MDEC™ t- academy | Future Skills Academy for Emerging Technologies

# CERTIFIED DEEP LEARNING PROFESSIONAL (CDLP)

ICTC International Council for Technology Certifications

**Digital Tech Faculty Expert (DTeX)**

ICTC International Council for Technology Certifications

# Reinforcement Learning

Foundations

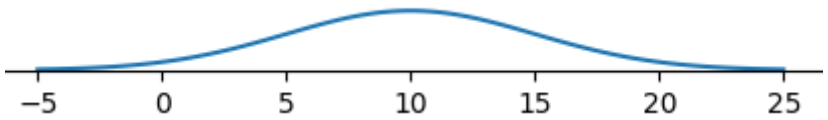
# Single / Multi armed Bandit


# Scenario 1

1. You are a visiting professor to a very small town. They've invited you to their local college to live there for 300 days and give some lectures
2. Every morning you go and give your lectures and every night you eat at one of the three restaurants in the city
3. There's only 3 restaurants and there's no reviews and you don't have any understanding about which of these restaurants you might like more
4. There are these distributions about how much happiness you're going to derive from eating at those restaurants

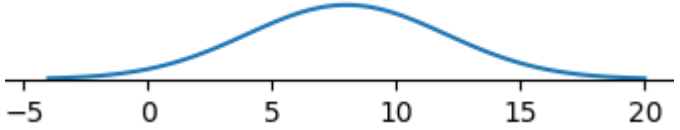
# Restaurants and Its Happiness

Restaurant One



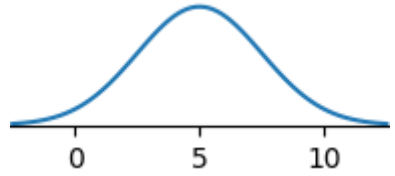

Average 10 Std 5

Restaurant Two




Average 8 Std 4

Restaurant Three




Average 5 Std 2.5

# Goal

1. However, the distribution of happiness for each restaurant is are hidden to you in the beginning
2. Since you have never visited any of these restaurants you have no idea how much happiness the restaurants are going to bring
3. This is an example of a multi-armed bandit problem. We must strike the balance between exploration and exploitation

# More Examples

**Picking your college major**

1. You might enter a college. You don't know exactly what you want to study. So, you spend some time “exploring” different courses
2. You might take statistics; you might take some language courses and so on
3. Finally, you must enter exploitation phase. Based on your current knowledge about the courses, which made you the happiest you're going to go ahead and pursue that major

# More Examples

**Personal work**

- After I graduated, I worked as a software engineer and develop new systems
- There are many ways to develop a new system
- So, I exploring all the different ways to develop this system trying to find the best way and just picking away and moving forward with the project

# Exploration vs Exploitation

Your goal over the course of these 300 days is, to strike a very good balance between two concepts called exploration and exploitation

1. **Exploration** which is basically going to any of these restaurants enough times to understand which one you like the best
2. **Exploitation** which means once you have a pretty good idea about which restaurant makes you the happiest continue going to that restaurant in order to derive as much happiness as possible before these 300 days run out

Note: The naive strategies would be “**explore only**” and “**exploit only**”

# Strategies – Explore Only

1. You're going to spend 300 days visiting a restaurant every night. That means an average of 100 days at each restaurant. From restaurant 1 you're going to derive on average 10 units of happiness and from restaurant 2 you'll get 8 units of happiness and from restaurant 3 you'll get 5 average units of happiness.
2. Let’s do the math, and your average happiness over the 300 days is 2300
3. Important metric **“regret (Greek symbol rho ρ)”** in multi-armed bandit, which helps us to find how good is our strategy. The regret is the difference between the average happiness from your strategy and the maximum happiness
4. If you knew happiness distribution earlier, you would 3000 optimal units of happiness by visiting restaurant 1 every day. So, this explore only strategy is 700 points away from that optimal 3000

# Strategies – Exploit Only

1. During the first 3 days, you will visit restaurant 1, restaurant 2 and restaurant 3. Based on best meal on those three days you will pick a restaurant and visit the same restaurant for the rest of 297 days
2. By chance, you might get a bad meal at the best restaurant. Just one meal at each restaurant is not good enough to decide about where to eat. Day 1 you get happiness of 7 from restaurant 1. Day 2 you get happiness of 8 from restaurant 2. Day 3 you get happiness of 5 from restaurant 3. In this exploit only, since you got best happiness from restaurant 2 you are going to eat there for rest of the 297 days
3. So, your happiness would basically be the sum of the first three days which is 20 (7 + 8 + 5) and plus 297 days times this 8 units of happiness (2396).
4. In this strategy, average regret is going to be 604 units.

# Strategies – Epsilon Greedy

1. Epsilon-Greedy is a simple and widely used algorithm for addressing the exploration-exploitation tradeoff in the multi-armed bandit problem.
2. The Epsilon-Greedy algorithm strikes a balance between exploration and exploitation by setting a small value to epsilon (ε)
3. Initialization – In our case let us set the parameter ε (epsilon) between 0 and 1, which determines the level of exploration in the strategy.
4. A typical value for ε is a small positive number, such as 0.1.

# Strategies – Epsilon Greedy

At each day (time step t):

1. Generate a random number between 0 and 1.
2. If the random number is less than or equal to ε, choose a random restaurant to explore.
3. If the random number is greater than ε, choose the restaurant with the highest estimated happiness based on past observations (exploit).
4. After selecting a restaurant, visit it and observe the happiness level. Update the average sum of total happines $\bar{R} = \frac{1}{n-1} \sum_{i=1}^{n-1} R_i$ f times selected restaurant has been visited.

# Strategies – Epsilon Greedy (Day 1)

1. Generate a random number between 0 and 1: Let's say the random number is 0.7.
2. Since 0.7 is greater than ε (0.1), we choose to exploit
3. However, we do not have enough information about restaurant let us explore a random restaurant. Randomly select Restaurant 1 and visit it.
4. Observe and record the happiness level at Restaurant 1.
5. Update the information for Restaurant 1: Increment the number of times Restaurant 1 has been visited and add the observed happiness to the total sum of happiness for Restaurant 1.

# Strategies – Epsilon Greedy (Day 2)

1. Generate a random number between 0 and 1: Let's say the random number is 0.05.
2. Since 0.05 is less than ε (0.1), we choose to explore a random restaurant.
3. Randomly select Restaurant 2 and visit it.
4. Observe and record the happiness level at Restaurant 2.
5. Update the information for Restaurant 2: Increment the number of times Restaurant 2 has been visited and add the observed happiness to the total sum of happiness for Restaurant 2.

# Strategies – Epsilon Greedy (Day 3)

1. Generate a random number between 0 and 1: Let's say the random number is 0.9.
2. Since 0.9 is greater than ε (0.1), let’s exploit best-known restaurant based on the current estimates.
3. At this point, we have information only about Restaurant 1 and 2, but not about Restaurant 3.
4. Based on the available information, Restaurant 1 has the highest estimated happiness so far.
5. Select Restaurant 1 and visit it.
6. Observe and record the happiness level at Restaurant 1.
7. Update the information for Restaurant 1: Increment the number of times Restaurant 1 has been visited and add the observed happiness to the total sum of happiness for Restaurant 1.

# Strategies – Epsilon Greedy (Day 4 to 300)

1. Continue the process
2. At each day, generate a random number to decide whether to explore or exploit.
3. Visit the selected restaurant.
4. Observe and record the happiness level and update the estimates for the chosen restaurant.
5. Based on the Average happiness of each restaurant and standard deviation it is calculated as our regret is around 100 units when we use Epsilon Greedy method.

# Important Terms

1. **Reward** – In our previous problem the reward it the happiness we derived from the restaurant (Restaurant 1 => 10, Restaurant 2 => 8 and Restaurant 3 => 5).
2. **Action** – Visiting the Restaurant is the action. In our case we have 3 Restaurants. So, we have 3 different actions to perform.
3. **Value of an Action** - The value of an action represents the expected reward you would receive on average by choosing that restaurant. Formally Value of an Action is denoted as Q(A).

# Value of an Action

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$$

| Trials (n) | Rn | Qn | Qn+1 | Formula |
|---|---|---|---|---|
| 0 | - | 0 | 0 | |
| 1 | +10 | 0 | 10 | 0 + 1/1 (10 – 0) = 10 |
| 2 | +7 | 10 | 8.5 | 10 + ½ (7 – 10) = 8.5 |
| 3 | +9 | 8.5 | 8.66 | 8.5 + 1/3 (9 – 8.5) = 8.66 |

# Strategies – Epsilon Greedy

Epsilon Greedy Strategy is good; however, the performance is going to depend on

1. The choice of epsilon
2. The happiness distributions 5, 8 and 10 (far apart)
3. The standard deviations (big enough where there's some ambiguity)

But if the happiness numbers were far apart then the “exploit only” strategy might start looking better because there's less of a chance that you're going to get a bad meal from the best restaurant. If the best restaurant is a lot better than the second-best restaurant obviously “exploit only” strategy is better.

# Disadvantage – Epsilon Greedy

- Epsilon greedy performs well, but it’s easy to see how selecting restaurant at random can be inefficient.
- If you have restaurant one with average happiness is 50 and restaurant two with average happiness is 8 and the restaurant three with average happiness is 5.
- Epsilon greedy is equally likely to pick either of these restaurants when exploring random.

# Strategies – UCB

1. UCB (Upper Confidence Bound) is another popular algorithm used in multi-armed bandit problems to balance exploration and exploitation.
2. UCB estimates the value of each action and considers both the mean reward and the uncertainty in the estimate when choosing which action to play.
3. UCB tends to select actions with higher potential for improvement while exploring arms that have not been sufficiently sampled.

Note: In real life we use to say, we haven’t visited enough this restaurant for a long time. Why not give a chance to them ? – That is UCB.

# Strategies – UCB

1. First 3 days visit the restaurants one by one and update the happiness values.
2. $4^{th}$ day onwards start applying UCB formula to find which restaurant to visit

$$Q_a = Q_a + \sqrt{\frac{2 \log (N)}{n_a}}$$

| | Q(1) | Q(2) | Q(3) | Q(4) | Q(5) | Q(6) | Q(7) | Q(8) | Q(9) | Q(10) | Q(11) | Q(12) | Q(13) | Q(14) | Q(15) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 10.00 | 10.78 | 11.47 | 12.10 | 12.69 | 13.25 | 13.83 | 14.43 | 15.05 | 15.68 | 16.33 | 16.98 | 17.65 | 18.33 | 19.01 |
| B | 8.00 | 8.78 | 9.75 | 10.85 | 12.03 | 13.28 | 14.20 | 14.98 | 15.67 | 16.30 | 16.89 | 17.49 | 18.10 | 18.71 | 19.34 |
| C | 5.00 | 5.78 | 6.75 | 7.85 | 9.03 | 10.28 | 11.58 | 12.92 | 14.31 | 15.72 | 17.16 | 18.20 | 19.06 | 19.82 | 20.51 |

# Multi-Armed Bandit - Non-Stationary

1. So far, whatever we have seen is Stationary, where the average happiness of each restaurant never change.
2. Let us assume every single day in a week the Restaurants have different chefs.
3. Obviously based on the chef the happiness reward is going to change for each Restaurant
4. And this is called Non-Stationary problem.

# Multi-Armed Bandit - Non-Stationary

1. Let us assume the happiness of each restaurant is as follows

| Restaurant | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|---|---|---|---|---|---|---|
| 1 | 10 | 7 | 4 | 10 | 8 | 9 |
| 2 | 8 | 10 | 9 | 12 | 3 | 7 |
| 3 | 5 | 10 | 12 | 9 | 11 | 8 |

2. One common approach is to use a weighted average or a moving average to track the evolving value of actions. We will add more weight to the recent rewards or the recent trials than then previous trials.

$$Q_{n+1} = Q_n + \alpha (R_n - Q_n)$$

3. Alpha is smoothing parameter between 0 and 1

# Simple Probability

Now what is the probability of getting a tail when a coin is tossed?

1. Number of total possible outcomes = 2
2. Favorable outcome of getting a tail = 1


3. Probability of getting a tail = Favorable Outcome / Total Outcomes
4. Result = ½

# Simple Probability

A perfect cubic die is thrown. Find the probability of following events

The sample Space S = {1, 2, 3, 4, 5, 6} => n(S) = 6

1. An even number comes up => { 2, 4, 6 } => 3/6 => ½
2. A number multiple of 3 or 5 => { 3, 6, 5 } => 3/6 => ½
3. A number multiple of 3 and 5 => { } => 0/6 => 0
4. Score is greater than => 2 { 3, 4, 5, 6 } => 4/6 => 2/3

# Conditional Probability

We want to figure out the probability of something, given that we have some other piece of information. The other piece of information that we can bring to the table, will influence the probability of whatever we are investigating.

Example, Out of 10 suspects 1 person is a Criminal. The probability of a specific person being the criminal depends on the probability of finding fingerprints or other evidence at the crime scene.

# Conditional Probability

## Bayes' theorem

Suppose there is a high probability (let's say 90%) that the criminal's fingerprints will be found at the crime scene. In this case, if a specific suspect's fingerprints are found, the likelihood that this person is the criminal becomes much higher.

Formula: $P(A|B) = P(B|A) * P(A) / P(B)$

# Conditional Probability

**Bayes' theorem** $P(A|B) = P(B|A) * P(A) / P(B)$

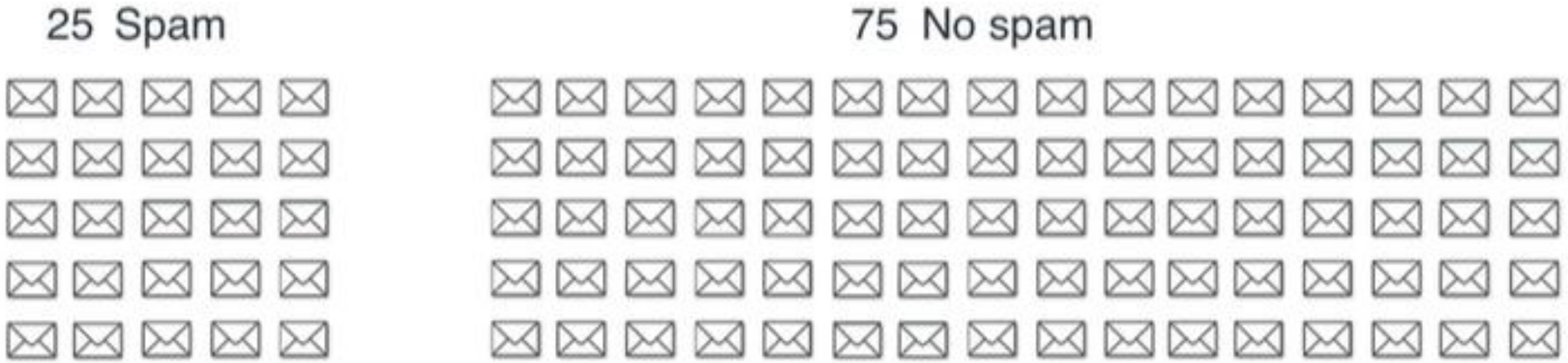
1. $P(A|B)$ = Probability of the person being the criminal given that their fingerprints were found at the crime scene.
2. $P(B|A)$ = Probability of finding fingerprints at the crime scene given that the person is the criminal.
3. $P(A)$ = Initial probability of the person being the criminal.
4. $P(B)$ = Probability of finding fingerprints at the crime scene.

# Naive Bayes Classifier – SPAM Detector

Total number of emails: 100

Spam Emails: 25

No Spam emails: 75

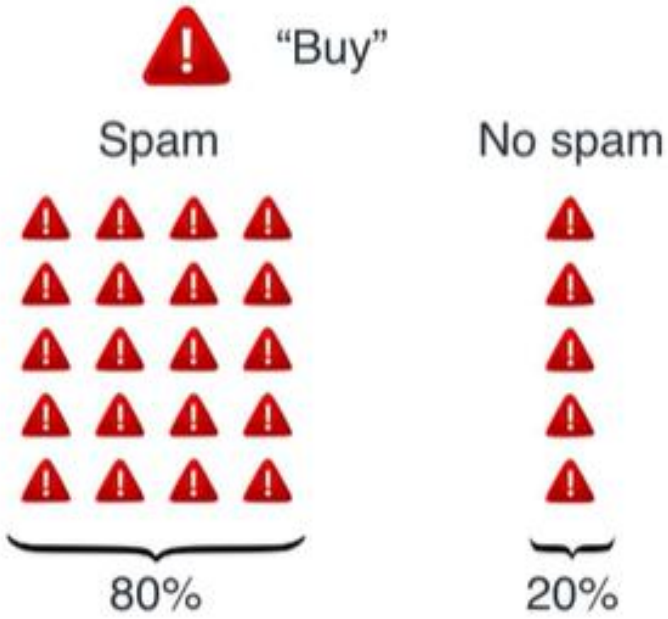


# Naive Bayes Classifier – SPAM Detector

Emails that contains the word Buy

# Naive Bayes Classifier – SPAM Detector

Let us take those emails that contains the word buy and ask a question




**Quiz:** If an e-mail contains the word "buy", what is the probability that it is spam?

- ○ 40%
- ○ 60%
- ● 80%
- ○ 100%

Even though it can be easily calculated, in the next few slides we going to prove it using bayes theorem.

# Naive Bayes Classifier – SPAM Detector

## Using Bayes Theorem

1. Total number of emails (n) = 100
2. Number of SPAM emails (S) = 25
3. Number of not SPAM emails (¬S) = 75
4. Number of SPAM emails containing the word "Buy" = 20
5. Number of not SPAM emails containing the word "Buy" = 5
6. Bayes' theorem: P(S|Buy) = (P(Buy|S) * P(S)) / P(Buy)

# Naive Bayes Classifier – SPAM Detector

Bayes' theorem: $P(S|Buy) = (P(Buy|S) * P(S)) / P(Buy)$

1. P(S|Buy) is the probability that an email is SPAM given that it contains the word "Buy" (posterior probability).
2. P(Buy|S) is the probability that the word "Buy" appears in SPAM emails (likelihood).
3. P(S) is the probability that any email is SPAM (prior probability of SPAM).
4. P(Buy) is the probability that the word "Buy" appears in any email (marginal probability of "Buy").

# Naive Bayes Classifier – SPAM Detector

P(Buy|S) = Number of SPAM emails containing "Buy" / Total number of SPAM emails

*P(Buy|S) = 20 / 25*

P(S) = Total number of SPAM emails / Total number of emails

*P(S) = 25 / 100 = 0.25*

P(Buy) = Total number of emails containing "Buy" / Total number of emails

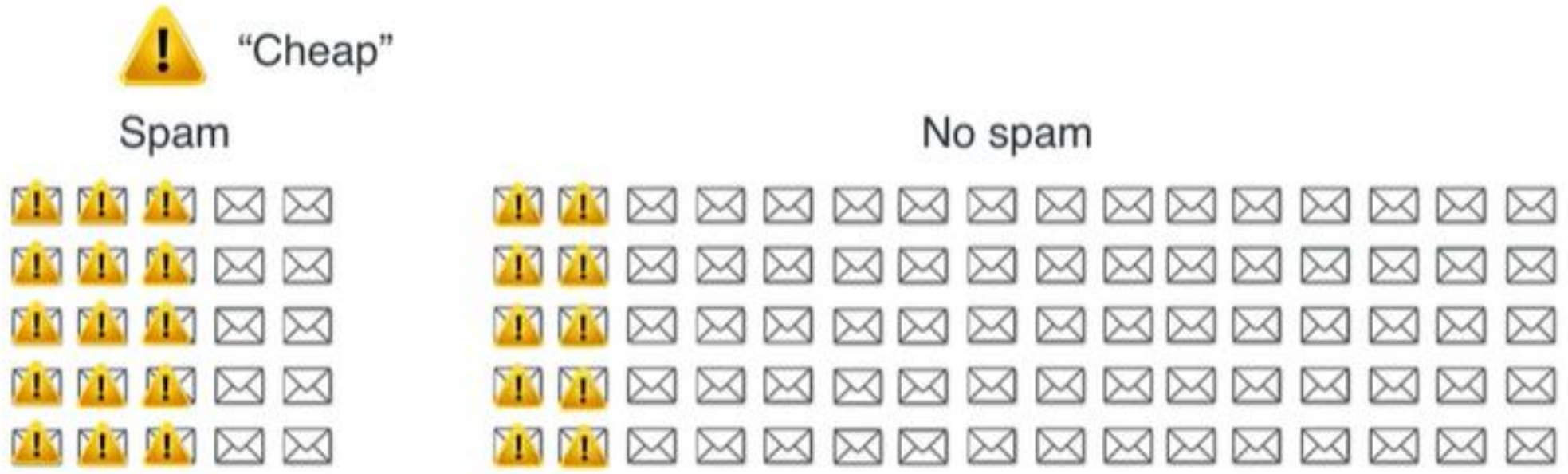
*P(Buy) = (20 + 5) / 100 = 25 / 100 = 0.25*

P(S|Buy) = (P(Buy|S) * P(S)) / P(Buy)

*P(S|Buy) = (20 / 25) * (0.25) / (25 / 100) => 0.8 =>* ***80%***

# Naive Bayes Classifier – SPAM Detector

Emails that contains the word Cheap



# Naive Bayes Classifier – SPAM Detector

Let us take those emails that contains the word Cheap and ask a question

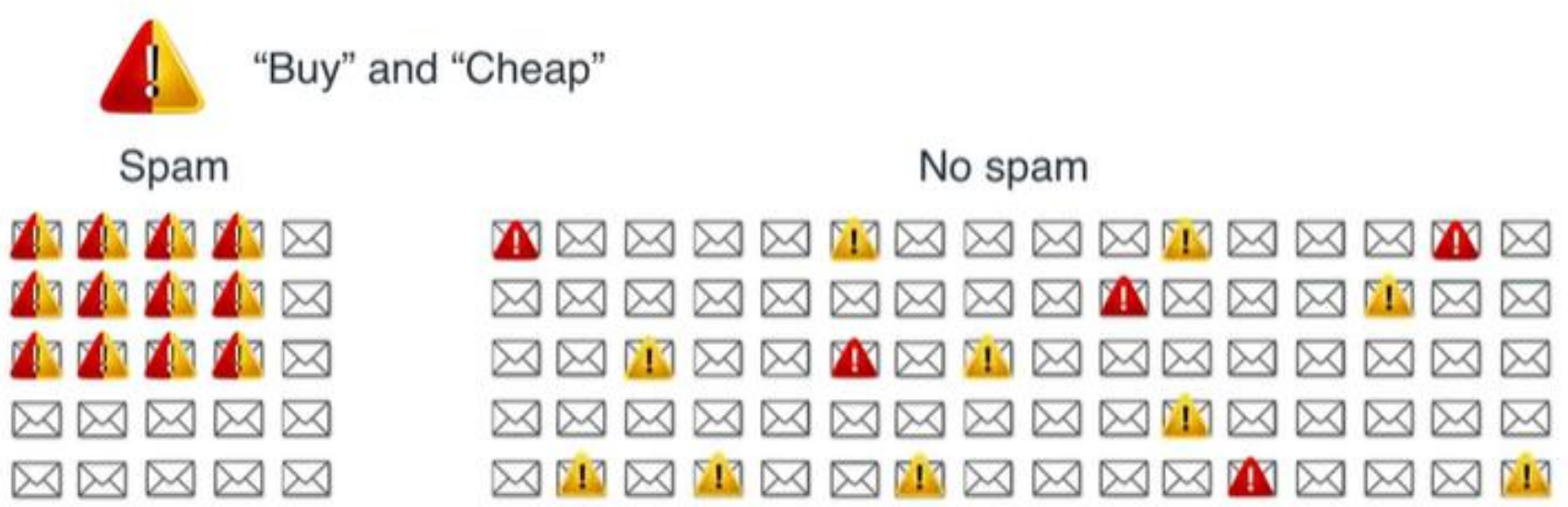

**Quiz:** If an e-mail contains the word "cheap", what is the probability that it is spam?

- ○ 40%
- ● 60%
- ○ 80%
- ○ 100%

**Solution:**
60%

# Naive Bayes Classifier – SPAM Detector

Emails that contains the word Buy and Cheap




Under Spam we got 12 emails contain both words "Buy" and "Cheap"

Under No Spam we got 0 email contain both words "Buy" and "Cheap"

# Naive Bayes Classifier – SPAM Detector

Let us take those emails contains both words Cheap and Buy


**Quiz:** If an e-mail contains the words "buy" and "cheap", what is the probability that it is spam?

- ○ 40%
- ○ 60%
- ○ 80%
- ● 100%

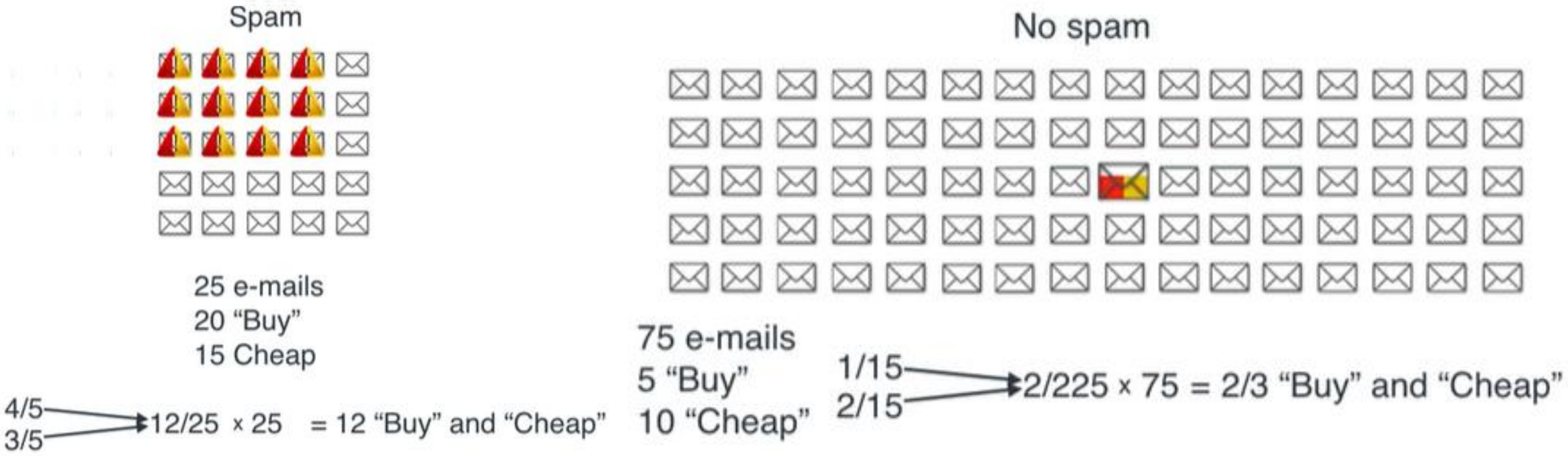
We are skeptical about this answer. Any classifier that tells you 100% is too strong. Either you collect more data or go for a "Naive" assumption.

# Naive Bayes Classifier – SPAM Detector

Try to come up with a sensible number (10% of 5%), even there is no email contain both words Buy and Cheap. This is a **Naive assumption**.


Note: That means ½ an email in 100 emails contain both words Buy and Cheap

# Naive Bayes Classifier – SPAM Detector



# Naive Bayes Classifier – SPAM Detector


**Quiz:** If an e-mail contains the words "buy" and "cheap", what is the probability that it is spam?

$$\frac{12}{12 + 2/3} = \frac{36}{38}$$

$$= 94.737\%$$

This is precisely Naïve Bayes Classifier – It is a combination of Bayes theorem and be naïve assumption

# Naive Bayes Classifier – SPAM Detector

| | SPAM | | No SPAM | |
|---|---|---|---|---|
| Total | 25 | | 75 | |
| Buy | 20 | 4/5 | 5 | 1/15 |
| Cheap | 15 | 3/5 | 10 | 2/15 |
| Buy & Cheap | 12 | 12/25 (4/5 * 3/5) | 2/3 (2/225 * 75) | 2/225 (1/15 * 2/15) |

$$\frac{12}{12 + 2/3} = \frac{36}{38} = 94.737\%$$

# Naive Bayes Classifier – SPAM Detector

| | SPAM | | No SPAM | |
|---|---|---|---|---|
| **Total** | **25** | | **75** | |
| **Buy** | **20** | **4/5** | **5** | **1/15** |
| **Cheap** | **15** | **3/5** | **10** | **2/15** |
| **Work** | **5** | **1/5** | **30** | **6/15** |
| **Buy Cheap Work** | **12/5** | **12/125** (4/5 * 3/5 * 1/5) | **4/5 (12/3375 * 75)** | **12/3375** (1/15 * 2/15 * 6/15) |

$$\frac{12/5}{12/5 + 4/15} = \frac{36}{40} = 90\%$$

# Naive Bayes Classifier – SPAM Detector

S: Spam
H: Ham (not spam)
B: 'Buy'

$$P(S \mid B) = \frac{P(B \mid S)\,P(S)}{P(B \mid S)\,P(S) + P(B \mid H)\,P(H)}$$

$$P(\text{spam if "Buy"}) = \frac{\frac{20}{25}\ \frac{25}{100}}{\frac{20}{25}\ \frac{25}{100} + \frac{5}{75}\ \frac{75}{100}} = 80\%$$

# Naive Bayes Classifier – SPAM Detector

S: Spam
H: Ham (not spam)
B: 'Buy'
C: 'Cheap'

$$P(S \mid B \cap C) = \frac{P(B|S)\,P(C|S)\,P(S)}{P(B|S)\,P(C|S)\,P(S) + P(B|H)\,P(C|H)\,P(H)}$$

$$P(\text{spam if "Buy" \& "Cheap"}) = \frac{\frac{20}{25}\,\frac{15}{25}\,\frac{25}{100}}{\frac{20}{25}\,\frac{15}{25}\,\frac{25}{100} + \frac{5}{75}\,\frac{10}{75}\,\frac{75}{100}}$$

$$= 94.737\%$$

# Beta Distribution


1. We have 3 biased coins.
2. The probability of getting head using the 1st coin is 0.4.
3. The probability of getting head using the 2nd coin is 0.6.
4. The probability of getting head using the 3rd coin is 0.8.

# Beta Distribution

1. We picked up one of the coin and we don’t know which coin we grabbed. We flip that coin 5 times.
2. We manage to get Head 3 times and Tail 2 times in this order


3. Now which coin we Grab ?

# Beta Distribution


Based on the probability

1. Coin 1 we will give us 2 heads out of 5 flips
2. Coin 2 we will give us 3 heads out of 5 flips
3. Coin 3 we will give us 4 heads out of 5 flips

So, Coin 2 can be the Grabbed on. However, this may not be correct 100%

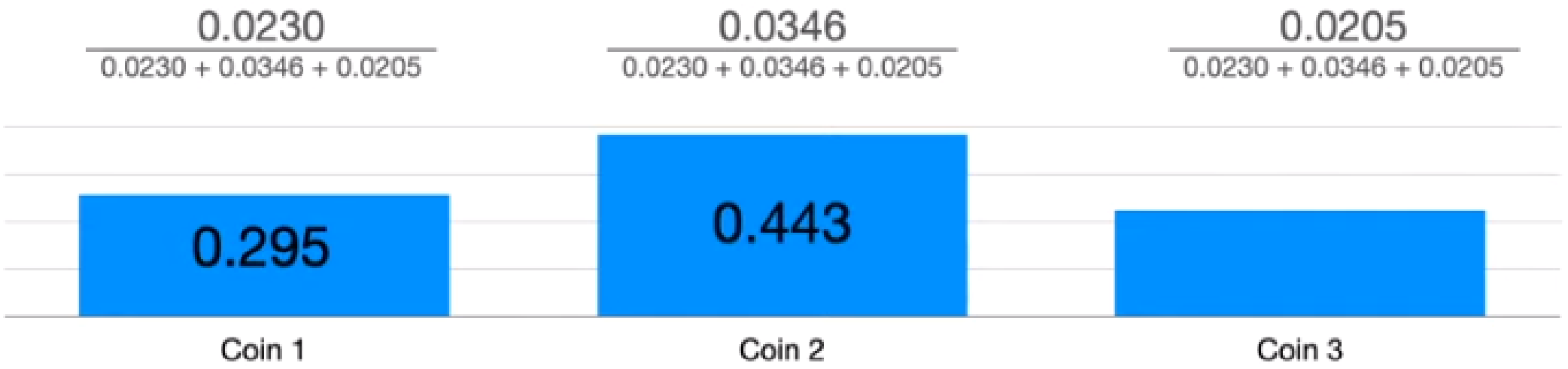
# Beta Distribution

What are the probabilities that we picked each one of the three coins ?

This can be calculated using Bayes Theorem.

# Beta Distribution

Based on Bayes Theorem we divide each probability by total



# Beta Distribution

1. Let us play the same game using 10 coins.
2. We grabbed one and flip we got 7 heads and 3 tails.
3. Probability for each coin to get 7 heads is listed in a table.


7 heads, 3 tails

| Coin 0 | Coin 1 | Coin 2 | Coin 3 | Coin 4 | Coin 5 | Coin 6 | Coin 7 | Coin 8 | Coin 9 | Coin 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| P(H) = 0 | P(H) = 0.1 | P(H) = 0.2 | P(H) = 0.3 | P(H) = 0.4 | P(H) = 0.5 | P(H) = 0.6 | P(H) = 0.7 | P(H) = 0.8 | P(H) = 0.9 | P(H) = 1 |

# Beta Distribution

Let us calculate probabilities that we picked each one of the 10 coins ?

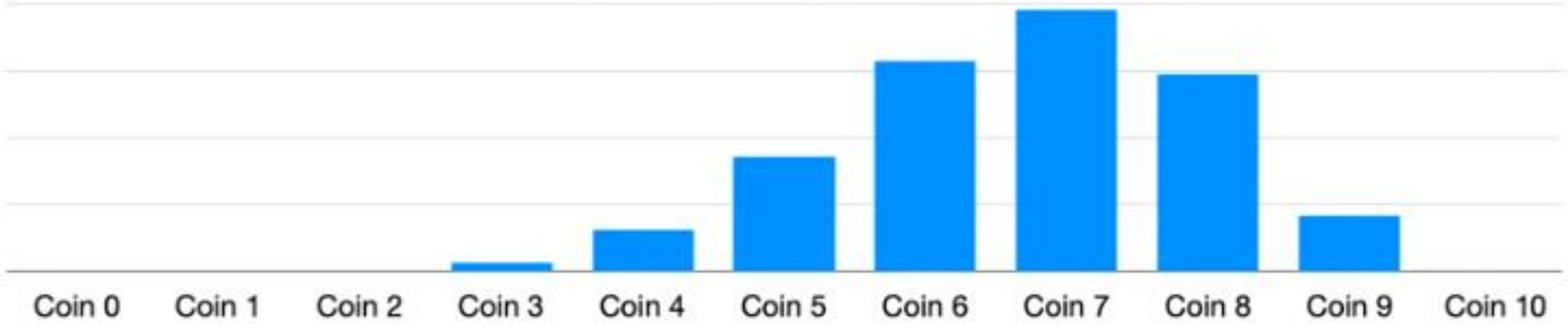

7 heads, 3 tails

| Coin 0 | Coin 1 | Coin 2 | Coin 3 | Coin 4 | Coin 5 | Coin 6 | Coin 7 | Coin 8 | Coin 9 | Coin 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| P(H) = 0 | P(H) = 0.1 | P(H) = 0.2 | P(H) = 0.3 | P(H) = 0.4 | P(H) = 0.5 | P(H) = 0.6 | P(H) = 0.7 | P(H) = 0.8 | P(H) = 0.9 | P(H) = 1 |
| $0^7 \cdot 1^3$ | $0.1^7 \cdot 0.9^3$ | $0.2^7 \cdot 0.8^3$ | $0.3^7 \cdot 0.7^3$ | $0.4^7 \cdot 0.6^3$ | $0.5^7 \cdot 0.5^3$ | $0.6^7 \cdot 0.4^3$ | $0.7^7 \cdot 0.3^3$ | $0.8^7 \cdot 0.2^3$ | $0.9^7 \cdot 0.1^3$ | $1^7 \cdot 0^3$ |
| 0 | 0.00001 | 0.001 | 0.01 | 0.04 | 0.13 | 0.24 | 0.29 | 0.21 | 0.06 | 0 |

# Beta Distribution

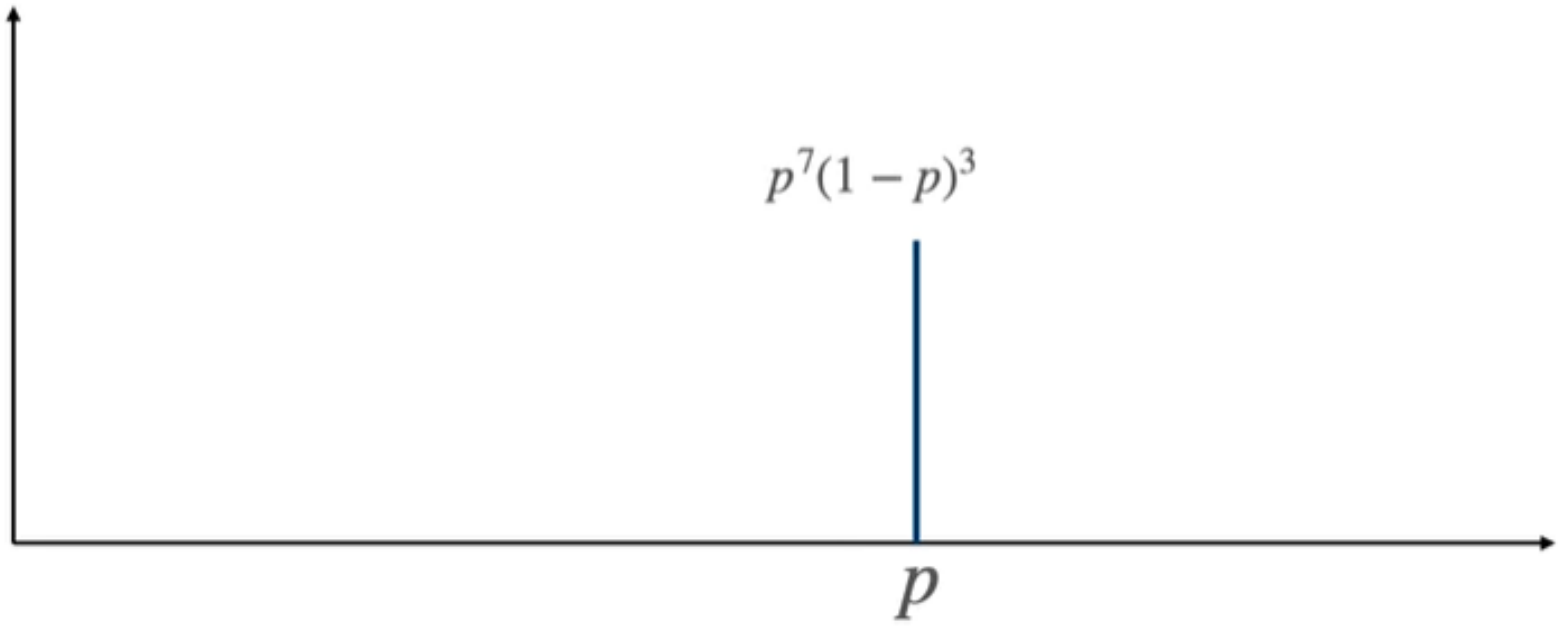

7 heads, 3 tails

| Coin 0 | Coin 1 | Coin 2 | Coin 3 | Coin 4 | Coin 5 | Coin 6 | Coin 7 | Coin 8 | Coin 9 | Coin 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| P(H) = 0 | P(H) = 0.1 | P(H) = 0.2 | P(H) = 0.3 | P(H) = 0.4 | P(H) = 0.5 | P(H) = 0.6 | P(H) = 0.7 | P(H) = 0.8 | P(H) = 0.9 | P(H) = 1 |
| $0^7 \cdot 1^3$ | $0.1^7 \cdot 0.9^3$ | $0.2^7 \cdot 0.8^3$ | $0.3^7 \cdot 0.7^3$ | $0.4^7 \cdot 0.6^3$ | $0.5^7 \cdot 0.5^3$ | $0.6^7 \cdot 0.4^3$ | $0.7^7 \cdot 0.3^3$ | $0.8^7 \cdot 0.2^3$ | $0.9^7 \cdot 0.1^3$ | $1^7 \cdot 0^3$ |
| 0 | 0.00001 | 0.001 | 0.01 | 0.04 | 0.13 | 0.24 | 0.29 | 0.21 | 0.06 | 0 |



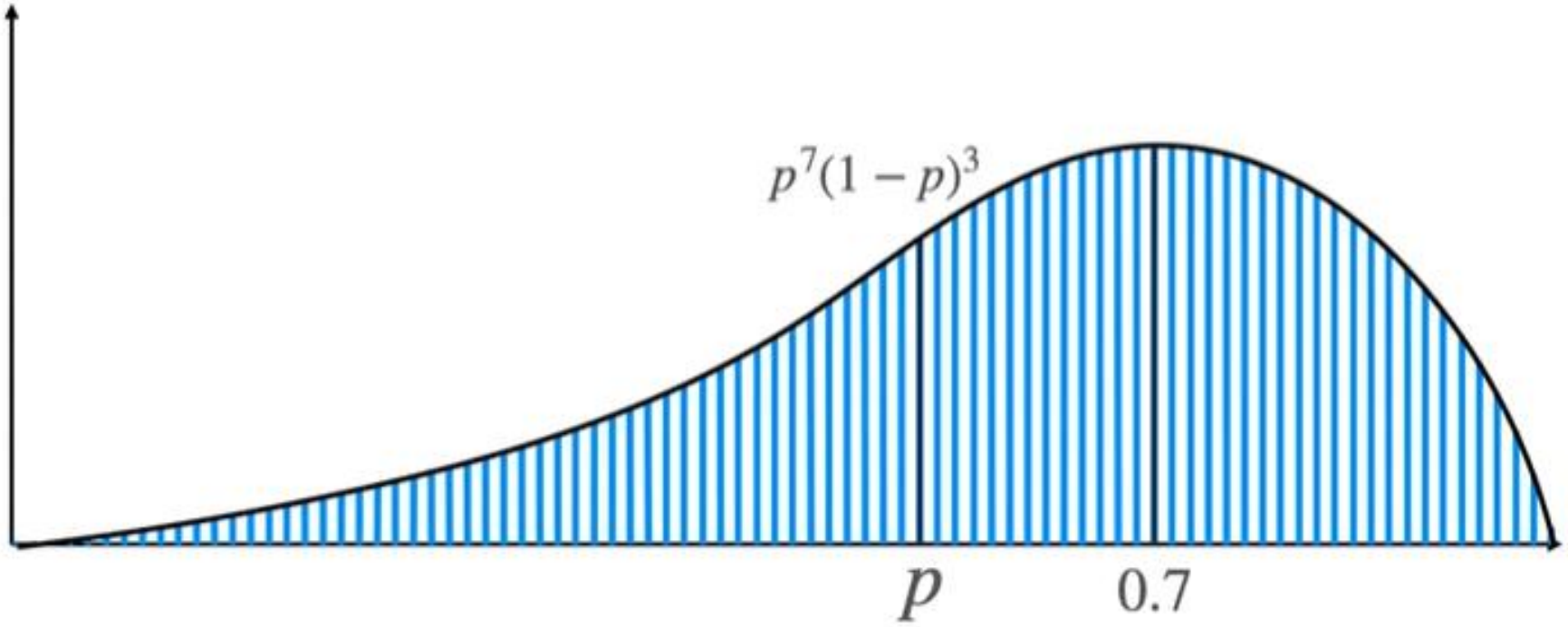
# Beta Distribution

1. Let us play the same game again using 10000 coins.
2. We grabbed one and flip we got 7 heads and 3 tails.



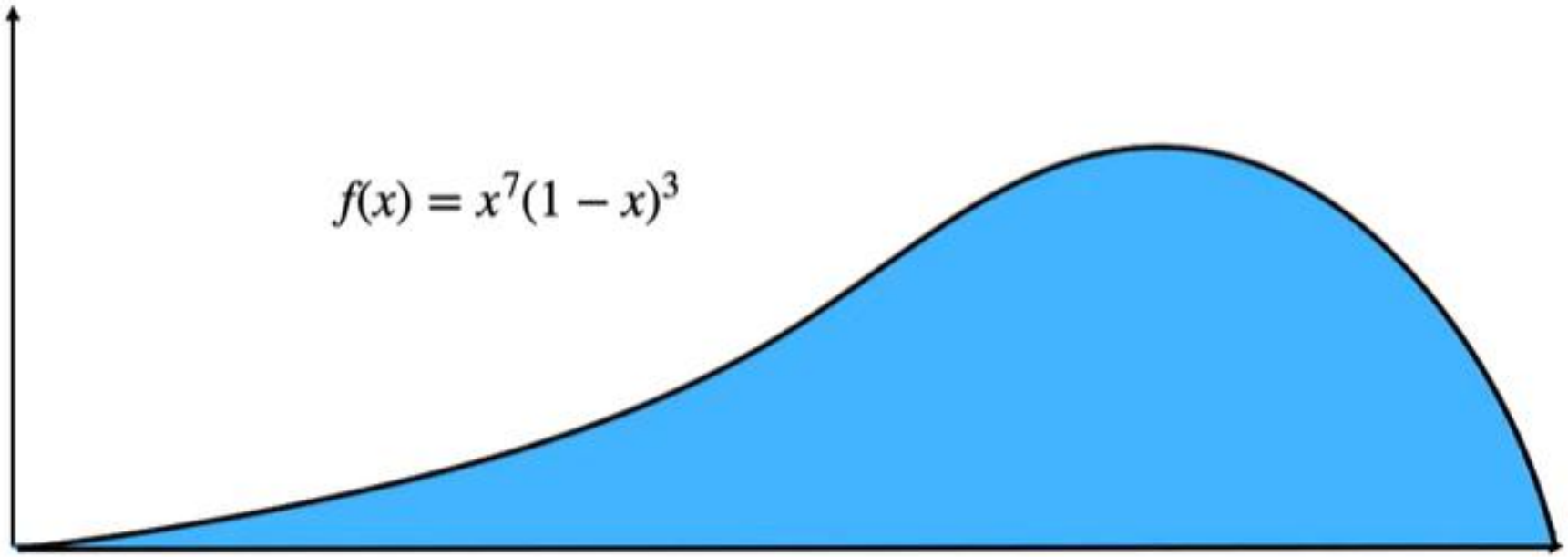
# Beta Distribution

Let us calculate probabilities that we picked each one of 10000 coins



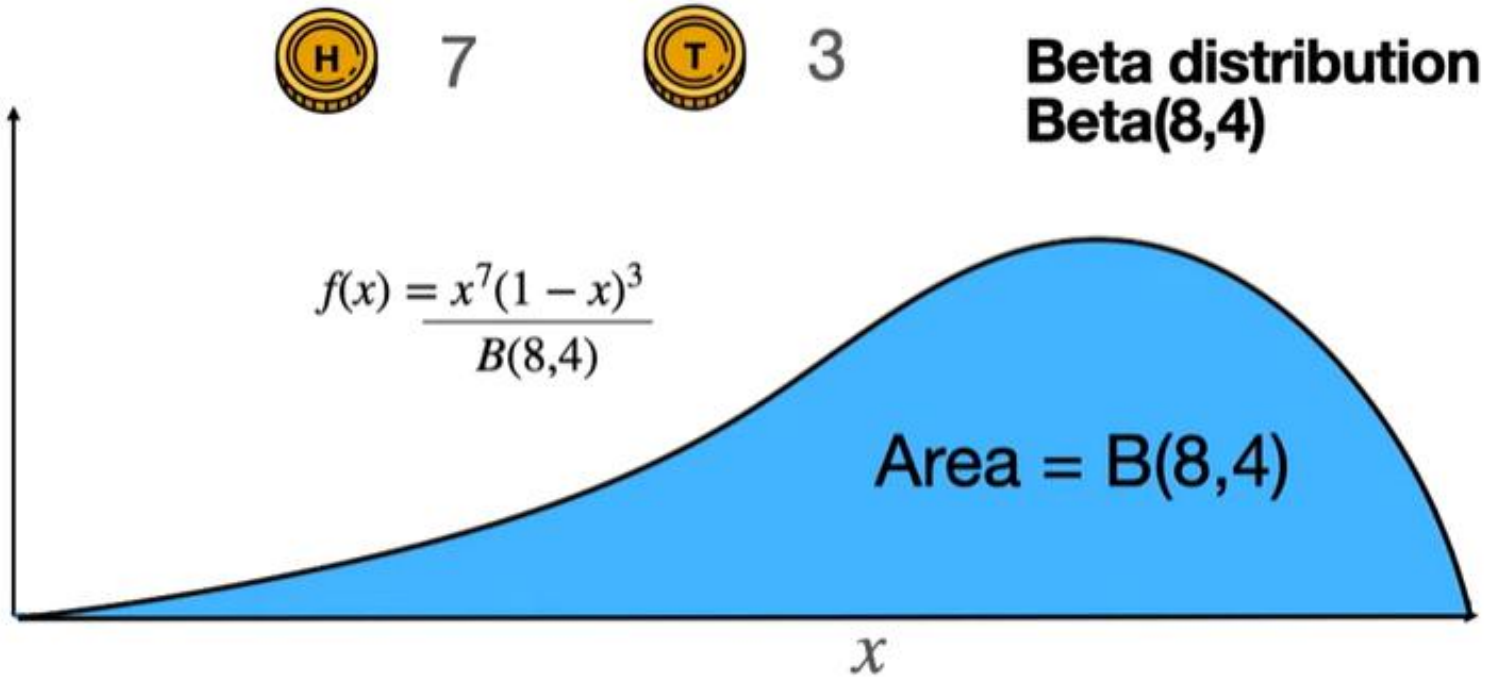
# Beta Distribution

Formula for this curve is



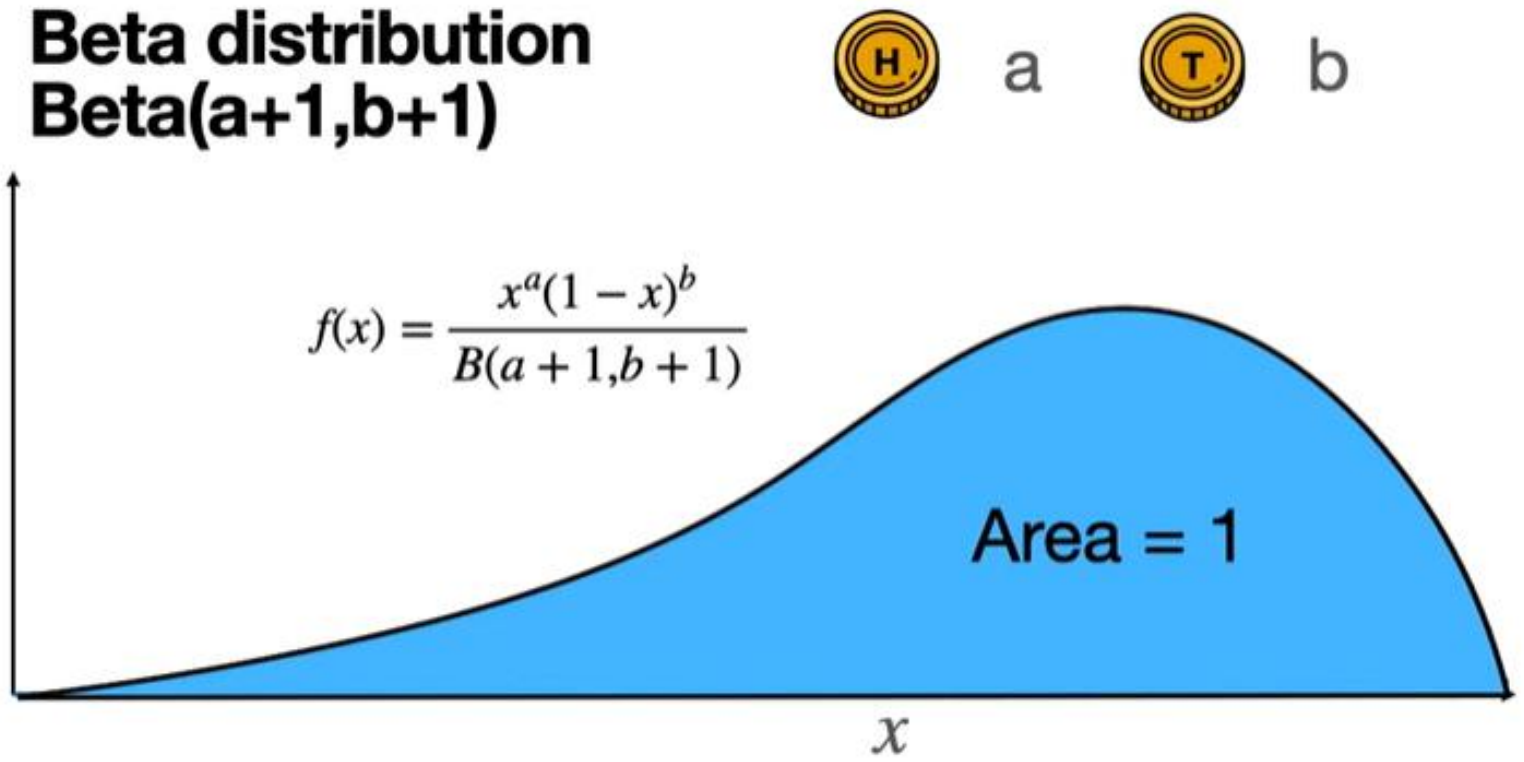
# Beta Distribution

To convert it to a probability distribution, we have to normalize and divide it by Area



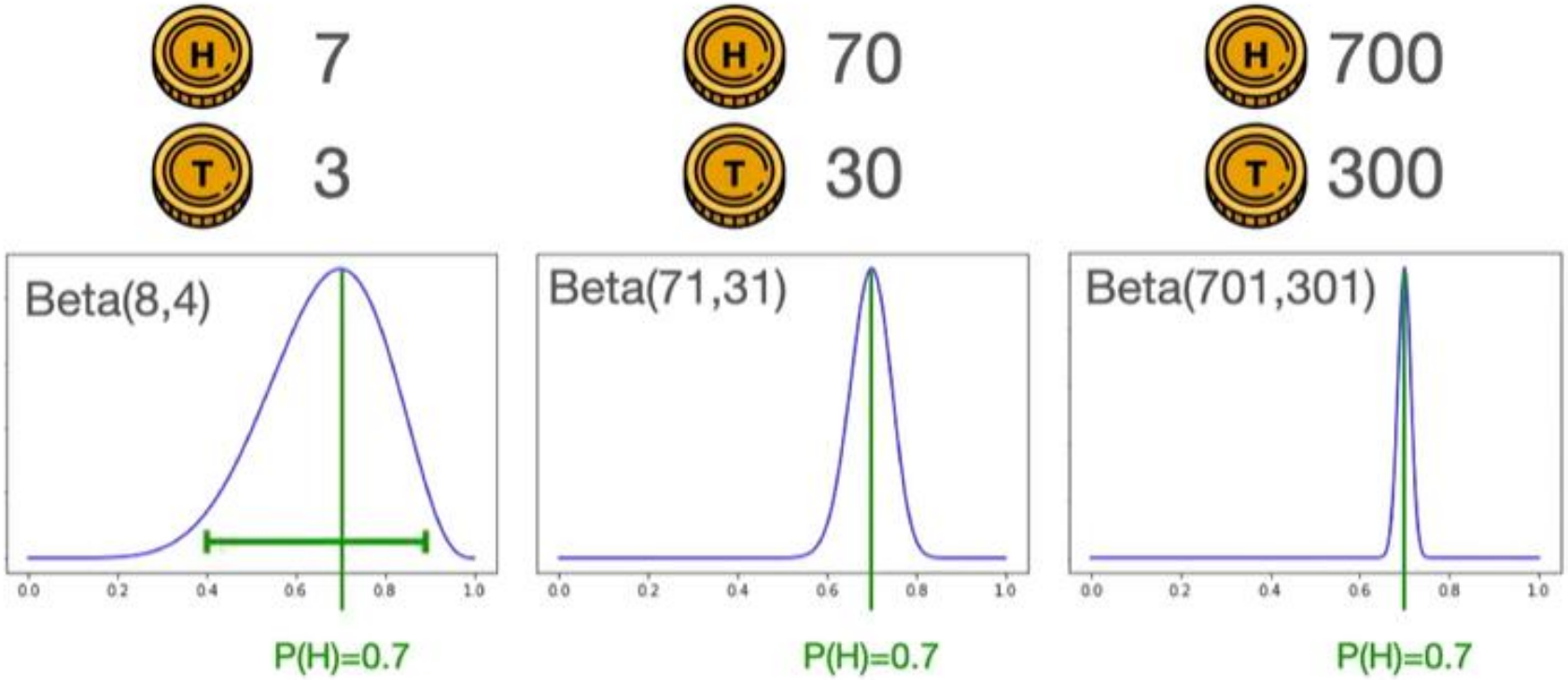
# Beta Distribution

General Formula



# Beta Distribution

We have 3 coins. And we flip them first one 10 items, second one 100 items and third one 1000 times.

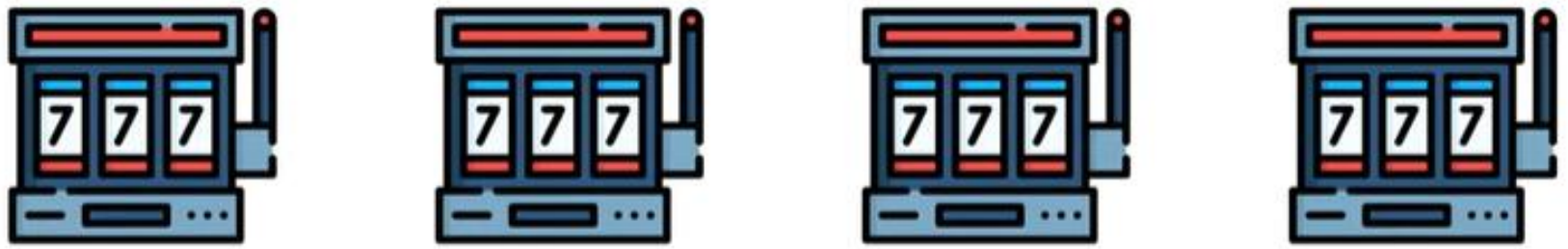


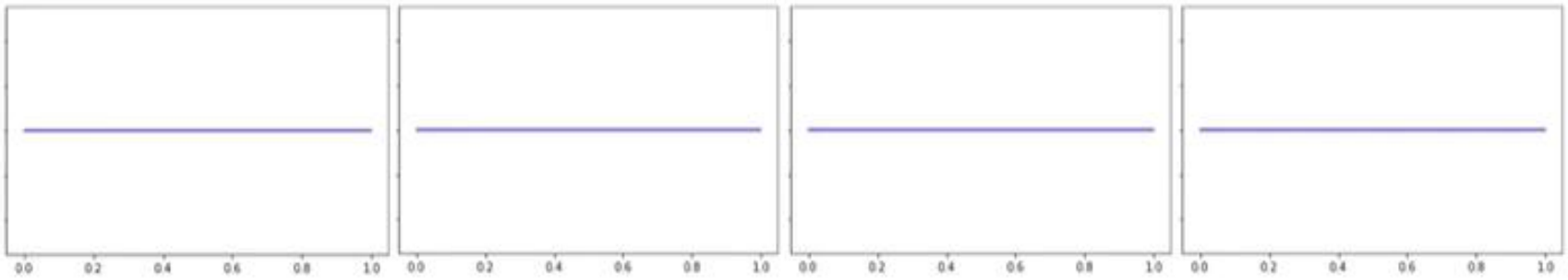
# Thomson Sampling

1. In the context of multi-armed bandit problems, Thomson Sampling is a popular algorithm used for sequential decision-making to determine which machine (or arm) to select in order to maximize the total reward over time.
2. The algorithm makes use of Bayesian probability to balance exploration and exploitation effectively.

# Thompson Sampling

Thompson Sampling initial phase where the Beta distribution is empty



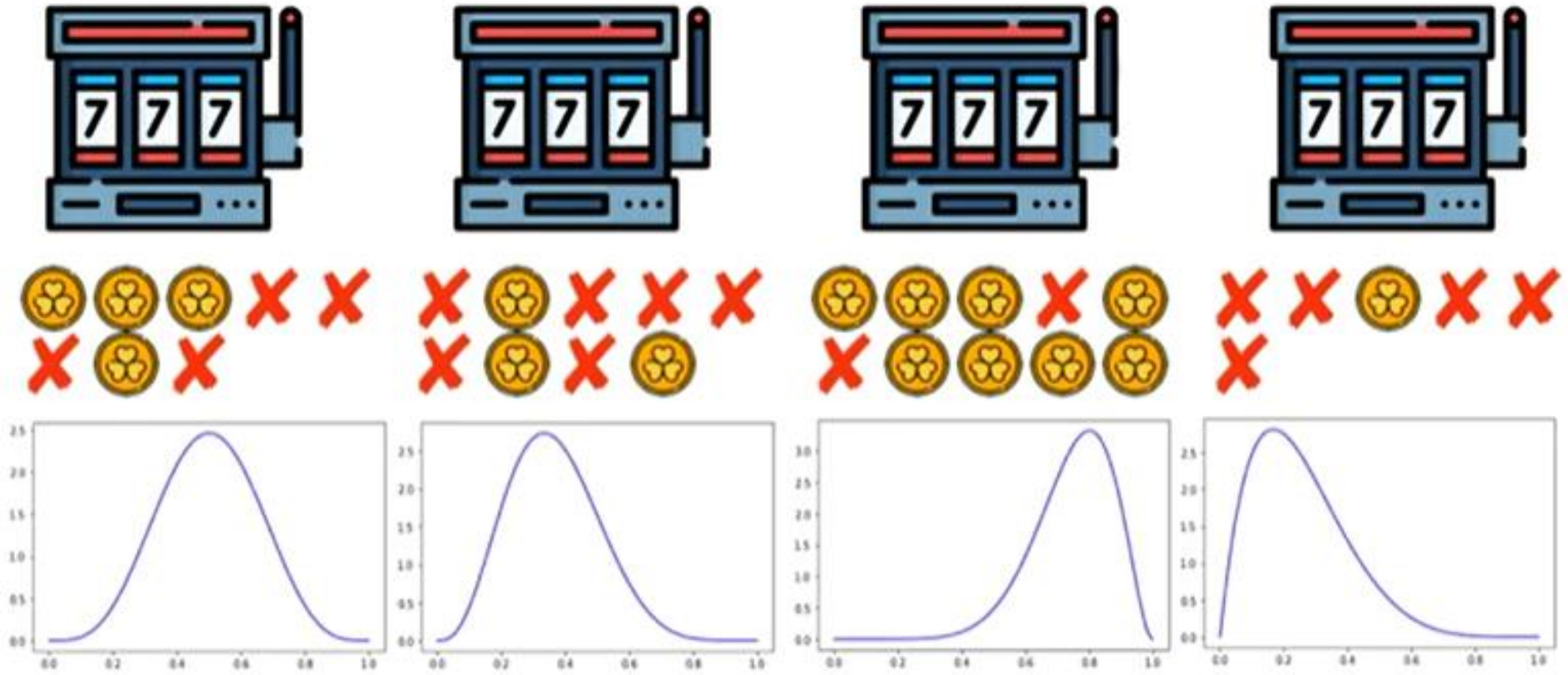


# Thompson Sampling

1. Initially machines are selected using exploration strategy.
2. During this phase, each machine is selected a fixed number of times (k), regardless of the observed rewards.
3. This allows the algorithm to build some initial knowledge about the machines' probabilities.
4. Once the initial phase is completed, the balance between exploration and exploitation is achieved through Beta Distribution.

# Thompson Sampling

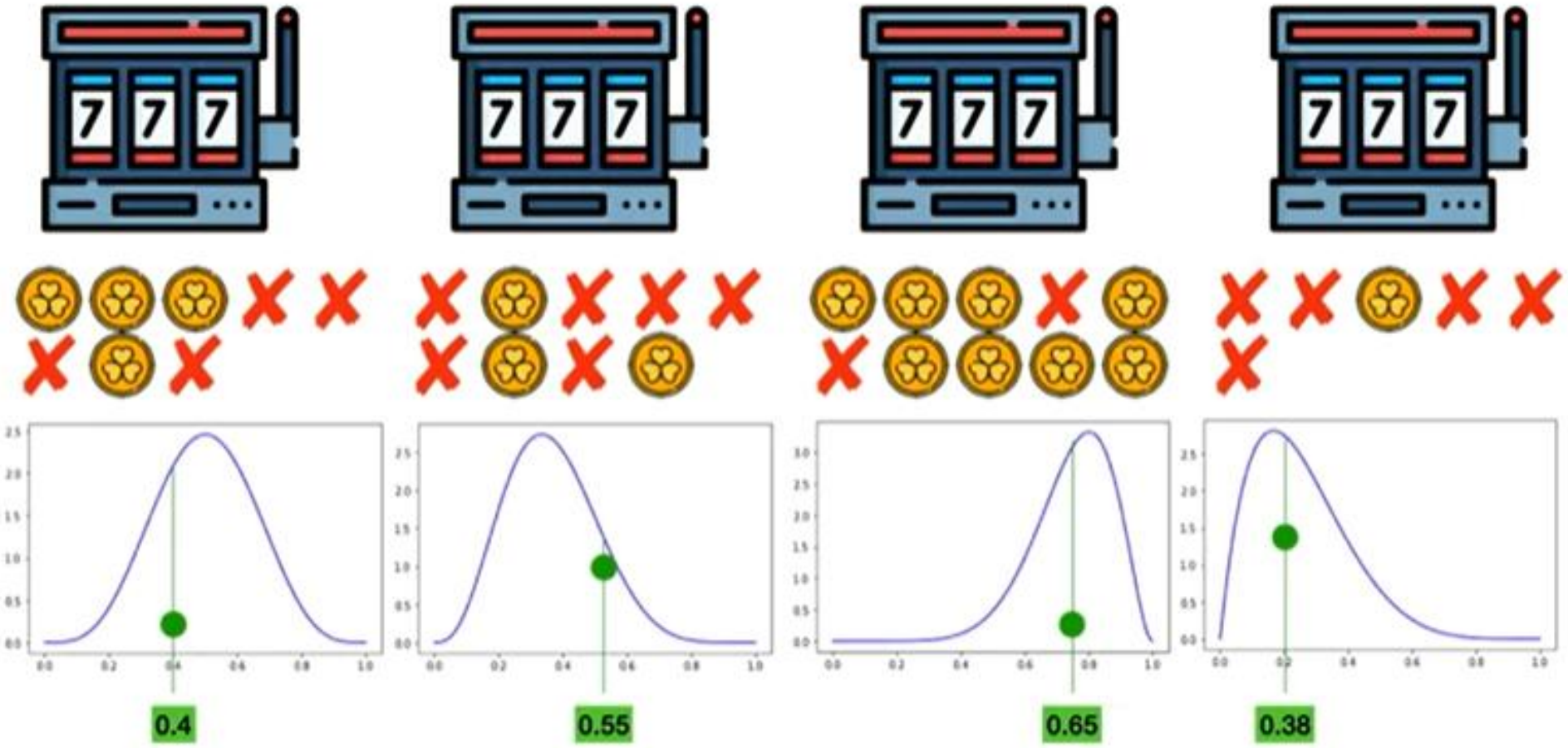
We have played the machines many times and our beta dis



# Thompson Sampling

1. In order to pick the next machine, we're going to make them compete with a little bit of randomness added.
2. The idea of the competition is that the strongest machines should have more probability of winning and the weaker machines should have less probability of winning
3. If we have machines that have not been explored very well, they should also be given a slightly higher probability of winning so that we can explore them more

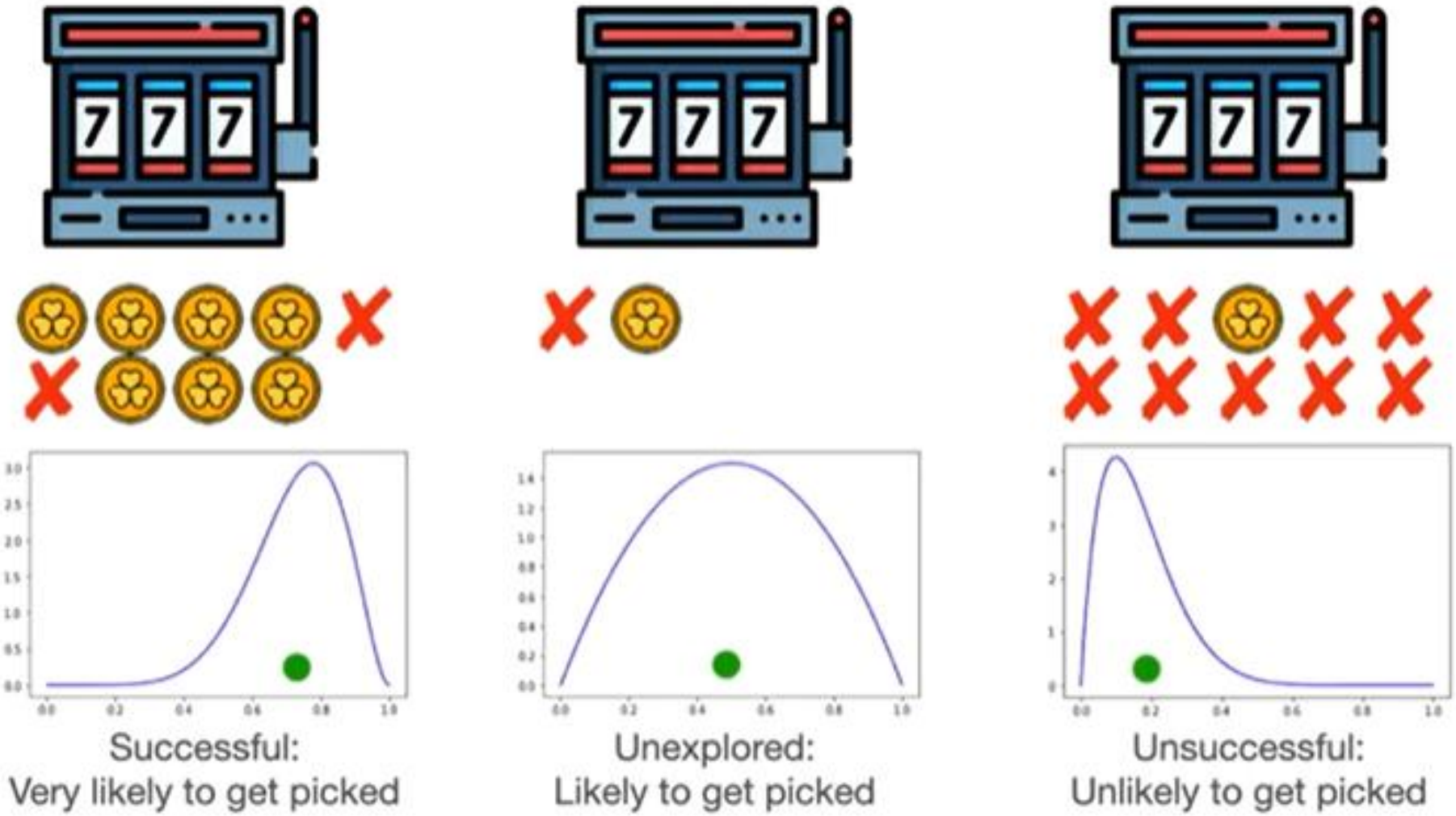
# Thompson Sampling



# Thompson Sampling

1. We are going to pick a random point from each of the beta distributions corresponding to the machines
2. Now we are going to take the value of the x coordinate at that point and the machine that gave us a higher value is the one that wins
3. In this case the third machine, which you can see is the strongest and we are going to play in that machine next
4. Then update the beta distribution and continue

# Thompson Sampling - Summary



# Thompson Sampling - Summary

1. The one in the left has proven to be successful because it has good results.
2. Its beta distribution is heavily skewed towards the right therefore if we pick a random point in the distribution is likely to be a high value.
3. A successful distribution is likely to win at the competition and likely to get picked.

# Thompson Sampling - Summary

1. A machine in the middle that hasn't been explored very well we don't know if it's good or bad yet.
2. Its distribution is very wide so a point underneath it could have a high value or a low value, so this is not as strong as the one on the left.
3. But it's also likely to get picked occasionally.
4. Unexplored machines are also likely to get picked

# Thompson Sampling - Summary

1. The machine on the right that has proven to be a weak machine
2. It has a beta distribution that is heavily skewed towards the left
3. If you pick a random point, it's very likely to give us a small value.
4. So, unsuccessful machines are unlikely to get picked
5. It could still happen, if it returns a high value or the other ones happen to return low values.

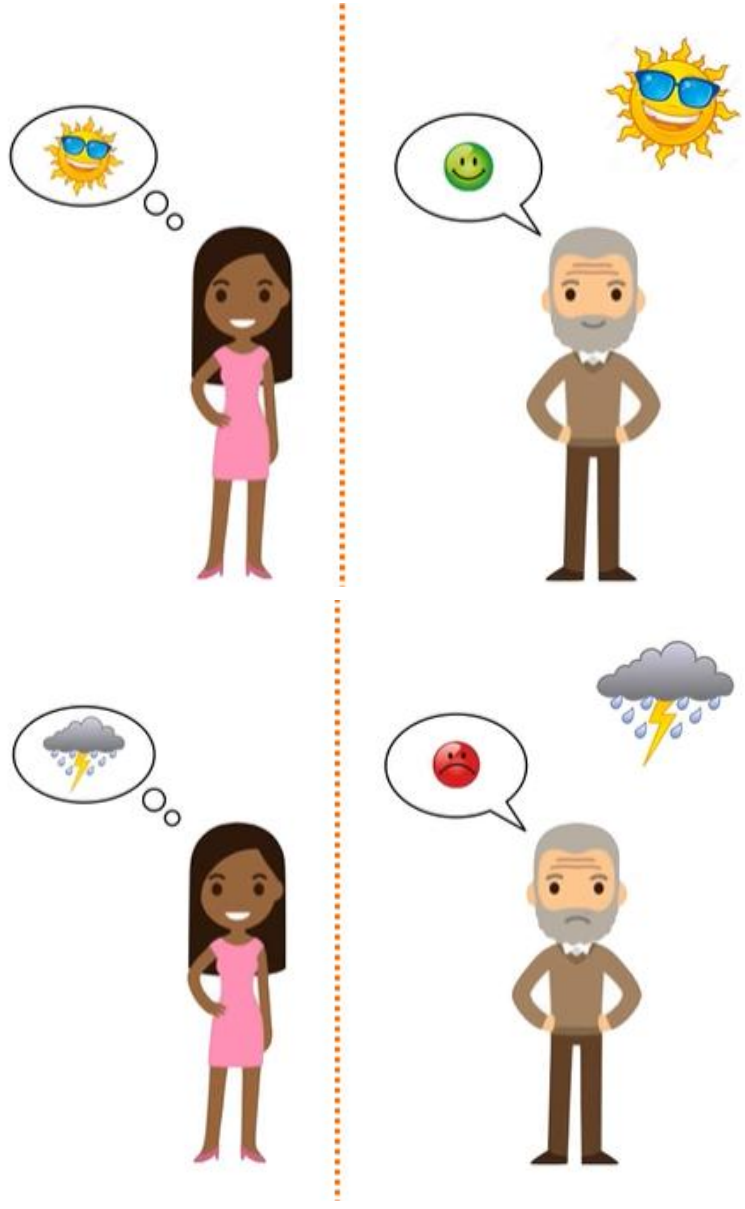
ICTC International Council for Technology Certifications

# Markov Decision Process

Foundations

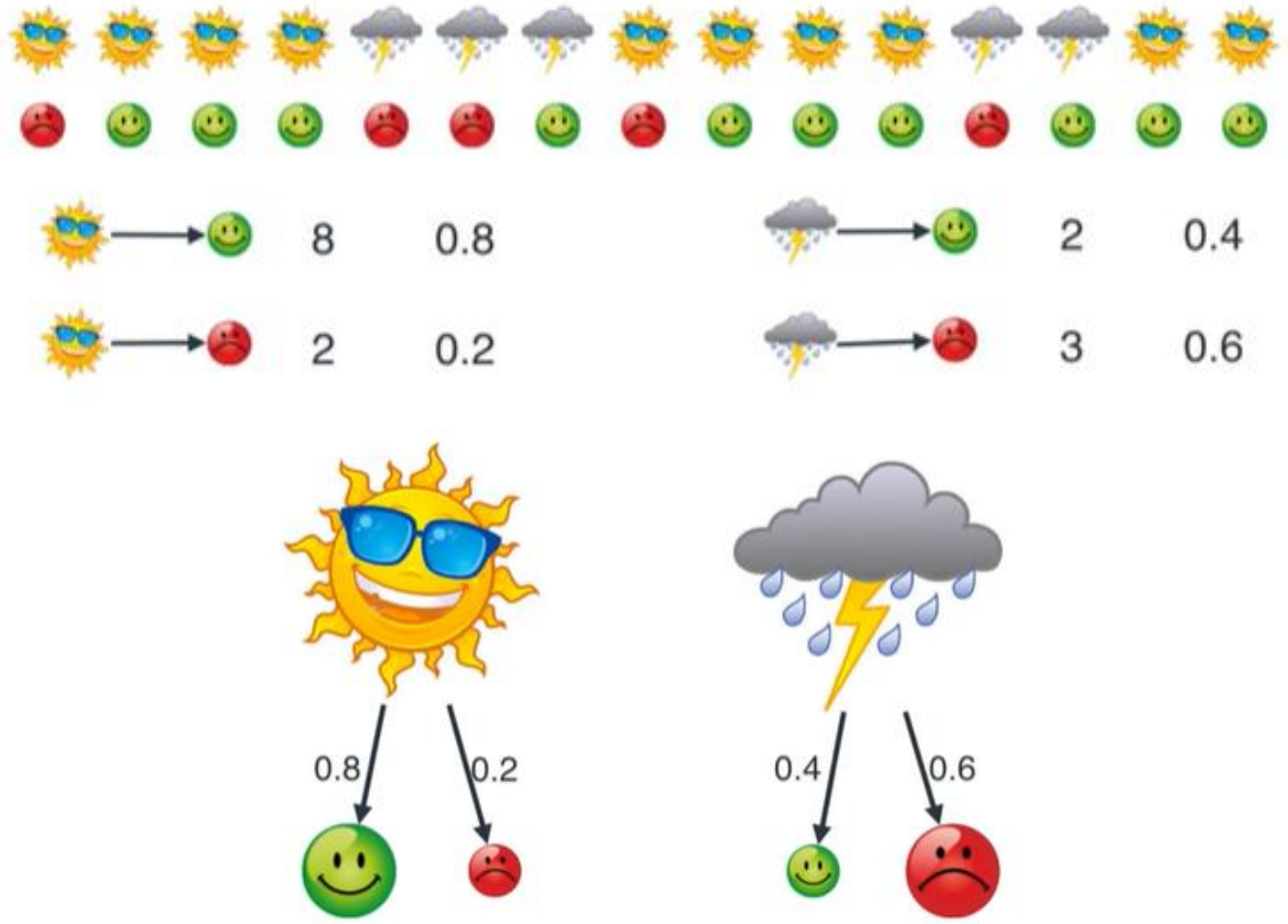
# Hidden Markov Model

1. Alice and Bob live far apart, and they talk on the phone
2. Bob has a mood that changes based on the weather
3. If it's sunny, Bob is happy and tells Alice on the phone he is happy
4. If it's rainy, Bob is grumpy and tells Alice on the phone he is grumpy
5. In summary Bob tells Alice his mood and Alice guesses the weather



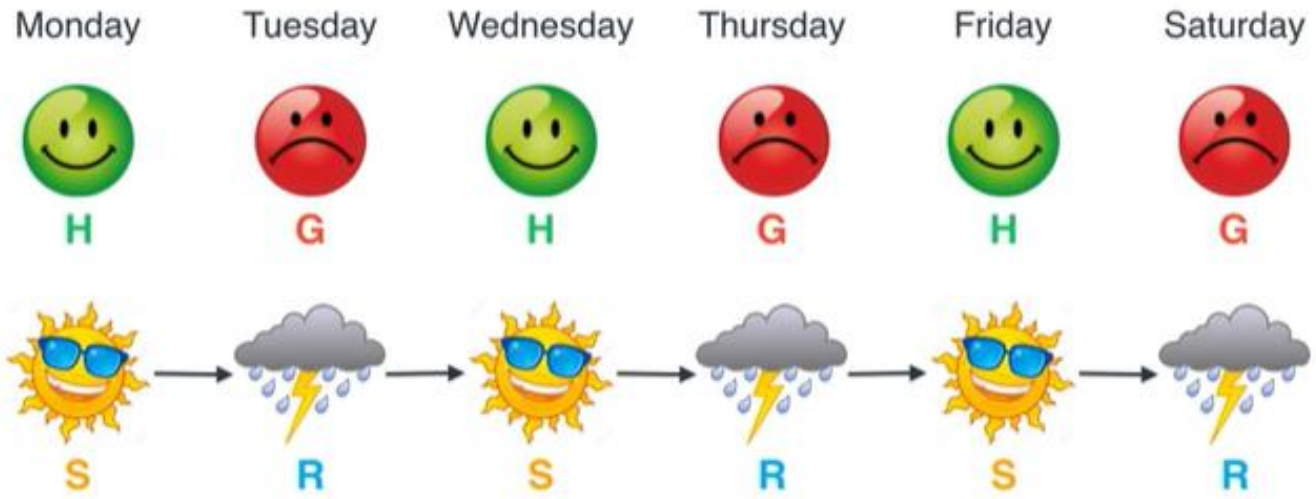
# Hidden Markov Model – Emission

1. Bob is mostly happy when the weather is sunny but there are some exceptions because due to other circumstances
2. Bob is mostly grumpy when the weather is rainy but there are also some exceptions because due to other circumstances



# Hidden Markov Model

Bob says this week was see an emotional roller coaster

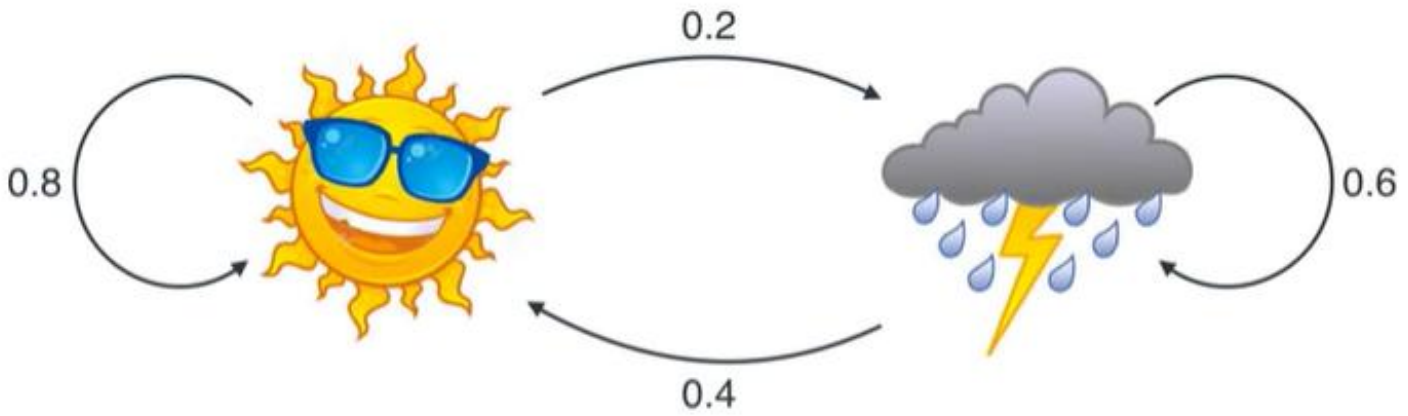



It's kind of strange to have sunny rainy sunny rainy sunny rainy. Usually if a day is sunny then it's likely that the next day will be sunny too. Similarly, if a day is rainy most likely the next day is rainy. We need to figure the probability of this weather Transition.

# Hidden Markov Model - Transitions

Using past data let's see

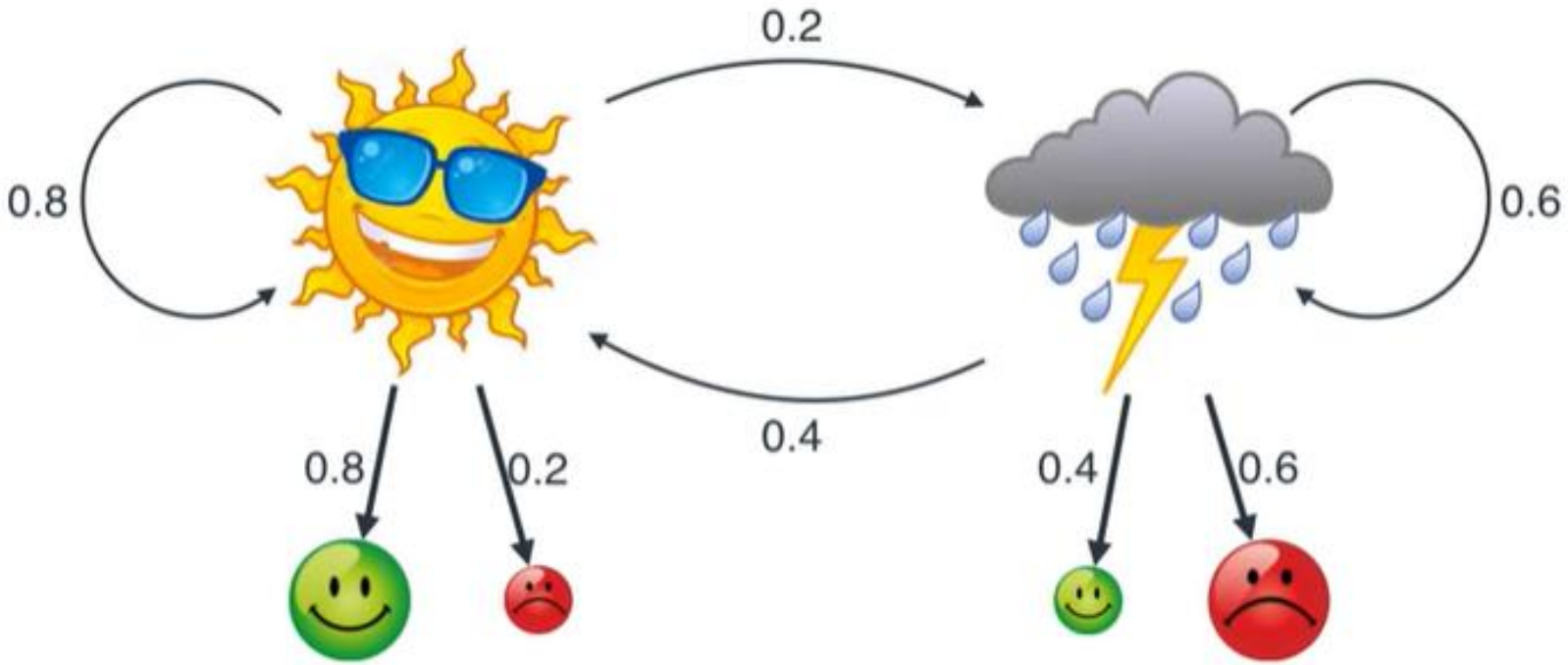
1. how many times a sunny day is followed by sunny
2. how many times a sunny day is followed by rainy
3. how many times a rainy day is followed by sunny
4. how many times a rainy day is followed by rainy

# Hidden Markov Model

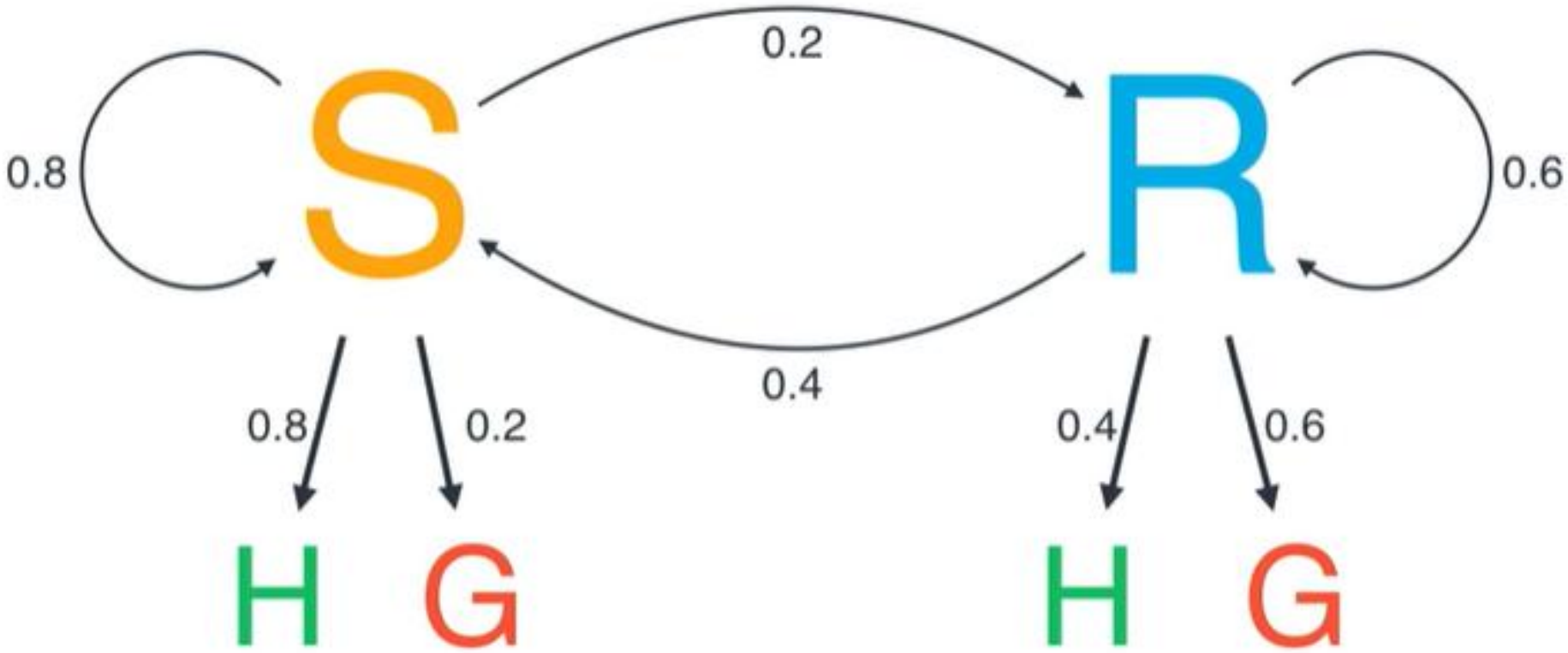
Now we have a complete Markov Hidden Model. In this what we have is 2 types of states

1. Emission / Observation (Alice can only see Bob's mood)
2. Transition (Alice doesn't get to see the weather – hidden states)

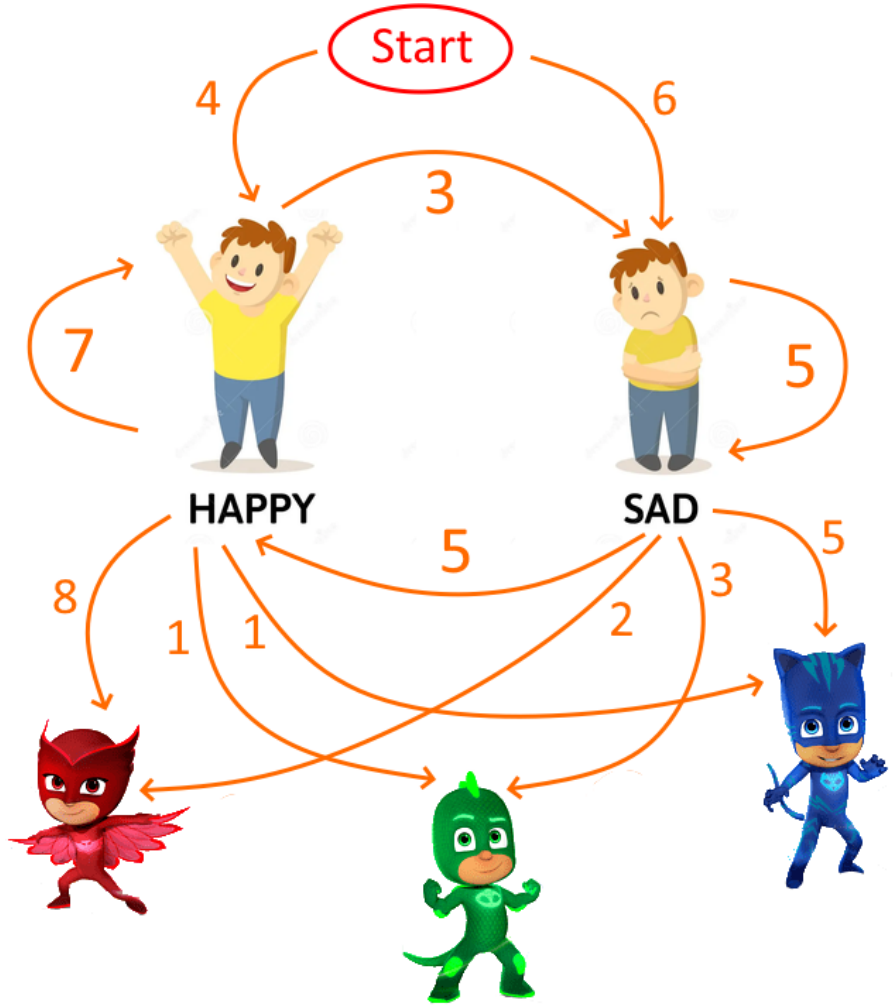


# Hidden Markov Model

If you like letters instead of symbols, then think of this as your hidden Markov model



# Hidden Markov Model – Another Example




1. There are 2 moods happy and sad.
2. In happy mood prefers Red dress and in sad mood prefers Blue dress

| Initial Values | |
|---|---|
| H | S |
| 0.4 | 0.6 |

| Transitions | | |
|---|---|---|
| | H | S |
| H | 0.7 | 0.3 |
| S | 0.5 | 0.5 |

| Emissions | | | |
|---|---|---|---|
| | R | G | B |
| H | 0.8 | 0.1 | 0.1 |
| S | 0.2 | 0.3 | 0.5 |

# Hidden Markov Model - Summary

HMM is a statistical model used to model a system that is assumed to be a Markov process with hidden states. An HMM consists of the following components

1. Hidden states: States that are not directly observable.
2. Observations/emissions: Observable outcomes associated with each hidden state.
3. State transition probabilities: Probabilities of transitioning between hidden states over time.
4. Emission probabilities: Probabilities of emitting certain observations from each hidden state.

# Markov Decision Process

1. Markov decision processes give us a way to formalize sequential decision.
2. This formalization is the basis for problems that are solved with reinforcement learning.

# Markov Decision Process - Components

Let's discuss the components involved in MDP.

1. In MDP we have a decision maker called an **agent** that interacts with the **environment** that it's placed in.
2. These interactions occur sequentially over time.
3. Each time, the agent will get some representation of the environment **State.**
4. Based on this representation that agent selects an **action** to take.
5. The environment is then transitioned into some new state and the agent is given a reward because of its previous action.

# Markov Decision Process - Components

To summarize the components of an MDP include

1. the environment
2. the agent
3. all the possible states of the environment
4. all the actions that the agent can take in the environment
5. all the rewards that the agent can receive from taking actions in the environment

# Markov Decision Process - Goal

1. This process of selecting an action from a given state transitioning to a new state and receiving a reward happens sequentially repeatedly which creates something called a trajectory (path).
2. Throughout the process it's the agent’s goal to maximize the total amount of rewards that it receives from taking actions and given states of the environment.
3. This means that the agent wants to maximize not just the immediate reward but the **cumulative rewards** that it will receive over time.

# Markov Decision Process - Math

1. In an MDP we have a set of states S a set of actions A and a set of rewards R. We will assume that each of these sets has a finite number of elements.
2. At each time step t = 0, 1, 2…. that agent receives some representation of the environment's state $S_t \in S$. Based on this state the agent selects an action $A_t \in A$. This gives us the state action pair $(S_t, A_t)$.

# Markov Decision Process - Math

3. Time is then incremented to the next time step t + 1, and the environment is transitioned into a new state $S_{t+1} \in S$. At this time the agent receives a numerical reward $R_t \in R$ for the action $A_t$ taken from the previous state $S_t$.
4. We can think of the process of receiving a reward as an arbitrary function that map's state action pairs to rewards. At each time t, we have

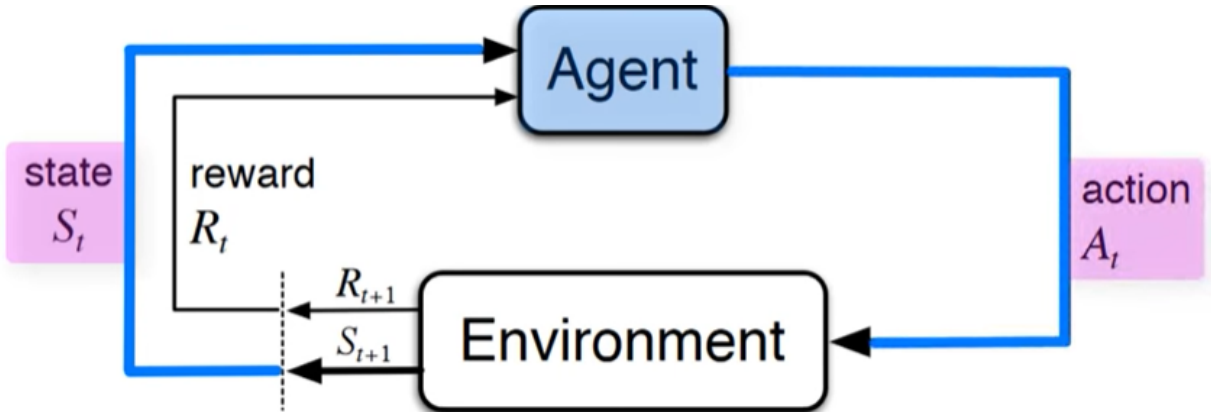
   **$f(S_t, A_t) = R_t + 1$**

# Markov Decision Process - Math

The trajectory representing the sequential process of selecting an action from a state and then transitioning to a new state and receiving a reward can be represented as

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \cdots$$

# Markov Decision Process - Diagram




1. Step 1 At time t, the environment is in $S_t$
2. step 2 The agent observes the current state and selects action a t
3. step 3 The environment transitions to state $S_{t+1}$ and grants the agent reward $R_{t+1}$
4. This process then starts over for the next time $S_{t+1}$ now.

# Exercise using GYM library