# Autonomous Networking

**Gaia Maselli**
Dept. of Computer Science

# Today's plan

- Another approach to action selection
- Regret
- Contextual bandits

# Learning methods based on action value estimation

- Strategies for action selection
  - Greedy
  - ε-greedy
  - Optimistic initial values
  - Upper Confidence Bound
- Estimate action values and use those estimates to select actions
- Alternative: learn a numerical preference $H_t(a)$ for each action a
- The larger the preference the more often the action is taken
- But the preference has no interpretation in terms of rewards

# Softmax function

- The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1
- The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities
- If one of the inputs is small or negative, the softmax turns it into a small probability
- If one input is large, then it turns it into a large probability, but it will always remain between 0 and 1.

$$S(\mathbf{a}) : \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_N \end{bmatrix}$$

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \qquad \forall j \in 1..N$$

# Softmax: example

$[a_1, a_2, a_3] = [8, 5, 0]$

$e^{a1} = e^8 = 2981.0$

$e^{a2} = e^5 = 148.4$

$e^{a3} = e^0 = 1.0$

$S_1 = 2981/(2981+148.4+1) = 0.95$

$S_2 = 148.4\,/(2981+148.4+1) = 0.0474$

$S_3 = 1\,/(2981+148.4+1) = 0.0003$

$[8, 5, 0] \rightarrow [0.95, 0.0474, 0.0003]$

$$S(\mathbf{a}) : \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_N \end{bmatrix}$$

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \qquad \forall j \in 1..N$$

# Action preference

- The idea is to consider the relative preference of one action over another
- Action probabilities are determined according to a ***soft-max distribution***:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Probability of taking action a at time t

- Initially all action preferences are the same
- $H_1(a)=0$ for all a
- All actions have an equal probability of being selected

# Gradient Bandit algorithm

- On each step, after selecting action $A_t$, and receiving the reward $R_t$, the action preferences are updated by:

$$
\begin{aligned}
H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), && \text{and} \\
H_{t+1}(a) &\doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), && \text{for all } a \neq A_t,
\end{aligned}
$$

- Where $\boldsymbol{\alpha} > 0$ is a step-size parameter, and $\overline{R} \in \mathbb{R}$ is the average of all the rewards up through and including time t, which can be computed incrementally
- The $\overline{R}$ term serves as a baseline with which the reward is compared
- If the reward > baseline, then the probability of taking $A_t$ in the future is increased
- If the reward < baseline, then the probability is decreased

- What it means to do well
- How much worse we do than the optimal value
- If we want to evaluate algorithm in this context a very useful tool is the notion of *regret*

# Regret

- The action-value is the mean reward for action a,

$$Q(a) = \mathbb{E}\left[r|a\right]$$

- The optimal value V∗ is

$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

- The regret is the opportunity loss for one step

$$l_t = \mathbb{E}\left[V^* - Q(a_t)\right]$$

- The total regret is the total opportunity loss

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$

- Maximize cumulative reward = minimize total regret

# Counting regret

- The count $N_t(a)$ is expected number of selections for action a
- The gap Δa is the difference in value between action a and optimal action a* , Δa = V* − Q(a)
- Regret is a function of gaps and the counts

$$
\begin{aligned}
L_t &= \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right] (V^* - Q(a)) \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right] \Delta_a
\end{aligned}
$$

- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known!

# Contextual bandit

- So far we have considered **nonassociative tasks**, that is, tasks in which there is no **need to associate different actions with different situations**
- Goal:
  - find a single best action when the task is with stationary
  - Tries to track the best action as it changes over time when the task is nonstationary
- Often ther is more than one situation
- Goal:
  - Associative search
  - learn a mapping from situations to actions that are the best in those situations

# Multi-armed bandit

- No context
- Try to do as well as best single action
  - Tacitly assuming there is one action that gives high reward
  - E.g., single treatment that is right for entire population

- Medical treatment example
- A single treatment that is perfect for all patients regardless of their symptoms, test results, gender, age, etc.

# Contextual bandits

- In contextual bandits setting, can use context to choose actions
- May exist good policy (decision rule) for choosing actions based on context
- E.g.:

| | |
|---|---|
| if (sex = male) | choose action 2 |
| Else is (age > 45) | choose action 1 |
| else | choose action 3 |

- Policy π : (content x) → (action a)