Bayesian Networks

+ 

Reinforcement Learning:

Markov Decision Processes
Reminders

- Homework 7: HMMs
  - Out: Fri, Apr. 1
  - Due: Tue, Apr. 12 at 11:59pm
GRAPHICAL MODELS:
DETERMNING CONDITIONAL INDEPENDENCIES
What Independencies does a Bayes Net Model?

- In order for a Bayesian network to model a probability distribution, the following must be true:
  Each variable is conditionally independent of all its non-descendants in the graph given the value of all its parents.

- This follows from
  \[ P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t)) \]
  \[ = \prod_{t=1}^{T} P(X_t \mid X_1, \ldots, X_{t-1}) \]

- But what else does it imply?
What Independencies does a Bayes Net Model?

Three cases of interest…

Cascade

Common Parent

V-Structure
What Independencies does a Bayes Net Model?

Three cases of interest...

- **Cascade**
  - $X \perp Z \mid Y$

- **Common Parent**
  - $X \perp Z \mid Y$

- **V-Structure**
  - $X \not\perp Z \mid Y$

Knowing $Y$ **decouples** $X$ and $Z$ for the Cascade and Common Parent cases. Knowing $Y$ **couples** $X$ and $Z$ for the V-Structure case.
Whiteboard

Proof of conditional independence

(The other two cases can be shown just as easily.)

\[ X \perp Z \mid Y \]
The “Burglar Alarm” example

- Your house has a twitchy burglar alarm that is also sometimes triggered by earthquakes.
- Earth arguably doesn’t care whether your house is currently being burgled.
- While you are on vacation, one of your neighbors calls and tells you your home’s burglar alarm is ringing. Uh oh!

Quiz: True or False?

\[ \text{Burglar} \perp \text{Earthquake} \mid \text{Phone Call} \]
The “Burglar Alarm” example

• After you get this phone call, suppose you learn that there was a medium-sized earthquake in your neighborhood. Oh, whew! Probably not a burglar after all.

• Earthquake “explains away” the hypothetical burglar.

• But then it must **not** be the case that

\[
\text{Burglar} \vDash \text{Earthquake} \mid \text{PhoneCall}
\]

even though

\[
\text{Burglar} \vDash \text{Earthquake}
\]
Markov boundary

Def: the co-parents of a node are the parents of its children.

Def: the Markov boundary of a node is the set containing the node’s parents, children, and co-parents.
Markov boundary

**Def:** the **co-parents** of a node are the parents of its children.

**Def:** the **Markov boundary** of a node is the set containing the node’s parents, children, and co-parents.

**Example:** The Markov boundary of $X_6$ is

$$\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$$
**Markov boundary**

**Def:** the co-parents of a node are the parents of its children.

**Def:** the Markov boundary of a node is the set containing the node’s parents, children, and co-parents.

**Theorem:** a node is conditionally independent of every other node in the graph given its Markov boundary.

**Example:** The Markov boundary of $X_6$ is \{ $X_3$, $X_4$, $X_5$, $X_8$, $X_9$, $X_{10}$ \}
**D-Separation**

**Definition #1:** Variables X and Z are d-separated given a set of evidence variables E (variables that are observed) iff every path from X to Z is “blocked”.

A path is “blocked” whenever:

1. \( \exists Y \text{ on path s.t. } Y \in E \text{ and } Y \text{ is a “common parent”} \)

   ![Diagram 1: Common Parent](image)

2. \( \exists Y \text{ on path s.t. } Y \in E \text{ and } Y \text{ is in a “cascade”} \)

   ![Diagram 2: Cascade](image)

3. \( \exists Y \text{ on path s.t. } \{Y, \text{ descendants}(Y)\} \notin E \text{ and } Y \text{ is in a “v-structure”} \)

   ![Diagram 3: V-Structure](image)

If variables X and Z are d-separated given a set of variables E

Then X and Z are conditionally independent given the set E
**D-Separation**

If variables X and Z are **d-separated** given a set of variables E
Then X and Z are **conditionally independent** given the set E

**Definition #2:**
Variables X and Z are **d-separated** given a set of evidence variables E iff there does not exist a path between X and Z in the **undirected ancestral moral graph** with E removed.

1. **Ancestral graph:** keep only X, Z, E and their ancestors
2. **Moral graph:** add undirected edge between all pairs of each node’s parents
3. **Undirected graph:** convert all directed edges to undirected
4. **Givens Removed:** delete any nodes in E

**Example Query:**  \( A \perp B \mid \{D, E\} \)

- **Original:**
  - A
  - B
  - C
  - D
  - E
  - F
- **Ancestral:**
  - A
  - B
  - C
  - D
  - E
- **Moral:**
  - A
  - B
  - C
  - D
  - E
- **Undirected:**
  - A
  - B
  - C
  - D
  - E
- **Givens Removed:**
  - A
  - B
  - C

\( \Rightarrow \text{A and B connected} \Rightarrow \text{not d-separated} \)
SUPERVISED LEARNING FOR
BAYES NETS
Recipe for Closed-form MLE

1. Assume data was generated i.i.d. from some model (i.e. write the generative story)
   \[ x^{(i)} \sim p(x|\theta) \]

2. Write log-likelihood
   \[ l(\theta) = \log p(x^{(1)}|\theta) + \ldots + \log p(x^{(N)}|\theta) \]

3. Compute partial derivatives (i.e. gradient)
   \[ \frac{\partial l(\theta)}{\partial \theta_1} = \ldots \]
   \[ \frac{\partial l(\theta)}{\partial \theta_2} = \ldots \]
   \[ \ldots \]
   \[ \frac{\partial l(\theta)}{\partial \theta_M} = \ldots \]

4. Set derivatives to zero and solve for \( \theta \)
   \[ \frac{\partial l(\theta)}{\partial \theta_m} = 0 \text{ for all } m \in \{1, \ldots, M\} \]
   \[ \theta^{\text{MLE}} = \text{solution to system of } M \text{ equations and } M \text{ variables} \]

5. Compute the second derivative and check that \( l(\theta) \) is concave down at \( \theta^{\text{MLE}} \)
Machine Learning

The **data** inspires the structures we want to predict.

Our **model** defines a score for each structure.

It also tells us what to optimize.

**Inference** finds \{best structure, marginals, partition function\} for a new observation.

**Learning** tunes the parameters of the model.

*(Inference is usually called as a subroutine in learning)*
Machine Learning

Data

Inference

(Inference is usually called as a subroutine in learning)

Model

Objective

Learning
Learning Fully Observed BNs

\[ p(X_1, X_2, X_3, X_4, X_5) = \]
\[ p(X_5|X_3)p(X_4|X_2, X_3) \]
\[ p(X_3)p(X_2|X_1)p(X_1) \]
Learning Fully Observed BNs

\[ p(X_1, X_2, X_3, X_4, X_5) = p(X_5|X_3)p(X_4|X_2, X_3)p(X_3)p(X_2|X_1)p(X_1) \]
Learning Fully Observed BNs

\[ p(X_1, X_2, X_3, X_4, X_5) = \]
\[ p(X_5|X_3)p(X_4|X_2, X_3) \]
\[ p(X_3)p(X_2|X_1)p(X_1) \]

How do we learn these conditional and marginal distributions for a Bayes Net?
Learning Fully Observed BNs

Learning this fully observed Bayesian Network is **equivalent** to learning five (small / simple) independent networks from the same data.

\[
p(X_1, X_2, X_3, X_4, X_5) = p(X_5|X_3)p(X_4|X_2, X_3)p(X_3)p(X_2|X_1)p(X_1)
\]
Learning Fully Observed BNs

How do we learn these conditional and marginal distributions for a Bayes Net?

\[
\theta^* = \underset{\theta}{\text{argmax}} \log p(X_1, X_2, X_3, X_4, X_5)
\]

\[
= \underset{\theta}{\text{argmax}} \log p(X_5|X_3, \theta_5) + \log p(X_4|X_2, X_3, \theta_4)
+ \log p(X_3|\theta_3) + \log p(X_2|X_1, \theta_2)
+ \log p(X_1|\theta_1)
\]

\[
\theta_1^* = \underset{\theta_1}{\text{argmax}} \log p(X_1|\theta_1)
\]

\[
\theta_2^* = \underset{\theta_2}{\text{argmax}} \log p(X_2|X_1, \theta_2)
\]

\[
\theta_3^* = \underset{\theta_3}{\text{argmax}} \log p(X_3|\theta_3)
\]

\[
\theta_4^* = \underset{\theta_4}{\text{argmax}} \log p(X_4|X_2, X_3, \theta_4)
\]

\[
\theta_5^* = \underset{\theta_5}{\text{argmax}} \log p(X_5|X_3, \theta_5)
\]
Example: Tornado Alarms

1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?
Example: Tornado Alarms

1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?

Figure from https://www.nytimes.com/2017/04/08/us/dallas-emergency-sirens-hacking.html

*Warning sirens in Dallas, meant to alert the public to emergencies like severe weather, started sounding around 11:40 p.m. Friday, and were not shut off until 1:20 a.m. Rex C. Curry for The New York Times*
Learning Fully Observed BNs

Example: Tornado Alarms

H ~ Bernoulli (η)
T ~ Bernoulli (ε)
A ~ Bernoulli (α_{H,T})
C ~ Uniform([1,...,63]) + A \times Uniform([1,...,63])

Data:

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>H</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

MLE's in Closed Form

\[ \ell(\eta, \varepsilon, \alpha) = \frac{12}{N} \sum_{i=1}^{N} \left( \log p(t^{(i)} | h^{(i)} , a^{(i)} , c^{(i)}) n y^{(i)} k^{(i)} \right) \]

\[ \hat{\eta}, \hat{\varepsilon}, \hat{\alpha} = \arg \max \ell(\eta, \varepsilon, \alpha) \]

\[ \tilde{\eta} = \arg \max_{\eta} \frac{12}{N} \sum_{i=1}^{N} \log p(h^{(i)} | \eta) = \frac{\#(T=1)}{N} \]

\[ \tilde{\varepsilon} = \arg \max_{\varepsilon} \frac{12}{N} \sum_{i=1}^{N} \log p(t^{(i)} | \eta) = \frac{\#(H=1)}{N} \]

\[ \tilde{\alpha} = \arg \max_{\alpha} \frac{12}{N} \sum_{i=1}^{N} \log p(a^{(i)} | h^{(i)} , t^{(i)} , c^{(i)}) \]

\[ \tilde{\alpha}_{ht} = \frac{\#(A+1,T=t,H=h)}{\#(T=t,H=h)} \]

No parameters

Parameters

What are the MLEs?

\[ \hat{\eta} = \frac{1}{2} \]

\[ \hat{\varepsilon} = \frac{1}{2} \]

\[ \hat{\alpha} = \frac{1}{3} \]

\[ \hat{\alpha}_{ht} = \frac{1}{3} \]
INFERENCE FOR BAYESIAN NETWORKS
A Few Problems for Bayes Nets

Suppose we already have the parameters of a Bayesian Network...

1. How do we compute the probability of a specific assignment to the variables?  
   \[ P(T=t, H=h, A=a, C=c) \]

2. How do we draw a sample from the joint distribution?  
   \[ t, h, a, c \sim P(T, H, A, C) \]

3. How do we compute marginal probabilities?  
   \[ P(A) = \ldots \]

4. How do we draw samples from a conditional distribution?  
   \[ t, h, a \sim P(T, H, A | C = c) \]

5. How do we compute conditional marginal probabilities?  
   \[ P(H | C = c) = \ldots \]

Can we use samples?
Gibbs Sampling

\[ p(x) \]

\[ p(x_1 | x_2(t)) \]

\[ x(t+1) \]

\[ x(t) \]
Gibbs Sampling

\[ p(x) \]

\[ p(x_2 | x_1^{(t+1)}) \]

\[ x_1 \]

\[ x_2 \]

\( x^{(t+2)} \)

\( x^{(t+1)} \)

\( x^{(t)} \)
Carlo sampling is a method for generating samples from a probability distribution. It is particularly useful in Bayesian inference where the posterior distribution is complex and difficult to sample from directly. The method is based on the Metropolis-Hastings algorithm and is a form of Markov chain Monte Carlo (MCMC) method.

In Gibbs sampling, one-variable-at-a-time, the algorithm works as follows:

1. **Initialization**: Start the algorithm with an initial state \( x^{(0)} \).
2. **Iteration**: For each variable \( x_j \) in the order \( j = 1, 2, \ldots, D \), where \( D \) is the number of variables, do the following:
   a. **Conditional Sampling**: Sample \( x_j^{(t+1)} \) from the conditional distribution of \( x_j \) given the previous values of all other variables, i.e., \( x_j^{(t+1)} \sim p(x_j | x_j^{(t-1)}, \ldots, x_j^{(0)}) \).
   b. **Update**: Set the new state \( x^{(t+1)} = (x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_D^{(t+1)}) \).
3. **Repeat**: Go back to step 2 for the next iteration until the desired number of samples is obtained.

The advantage of Gibbs sampling is that it requires only the conditional distributions of the variables, which are usually easier to sample from than the joint distribution. It is particularly useful when the conditional distributions are known and easy to sample from, but the joint distribution is difficult to work with.
Gibbs Sampling

**Question:** How do we draw samples from a conditional distribution?

\[ y_1, y_2, \ldots, y_J \sim p(y_1, y_2, \ldots, y_J \mid x_1, x_2, \ldots, x_J) \]

**(Approximate) Solution:**
- Initialize \( y_1^{(0)}, y_2^{(0)}, \ldots, y_J^{(0)} \) to arbitrary values
- For \( t = 1, 2, \ldots \):
  - \( y_1^{(t+1)} \sim p(y_1 \mid y_2^{(t)}, \ldots, y_J^{(t)}, x_1, x_2, \ldots, x_J) \)
  - \( y_2^{(t+1)} \sim p(y_2 \mid y_1^{(t+1)}, y_3^{(t)}, \ldots, y_J^{(t)}, x_1, x_2, \ldots, x_J) \)
  - \( y_3^{(t+1)} \sim p(y_3 \mid y_1^{(t+1)}, y_2^{(t+1)}, y_4^{(t)}, \ldots, y_J^{(t)}, x_1, x_2, \ldots, x_J) \)
  - \( \ldots \)
  - \( y_J^{(t+1)} \sim p(y_J \mid y_1^{(t+1)}, y_2^{(t+1)}, \ldots, y_{J-1}^{(t+1)}, x_1, x_2, \ldots, x_J) \)

**Properties:**
- This will eventually yield samples from \( p(y_1, y_2, \ldots, y_J \mid x_1, x_2, \ldots, x_J) \)
- But it might take a long time -- just like other Markov Chain Monte Carlo methods
Gibbs Sampling

Full conditionals only need to condition on the Markov boundary

- Must be “easy” to sample from conditionals
- Many conditionals are log-concave and are amenable to adaptive rejection sampling
Learning Objectives

Bayesian Networks

You should be able to...

1. Identify the conditional independence assumptions given by a generative story or a specification of a joint distribution
2. Draw a Bayesian network given a set of conditional independence assumptions
3. Define the joint distribution specified by a Bayesian network
4. User domain knowledge to construct a (simple) Bayesian network for a real-world modeling problem
5. Depict familiar models as Bayesian networks
6. Use d-separation to prove the existence of conditional independencies in a Bayesian network
7. Employ a Markov boundary to identify conditional independence assumptions of a graphical model
8. Develop a supervised learning algorithm for a Bayesian network
9. Use samples from a joint distribution to compute marginal probabilities
10. Sample from the joint distribution specified by a generative story
11. Implement a Gibbs sampler for a Bayesian network
LEARNING PARADIGMS
# Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$\mathcal{D} = {x^{(i)}, y^{(i)}}_{i=1}^N$  \hspace{1cm} $x \sim p^<em>(\cdot)$ and $y = c^</em>(\cdot)$</td>
</tr>
<tr>
<td>$\rightarrow$ Regression</td>
<td>$y^{(i)} \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\rightarrow$ Classification</td>
<td>$y^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>$\rightarrow$ Binary classification</td>
<td>$y^{(i)} \in {+1, -1}$</td>
</tr>
<tr>
<td>$\rightarrow$ Structured Prediction</td>
<td>$y^{(i)}$ is a vector</td>
</tr>
</tbody>
</table>
# Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$\mathcal{D} = {x^{(i)}, y^{(i)}}_{i=1}^{N}$ \quad $x \sim p^<em>(\cdot)$ and $y = c^</em>(\cdot)$</td>
</tr>
<tr>
<td>$\rightarrow$ Regression</td>
<td>$y^{(i)} \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\rightarrow$ Classification</td>
<td>$y^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>$\rightarrow$ Binary classification</td>
<td>$y^{(i)} \in {+1, -1}$</td>
</tr>
<tr>
<td>$\rightarrow$ Structured Prediction</td>
<td>$y^{(i)}$ is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>$\mathcal{D} = {x^{(i)}}_{i=1}^{N}$ \quad $x \sim p^*(\cdot)$</td>
</tr>
</tbody>
</table>
# Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>( \mathcal{D} = { \mathbf{x}^{(i)}, y^{(i)} }_{i=1}^{N} ) ( \mathbf{x} \sim p^<em>(\cdot) ) and ( y = c^</em>(\cdot) )</td>
</tr>
<tr>
<td>( \mapsto ) Regression</td>
<td>( y^{(i)} \in \mathbb{R} )</td>
</tr>
<tr>
<td>( \mapsto ) Classification</td>
<td>( y^{(i)} \in {1, \ldots, K} )</td>
</tr>
<tr>
<td>( \mapsto ) Binary classification</td>
<td>( y^{(i)} \in {+1, -1} )</td>
</tr>
<tr>
<td>( \mapsto ) Structured Prediction</td>
<td>( y^{(i)} ) is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>( \mathcal{D} = { \mathbf{x}^{(i)} }_{i=1}^{N} ) ( \mathbf{x} \sim p^*(\cdot) )</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>( \mathcal{D} = { \mathbf{x}^{(i)}, y^{(i)} }<em>{i=1}^{N_1} \cup { \mathbf{x}^{(j)} }</em>{j=1}^{N_2} )</td>
</tr>
</tbody>
</table>
## Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>( \mathcal{D} = { \mathbf{x}^{(i)}, y^{(i)} }_{i=1}^{N} \quad \mathbf{x} \sim p^<em>(\cdot) ) and ( y = c^</em>(\cdot) )</td>
</tr>
<tr>
<td>↔ Regression</td>
<td>( y^{(i)} \in \mathbb{R} )</td>
</tr>
<tr>
<td>↔ Classification</td>
<td>( y^{(i)} \in {1, \ldots, K} )</td>
</tr>
<tr>
<td>↔ Binary classification</td>
<td>( y^{(i)} \in {+1, -1} )</td>
</tr>
<tr>
<td>↔ Structured Prediction</td>
<td>( y^{(i)} ) is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>( \mathcal{D} = { \mathbf{x}^{(i)} }_{i=1}^{N} \quad \mathbf{x} \sim p^*(\cdot) )</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>( \mathcal{D} = { \mathbf{x}^{(i)}, y^{(i)} }<em>{i=1}^{N_1} \cup { \mathbf{x}^{(j)} }</em>{j=1}^{N_2} )</td>
</tr>
<tr>
<td>Online</td>
<td>( \mathcal{D} = {(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \ldots} )</td>
</tr>
</tbody>
</table>
# Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$\mathcal{D} = { \mathbf{x}^{(i)}, y^{(i)} }_{i=1}^{N}$ $\mathbf{x} \sim p^<em>(\cdot)$ and $y = c^</em>(\cdot)$</td>
</tr>
<tr>
<td>$\rightarrow$ Regression</td>
<td>$y^{(i)} \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\rightarrow$ Classification</td>
<td>$y^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>$\rightarrow$ Binary classification</td>
<td>$y^{(i)} \in {+1, -1}$</td>
</tr>
<tr>
<td>$\rightarrow$ Structured Prediction</td>
<td>$y^{(i)}$ is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>$\mathcal{D} = { \mathbf{x}^{(i)} }_{i=1}^{N}$ $\mathbf{x} \sim p^*(\cdot)$</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>$\mathcal{D} = { \mathbf{x}^{(i)}, y^{(i)} }<em>{i=1}^{N_1} \cup { \mathbf{x}^{(j)} }</em>{j=1}^{N_2}$</td>
</tr>
<tr>
<td>Online</td>
<td>$\mathcal{D} = { (\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \ldots }$</td>
</tr>
<tr>
<td>Active Learning</td>
<td>$\mathcal{D} = { \mathbf{x}^{(i)} }_{i=1}^{N}$ and can query $y^{(i)} = c^*(\cdot)$ at a cost</td>
</tr>
</tbody>
</table>
# Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$\mathcal{D} = {x^{(i)}, y^{(i)}}_{i=1}^N$ \hspace{1cm} $x \sim p^<em>(\cdot)$ and $y = c^</em>(\cdot)$</td>
</tr>
<tr>
<td>$\leftarrow$ Regression</td>
<td>$y^{(i)} \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\leftarrow$ Classification</td>
<td>$y^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>$\leftarrow$ Binary classification</td>
<td>$y^{(i)} \in {-1, +1}$</td>
</tr>
<tr>
<td>$\leftarrow$ Structured Prediction</td>
<td>$y^{(i)}$ is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>$\mathcal{D} = {x^{(i)}}_{i=1}^N$ \hspace{1cm} $x \sim p^*(\cdot)$</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>$\mathcal{D} = {x^{(i)}, y^{(i)}}<em>{i=1}^{N_1} \cup {x^{(j)}}</em>{j=1}^{N_2}$</td>
</tr>
<tr>
<td>Online</td>
<td>$\mathcal{D} = {(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots}$</td>
</tr>
<tr>
<td>Active Learning</td>
<td>$\mathcal{D} = {x^{(i)}}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost</td>
</tr>
<tr>
<td>Imitation Learning</td>
<td>$\mathcal{D} = {(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \ldots}$</td>
</tr>
</tbody>
</table>
## Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$\mathcal{D} = {\mathbf{x}^{(i)}, y^{(i)}}_{i=1}^N$ $\mathbf{x} \sim p^<em>(\cdot)$ and $y = c^</em>(\cdot)$</td>
</tr>
<tr>
<td>Regression</td>
<td>$y^{(i)} \in \mathbb{R}$</td>
</tr>
<tr>
<td>Classification</td>
<td>$y^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>Binary classification</td>
<td>$y^{(i)} \in {+1, -1}$</td>
</tr>
<tr>
<td>Structured Prediction</td>
<td>$y^{(i)}$ is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>$\mathcal{D} = {\mathbf{x}^{(i)}}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>$\mathcal{D} = {\mathbf{x}^{(i)}, y^{(i)}}<em>{i=1}^{N_1} \cup {\mathbf{x}^{(j)}}</em>{j=1}^{N_2}$</td>
</tr>
<tr>
<td>Online</td>
<td>$\mathcal{D} = {(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \ldots}$</td>
</tr>
<tr>
<td>Active Learning</td>
<td>$\mathcal{D} = {\mathbf{x}^{(i)}}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost</td>
</tr>
<tr>
<td>Imitation Learning</td>
<td>$\mathcal{D} = {(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \ldots}$</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td>$\mathcal{D} = {(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \ldots}$</td>
</tr>
</tbody>
</table>
REINFORCEMENT LEARNING
Reinforcement Learning

Source: https://www.xkcd.com/242/
RL: Examples

Source: https://twitter.com/alphagomovie

Source: https://techobserver.net/2019/06/argo-ai-self-driving-car-research-center/
Source: https://www.wired.com/2012/02/high-speed-trading/
AlphaGo

Source: https://www.youtube.com/watch?v=WXuK6gekU1Y&ab_channel=DeepMind
History of Reinforcement Learning

• Roots in the psychology of animal learning (Thorndike, 1911).

• Another independent thread was the problem of optimal control, and its solution using dynamic programming (Bellman, 1957).

• Idea of temporal difference learning (on-line method), e.g., playing board games (Samuel, 1959).

• A major breakthrough was the discovery of Q-learning (Watkins, 1989).
What is special about RL?

• RL is learning how to map states to actions, so as to maximize a numerical reward over time.

• Unlike other forms of learning, it is a multistage decision-making process (often Markovian).

• An RL agent must learn by trial-and-error. (Not entirely supervised, but interactive)

• Actions may affect not only the immediate reward but also subsequent rewards (Delayed effect).
Elements of RL

• A policy
  - A map from state space to action space.
  - May be stochastic.

• A reward function
  - It maps each state (or, state-action pair) to a real number, called reward.

• A value function
  - Value of a state (or, state-action pair) is the total expected reward, starting from that state (or, state-action pair).
Example: Robot in a Room

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
Question: Is this policy optimal: yes or no? Briefly justify your answer.

Answer: (Hint: both yes and no are acceptable answers, I’m interested in your justification.)
Example: Robot in a Room

- Reward for each step -2
Example: Robot in a Room

• Reward for each step: -0.1
The Precise Goal

• To find a policy that maximizes the Value function.
  – transitions and rewards usually not available

• There are different approaches to achieve this goal in various situations.

• Value iteration and Policy iteration are two more classic approaches to this problem. But essentially both are dynamic programming.

• Q-learning is a more recent approaches to this problem. Essentially it is a temporal-difference method.