of our classification. It is quite similar to what we do with multiple Tx/Rx antennas to get diversity

Aggregation Method

What do these ways actually mean in mathematical sense?:

- Mix the predictions from all your models uniformly, it means let all the model added linearly
  $G(\mathbf{x}) = sign(\sum_{t=1}^{T} 1 \cdot g_t(\mathbf{x}))$
- Mix the predictions from all your models non-uniformly, it means let all the model added linearly but with weights
  $G(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t \cdot g_t(\mathbf{x}))$, where $\alpha_t \geq 0$
- Combine the predictions conditionally, it means if [t satisfies some condition] collect the model t
  $G(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t(\mathbf{x}) \cdot g_t(\mathbf{x}))$, where $\alpha_t(\mathbf{x}) \geq 0$
  $\alpha_t(\mathbf{x}) > 0$ if t satisfies some condition
  else $\alpha_t(\mathbf{x}) = 0$

Blending, Bagging

- In case all the T models $y = g_t(\mathbf{x})$, t = 1, 2, ..., T, are already gotten(e,g, already learned from observed training data set) before selecting/combination/aggregation. This is called blending
- In case all the T models $y = g_t(\mathbf{x})$, t = 1, 2, ..., T, are NOT gotten(e,g, NOT learned from observed training data set) before combination/aggregation/selecting, we actually learn the model while combination/aggregation/selecting. This is called bagging

Bootstrap, Identical Idea as Transmit/Receive Diversity

Previously we discussed the idea of aggregation, we can have better performance by combining different models from different algorithms, which is quite similar to idea of TX/Rx diversity in communication theory.
In Rx diversity(e.g. MRC), we use the combination of weighted 'same copies of signal' from different fading channels to get diversity gain.(There same signal experienced uncorrelated fading might compensate each other)

- But, we usually don't really want to learn lots of models by different algorithms
- However, we want the 'diversity'/way of aggregation to enhance the performance of our classifying

Bootstrap, Identical Idea as Transmit/Receive Diversity

What could we do?

- Recall diversity in communication theory
- Besides Rx diversity, we also have Tx diversity which equips multi-antennas at Tx side with transmitting coded signals(e.g. STBC codes) of original signal. Why we do this?
- Cause we actually try to create 'man made' uncorrelated multiple fading channels. The 'diversity' could be created cause 'we somehow operate on transmitting signal(coded signals) and transmit over normal(correlated) multiple fading channels' is equivalent to 'we do nothing on signal but transmit/receive over uncorrelated multiple fading channels'. The operation on signal creates the 'non-correlation' on multiple fading channels

Bootstrap, Identical Idea as Transmit/Receive Diversity

- If you are familiar with OFDM with CP to combat with ISI, it is the similar idea as well
- 'The operation on OFDM symbol, i.e. adding CP, then transmit over fading channel', is equivalent to 'transmit normal OFDM symbol without CP over circular matrix'. By adding CP, we actually lead to same result but migrate the operation from fading channel matrix to transmit signal. (We will see this in details in 5th book circle together with SVD and PCA)
- The same for Tx diversity, we can't control the how correlated multi-fading channels is but we migrate the operation to transmit signal to get the same performance, i.e. diversity

Bootstrap

- Now go back to question we asked, we don't want to learn lots of models by different algorithms but want to remain the performance from aggregation, how to do it?
- Yes, similar with we do in signal processing for wireless communication! We migrate from 'operation on different models, i.e. combination of models from different algorithms' to 'operation on training sample data set', how?
- We use Bootstrap method to organize several new training sample data sets with diversity, instead of aggregation of models from different algorithms

Bootstrap

Bootstrap

- Suppose we have L training data samples **x**. We want to regenerate T different training data sets and each of them has N training data samples **x** with $N \leq L$
- For each new different training data set t = 1, 2, ... T, we sample a data sample **x** from L samples randomly and repeat this operation N times. Then we all get N samples for each t. Note the some of N samples are same but each N samples for t^{th} set is different

- By bootstrap, we introduce 'diversity' on T training data sets, from only one training data set(origin data set)
- We avoid using models from different algorithms but use only one model from a selected algorithm, however remain the enhancement of performance by using bootstrap on data set

Random Forest

Outline for Section 3

k Nearest Neighbor

How to decide a new point belongs to which? +1 or -1? Based on observed training data samples.
We answered this question several times by using logistic regression, SVM, decision tree and random forest, etc. Now let us check a very simple algorithm k nearest neighbor(kNN)

k Nearest Neighbor

- For new data $\mathbf{x}_{predication}$, we find k points which are with closest distance to x, indexed as x_1, x_2,...,x_k
- We check what value most of the k points belong to, return the major color as the value y of new data point $\mathbf{x}_{predication}$

k Nearest Neighbor

How to define 'closest distance'?

- Euclidean distance.

$$\sum_{d=1}^{D}\sqrt{x^2 - x_i^2}$$

, d stands for number of features

- Absolute distance.

$$\sum_{d=1}^{D}|x - x_i|$$

, d stands for number of features

and many more

k Nearest Neighbor

- kNN really naive algorithm, NOT recommend
- It could be designed for regression. return y value which is the weighted average of y value all other chosen k points

Outline for Section 4

Conclusion

- We finish all algorithms for classification problem in Python Machine Learning book
- From next circle, we will step into practical methods of using Pandas/Scikit Learn in Python to preprocess data

References

 Aarti Singh and Barnabas Poczos, Machine Learning Course, Carnegie Mellon University

 Tom M. Mitchell, Machine Learning Course, Carnegie Mellon University

 Nando de Freitas, Machine Learning Course, The University of British Columbia

 Hsuan-Tien Lin, Machine Learning Technique Course, National Taiwan University

 Bastian Leibe, Machine Learning Course, RWTH Aachen

 Sebastian Raschka, Python Machine Learning

 Zhihua Zhou, Machine Learning(Chinese Version)

Question?