Logic

Given two outcomes exactly one of these holds:

$$x \sim y,\ x \prec y,\ x \succ y$$

or

$$x \succcurlyeq y,\ x \preccurlyeq y$$

Summary:
$\prec$ is **anti-symmetric, transitive** and **negative transitive**
$\preccurlyeq$ is **complete** and **transitive**
$\sim$ is **reflexive**, **symmetric** and **transitive**
Rational preference if complete and trans. ($\preccurlyeq$)

Utility function

If $\preccurlyeq$ holds one possible outcomes there exists a function $u : \mathcal{X} \to \mathbb{R}$ such that $u(x) \geq u(y)$ iff $x \succcurlyeq y$ for all $x, y \in \mathbb{R}$
$X(a)$ is the outcome of action $a \in A$

$$\begin{gathered}Q(a) = \sum_{x \in \mathcal{X}} u(x)\mathbb{I}[x = X(a)] \\ P(x|a) = \mathbb{P}[\mathrm{x} = x|\mathrm{a} = a] \\ \downarrow \\ Q(a) = \sum_{x \in \mathcal{X}} u(x)P(x|a) \\ = \mathbb{E}[u(x)|a]\end{gathered}$$

POMDP

Description

$(X, A, P, c, Z, O, \gamma)$

- X: is the set of states (could be positions, states of actors etc.)

- A: set of actions (all the things the agent can do)
- P: transition probabilities between states given an action ($P_a : S \times S \to \mathbb{R}, a \in A, P = \{P_a | a \in A\}$)
- c: cost function mapping action and state to a cost ($c : S \times A \to \mathbb{R}$)
- Z: the set of all observations, should be the set where all actions will have an observation that should be able to map to a state, this depends on **the current state and the previous action**
- O: the given observation maps to a state with a probability ($O_a : S \times Z \to \mathbb{R}, a \in A$)($[\mathbf{O}_a]_{xz} = \mathbb{P}[\mathbf{z}_t = z | x_t = x, a_{t-1} = a]$)
- γ: the discounting factor

Policy iteration

Policies are mappings from **histories to actions** and represent as tree.
Each node is action form that history and each branch is an observation
Each history leading to the same belief will have equivalent subtrees
Each node corresponds to a α-vector

Key points

- POMDP ↔ Belief MDP
  - Can use VI → Cost-to-go is PWLC
  - Can use PI → Policy graphs w/finite number of nodes
- Approximate methods:
  - MDP heuristics
  - Point-based methods

Belief MDP

Heuristics

MLS (MOST LIKELY STATE)

Select the most likely state
For the tiger problem you would never listen because you would always map actions to either open left or open right
$\pi_{MLS}(\mathbf{b}) = \pi_{MDP}(\underset{x\in\mathcal{X}}{argmax}\ \mathbf{b}(x))$

AV(action vote or something)

Each prob votes for the action proportionally to their probability
For tiger problem no one will vote for listen
$\pi_{AV}(\mathbf{b}) = \underset{a\in\mathcal{A}}{argmax} \sum_{x\in\mathcal{X}} \mathbf{b}(x)\mathbb{I}(a = \pi_{MDP}(x))$

Q-MDP

Partial observ. is over at time t+1, agent is then overly optemistic
Cost to go is then: $J^*(\mathbf{b}) = \underset{a\in\mathcal{A}}{min} \sum_{x\in\mathcal{X}} \mathbf{b}(x) Q^*_{MDP}(x,a)$
Heurisitc is then: $\pi_{QMDP}(\mathbf{b}) = \underset{a\in\mathcal{A}}{arg\ min} \sum_{x\in\mathcal{X}} \mathbf{b}(x) Q^*_{MDP}(x,a)$

FIB

Takes into consideration partial observability
$\pi_{FIB}(\mathbf{b}) = \underset{a\in\mathcal{A}}{arg\ min}\ \sum_{x\in\mathcal{X}} \mathbf{b}(x) Q_{FIB}(x,a)$

Value iteration

Building Γ_{k+1} from Γ_k

- **Region-based methods:**
  - Start with empty Γ_{k+1} and only add vectors that are necessary (Witness algo)
- **Prunning-based methods**
  - Start with full Γ_{k+1} and remove unnecessary (Incremental pruning algo)
- **Point based methods**
  - Only compute α that truly matter

    - Will never visit points which are not close to other believes
- Approximate but much faster
    - Sample the believes, compute α-s from there and then maybe recompute the sampled beliefs

Markov chain

Markov property:
$\mathbb{P}(X_{t+1} = s | X_t = s_t, X_{t-1} = s_{t-1}, \dots, X_0 = s_0) = \mathbb{P}(X_{t+1} = s | X_t = x_t)$

Update markov chain:
$\mu_{t+1} = \mu_t P$ e.g.

$$P = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.0 & 0.7 & 0.3 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, \mu_0 = \begin{bmatrix} 0.0 & 0.5 & 0.5 \end{bmatrix}$$

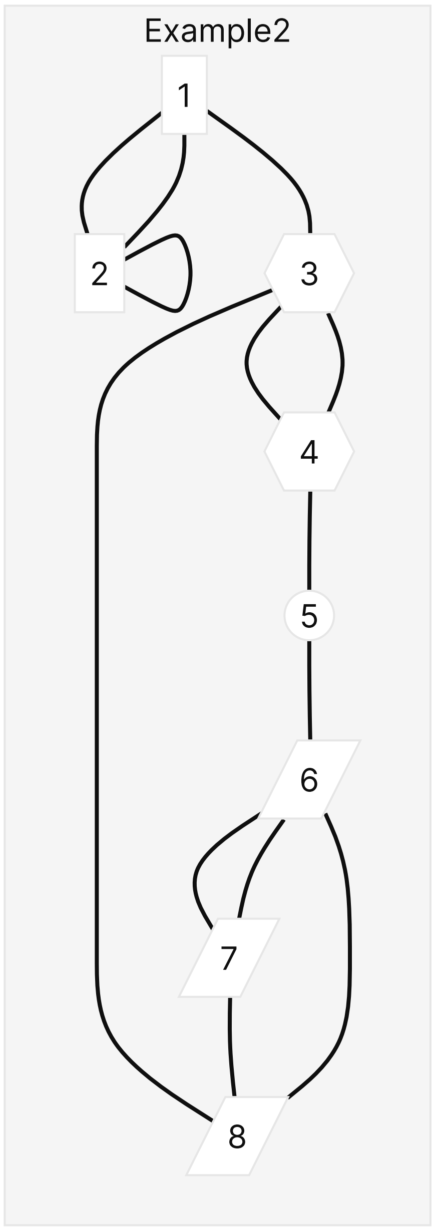
$$\begin{aligned} \mu_1 =& \mu_0 P \\ =& [(0 \times 0 + 0 \times 0 + 0 \times 0)(0 \times 0.5 + 0.5 \times 0.7 + 0.5 \times 0)(0 \times 0.5 + 0.5 \times 0.3 + 0. \\ =& \begin{bmatrix} 0.0 & 0.35 & 0.65 \end{bmatrix} \end{aligned}$$

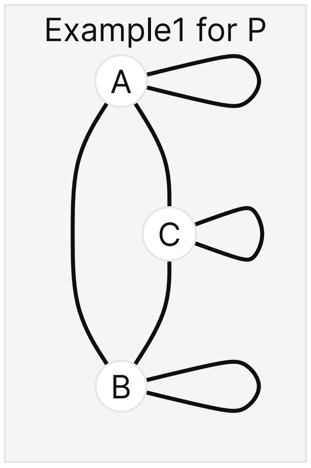
Communicating classes:

Subset of all states where all state in a class "communicate"

$x, y \in \mathcal{X}$ communicate if $x \to y$ and $y \to x$ i.e. $x \leftrightarrow y$

$$\mathbf{P} = \begin{bmatrix} 0.3 & 0.1 & 0.6 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.7 & 0.3 \end{bmatrix}$$





Giving communicating classes $\{A\},\{B\},\{C\}$ meaning that it is reducible since redusible classes > 1

Ergodic

The chain is ergodic if it is irreducible and aperiodic, and $lim_{t \to \infty} \mu_0 \mathbf{P}^t = \mu^*$ (μ^* is the stationary dist)

HMM (Hidden markov model)

Has: (P, O, X, Z)
P: The transition probabilities

X: The hidden states
O: The probabilities for state and observation
Z: The observable states

Filtering (Forward algorithm)

Sequence of observations → final state
Forward mapping defined as: $\alpha_t(x) = \mathbb{P}_{\mu_0}[x_t = x, \mathrm{z}_{0:t} = z_{0:t}]$
We want $\mu_{T|0:T}$ from α_T

$$\mu_{T|0:T}(x) = \frac{\alpha_T(x)}{\Sigma_{y\in\chi}\alpha_T(y)}$$

$$\alpha_T(x) = \mathbf{O}(z_T|x)\sum_{y\in\mathcal{X}}\mathbf{P}(x|y)\alpha_{T-1}(y)$$

Require observation sequence $z_{0:T}$

```
1. Initialize $\alpha_0 = diag(\mathbf{O}(z_0|\cdot))\mu_0^\top$
2. for t=1, ...., T do
       $\alpha_t = diag(\mathbf{O}(z_t|\cdot))\mathbf{P}^\top\alpha_{t-1}$
3. end
4. return $\mu_{T|0:T} = \frac{\alpha_\mathbf{T}}{sum(\alpha_\mathbf{T})}$
```

Example: urn

$$\mu_\mathbf{0} = [0.125 \quad 0.375 \quad 0.375 \quad 0.125]$$
$$z_{0:2} = \{w, w, b\}$$
$$\mathbf{O} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

what is state at $t = 2$?

–

$$\alpha_0 = diag(\mathbf{O}(z_0|\cdot))\mu_0^\top$$
$$\downarrow$$
$$\mathbf{O} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\alpha_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.125 \\ 0.375 \\ 0.375 \\ 0.125 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0 \\ 0.375 \\ 0.0 \end{bmatrix}$$

$$\alpha_1 = diag(\mathbf{P}(z_1|\cdot))\mathbf{P}^\top \alpha_0$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.2 & 0.2 & 0.05 & 0.05 \\ 0.6 & 0.6 & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.6 & 0.6 \\ 0.05 & 0.05 & 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0.125 \\ 0 \\ 0.375 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.04 \\ 0 \\ 0.24 \\ 0 \end{bmatrix}$$

$$\alpha_2 = diag(\mathbf{P}(z_2|\cdot))\mathbf{P}^\top \alpha_1$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.2 & 0.2 & 0.05 & 0.05 \\ 0.6 & 0.6 & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.6 & 0.6 \\ 0.05 & 0.05 & 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0.04 \\ 0 \\ 0.24 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.06 \\ 0 \\ 0.05 \end{bmatrix}$$

$$\mu_{\mathbf{2|0:2}} = \alpha_2/sum(\alpha_2) = [0 \quad 0.552 \quad 0 \quad 0.448]$$

Smoothing

Forward backward

Defined as: $\beta_t(x) = \mathbb{P}_{\mu_0}[\mathbf{z}_{t+1:T} = z_{t+1:T} \mid \mathbf{x}_t = x]$
Algo:
Observation sequence $z_{0:T}$

1. Init $\alpha_0 = diag(\mathbf{0}(z_0|\cdot))\mu_0^\top$

```
2. For $\tau = 1, \cdots, t$ do
       $$\alpha_t = diag(\mathbf{O}(z_\tau|\cdot))\mathbf{P}^\top \alpha_{\tau-1}$$
3. end for
4. For $\tau = T-1, \cdots, t$ do
       $$\beta_\tau = \mathbf{P} diag(\mathbf{O}(z_{\tau+1}|\cdot))\beta_{\tau+1}$$
5. end for
   return $\alpha_t \odot \beta_t / sum(\alpha_t \odot \beta_t)$
```

Viterbi

Sequence of observations → sequence of hidden states
Require: Observation sequence $z_{0:T}$

```
1. Initialize $\mathbf{m}_0 = diag(\mathbf{O}(z_0|\cdot))\mu_0^\top$ or if no $z_0$ → $m_0 = \mu_0$
2. for $t = 1, \ldots, T$
       $$\mathbf{m}_t = max(col)\{diag(\mathbf{m}_{t-1})\mathbf{P}\}diag(\mathbf{O}(z_t|\cdot))$$
       $$\mathbf{i}_t = argmax(col)\{diag(\mathbf{m}_{t-1})\mathbf{P}\}$$
3. end for
4. Let $x_T^* = \arg\max_{x \in \mathcal{X}} m_T(x)$
5. for $t = T-1, \ldots, 0$ do
       $$x_t^* = i_{t+1}(x_{t+1}^*)$$
6. end for
   return $\mathbf{x}_{0:T}^*$
```

Prediction

Sequence of observations → future states
Compute with the forward algo.
Might have to change μ with b

MDP

Policy types

Deterministic:

Actions are selected with prob. 1

Memoryless:

The actions chosen only depends on the last observation (and t)

Stationary:

Actions selected only depend on the last observation (**and not t**)

Types of policies

- Non-deterministic (choose actions randomly) vs deterministic(not random action) (simpler)
- Non-markov vs markov (memory vs memory-less)
- Non-stationary vs stationary(fixed through time)
  We want Deterministic and Stationary

Stationary policy

$$\mathcal{X} = \{0, 1, 2\}$$
$$\mathcal{A} = \{a, b\}$$
$$\mathbf{P} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$x \in \mathcal{X}, a \in \mathcal{A}$$
$$\mathbf{P}_\pi(y|x) = \mathbb{E}_{a\sim\pi(x)}[\mathbf{P}(y|x,a)] = \sum_{a\in\mathcal{A}} \pi(a|x)\mathbf{P}(y|x,a)$$

$$c_\pi(x) = \mathbb{E}_{a\sim\pi(x)}[c(x,a)] = \sum_{a\in\mathcal{A}} \pi(a|x)c(x,a)$$

Cost to go

$$J^{\pi}(x) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c_t | \mathrm{x}_0 = x\right]$$
$$or$$
$$\mathbf{J}^{\pi} = (\mathbf{I} - \gamma \mathbf{P}_{\pi})^{-1} \mathbf{c}_{\pi}$$

Value iteration

$$\mathbf{J}_t = c_{\pi} + \gamma \mathbf{P}_{\pi} \mathbf{J}_{t-1}$$
$$\mathbf{J}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

sl

Optimal

$$J^{\pi^*}(x) \leq J^{\pi}(x)$$

Q-function

$$Q^{\pi}(x,a) = c(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}(x'|x,a) J^{\pi}(x')$$

$$Q_a^{\pi} = c_a + \gamma \mathbf{P}_a \mathbf{J}^{\pi}$$

$$J^*(x) = \underset{a \in \mathcal{A}}{min} Q^*(x,a)$$

$$J^{\pi}(x) = \sum_{a \in \mathcal{A}} \pi(x,a) Q^{\pi}(x,a)$$

$$\underset{a \in \mathcal{A}}{min}\left[c(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}(x'|x,a) J^{\pi}(x')\right] = J^*(x)$$

$$Q^{\pi}(x,a) = c(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}_a(x'|x) \mathbb{E}_{a' \sim \pi(x')}[Q^{\pi}(x',a')] = \mathbf{H}_{\pi}$$
$$\mathbf{H}_{\pi}\mathbf{Q} = \mathbf{C} + \gamma \mathbf{P} \mathbb{E}_{\pi}[\mathbf{Q}]$$

$$Q^*(x,a) = c(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}_a(x'|x) \underset{a' \in \mathcal{A}}{min}[Q^{\pi}(x',a')] = \mathbf{H}$$
$$\mathbf{HQ} = \mathbf{C} + \gamma \mathbf{P} min Q$$

Policy iteration:
compute J, compute Q find greedy policy from Q, compute J from new Q and so on

Greedy policy

Find the best immediate actions and break ties uniformly

$$\pi_g^{J^\pi}(x) = \arg\min_{a \in \mathcal{A}} Q^\pi(x, a)$$

Is optimal in reference to J^*

$$J^{\pi_g}(x) \leq J^\pi(x)$$

Meaning the the policy π_g is better than π

Learning

Nomeclature:
Expected cost:$L(\pi) = \mathbf{E}_{\mu_D}[l(x, a, ; \pi)]$
Goal: $\pi^* = arg \underset{\pi}{min} L(\pi)$
Inductive learning: If we learn from a sufficiently large set of examples we will do well in the actual task
Need metrics between states in mdp to say something about similarity, has the **inductive bias assumption**: all costs are the same for all actions
"Long term similarity": Bissimulation metric, Earth Movers Distance → Computationally expensive

Model based learning

Computing J^*

Given sample (x_t, c_t, x_{t+1}) selected from π
Perform the updates

- start with:

$$\hat{c} = \begin{bmatrix}0\\0\\0\end{bmatrix} \hat{\mathbf{P}} = \begin{bmatrix}0.33 & 0.33 & 0.33\\0.33 & 0.33 & 0.33\\0.33 & 0.33 & 0.33\end{bmatrix}$$

do:

- $\hat{c}(x_t) = \hat{c}(x_t) + \alpha_t(c_t - \hat{c}(x_t))$
- $\hat{\mathbf{P}}(x'|x_t) = \hat{\mathbf{P}}(x'|x_t)\alpha(\mathbb{I}[x_{t+1} = x'] - \hat{\mathbf{P}}(x'|x_t))$
- $J_{t+1}(x_t) = \hat{c}(x_t) + \gamma \sum_{x' \in \mathcal{X}} \hat{\mathbf{P}}(x'|x_t) J_t(x')$

Computing Q^*

Given sample (x_t, c_t, x_{t+1}) selected from π
Perform the updates

- start with:

$$\hat{c} = \begin{bmatrix}0\\0\\0\end{bmatrix} \hat{\mathbf{P}} = \begin{bmatrix}0.33 & 0.33 & 0.33\\0.33 & 0.33 & 0.33\\0.33 & 0.33 & 0.33\end{bmatrix}$$

do:

- $\hat{c}(x_t, a_t) = \hat{c}(x_t, a_)) + \alpha_t(c_t - \hat{c}(x_t, a_t))$
- $\hat{\mathbf{P}}(x'|x_t, a_t) = \hat{\mathbf{P}}(x'|x_t, a_t)\alpha(\mathbb{I}[x_{t+1} = x'] - \hat{\mathbf{P}}(x'|x_t, a_t))$
- $Q_{t+1}(x_t, a_t) = \hat{c}(x_t, a_t) + \gamma \sum_{x' \in \mathcal{X}} \hat{\mathbf{P}}_a(x'|x_t) \underset{a' \in \mathcal{A}}{min} Q_t(x', a')$
  Both work if every state for J^π or every-action pair for Q^* is visited infinitely often.

Boltzmann policy

η: how greedy
Actions with large Q smaller prob

$$\pi(a|x) = \frac{e^{-\eta Q_t(x,a)}}{\sum_{a' \in \mathcal{A}} e^{-\eta Q_t(x,a')}}$$

Q-values are init to 0

Monte Carlo

No bias, does not bootstrap, long traject much variance

RL

We have a trajectory $\tau_k = \{x_{k,0}, c_{k,0}, x_{k,1}, c_{k,1}, \dots, c_{k,T-1}, x_{k,T}\}$ obtained by π
Compute loss

$$L(\tau_k) = \sum_{t=0}^{T-1} \gamma^t c_{k,t}$$

Update

$$J_{k+1}(x_{k,0}) = J_k(x_{k,0}) + \alpha_k(L(\tau_k) - J_k(x_{k,0}))$$

Policy evaluation

We have a trajectory $\tau_k = \{x_{k,0}, a_{k,0}, c_{k,0}, \dots, a_{k,T-1}, c_{k,T-1}, x_{k,T}\}$ obtained with π
Compute loss

$$L(\tau_k) = \sum_{t=0}^{T-1} \gamma^t c_{k,t}$$

Update

$$Q_{k+1}(x_k, a_k) = Q_k(x_k, a_k) + \alpha_k(L(\tau_k) - Q_k(x_k, a_k))$$

TD(λ)

On-policy
Given a sample (x_t, c_t, x_{t+1}) where actions are selected from π
Compute

$$\begin{gathered} z_{x+1}(x) = \lambda\gamma z_t(x) + \mathbb{I}(x = x_t) \\ J_{t+1}(x) = J_t(x) + \alpha_t z_{t+1}(x)[c_t + \gamma J_t(x_{t+1}) - J_t(x_t)] \end{gathered}$$

For $\lambda = 0$ we recover TD(0)

TD(0)

Only single transition to update, but much bias because of the use of current J to compute next estimate

$$J_{k+1}(x_k) = J_k(x_k) + \alpha(c_k + \gamma J_k(x_{k+1}) - J_k(x_k))$$

Temporal difference

$$\delta_k = c_k + \gamma J_t(x_{t+1} - J_t(x_t))$$

Q-learning

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t \left[c_t + \gamma \underset{a' \in \mathcal{A}}{min} Q_t(x_{t+1}, a') - Q_t(x_t, a_t)\right]$$

Off-policy
Converges if every stat-action pair is visited infinitely often
May suck if the exploration-exploitation policy is not well-tuned or high variance env.

Sarsa

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t \left[c_t + \gamma Q_t(x_{t+1}, a_{t+1}) - Q_t(x_t, a_t)\right]$$

On-policy: learns one policy while following another
May be more appropriate of offline learning where agent does not explore much
Needs to be combined with policy improvement, ϵ-greedy with decaying ϵ
Often leads to more stable updates

Exploration vs exploitation

UCB

Observation of single cost, but underlying distribution. Bandit

- Execute each action once
  $\hat{c}$: average cost
  after -: confidence interval
  $a^* = argmin(\hat{c}\ (a) - \sqrt{\frac{2logt}{N_t(a)}})$
  update $\hat{c}$ for selected action a: $\hat{c}(a) = \hat{c}(a) + \frac{1}{N_{t+1}(a)}(c_t(a) - \hat{c}(a))$
  bound: $R_T \leq \sum_{a \neq a^*} \frac{8logT}{\Delta(a)} + 2\sum_{a \in \mathcal{A}} \Delta(a)$
  $R_T \leq \sqrt{CNTlogT}$
  Grows slower than just T
  Used for MCTS with $\min_i Q(i) - c\sqrt{\frac{log(t)}{N(i)}}$

Weighted majority algo.

- Given set of N "predictors" $\eta < \frac{1}{2}$
- Init predictor weights $w_0(n) = 1, n, \cdots, N$
- Make prediction based on the weighted majority vote
- Update wights of all wrong predictors as

$$w_{t+1}(n) = w_t(n)(1-\eta)$$

EWA

Can see all costs. Weather

- Set of N actions $\eta > 0$
- Init weights to $w_0(a) = 1, a \in \mathcal{A}$
- Select action according to prob

$$p_t(a) = \frac{w_t(a)}{\sum_a{}' \in \mathcal{A} w_t(a')}$$

- Update all weights of all actions as

$$w_{t+1}(a) = w_t(a)e^{-\eta c_t(a)}$$

Bound: $R_T \leq \sqrt{\frac{T}{2} log\ N}$

Average regret per step: $\frac{R_T}{T} \leq \sqrt{\frac{log\ N}{2T}} \underset{T \to \infty}{\to} 0$

EXP3

Only get cost from the one you choose. Bandit

- Given a set of N actions and $\eta > 0$
- Init weights $w_0(a) = 1, a \in \mathcal{A}$
- Select action according to

$$p_t(a) = \frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')}$$

- Update weights of all actions as

$$w_{t+1}(a_t) = w_t(a_t) e^{-\eta \frac{c_t(a_t)}{p_t(a_t)}}$$

Regret bound: $R_T \leq \sqrt{2TNlogN}$

Tidbits

No free lunches: for every task in which your decision rule does great, there is another task in which it does terrible
Inductive bias: what you get if you make assumptions, but i greatly decreases the search space. **Wrong assumptions** lead to missing the bullseye (has the ability to change the accuracy)
Variance: due to "noise in the training data", a source to low acc. and could show as outputs of similar inputs are very different. (has ability to change precision)

Inversion

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix} \mathbf{A}^{-1} = \begin{bmatrix} \frac{1}{a} & -\frac{b}{ad} & \frac{be-cd}{adf} \\ 0 & \frac{1}{d} & -\frac{e}{df} \\ 0 & 0 & \frac{1}{f} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix} \mathbf{A}^{-1} = \begin{bmatrix} \frac{1}{a} & & \\ -\frac{b}{ad} & \frac{1}{d} & \\ \frac{be-cd}{adf} & -\frac{e}{df} & \frac{1}{f} \end{bmatrix}$$