Lecture 3: Learning as Search

COMP 343, Spring 2022

Victoria Manfredi

Acknowledgements: These slides are based primarily on content from the book "Machine Learning" by Tom Mitchell, slides created by Vivek Srikumar (U of Utah), Dan Roth (UPenn), and lectures by Balaraman Ravindran (IIT Madras), as well as from courses at Stanford (http://cs229.stanford.edu/) and UC Berkeley (http://ai.berkeley.edu/)

Today's Topics

Homework 1 out

- Due Wed., February 9 by 5p

Using supervised learning

1. What is our instance space?
2. What is our label space?
3. What is our hypothesis space?

Recap

Instances and Labels

X: Instance space

The set of all possible examples to be classified

Y: Label space

The set of all possible labels

E.g., the set of all possible names, documents, sentences, images, emails, …

E.g., {Spam, Not-Spam}, {+, -}, …

X: instance space

$\mathbf{x}$: individual example in X

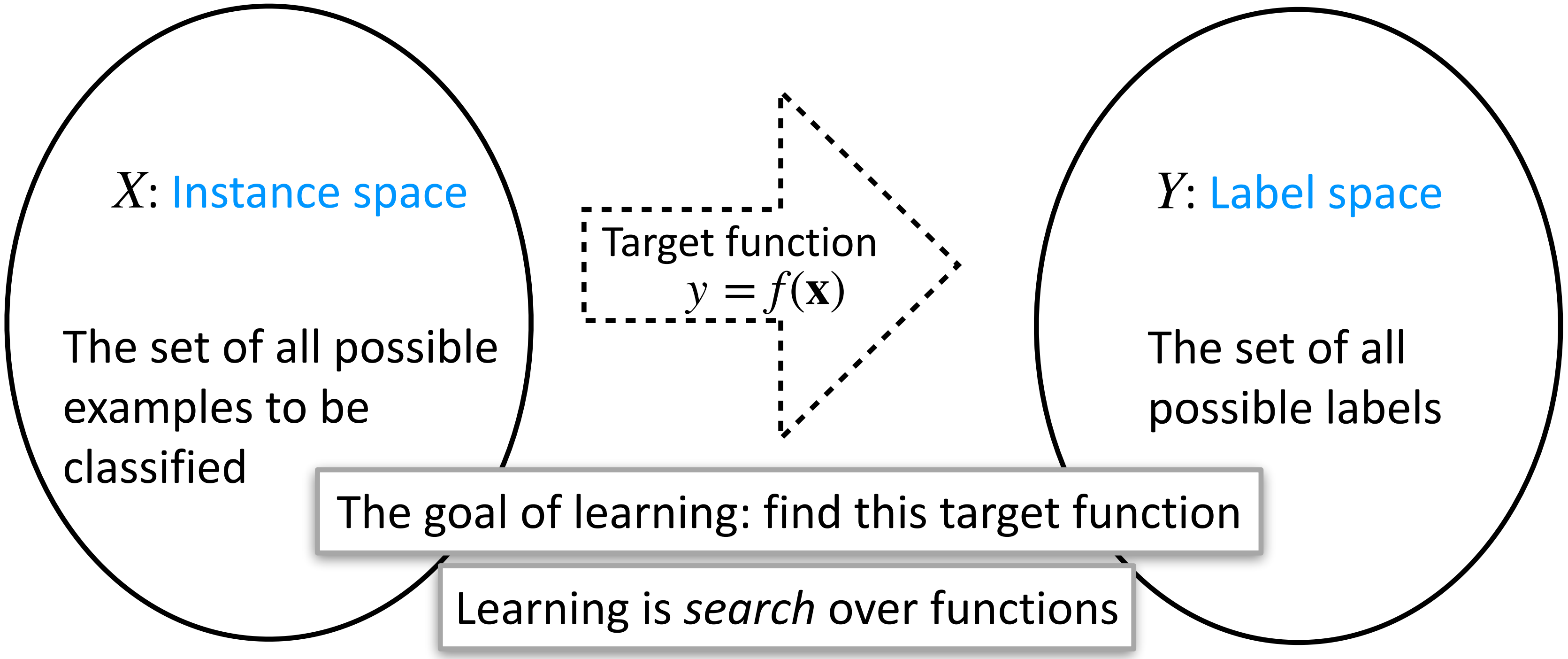
$$\mathbf{x} \in X$$

Y: instance label

y: individual label in Y

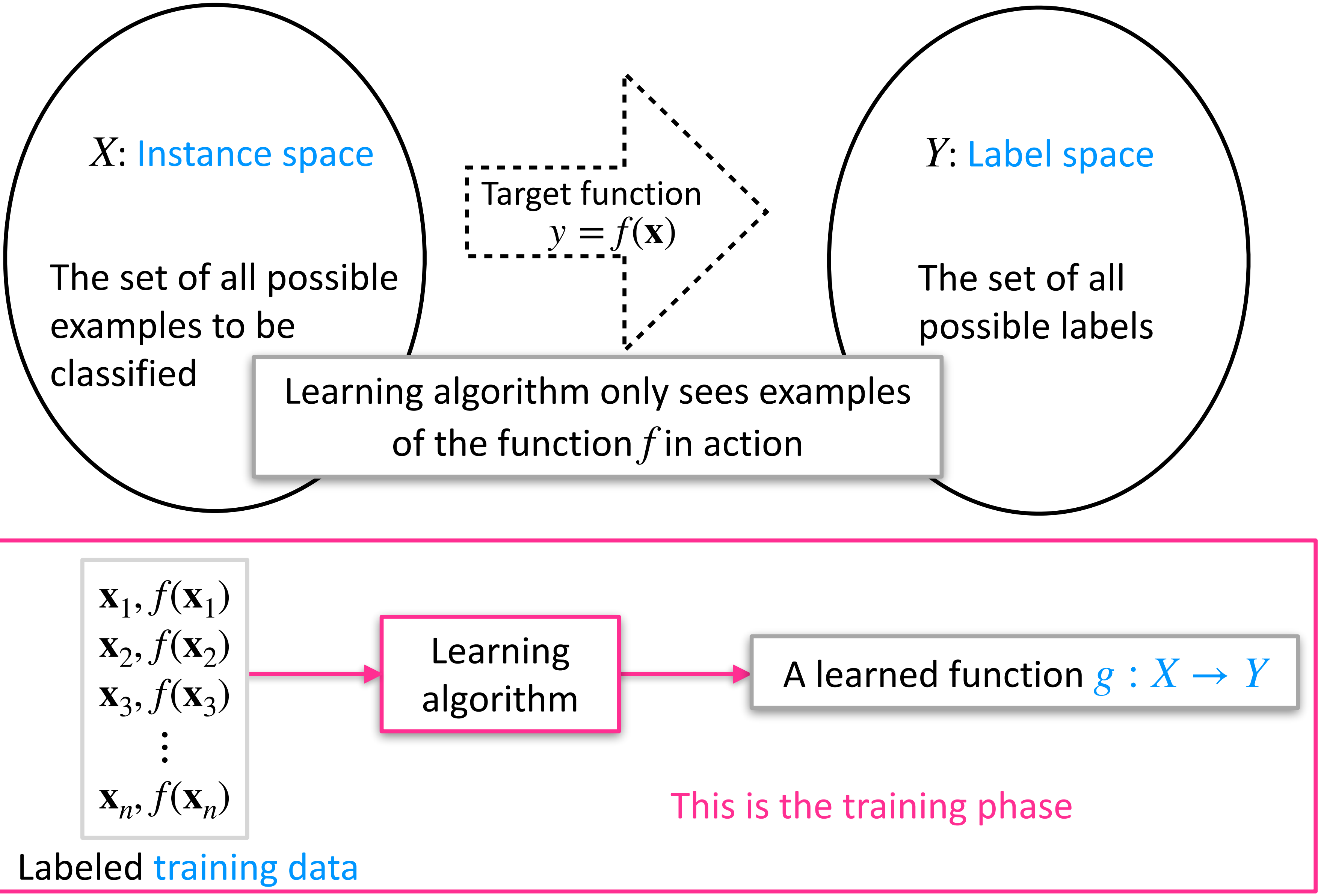
$$y \in Y$$

Target function

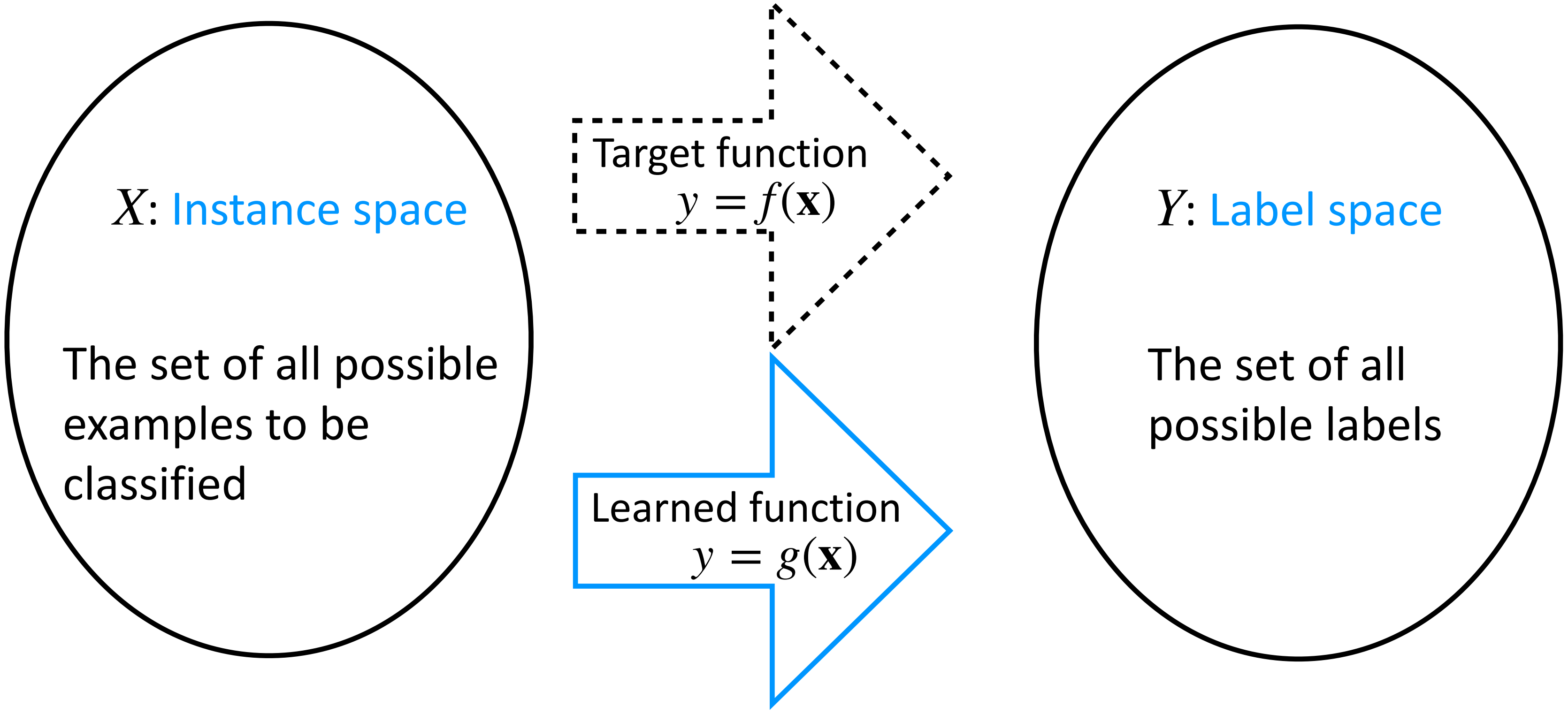

X: Instance space
The set of all possible examples to be classified
Target function $y = f(\mathbf{x})$
Y: Label space
The set of all possible labels
The goal of learning: find this target function
Learning is *search* over functions

We need to search over the set of possible functions that exist to find the one function that maps instances to labels in the way we want

Supervised learning


X: Instance space
The set of all possible examples to be classified
Target function $y = f(\mathbf{x})$
Y: Label space
The set of all possible labels
Learning algorithm only sees examples of the function f in action
$\mathbf{x}_1, f(\mathbf{x}_1)$
$\mathbf{x}_2, f(\mathbf{x}_2)$
$\mathbf{x}_3, f(\mathbf{x}_3)$
$\vdots$
$\mathbf{x}_n, f(\mathbf{x}_n)$
Labeled training data
Learning algorithm
A learned function $g : X \rightarrow Y$
This is the training phase

Supervised learning: evaluation


X: Instance space
The set of all possible examples to be classified
Target function $y = f(\mathbf{x})$
Learned function $y = g(\mathbf{x})$
Y: Label space
The set of all possible labels

$$\begin{tikzcd}
 & f(\mathbf{x}) \arrow[dr, dashed] & \\
\text{Draw } \textbf{\textit{test}} \text{ example } \mathbf{x} \in X \arrow[ur, dashed] \arrow[dr] & & \text{Are they different? How different?} \\
 & g(\mathbf{x}) \arrow[ur] &
\end{tikzcd}$$

Apply model to many test examples and compare to the target's prediction
Aggregate these results to get a quality measure

The general setting for supervised learning

Given: training examples that are pairs of the form $(\mathbf{x}, f(\mathbf{x}))$

The goal of learning: use the training examples to find a good approximation for f

The label determines the kind of problem we have

- **Binary classification**: label space = $\{-1, 1\}$
- **Multiclass classification**: label space = $\{1, 2, 3, \ldots, K\}$
- **Regression**: label space $= \Re$

The general setting for supervised learning

Given: training examples that are pairs of the form $(\mathbf{x}, f(\mathbf{x}))$

Typically the input x is represented as feature vectors

- E.g.,: $\mathbf{x} \in \{0,1\}^d$ or $\mathbf{x} \in \Re^d$ (d-dimensional vectors)
- A deterministic mapping from instances in your problem (e.g., news articles) to features

The function f is unknown

Instances: real things to categorize, like emails, articles, pictures

Features: "interesting" attributes of the instances, like 1 if email contains word free, 0 otherwise, pixel patterns, …

Features form a vector space: d-dimensional vectors form a d-dimensional vector space. Each number in vector is a single dimension that captures one feature

Using supervised learning

We should be able to specify

1. What is our instance space?
   - What are the inputs to the problem? What are the features?

2. What is our label space?
   - What kind of learning task are we dealing with?

3. What is our hypothesis space?
   - What functions should the learning algorithm search over?

4. What is our learning algorithm?
   - How do we learn the model from the labeled data?

5. What is our loss function or evaluation metric?
   - How do we measure success? What drives learning?

Much of the rest of the semester

Using Supervised Learning

EXAMPLES

1. The instance space X

X: Instance space

The set of all possible examples to be classified

E.g., the set of all possible names, documents, sentences, images, emails, …

Designing an appropriate *feature representation* of the instance space is crucial

Instances $x \in X$ are defined by features/ attributes

What might features be?

1. The instance space X

X: Instance space

The set of all possible examples to be classified

E.g., the set of all possible names, documents, sentences, images, emails, …

Designing an appropriate *feature representation* of the instance space is crucial

Instances $x \in X$ are defined by features/ attributes

Features could be Boolean

- Example: does the email contain the word "free"

Features could be real-valued

- Example: what is the height of the person?
- Example: what was the stock price yesterday?

Features could be hand-crafted or themselves learned

1. The instance space X

Features are supposed to capture all of the information needed for a learned system to make its prediction

- Think of them as the sensory inputs for the learned system

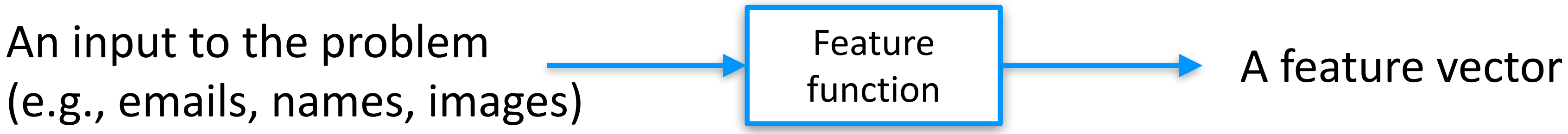
Not all information about the instances is necessary or relevant

- Bad features could even confuse a learner

Instances as feature vectors

$$\begin{tikzcd}
\begin{array}{l}\text{An input to the problem}\\ \text{(e.g., emails, names, images)}\end{array} \arrow[r] & \boxed{\begin{array}{c}\text{Feature}\\ \text{function}\end{array}} \arrow[r] & \text{A feature vector}
\end{tikzcd}$$

Instances as feature vectors



Feature functions, also known as feature extractors

- Often deterministic, but could also be learned
- Convert the examples to a collection of attributes (typically thought of as high-dimensional vectors)

Important part of the design of a learning based solution

Instances as feature vectors

Feature functions convert inputs to vectors

Instances as feature vectors

Feature functions convert inputs to vectors

The input space X is a d-dimensional vector space (e.g., $\Re^d$ or $\{0,1\}^d$)

– Each dimension is one feature, we have d features in all

Instances as feature vectors

Feature functions convert inputs to vectors

The input space X is a d-dimensional vector space (e.g., $\Re^d$ or $\{0,1\}^d$)

- Each dimension is one feature, we have d features in all

Each $\mathbf{x} \in X$ is a feature vector

- Each $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is a point in the vector space with d dimensions (hence the boldface $\mathbf{x}$)

Instances as feature vectors

Feature functions convert inputs to vectors

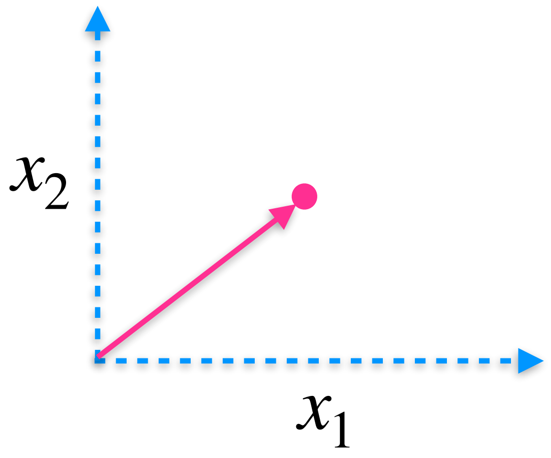
The input space X is a d-dimensional vector space (e.g., $\Re^d$ or $\{0,1\}^d$)

– Each dimension is one feature, we have d features in all

Each $\mathbf{x} \in X$ is a feature vector

– Each $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is a point in the vector space with d dimensions (hence the boldface $\mathbf{x}$)

$$\mathbf{x} = [x_1, x_2]$$



Feature functions produce feature vectors

What were some features for the badges game?

$$\begin{tikzcd}\text{Names} \arrow[r] & \boxed{\begin{array}{c}\text{Feature}\\\text{function}\end{array}} \arrow[r] & ?\end{tikzcd}$$

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: “The second letter of the name”

3min: Think about we might represent this feature using a feature vector

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: “The second letter of the name”
  - Norman o → [0 0 0 0 . . . 1 . . .]
  - Karen a → [1 0 0 0 . . . 0 . . .]
  - Dan a → [1 0 0 0 . . . 0 . . .]
  - Danny a → [1 0 0 0 . . . 0 . . .]
  - Saray a → [1 0 0 0 . . . 0 . . .]
  - Wai a → [1 0 0 0 . . . 0 . . .]

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: "The second letter of the name"
  - Norman a → [0 0 0 0 … 1 …]
  - Karen a → [1 0 0 0 … 0 …]
  - Dan a → [1 0 0 0 … 0 …]
  - Danny a → [1 0 0 0 … 0 …]
  - Saray a → [1 0 0 0 … 0 …]
  - Wai a → [1 0 0 0 … 0 …]

What is the dimensionality of these feature vectors?

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: "The second letter of the name"
    - Norman a $\rightarrow$ [0 0 0 0 … 1 …]
    - Karen a $\rightarrow$ [1 0 0 0 … 0 …]
    - Dan a $\rightarrow$ [1 0 0 0 … 0 …]
    - Danny a $\rightarrow$ [1 0 0 0 … 0 …]
    - Saray a $\rightarrow$ [1 0 0 0 … 0 …]
    - Wai a $\rightarrow$ [1 0 0 0 … 0 …]

What is the dimensionality of these feature vectors?

26 (one dimension per letter)

Feature functions produce feature vectors

When designing feature functions, think of them as templates

– Feature: “The second letter of the name”

• Norman	$a \rightarrow [0\ 0\ 0\ 0 \ldots 1 \ldots]$
• Karen	$a \rightarrow [1\ 0\ 0\ 0 \ldots 0 \ldots]$
• Dan	$a \rightarrow [1\ 0\ 0\ 0 \ldots 0 \ldots]$
• Danny	$a \rightarrow [1\ 0\ 0\ 0 \ldots 0 \ldots]$
• Saray	$a \rightarrow [1\ 0\ 0\ 0 \ldots 0 \ldots]$
• Wai	$a \rightarrow [1\ 0\ 0\ 0 \ldots 0 \ldots]$

What is the dimensionality of these feature vectors?

26 (one dimension per letter)

Vectors where exactly one dimension is 1 and all others are 0 are called one-hot vectors

This is the one-hot representation of the feature “The second letter of the name”

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: "The second letter of the name"
  - Norman o → [0 0 0 0 … 1 …]
  - Karen a → [1 0 0 0 … 0 …]
  - Dan a → [1 0 0 0 … 0 …]
  - Danny a → [1 0 0 0 … 0 …]
  - Saray a → [1 0 0 0 … 0 …]
  - Wai a → [1 0 0 0 … 0 …]
- Feature: "The length of the first name"
  - Norman → 6
  - Karen → 5

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: "The second letter of the name"
  - Norman o $\rightarrow$ [0 0 0 0 ... 1 ...]
  - Karen a $\rightarrow$ [1 0 0 0 ... 0 ...]
  - Dan a $\rightarrow$ [1 0 0 0 ... 0 ...]
  - Danny a $\rightarrow$ [1 0 0 0 ... 0 ...]
  - Saray a $\rightarrow$ [1 0 0 0 ... 0 ...]
  - Wai a $\rightarrow$ [1 0 0 0 ... 0 ...]
- Feature: "The length of the first name"
  - Norman $\rightarrow$ 6
  - Karen $\rightarrow$ 5
- "The second letter of the name, The length of the first name, The length of the last name"
  - Norman Danner $\rightarrow$ [0 0 0 0 ... 1 ... 6 6]
  - Karen Collins $\rightarrow$ [1 0 0 0 ... 0 ... 5 7]

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: "The second letter of the name"
  - Norman o → [0 0 0 0 … 1 …]
  - Karen a → [1 0 0 0 … 0 …]
  - Dan a → [1 0 0 0 … 0 …]
  - Danny a → [1 0 0 0 … 0 …]
  - Saray a → [1 0 0 0 … 0 …]
  - Wai a → [1 0 0 0 … 0 …]
- Feature: "The length of the first name"
  - Norman → 6
  - Karen → 5
- "The second letter of the name, The length of the first name, The length of the last name"
  - Norman Danner → [0 0 0 0 … 1 … 6 6]
  - Karen Collins → [1 0 0 0 … 0 … 5 7]

How do we work with multiple features?

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: "The second letter of the name"
  - Norman o → [0 0 0 0 … 1 …]
  - Karen a → [1 0 0 0 … 0 …]
  - Dan a → [1 0 0 0 … 0 …]
  - Danny a → [1 0 0 0 … 0 …]
  - Saray a → [1 0 0 0 … 0 …]
  - Wai a → [1 0 0 0 … 0 …]
- Feature: "The length of the first name"
  - Norman → 6
  - Karen → 5
- "The second letter of the name, The length of the first name, The length of the last name"
  - Norman Danner → [0 0 0 0 … 1 … 6 6]
  - Karen Collins → [1 0 0 0 … 0 … 5 7]

How do we work with multiple features?

Features can be accumulated by concatenating the vectors

Good features are essential

Features determine how well a task can be learned

- E.g., a bad feature for our game: “Is there a day of the week that begins with the last letter of the first name?
- *Why would we think that this is a bad feature?*

Much effort goes into designing (or learning) features

Will touch upon general principles for designing good features

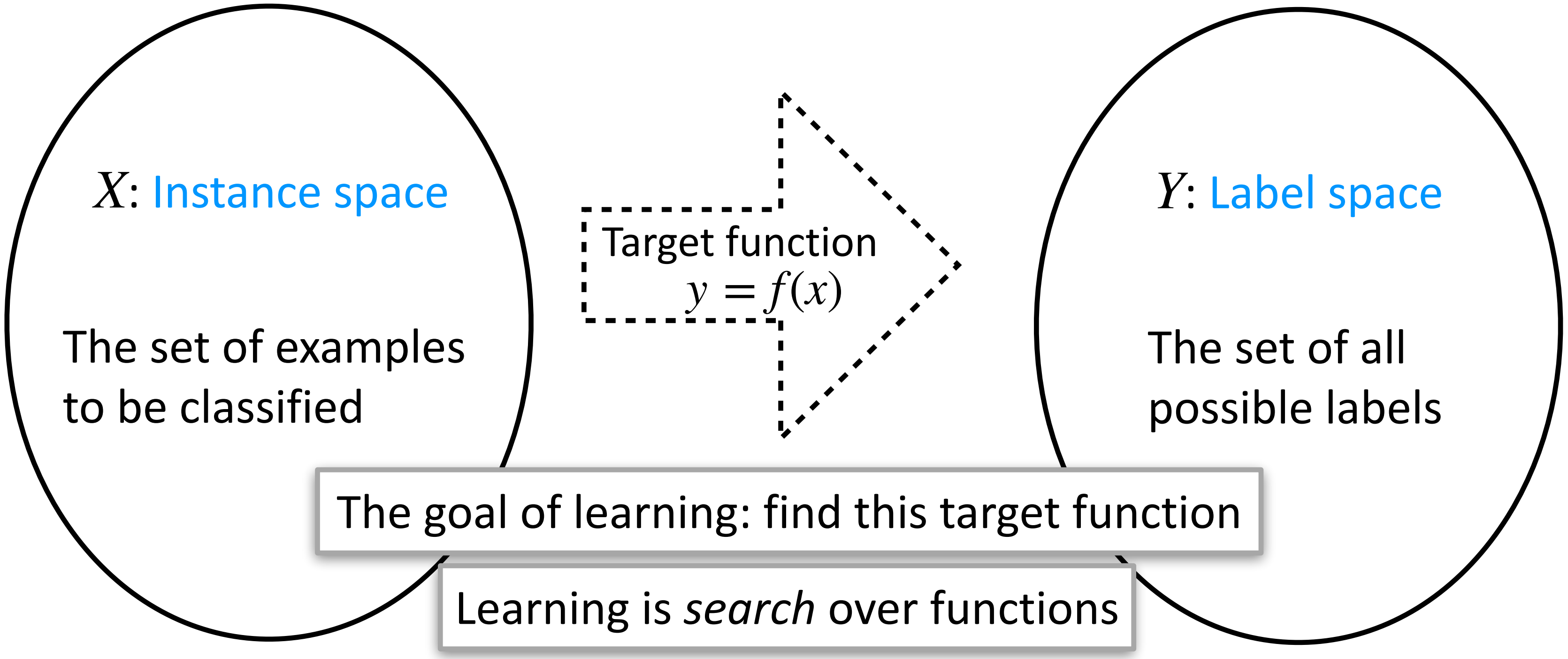
- But feature definition largely domain specific
- Comes with experience

Using supervised learning

✓ What is our instance space?
- What are the inputs to the problem? What are the features?

2. What is our label space?
   - What kind of learning task are we dealing with?
3. What is our hypothesis space?
   - What functions should the learning algorithm search over?
4. What is our learning algorithm?
   - How do we learn the model from the labeled data?
5. What is our loss function or evaluation metric?
   - How do we measure success? What drives learning?

2. The label space Y


X: Instance space
The set of examples to be classified
Target function $y = f(x)$
Y: Label space
The set of all possible labels
The goal of learning: find this target function
Learning is *search* over functions

E.g., the set of all possible names, documents, sentences, images, emails, …

E.g., {Spam, Not-Spam}, {+, -}, …

Label space depends on nature of the problem

Classification: the outputs are categorical

- *Binary* classification: two possible labels
- *Multiclass* classification: K possible labels
- *Structured* classification: e.g., graph valued outputs

Label space depends on nature of the problem

Classification: the outputs are categorical

- *Binary* classification: two possible labels
- *Multiclass* classification: K possible labels
- *Structured* classification: e.g., graph valued outputs

Classification is the primary focus of this class

Using supervised learning

✓ What is our instance space?

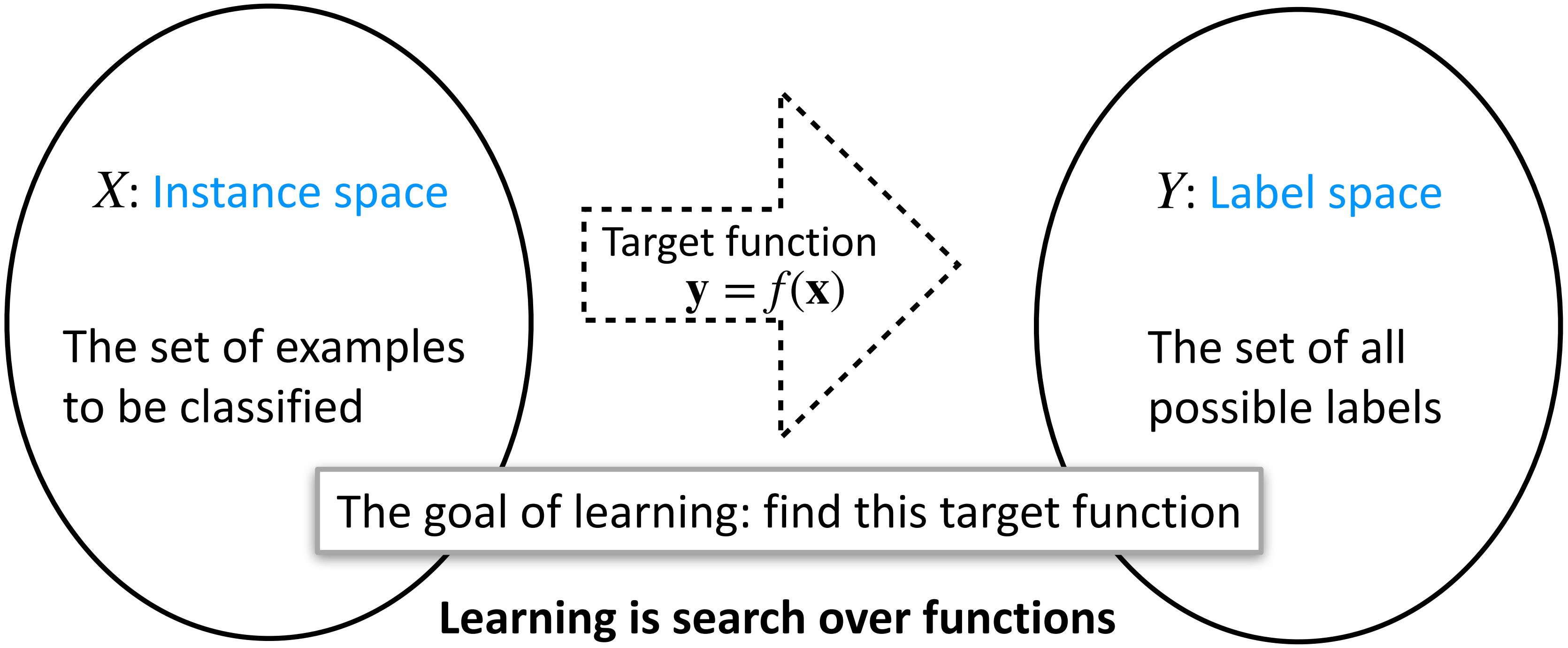
- What are the inputs to the problem? What are the features?

✓ What is our label space?

- What kind of learning task are we dealing with?

3. What is our hypothesis space?
   - What functions should the learning algorithm search over?
4. What is our learning algorithm?
   - How do we learn the model from the labeled data?
5. What is our loss function or evaluation metric?
   - How do we measure success? What drives learning?

Target function

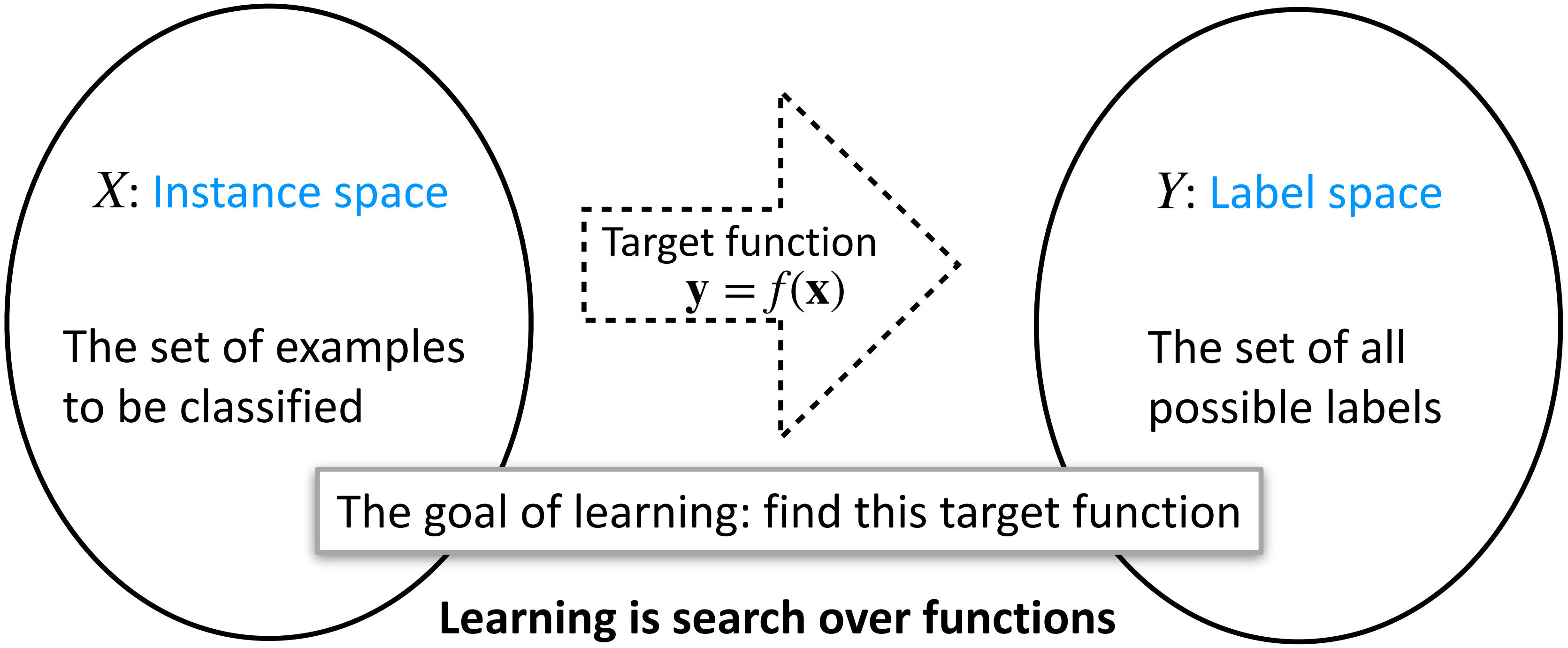

X: Instance space
The set of examples to be classified
Target function $\mathbf{y} = f(\mathbf{x})$
Y: Label space
The set of all possible labels
The goal of learning: find this target function

Learning is search over functions

E.g., the set of all possible names, documents, sentences, images, emails, …

E.g., {Spam, Not-Spam}, {+, -}, …

Target function


X: Instance space
The set of examples to be classified
Target function $\mathbf{y} = f(\mathbf{x})$
Y: Label space
The set of all possible labels
The goal of learning: find this target function

Learning is search over functions

The *hypothesis space* is the set of functions we consider for this search

Example of search over functions

0=False, 1=True

$$\begin{tikzcd} x_1 \arrow[dr] & & \\ & \boxed{\text{Unknown function } f} \arrow[r] & y = f(x_1, x_2) \\ x_2 \arrow[ur] & & \end{tikzcd}$$

Features

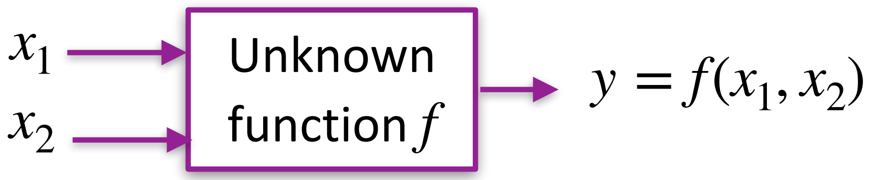
Label

Our dataset

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Example of search over functions

0=False, 1=True

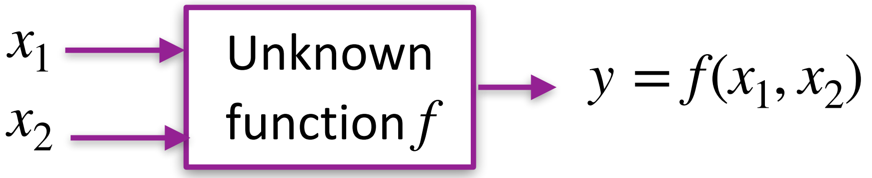


x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Can you learn this function? What is it?

Example of search over functions

0=False, 1=True



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

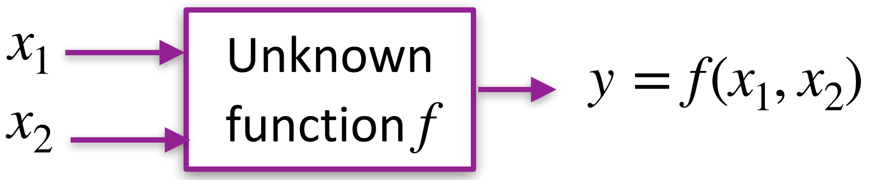
Can you learn this function? What is it?

$(x_1$ AND $x_2)$

$(x_1 \wedge x_2)$

Example of search over functions

0=False, 1=True



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

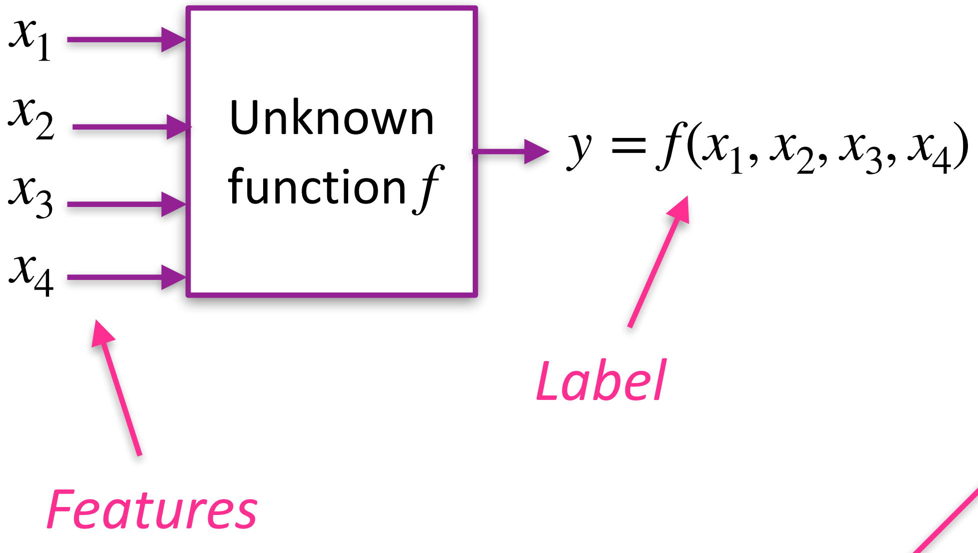
Can you learn this function? What is it?

$(x_1$ AND $x_2)$
$(x_1 \wedge x_2)$

Are we sure this function is correct? Why?

Example of search over functions

0=False, 1=True

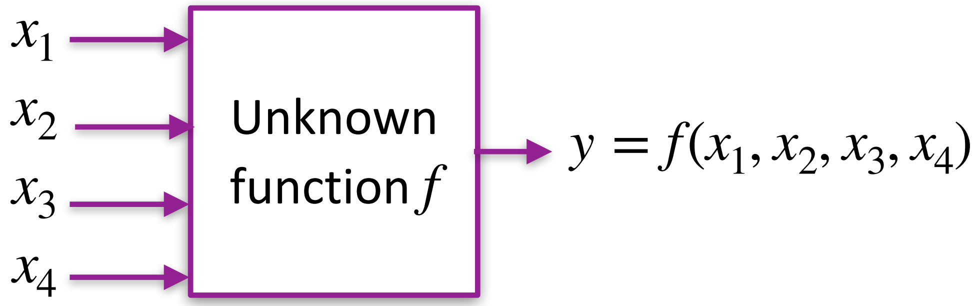


Our dataset

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Example of search over functions

0=False, 1=True



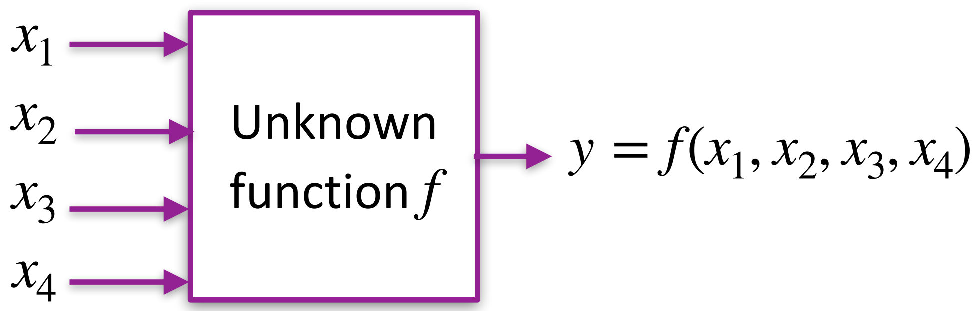
x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Can you learn this function? What is it?

3 min: Talk with your neighbors

Example of search over functions

0=False, 1=True



x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Can you learn this function? What is it?

(x_3 AND x_4) OR (x_2 AND x_4)

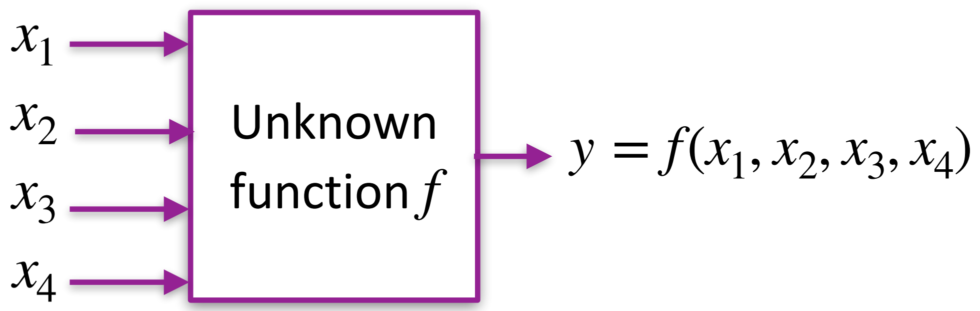
$(x_3 \land x_4) \lor (x_1 \land x_4)$

x_4 AND(NOT x_2)

$x_4 \land \neg x_2$

Example of search over functions

0=False, 1=True



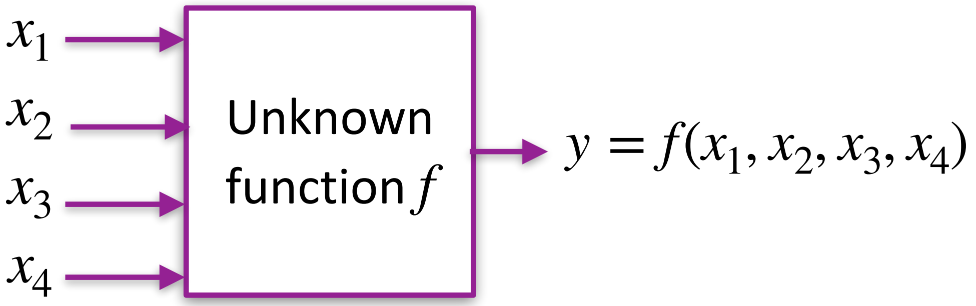
x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Can you learn this function? What is it?

How do you know function is correct? We have only seen 7 outputs after all …

Example of search over functions

0=False, 1=True



x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Can you learn this function? What is it?

How do you know function is correct? We have only seen 7 outputs after all …

The fundamental problem: machine learning is ill-posed

Is learning possible at all?

Complete ignorance: How many possible Boolean functions over 4 input features?

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Is learning possible at all?

Complete ignorance: How many possible Boolean functions over 4 input features?

4 Boolean input features $\rightarrow 2^4 = 16$ permutations of values, i.e., 4 rows in table

For these rows, how many different ways to set labels?

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Is learning possible at all?

Complete ignorance: How many possible Boolean functions over 4 input features?

4 Boolean input features $\rightarrow 2^4 = 16$ permutations of values, i.e., 4 rows in table

16 rows → each way to fill outputs gives a different function. $2^{16} = 65536$ different ways target outputs can be set, so 2^{16} functions in all

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Is learning possible at all?

Complete ignorance: How many possible Boolean functions over 4 input features?

4 Boolean input features $\rightarrow 2^4 = 16$ permutations of values, i.e., 4 rows in table

16 rows $\rightarrow$ each way to fill outputs gives a different function. $2^{16} = 65536$ different ways target outputs can be set, so 2^{16} functions in all

x_1	x_2	x_3	x_4	y	
0	0	0	0	?	0
0	0	0	1	?	1
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	?	1
1	0	0	0	?	1
1	0	0	1	1	1
1	0	1	0	?	1
1	0	1	1	?	1
1	1	0	0	0	1
1	1	0	1	?	1
1	1	1	0	?	1
1	1	1	1	?	1

Is learning possible at all?

Complete ignorance: How many possible Boolean functions over 4 input features?

4 Boolean input features $\rightarrow 2^4 = 16$ permutations of values, i.e., 4 rows in table

16 rows $\rightarrow$ each way to fill outputs gives a different function. $2^{16} = 65536$ different ways target outputs can be set, so 2^{16} functions in all

x_1	x_2	x_3	x_4	y	
0	0	0	0	?	1
0	0	0	1	?	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	?	1
1	0	0	0	?	1
1	0	0	1	1	1
1	0	1	0	?	1
1	0	1	1	?	1
1	1	0	0	0	1
1	1	0	1	?	1
1	1	1	0	?	1
1	1	1	1	?	1

Is learning possible at all?

Complete ignorance: There are $2^{16} = 65536$ possible Boolean functions over 4 input features

- Why? There are 16 possible outputs. Each way to fill these 16 slots is a different function, giving 2^{16} functions.

Problem: We have seen only 7 outputs, leaving 2^9 possibilities for f

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Is learning possible at all?

Complete ignorance: There are $2^{16} = 65536$ possible Boolean functions over 4 input features

- Why? There are 16 possible outputs. Each way to fill these 16 slots is a different function, giving 2^{16} functions.

Problem: We have seen only 7 outputs, leaving 2^9 possibilities for f

How could we possibly know the rest without seeing every label?

- Think of an adversary filling in the labels every time you make a guess at the function

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Is learning possible at all?

Complete ignorance: There are $2^{16} = 65536$ possible Boolean functions over 4 input features

- Why? There are 16 possible outputs. Each way to fill these 16 slots is a different function, giving 2^{16} functions.

Problem: We have seen only 7 outputs, leaving 2^9 possibilities for f

How could we possibly know the rest without seeing every label?

- Think of an adversary filling in the labels every time you make a guess at the function

Is learning possible? What do you think?

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Solution: restrict the search space

The “When in doubt, make an assumption” school of thought!

Hypothesis space

– set of possible functions we consider

Solution: restrict the search space

The “When in doubt, make an assumption” school of thought!

Hypothesis space

– set of possible functions we consider

We were looking at the space of all Boolean functions …

Instead, choose a hypotheses space that is ***not*** all possible functions

– Only **simple conjunctions**: with 4 variables, there are only 16 conjunctions without negations

– **m-of-n rules**: pick a set of n variables. At least m of them must be true

– **Linear functions**

– **Deep neural networks**

– …

Example hypothesis space #1

Simple conjunctions: how many simple conjunctive rules of the form $g(x) = x_i \wedge x_j \wedge x_k \cdots$?

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Example hypothesis space #1

Simple conjunctions: how many simple conjunctive rules of the form $g(x) = x_i \wedge x_j \wedge x_k \cdots$?

16 functions: $x_1 = 1$, everything else 0, and so on

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Example hypothesis space #1

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

What are some of these rules?

Example hypothesis space #1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

Rule	Rule
Always False	
x_1	
x_2	
x_3	
x_4	

Example hypothesis space #1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

Rule	Rule
Always False	$x_2 \wedge x_3$
x_1	$x_2 \wedge x_4$
x_2	$x_3 \wedge x_4$
x_3	
x_4	
$x_1 \wedge x_2$	
$x_1 \wedge x_3$	
$x_1 \wedge x_4$	

Example hypothesis space #1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

Rule	Rule
Always False	$x_2 \wedge x_3$
x_1	$x_2 \wedge x_4$
x_2	$x_3 \wedge x_4$
x_3	$x_1 \wedge x_2 \wedge x_3$
x_4	$x_1 \wedge x_2 \wedge x_4$
$x_1 \wedge x_2$	$x_1 \wedge x_3 \wedge x_4$
$x_1 \wedge x_3$	$x_2 \wedge x_3 \wedge x_4$
$x_1 \wedge x_4$	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$

Example hypothesis space #1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

Rule	Rule
Always False	$x_2 \wedge x_3$
x_1	$x_2 \wedge x_4$
x_2	$x_3 \wedge x_4$
x_3	$x_1 \wedge x_2 \wedge x_3$
x_4	$x_1 \wedge x_2 \wedge x_4$
$x_1 \wedge x_2$	$x_1 \wedge x_3 \wedge x_4$
$x_1 \wedge x_3$	$x_2 \wedge x_3 \wedge x_4$
$x_1 \wedge x_4$	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$

Does one of these rules explain everything? What do you think?

Example hypothesis space #1

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Rule	Counter-example	Rule	Counter-example
Always False	1001	$x_2 \wedge x_3$	0011
x_1	1100	$x_2 \wedge x_4$	0011
x_2	0100	$x_3 \wedge x_4$	1001
x_3	0110	$x_1 \wedge x_2 \wedge x_3$	0011
x_4	0101	$x_1 \wedge x_2 \wedge x_4$	0011
$x_1 \wedge x_2$	1100	$x_1 \wedge x_3 \wedge x_4$	0011
$x_1 \wedge x_3$	0011	$x_2 \wedge x_3 \wedge x_4$	0011
$x_1 \wedge x_4$	0011	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011

Example hypothesis space #1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

Is there a consistent hypothesis in this space?

Rule	Counter-example	Rule	Counter-example
Always False	1001	$x_2 \wedge x_3$	0011
x_1	1100	$x_2 \wedge x_4$	0011
x_2	0100	$x_3 \wedge x_4$	1001
x_3	0110	$x_1 \wedge x_2 \wedge x_3$	0011
x_4	0101	$x_1 \wedge x_2 \wedge x_4$	0011
$x_1 \wedge x_2$	1100	$x_1 \wedge x_3 \wedge x_4$	0011
$x_1 \wedge x_3$	0011	$x_2 \wedge x_3 \wedge x_4$	0011
$x_1 \wedge x_4$	0011	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011

Example hypothesis space #1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Simple conjunctions: there are only 16 simple conjunctive rules of the form

$$g(x) = x_i \wedge x_j \wedge x_k \cdots$$

Is there a consistent hypothesis in this space?

Rule	Counter-example	Rule	Counter-example

No simple conjunction explains the data!

(Confirm each counterexample by going through the list)

Our hypothesis space is too small and the true function we are looking or is not in it!

Solution: restrict the search space

The “When in doubt, make an assumption” school of thought!

Hypothesis space

– set of possible functions we consider

Solution: restrict the search space

The "When in doubt, make an assumption" school of thought!

Hypothesis space

- set of possible functions we consider

We were looking at the space of all Boolean functions. Instead, choose a hypotheses space that is ***not*** all possible functions

Solution: restrict the search space

The “When in doubt, make an assumption” school of thought!

Hypothesis space

– set of possible functions we consider

We were looking at the space of all Boolean functions. Instead, choose a hypotheses space that is ***not*** all possible functions

How do we pick hypothesis space? Use some prior knowledge or guess

What if the hypothesis space is so small that nothing in it agrees with the data? Need a hypothesis space that is flexible enough

Example hypothesis space #2

m-of-n rules: pick a subset with n variables. The label y is 1 if at least m of the variables are 1

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Example hypothesis space #2

m-of-n rules: pick a subset with n variables. The label y is 1 if at least m of the variables are 1

Example: if at least 2 of $\{x_1, x_2, x_3, x_4\}$ are 1, then the output is 1. Otherwise the output is 0

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

Example hypothesis space #2

m-of-n rules: pick a subset with n variables. The label y is 1 if at least m of the variables are 1

Example: if at least 2 of $\{x_1, x_2, x_3, x_4\}$are 1, then the output is 1. Otherwise the output is 0

Is there a consistent hypothesis in this space?

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

m-of-n rules for 4 variables

Index	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

Notation: 1 variable from the set on the left is 1

Variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	2	3	–	–
$\{x_1, x_3\}$	1	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	*	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

m-of-n rules for 4 variables

Index	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

Notation: 2 variables from the set on the left are 1

Variables	1-of	2-of	3-of	4-of	Variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–	$\{x_2, x_4\}$	2	3	–	–
$\{x_2\}$	2	–	–	–	$\{x_2, x_4\}$	2	3	–	–
$\{x_3\}$	1	–	–	–	$\{x_3, x_4\}$	4	4	–	–
$\{x_4\}$	7	–	–	–	$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2\}$	2	3	–	–	$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3\}$	1	3	–	–	$\{x_1, x_3, x_4\}$	1	$*$	3	–
$\{x_1, x_4\}$	6	3	–	–	$\{x_2, x_3, x_4\}$	1	5	3	–
					$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

m-of-n rules for 4 variables

Index	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

Notation: 2 variables from the set on the left are 1

Variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	2	3	–	–
$\{x_1, x_3\}$	1	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	*	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

If set has k variables, than can choose at most k

m-of-n rules for 4 variables

Index	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

Variables	1-of	2-of	3-of	4-of	Variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–	$\{x_2, x_4\}$	2	3	–	–
$\{x_2\}$	2	–	–	–	$\{x_2, x_4\}$	2	3	–	–
$\{x_3\}$	1	–	–	–	$\{x_3, x_4\}$	4	4	–	–
$\{x_4\}$	7	–	–	–	$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2\}$	2	3	–	–	$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3\}$	1	3	–	–	$\{x_1, x_3, x_4\}$	1	*	3	–
$\{x_1, x_4\}$	6	3	–	–	$\{x_2, x_3, x_4\}$	1	5	3	–
					$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Value: index of the counterexample

m-of-n rules for 4 variables

Index	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

Variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	2	3	–	–
$\{x_1, x_3\}$	1	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	*	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Are there any consistent hypotheses?
(I.e., does any hypothesis satisfy all of the data?)

m-of-n rules for 4 variables

Index	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

Variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	2	3	–	–
$\{x_1, x_3\}$	1	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	*	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Found a consistent hypothesis!
(In practice find, e.g., with neural net)

General strategies for Machine Learning

Pick expressive hypothesis spaces

- Decision trees, neural networks, m-of-n functions, linear functions, …

Develop algorithms for finding a hypothesis in our hypothesis space, that fits the data well (or well enough)

Hope that the hypothesis generalizes

Perspectives on learning

Learning is the removal of our ***remaining*** uncertainty over a hypothesis space

- If we ***knew*** that the unknown function is a simple conjunction, then we could use the training data to infer which one it is.

Perspectives on learning

Learning is the removal of our ***remaining*** uncertainty over a hypothesis space

- If we ***knew*** that the unknown function is a simple conjunction, then we could use the training data to infer which one it is.

Learning requires guessing a ***good, small*** hypothesis class

- We can start with a very small class and enlarge it until it contains an hypothesis that fits the data.
- And we could be wrong. We could find a consistent hypothesis and still be incorrect when given a new example!

Using supervised learning

✓ What is our instance space?

- What are the inputs to the problem? What are the features?

✓ What is our label space?

- What kind of learning task are we dealing with?

✓ What is our hypothesis space?

- What functions should the learning algorithm search over?

4. What is our learning algorithm?
   - How do we learn the model from the labeled data?
5. What is our loss function or evaluation metric?
   - How do we measure success? What drives learning?

Much of the rest of the semester