1 Chapter 18 Learning from Examples

1.1 Introduction

Why would we want an agent to learn? If we can improve the design of an agent, why don't we program in that improvement to begin with?

- The designer cannot anticipate all possible situations that the agent may be in.

  To build an agent to navigate mazes, we cannot program in all possible mazes.

- The designer cannot anticipate all changes over time.

  The stock market goes up and down. A static program will not do well when predicting the stock market price tomorrow.

- Sometimes, we have no idea how to program a solution.

  How do we recognize faces? Why are we good at solving certain problems? We don't even know why we have certain intuition into some problems.

We will focus on one class of learning problems which have vast applicability: from a collection of input–output pairs, learn a function that predicts the output for new inputs.

Types of learning problems based on feedback available

- Unsupervised learning:

  no explicit feedback is given. (There is no output or the output is the same for every input.)

  want to learn pattern in the input.

  The most common task is "clustering" – detecting clusters of input examples.

- Supervised learning:

  given input output pairs

  learn a function which maps from input to output

- A continuum between supervised and unsupervised learning because of noise and lack of labels.

1.2 Supervised Learning

The problem: Given a training set of N input-output pairs, $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, where each y_j was generated by an unknown function $y = f(x)$, discover a function h that approximates the true function f.

x and y can be any values and need not be numbers. x can be a vector of values. y can be one value in a discrete set of values or a value in a continuous interval of values.

h is a hypothesis.

We search through the hypothesis space for a function that performs well even on new examples beyond the training set.

We measure the accuracy of the hypothesis by testing it on a test set that is distinct from the training set.

A hypothesis generalizes well if it correctly predicts the values of y for novel examples.

Classification: if the output is one of a finite set of values

Regression: if the output is a continuous value.

Given a hypothesis space, there could be multiple hypotheses that agree with all of the data points. How do we choose among these hypotheses?

We prefer the simplest hypothesis that agrees with all the data (or makes small errors.)

- Generalize better.
- Easy to compute.

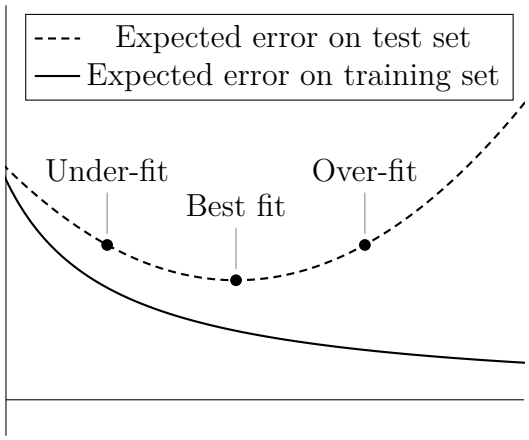
Ockham's razor: - What does it mean for a hypothesis to be the simplest? We will discuss this more later.

There is a trade-off between complex hypotheses that fit the data well and simpler hypotheses that may generalize better.

Over-fitting and Under-fitting

Learning curve: plot the average error of the hypothesis with respect to the complexity of the hypothesis

Measure the complexity of the hypothesis by the number of parameters and the size of the hypothesis.



All the hypothesis is trying to do is to capture patterns in the data.

- A simple hypothesis does not have a lot of degrees of freedom. A complex hypothesis can capture more pattern because it has a lot more degrees of freedom.

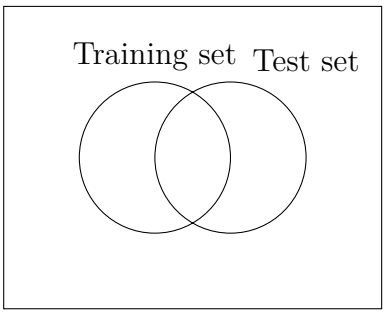
  A line: slope and y-intercept. A 4th degree polynomial: 5 parameters.

- In the under-fit region:

  The hypothesis has not captured all of the patterns that are common in both the training and the test set.

- In the over-fit region:

  The hypothesis has captured patterns that are only present in the training set and not present in the test set.



1.3 Decision Trees

1.3.1 Introducing a decision tree

One of the simplest yet most successful forms of machine learning

Take as input a vector of feature values and return a single output value.

For now: inputs have discrete values and the output has two possible values — a Binary classification

Each example input will be classified as true (positive example) or false (negative example).

We will use the following example to illustrate the decision tree learning algorithm.

Example: Jeeves the valet

Jeeves is a valet to Bertie Wooster. On some days, Bertie likes to play tennis and asks Jeeves to lay out his tennis things and book the court. Jeeves would like to predict whether Bertie will play tennis (and so be a better valet). Each morning over the last two weeks, Jeeves has recorded whether Bertie played tennis on that day and various attributes of the weather.

Jeeves would like to evaluate the classifier he has come up with for predicting whether Bertie will play tennis. Each morning over the next two weeks, Jeeves records the following data.

Jeeves the valet – the training set

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Jeeves the valet – the test set

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Mild	High	Strong	No
2	Rain	Hot	Normal	Strong	No
3	Rain	Cool	High	Strong	No
4	Overcast	Hot	High	Strong	Yes
5	Overcast	Cool	Normal	Weak	Yes
6	Rain	Hot	High	Weak	Yes
7	Overcast	Mild	Normal	Weak	Yes
8	Overcast	Cool	High	Weak	Yes
9	Rain	Cool	High	Weak	Yes
10	Rain	Mild	Normal	Strong	No
11	Overcast	Mild	High	Weak	Yes
12	Sunny	Mild	Normal	Weak	Yes
13	Sunny	Cool	High	Strong	No
14	Sunny	Cool	High	Weak	No

A decision tree performs a sequence of tests in the input features.

- Each node performs a test on one input feature.
- Each arc is labeled with a value of the feature.
- Each leaf node specifies an output value.

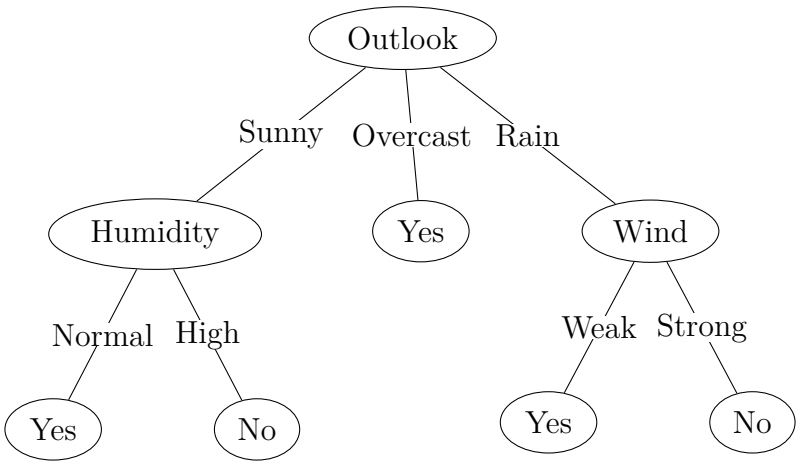
Using the Jeeves training set, we will construct two decision trees using different orders of testing the features.

Example 1: Let's construct a decision tree using the following order of testing features.

Test Outlook first.

For Outlook=Sunny, test Humidity. (After testing Outlook, we could test any of the three remaining features: Humidity, Wind, and Temp. We chose Humidity here.)

For Outlook=Rain, test Wind. (After testing Outlook, we could test any of the three remaining features: Humidity, Wind, and Temp. We chose Wind here.)



Example 2: Let's construct another decision tree by choosing Temp as the root node. This choice will result in a really complicated tree shown on the next page.

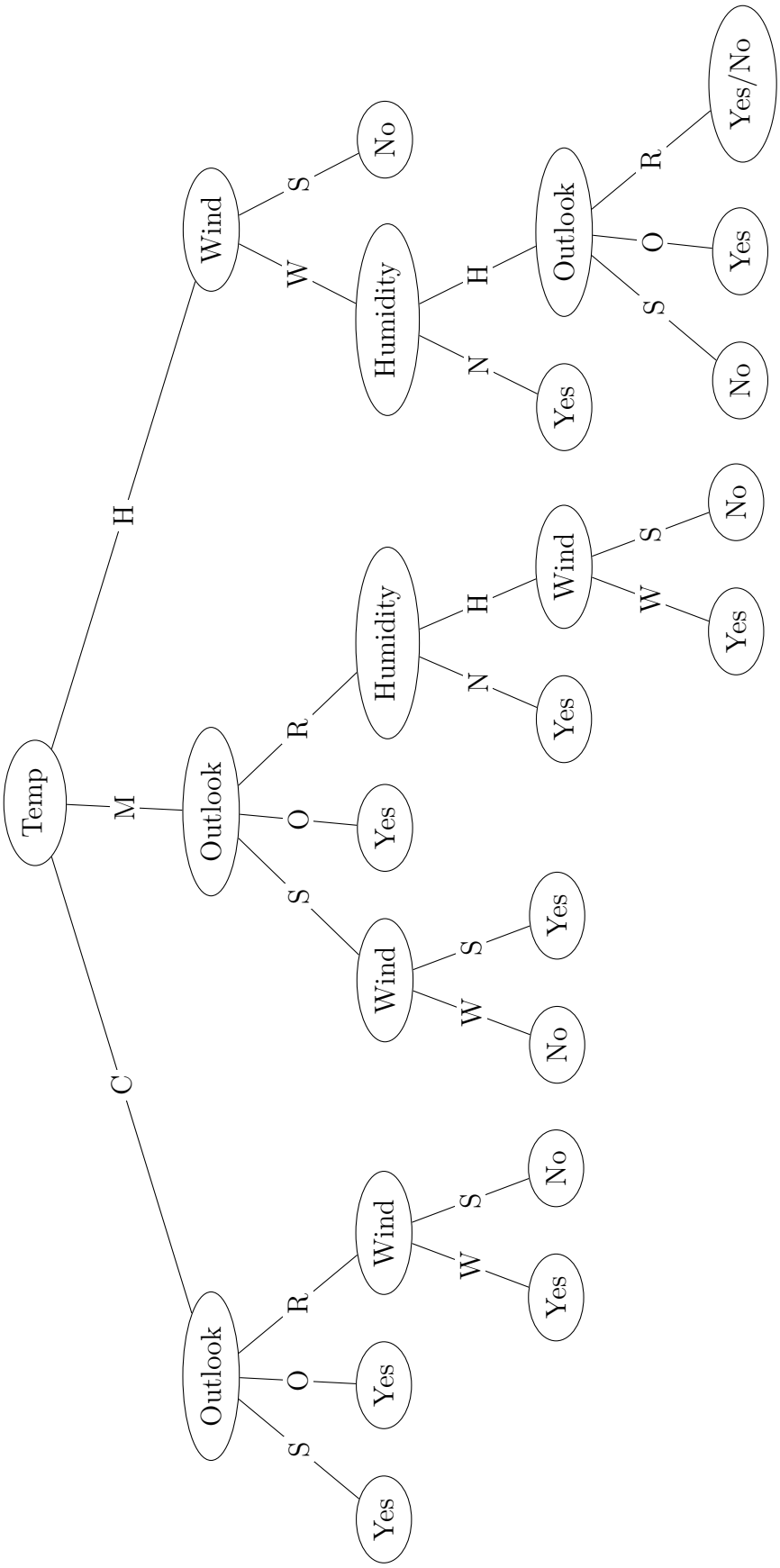
We have constructed two decision trees and both trees can classify the training examples perfectly. Which tree would you prefer?

One way to choose between the two is to evaluate them on the test set.

The first (and simpler) tree classifies 14/14 test examples correctly. Here are the decisions given by the first tree on the test examples. (1. No. 2. No. 3. No. 4. Yes. 5. Yes. 6. Yes. 7. Yes. 8. Yes. 9. Yes. 10. No. 11. Yes. 12. Yes. 13. No. 14. No.)

The second tree classifies 7/14 test examples correctly. Here are the decisions given by the second tree on the test examples. (1. Yes. 2. No. 3. No. 4. No. 5. Yes. 6. Yes/No. 7. Yes. 8. Yes. 9. Yes. 10. Yes. 11. Yes. 12. No. 13. Yes. 14. Yes.)

The second and more complicated tree performs worse on the test examples than the first tree, possibly because the second tree is overfitting to the training examples.


Temp
H
Wind
W
Humidity
H
Outlook
R
Yes/No
O
Yes
S
No
N
Yes
S
No
M
Outlook
R
Humidity
H
Wind
W
Yes
S
No
N
Yes
O
Yes
S
Wind
W
No
S
Yes
C
Outlook
R
Wind
W
Yes
S
No
O
Yes
S
Yes

Every decision tree corresponds to a propositional formula.

For example, our simpler decision tree corresponds to the propositional formula.

$(Outlook = Sunny \wedge Humidity = Normal) \vee (Outlook = Overcast) \vee (Outlook = Rain \wedge Wind = Weak)$

If we have n features, how many different functions can we encode with decisions trees? (Let's assume that every feature is binary.)

Each function corresponds to a truth table. Each truth table has 2^n rows. There are 2^{2^n} possible truth tables.

With $n = 10$, $2^{1024} \approx 10^{308}$

How do we find a good hypothesis in such a large space?