

From Policy Learning to Autonomous Algorithm Design: AI for Network Optimization
Universidad Carlos III de Madrid

Learning to decide in networks: from multi-armed bandits to contextual bandits

2026-09-21

Jose A. Ayala Romero · NEC Laboratories Europe

Agenda

1. Introduction: decisions in networks
2. Multi-armed bandits: the technique
3. Multi-armed bandits: networking examples
4. Contextual bandits: the technique
5. Contextual bandits: vrAI
6. Summary

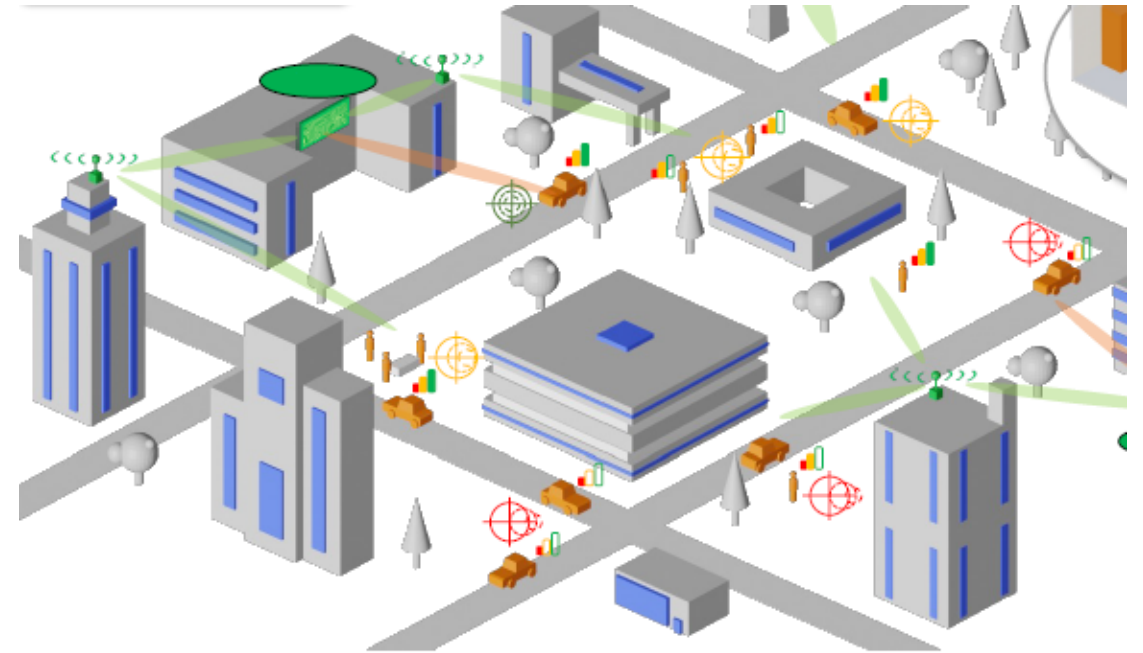
Introduction: decisions in networks

Why online learning is a natural fit for network configuration problems

Networks are full of decisions nobody can model exactly

Configuration knobs everywhere, and a performance response that is unknown, noisy and non-linear

- **Interference coordination:** how many subframes should a macro cell mute, and how far should small cells extend their range?
- **Computing for virtual base stations:** how much CPU does this cell need under this traffic and this channel?
- **Sleep modes:** how long can a radio unit sleep before a delay budget breaks?
- **Semantic image transmission:** how much of an image must reach a satellite link's receiver intact?



Model-driven versus data-driven optimisation

Model-driven

- Write an analytical model of the system
- Optimise the model, apply the result
- Fast and explainable, but the model's assumptions (traffic, topology, scheduler, hardware) rarely hold in production
- The optimum of the model is not the optimum of the network

Data-driven

- Apply a configuration, measure the KPI, learn, repeat
- No model of the network is needed
- The price: every trial runs on real users, so bad trials cost performance
- The whole seminar is about paying that price wisely

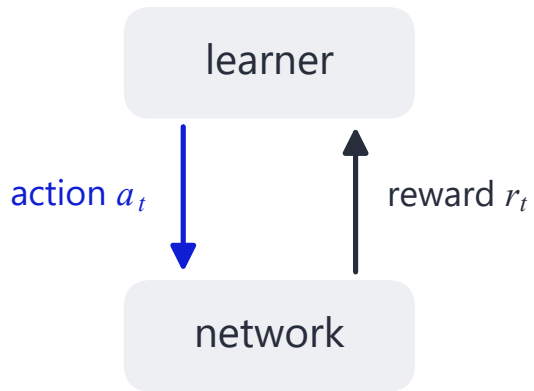
Four things that make learning in networks hard

- **Unknown response**: the KPI as a function of the configuration is not available in closed form
- **Noisy observations**: every measurement is a random sample, not the mean
- **Non-stationarity**: traffic, users and topology change, so the optimum moves
- **Constraints**: reliability, delay or power limits must hold *while* we learn, not only at the end

Each of the four returns as a technique: bandits, confidence bounds, change detection and stopping rules, and (tomorrow's first topic) safe and risk-aware learning.

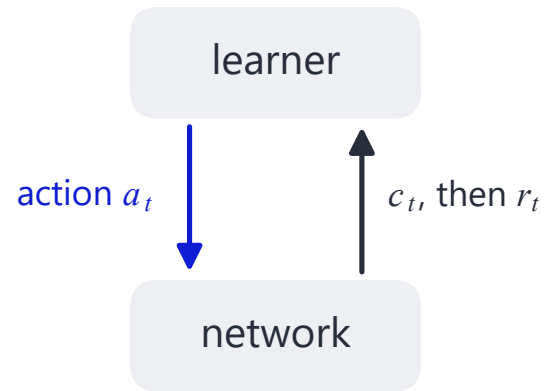
The three settings we will cover

Multi-armed bandit



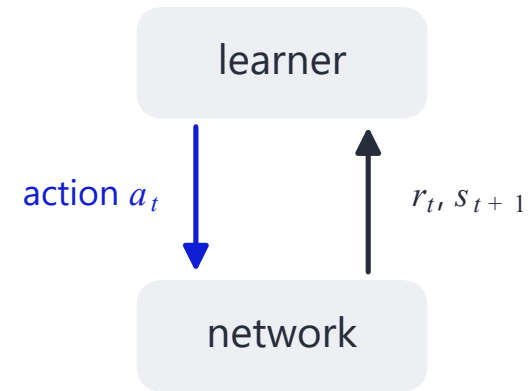
only the reward of the pulled arm is seen

Contextual bandit



context c_t observed before acting; it does not depend on the previous actions

Reinforcement learning



state s_{t+1} depends on (s_t, a_t) ; reward may be delayed

Every technique is presented in the same four steps

1. **The problem it solves** and the assumptions it needs
2. **The formulation:** what is observed, what is chosen, what is optimised
3. **The algorithm**, as a short recipe rather than pseudo-code
4. **Networking examples** from our own work, each mapped to the formulation

Agenda

1. Introduction: decisions in networks
2. **Multi-armed bandits: the technique**
3. Multi-armed bandits: networking examples
4. Contextual bandits: the technique
5. Contextual bandits: vrAI
6. Summary

Multi-armed bandits: the technique

The simplest setting in which exploration has a price

Which slot machine should I play next

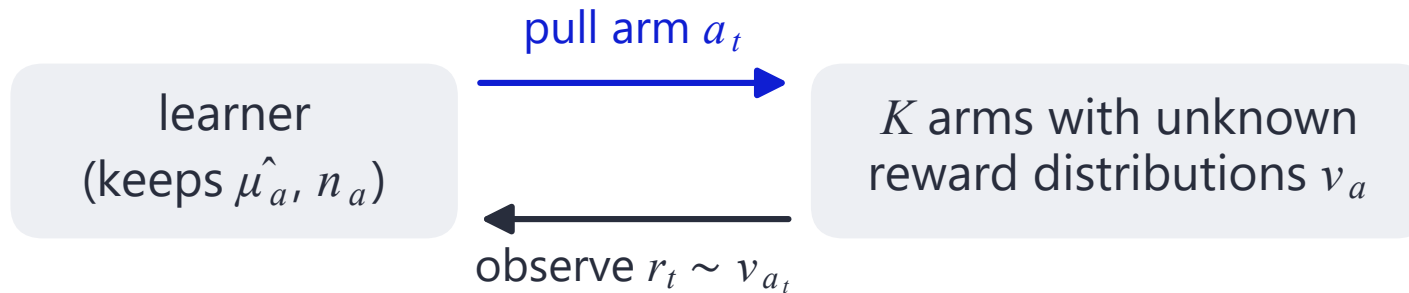
Each machine pays from its own unknown distribution, and I want the largest total payout



- Four machines, one lever each; every pull returns a random amount
- After a few pulls one machine looks best. Two options:
 - **exploit**: keep pulling the best-looking machine
 - **explore**: try the others in case my estimate is wrong
- The **exploration/exploitation dilemma** is the whole problem

In a network, the machines are configurations and the payout is a KPI measured on real users.

The bandit protocol



- K arms, each with an unknown reward distribution ν_a and mean μ_a
- In round $t = 1, \dots, T$: choose $a_t \in \mathcal{A}$, observe $r_t \sim \nu_{a_t}$
- **Only the pulled arm reveals anything:** no feedback on the others
- Goal: maximise the expected total reward $\mathbb{E}\left[\sum_{t=1}^T r_t\right]$

Regret measures what exploration costs

Compare the learner with the oracle that always plays the best arm

$$R_T = T \mu^* - \mathbb{E} \left[\sum_{t=1}^T \mu_{a_t} \right], \quad \mu^* = \max_a \mu_a$$

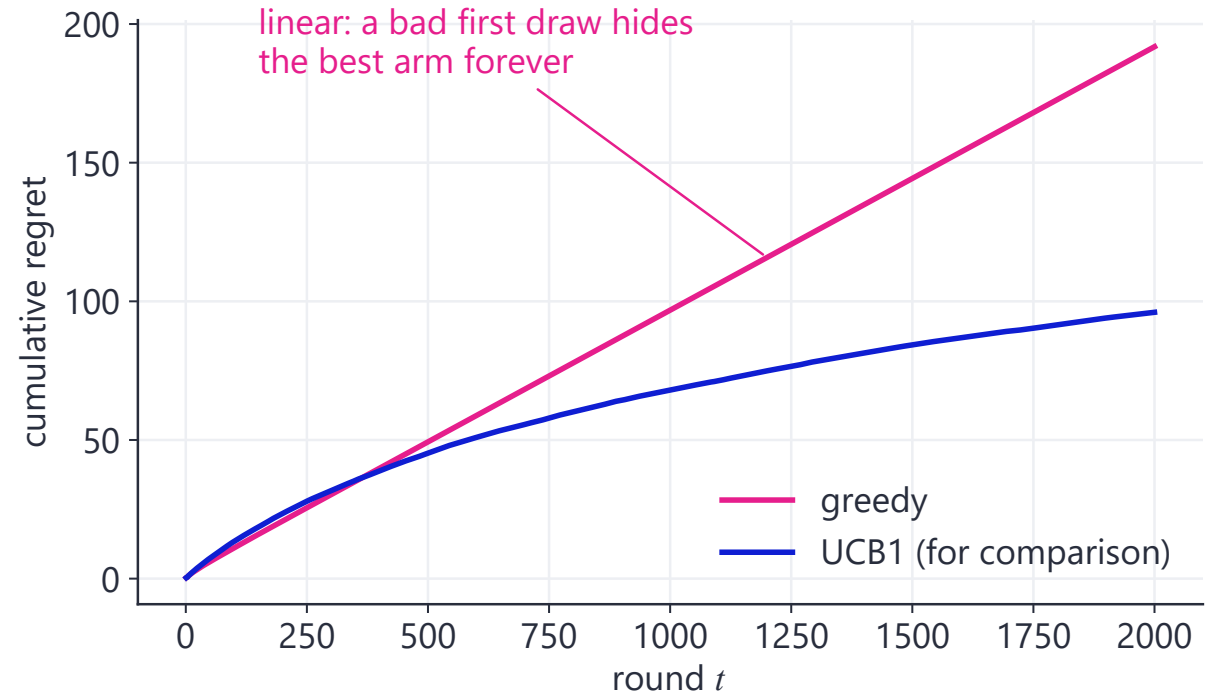
- The oracle strategy is *play the best arm every round*; regret is the expected shortfall against it
- **Sublinear** regret ($R_T/T \rightarrow 0$) means the learner eventually plays the best arm almost always
- **Linear** regret means it keeps paying a fixed price per round: it never learned
- Good algorithms achieve $R_T = O(\log T)$ for stochastic bandits ([Auer et al. 2002](#); [Bubeck and Cesa-Bianchi 2012](#))

The greedy player gets stuck

- Empirical mean of arm a after n_a pulls:

$$\hat{\mu}_a = \frac{1}{n_a} \sum_{t: a_t=a} r_t$$

- **Greedy:** always play $\arg \max_a \hat{\mu}_a$
- One unlucky draw on the best arm and it is never tried again: the estimate cannot recover because no new samples arrive
- Result: a constant per-round loss, i.e. linear regret



ϵ -greedy: explore at random with probability ϵ

At each round t

1. With probability ϵ : play an arm chosen uniformly at random
2. Otherwise: play $\arg \max_a \hat{\mu}_a$
3. Update $\hat{\mu}_{a_t}$ and n_{a_t}

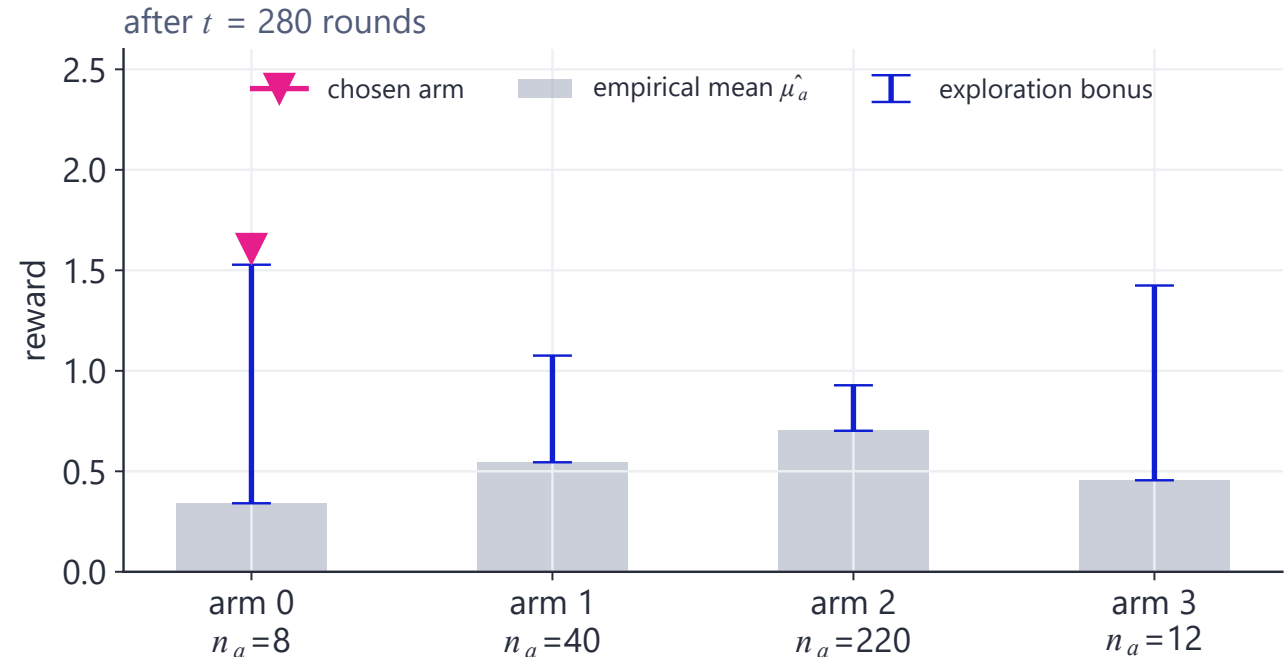
- Simple and surprisingly hard to beat in practice
- Exploration never stops, so regret is linear with slope $\propto \epsilon$
- A decaying $\epsilon_t \propto 1/t$ fixes that but adds a schedule to tune
- Exploration is *blind*: it samples arms known to be bad as often as promising ones

UCB1: optimism in the face of uncertainty

Play the arm with the highest plausible mean, not the highest estimated mean

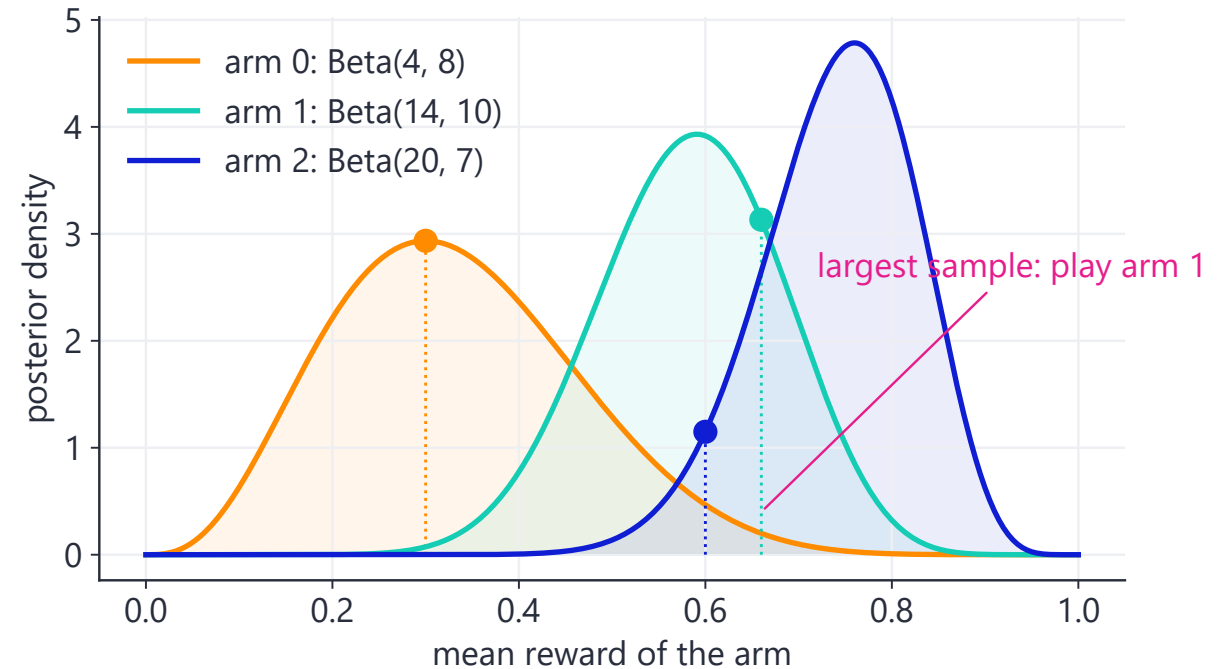
$$a_t = \arg \max_a \left(\hat{\mu}_a + \sqrt{\frac{2 \log t}{n_a}} \right)$$

- The bonus is a confidence width: large for rarely pulled arms, shrinking with n_a
- Uncertain arms get pulled *until* they are shown to be bad
- No parameter to tune; regret $O(\log T)$ (Auer et al. 2002)

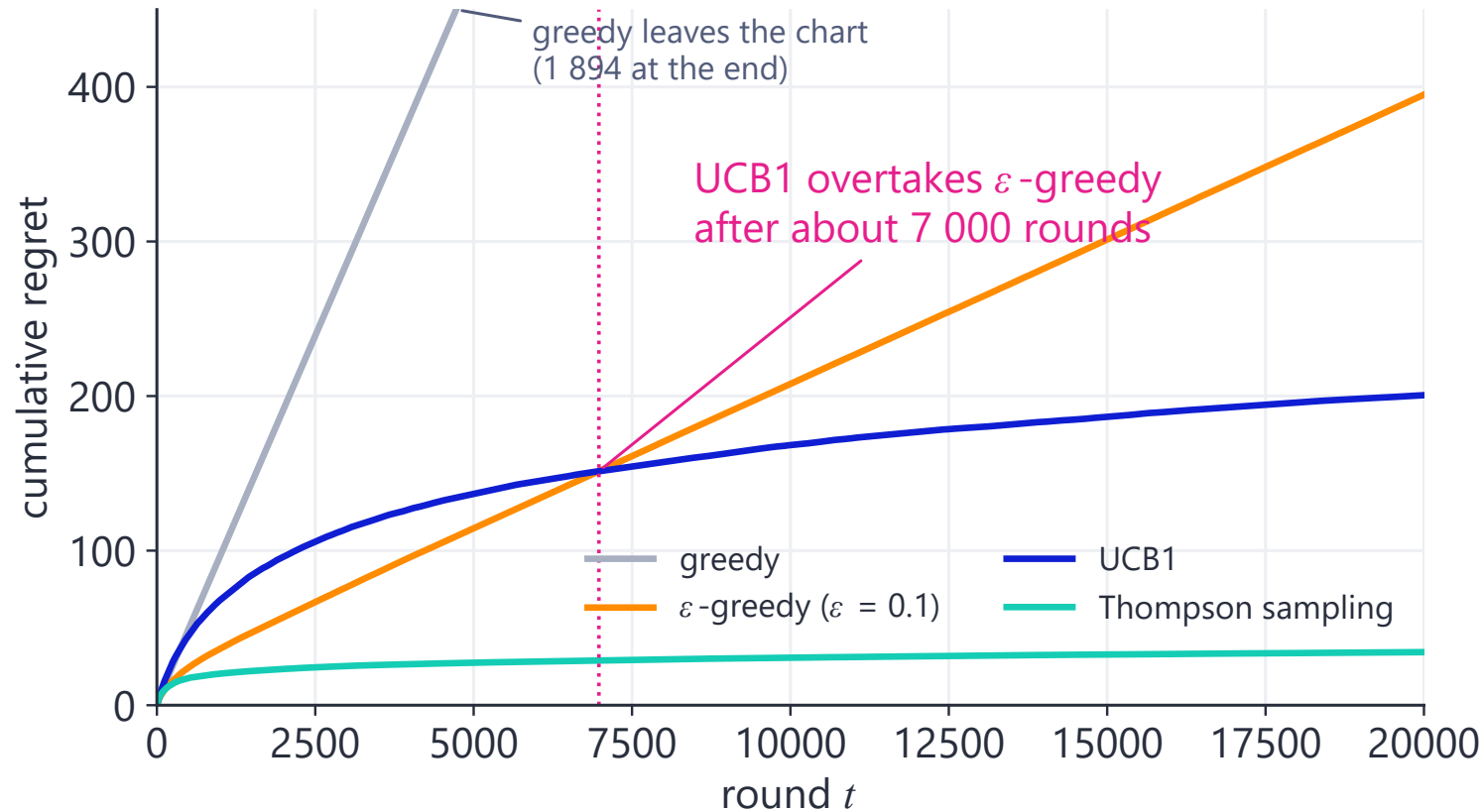


Thompson sampling: play a sample from your belief

- Keep a posterior distribution over each arm's mean (for 0/1 rewards: a Beta distribution updated by counting)
- Each round: **draw one sample per arm** from its posterior, play the arm with the largest sample
- Arms with wide posteriors sometimes produce large samples, so they get explored
- Exploration decreases automatically as posteriors sharpen



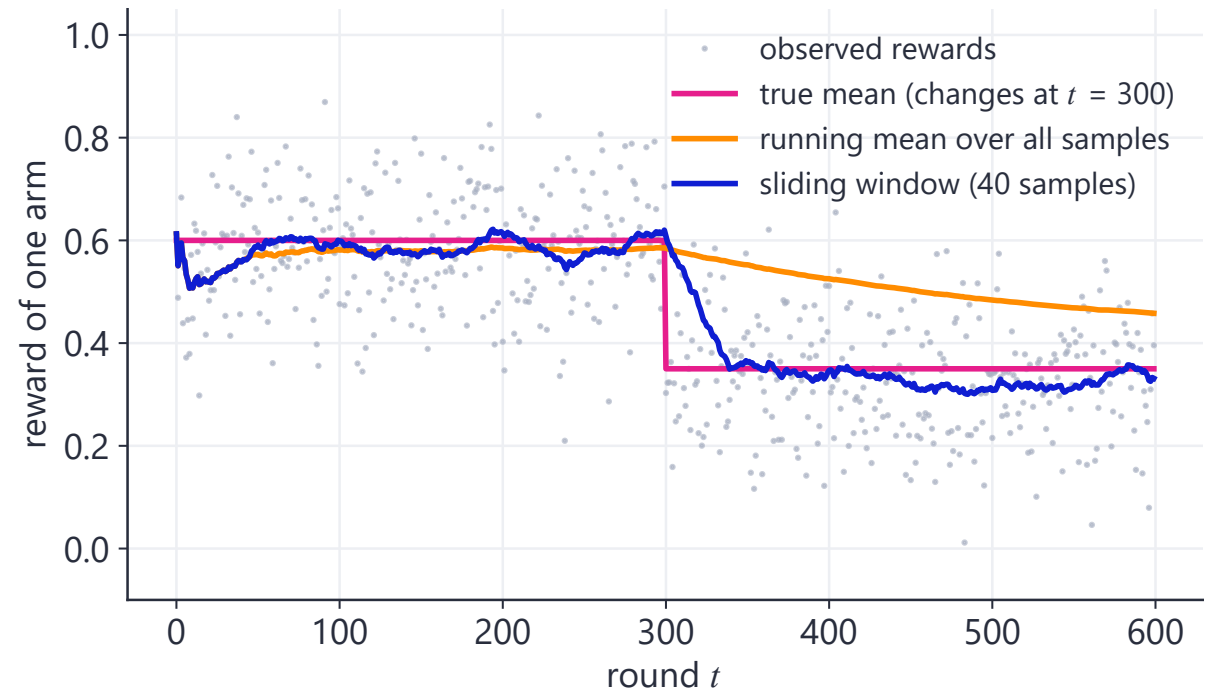
The three algorithms on the same problem



Greedy never recovers. ϵ -greedy looks good early but keeps exploring blindly, so its regret grows linearly and UCB1 overtakes it; UCB1 and Thompson sampling flatten because they stop sampling arms that are clearly worse.

When the world changes: non-stationary bandits

- Network conditions drift or jump: users move, small cells switch off, traffic peaks
- The mean reward of an arm is then a function of time, and old samples mislead
- **Sliding window**: estimate from the last W samples only
- **Change detection**: test whether recent samples still match the reference, and reset when they do not
- Regret per round can no longer go to zero: arms must be re-sampled to track changes

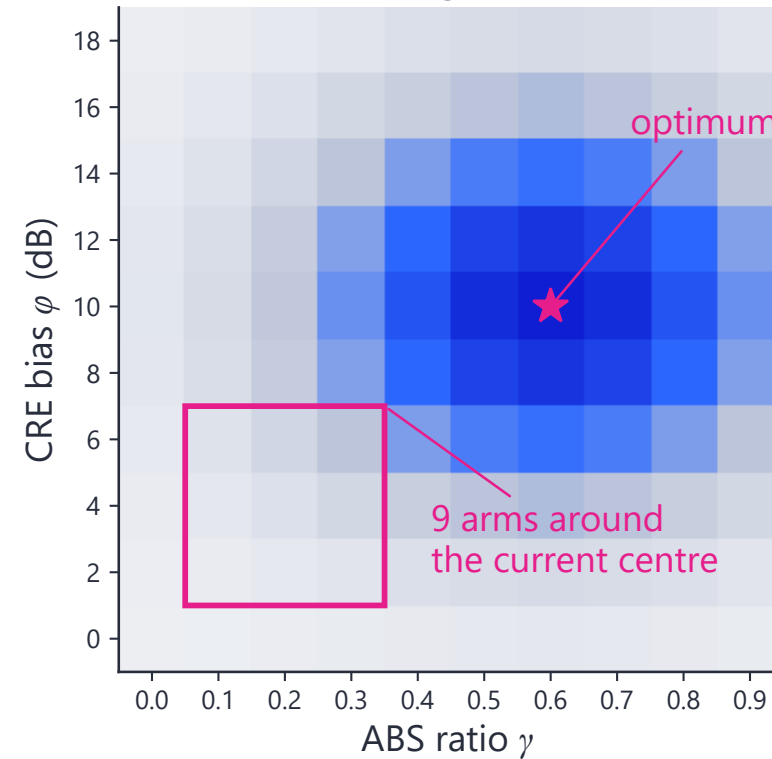


When there are too many arms: use structure

Configurations on a grid have correlated rewards, so explore locally and climb

- A 10×10 grid of configurations is a 100-armed bandit: too many arms to sample each one properly
- But neighbouring configurations behave similarly: the response is smooth and, often, **unimodal**
- Restrict the bandit to the 3×3 neighbourhood of a centre, find the local best, move the centre there
- A sequence of 9-armed problems instead of one 100-armed problem

expected 5th-percentile throughput (unknown to the learner)



Checklist: is my problem a bandit

Yes, if

- one decision at a time from a fixed set
- the reward arrives immediately and depends only on that decision
- nothing carries over between rounds

Not yet, if

- the reward arrives later or accumulates over many rounds
- today's action changes what tomorrow looks like
- the best action depends on observable side information

If the reward is immediate and rounds are independent, start with a bandit

Exercise 1: which arm next

Three arms, round $t = 20$. Which arm does **greedy** play? Which arm does **UCB1** play? Why do they differ?

Arm	Pulls n_a	Empirical mean $\hat{\mu}_a$
0	14	0.62
1	4	0.55
2	2	0.40

- Greedy plays arm 0 (highest mean)
- UCB1 bonus $\sqrt{2 \log 20 / n_a}$: arm 0 gets 0.65, arm 1 gets 1.22, arm 2 gets 1.73, so the indices are 1.27, 1.77 and **2.13**
- UCB1 plays arm 2: two samples say almost nothing, and optimism insists on checking

Agenda

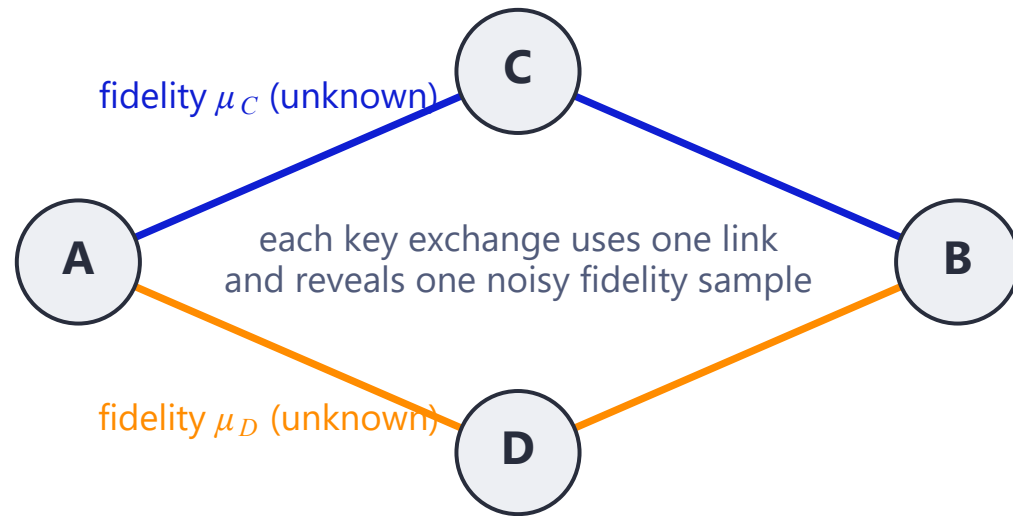
1. Introduction: decisions in networks
2. Multi-armed bandits: the technique
3. **Multi-armed bandits: networking examples**
4. Contextual bandits: the technique
5. Contextual bandits: vrAI
6. Summary

Multi-armed bandits: networking examples

Two problems, two ways of exploiting structure

Example 1: choosing a quantum link

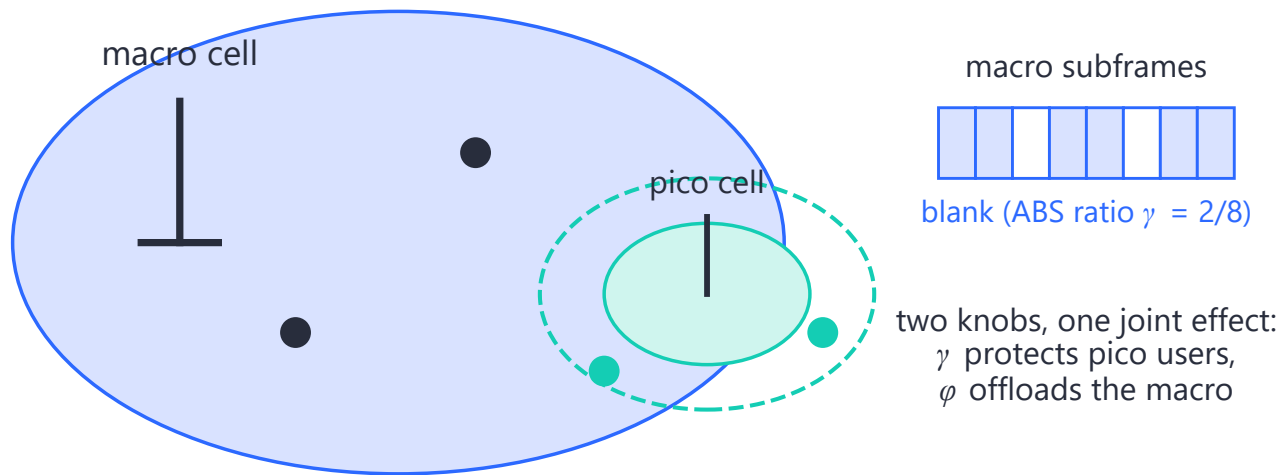
Two candidate paths, an unknown quality per path, and a cost for every measurement



- A wants to exchange a key with B via C or D
- The **fidelity** of a link is its quality KPI; it depends only on the optical fibre, so it is fixed
- Fidelity can only be measured by *using* the link, which spends an entangled qubit
- Arms: the two paths. Reward: the fidelity sample. A 2-armed bandit ([Liu et al. 2024](#))

Example 2: interference coordination in HetNets

Two eICIC knobs with a joint effect on a KPI that has no closed form



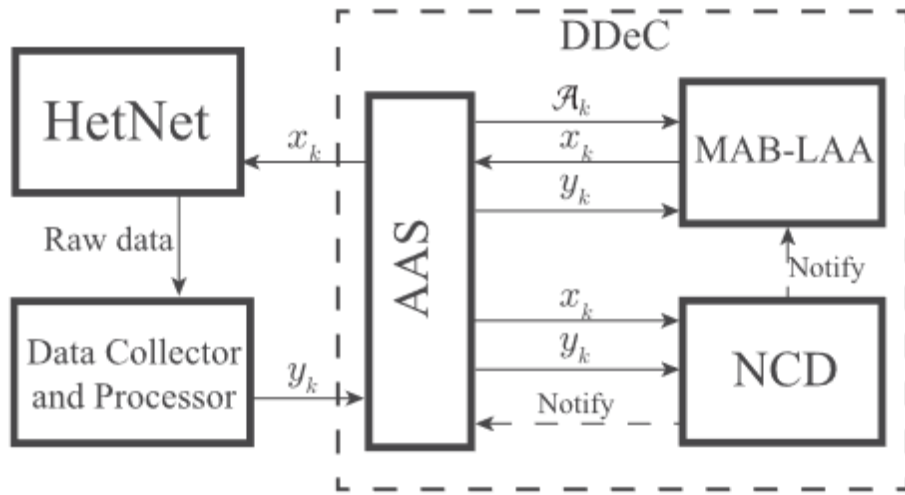
CRE region (dashed): users attached to the pico by the bias φ , but hearing the macro

- Macro and pico cells share the spectrum
- **CRE bias φ** : attach more users to the pico (offloading), at the price of macro interference for them
- **ABS ratio γ** : the macro mutes a fraction of subframes to protect those users
- Target KPI: **5th-percentile user throughput**, a 3GPP fairness metric that is analytically intractable

Formulation: arms are configurations, rewards are measurements

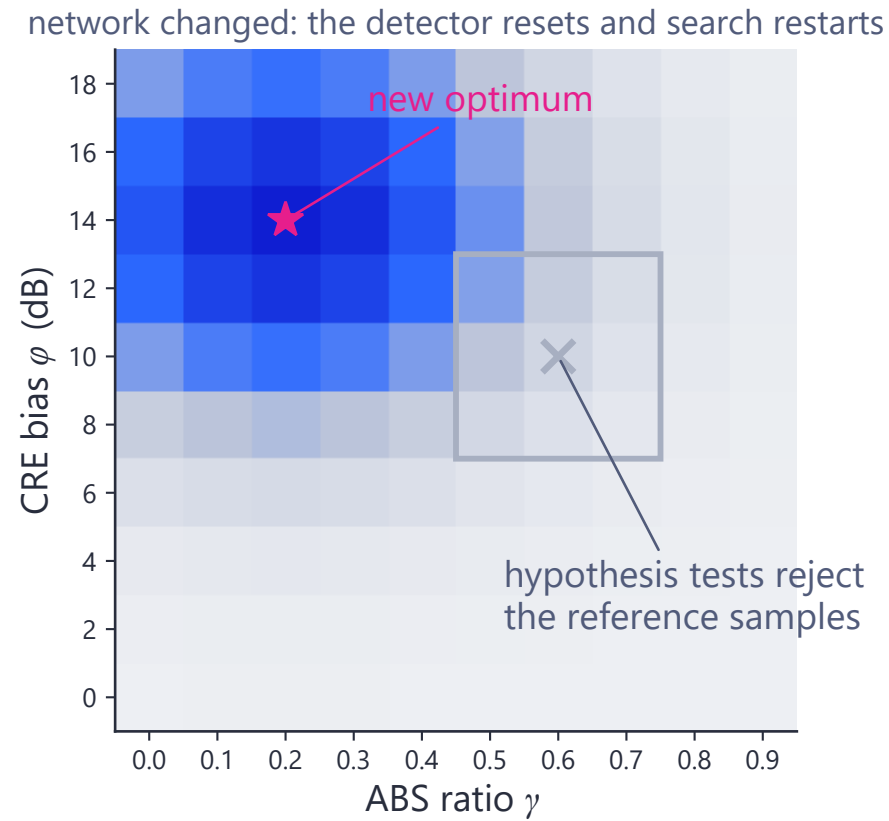
- Arm $a = (\gamma, \varphi)$ on a discrete grid, one instance per cluster of macro cells
- At stage k : apply a_k , collect throughput samples from the network, observe a noisy KPI sample $y_k = F(a_k)$
- The system response $\mathbb{E}[F(a)]$ is unknown and **changes with network conditions**
- Goal: minimise the pseudo-regret $k \mathbb{E}[F(a^*)] - \sum_{i \leq k} \mathbb{E}[F(a_i)]$ while **tracking a moving optimum** a^*
- No knowledge of topology, traffic or scheduling is assumed ([Ayala-Romero et al. 2018](#))

DDeC: local exploration with a bandit inside



- **Action availability selector**: offers only the 3×3 neighbourhood of the current centre
- **MAB with limited action availability**: a bandit policy (Thompson sampling, normal model) over the available arms
- **Network change detector**: hypothesis tests comparing reference samples with the latest ones; after repeated rejections it resets the bandit and restarts the search
- Constant cost per decision, three scalars per arm

DDeC in one picture

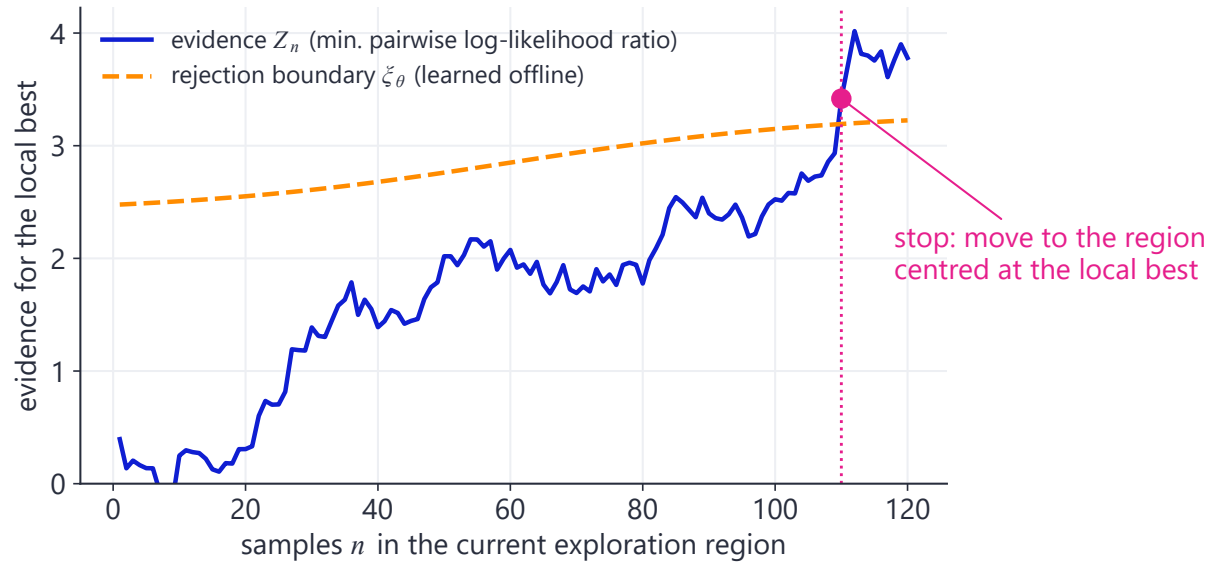


LEOS: how long should each local exploration last

Fixed-length stages waste samples; the right length is itself a decision to learn

- Each exploration stage samples a region $\mathcal{A}_j$ (centre plus neighbours) with a sliding-window UCB rule, then moves to the region centred at the estimated local best $\hat{a}_j$
- Stopping too early moves to a wrong centre; stopping too late wastes samples while the network drifts
- The **continue / move** decision is a Markov decision process whose state is the observation history: far too large to solve directly
- Rewriting the Bellman comparison shows the optimal policy is a **threshold rule**: move when the probability that $\hat{a}_j$ is not the local best drops below a threshold ([Alcaraz et al. 2020](#))

LEOS: the threshold as a likelihood-ratio test



- The error probability is not computable, so compare $\hat{a}_j$ against each other arm with a likelihood ratio; Z_n is the smallest of them
- Stop when Z_n exceeds a **rejection boundary** ξ_θ , a sigmoid of the attained performance: the better we already are, the more evidence a move requires
- θ is fitted offline on past data (model-based RL)
- Theory: bounds on long-term regret per sample and on convergence time

What the two examples teach

Exploit structure

- Unimodality turned a 100-arm problem into a sequence of 9-arm problems
- The bandit policy inside can stay textbook

Plan for change

- A change detector (DDeC) or a learned stopping rule (LEOS) decides *when* to explore again
- Regret per round cannot vanish when the optimum moves

Exploit structure, and plan for change

Exercise 2: formulate your own bandit

A Wi-Fi access point (AP) serves all its clients on **one channel at a time**. Every minute it may switch to any of **8 channels**, and its clients follow it automatically. How much throughput a channel gives depends on how busy it is with **neighbouring networks**, which the AP cannot see in advance; it only measures the total throughput of its clients after each minute.

1. What are the arms, the reward and the horizon?
2. What breaks if the neighbouring APs are *also* switching channels to maximise their own throughput?
3. Interference is heavy in office hours and light at night, and the AP knows the time and its number of clients. How should it use that?

- Arms: the 8 channels. Reward: the clients' throughput in the last minute. Horizon: as long as the AP is on, so regret per minute is what matters
- The best channel moves when the neighbours move: rewards become **non-stationary**, so a sliding window or change detector is needed
- Time and client count are observed before choosing and the AP does not cause them: they are a **context**, so a contextual bandit learns the best channel for each situation

Agenda

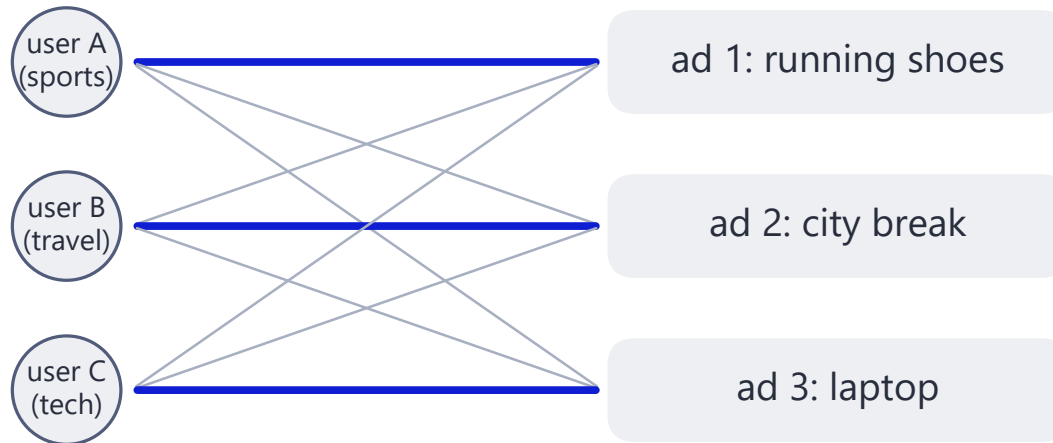
1. Introduction: decisions in networks
2. Multi-armed bandits: the technique
3. Multi-armed bandits: networking examples
4. **Contextual bandits: the technique**
5. Contextual bandits: vrAI
6. Summary

Contextual bandits: the technique

When the best action depends on what you observe before acting

Example: which advertisement to show

Same arms, same reward, but the best arm changes with the user



the best arm depends on who is asking: that is the context

- Arms: ads. Reward: click (1) or no click (0)
- A single bandit learns the ad that is best *on average* over all users
- Yet each user has preferences, and we see who is asking before we choose
- The user description is the **context** ([Li et al. 2010](#))

One bandit per user, or one bandit that sees the user

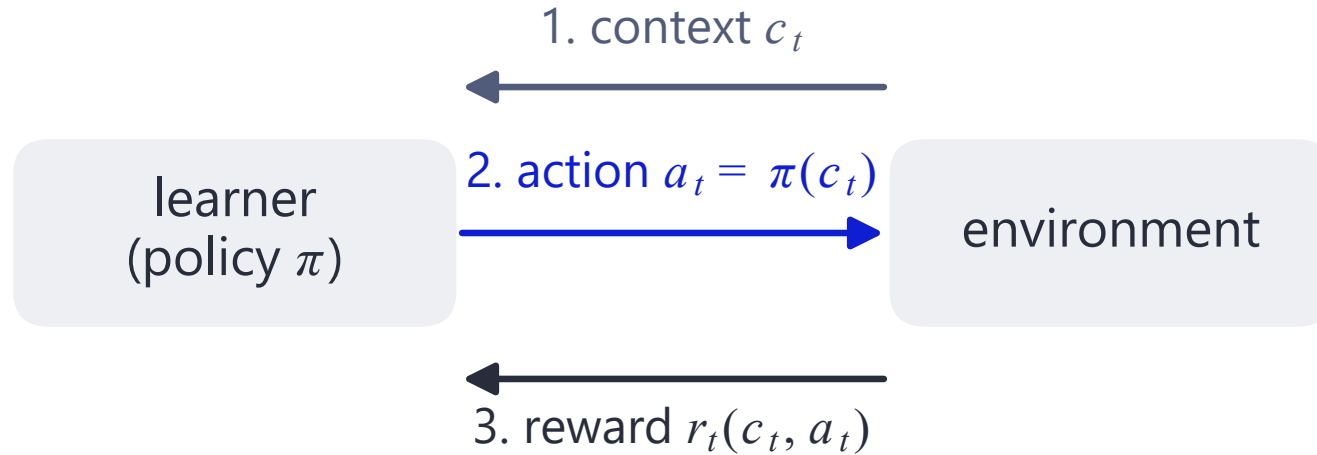
Option 1: one bandit per user

- Learns from zero for every new user
- Needs a lot of data and never transfers what it learned about similar users
- Cold start on every arrival

Option 2: a contextual bandit

- One model whose reward estimate depends on the context
- Similar users share statistical strength, so it converges much faster
- New users are handled by their features, not by their history

The contextual bandit protocol



- Round t : observe c_t , choose $a_t = \pi(c_t)$, observe $r_t(c_t, a_t)$, update π
- Contextual regret compares with the best **policy**, not the best arm: $R_T = \mathbb{E} \left[\sum_t r_t(c_t, \pi^*(c_t)) \right] - \mathbb{E} \left[\sum_t r_t(c_t, a_t) \right]$
- The plain bandit is the special case of a constant context

The two assumptions that make it a bandit and not RL

Rounds must be independent given the context

- **Actions do not affect future contexts.** The context is exogenous: user arrivals, channel quality, traffic demand
- **The reward depends only on the current context and action.** No delayed or accumulated effects
- **Timescale argument:** an orchestrator that decides every few seconds sees a scheduler that reacts in milliseconds as part of the environment, so each decision is effectively independent
- When either assumption fails, the problem is a Markov decision process

LinUCB: a linear model per arm

Assume $\mathbb{E}[r_{t,a} \mid c_{t,a}] = c_{t,a}^\top \theta_a^*$ for an unknown θ_a^* per arm, where $c_{t,a}$ collects user and arm features.

Ridge-regression estimate from the rounds in which arm a was played:

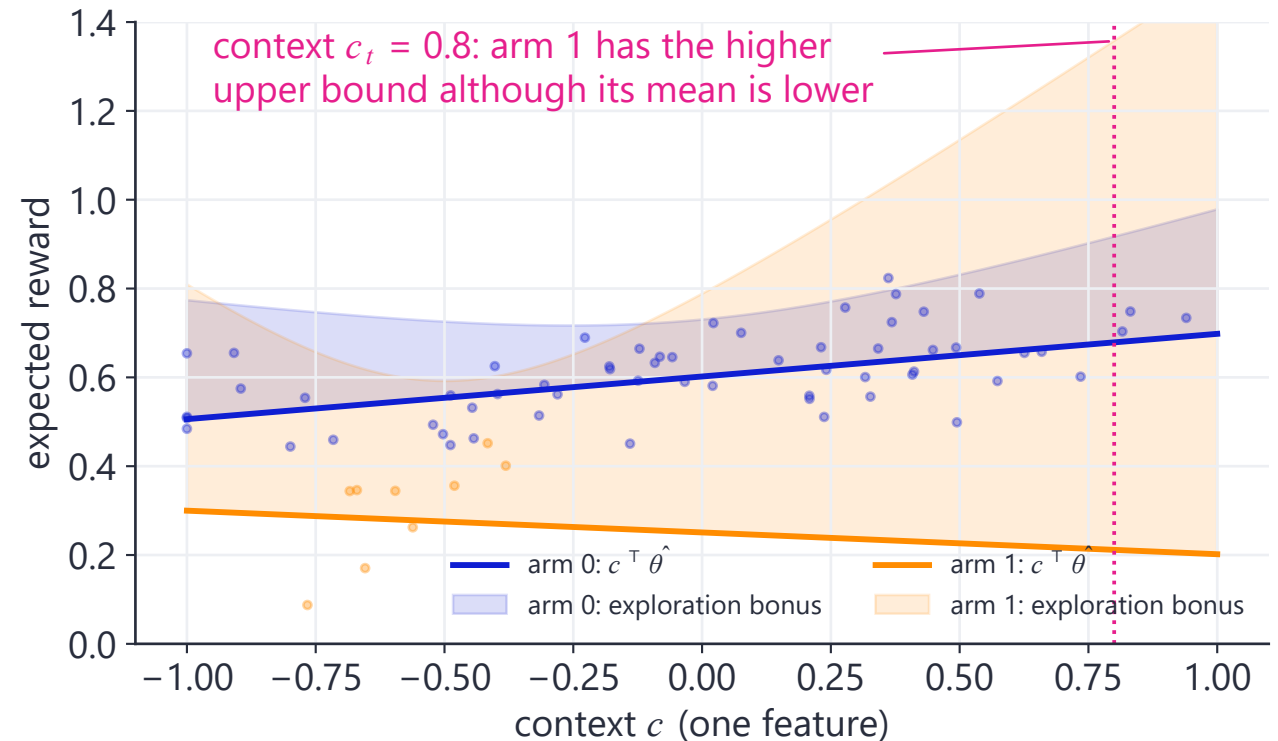
$$\hat{\theta}_a = A_a^{-1} b_a, \quad A_a = D_a^\top D_a + I, \quad b_a = D_a^\top r_a$$

- D_a stacks the contexts observed when a was played, r_a the corresponding rewards
- Both A_a and b_a are updated in constant time per round
- Linear in the arms, at most cubic in the number of features ([Li et al. 2010](#))

LinUCB: the exploration bonus has a closed form

$$a_t = \arg \max_a \left(c_{t,a}^\top \hat{\theta}_a + \alpha \sqrt{c_{t,a}^\top A_a^{-1} c_{t,a}} \right)$$

- The bonus is the **predictive standard deviation** of the linear model: large where few contexts have been seen
- Same principle as UCB1, now context-dependent
- Regret $\tilde{O}(\sqrt{KdT})$ for K arms and d features
- Works with arms that appear and disappear



Evaluating a policy offline from logged data

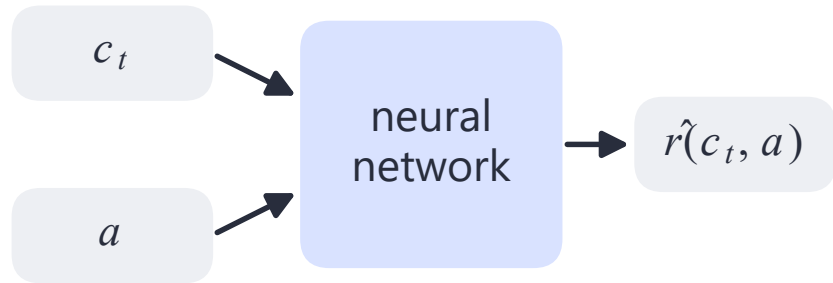
logged events (uniform random logging policy)	policy under test picks	kept?
c_1 , arm 3, click	arm 3	yes, count the reward
c_2 , arm 1, no click	arm 2	no, skip
c_3 , arm 2, click	arm 2	yes, count the reward
c_4 , arm 3, no click	arm 1	no, skip
c_5 , arm 1, click	arm 1	yes, count the reward

the kept events have the same distribution as running the policy live

- We cannot run every candidate policy on production users
- **Replay**: walk through logged events; keep an event only if the policy picks the same arm as the logger
- Unbiased when the logging policy was uniform random; about K logged events per retained event
- For networks: log some randomised exploration once, evaluate many algorithms later ([Li et al. 2010](#))

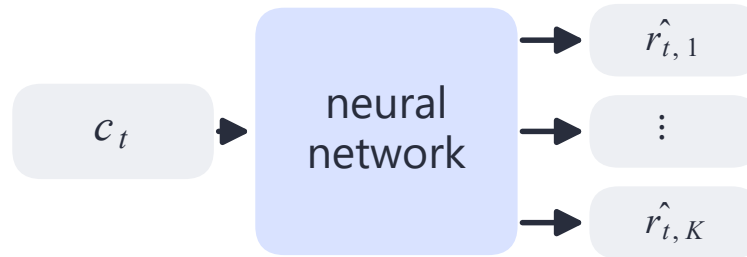
Neural contextual bandits: when the reward is not linear

one network, action as an input



evaluate every candidate action;
works with continuous actions

one network, one output per arm

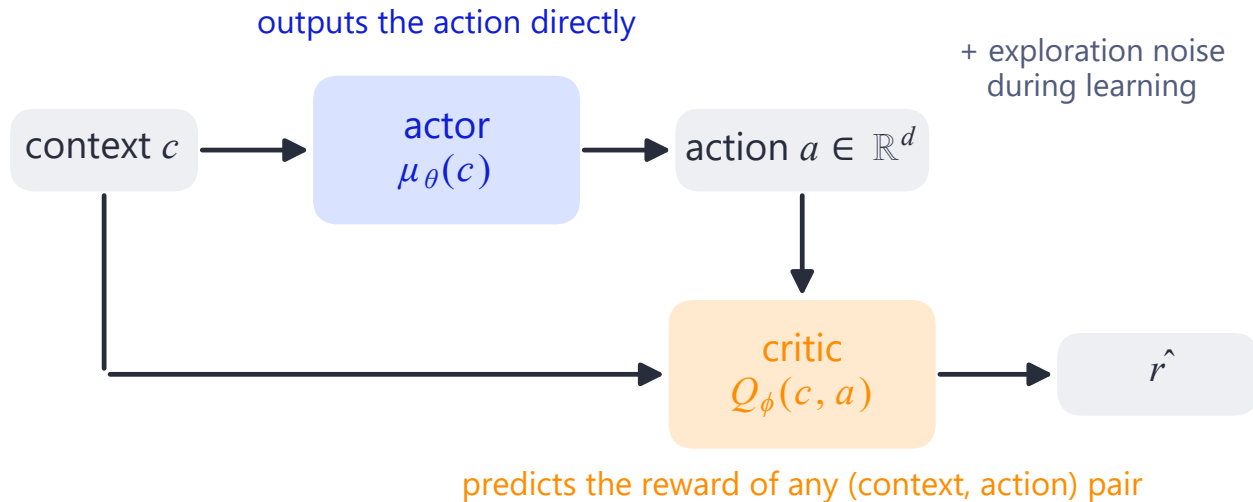


one forward pass gives all arms;
needs a fixed, discrete arm set

- Replace $c^\top \theta_a$ by a neural network when the reward is non-linear or its form is unknown
- Exploration bonus from the network's features (NeuralUCB, (Zhou et al. 2020)) or from a shallow last layer (Xu et al. 2020); theoretical guarantees exist
- Left: any action, including continuous ones, can be scored. Right: one pass gives all arms

Continuous actions: learn a critic and an actor

When the action is a real number, $\arg \max$ over actions is itself a problem: let a network output it



- **Critic** $Q_{\phi}(c, a)$: a neural reward model with the action as an input
- **Actor** $\mu_{\theta}(c)$: outputs the action directly, so no search over actions at decision time
- Exploration: add noise to the actor's output while learning

The deterministic policy gradient in three steps

We want the actor parameters θ that maximise the expected reward under the context distribution:

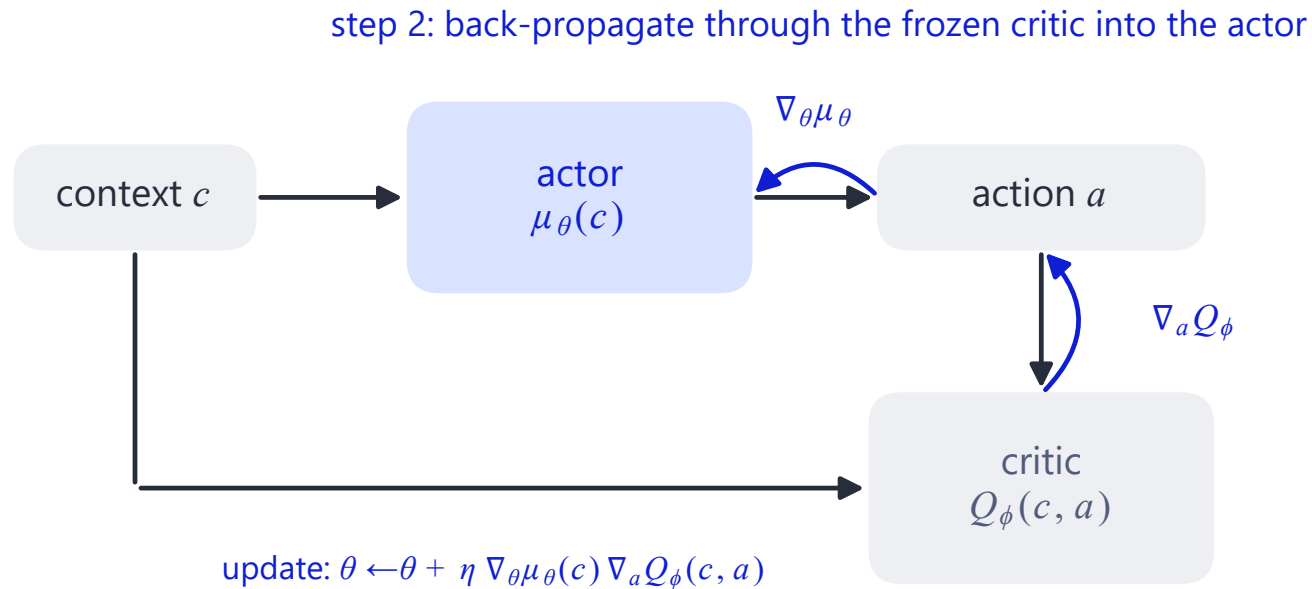
$$J(\theta) = \mathbb{E}_c [Q(c, \mu_\theta(c))]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_c \left[\nabla_\theta \mu_\theta(c) \nabla_a Q(c, a) \Big|_{a=\mu_\theta(c)} \right] \quad (\text{chain rule})$$

$$\theta \leftarrow \theta + \eta \nabla_\theta \mu_\theta(c) \nabla_a Q_\phi(c, a) \Big|_{a=\mu_\theta(c)} \quad \text{with the learned critic } Q_\phi$$

It is the zero-variance limit of the stochastic policy gradient, and needs no integral over actions ([Silver et al. 2014](#))

Training loop: critic on observed rewards, actor through a frozen critic



- Store (c, a, r) in a **replay buffer**; sample mini-batches
- Alternate the two steps; the critic is not updated by the actor's gradient
- This is DDPG ([Lillicrap et al. 2016](#)) restricted to one-step problems

Exercise 3: bandit, contextual bandit, or neither

Classify each scenario. Use the two assumptions: is the context exogenous, and is the reward immediate?

- (a) Choose the modulation and coding scheme for a user given its reported channel quality
- (b) Decide whether a base station sleeps for the next second; sleeping now lets queues grow
- (c) Pick a routing path for a flow given the current load measurements of the links
- (d) Set the TCP congestion window from the last round-trip time

- (a) contextual bandit: channel quality is exogenous, the block error is immediate
- (b) neither: sleeping changes the next state (the queues), so it is an MDP
- (c) contextual bandit if my flow is small; an MDP if my routing changes the loads I will observe next
- (d) neither: the window changes the queueing delay that produces the next round-trip time

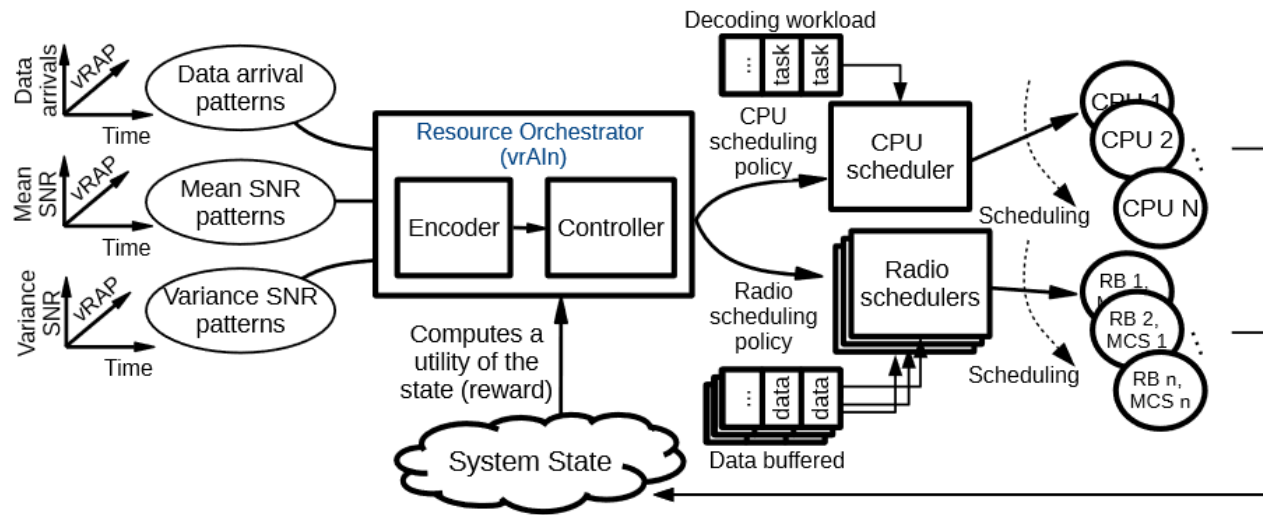
Agenda

1. Introduction: decisions in networks
2. Multi-armed bandits: the technique
3. Multi-armed bandits: networking examples
4. Contextual bandits: the technique
5. **Contextual bandits: vrAln**
6. Summary

Contextual bandits: vrAI

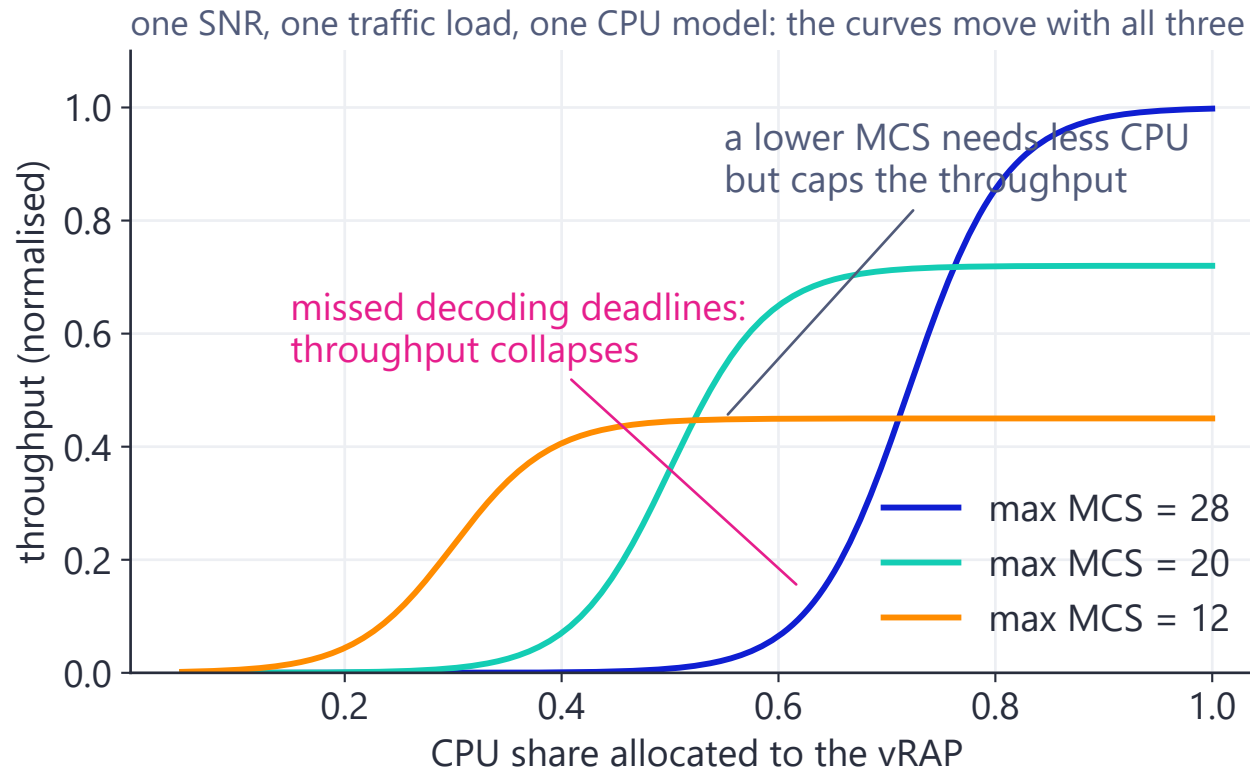
Orchestrating computing and radio resources in virtualised RANs

vRAN: base stations as software on shared CPUs



- Virtual radio access points run on a shared CPU pool, usually at the edge where CPU is scarce
- Today capacity is **over-dimensioned for peak load**: pooling saves little
- CPU and radio decisions are **coupled**: a higher modulation needs more CPU, and missed decoding deadlines look like a bad channel
- Nobody has a model of that coupling per platform ([Ayala-Romero et al. 2022](#))

The response is non-linear and platform dependent



- Throughput as a function of (CPU share, MCS bound) has a cliff where decoding deadlines are missed
- The cliff moves with the SNR, the load and the CPU model
- Tractable models from the literature need calibration per platform and still fail
- Hence a **model-free** learner

Formulation as a contextual bandit

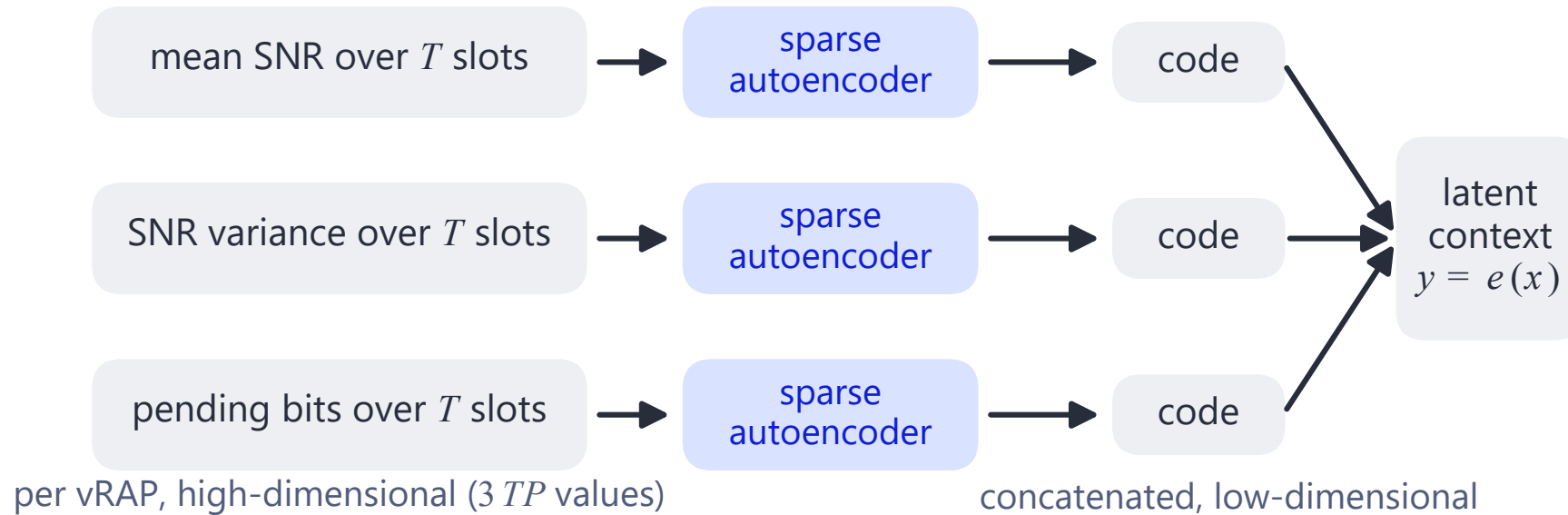
Context c : per vRAP, over T monitoring slots, the mean SNR, the SNR variance and the pending bits (buffer reports)

Action a : per vRAP, a maximum CPU share $c_i \in [0, 1]$ with $\sum_i c_i \leq 1$ (continuous) and a maximum MCS m_i (discrete)

Reward: $r = \sum_i (J_i - M \varepsilon_i - \lambda c_i)$

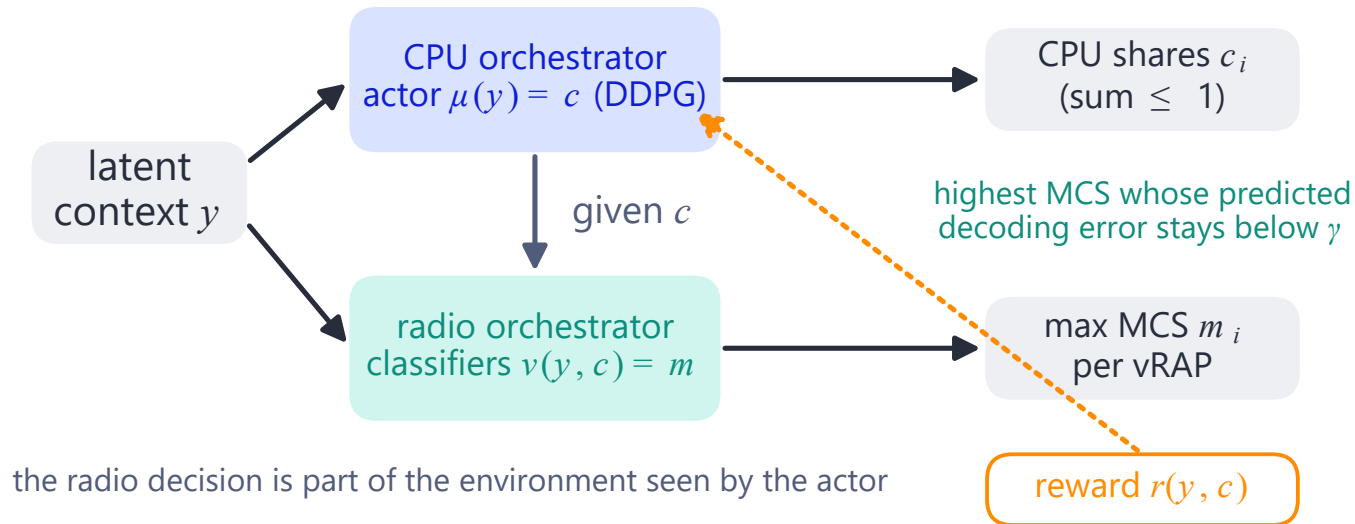
- J_i : probability that the buffer stays below a target (QoS)
- ε_i : decoding error probability, weighted heavily by M
- λ : price of CPU
- Actions are upper bounds any scheduler can enforce
- **Why a bandit:** the orchestrator runs on seconds, the schedulers on milliseconds, so decisions barely affect future contexts

Compressing the context with sparse autoencoders



- Raw context is a set of time series per vRAP: high-dimensional and redundant
- One sparse autoencoder per signal type, applied per vRAP, keeps the temporal structure
- The concatenated codes form the latent context $y = e(x)$ that the controller sees

Decoupling a hybrid action space



- The actor (DDPG, softmax output so shares sum to one) handles the continuous part
- A small classifier per vRAP predicts whether an MCS bound keeps decoding errors below γ for a given CPU share; scan MCSs downwards and take the first feasible one
- The radio decision becomes part of the environment, so optimising c implicitly optimises m

Deploying without exploring in production

Offline pre-training

- A digital twin per vRAP: networks predicting QoS and decoding errors from (context, controls)
- A replay buffer of real experiences (DDPG is off-policy)
- Batch training mixing both, then online fine-tuning with exploration switched off

What it achieved

- Up to 30% less computing capacity than CPU-agnostic methods
- Higher probability of meeting QoS targets than static policies
- Performance close to an offline oracle, on a real srsLTE prototype and in simulations of a production RAN

The penalty weights in the reward are the weak point: tomorrow's first topic

Agenda

1. Introduction: decisions in networks
2. Multi-armed bandits: the technique
3. Multi-armed bandits: networking examples
4. Contextual bandits: the technique
5. Contextual bandits: vrAI
6. **Summary**

Summary

What we covered

Multi-armed bandits

- arms, regret, partial feedback
- ϵ -greedy, UCB1, Thompson sampling
- structure (local exploration) and change (windows, detectors, stopping rules)

Contextual bandits

- exogenous context, immediate reward
- LinUCB and its bonus
- neural rewards, actor-critic and the deterministic policy gradient

Networking examples

- quantum link selection
- eICIC configuration: DDeC, LEOS
- vRAN orchestration: vrAIn

Next session

- **Constraints while learning:** penalties, Gaussian-process safe optimisation, distributional critics
- **When the bandit assumptions fail:** Markov decision processes and reinforcement learning, with AegisRAN
- Bring the two assumptions of today's slide "bandit and not RL": we will use them again

References

- Alcaraz, Juan J., Jose A. Ayala-Romero, Javier Vales-Alonso, and Fernando Losilla-López. 2020. "Online Reinforcement Learning for Adaptive Interference Coordination." *Transactions on Emerging Telecommunications Technologies* 31 (10): e4087.
- Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. "Finite-Time Analysis of the Multiarmed Bandit Problem." *Machine Learning* 47 (2–3): 235–56.
- Ayala-Romero, Jose A., Juan J. Alcaraz, and Javier Vales-Alonso. 2018. "Data-Driven Configuration of Interference Coordination Parameters in HetNets." *IEEE Transactions on Vehicular Technology* 67 (6): 5174–87.
- Ayala-Romero, Jose A., Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Albert Banchs, and Juan J. Alcaraz. 2022. "vrAln: Deep Learning Based Orchestration for Computing and Radio Resources in vRANs." *IEEE Transactions on Mobile Computing* 21 (7): 2652–70.
- Bubeck, Sébastien, and Nicolò Cesa-Bianchi. 2012. "Regret Analysis of Stochastic and Nonstochastic Multi-Armed Bandit Problems." *Foundations and Trends in Machine Learning* 5 (1): 1–122.
- Li, Lihong, Wei Chu, John Langford, and Robert E. Schapire. 2010. "A Contextual-Bandit Approach to Personalized News Article Recommendation." *Proceedings of the 19th International Conference on World Wide Web (WWW)*, 661–70.
- Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, et al. 2016. "Continuous Control with Deep Reinforcement Learning." *International Conference on Learning Representations (ICLR)*.
- Liu, Maoli, Zhuohua Li, Kechao Li, and John C. S. Lui. 2024. "LinkSelfIE: Link Selection and Fidelity Estimation in Quantum Networks." *IEEE INFOCOM*.
- Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. "Deterministic Policy Gradient Algorithms." *International Conference on Machine Learning (ICML)*, 387–95.
- Xu, Pan, Zheng Wen, Handong Zhao, and Quanquan Gu. 2020. "Neural Contextual Bandits with Deep Representation and Shallow Exploration." *arXiv Preprint arXiv:2012.01780*.
- Zhou, Dongruo, Lihong Li, and Quanquan Gu. 2020. "Neural Contextual Bandits with UCB-Based Exploration." *International Conference on Machine Learning (ICML)*.

NEC

\Orchestrating a brighter world