Supervised Learning: The Setup

Machine Learning

THE UNIVERSITY OF UTAH

Last lecture

We saw

- What is learning?
  Learning as generalization

- The badges game

This lecture

- More badges

- Formalizing supervised learning
  - Instance space and features

    What are inputs to the learning problem?
  - Label space

    What is the output of the learned function
  - Hypothesis space

    What is being learned?

The badges game

Let’s play

Name	Label
Claire Cardie	–
Jude Shavlik	+
Eric Baum	–
Haym Hirsh	–
Leslie Pack Kaelbling	+
Philip Chan	+

(Full data on the class website, you can stare at it longer if you want)

Let’s play

Name	Label
Claire Cardie	–
Jude Shavlik	+
Eric Baum	–
Haym Hirsh	–
Leslie Pack Kaelbling	+
Philip Chan	+

What is the label for *Indiana Jones*?

(Full data on the class website, you can stare at it longer if you want)

Let's play

Name	Label
Claire Cardie	–
Jude Shavlik	+
Eric Baum	–
Haym Hirsh	–
Leslie Pack Kaelbling	+
Philip Chan	+

How were the labels generated?

(Full data on the class website, you can stare at it longer if you want)

Let's play

Name	Label
Claire Cardie	–
Jude Shavlik	+
Eric Baum	–
Haym Hirsh	–
Leslie Pack Kaelbling	+
Philip Chan	+

How were the labels generated?

```
If third letter of last name is in first half of the alphabet:
    label = +
else
    label = -
```

(Full data on the class website, you can stare at it longer if you want)

Questions to think about

How could you be certain that you got the right function?

- How did you arrive at it?

Learning issues:

- Is this prediction or just modeling data? Is there a difference?
- How did you know that you should look at the letters?
- What background knowledge about letters did you use? How did you know that it is relevant?
- What “learning algorithm” did you use?

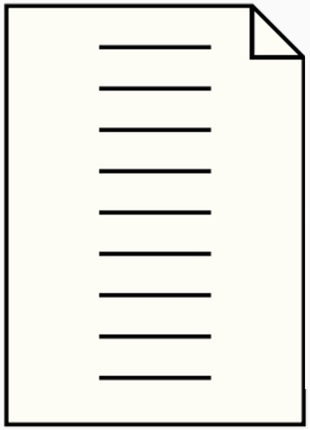
What is supervised learning?

Instances and Labels

Running example: Automatically tag news articles

Instances and Labels

Running example: Automatically tag news articles



An instance of a news article that needs to be classified

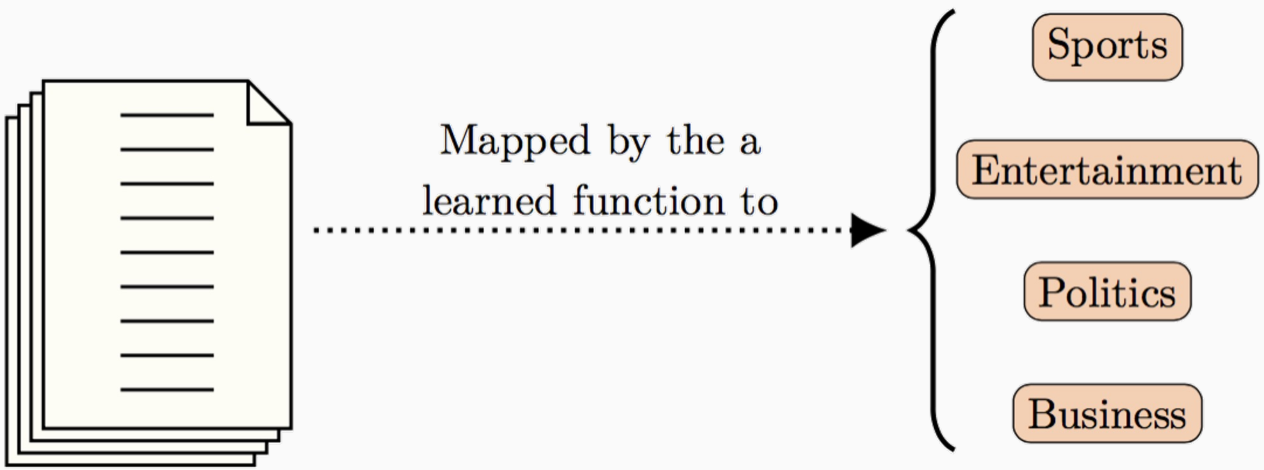
Instances and Labels

Running example: Automatically tag news articles

An instance of a news article that needs to be classified

Instances and Labels

Running example: Automatically tag news articles



Instance Space: All possible news articles

Label Space: All possible labels

Instances and Labels

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Eg: The set of all possible names, documents, sentences, images, emails, etc

Instances and Labels

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Eg: The set of all possible names, documents, sentences, images, emails, etc

$\mathcal{Y}$: Label Space

The set of all possible labels

Eg: {Spam, Not-Spam}, {+,-}, etc.

Instances and Labels

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function

$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {`Spam`, `Not-Spam`}, {+,−}, etc.

Instances and Labels

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

The goal of learning: Find this target function

Learning is search over functions

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {Spam, Not-Spam}, {+,−}, etc.

Supervised learning

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

Learning algorithm only sees examples of the function f in action

Supervised learning

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function

$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

Learning algorithm only sees examples of the function f in action

$$\begin{array}{c} x_1, f(x_1) \\ x_2, f(x_2) \\ x_3, f(x_3) \\ \vdots \\ x_n, f(x_n) \end{array}$$

Labeled training data

Supervised learning

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

Learning algorithm only sees examples of the function f in action

$x_1, f(x_1)$
$x_2, f(x_2)$
$x_3, f(x_3)$
$\vdots$
$x_n, f(x_n)$

Labeled training data

Learning algorithm

Supervised learning

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

Learning algorithm only sees examples of the function f in action

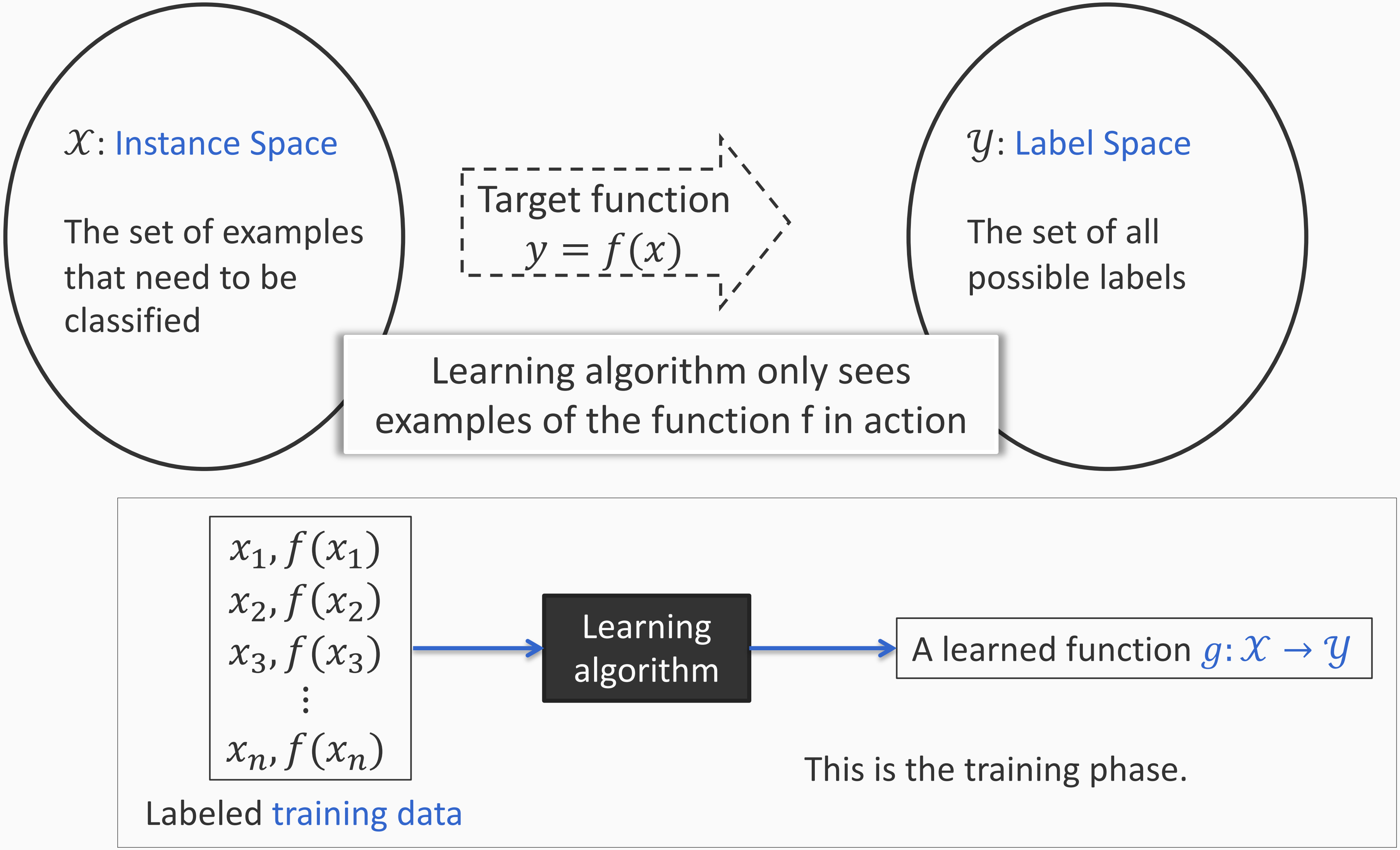
$$x_1, f(x_1)$$
$$x_2, f(x_2)$$
$$x_3, f(x_3)$$
$$\vdots$$
$$x_n, f(x_n)$$

Labeled training data

Learning algorithm

A learned function $g: \mathcal{X} \rightarrow \mathcal{Y}$

Supervised learning



Supervised learning

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

Learning algorithm only sees examples of the function f in action

$x_1, f(x_1)$
$x_2, f(x_2)$
$x_3, f(x_3)$
$\vdots$
$x_n, f(x_n)$

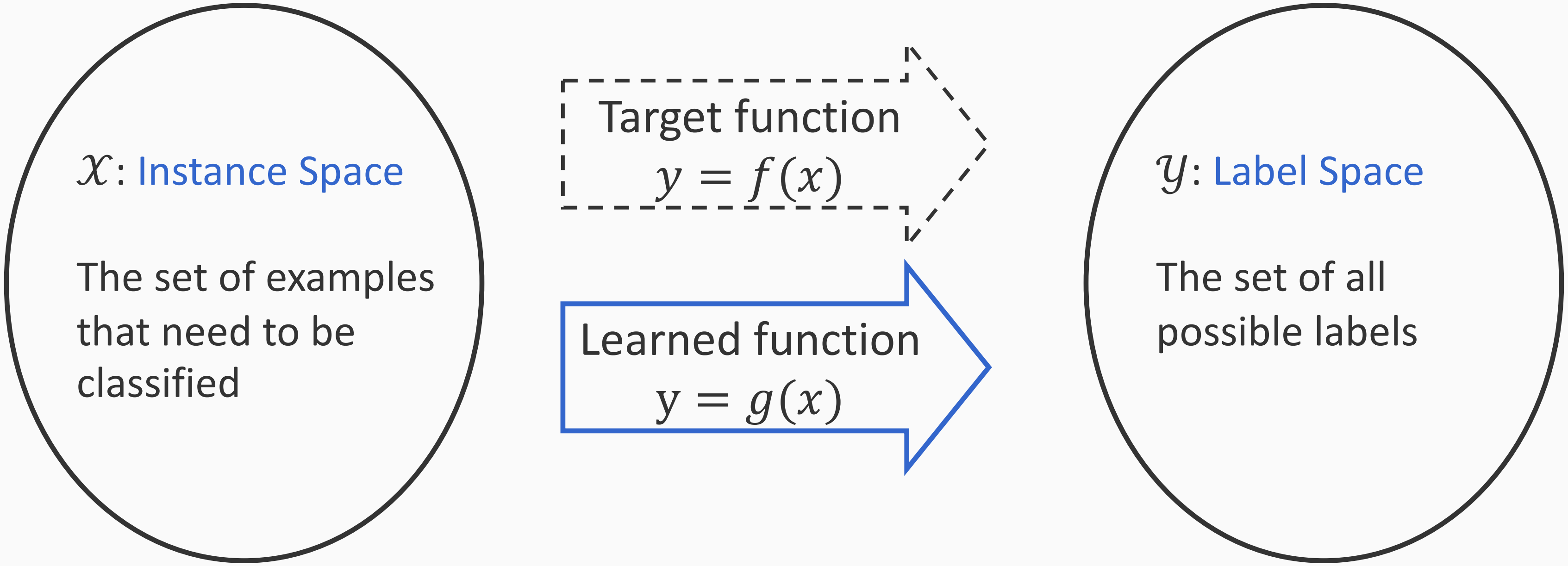
Labeled training data

Learning algorithm

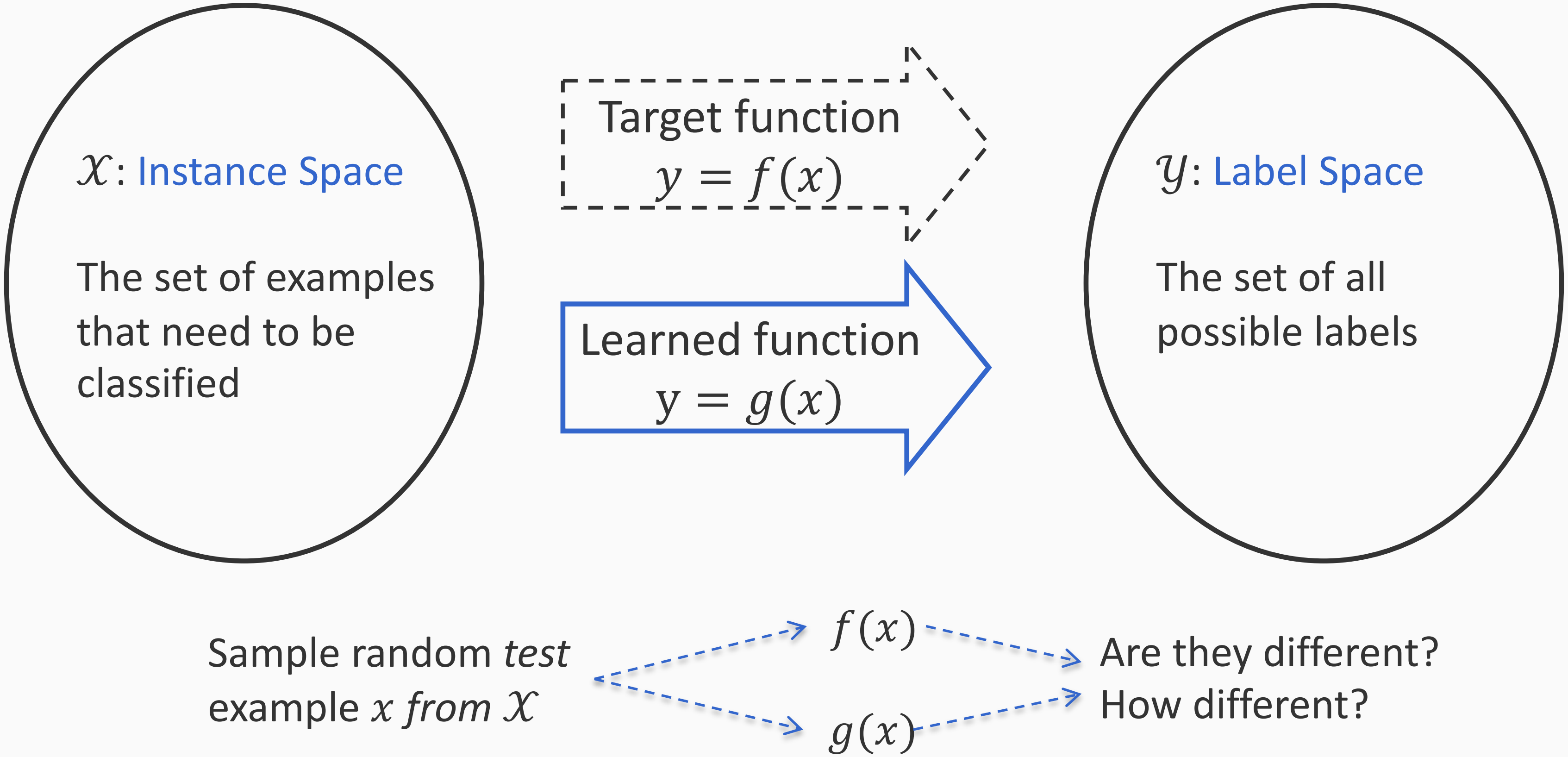
A learned function $g: \mathcal{X} \rightarrow \mathcal{Y}$

Can you think of other training protocols?

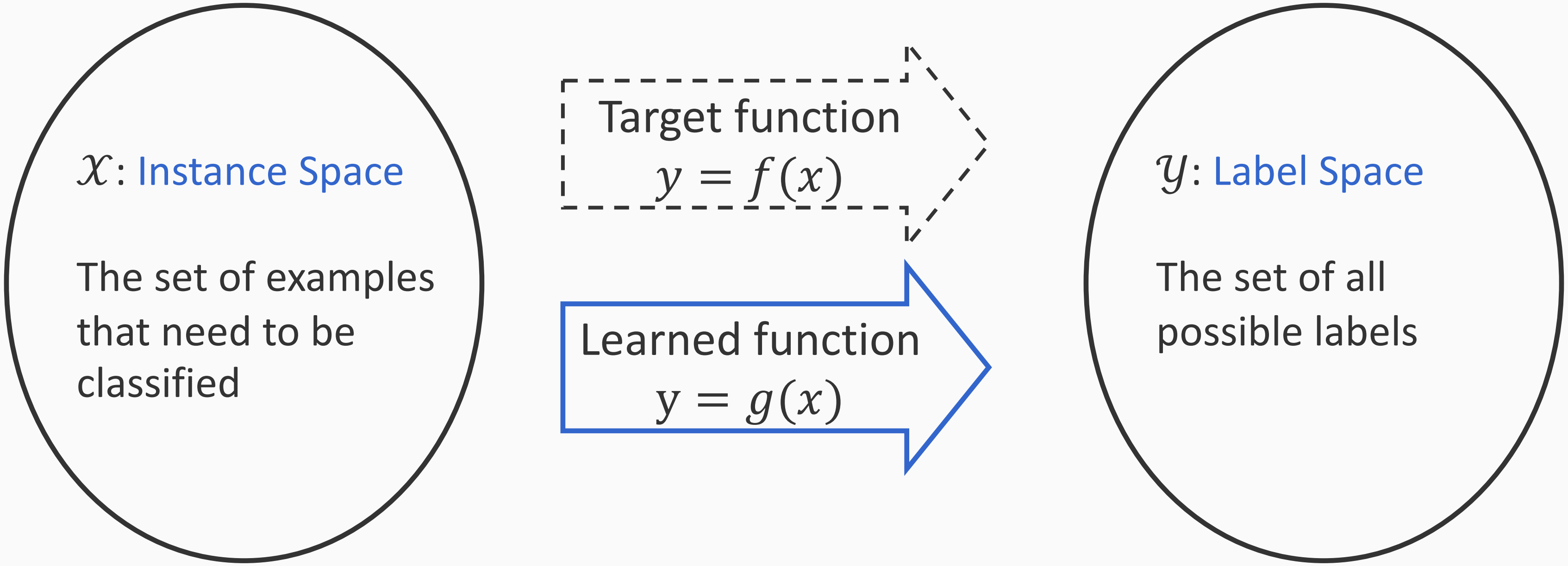
Supervised learning: Evaluation



Supervised learning: Evaluation



Supervised learning: Evaluation



Sample random *test* example x *from* $\mathcal{X}$

$f(x)$

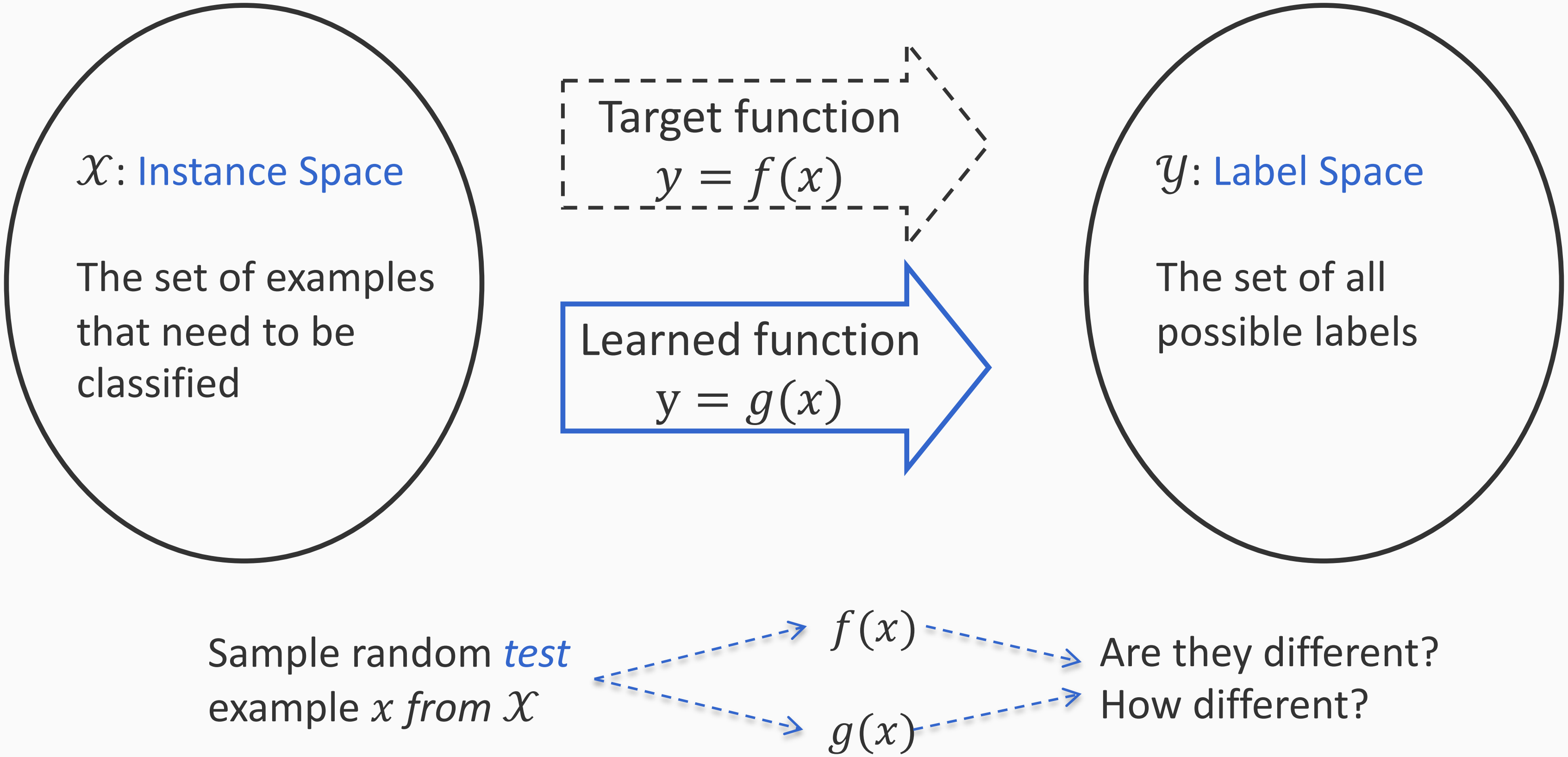
$g(x)$

Are they different?
How different?

Apply the model to many test examples and compare to the target's prediction

Aggregate these results to get a quality measure

Supervised learning: Evaluation



Apply the model to many test examples and compare to the target's prediction

Can we use these test examples during the training phase?

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

The function f is unknown

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

Typically the input x is represented as *feature vectors*

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

The function f is unknown

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

Typically the input x is represented as *feature vectors*

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

The function f is unknown

For a training example $(x, f(x))$, the value of $f(x)$ is called its *label*

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

Typically the input x is represented as *feature vectors*

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

The function f is unknown

For a training example $(x, f(x))$, the value of $f(x)$ is called its *label*

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

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

Typically the input x is represented as *feature vectors*

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

The function f is unknown

For a training example $(x, f(x))$, the value of $f(x)$ is called its *label*

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, $\cdots$, K}
- Regression: label space = $\Re$

Supervised learning: General setting

Given: Training examples that are pairs of the form $(x, f(x))$

Typically the input x is represented as *feature vectors*

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

The function f is unknown

For a training example $(x, f(x))$, the value of $f(x)$ is called its *label*

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, ⋯, K}
- Regression: label space = $\Re$

Questions?

Examples of binary classification

(the label space consists of two elements)

- Spam filtering
  - Is an email spam or not?
- Recommendation systems
  - Given user’s movie preferences, will she like a new movie?
- Anomaly detection
  - Is a smartphone app malicious?
  - Is a Twitter user a bot?
- Authorship identification
  - Were these two documents written by the same person?
- Time series prediction
  - Will the future value of a stock increase or decrease with respect to its current value?

On supervised learning

We should be able to decide:

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

2. What is our label space?
   What is the prediction task?

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

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

5. What is our loss function or evaluation metric?
   What is success?

1. The Instance Space $\mathcal{X}$

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function

$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

The goal of learning: Find this target function

Learning is search over functions

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {Spam, Not-Spam}, {+,-}, etc.

1. The Instance Space $\mathcal{X}$

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

The goal o

Lea

Eg: The set of all possible names, documents, sentences, images, emails, etc

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

Instances $x \in \mathcal{X}$ 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

Instances as feature vectors

An input to the problem
(Eg: emails, names, images)

Feature function

A feature vector

Instances as feature vectors

An input to the problem
(Eg: emails, names, images)

$\xrightarrow{\text{Feature function}}$

A feature vector

Feature functions, also known as feature extractors

- Often deterministic, but could also be learned
- Convert the examples a collection of attributes
  Typically thought of as **high-dimensional vectors**

Important part of the design of a learning based solution

1. The Instance Space $\mathcal{X}$

Features are supposed to capture all 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

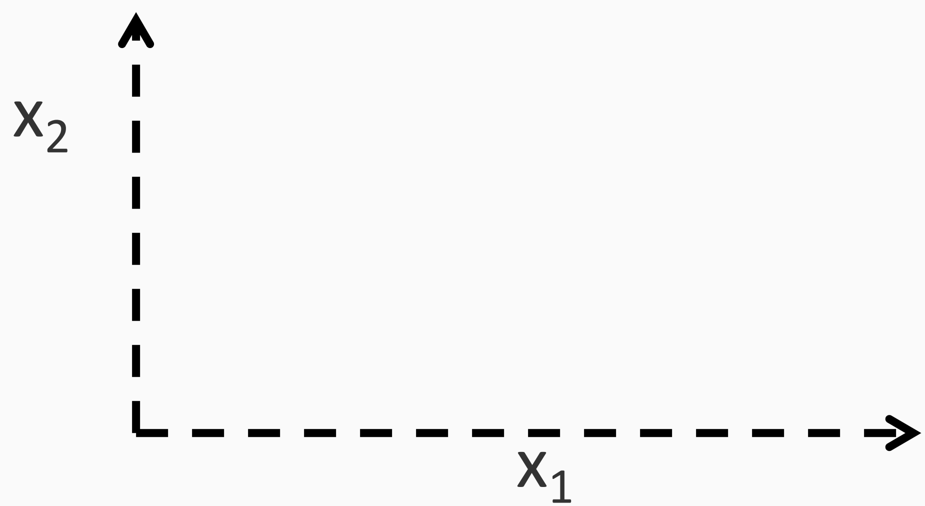
What might be good features for the badges game?

Instances as feature vectors

- Features functions convert inputs to vectors
- The instance space $\mathcal{X}$ ***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 $\mathrm{x} \in \mathcal{X}$ is a feature vector
  - Each $\mathrm{x} = [\mathrm{x}_1, \mathrm{x}_2, \cdots, x_d]$ is a point in the vector space with d dimensions

Instances as feature vectors

- Features functions convert inputs to vectors
- The instance space $\mathcal{X}$ ***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 $\mathrm{x} \in \mathcal{X}$ is a feature vector
  - Each $\mathrm{x} = [\mathrm{x}_1, \mathrm{x}_2, \cdots, x_d]$ is a point in the vector space with d dimensions

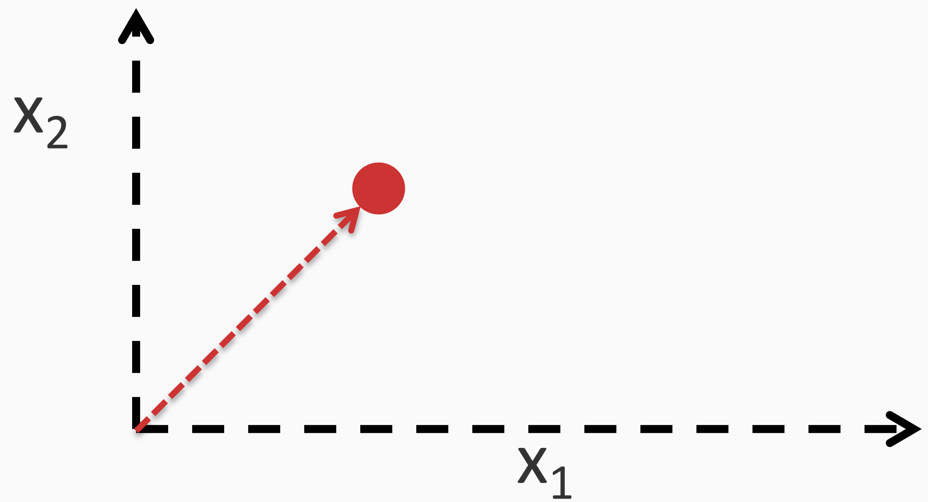


Instances as feature vectors

- Features functions convert inputs to vectors

- The instance space $\mathcal{X}$ ***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 $\mathrm{x} \in \mathcal{X}$ is a feature vector
  - Each $\mathrm{x} = [\mathrm{x}_1, \mathrm{x}_2, \cdots, x_d]$ is a point in the vector space with d dimensions



Feature functions produce feature vectors

When designing feature functions, think of them as templates

- 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"*
  - `Naoki      a` → [*1* 0 0 0 …]
  - `Abe        b` → [0 *1* 0 0 …]
  - `Manning    a` → [*1* 0 0 0 …]
  - `Scrooge    c` → [0 0 *1* 0 …]

Feature functions produce feature vectors

When designing feature functions, think of them as templates

– Feature: *"The second letter of the name"*

- `Naoki     a → [1 0 0 0 …]`
- `Abe       b → [0 1 0 0 …]`
- `Manning   a → [1 0 0 0 …]`
- `Scrooge   c → [0 0 1 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"*
  - `Naoki` a → [*1* 0 0 0 …]
  - `Abe` b → [0 *1* 0 0 …]
  - `Manning` a → [*1* 0 0 0 …]
  - `Scrooge` c → [0 0 *1* 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"*

- `Naoki` a → [*1* 0 0 0 …]
- `Abe` b → [0 *1* 0 0 …]
- `Manning` a → [*1* 0 0 0 …]
- `Scrooge` c → [0 0 *1* 0 …]

What is the dimensionality of these feature vectors?

26 (One dimension per letter)

Such vectors where exactly one dimension is 1 and all others are zero 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"*
  - `Naoki` a → [*1* 0 0 0 …]
  - `Abe` b → [0 *1* 0 0 …]
  - `Manning` a → [*1* 0 0 0 …]
  - `Scrooge` c → [0 0 *1* 0 …]
- Feature: *"The length of the name"*
  - `Naoki` → 5
  - `Abe` → 3

Feature functions produce feature vectors

When designing feature functions, think of them as templates

- Feature: *"The second letter of the name"*
  - Naoki a → [*1* 0 0 0 …]
  - Abe b → [0 *1* 0 0 …]
  - Manning a → [*1* 0 0 0 …]
  - Scrooge c → [0 0 *1* 0 …]
- Feature: *"The length of the name"*
  - Naoki → 5
  - Abe → 3
- *"The second letter of the name, Length of the first name, length of the last name"*
  - Naoki Abe → [*1* 0 0 0 … 5 3]

Features can be accumulated by concatenating the vectors

Good features are essential

- Good features decide how well a task can be learned
  - Eg: A bad feature for the badges game
    - "Is there a day of the week that begins with the last letter of the first name?"

Something to think about: Why would we think that this is a bad feature?

- Much effort goes into designing features
  - Or learning them

- Will touch upon general principles for designing good features
  - But feature definition largely domain specific
  - Comes with experience

On 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 is the learning task?

3. What is our hypothesis space?

    What functions should the learning algorithm search over?

4. What is our learning algorithm?

    How do we learn from the labeled data?

5. What is our loss function or evaluation metric?

    What is success?

2. The Label Space $\mathcal{Y}$

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

The goal of learning: Find this target function

Learning is search over functions

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {`Spam`, `Not-Spam`}, {+,−}, etc.

2. The Label Space $\mathcal{Y}$

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

The goal of learning: Find this target function

Learning is search over functions

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {`Spam`, `Not-Spam`}, {+,−}, etc.

The label space depends on the nature of the problem

Classification: The outputs are categorical

- Binary classification: Two possible labels
  - We will see a lot of this
- Multiclass classification: K possible labels
  - We may see a bit of this if time permits
- Structured classification: Graph valued outputs
  - A different class

Classification is the primary focus of this class

The label space depends on the nature of the problem

The output space can be numerical/ordinal

- Regression
  - The label space $\mathcal{Y}$ is the set (or a subset) of real numbers

- Ranking
  - Labels are ordinal
  - That is, there is an ordering over the labels
  - Eg: A Yelp 5-star review is only slightly different from a 4-star review, but very different from a 1-star review

On supervised learning

✓ What is our instance space?

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

✓ What is our label space?

What is the learning task?

3. What is our hypothesis space?

   What functions should the learning algorithm search over?

4. What is our learning algorithm?

   How do we learn from the labeled data?

5. What is our loss function or evaluation metric?

   What is success?

3. The Hypothesis Space

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function
$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

The goal of learning: Find this target function

Learning is search over functions

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {`Spam`, `Not-Spam`}, {+,−}, etc.

3. The Hypothesis Space

$\mathcal{X}$: Instance Space

The set of examples that need to be classified

Target function

$y = f(x)$

$\mathcal{Y}$: Label Space

The set of all possible labels

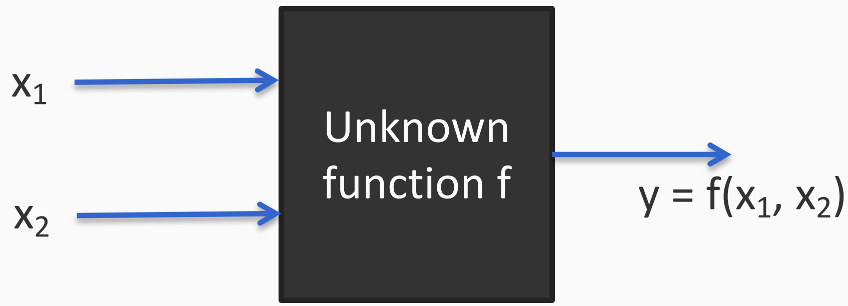
The goal of learning: Find this target function

Learning is search over functions

Eg: The set of all possible names, documents, sentences, images, emails, etc

Eg: {`Spam`, `Not-Spam`}, {+,−}, etc.

Example of search over functions



x_1	x_2	$y = f(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

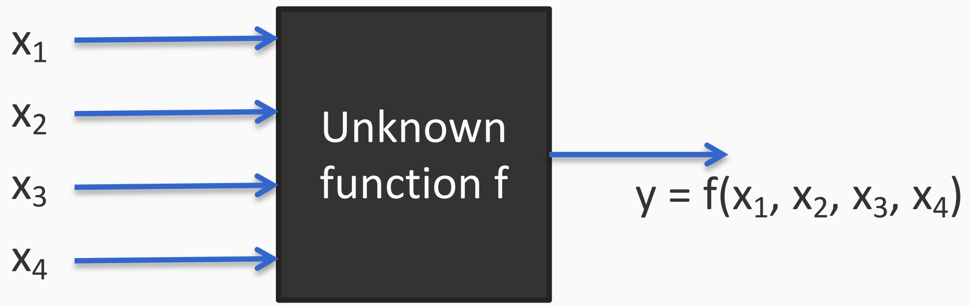
Assume that 1 stands for `True`
0 stands for `False`

Can you learn this function?

What is it?

The fundamental problem

Machine learning is ill-posed!



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?

Is learning possible at all?

There are $2^{16} = 65536$ possible Boolean functions over 4 inputs

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

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

Is learning possible at all?

There are $2^{16} = 65536$ possible Boolean functions over 4 inputs

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

We have seen only 7 outputs

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?

There are $2^{16} = 65536$ possible Boolean functions over 4 inputs

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

We have seen only 7 outputs

- *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?

There are $2^{16} = 65536$ possible Boolean functions over 4 inputs

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

We have seen only 7 outputs

How could we possibly learn anything?

- *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** ←
[illegible]	[illegible]	[illegible]	1	**0** ←
[illegible]	[illegible]	[illegible]	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!)

A *hypothesis space* is the set of possible functions we consider

- We were looking at the space of all Boolean functions
- Instead choose a hypothesis space that is not *all possible* functions
  - Only simple conjunctions (with four 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

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

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

Always False

x_1

x_2

x_3

x_4

$x_1 \wedge x_2$

$x_1 \wedge x_3$

$x_1 \wedge x_4$

Rule

$x_2 \wedge x_3$

$x_1 \wedge x_2 \wedge x_3$

$x_1 \wedge x_2 \wedge x_4$

$x_1 \wedge x_3 \wedge x_4$

$x_2 \wedge x_3 \wedge x_4$

$x_1 \wedge x_2 \wedge x_3 \wedge x_4$

Exercise: How many simple conjunctions are possible when there are n inputs instead of 4?

Example

Hypothesis space 1

Simple conjunctions

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

There are only 16 simple **conjunctive rules** of the form $g(x) = x_i \land x_j \land x_k \cdots$

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

Rule
$x_2 \land x_3$
$x_2 \land x_4$
$x_3 \land x_4$
$x_1 \land x_2 \land x_3$
$x_1 \land x_2 \land x_4$
$x_1 \land x_3 \land x_4$
$x_2 \land x_3 \land x_4$
$x_1 \land x_2 \land x_3 \land x_4$

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

Rule	Counter-example	Rule	Counter-example
Always False	1001	$x_2 \wedge x_3$	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

No simple conjunction explains the data!

(Confirm each counterexample by going through the list afterwards)

Our hypothesis space is too small and
the true function we were looking for is not in it. ☹

Solution: Restrict the search space

A *hypothesis space* is the set of possible functions we consider

- We were looking at the space of all Boolean functions
- Instead choose a hypothesis space that is not *all possible* functions
  - Only simple conjunctions (with four 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
  - …

- How do we pick a hypothesis space?
  - Using some prior knowledge (or by guessing)

Solution: Restrict the search space

A *hypothesis space* is the set of possible functions we consider

- We were looking at the space of all Boolean functions
- Instead choose a hypothesis space that is not ***all possible*** functions
  - Only simple conjunctions (with four 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
  - …

- How do we pick a hypothesis space?
  - Using some prior knowledge (or by guessing)
- What if the hypothesis space is so small that nothing in it agrees with the data?
  - We 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 = 1 if at least m of them are 1

Example:
If at least 2 of $\{x_1, x_3, x_4\}$ 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

Exercise: Check if there is one
First, how many m-of-n rules are there for four variables?

Restricting the hypothesis space

- Our guess of the hypothesis space may be incorrect

- General strategy
  - Pick an expressive hypothesis space expressing concepts
    - Concept = the target classifier that is hidden from us. Sometimes we may call it the oracle.
    - Example hypothesis spaces: m-of-n functions, decision trees, linear functions, grammars, multi-layer perceptron, transformer networks, convolutional neural networks, etc

  - Develop algorithms that find an element the hypothesis space that fits data well (or well enough)

  - Hope that it generalizes

Perspectives on learning

- Learning is the removal of ***remaining*** uncertainty over a hypothesis space
  - If we knew that the unknown function is a simple conjunction, we could use the training data to figure out which one it is

- Requires guessing a ***good, small*** hypothesis class
  - And we could be wrong
  - We could find a consistent hypothesis and still be incorrect with a new example!

On 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 is the learning task?

✓ What is our hypothesis space?

What functions should the learning algorithm search over?

4. What is our learning algorithm?

   How do we learn from the labeled data?

5. What is our loss function or evaluation metric?

   What is success?

Much of the rest of this semester