An introduction to contextual bandits

Wilson Pok

Goal and rules of the game

Sequential decision-making under uncertainty

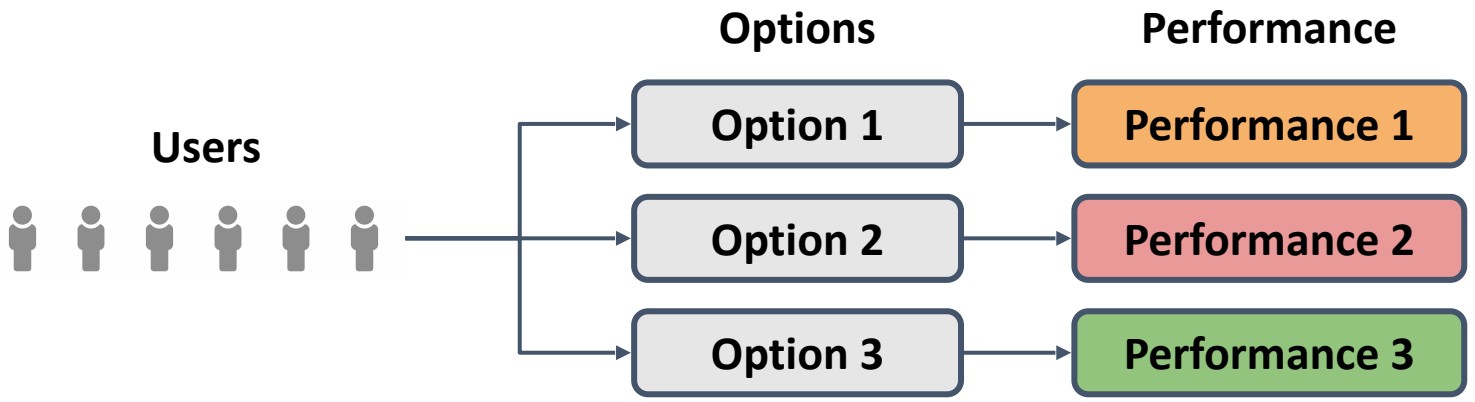
Each round is independent

Learn by trying - but it costs to try

Only see results of decisions made

How do you quickly and confidently learn the best decision each round?

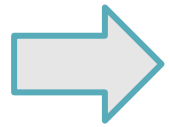
We know how experimentation works



We know how machine learning works

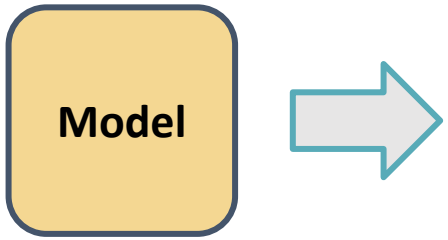
Create model from training data

	Features					Target
	x11	x12	x13	x14	x15	N
	x21	x22	x23	x24	x25	Y
	x31	x32	x33	x34	x35	Y
	x41	x42	x43	x44	x45	N



Use model to predict new data

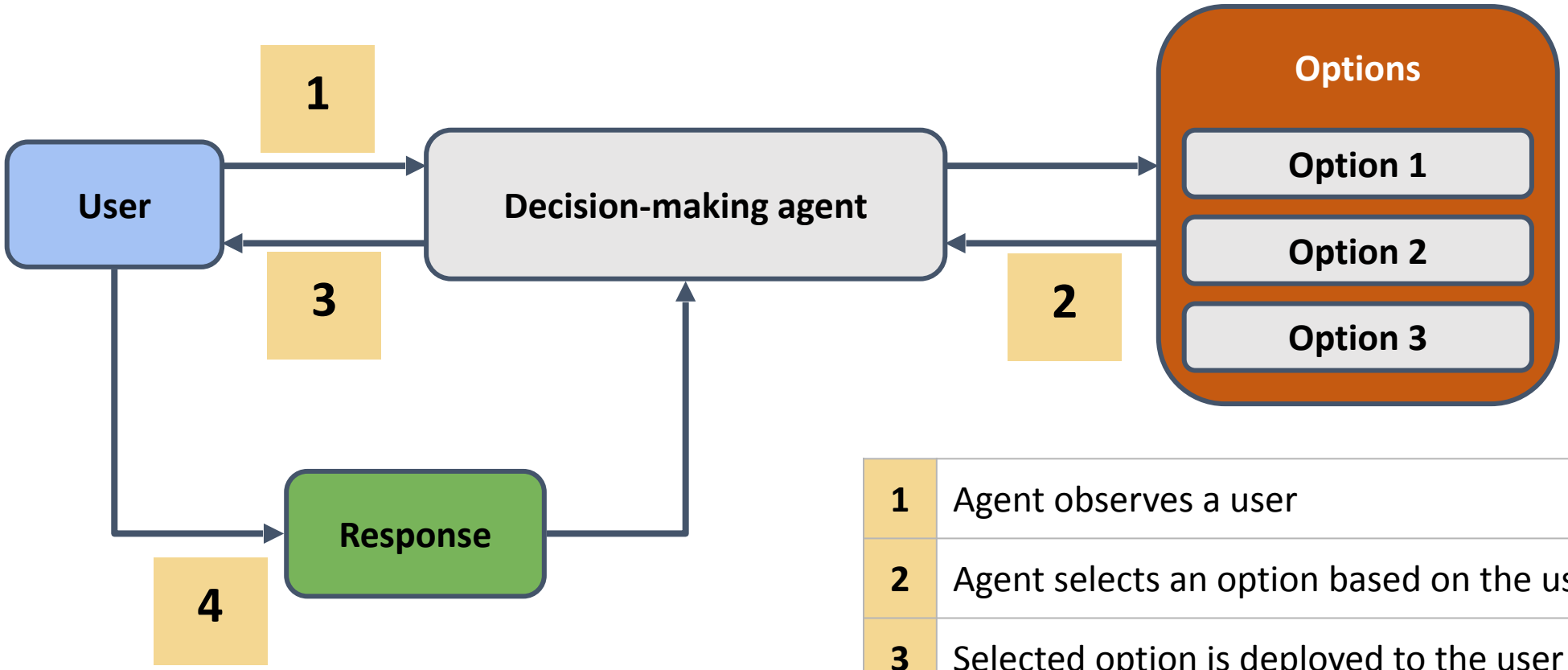
	x51	x52	x53	x54	x55	?
	x61	x62	x63	x64	x65	?
	x71	x72	x73	x74	x75	?
	x81	x82	x83	x84	x85	?



Predictions

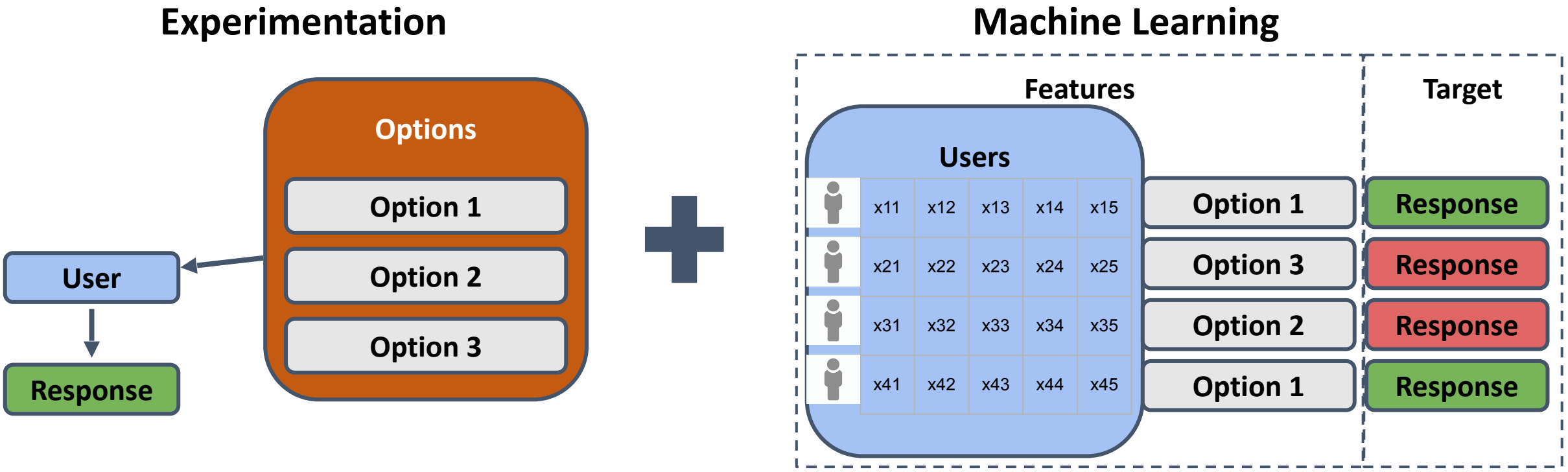
Y
Y
N
N

How real-time decision-making works



1	Agent observes a user
2	Agent selects an option based on the user
3	Selected option is deployed to the user
4	User provides a response, which the agent learns from

This combines experimentation with ML



Explore/exploit tradeoff

Experimentation

On one hand, we need experimentation to ***explore*** our options.

By trying all options on different types of users, we can generate enough training data confidently predict their responses.

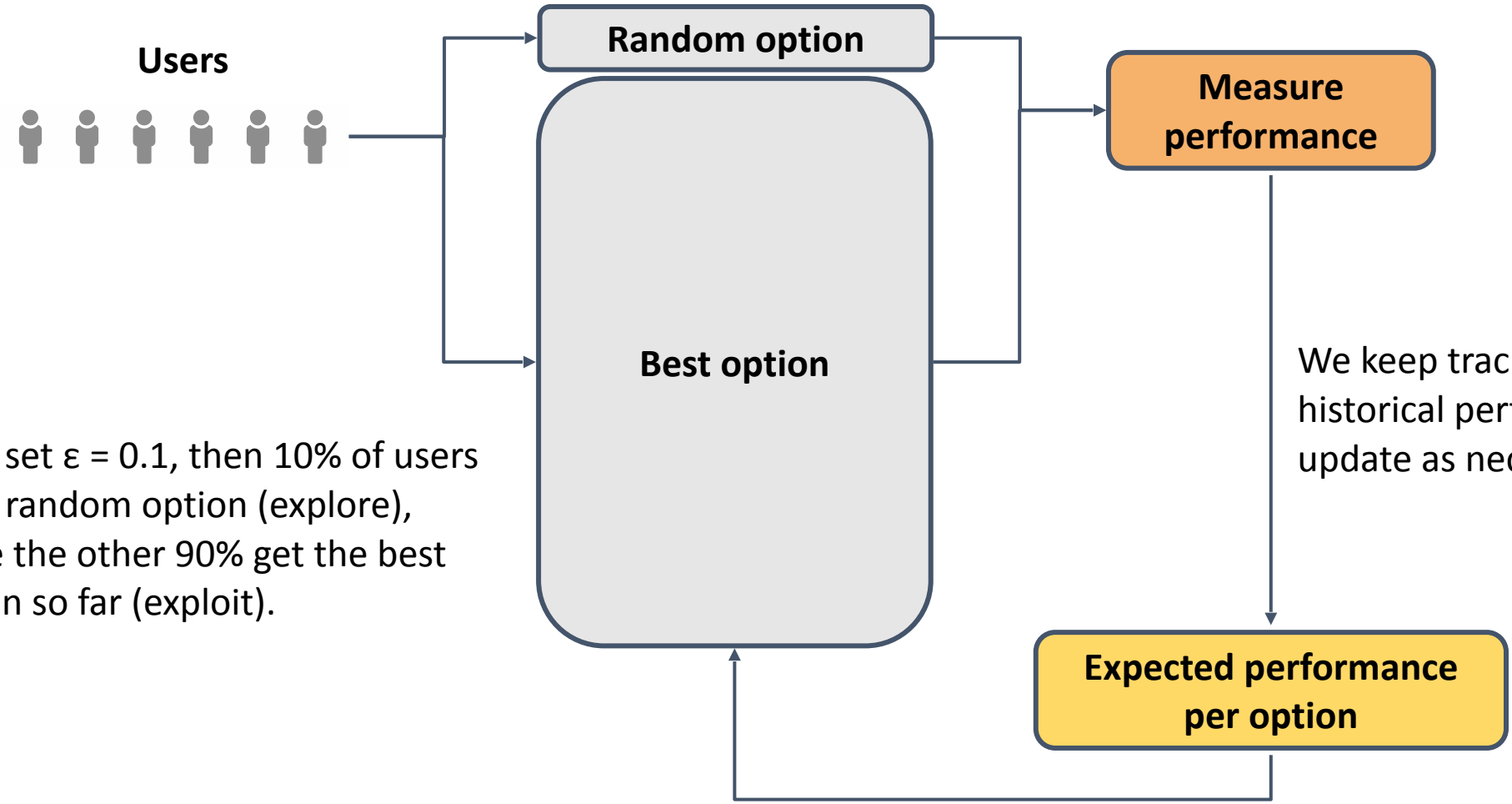
If only there were some way to tune the amount the decision-making agent explores/exploits...

Machine Learning

On the other hand, we need to use the machine learning model to ***exploit*** what we know.

If we are confident that a particular option is best for a certain type of user, then why would we give that user anything else?

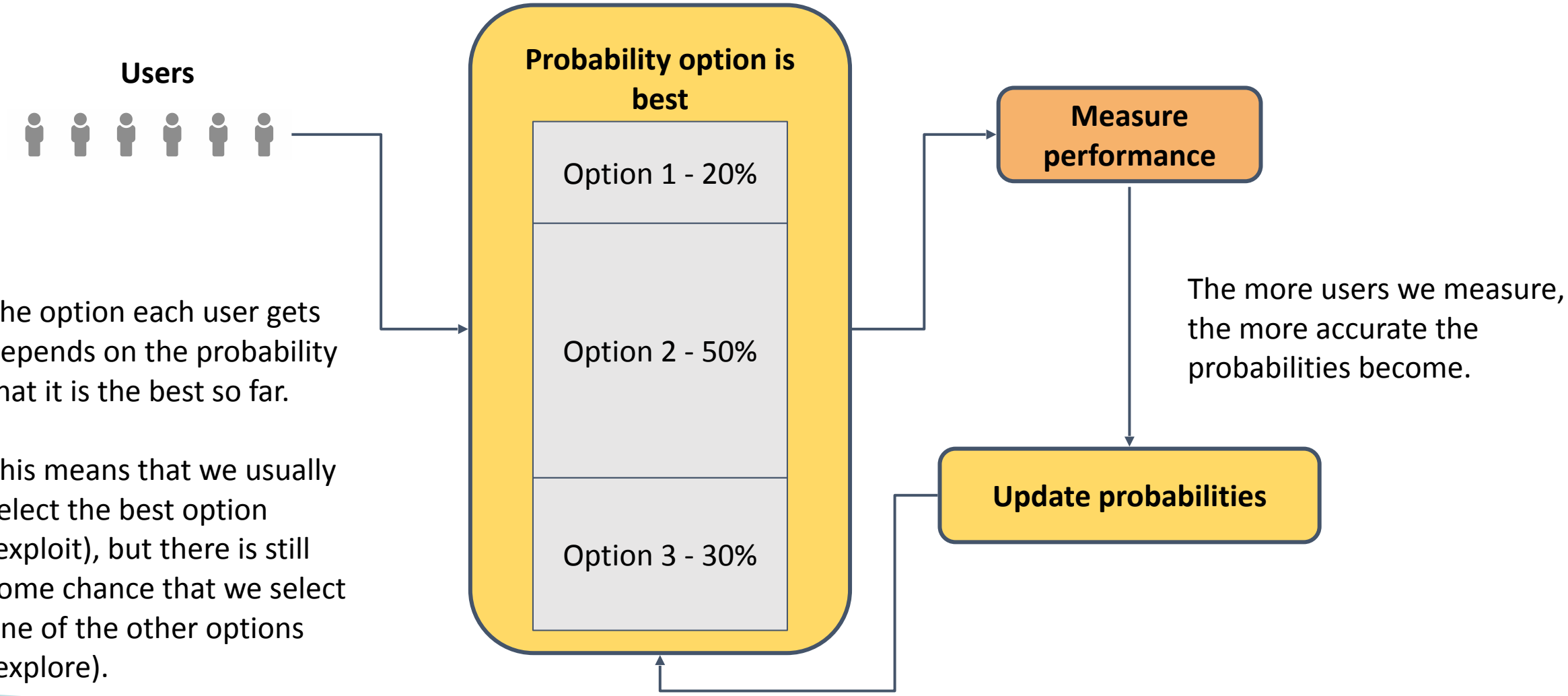
A simple approach is ε-greedy



If we set ε = 0.1, then 10% of users get a random option (explore), while the other 90% get the best option so far (exploit).

We keep track of each option's historical performance and update as necessary.

Another approach is Thompson sampling



The option each user gets depends on the probability that it is the best so far.

This means that we usually select the best option (exploit), but there is still some chance that we select one of the other options (explore).

The more users we measure, the more accurate the probabilities become.

Practical applications

Recommender systems	Recommending news articles to readers on a news website, personalising marketing content
Healthcare	- Adaptive allocation in clinical trials - give more patients treatments that look more promising - Dosing strategies - finding the right dose for drugs that are highly variable depending on the patient
Finance	Automated strategies for constructing portfolios that balance risk and reward
Dynamic pricing	Automated strategies for maximising revenue based on observed demand
Influence maximisation	Maximizing the number of users that become aware of a product by selecting a set of seed users to expose the product to. The agent finds the best influencers in a social network by repeatedly interacting with it.
Dialogue systems	Creating conversational agents that can learn what the best response is for a given input

Current implementations

Microsoft Azure

[Personalizer]

Adobe Target

[Automated Personalization]

Optimizely

[Personalization]

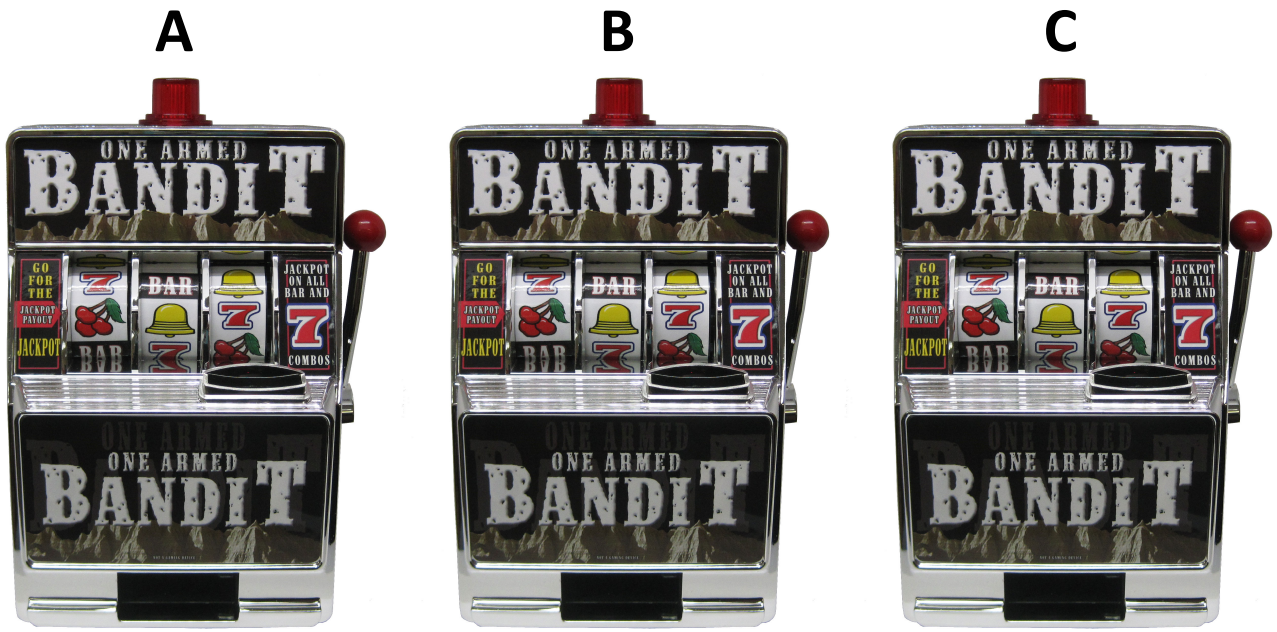
Etymology

Multi-armed bandits

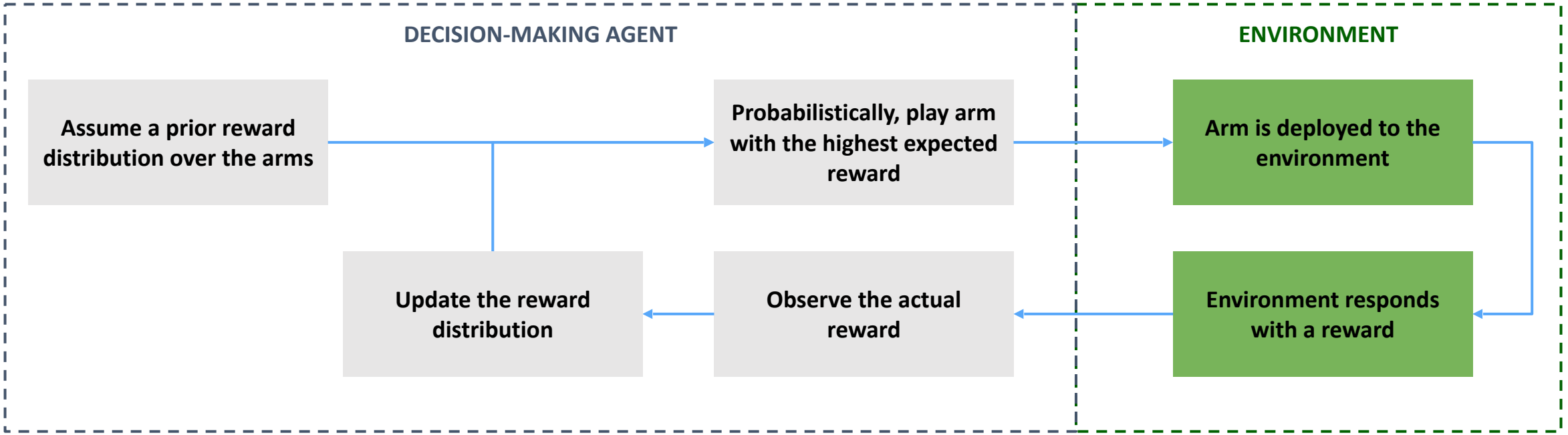
You are given a row of slot machines, each with different payout rates. The only way to estimate the payout rate of a slot machine is to play it.

How do you maximise your winnings?

Specifically, how do you identify the best arm to play **quickly and confidently**?

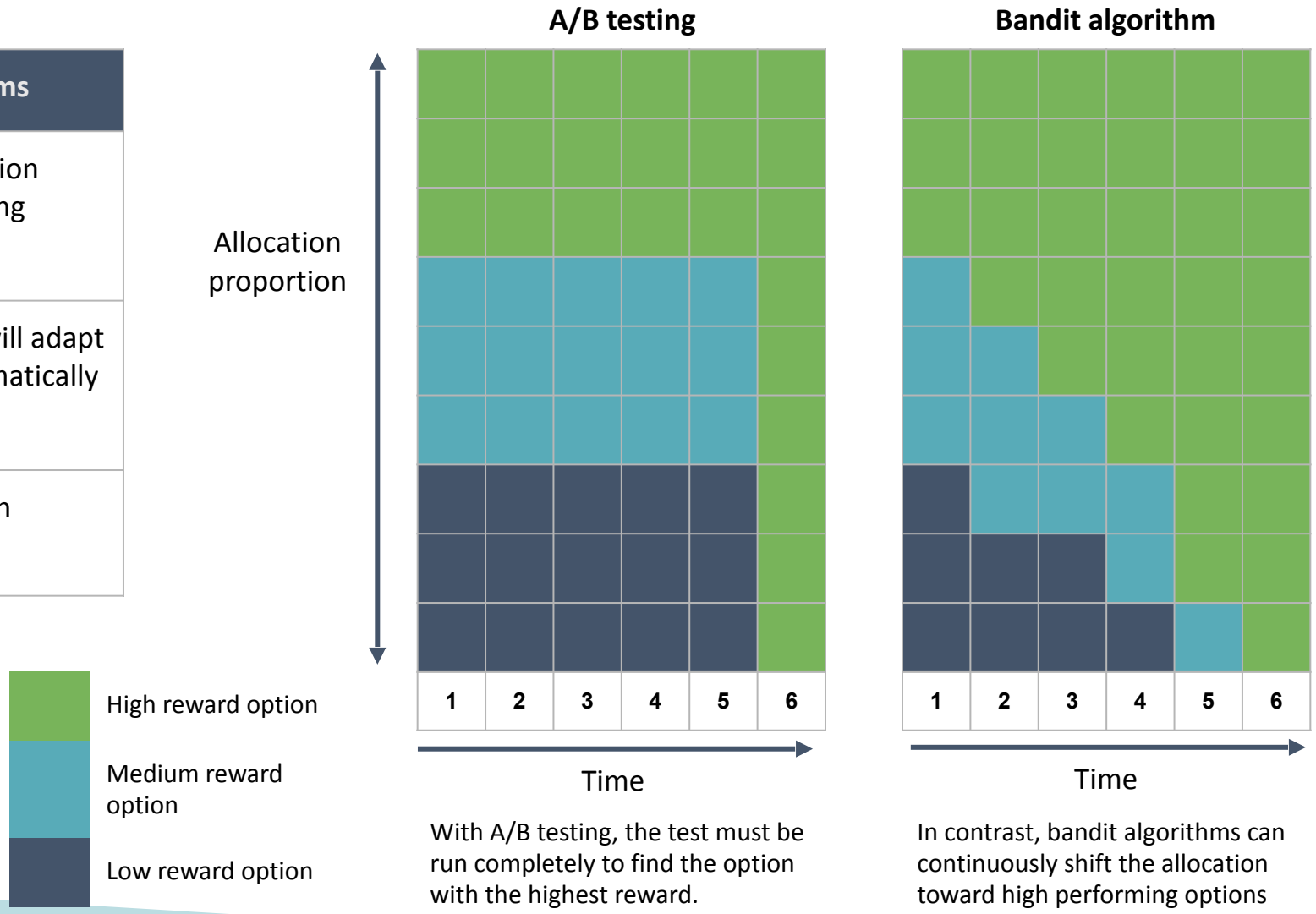


Learn by interacting with environment



Bandits automate A/B testing

A/B testing	Bandit algorithms
Need to wait for a final answer (losing reward until then)	Converges to best option throughout run (gaining reward continuously)
Needs to be rerun if environment changes	Can be left running, will adapt to environment automatically
Can only find the best overall option	Can do personalisation (***contextual bandits***)



With A/B testing, the test must be run completely to find the option with the highest reward.

In contrast, bandit algorithms can continuously shift the allocation toward high performing options faster.

Contextual bandits

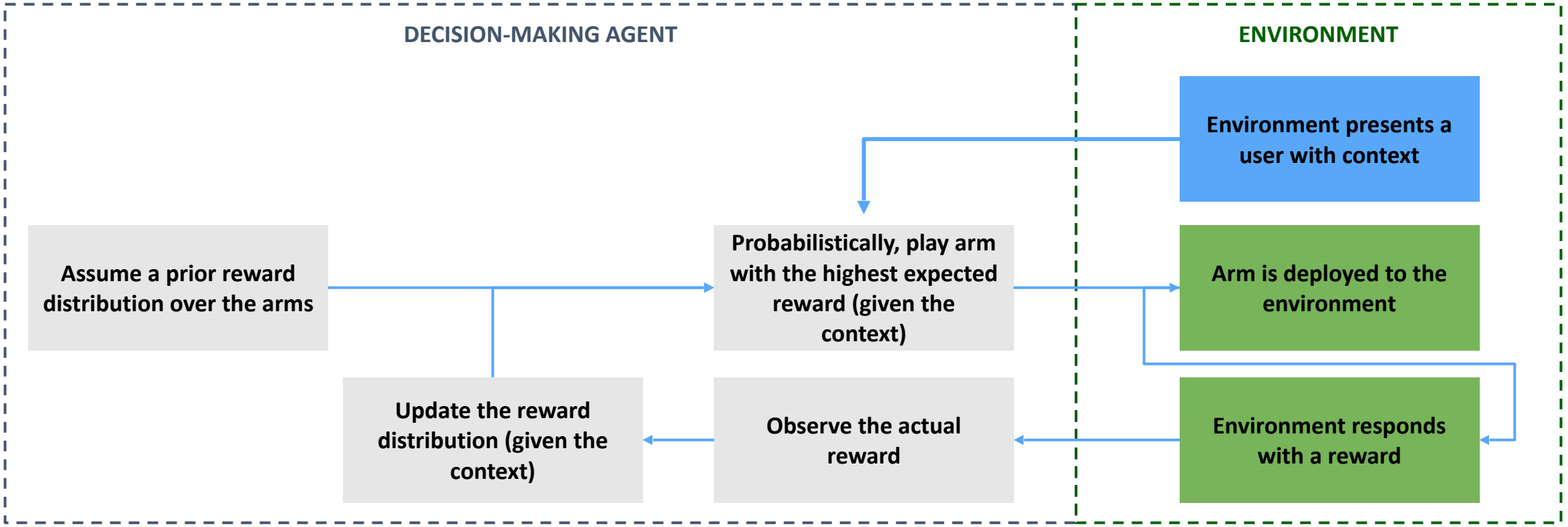
Now suppose that payout rates change depending on the attributes of the player (i.e. rewards depend on ***context***). We have a series of players (with different attributes) who will play one arm each, sequentially.

How do we maximise the overall winnings?

Why do we care about this?
Solving this problem allows us to tailor the arm to fit each particular context.

Specifically, contextual bandits enable personalised actions to be applied to individual users.

Learn by interacting with environment (CB)



Multi-armed vs contextual bandits

Multi-armed bandits	Contextual bandits
Automated A/B testing	Automated personalisation for each user
Decisions are made based on past performance of each option	Decisions are made based on past performance of each option on each user type
Aims to give most users the best overall option	Aims to give most users the best option for them