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?

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?

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?
  - Choice 1: History of pottery
    - ➡ Alice’s performance is not indicative of what she learned in ML

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?
  - 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

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?
  - 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 weather Alice has an ability to generalize
  - Generalization is a one of the central concepts in ML

What does it mean to learn?

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)

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**?
    - Unfair, because the system doesn't even know what that is

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**? ← too much generalization
    - ➡ Unfair, because the system doesn't even know what that is

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**? ← too much generalization
    - Unfair, because the system doesn't even know what that is
  - Will Alice like **AI**?
    - Easy if Alice took AI last year and said it was +2 (awesome)

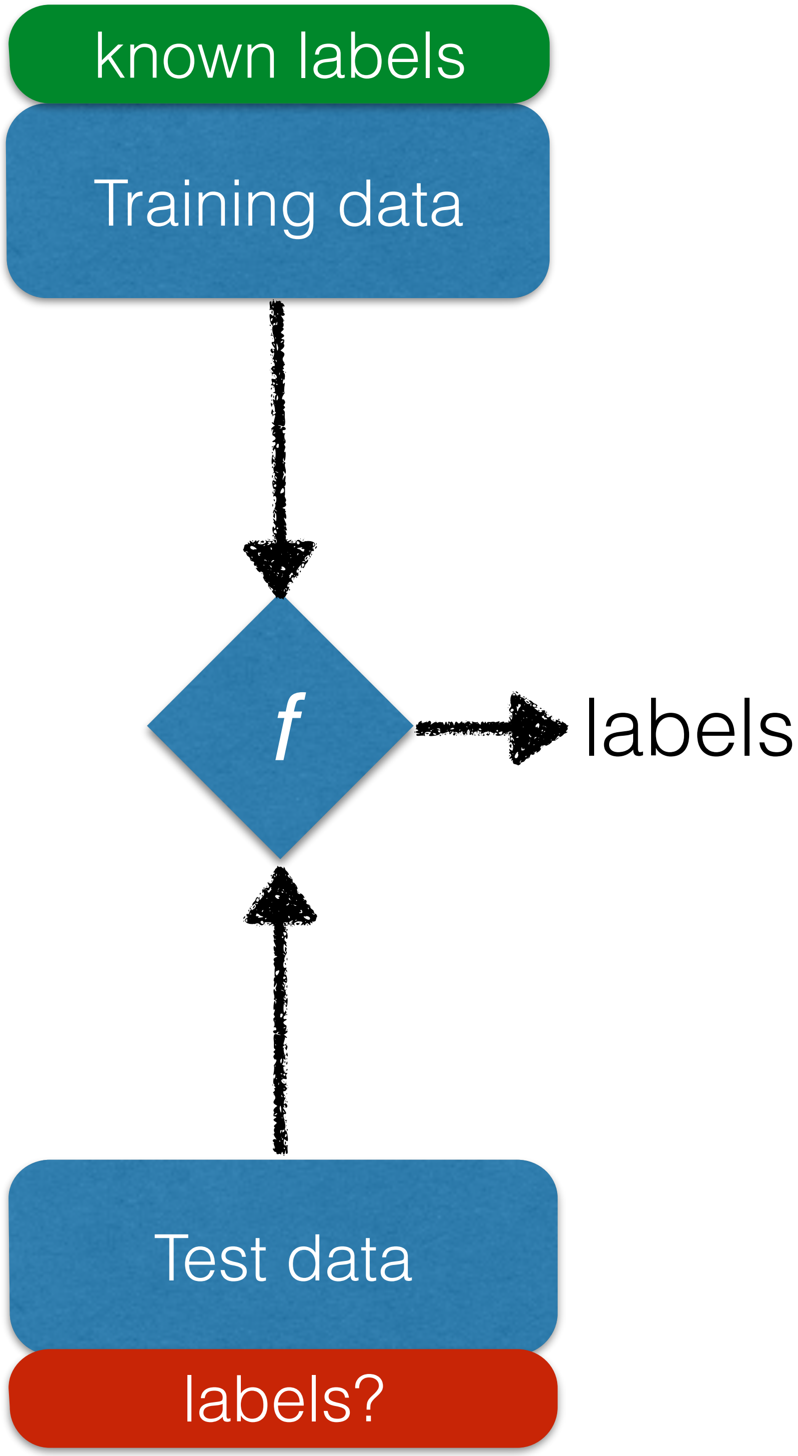
What does it mean to learn?

- 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**? ← too much generalization
    - Unfair, because the system doesn't even know what that is
  - Will Alice like **AI**? ← 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

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?

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Course ratings dataset

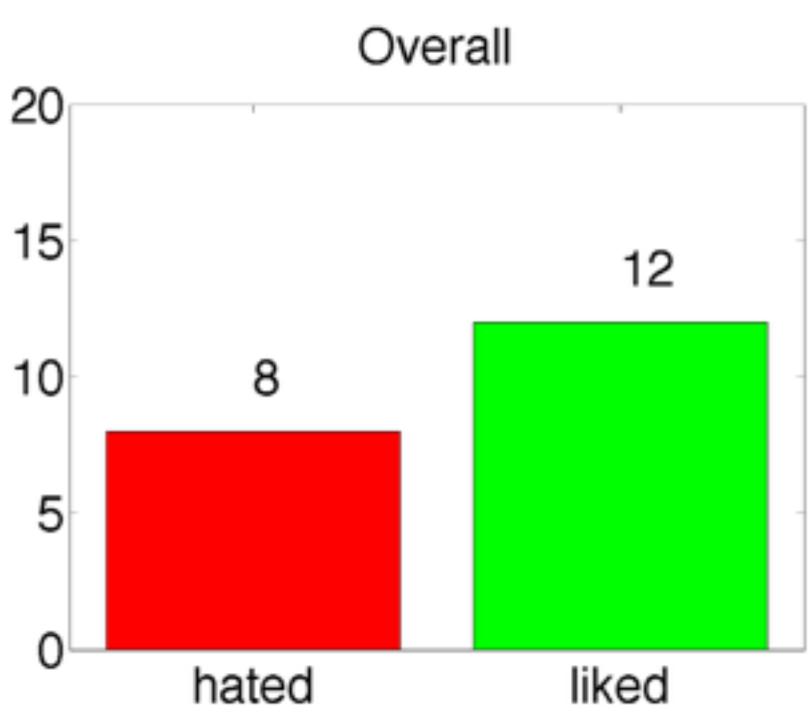
Greedy decision tree learning

- 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

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

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



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

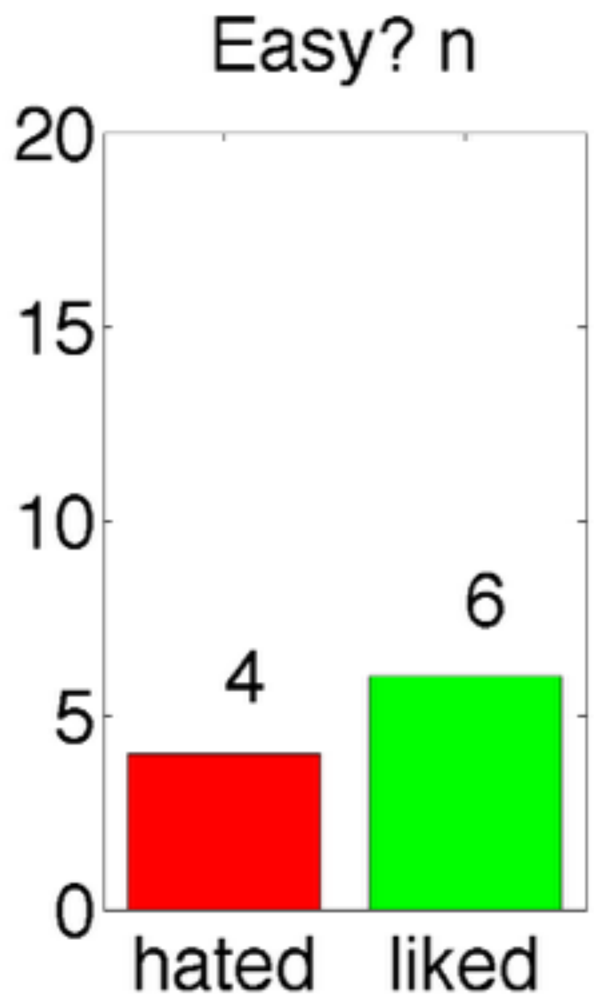
What attribute is useful?

Attribute = Easy?

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y				
+2	y				
+2	n				
+2	n				
+2	n				
+1	y				
+1	y				
+1	n				
0	n				
0	y				
0	n				
0	y				
-1	y				
-1	n				
-1	n				
-1	y				
-2	n				
-2	n				
-2	y				
-2	y				

What attribute is useful?

Attribute = Easy?

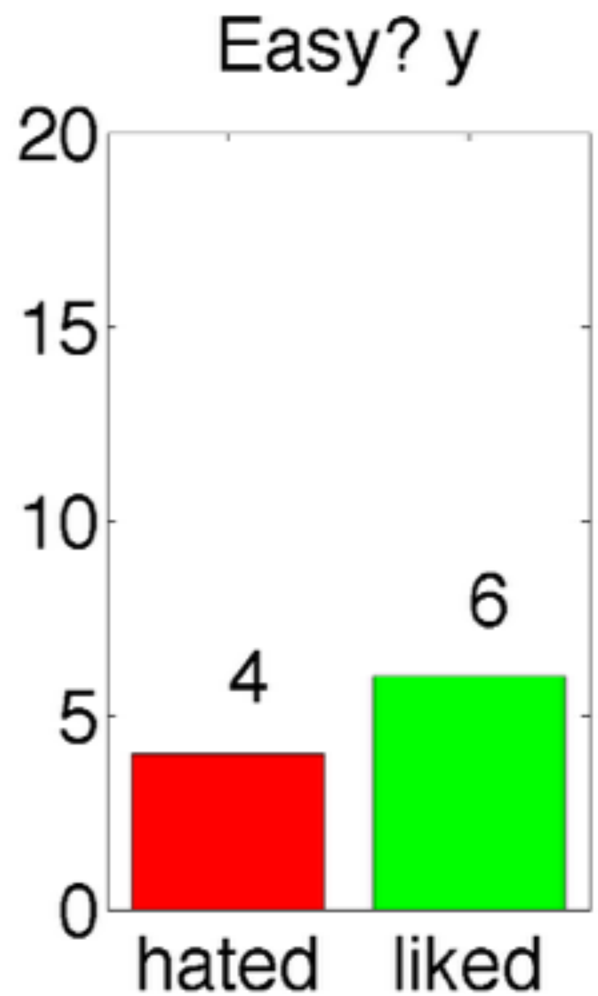


correct = 6

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	n				
+2	n				
+2	n				
+1	n				
0	n				
0	n				
-1	n				
-1	n				
-2	n				
-2	n				

What attribute is useful?

Attribute = Easy?

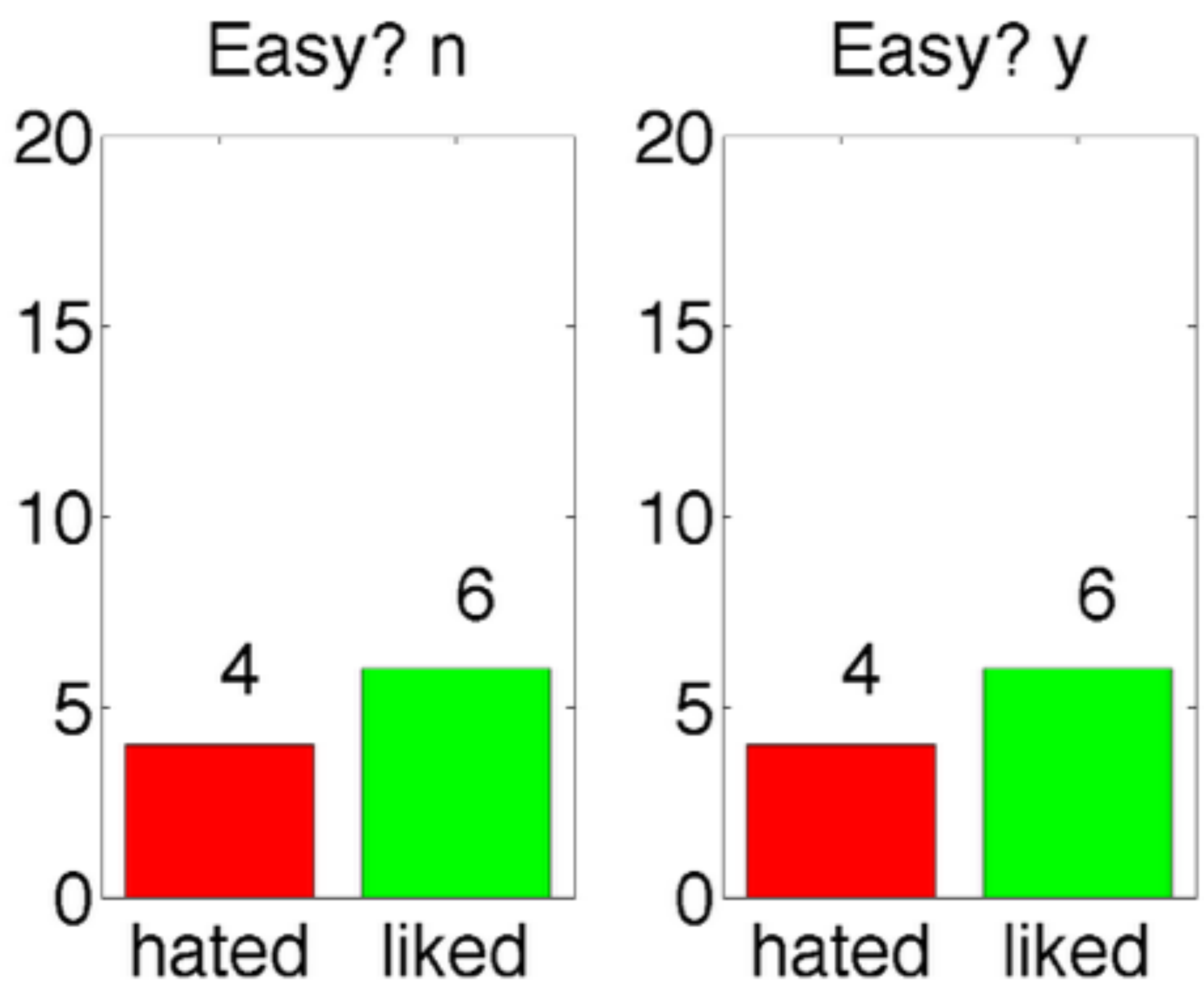


Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y				
+2	y				
+1	y				
+1	y				
0	y				
0	y				
-1	y				
-1	y				
-2	y				
-2	y				

\# correct = 6

What attribute is useful?

Attribute = Easy?



correct = **12**

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y				
+2	y				
+2	n				
+2	n				
+2	n				
+1	y				
+1	y				
+1	n				
0	n				
0	y				
0	n				
0	y				
-1	y				
-1	n				
-1	n				
-1	y				
-2	n				
-2	n				
-2	y				
-2	y				

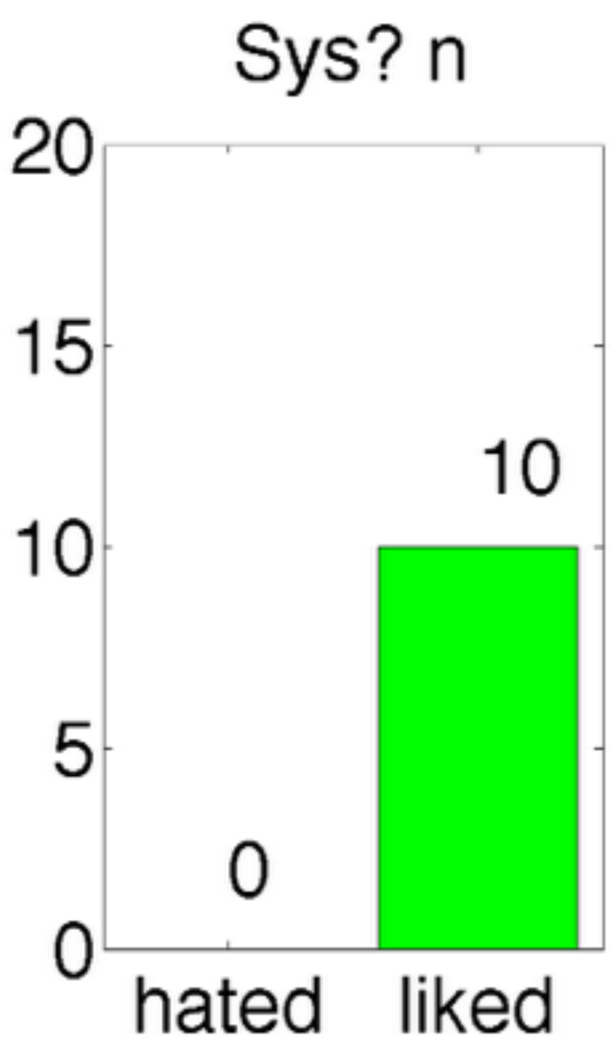
What attribute is useful?

Attribute = Sys?

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			n		
+2			n		
+2			n		
+2			n		
+2			y		
+1			n		
+1			n		
+1			n		
0			n		
0			n		
0			n		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

What attribute is useful?

Attribute = Sys?

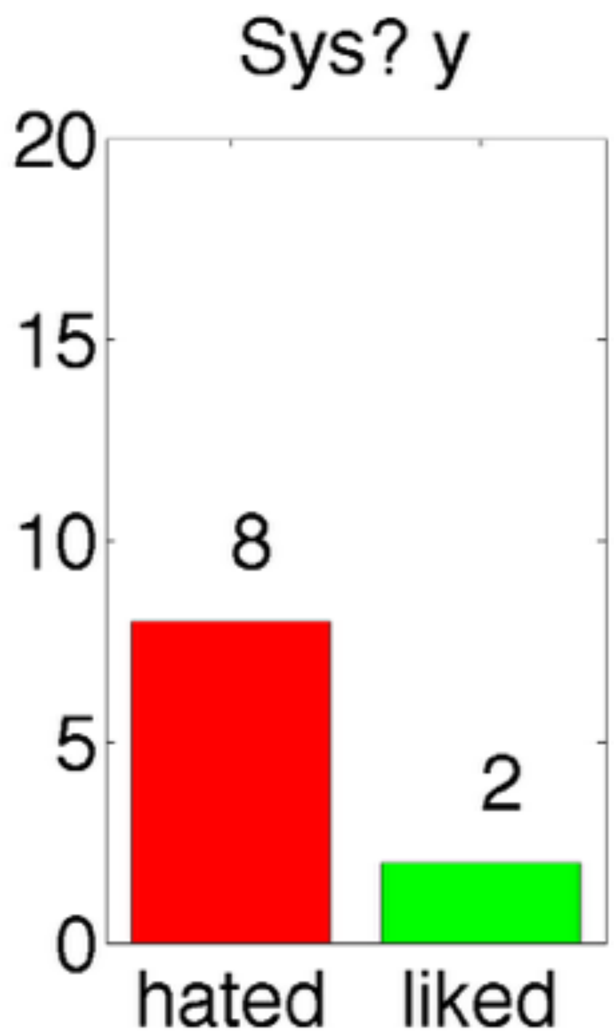


correct = 10

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			n		
+2			n		
+2			n		
+2			n		
+1			n		
+1			n		
+1			n		
0			n		
0			n		
0			n		

What attribute is useful?

Attribute = Sys?

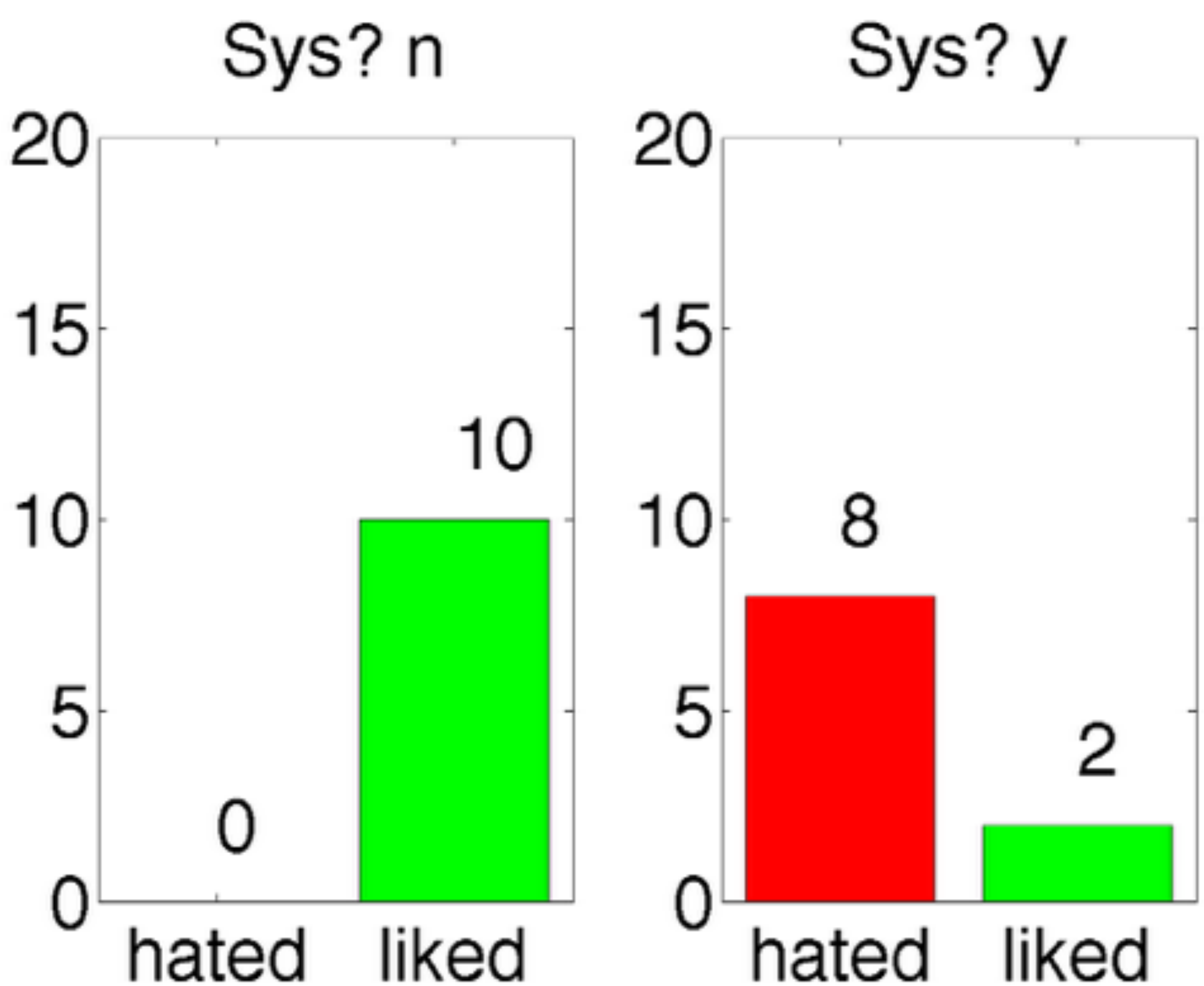


Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			y		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

correct = 8

What attribute is useful?

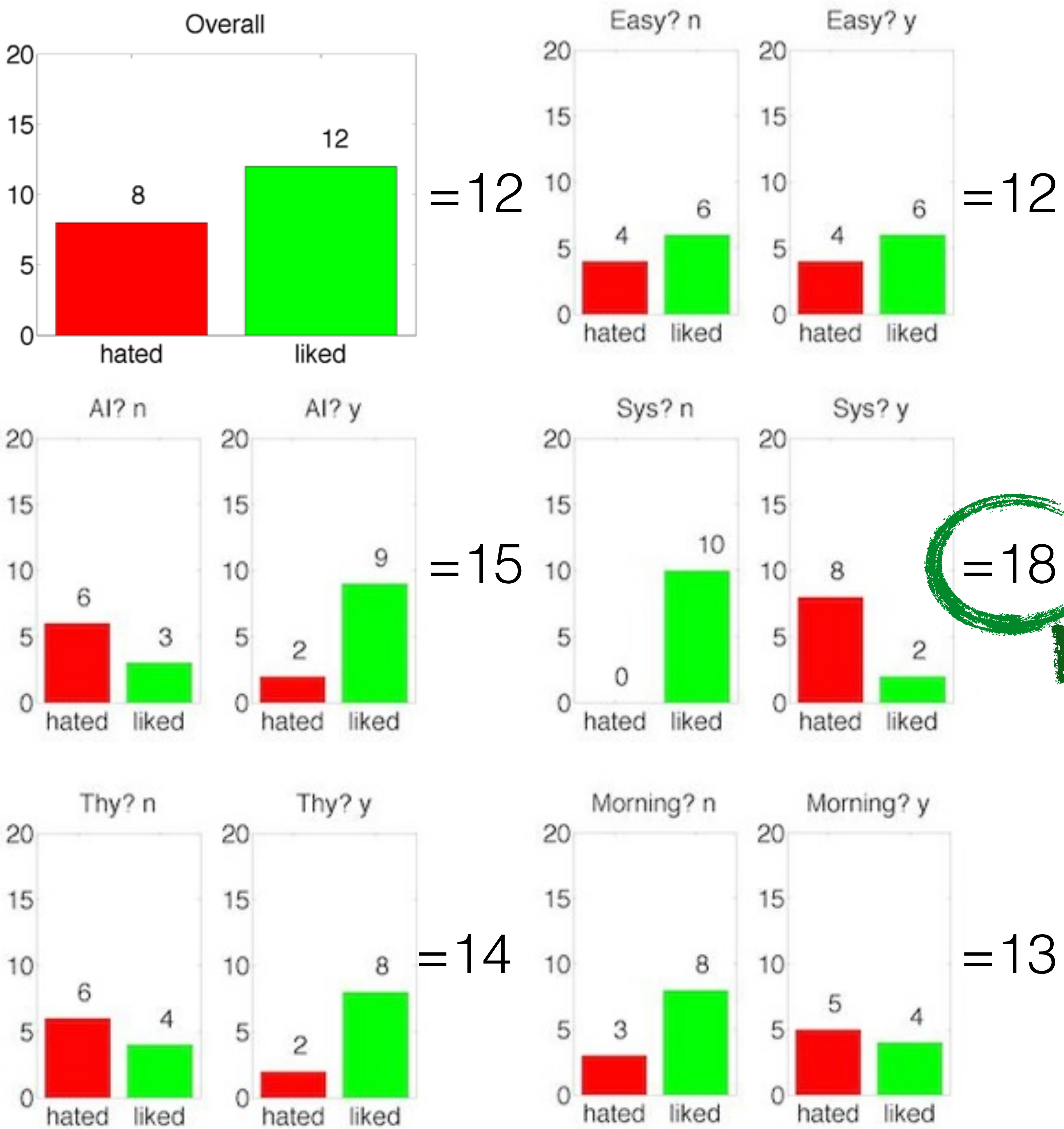
Attribute = Sys?



correct = **18**

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			n		
+2			n		
+2			n		
+2			n		
+2			y		
+1			n		
+1			n		
+1			n		
0			n		
0			n		
0			n		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

Picking the best attribute



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

best attribute

Decision tree train

Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data                          // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                        // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                        // base case: cannot split further
else                                                          // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                    + # of majority vote answers in YES
                                        // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

Decision tree test

Algorithm 2 DecisionTreeTest(*tree*, *test point*)

```
if tree is of the form Leaf(guess) then
    return guess
else if tree is of the form Node(f, left, right) then
    if f = yes in test point then
        return DecisionTreeTest(left, test point)
    else
        return DecisionTreeTest(right, test point)
    end if
end if
```

Formalizing the learning problem

- **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

Formalizing the learning problem

- **Loss function:** $\ell(y, \hat{y})$
- **Data generating distribution:** $\mathcal{D}(\mathbf{x}, y)$
  - $\mathcal{D}(\mathbf{x}, y)$**:** probability distribution from which the data comes from
    - Assigns *high probability* to reasonable $(\mathbf{x}, y)$ pairs
    - Assigns low probability to unreasonable $(\mathbf{x}, y)$ pairs
    - Examples:
      - Reasonable $\mathbf{x}$: “Intro to Python”
      - Unreasonable $\mathbf{x}$: “Intro to Quantum Pottery”
      - Unreasonable $(\mathbf{x}, y)$: (AI,unlike)

Formalizing the learning problem

- **Loss function:** $\ell(y, \hat{y})$
- **Data generating distribution:** $\mathcal{D}(\mathbf{x}, y)$
  - $\mathcal{D}(\mathbf{x}, y)$**:** probability distribution from which the data comes from
    - Assigns *high probability* to reasonable $(\mathbf{x}, y)$ pairs
    - Assigns low probability to unreasonable $(\mathbf{x}, y)$ pairs
    - Examples:
      - Reasonable $\mathbf{x}$: "Intro to Python"
      - Unreasonable $\mathbf{x}$: "Intro to Quantum Pottery"
      - Unreasonable $(\mathbf{x}, y)$: (AI,unlike)

- We don't know what $\mathcal{D}$ is!
- All we have is access to training samples drawn from $\mathcal{D}$

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

Formalizing the learning problem

- **Loss function:** $\ell(y, \hat{y})$
- **Training samples:** $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$ drawn from an unknown distribution $\mathcal{D}$

- **Learning problem:** Compute a function f that minimizes the expected loss ϵ over the distribution $\mathcal{D}(\mathbf{x}, y)$

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\ell(y, f(\boldsymbol{x}))\right] = \sum_{(\boldsymbol{x},y)} \mathcal{D}(\boldsymbol{x}, y)\ell(y, f(\boldsymbol{x}))$$

Training error $\hat{\epsilon} \triangleq \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n))$

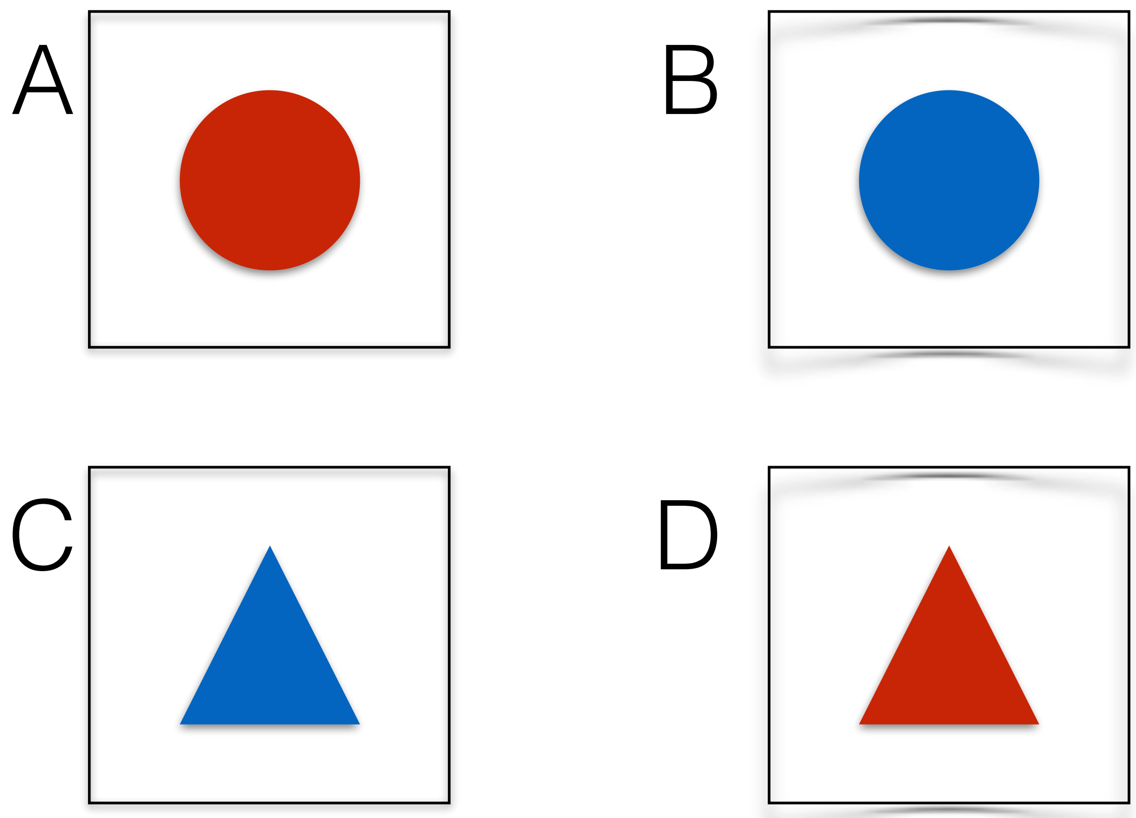
Inductive bias

Inductive bias

- What do we know *before* we see the data?

Inductive bias

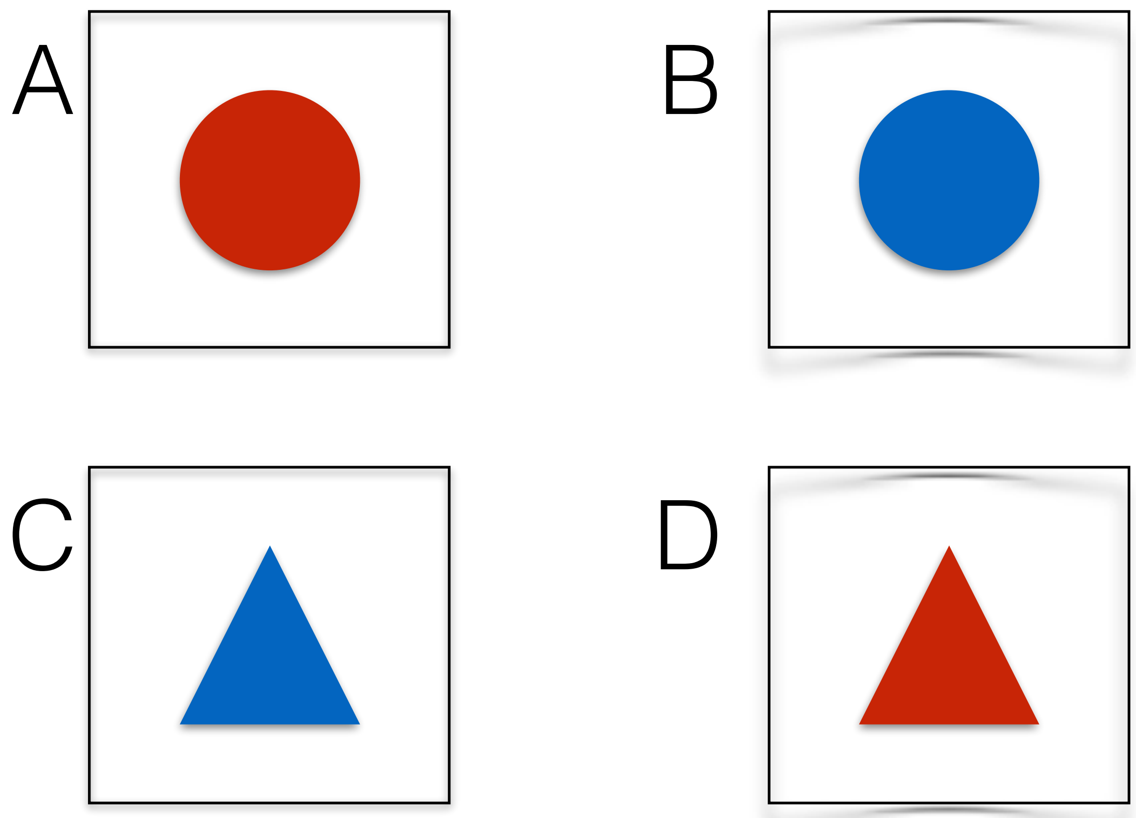
- What do we know *before* we see the data?



Partition these into two groups

Inductive bias

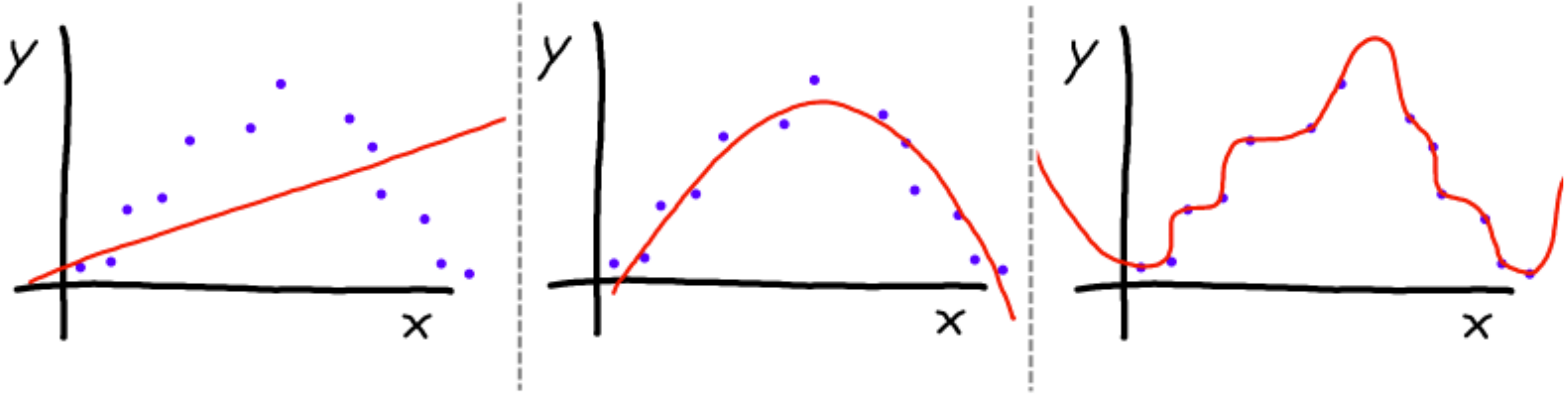
- What do we know *before* we see the data?



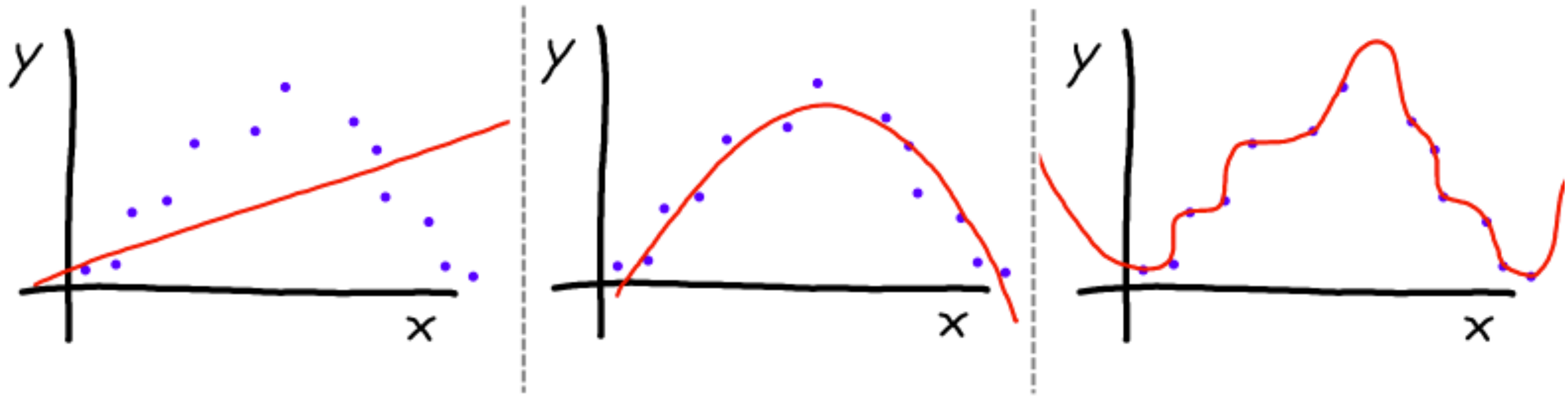
Partition these into two groups

- What is the inductive bias of the decision tree algorithm?

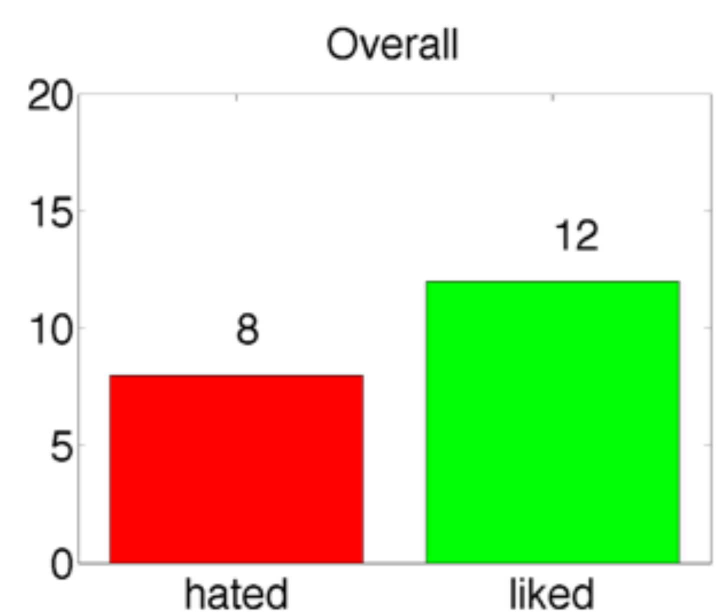
Underfitting and overfitting



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

Slides credit

- Many slides are adapted from the book “Course in Machine Learning” by Hal Daume