Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 5/10: Learning, Markov Decision Processes, and Decision Network

DR VARUN OJHA

Department of Computer Science

University of Reading

Learning objectives

- By the end of this week, you will be able to
- Learn Bayesian Classifier
- Markov Chain and Markova Decision Process
- Valuer Function and Optima Policy Design
- Decision Network
- Apply concept of Bayesian classify to two or more objects.

Content of this week

- Part 1: Basics of Bayesian Theorem
- Part 2: Naïve Bayesian Classifier (NBC)
    - Discrete Values Attributes
    - Continuous Values Attributes
- Part 3: Markov Decision Process (MDP)
    - Markov Chain
    - Value Function
    - Policy design
- Part 4: Decision Network (DN)
- Part 5: Practical Exercise (NBC)
- Quiz

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 5/10: Learning (Algorithms)

Part 1

Bayesian Theorem

DR VARUN OJHA

Department of Computer Science

University of Reading

Probability

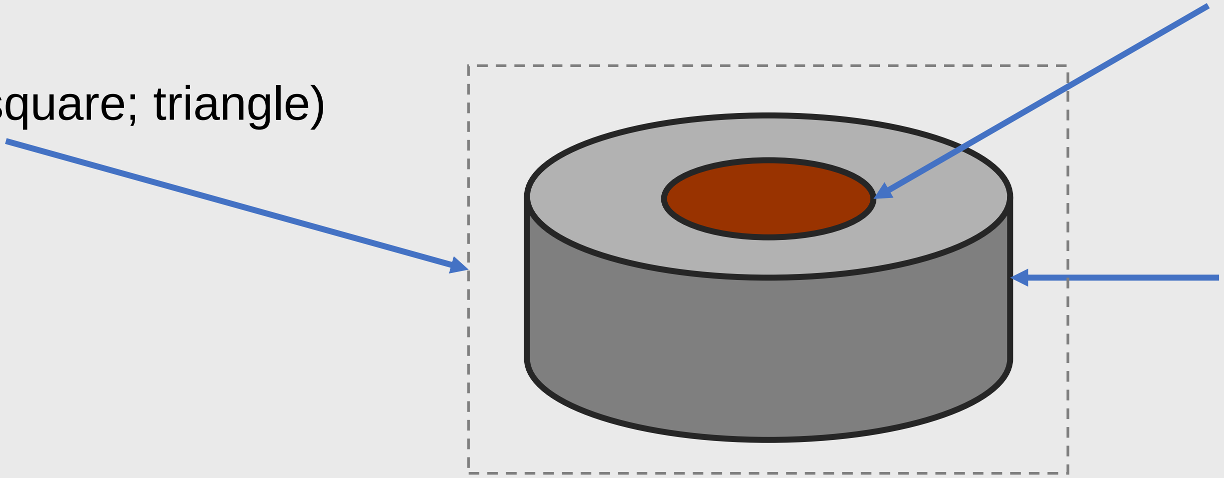
Probability of choosing either 0 or 1

$$P\,(\text{either } 0 \text{ or } 1) = \frac{1}{2}$$

Probability of choosing either 1 and 10

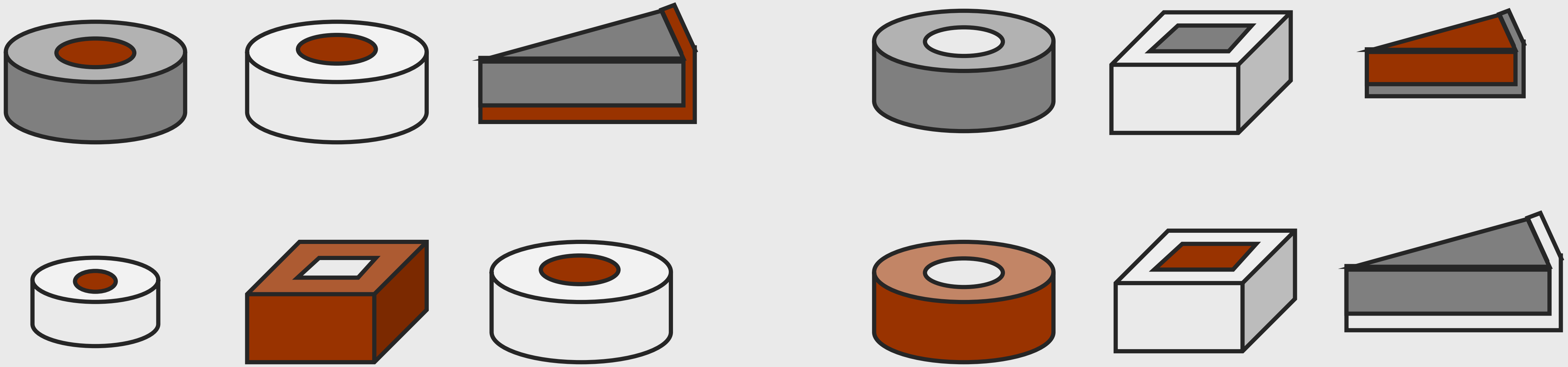
$$P\,(\text{EVEN number between 1 and 10}) = \frac{5}{10}$$

Training data



Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

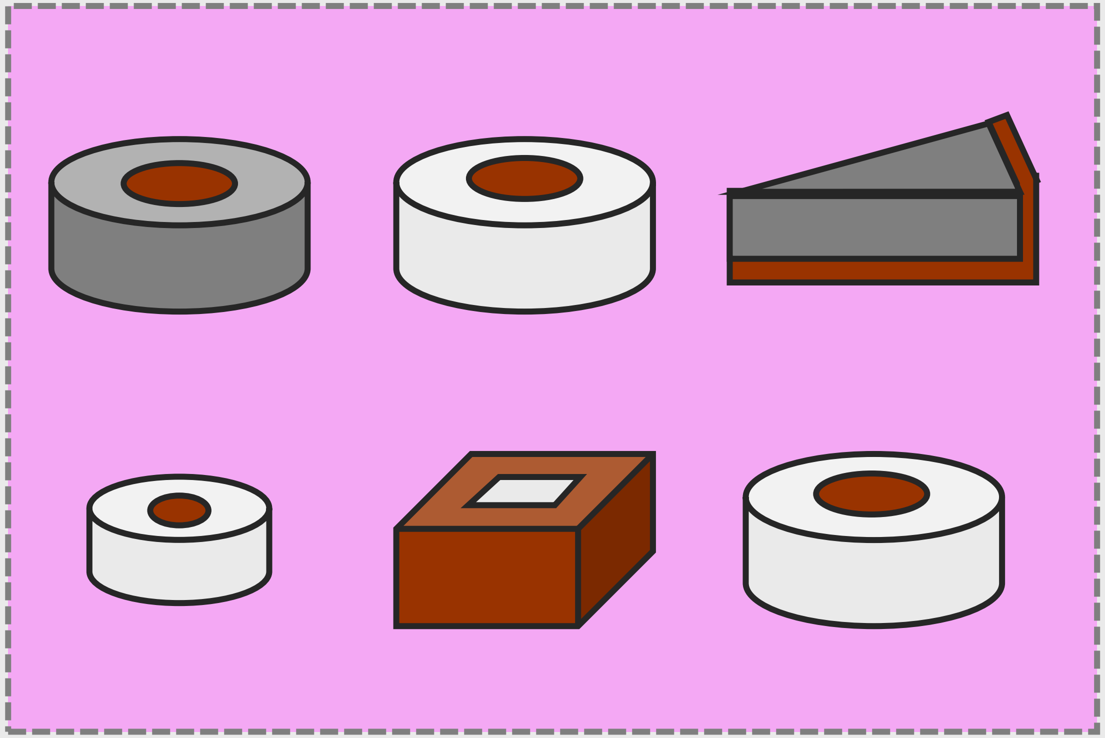
Training data



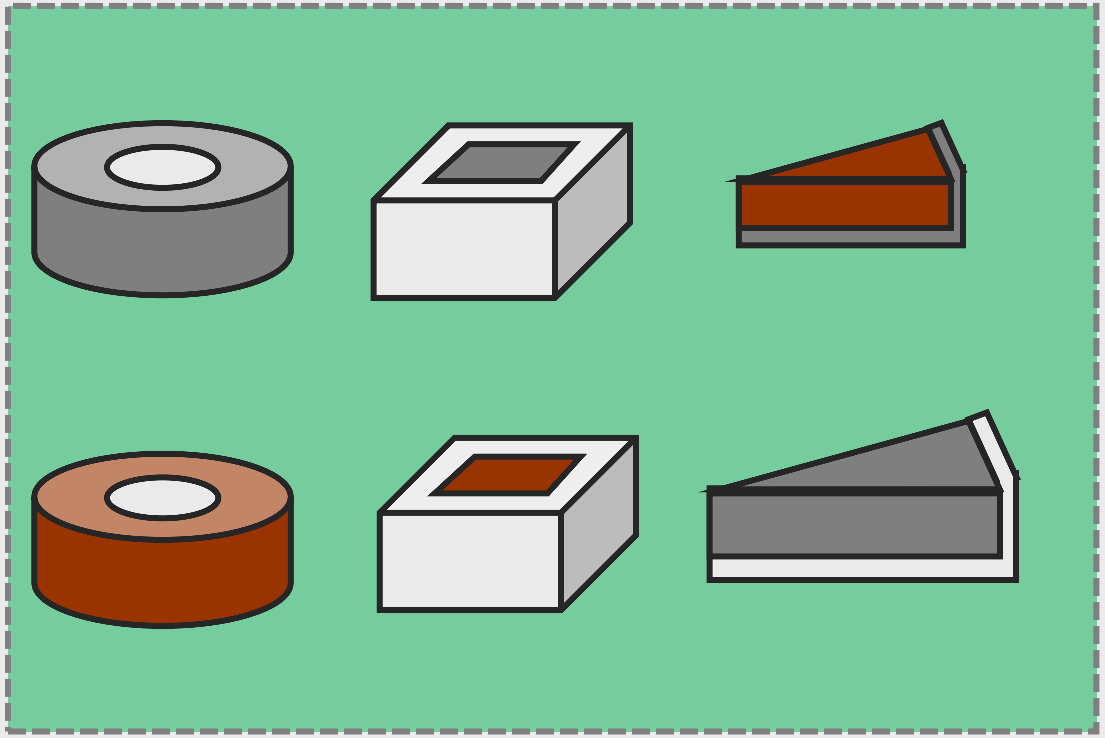
Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

Training data

I **LIKE** these types of cake



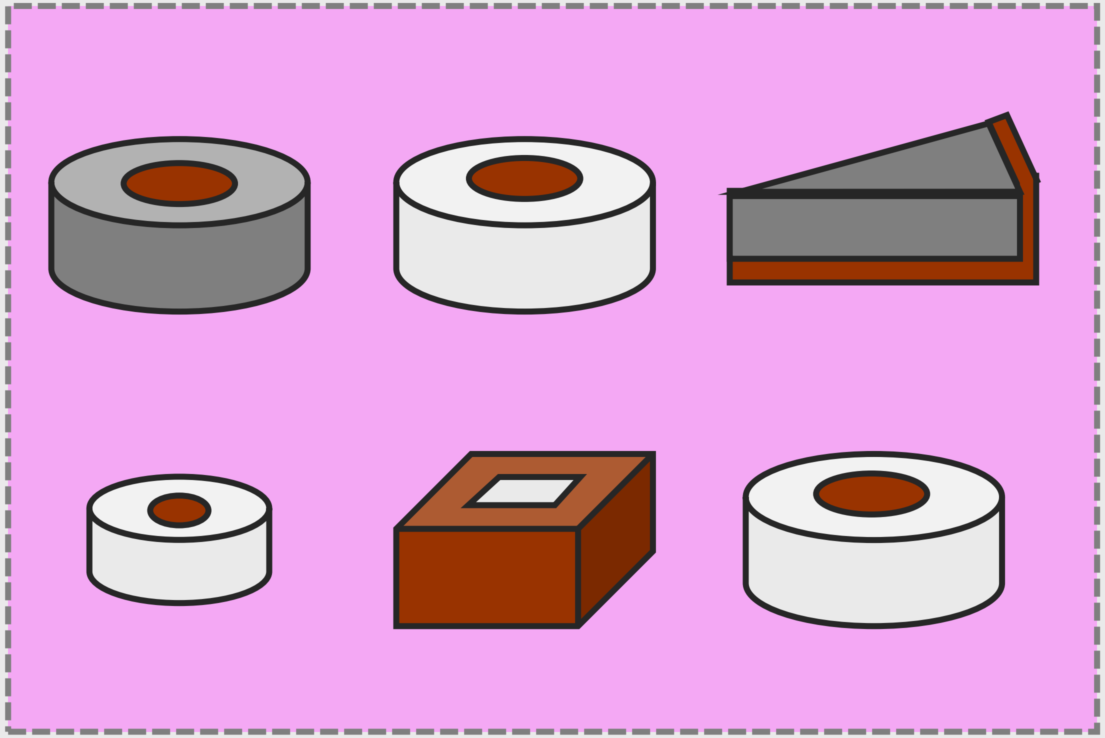
I **DO NOT LIKE** these types of cake



Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

Training data: Positive example

I **LIKE** these types of cake

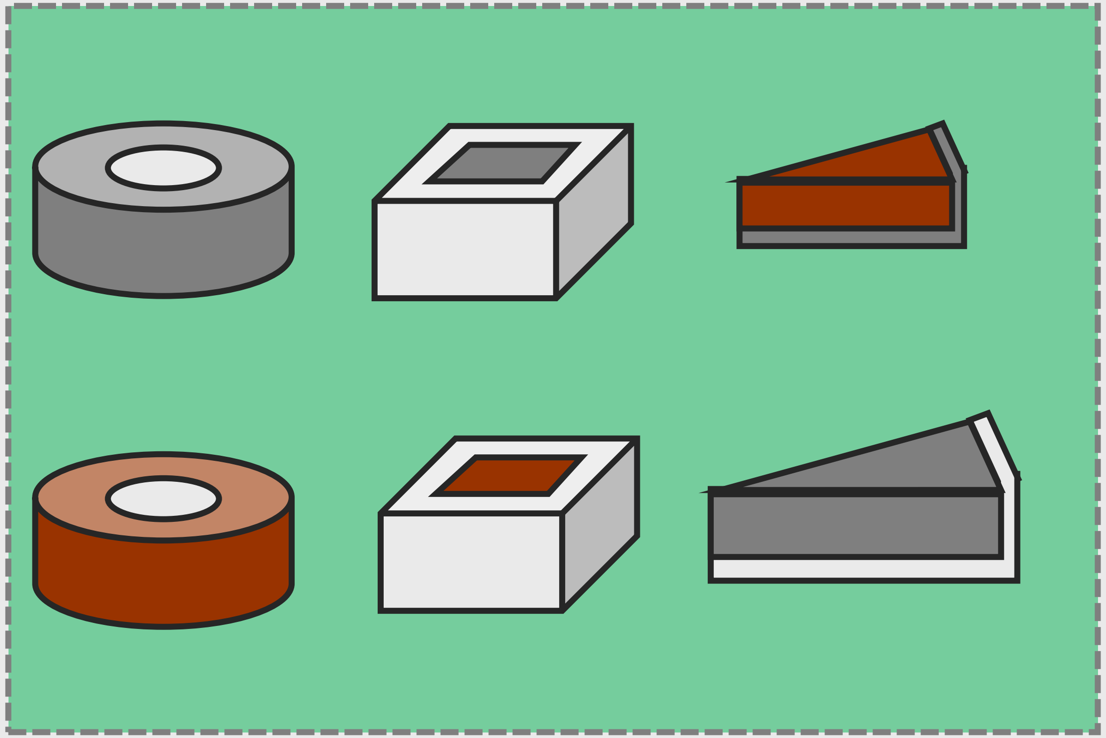


Example	Shape	Crust		Filling		Class
		Size	Shade	Size	Shade	
Ex1	Circle	Thick	Grey	Thick	Dark	Pos
Ex2	Circle	Thick	White	Thick	Dark	Pos
Ex3	Triangle	Thick	Dark	Thick	Grey	Pos
Ex4	Circle	Thin	White	Thin	Dark	Pos
Ex5	Square	Thick	Dark	Thin	White	Pos
Ex6	Circle	Thick	White	Thin	Dark	Pos

Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

Training data: Negative example

I **DO NOT LIKE** these types of cake



Example	Shape	Crust		Filling		Class
		Size	Shade	Size	Shade	
Ex7	Circle	Thick	Grey	Thick	White	Neg
Ex8	Square	Thick	White	Thick	Grey	Neg
Ex9	Triangle	Thin	Grey	Thin	Dark	Neg
Ex10	Circle	Thick	Dark	Thin	White	Neg
Ex11	Square	Thick	White	Thick	Dark	Neg
Ex12	Triangle	Thick	White	Thick	Grey	Neg

Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

Training data: All examples

#		Crust		Filling		
	Shape	Size	Shade	Size	Shade	Class
Ex1	Circle	Thick	Grey	Thick	Dark	Pos
Ex2	Circle	Thick	White	Thick	Dark	Pos
Ex3	Triangle	Thick	Dark	Thick	Grey	Pos
Ex4	Circle	Thin	White	Thin	Dark	Pos
Ex5	Square	Thick	Dark	Thin	White	Pos
Ex6	Circle	Thick	White	Thin	Dark	Pos
Ex7	Circle	Thick	Grey	Thick	White	Neg
Ex8	Square	Thick	White	Thick	Grey	Neg
Ex9	Triangle	Thin	Grey	Thin	Dark	Neg
Ex10	Circle	Thick	Dark	Thick	White	Neg
Ex11	Square	Thick	White	Thick	Dark	Neg
Ex12	Triangle	Thick	White	Thick	Grey	Neg

Instance space

$$|\text{Shape}| \times |\text{Crust}_{\text{size}}| \times |\text{Crust}_{\text{shape}}| \times |\text{Fill}_{\text{size}}| \times |\text{Fill}_{\text{shade}}$$

$$3 \times 2 \times 3 \times 2 \times 3$$

$$108$$

Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

Bayes Theorem

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)}$$

Thomas Bayes
(1701 – 1761)

Probability picking a “pos” example randomly?

$$P(\text{pos}) = \frac{N_{\text{pos examples}}}{N_{\text{all examples}}} = \frac{6}{12} = 0.5$$

The Prior Probability:

probability of picking a “pos” example randomly?

$$P(\text{pos}) = \frac{N_{\text{pos examples}}}{N_{\text{all examples}}} = \frac{6}{12} = 0.5$$

The Conditional Probability:

The probability of picking a “pos” from all “thick filling” example randomly?

$$P(\text{pos} \mid \text{thick}) = \frac{N_{\text{pos} \mid \text{thick}}}{N_{\text{thick}}} = \frac{3}{8} = 0.375$$

The Conditional Probability:

The probability of picking a “thick filling” from all “pos” example randomly?

$$P(\text{thick} \mid \text{pos}) = \frac{N_{\text{thick} \mid \text{pos}}}{N_{\text{pos}}} = \frac{3}{6} = 0.5$$

The Joint Probability:

The probability of picking a “pos” **and** “thick filling” example randomly?

$$P(\text{pos}\,,\text{thick}) = P(\text{pos} \mid \text{thick}).\,P(\text{thick})$$

$$= \frac{3}{8} \cdot \frac{8}{12} = \frac{3}{12}$$

The Joint Probability:

The probability of picking a "thick filling" **and** "pos" example randomly?

$$P(\text{thick}, \text{pos}) = P(\text{thick} \mid \text{pos}).P(\text{pos})$$

$$= \frac{3}{6} \cdot \frac{6}{12} = \frac{3}{12}$$

The Joint Probability:
Two important things

$$P(\text{pos}, \text{thick}) \leq P(\text{pos} \mid \text{thick})$$

Joint probability of two events will always be ≤ their conditional probability

$$P(\text{pos}, \text{thick}) = P(\text{thick}, \text{pos})$$

The Posterior Probability:

$$P(\text{pos} \mid \text{thick}) = \frac{P(\text{ thick} \mid \text{pos})P(\text{pos})}{P(\text{ thick })}$$

The Posterior Probability:

Bayes Theorem

$$P(\text{pos} \mid \text{thick}) = \frac{P(\text{thick} \mid \text{pos})P(\text{pos})}{P(\text{thick})}$$

Bayes Theorem

likelihood

prior

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)}$$

posterior

normalising constant

Where H and E are events

$P(H \mid E)$ is a conditional probability, the likelihood of H given E is true.

$P(E \mid H)$ is a conditional probability, the likelihood of E given H is true.

$P(H)$ and $P(E)$ are probabilities of observing H and E

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 5/10: Learning (Algorithms)

Part 2

Bayesian Classifier

DR VARUN OJHA

Department of Computer Science

University of Reading

Training data: All examples

#		Crust		Filling		
	Shape	Size	Shade	Size	Shade	Class
Ex1	Circle	Thick	Grey	Thick	Dark	Pos
Ex2	Circle	Thick	White	Thick	Dark	Pos
Ex3	Triangle	Thick	Dark	Thick	Grey	Pos
Ex4	Circle	Thin	White	Thin	Dark	Pos
Ex5	Square	Thick	Dark	Thin	White	Pos
Ex6	Circle	Thick	White	Thin	Dark	Pos
Ex7	Circle	Thick	Grey	Thick	White	Neg
Ex8	Square	Thick	White	Thick	Grey	Neg
Ex9	Triangle	Thin	Grey	Thin	Dark	Neg
Ex10	Circle	Thick	Dark	Thick	White	Neg
Ex11	Square	Thick	White	Thick	Dark	Neg
Ex12	Triangle	Thick	White	Thick	Grey	Neg

Instance space

$$|\text{Shape}| \times |\text{Crust}_{\text{size}}| \times |\text{Crust}_{\text{shape}}| \times |\text{Fill}_{\text{size}}| \times |\text{Fill}_{\text{shade}}|$$

$$3 \times 2 \times 3 \times 2 \times 3$$

$$108$$

Example Source: Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.

The Posterior Probability:

The posterior probability of a class given input data

$$P(\text{class} \mid \text{data}) = \frac{P(\text{data} \mid \text{class})P(\text{class})}{P(\text{data})}$$

The Posterior Probability:

The posterior probability of a class c_j given an input vector $\mathbf{x}$

$$P(c_j \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c_j)P(c_j)}{P(\mathbf{x})}$$

Where $\mathbf{x} = \langle \boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n} \rangle$ is a data and $c_j = f(\mathbf{x})$, class label, e.g., $c_1 = pos$ and $c_2 = neg$

The Posterior Probability:

The posterior probability of a class c_j given an input vector $\mathbf{x}$

$$P(c_j \mid \mathbf{x}) = P(\mathbf{x} \mid c_j)P(c_j)$$

Where $\mathbf{x} = \langle \boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n} \rangle$ is a data and $c_i = f(\mathbf{x})$, class label

Since $P(\mathbf{x})$ being same for all classes in question, we will label data with class which maximises the numerator $P(\mathbf{x} \mid c_j)P(c_j)$

The Prior Probability $P(c_j)$ is easy!

$$P(c_j) = \frac{N_{\text{examples of class labled } c_j}}{N_{\text{all examples}}}$$

The Prior Probability $P(\mathbf{x} \mid c_j)$ is hard!

$$P(\mathbf{x} \mid c_j) = \frac{N_{\text{examples represent vector } \mathbf{x} \text{ where class is } c_j}}{N_{\text{all examples labled as class } c_j}}$$

The Prior Probability $P(\mathbf{x} \mid c_j)$ is hard!

Instance space can be huge!

What if the vector $\mathbf{x}$ does not belong to the training set?

$$P(\mathbf{x} \mid c_j) = \frac{N_{\text{examples represent vector } \mathbf{x} \text{ where class is } c_j}}{N_{\text{all examples labled as class } c_j}}$$

X = | Triangle Thick Grey Thin Grey |

This example does **NOT** belong to the training data

The Prior Probability $P(\mathbf{x} \mid c_j)$ is hard!

Instance – space can be huge!

What if the vector $\mathbf{x}$ does not belong to the training set?

$$P(\mathbf{x} \mid c_j) = 0$$

Bayes formula will give **0**

Hope! Try individual attributes

Assumption! Mutually independent attributes

$$P(\mathbf{x} \mid c_j) = \prod_{i=1}^{n} P(x_i \mid c_j)$$

Where $\mathbf{x} = \langle \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n} \rangle$ is a data and $c_j = f(\mathbf{x})$, class label

The Posterior Probability:

The posterior probability of a class given input data vector

$$P(c_j \mid \mathbf{x}) = P(c_j) \cdot \prod_{i=1}^{n} P(x_i \mid c_j)$$

Since $P(\mathbf{x})$ being same for all classes in question, we will label data with class which maximises the numerator $P(\mathbf{x} \mid c_j)P(c_j)$.

Because we make a naïve assumption

Naïve Bayes Classifier (NBC):

Maximum a Posteriori (MAP)

$$\hat{y} = \underset{c_j \in \{1,2,\ldots,k\}}{\operatorname{argmax}} P(c_j) \cdot \prod_{i=1}^{n} P(x_i \mid c_j)$$

Because we make a naïve assumption

6:56 PM

Naïve Bayes Classifier (NBC):

Maximum a Posteriori (MAP) using **log likelihood**

$$\hat{y} = \underset{c_j \in \{1,2,\ldots,k\}}{\operatorname{argmax}} \log\left(P(c_j)\right) \sum_{i=1}^{n} \log\left(P(x_i \mid c_j)\right)$$

Because we make a naïve assumption

Naïve Bayes Classifier:

Assuming a **uniform** prior $P(c_j)$ over the hypothesis space, MAP reduces to Maximum Likelihood learning:

$$\hat{y} = \underset{c_j \in \{1,2,\ldots,k\}}{\operatorname{argmax}} \prod_{i=1}^{n} P(x_i \mid c_j)$$

Maximum Likelihood learning :

Assuming a **uniform** prior $P(c_j)$ over the hypothesis space, MAP reduces to Maximum Likelihood learning:

$$\hat{y} = \underset{c_j \in \{1,2,\ldots,k\}}{\operatorname{argmax}} \prod_{i=1}^{n} P(\, x_i \mid c_j)$$

Bayesian Classifier

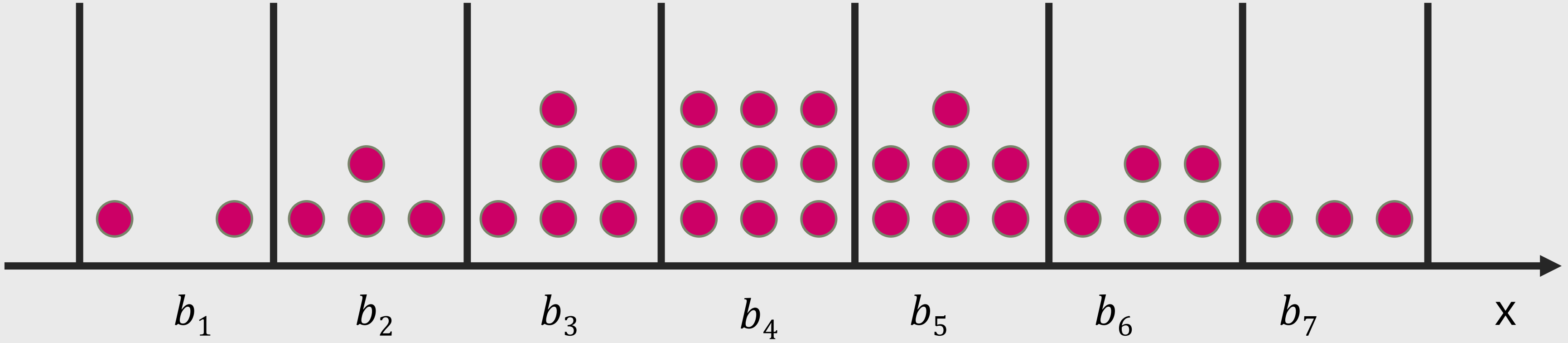
Continuous domain

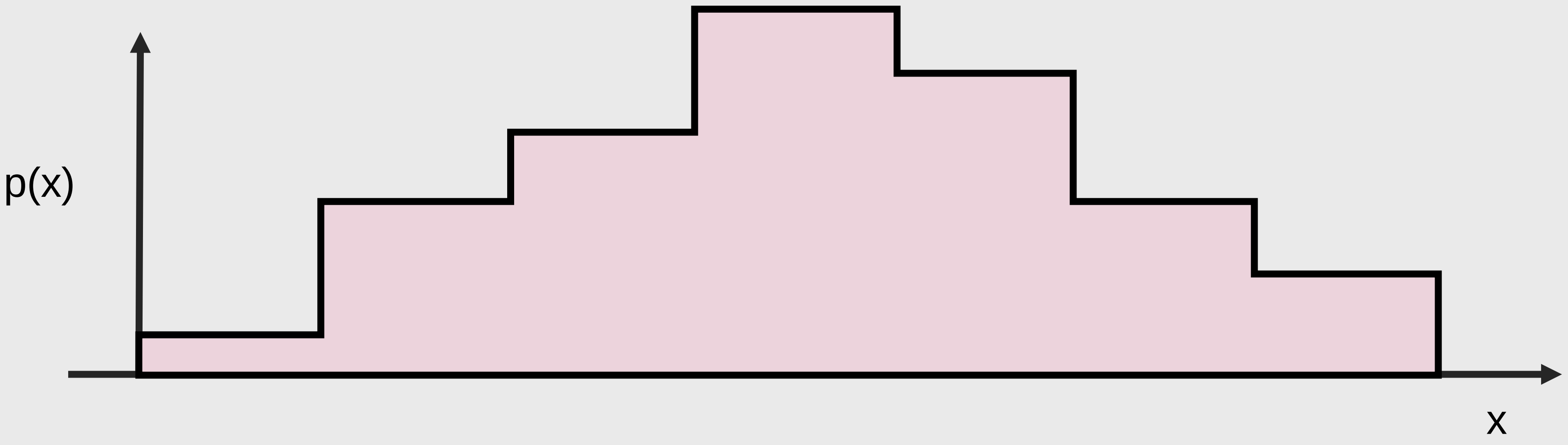
Does NBC also work for continuous attributes?

If we try to find frequency of mutually independent attribute x_i which is a *real number* and not discrete, the Prior Probability $P(\mathbf{x} \mid c_j)$ is super hard to find in a continuous domain.

Instance-space is too vast!

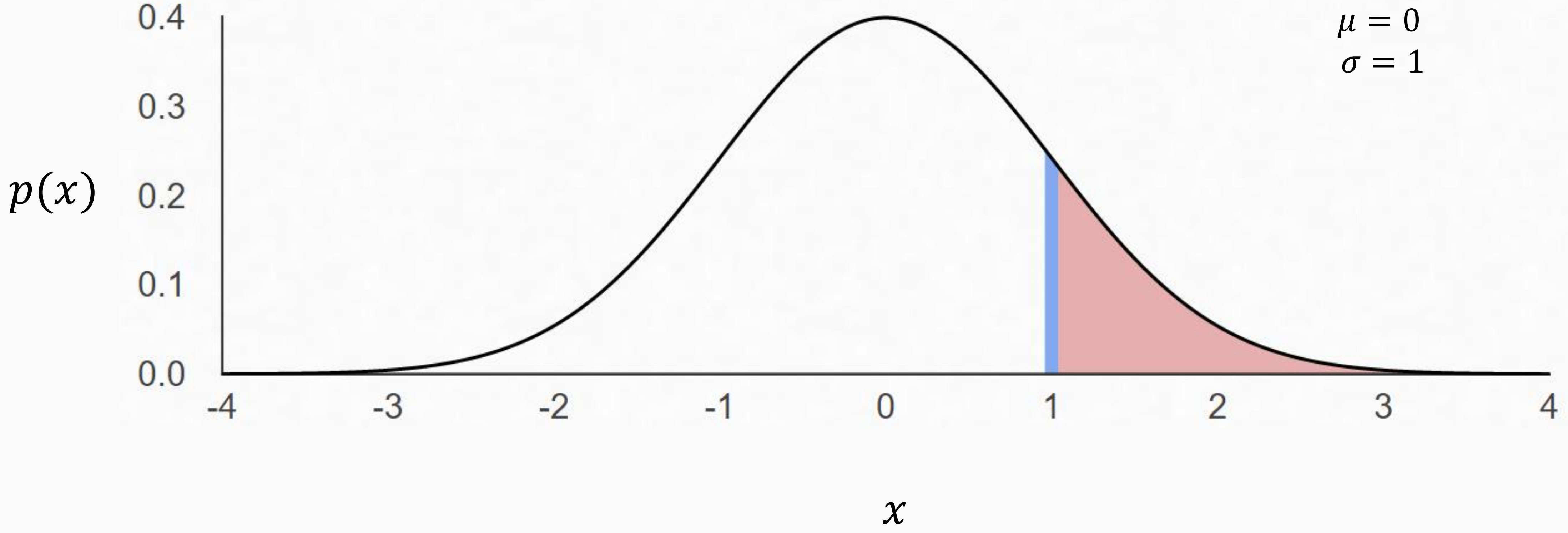
Binning of Continues Variables





Probability Density Function (pdf)

Gaussian function



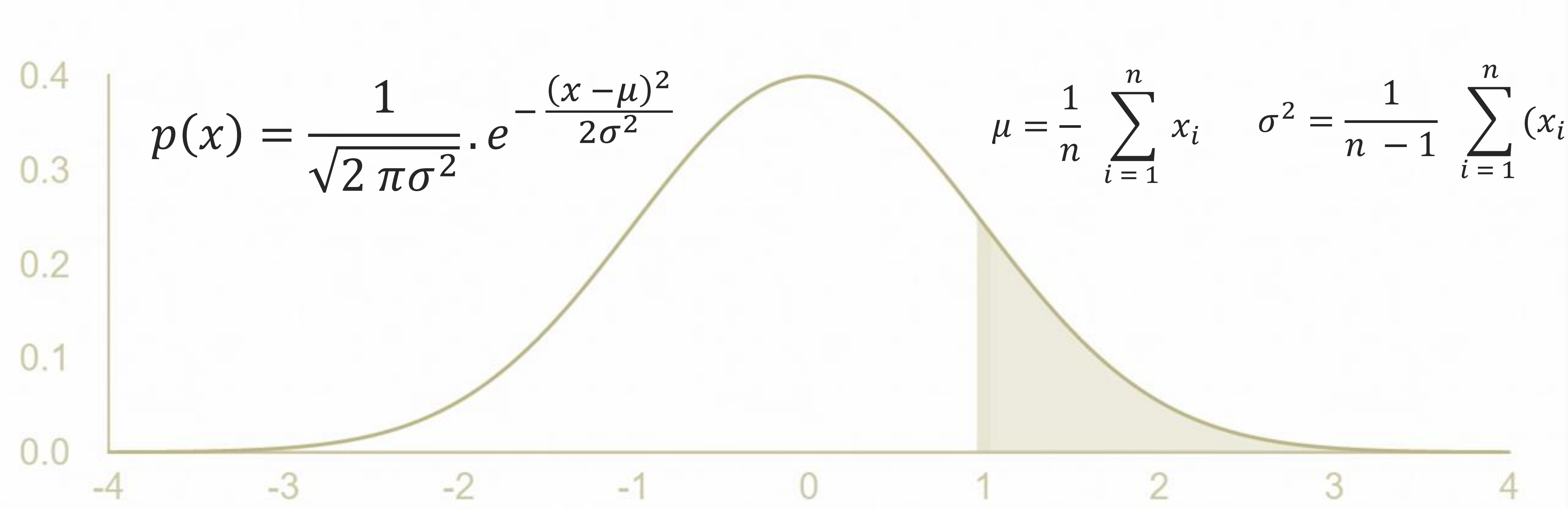
Probability Density Function (pdf)

Gaussian function

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} . e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2$$

$p(x)$



x

The Posterior Probability:

The posterior probability of a class given input (continuous) variable x

$$P(c_j \mid x_i) = \frac{p_{c_j}(x_i).\,P(c_j)}{p_{c_j}(x_i)}$$

$p(x)$ is a probability density function over variable x and $c_j = f(x)$ is a class label

Naïve Bayes Classifier:

Assuming a **uniform** prior $P(c_j)$ over the hypothesis space, MAP reduces to Maximum Likelihood learning:

$$\hat{y} = \underset{c_j \in \{1,2,\ldots,k\}}{\operatorname{argmax}} \prod_{i=1}^{n} p_{c_j}(\, x_i \mid c_j)$$

Combining Gaussian function (pdfs)

$$p_{c_j}(x_i) = \frac{1}{m\sqrt{2\pi\sigma^2}} \sum_{k=1}^{m} e^{-\frac{(x_i - x_k)^2}{2\sigma^2}}$$

m being total number of examples in a training set labelled as c_j

Example Table

Set	Examples	Attributes			Class
		A	B	C	
Training	Ex1	3.2	2.1	2.1	Pos
	Ex2	5.2	6.1	7.5	Pos
	Ex3	8.5	1.3	0.5	Pos
	Ex4	2.3	5.4	2.45	Neg
	Ex5	6.2	3.1	4.4	Neg
	Ex6	1.3	6.0	3.35	Neg
Test	**Ex7**	9.0	3.6	3.3	Pos / Neg ?

Naïve Bayes Classifier: Homework

For the given training example (Table in Slide #44), computer the following

If $p_{\text{pos}}\left(A_{ex_7}\right) = \frac{1}{3\sqrt{2\,\pi}}\left[e^{-\,0.5\,\left(A_{ex_7} - A_{ex_1}\right)} + e^{-\,0.5\,\left(A_{ex_7} - A_{ex_2}\right)} + e^{-\,0.5\,\left(A_{ex_7} - A_{ex_3}\right)}\right]$

compute $p_{\text{pos}}(\mathbf{x}_7) = \; p_{\text{pos}}\left(A_{ex_7}\right) \,.\, p_{\text{pos}}\left(B_{ex_7}\right) \,.\, p_{\text{pos}}\left(C_{ex_7}\right)$

If $p_{\text{neg}}\left(A_{ex_7}\right) = \frac{1}{3\sqrt{2\,\pi}}\left[e^{-\,0.5\,\left(A_{ex_7} - A_{ex_4}\right)} + e^{-\,0.5\,\left(A_{ex_7} - A_{ex_5}\right)} + e^{-\,0.5\,\left(A_{ex_7} - A_{ex_6}\right)}\right]$

compute $p_{\text{neg}}(\mathbf{x}_7) = \; p_{\text{neg}}\left(A_{ex_7}\right) \,.\, p_{\text{neg}}\left(B_{ex_7}\right) \,.\, p_{\text{neg}}\left(C_{ex_7}\right)$

determine the output class by computing $\hat{y} = \text{argmax}(p_{\text{pos}}(\mathbf{x}_7), p_{\text{neg}}(\mathbf{x}_7))$

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 5/10: MDP

Part 3

Markov Decision Process

DR VARUN OJHA

Department of Computer Science

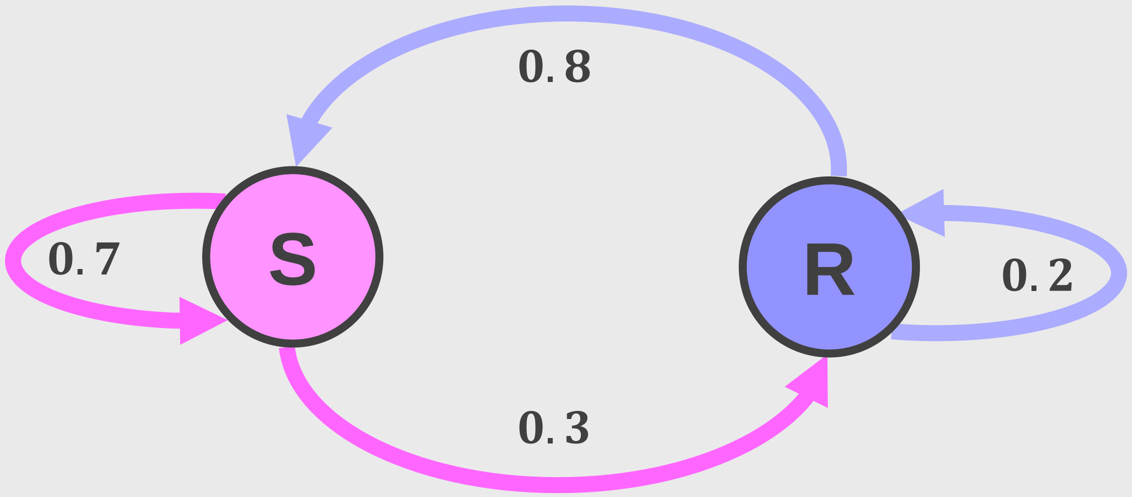
University of Reading

Markov Process/ Markov Chain

A sequence of random states $S_1, S_2, \cdots S_t$ with the *Markov property,* i.e., future state S_{t+1} depends only on current state S_t, where S_t capture all relevant information for S_{t+1} from past sequence $S_{t-1}, S_{t-2}, \cdots S_0$.

Source Demo: http://setosa.io/ev/markov-chains/

Markov Process/ Markov Chain



Markov Chain State Space

	S: Sunny	R: Rain
S: Sunny	$P(S \mid S)$: 0.7	$P(S \mid R)$: 0.3
R: Rain	$P(R \mid S)$: 0.8	$P(R \mid R)$: 0.2

Transition Probability Matrix
(State Transition Matrix)

Source Demo: http://setosa.io/ev/markov-chains/
(Accessed on 07 Feb 2021)

Markov Decision Process

- Markov decision processes (MDPs) are an **extension of Markov Chains** with the addition of **rewards for each action**.

- Conversely, if only one **action** exists for each state (e.g. "wait") and all rewards are the same (e.g. "zero"), a Markov decision process reduces to a Markov Chain.

Markov Decision Processes

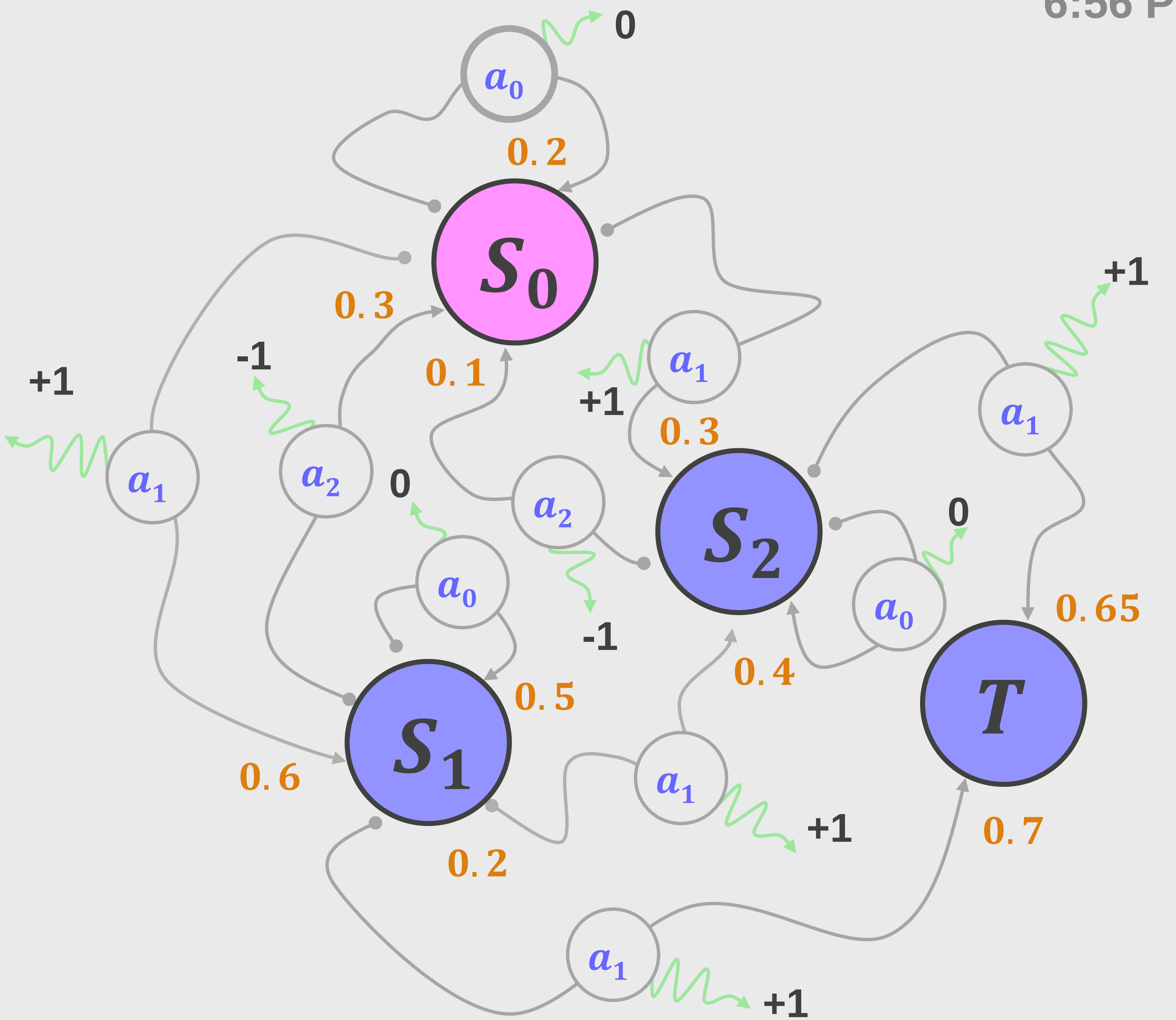
- A Markov decision process (MDP) is a **discrete time stochastic control process**.
- It provides **a mathematical framework for modelling decision making** in situations where outcomes are partly random and partly under the control of a decision maker.
- MDPs are useful for studying **optimization problems** solved via dynamic programming and reinforcement learning.

Markov Decision Process: Definition

- A Markov Decision Process is a **4-tuple** (S, A, P_a, R_a, γ), where
  - $S = \{s_1, s_2, \dots\}$ is a finite set of states.
  - $A = \{a_1, a_2, \dots\}$ is a finite set of actions (alternatively, A_s is the finite set of actions available from state S),
  - $P_a(s, s') = \mathrm{P}\,(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$,
  - $R_a(s, s')$ is the immediate **reward** (or expected immediate reward) received after transitioning from state s to state s', due to action a.
  - $\gamma \in [0, 1]$ is a **discount factor**, 0 being insignificant and 1 being significant reward

MDP: Example

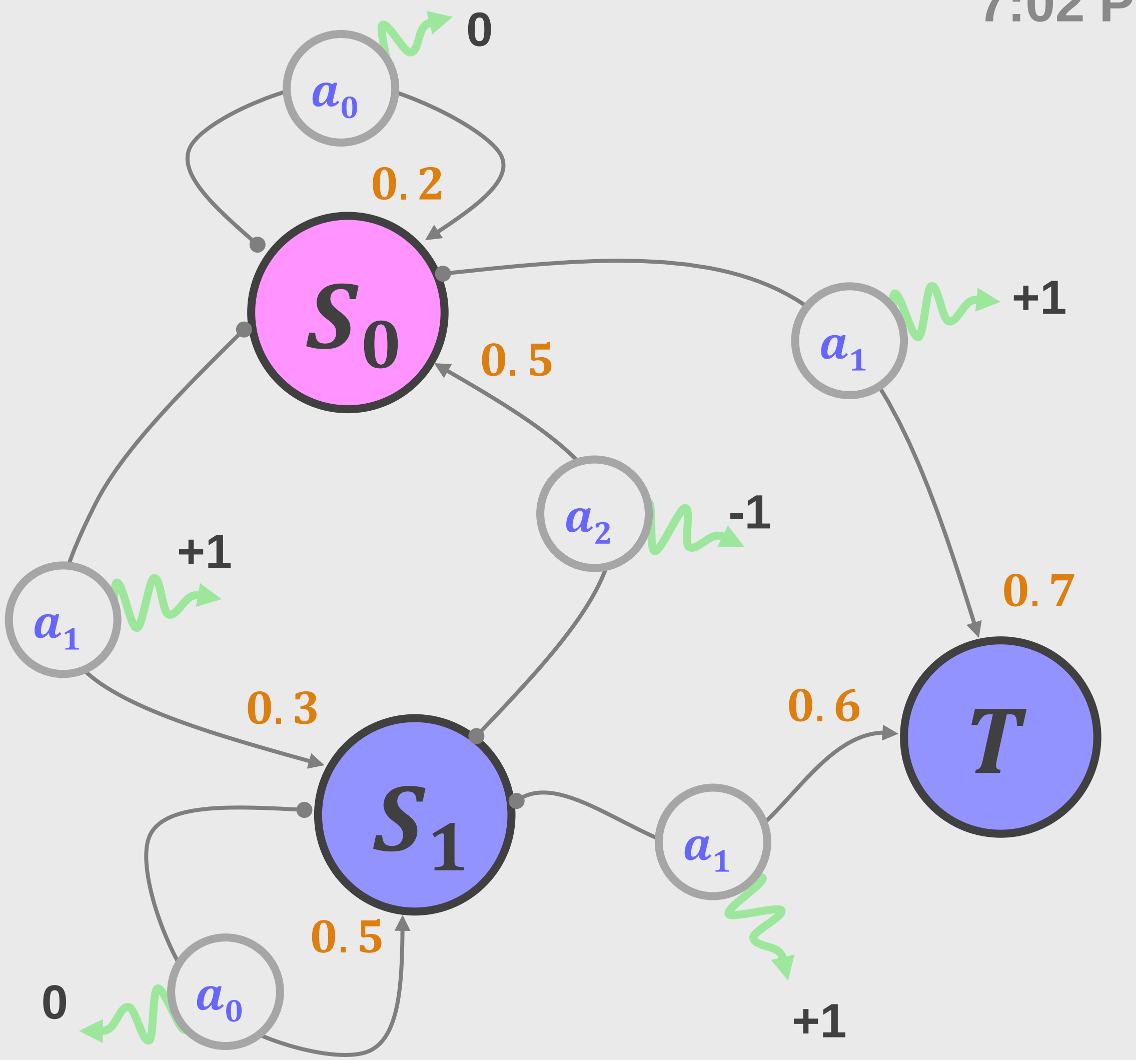
- three states S_0, S_1, S_2, T.
- two actions a_0, a_1, a_2.
- $P_a(s, s')$
- $R_a(s, s')$
- e.g., rewards {**-1, 0,** and **+1**} (green arrows).



MDP: Example

- three states S_0, S_1, T.
- two actions a_0, a_1, a_2.
- $P_a(s, s')$
- $R_a(s, s')$

e.g., rewards {**-1, 0,** and **+1**} (green arrows).



Policy π

- Optimal "**policy**" design for the decision maker is the core problem of MDPs.
- A **policy** denoted as π is its recommended action a for state s, i.e., what to do next when at state s.

$$\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s}) = \boldsymbol{P}[\boldsymbol{A} = \boldsymbol{a} \mid \boldsymbol{S} = \boldsymbol{s}]$$

- It is the of distribution over actions given states. It fully defines the behaviour of an agent. This means a MDP depends on current state not all the history.
- If a decision maker has a complete policy π it will know what to do next.

State Value Function $v_\pi(s)$

State Function $v_\pi(s)$ is the expected return (reward) value accumulated at a state s and following policy π.

$$\boldsymbol{v_\pi(s) = \mathbb{E}_\pi[G_t \,|S = s]}$$

$$\boldsymbol{v_\pi(s) = \sum_{a \in A} \pi(\, a \,|\, s) \; (R_s + \gamma \sum_{s' \in S} P(s', s) v_\pi(s'))}$$

This is **Bellman Expectation State-Value Equation**. Where reward intermediate R and state value s.

State-value function tells us how good is it to be in state $\boldsymbol{s}$ by following policy $\boldsymbol{\pi}$.

Action Value Function $q_\pi(s, a)$

Action Function $q_\pi(s, a)$ is the expected return (reward) value accumulated at a state s and following policy π.

$$
\begin{aligned}
q_\pi(s, a) &= \mathbb{E}_{\boldsymbol{\pi}}[\boldsymbol{G_t} \mid \boldsymbol{S} = \boldsymbol{s}, \boldsymbol{A} = \boldsymbol{a}] \\
&= \boldsymbol{R_s^a} + \boldsymbol{\gamma} \sum_{\boldsymbol{s'} \in \boldsymbol{S}} \boldsymbol{P^a}(\boldsymbol{s'}, \boldsymbol{s}) \boldsymbol{v_\pi}(\boldsymbol{s'}) \\
&= \boldsymbol{R_s^a} + \boldsymbol{\gamma} \sum_{\boldsymbol{s'} \in \boldsymbol{S}} \boldsymbol{P^a}(\boldsymbol{s'}, \boldsymbol{s}) \sum_{\boldsymbol{a'} \in \boldsymbol{A}} \boldsymbol{\pi}(\boldsymbol{a'} \mid \boldsymbol{s'})\, \boldsymbol{q_\pi}(\boldsymbol{s'}, \boldsymbol{a'})
\end{aligned}
$$

This **Bellman Action-Value Equation function** tells us how good is it to take an action at state $\boldsymbol{s}$ by following policy π.

This give us an idea how good it is to take an action a at state s.

π: Policy Design

- **Policy** design for the decision maker is the core problem of MDPs.

- A **policy** denoted as π and $\pi(s)$ for its recommended action for state s, i.e., what to do next when at state s.

- If a decision maker has a **complete policy** π it will know what to do next when on state s. In that case, Markov Decision process will behave like a Markov Chain since $\boldsymbol{P}\,(\boldsymbol{s_{t+1}} = \boldsymbol{s'} \mid \boldsymbol{s_t} = \boldsymbol{s}\,, \boldsymbol{a_t} = \boldsymbol{a})$ will reduce to $\boldsymbol{P}\,(\boldsymbol{s_{t+1}} = \boldsymbol{s'} \mid \boldsymbol{s_t} = \boldsymbol{s})$ because determinacy of $\pi(s)$.

π^*: Optimal Policy Design

Optimal Policy π^* is the one that gives the highest **expected utility**. That is a policy π that will **maximise** some cumulative function of the random rewards R_{a_t}, typically, the expected discounted sum over a potentially infinite horizon:

$$U = \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$$

where $\gamma \in [0, 1]$ is a **discount factor**, 0 being insignificant and 1 being significant reward, and $a_t = \pi(s_t)$

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 5/10: DN

Part 4

Decision Network

DR VARUN OJHA

Department of Computer Science

University of Reading

Decision Network (Influence Diagrams)

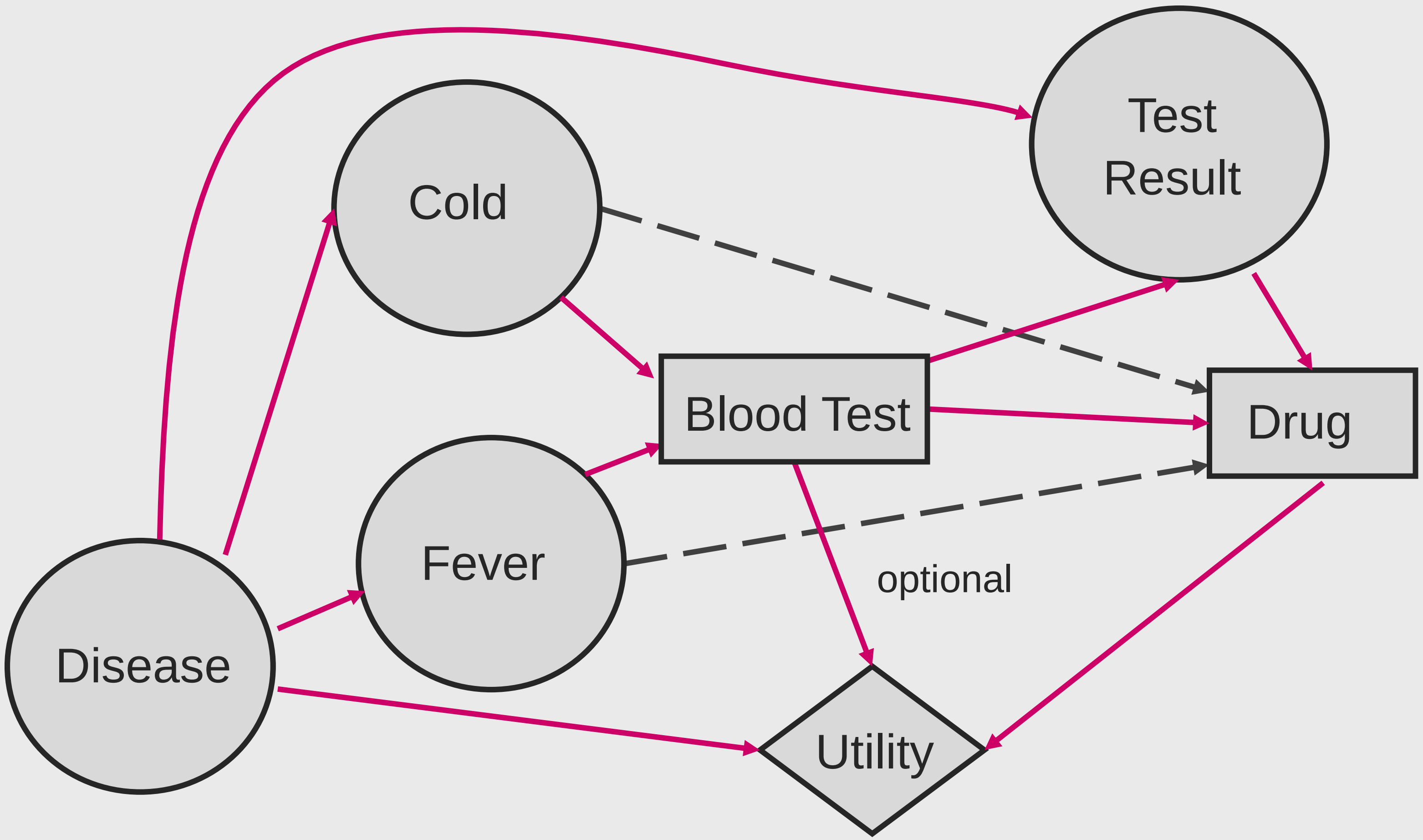
- Decision networks (aka influence diagrams) provide a representation for sequential decision making

- Basic idea
  - **Random variables** like in Bayes Nets
  - **Decision variables** that you “control”
  - **Utility variables** which state how good certain states are (e.g., metrics, objective function, measurements).

Decision Network: **Example**

Random Variables (denoted by circles).

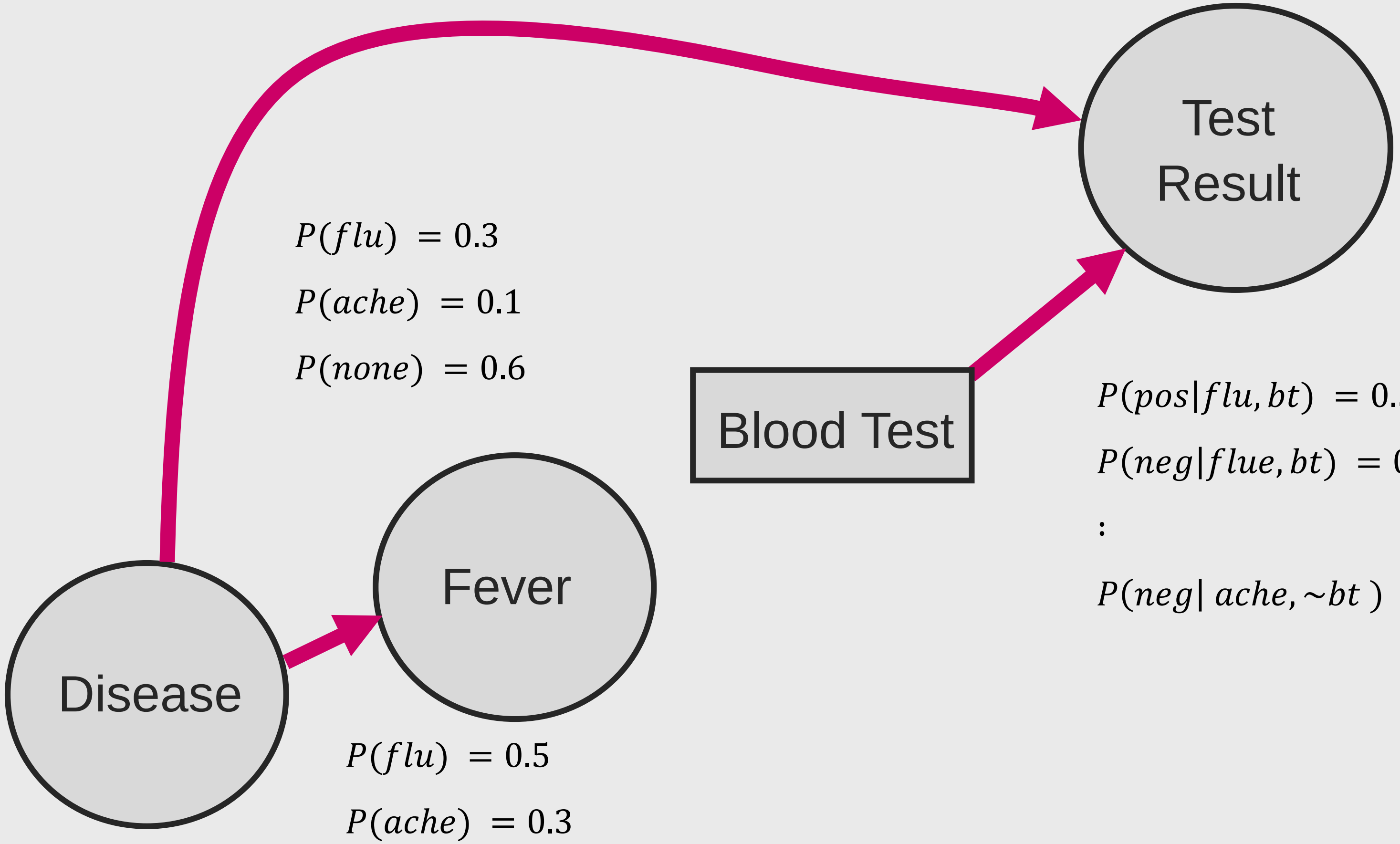
Variables the **decision maker** sets (denoted by squares).



Decision Network: **Chance Node**

Random Variable
(denoted by circles).

Each nodes have probabilistic dependence on parent nodes



$P(flu) = 0.3$

$P(ache) = 0.1$

$P(none) = 0.6$

$P(pos|flu, bt) = 0.5$

$P(neg|flue, bt) = 0.8$

$\vdots$

$P(neg|\ ache, \sim bt\ ) = 0.9$

$P(flu) = 0.5$

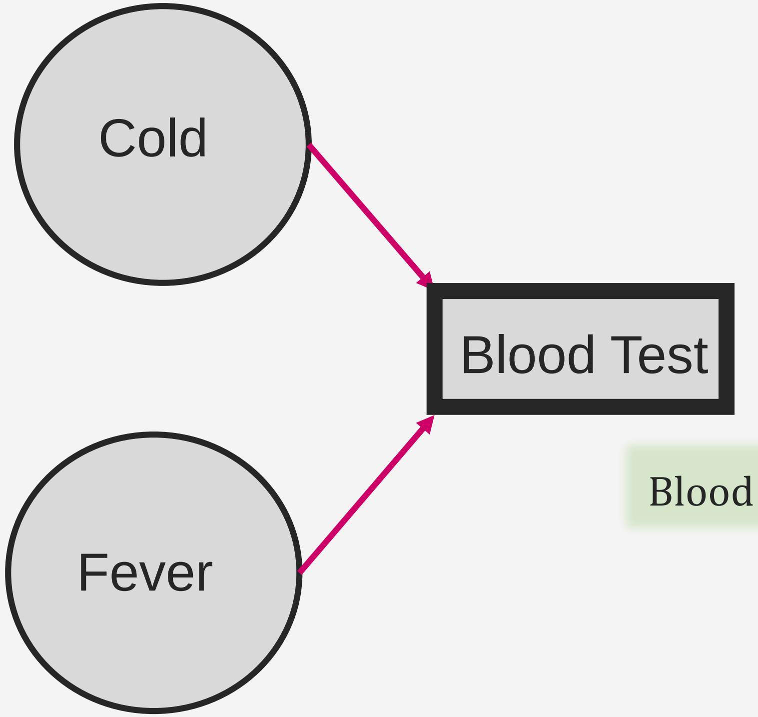
$P(ache) = 0.3$

$P(none) = 0.05$

Decision Network: **Decision Node**

Variables the **decision maker** sets (denoted by squares).

Parents reflect the **information available** at a time of decision making



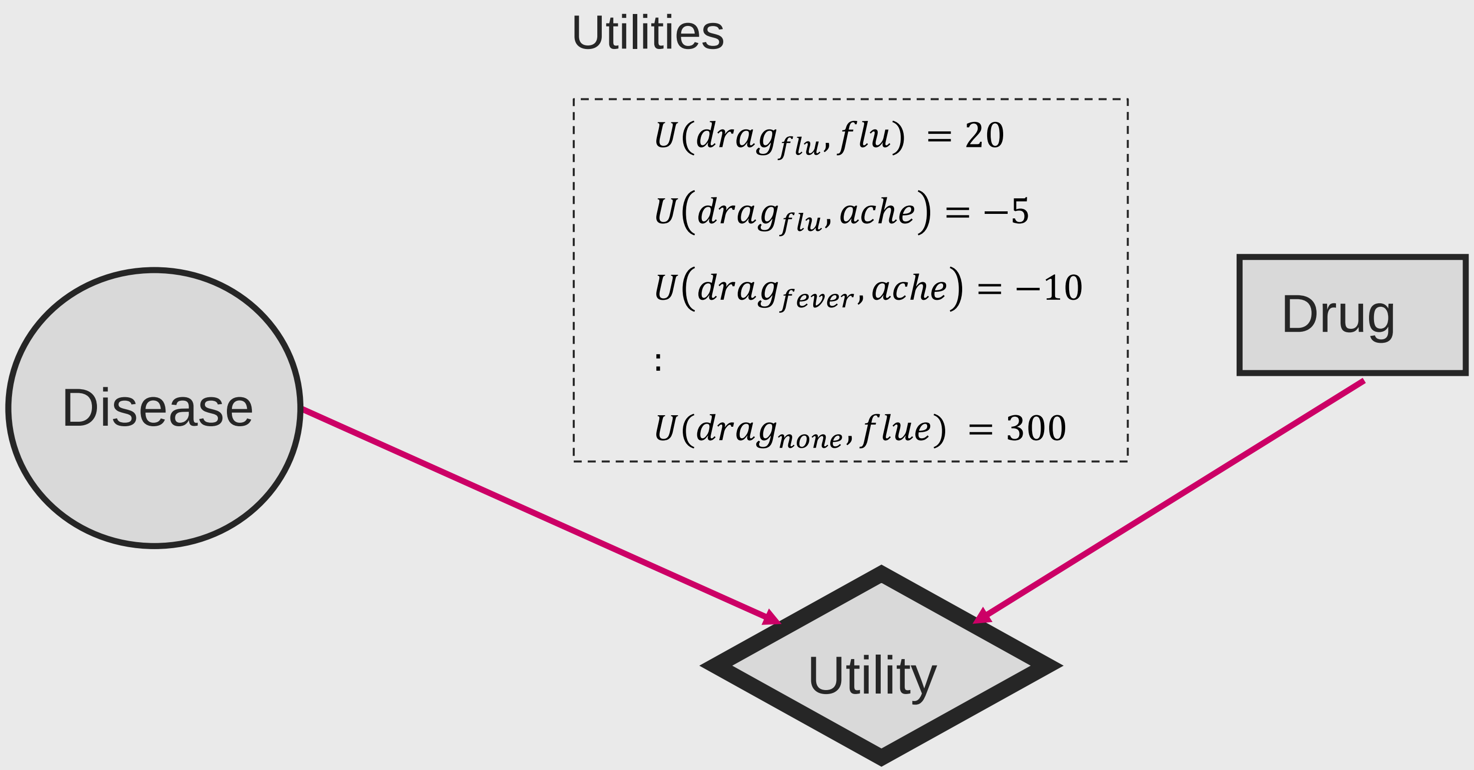
Blood Test ∈ $\{BT, \sim BT\}$

Decision Network: **Value Node**

Value node: Specify the utility of a state.
(denoted by diamond).

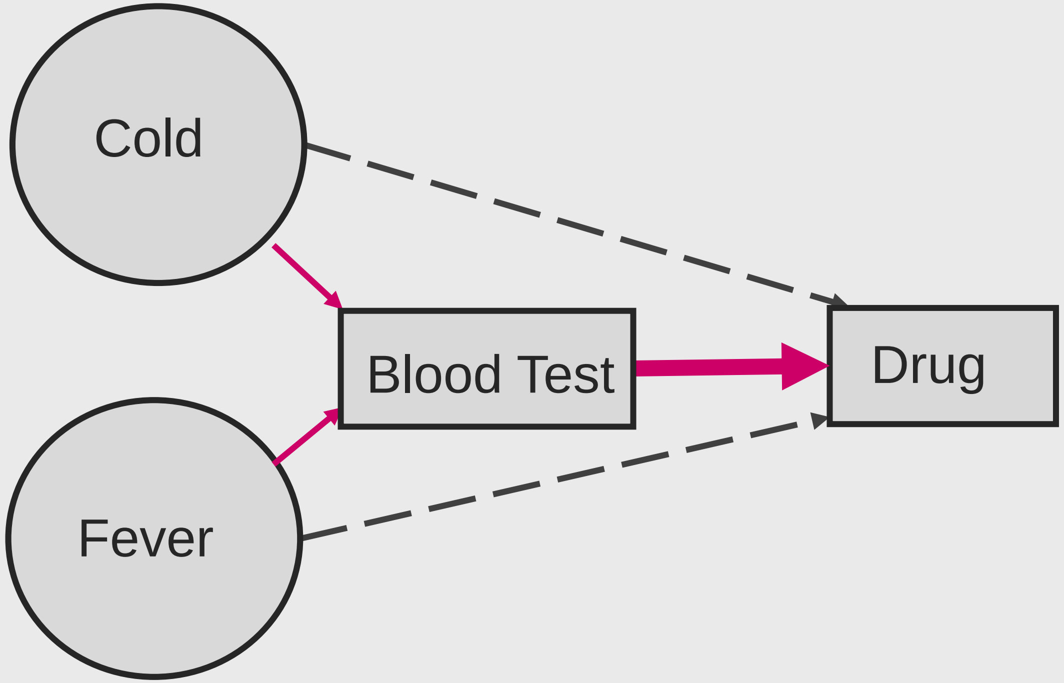
Utility depends only on state of the parent

Generally, only one value node exists in a network



Assumptions

- Decision nodes are **totally ordered**
  - Given variables $D_1, D_2, \ldots, D_n$, the decision nodes are made in sequence.
- **No forgetting property**
  - Any information available for a decision D_i is available for decision D_j for $j > i$
  - All parents of decision D_i are also the parents of decision D_j for $j > i$



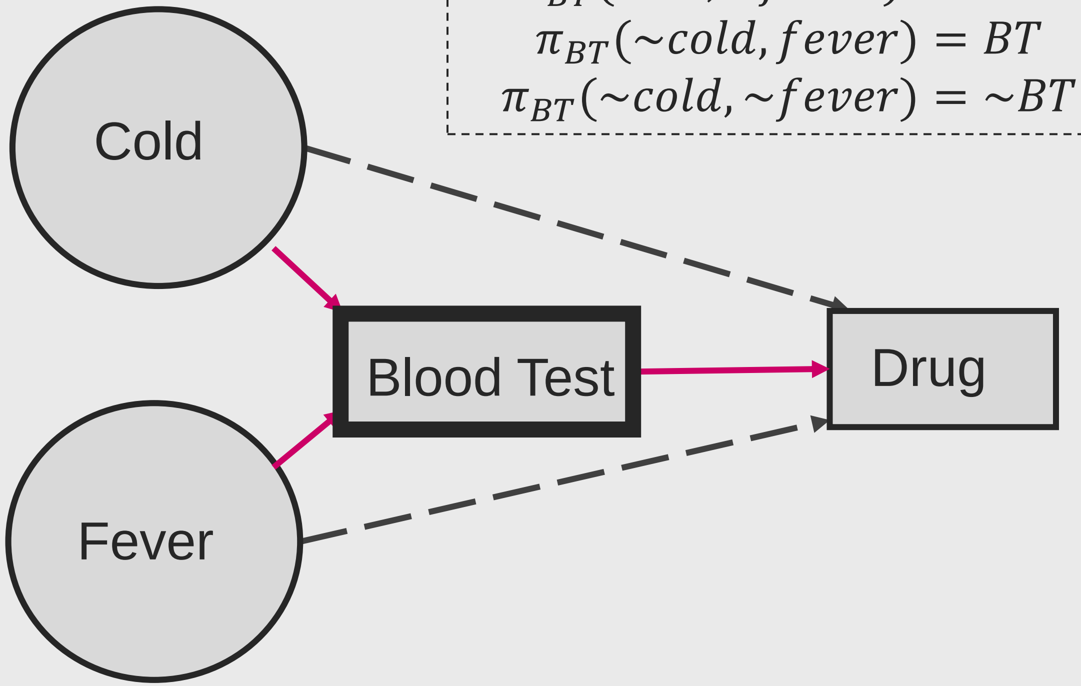
Policy

- Let $Parent(Di)$ be the **parents** of a decision node D_i
  - $Domain(Parent(Di))$ is the set of **assignments** to $Parent(D_i)$,
  - e.g., $\{cold, \sim cold)$ and $\{fever, \sim fever\}$

- A **policy** π is a set of mappings π_i, one for each decision node D_i
  - $\pi_i\,(Di)$ associates a decision for each parent assignment
  - $\pi_i : Domain(Parent(Di)) \rightarrow Domain(Di)$

Policy

$$\pi_{BT}(cold, fever) = BT$$
$$\pi_{BT}(cold, \sim fever) = \sim BT$$
$$\pi_{BT}(\sim cold, fever) = BT$$
$$\pi_{BT}(\sim cold, \sim fever) = \sim BT$$



Value of a Policy

- Given assignment x to random variables X, let $\pi(x)$ be the assignment to decision variables denoted byπ.

- Value of π, i.e., the expected utility, $\boldsymbol{EU(\pi)}$, is:

$$\boldsymbol{EU(\pi) = \sum_x P\big(x, \pi(x)\big)\, U\big(x, \pi(x)\big)}$$

- Where, $\boldsymbol{P}$ is the probability of the outcome and $\boldsymbol{U}$ is utility function and

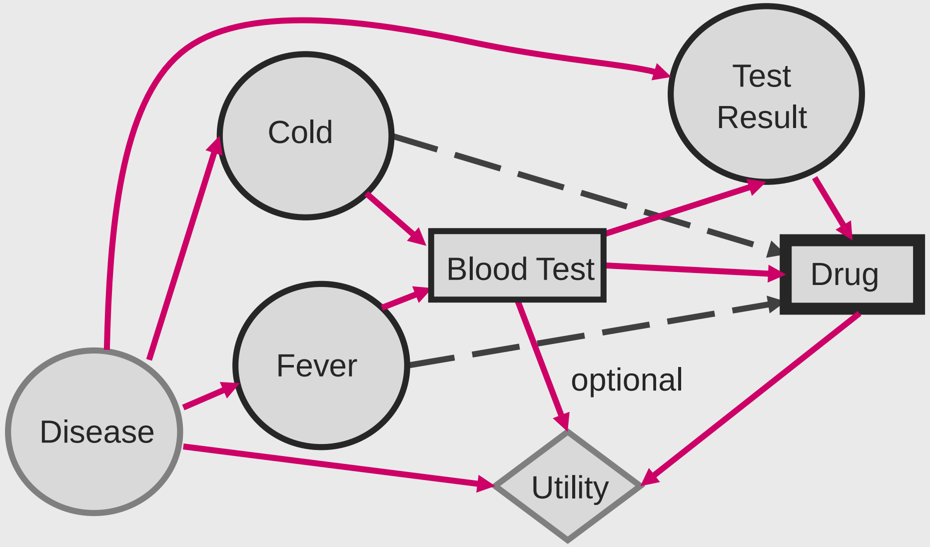
Optimal Policy $\boldsymbol{\pi}^*$

An **optimal** policy $\boldsymbol{\pi}^*$ is given by $\boldsymbol{EU}(\boldsymbol{\pi}^*) \geq \boldsymbol{EU}(\pi)$ for all $\boldsymbol{\pi}$

Maximisation over all other

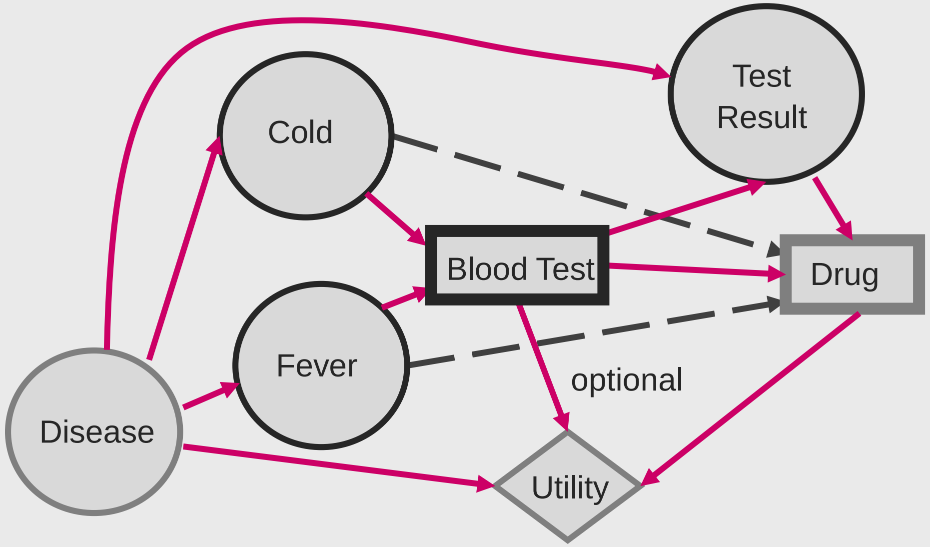
Computing Optimal Policy

- Compute backward direction
  - Compute optimal policy of the last decision node in a sequence
  - E.g., Drugs in this case
  - **For each** parent $\{C, F, BT, TR\}$ and **for each** decision value $D \in \{drug_{flue}, drug_{ache}, no_{drug}\}$, compute the **expected utility,** EU of choosing a decision value *D.*
  - For each $Domain(Parent(D))$, set a **policy choice** $\boldsymbol{\pi_D}(C, F, BT, TR)$ where the value of D is maximum.



Computing Optimal Policy

- Compute backward direction
  - Compute optimal policy of the second last decision node in a sequence based on last policy $\boldsymbol{\pi_D}(C, F, BT, TR)$
  - E.g., Blood Test is just before Drug
  - Since $\boldsymbol{\pi_D}(C, F, BT, TR)$ is already computed its **fixed**.
  - Treat $\boldsymbol{D}$ as a random variable with a deterministic probability
  - Computer **policy choice** for Blood Test, BT, where the value of BT is maximum.

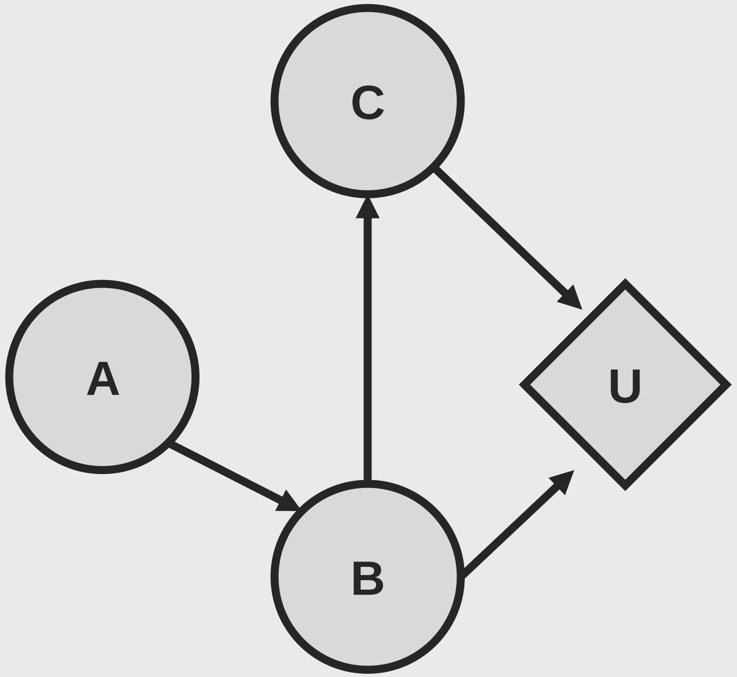


Computing Expected Utilities

- Computing expected utilities with Bayes Net is straightforward

- Utility nodes are just **factors** that can be dealt with using **variable elimination**

- $\boldsymbol{EU = \sum_{A,B,C} P(A,B,C)\, U(B,C)}$
- $\boldsymbol{EU = \sum_{A,B,C} P(A) \cdot P(B|A) \cdot P(C|B) \cdot U(B,C)}$



Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 5/10: Learning (Algorithms)

Part 5

Practical Exercise

(available In a separate video)

DR VARUN OJHA

Department of Computer Science

University of Reading