Decision trees

Subhransu Maji

CMPSCI 689: Machine Learning

22 January 2015

Overview

- What does it mean to learn?
- Machine learning framework
- Decision tree model
  - a greedy learning algorithm
- Formalizing the learning problem
- Inductive bias
- Underfitting and overfitting
- Model, parameters, and hyperparameters

What does it mean to learn?

- Alice has just begun taking a machine learning course
- Bob, the instructor has to ascertain if Alice has “learned” the topics covered, at the end of the course
- A common way of doing this to give her an “exam”
- What is a reasonable exam?
Alice has just begun taking a machine learning course
Bob, the instructor has to ascertain if Alice has “learned” the topics covered, at the end of the course
A common way of doing this to give her an “exam”
What is a reasonable exam?
  › Choice 1: History of pottery
    - Alice’s performance is not indicative of what she learned in ML
  › Choice 2: Questions answered during lectures
    - Bad choice, especially if it is an open book

A good test should test her ability to answer “related” but “new” questions on the exam
This tests whether Alice has an ability to generalize
  - Generalization is one of the central concepts in ML
Student ratings of undergrad CS courses
Collection of students and courses
The evaluation is a score -2 (terrible), +2 (awesome)
The job is to say if a particular student (say, Alice) will like a particular course (say, Algorithms)
We are given historical data, i.e., course ratings in the past, we are trying to predict unseen ratings (i.e., the future)

We can ask if:
- Will Alice will like History of pottery?
  - Unfair, because the system doesn’t even know what that is

We can ask if:
- Will Alice like AI?
  - Easy if Alice took AI last year and said it was +2 (awesome)
What does it mean to learn?

- Student ratings of undergrad CS courses
- Collection of students and courses
- The evaluation is a score -2 (terrible), +2 (awesome)
- The job is to say if a particular student (say, Alice) will like a particular course (say, Algorithms)
- We are given historical data, i.e., course ratings in the past, we are trying to predict unseen ratings (i.e., the future)

We can ask if:
- Will Alice will like History of pottery? \(\text{too much generalization}\)
  - Unfair, because the system doesn’t even know what that is
- Will Alice like AI? \(\text{too little generalization}\)
  - Easy if Alice took AI last year and said it was +2 (awesome)

Machine learning framework

- Training data:
  - Alice in ML course: concepts that she encounters in the class
  - Recommender systems: past course ratings

- Learning algorithm induces a function \(f\) that maps examples to labels

- The set of new examples is called the “test” set
  - Closely guarded secret: it is the final exam where the learner is going to be tested
  - A ML algorithm has succeeded if its performance on the test data is good

- We will focus on a simple model of learning called a decision tree

The decision tree model of learning

- Classic and natural model of learning
The decision tree model of learning

- Classic and natural model of learning

- **Question:** Will an unknown user enjoy an unknown course?
  - **You:** Is the course under consideration in Systems?
  - **Me:** Yes
  - **You:** Has this student taken any other Systems courses?
  - **Me:** Yes
  - **You:** Has this student liked most previous Systems courses?
  - **Me:** No
  - **You:** I predict this student will not like this course.

The decision tree model of learning

- Classic and natural model of learning

- **Question:** Will an unknown user enjoy an unknown course?
  - **You:** Is the course under consideration in Systems?
  - **Me:** Yes
  - **You:** Has this student taken any other Systems courses?
  - **Me:** Yes
  - **You:** Has this student liked most previous Systems courses?
  - **Me:** No
  - **You:** I predict this student will not like this course.

Goal of learner: Figure out what questions to ask, and in what order, and what to predict when you have answered enough questions.

Learning a decision tree

- Recall that one of the ingredients of learning is training data
  - I’ll give you $(x, y)$ pairs, i.e., set of (attributes, label) pairs
  - We will simplify the problem by
    - $\{0,+1, +2\}$ as “liked”
    - $\{-1,-2\}$ as “hated”
- Here:
  - Questions are features
  - Responses are feature values
  - Rating is the label

- Lots of possible trees to build
- Can we find good one quickly?

Greedy decision tree learning

- If I could ask one question, what question would I ask?
  - You want a feature that is most useful in predicting the rating of the course
  - A useful way of thinking about this is to look at the histogram of the labels for each feature

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>
Greedy decision tree learning

- If I could ask one question, what question would I ask?
  - You want a feature that is most useful in predicting the rating of the course
  - A useful way of thinking about this is to look at the histogram of the labels for each feature

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>

What attribute is useful?

Attribute = Easy?

Overall

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\# correct = 6

What attribute is useful?

Attribute = Easy?

Easy? n

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\# correct = 6

What attribute is useful?

Attribute = Easy?

Easy? y

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\# correct = 6
What attribute is useful?

Attribute = Easy?

# correct = 12

What attribute is useful?

Attribute = Sys?

# correct = 10

What attribute is useful?

Attribute = Sys?

# correct = 8
What attribute is useful?

Attribute = Sys?

# correct = 18

Picking the best attribute

Decision tree train

Algorithm 1 DecisionTreeTrain(data, remaining features)
1. guess ← most frequent answer in data // default answer for this data
2. if the labels in data are unambiguous
   return Leaf(guess) // base case: no need to split further
3. else if remaining features is empty
   return Leaf(guess) // base case: cannot split further
4. else
5. for all f ∈ remaining features do
6.   NO ← the subset of data on which f=no
7.   YES ← the subset of data on which f=yes
8.   score[f] ← # of majority vote answers in NO + # of majority vote answers in YES
9.   // the accuracy we would get if we only queried on f
10.   end for
11. f ← the feature with maximal score(f)
12. NO ← the subset of data on which f=no
13. YES ← the subset of data on which f=yes
14. left ← DecisionTreeTrain(NO, remaining features \ {f})
15. right ← DecisionTreeTrain(YES, remaining features \ {f})
16. return Node(f, left, right)
17. end if
18. end if

Decision tree test

Algorithm 2 DecisionTreeTest(tree, test point)
1. if tree is of the form Leaf(guess) then
2.   return guess
3. else if tree is of the form Node(f, left, right)
4.   if f = yes in test point then
5.     return DecisionTreeTest(left, test point)
6.   else
7.     return DecisionTreeTest(right, test point)
8.   end if
9. end if
Loss function: $\ell(y, \hat{y})$

- The way we measure performance of the classifier
  - Examples:
    - Regression: squared loss: $\ell(y, \hat{y}) = (y - \hat{y})^2$
    - or, absolute loss: $\ell(y, \hat{y}) = |y - \hat{y}|
    - Binary classification: zero-one loss
      $$\ell(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$
    - Multiclass classification: also, zero-one loss

Data generating distribution: $D(x, y)$

- $D(x, y)$: probability distribution from which the data comes from
  - Assigns high probability to reasonable $(x, y)$ pairs
  - Assigns low probability to unreasonable $(x, y)$ pairs
  - Examples:
    - Reasonable $x$: “Intro to Python”
    - Unreasonable $x$: “Intro to Quantum Pottery”
    - Unreasonable $(x, y)$: (AI, unlike)

We don’t know what $D$ is!

All we have is access to training samples drawn from $D$

$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

Training error

$$\hat{e} \triangleq \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n))$$

Expected loss over the distribution $D(x, y)$

$$e \triangleq \mathbb{E}_{(x,y) \sim D}[\ell(y, f(x))] = \sum_{(x,y)} D(x, y) \ell(y, f(x))$$
What do we know before we see the data?

Partition these into two groups

What is the inductive bias of the decision tree algorithm?
Underfitting and overfitting

**Decision trees:**
- **Underfitting**: an empty decision tree
  - Test error: ?
- **Overfitting**: a full decision tree
  - Test error: ?

Model, parameters and hyperparameters

- **Model**: decision tree
- **Parameters**: learned by the algorithm
- **Hyperparameter**: depth of the tree to consider
  - A typical way of setting this is to use validation data
  - Usually set 2/3 training and 1/3 testing
    - Split the training into 1/2 training and 1/2 validation
    - Estimate optimal hyperparameters on the validation data

Summary

- **Generalization is key**
- **Inductive bias is needed to generalize beyond training examples**
- **Decision tree model**
  - a greedy learning algorithm
  - Inductive bias of the learner
  - Underfitting and overfitting
  - Model, parameters, and hyperparameters
Many slides are adapted from the book “Course in Machine Learning” by Hal Daume