Supervised Learning: The Setup
Last lecture

• We saw
  – What is learning?
    • Learning as generalization
  – The badges game
This lecture

• More badges

• Formalizing supervised learning
  – Instance space and features
  – Label space
  – Hypothesis space
The badges game
Let’s play

<table>
<thead>
<tr>
<th>Name</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claire Cardie</td>
<td>-</td>
</tr>
<tr>
<td>Peter Bartlett</td>
<td>+</td>
</tr>
<tr>
<td>Eric Baum</td>
<td>+</td>
</tr>
<tr>
<td>Haym Hirsh</td>
<td>+</td>
</tr>
<tr>
<td>Shai Ben-David</td>
<td>+</td>
</tr>
<tr>
<td>Michael I. Jordan</td>
<td>-</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claire Cardie</td>
<td>-</td>
</tr>
<tr>
<td>Peter Bartlett</td>
<td>+</td>
</tr>
<tr>
<td>Eric Baum</td>
<td>+</td>
</tr>
<tr>
<td>Haym Hirsh</td>
<td>+</td>
</tr>
<tr>
<td>Shai Ben-David</td>
<td>+</td>
</tr>
<tr>
<td>Michael I. Jordan</td>
<td>-</td>
</tr>
</tbody>
</table>

What is the label for “Peyton Manning”? What about “Eli Manning”?

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

How were the labels generated?

<table>
<thead>
<tr>
<th>Name</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claire Cardie</td>
<td>-</td>
</tr>
<tr>
<td>Peter Bartlett</td>
<td>+</td>
</tr>
<tr>
<td>Eric Baum</td>
<td>+</td>
</tr>
<tr>
<td>Haym Hirsh</td>
<td>+</td>
</tr>
<tr>
<td>Shai Ben-David</td>
<td>+</td>
</tr>
<tr>
<td>Michael I. Jordan</td>
<td>-</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claire Cardie</td>
<td>-</td>
</tr>
<tr>
<td>Peter Bartlett</td>
<td>+</td>
</tr>
<tr>
<td>Eric Baum</td>
<td>+</td>
</tr>
<tr>
<td>Haym Hirsh</td>
<td>+</td>
</tr>
<tr>
<td>Shai Ben-David</td>
<td>+</td>
</tr>
<tr>
<td>Michael I. Jordan</td>
<td>-</td>
</tr>
</tbody>
</table>

How were the labels generated?

If length of first name $\leq 5$, then + else –

(Full data on the class website, you can stare at it longer if you want)
1. Are you sure you got the correct function?

2. How did you arrive at it?

3. Learning issues:
   - Is this prediction or just modeling data?
   - How did you know that you should look at the letters?
   - All words have a length. Background knowledge.
   - 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

A label

Sports
Instances and Labels

Running example: Automatically tag news articles

**Instance Space**: All possible news articles

**Label Space**: All possible labels

Mapped by the classifier to:

- Sports
- Business
- Politics
- Entertainment
Instances and Labels

$X$: 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**

- **$X$: Instance Space**
  - The set of examples that need to be classified

- **$Y$: Label Space**
  - The set of all possible labels

**Examples:**
- $X$: The set of all possible names, documents, sentences, images, emails, etc.
- $Y$: \{Spam, Not-Spam\}, \{+, -\}, etc.
Instances and Labels

\( X: \text{Instance Space} \)

The set of examples that need to be classified

\( Y: \text{Label Space} \)

The set of all possible labels

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

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

Target function

\[ y = f(x) \]
Instances and Labels

\[ X: \text{Instance Space} \]

The set of examples that need to be classified

\[ y = f(x) \]

Target function

\[ Y: \text{Label Space} \]

The set of all possible labels

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

\text{Learning is search over functions}

\text{The goal of learning: Find this target function}

\text{Eg: The set of all possible names, documents, sentences, images, emails, etc}
Supervised learning

**$X$: Instance Space**
The set of examples

**$Y$: Label Space**
The set of all possible labels

**Target function**

\[ y = f(x) \]

Learning algorithm only sees examples of the function $f$ in action
Supervised learning: Training

$X$: Instance Space
The set of examples

$Y$: Label Space
The set of all possible labels

Target function
$y = f(x)$

Learning algorithm only sees examples of the function $f$ in action

Labeled training data
**Supervised learning: Training**

- **$X$: Instance Space**
  - The set of examples

- **$Y$: Label Space**
  - The set of all possible labels

Target function: $y = f(x)$

Learning algorithm only sees examples of the function $f$ in action

Labeled training data:

1. $(x_1, f(x_1))$
2. $(x_2, f(x_2))$
3. $(x_3, f(x_3))$
4. \vdots
5. $(x_N, f(x_N))$
Supervised learning: Training

- **$X$: Instance Space**
  - The set of examples

- **$Y$: Label Space**
  - The set of all possible labels

**Target function** $y = f(x)$

Learning algorithm only sees examples of the function $f$ in action

Labeled training data

A learned function $g: X \rightarrow Y$
Supervised learning: Training

**Instance Space**

- The set of examples
- \( X \) = set of examples

**Label Space**

- The set of all possible labels
- \( Y \) = set of possible labels

**Target function**

\[ y = f(x) \]

\[ (x_1, f(x_1)) \]
\[ (x_2, f(x_2)) \]
\[ (x_3, f(x_3)) \]
\[ \vdots \]
\[ (x_N, f(x_N)) \]

**Learning algorithm**

Labeled training data

A learned function \( g: X \rightarrow Y \)

Can you think of other training protocols?
Supervised learning: Evaluation

X: Instance Space
The set of examples

\( y = f(x) \)
Target function

Y: Label Space
The set of all possible labels

\( y = g(x) \)
Learned function

\( X \): Instance Space

\( Y \): Label Space

The set of examples

The set of all possible labels
Supervised learning: Evaluation

\[ y = f(x) \]

Target function

\[ y = g(x) \]

Learned function

\[ f(x) \]

\[ g(x) \]

Are they different?

\[ X: \text{Instance Space} \]

The set of examples

\[ Y: \text{Label Space} \]

The set of all possible labels

Draw test example \( x \in X \)
Supervised learning: Evaluation

\[ X: \text{Instance Space} \]
\[ Y: \text{Label Space} \]

The set of examples

The set of all possible labels

Target function
\[ y = f(x) \]

Learned function
\[ y = g(x) \]

Draw test example \( x \in X \)

Are they different?

Apply the model to many test examples and compare to the target’s prediction
Supervised learning: Evaluation

\( X: \text{Instance Space} \)

The set of examples

\( Y: \text{Label Space} \)

The set of all possible labels

Target function

\[ y = f(x) \]

Learned function

\[ y = g(x) \]

Draw test example \( x \in X \)

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

Can you use test examples during training?
Supervised learning: General setting

• Given: Training examples of the form \(<x, f(x)>\)
  – The function \(f\) is an unknown function

• Typically the input \(x\) is represented in a feature space
  – Example: \(x \in \{0,1\}^n\) or \(x \in \mathbb{R}^n\)
  – A deterministic mapping from instances in your problem (e.g., news articles) to features

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

• Goal: Find a good approximation for \(f\)

• The label determines the kind of problem we have
  – Binary classification: \(f(x) \in \{-1,1\}\)
  – Multiclass classification: \(f(x) \in \{1, 2, 3, \ldots, K\}\)
  – Regression: \(f(x) \in \mathbb{R}\)

Questions?
Nature of applications

• There is no human expert
  – Eg: Identify DNA binding sites

• Humans can perform a task, but can’t describe how they do it
  – Eg: Object detection in images

• The desired function is hard to obtain in closed form
  – Eg: Stock market
Binary classification

Where 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?

• Malware detection
  – Is an Android app malicious?

• Time series prediction
  – Will the future value of a stock increase or decrease with respect to its current value?
On using 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 \( X \)

\( X \): Instance Space

The set of examples

\( Y \): Label Space

The set of all possible labels

\( y = f(x) \)

Target function

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 $X$

$X$: Instance Space

The set of examples

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

$Y$: Label Space

The set of all possible labels

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

Target function $y = f(x)$

The goal of learning: Find this target function

Learning is search over functions

Designing an appropriate feature representation of the instance space is crucial

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

Example: Boolean features
• Does the email contain the word “free”?

Example: Real valued features
• What is the height of the person?
• What was the stock price yesterday?
1. The Instance Space $X$

Let’s brainstorm some features for the badges game
Instances as feature vectors

An input to the problem (Eg: emails, names, images) → Feature function → A feature vector
Instances as feature vectors

Feature functions a.k.a feature extractors

- Deterministic (for the most part)
- Convert the examples a collection of attributes
  - Very often easy to think of them as vectors
- Important part of the design of a learning based solution
Instances as feature vectors

- Features functions convert inputs to vectors
  - Fixed mapping
- The instance space $X$ is a $N$-dimensional vector space (e.g. $\mathbb{R}^N$ or $\{0,1\}^N$)
  - Each dimension is one feature
- Each $x \in X$ is a feature vector
  - Each $x = [x_1, x_2, \ldots, x_N]$ is a point in the vector space
Instances as feature vectors

- Features functions convert inputs to vectors
  - Fixed mapping
- The instance space $X$ is a $N$-dimensional vector space (e.g. $\mathbb{R}^N$ or $\{0,1\}^N$)
  - Each dimension is one feature
- Each $x \in X$ is a feature vector
  - Each $x = [x_1, x_2, \ldots, x_N]$ is a point in the vector space
Instances as feature vectors

- Features functions convert inputs to vectors
  - Fixed mapping
- The instance space $X$ is a N-dimensional vector space (e.g. $\mathbb{R}^N$ or $\{0,1\}^N$)
  - Each dimension is one feature
- Each $\mathbf{x} \in X$ is a feature vector
  - Each $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ is a point in the vector space
Feature functions produce feature vectors

When designing feature functions, think of them as templates

– Feature: “The second letter of the name”
  - Naoki: \(a \rightarrow [1 \ 0 \ 0 \ 0 \ ...]\)
  - Abe: \(b \rightarrow [0 \ 1 \ 0 \ 0 \ ...]\)
  - Manning: \(a \rightarrow [1 \ 0 \ 0 \ 0 \ ...]\)
  - Scrooge: \(c \rightarrow [0 \ 0 \ 1 \ 0 \ ...]\)
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 …]

Question: What is the length of this feature vector?
Feature functions produce feature vectors

When designing feature functions, think of them as templates

– Feature: “The second letter of the name”

• Naoki \( a \rightarrow [1 \ 0 \ 0 \ 0 \ ...] \)  
  Question: What is the length of this feature vector?

• Abe \( b \rightarrow [0 \ 1 \ 0 \ 0 \ ...] \)  

• Manning \( a \rightarrow [1 \ 0 \ 0 \ 0 \ ...] \)  

• Scrooge \( c \rightarrow [0 \ 0 \ 1 \ 0 \ ...] \)  

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 ...]  Question: What is the length of this feature vector?
  • Abe    b → [0 1 0 0 ...]
  • Manning a → [1 0 0 0 ...]  26 (One dimension per letter)
  • Scrooge c → [0 0 1 0 ...]

– Feature: “The length of the name”
  • Naoki → 5
  • Abe   → 3
Good features are essential

• Good features decide how well a task can be learned
  – Eg: A bad feature function the badges game
    • “Is there a day of the week that begins with the last letter of the first name?”

• Much effort goes into designing features
  – Or maybe learning them

• Will touch upon general principles for designing good features
  – But feature definition largely domain specific
  – Comes with experience
On using supervised learning

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 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 $Y$

---

**X: Instance Space**

The set of examples

**Y: Label Space**

The set of all possible labels

**Target function**

$y = f(x)$

**The goal of learning:** Find this target function

*Learning is search over functions*

---

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

Eg: {Spam, Not-Spam}, {+, -}, etc.
2. The Label Space $Y$

X: Instance Space
The set of examples

$y = f(x)$
Target function

Y: Label Space
The set of all possible labels

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

Eg: {Spam, Not-Spam}, {+,-}, etc.
2. The Label Space $Y$

*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

- **Structured** classification: Graph valued outputs
  - A different class

Classification is the primary focus of this class
2. The Label Space $Y$

- The output space can be numerical
  - **Regression:**
    - $Y$ 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 using supervised learning

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 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**

**X: Instance Space**
- The set of examples
  - Eg: The set of all possible names, sentences, images, emails, etc.

**Y: Label Space**
- The set of all possible labels
  - Eg: {Spam, Not-Spam}, {+, -}, etc.

**Target function**
\[ y = f(x) \]

**The goal of learning:** Find this target function

*Learning is search over functions*
3. The **Hypothesis Space**

**X: Instance Space**
The set of examples

**Y: Label Space**
The set of all possible labels

The goal of learning: Find this target function

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

Learning is search over functions
Example of search over functions

Can you learn this function?

What is it?
The fundamental problem: Machine learning is ill-posed!

Unknown function

\[ y = f(x_1, x_2, x_3, x_4) \]

Can you learn this function?

What is it?

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 ←</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 ←</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 ←</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 ←</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 ←</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 ←</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 ←</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>
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 smaller than the space of all functions
  - Only *simple conjunctions* (with four variables, there are only 16 conjunctions without negations)
  - *Simple disjunctions*
  - *m-of-n rules*: Fix a set of $n$ variables. At least $m$ of them must be true
  - *Linear functions*
Hypothesis space 1

Simple conjunctions

There are only 16 simple conjunctive rules of the form $g(x) = x_i \land x_j \land x_k$
Hypothesis space 1

Simple conjunctions

There are only 16 simple *conjunctive rules* of the form \( g(x) = x_i \wedge x_j \wedge x_k \)
Hypothesis space 1

Simple conjunctions

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always False</td>
<td>( x_2 \land x_3 )</td>
</tr>
<tr>
<td>( x_1 )</td>
<td>( x_2 \land x_4 )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( x_3 \land x_4 )</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>( x_1 \land x_2 \land x_3 )</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>( x_1 \land x_2 \land x_4 )</td>
</tr>
<tr>
<td>( x_1 \land x_2 )</td>
<td>( x_1 \land x_3 \land x_4 )</td>
</tr>
<tr>
<td>( x_1 \land x_3 )</td>
<td>( x_2 \land x_3 \land x_4 )</td>
</tr>
<tr>
<td>( x_1 \land x_4 )</td>
<td>( x_1 \land x_2 \land x_3 \land x_4 )</td>
</tr>
</tbody>
</table>
Example

Hypothesis space 1

Simple conjunctions

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always False</td>
<td>$x_2 \land x_3$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_1 \land x_2 \land x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_1 \land x_2 \land x_4$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_1 \land x_3 \land x_4$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2 \land x_3 \land x_4$</td>
</tr>
<tr>
<td>$x_1 \land x_2$</td>
<td>$x_1 \land x_2 \land x_3 \land x_4$</td>
</tr>
</tbody>
</table>

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

Hypothesis space 1

Simple conjunctions

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always False</td>
<td></td>
</tr>
<tr>
<td>( x_1 )</td>
<td>( x_2 \land x_3 )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( x_2 \land x_4 )</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>( x_3 \land x_4 )</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>( x_1 \land x_2 \land x_3 )</td>
</tr>
<tr>
<td>( x_1 \land x_2 )</td>
<td>( x_1 \land x_2 \land x_4 )</td>
</tr>
<tr>
<td>( x_1 \land x_3 )</td>
<td>( x_1 \land x_3 \land x_4 )</td>
</tr>
<tr>
<td>( x_1 \land x_4 )</td>
<td>( x_2 \land x_3 \land x_4 )</td>
</tr>
</tbody>
</table>

Is there a consistent hypothesis in this space?

<table>
<thead>
<tr>
<th></th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

Hypothesis space 1

Simple conjunctions

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Counter-example</th>
<th>Rule</th>
<th>Counter-example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always False</td>
<td>1001</td>
<td>( x_2 \land x_3 )</td>
<td>0011</td>
</tr>
<tr>
<td>( x_1 )</td>
<td>1100</td>
<td>( x_2 \land x_4 )</td>
<td>0011</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0100</td>
<td>( x_3 \land x_4 )</td>
<td>1001</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0110</td>
<td>( x_1 \land x_2 \land x_3 )</td>
<td>0011</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>0101</td>
<td>( x_1 \land x_2 \land x_4 )</td>
<td>0011</td>
</tr>
<tr>
<td>( x_1 \land x_2 )</td>
<td>1100</td>
<td>( x_1 \land x_3 \land x_4 )</td>
<td>0011</td>
</tr>
<tr>
<td>( x_1 \land x_3 )</td>
<td>0011</td>
<td>( x_2 \land x_3 \land x_4 )</td>
<td>0011</td>
</tr>
<tr>
<td>( x_1 \land x_4 )</td>
<td>0011</td>
<td>( x_1 \land x_2 \land x_3 \land x_4 )</td>
<td>0011</td>
</tr>
</tbody>
</table>
Example

Hypothesis space 1

Simple conjunctions

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Counter-example</th>
<th>Rule</th>
<th>Counter-example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always False</td>
<td>1001</td>
<td>$x_1 \land x_2 \land x_4$</td>
<td>0011</td>
</tr>
<tr>
<td>$x_1$</td>
<td>0110</td>
<td>$x_1 \land x_2 \land x_3$</td>
<td>0011</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0101</td>
<td>$x_1 \land x_2 \land x_4$</td>
<td>0011</td>
</tr>
<tr>
<td>$x_3$</td>
<td>1100</td>
<td>$x_1 \land x_3 \land x_4$</td>
<td>0011</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0011</td>
<td>$x_2 \land x_3 \land x_4$</td>
<td>0011</td>
</tr>
<tr>
<td>$x_1 \land x_2$</td>
<td>0011</td>
<td>$x_1 \land x_2 \land x_3 \land x_4$</td>
<td>0011</td>
</tr>
</tbody>
</table>

**No simple conjunction explains the data!**

Our hypothesis space is too small
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 smaller than the space of all 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*

• 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
**Hypothesis space 2**

**m-of-n rules**

Pick a subset with $n$ variables. $Y = 1$ 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.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Is there a consistent hypothesis in this space?

Try to 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 even call it the oracle.
    • Example hypothesis spaces: m-of-n functions, decision trees, linear functions, grammars, multi-layer deep networks, etc
  
  – Develop algorithms that find an element the hypothesis space that fits data well (or well enough)
  
  – Hope that it generalizes
Views of learning

• Learning is the removal of **remaining** uncertainty
  – 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?