Algorithm

Algorithm 1: Social multi-agent multi-armed bandits **S-MAMAB**

Input : Task settings $\mathcal{K}\left(\overrightarrow{\mu}, \overrightarrow{\sigma}^2, \overrightarrow{\rho}(t)\right)$, Social settings $\mathcal{S}$, Number of trials T

Param : Parameters $\Theta = \left\{\Theta^{(0)}, \Theta^{(\mathrm{AcS})}, \Theta^{(\mathrm{BeU})}, \Theta^{(\mathrm{AcL})}, \Theta^{(\mathrm{SoS})}, \Theta^{(\mathrm{SoL})}\right\}$

$\Theta^{(0)} = \left\{\mu_0, \sigma_0^2 \dots\right\}$ $\quad\Theta^{(\mathrm{AcS})} = \left\{\tau_{\mathrm{s}}, \epsilon_{\mathrm{g}} \dots\right\}$ $\quad\Theta^{(\mathrm{BeU})} = \left\{\overrightarrow{\sigma}^2 \text{ or } \sigma_\epsilon^2 \dots\right\}$

$\Theta^{(\mathrm{AcL})} = \left\{\beta_{\mathrm{u}} \dots\right\}$ $\quad\Theta^{(\mathrm{SoS})} = \left\{\beta_h, \dots\right\}$ $\quad\Theta^{(\mathrm{SoL})} = \left\{\eta_{\mathrm{s}}, \alpha_{\mathrm{s}}, \dots\right\}$

`// Have not considered drift noise` ξ

Output: $\mathcal{Z}_t \leftarrow \left\{\mathbf{Q}_t, \mathbf{A}_t, \mathbf{Y}_t, \mathbf{Y}_t^C, \mathbf{M}_t, \mathbf{V}_t, \left[\mathbf{P}_t, \mathbf{G}_t, \mathbf{W}_t, \mathbf{C}_t, \mathbf{Q}_t^{(\mathrm{AcL})}, \mathbf{Q}_t^{(\mathrm{SoL})}\right]\right\}$

Initialization
- `Initialize`: $\Theta^{(0)} \longmapsto (\mathbf{M}_0, \mathbf{V}_0, \mathbf{P}_0)$
  - $\hookrightarrow$ `Initialize` $\in \{$`InitEqualProb`, `InitNormUnifProb`$\}$

for $t = 1 \to T$ **do**
- *ActionSampling (AcS)*
  - $$(\mathbf{A}_t, [\mathbf{P}_t]) \leftarrow \begin{cases} \texttt{WeightedChoice}\left(\mathbf{P}_0 \odot \overrightarrow{\rho}(t)\right) & \text{if } t = 1 \text{ (or } t < T_{AcS}) \\ \texttt{SampleAction}\left(\mathbf{Q}_{t-1}, \overrightarrow{\rho}(t), \Theta^{(\mathrm{AcS})}\right) & \text{otherwise} \end{cases}$$
  - $\hookrightarrow$ `SampleAction` $\in \{$`Softmax`(τ_{s}), `Argmax`, `Greedy`(ϵ_{g}), `Thompson`$\}$
- *RewardSampling (ReS)*
  - `SampleReward`: $(\mathbf{A}_t, \mathcal{K}) \longmapsto \mathbf{Y}_t$
- *BeliefUpdating (BeU)*
  - `UpdateBelief`: $\left(\mathbf{M}_{t-1}, \mathbf{V}_{t-1}, \mathbf{A}_t, \mathbf{Y}_t, \Theta^{(\mathrm{BeU})}\right) \longmapsto (\mathbf{M}_t, \mathbf{V}_t, [\mathbf{G}_t])$
    - $\hookrightarrow$ `UpdateBelief` $\in \left\{\texttt{BMT}\left(\overrightarrow{\sigma}^2 \text{ or } \sigma_\epsilon^2\right)\right\}$
- *UtilityUpdating*
  - *ActionLearning (AcL)*
    - `LearnAction`: $\left(\mathbf{M}_t, \mathbf{V}_t, \Theta^{(\mathrm{AcL})}\right) \longmapsto \mathbf{Q}_t^{(\mathrm{AcL})}$
      - $\hookrightarrow$ `LearnAction` $\in \{\texttt{UCB}\left(\beta_{\mathrm{u}}\right), \texttt{MGE}, \texttt{VGE}\}$
  - *SocialLearning (SoL) & SocialSetting (SoS)*
    - **if** $t = 1$ *(or* $t < T_{SoL}$*)* **then**
      - $\mathbf{Q}_t^{(\mathrm{SoL})} \leftarrow \mathbf{0}$
    - **else**
      - `SetSocial`: $\left(\mathcal{Z}_{t-1}, \mathcal{S}, \overrightarrow{\rho}(t-1), \Theta^{(\mathrm{SoS})}\right) \longmapsto (\mathbf{C}_t, \mathbf{W}_t)$
      - `LearnSocial`: $\left(\mathbf{C}_t, \mathbf{W}_t, \Theta^{(\mathrm{SoL})}\right) \longmapsto \mathbf{Q}_t^{(\mathrm{SoL})}$
  - `UpdateUtility`: $\left(\mathbf{Q}_t^{(\mathrm{AcL})}, \mathbf{Q}_t^{(\mathrm{SoL})}\right) \longmapsto \mathbf{Q}_t$
- Update cumulative rewards $\mathbf{Y}_t^C \leftarrow \mathbf{Y}_t + \mathbf{Y}_{t-1}^C$
- Save $\mathcal{Z}_t \leftarrow \left\{\mathbf{Q}_t, \mathbf{A}_t, \mathbf{Y}_t, \mathbf{Y}_t^C, \mathbf{M}_t, \mathbf{V}_t, \left[\mathbf{P}_t, \mathbf{G}_t, \mathbf{W}_t, \mathbf{C}_t, \mathbf{Q}_t^{(\mathrm{AcL})}, \mathbf{Q}_t^{(\mathrm{SoL})}\right]\right\}$

Descriptions of parameters and states

Notations and supporter functions

- If a matrix $\mathbf{X}$ is of dimension K arms $\times$ N agents, unless specified otherwise:
  - ▷ $\mathbf{x}_i$ signifies the *column* vector of values for the i-th agent,
  - ▷ $\mathbf{x}'_k$ signifies the *row* vector of values for the k-th arm
  - ▷ In other words, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N] = [\mathbf{x}'_1, \mathbf{x}'_2 \dots \mathbf{x}'_K]^\top$
- $x \sim \mathcal{X}$ is a random variable sampled from $\mathcal{X}$ then $\mathbf{X} \overset{iid}{\sim} \mathcal{X}^{m \times n}$ represents the matrix $\mathbf{X}$ of size $m \times n$ of random variables sampled from $\mathcal{X}$ independently, i.e. $x_{ij} \sim \mathcal{X}$. Similarly, if $\mathcal{X}$ is parameterized by θ then $\mathbf{X} \overset{iid}{\sim} \mathcal{X}(\Theta)$ where $\dim \Theta = (m, n)$ then $x_{ij} \sim \mathcal{X}(\theta_{ij})$ sampled independently.
- $\odot$ is the element-wise multiplication. If $\dim \mathbf{A} = \dim \mathbf{B} = (m, n)$, $\dim \mathbf{a} = (m, 1)$ and $\dim \mathbf{a}' = (1, n)$:

- ▷ $(\mathbf{A} \odot \mathbf{B})_{ij} = (\mathbf{B} \odot \mathbf{A})_{ij} = a_{ij} b_{ij}$
  - ▷ $(\mathbf{a} \odot \mathbf{B})_{ij} = (\mathbf{B} \odot \mathbf{a})_{ij} = a_i b_{ij}$
  - ▷ $(\mathbf{a}' \odot \mathbf{B})_{ij} = (\mathbf{B} \odot \mathbf{a}')_{ij} = a'_j b_{ij}$
- Special matrices and vectors:
  - ▷ $\mathbf{e}_i^{(n)}$ is a column unit vector of size n where only $e_i^{(n)} = 1$, and $e_j^{(n)} = 0 \ \forall j \neq i$.
  - ▷ Hence the identity matrix of size $n \times n$ is $\mathbf{I}_n = \left[\mathbf{e}_1^{(n)} \dots \mathbf{e}_n^{(n)}\right]$
  - ▷ The column vector of n ones is $\mathbf{1}_n$ while the matrix of all ones of size $m \times n$ is $\mathbf{1}_{m \times n}$
- L^p Normalization:
  - ▷ $\|\cdot\| = \|\cdot\|_2$ is the L^2 norm, while $\|\cdot\|_p$ is the L^p norm: $\mathbf{x} \longmapsto (\sum_{i=1}^n x_i^p)^{1/p}$ where $n = \dim \mathbf{x}$ and $p \neq 0$
  - ▷ Column L^p normalization ψ_p. For column vectors: $\mathbf{x} \longmapsto \mathbf{x}/\|\mathbf{x}\|_p$. For matrix: $\mathbf{X} \longmapsto [\psi_p(\mathbf{x}_1) \dots \psi_p(\mathbf{x}_n)]$
  - ▷ Row L^p normalization ψ'_p: $\mathbf{X} \longmapsto \psi_p\left(\mathbf{X}^\top\right)^\top$
  - ▷ For the sake of completion, though not necessary, to use all elements (like Frobenius norm), $\Psi_p : \mathbf{X} \longmapsto \mathbf{X}/\|\mathbf{X}\|_p$ where $\|\mathbf{X}\|_p = \left(\sum_{i,j} x_{ij}^p\right)^{1/p}$
- Min/max normalization.
  - ▷ Max-normalization per column for non-negative matrix: $\psi_{\max} : \mathbb{R}_{\geq 0}^{m \times n} \to \mathbb{R}_{\geq 0}^{m \times n} : \mathbf{X} \longmapsto \left[\frac{\mathbf{x}_1}{\max \mathbf{x}_1} \dots \frac{\mathbf{x}_n}{\max \mathbf{x}_n}\right]$. Limit to only positive or non-negative matrices (i.e. all elements are either > 0 or ≥ 0, respectively). For all zeros columns, either turn them all to 1's or 0's, depending on the need.
  - ▷ Min-max normalization per columns $\psi_{\text{minmax}} : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n} : \mathbf{x}_i = [\mathbf{X}]_i \longmapsto \frac{\mathbf{x}_i - \min \mathbf{x}_i}{\max \mathbf{x}_i - \min \mathbf{x}_i}$ Again, depending on the need, columns where $\min \mathbf{x} = \max \mathbf{x}$ can be turned to all 1's or 0's. Additionally, could also bottom-clip with $\psi_{\text{minmax}}(\mathbf{X}, x_\star)$ so $\mathbf{x}_i \longmapsto \max\left[\psi_{\text{minmax}}(\mathbf{x}_i), x_\star\right]$
  - ▷ Similarly, one can define $\psi_{\min}$ for column min normalization
  - ▷ And for row normalizations: $\psi'_{\max}$, $\psi'_{\min}$, ψ'_{minmax}
  - ▷ For the sake of completion, to use all elements for global min/max $\Psi_{\max}$, $\Psi_{\min}$, Ψ_{minmax}
- Normalized uniform `NormUniform`: $(m, n) \in \mathbb{N} \times \mathbb{N} \longmapsto \mathbf{X} = \psi_1(\mathbf{U}) \in \mathbb{R}^{m \times n}$, in which $\mathbf{U} \overset{iid}{\sim} \mathcal{U}_{[0,1]}^{m \times n}$

  Alternative notation: matrix of random variables $\mathbf{X} \sim \widehat{\mathcal{U}}^{m \times n}$
- Index sets, for $\mathbf{x} \in \mathbb{Z}_2^n$ where $\mathbb{Z} = \{0, 1\}$
  - ▷ For (column or row) vectors: $\mathcal{I}(\mathbf{x}) = \{i | x_i = 1\}$
  - ▷ For matrix, column-wise: $\mathcal{I}(\mathbf{X}) = \{i | x_{ij} = 1\}$
  - ▷ For matrix, row-wise: $\mathcal{I}'(\mathbf{X}) = \{j | x_{ij} = 1\}$
- Choices
  - ▷ `WeightedChoice`
    - ◦ For column vector $\mathbf{w} \in \mathbb{R}_+^n \longmapsto \mathbf{e}_i^{(n)} \in \mathbb{Z}_2^n$, where index $i \in [1, n] \subset \mathbb{N}$ is chosen with probability $\mathbf{p} = \psi_1(\mathbf{w})$
    - ◦ For matrix $\mathbf{W} \in \mathbb{R}_+^{m \times n} \longmapsto [\texttt{WeightedChoice}(\mathbf{w}_1) \dots \texttt{WeightedChoice}(\mathbf{w}_n)]$
  - ▷ `Argmax` (to vectorize or matricize `argmax`)
    - ◦ For column vector $\mathbf{x} \in \mathbb{R}^n \longmapsto \mathbf{e}_i^{(n)} \in \mathbb{Z}_2^n$, where index $i = \texttt{argmax}(\mathbf{x})$,
    - ◦ For matrix $\mathbf{X} \in \mathbb{R}^{m \times n} \longmapsto [\texttt{Argmax}(\mathbf{x}_1) \dots \texttt{Argmax}(\mathbf{x}_n)]$
- Moving average (TBD):
  - ▷ Cumulative moving average `CMA`
  - ▷ Exponential moving average $\texttt{EMA}(\alpha_{\text{EMA}})$

General inputs and settings

- T is the number of *trials*, i.e., discrete time steps $t \in \mathbb{N}$.
- $\mathcal{K}$ describes the K *arms* (tasks) with
  - ▷ mean *reward* (column) vector $\overrightarrow{\mu} \in \mathbb{R}^K$
  - ▷ *uncertainty* (variance) vector $\overrightarrow{\sigma}^2 \in \mathbb{R}_+^K$
  - ▷ and arm dynamic *availability* vector $\overrightarrow{\rho}(t)$

    ⋆ For now $\overrightarrow{\rho}(t) \in \mathbb{Z}_2^K$ can just be a binary mask as a function of time, but could also be considered as a probability to signify probabilistic availability of the arms.
- The *social* settings $\mathcal{S}$ contains information about the N agents and how to construct
  - ▷ the *content* matrix $\mathbf{C}_t = \mathbf{C}(t) \in \mathcal{C}^{K \times N}$, i.e. social "mass" to influence utility, where $\mathcal{C} = \mathbb{Z}_2$ or $\mathbb{R}_+$
  - ▷ and the agent *social network* $\mathbf{W}_t = \mathbf{W}(t) \in \mathcal{W}^{N \times N}$ where $\mathcal{W} = \mathbb{Z}_2$ or $\mathbb{R}_+$; which can be
    - ◦ either a predefined $\mathbf{W}^{(0)}$ adjacency network, In other words, static social network $\mathbf{W}_t = \mathbf{W}^{(0)} \ \forall t$

- ∘ or defined with a homophily constructor, defining which *content* matrix $\mathbf{C}^{(h)}$ to build from, and homophily factor $\beta_h \in \Theta^{(\text{SoS})}$, and how/whether to normalize

  ⋆ The content matrices $\mathbf{C}$ or $\mathbf{C}^{(h)}$ do not have to be similar, and can be constructed from the previous arm choice bipartite matrix $\mathbf{A}_{t-1}$, or from the maximum belief mean $\mathbf{M}_{t-1}$ (or past reward $\mathbf{Y}_{t-1}$ or cumulative rewards $\mathbf{Y}^C_{t-1}$)

Hyper/free parameters

- Initial/prior parameters $\Theta^{(0)}$
  - ▷ μ_0 is the initial belief mean, set to be optimistic
  - ▷ σ_0^2 is the initial belief uncertainty
- Action sampling parameters $\Theta^{(\text{AcS})}$
  - ▷ τ_s sets `Softmax`'s temperature
  - ▷ ϵ_g is the free parameter for the ϵ-greedy algorithm `Greedy`
- Belief updating parameters $\Theta^{(\text{BeU})}$
  - ▷ Usage of either task/arm uncertainty $\vec{\sigma}^2$ or a scalar error term σ^2 for *Bayesian mean tracker* (`BMT`)
    - ⋆ Right now not considering drift noise ξ
- Action learning parameters for exploitation-exploration learning $\Theta^{(\text{AcL})}$
  - ▷ β_u sets the exploration factor for the *Upper-confidence-bound* sampling (`UCB`)
- Social setting parameters $\Theta^{(\text{SoS})}$
  - ▷ β_h sets the homophily factor
- Social learning parameters $\Theta^{(\text{SoL})}$
  - ▷ η_s is the scaling factor
  - ▷ α_s is the power factor

Outputs and States

The state sets $\mathcal{Z}(t)$. (*aux*) signifies which ones are intermediate and optional to save, also not necessarily appearing in outputs of all function and hyperparameter choices.

- $\mathbf{Q}_t \in \mathbb{R}^{K\times N}$ is the utility matrix of each agent per each task
- $\mathbf{A}_t \in \mathbb{Z}_2^{K\times N}$ is the binary choice matrix at time t. Each agent only chooses 1 arm at each time step. (i.e. $\|\mathbf{a}_i(t)\| = 1$, and cardinality $|\mathcal{I}(\mathbf{a}_i(t))| = 1$)
- $\mathbf{Y}_t, \mathbf{Y}^C_t \in \mathbb{R}^{K\times N}$ are the actual reward at time t, from the reward sampling steps, and the cumulative reward matrix
- $\mathbf{M}_t \in \mathbb{R}^{K\times N}$ and $\mathbf{V}_t \in \mathbb{R}_+^{K\times N}$ are the posterior (belief) mean reward matrix and uncertainty (variance) matrix
- (*aux*) $\mathbf{P}_t \in \mathbb{R}_+^{K\times N}$ is the probability matrix constructed from $\mathbf{Q}_{t-1}$, most likely from using `Sotfmax`, which can then be used to decide $\mathbf{A}_t$
- (*aux*) $\mathbf{G}_t \in \mathbb{R}_+^{K\times N}$ is the Kalman gain constructed from the Bayesian mean tracker process
- (*aux*) $\mathbf{W}_t \in \mathcal{W}^{N\times N}$ and $\mathbf{C}_t \in \mathcal{C}^{K\times N}$ are the social agent network and content bipartite matrix, respectively. See the above section describing *social settings* $\mathcal{K}$ for more description
- (*aux*) $\mathbf{Q}_t^{(\text{AcL})}, \mathbf{Q}_t^{(\text{SoL})} \in \mathbb{R}^{K\times N}$ are the utility matrices constructed from the action learning (e.g. `UCB`) and social learning processes, respectively.

Processes and functions

Initialization

- `InitBelief`: $\left(\mu_0, \sigma_0^2\right) \longmapsto \left(\mathbf{M}_0 = \mu_0 \mathbf{1}_{K\times N}, \mathbf{V}_0 = \sigma_0^2 \mathbf{1}_{K\times N}\right)$
- `InitEqualProb`: $\left(\mu_0, \sigma_0^2\right) \longmapsto \texttt{InitBelief}\left(\mu_0, \sigma_0^2\right) \cup \left(\mathbf{P}_0 = \frac{1}{K}\mathbf{1}_{K\times N}\right)$
- `InitNormUnifProb`: $\left(\mu_0, \sigma_0^2\right) \longmapsto \texttt{InitBelief}\left(\mu_0, \sigma_0^2\right) \cup \left(\mathbf{P}_0 \sim \widehat{\mathcal{U}}^{K\times N}\right)$
- (TBD) maybe somehow allowing an initial exploring phase, e.g. without social learning

Action Sampling (AcS)

- General inputs $(\mathbf{Q} \leftarrow \mathbf{Q}_{t-1}, \vec{\rho} \leftarrow \vec{\rho}(t))$
- `Softmax`(τ_s)
  - ▷ $\mathbf{P} = \psi_1 \left[\exp\left(\mathbf{Q}/\tau_\text{s}\right) \odot \vec{\rho}\,\right]$ where exp is just element-wise exponential function
  - ▷ $\mathbf{A} = \texttt{WeightedChoice}(\mathbf{P})$
- `Argmax`: $\mathbf{A} = \texttt{Argmax}\left(\mathbf{Q} \odot \vec{\rho}\,\right)$

- `Greedy`(ϵ_g)
    - ▷ $\mathbf{I}_\text{max} = \texttt{Argmax}(\mathbf{Q} \odot \overrightarrow{\rho})$
    - ▷ $\mathbf{I}_\text{other} = (\mathbf{1}_{K\times N} - \mathbf{I}_\text{max}) \odot \overrightarrow{\rho}$
    - ▷ $\mathbf{P} = (1-\epsilon_\text{g})\mathbf{I}_\text{max} + \epsilon_\text{g}\psi_1(\mathbf{I}_\text{other})$
    - ▷ $\mathbf{A} = \texttt{WeightedChoice}(\mathbf{P})$
- `Thompson` (TBD) unclear how to integrate social learning into distribution
    - ▷ Generally $\mathbf{A} = \texttt{Argmax}\left[\mathbf{X} \overset{iid}{\sim} \mathcal{N}\left(\mathbf{\Lambda}, \beta_\text{u}\mathbf{\Sigma}\right)\right]$ where $\mathbf{\Lambda} \leftarrow \mathbf{M}_{t-1} \odot \overrightarrow{\rho}, \mathbf{\Sigma} \leftarrow \mathbf{V}_{t-1} \odot \overrightarrow{\rho}$, and $\beta_\text{u} \in \Theta^\text{AcL}$ is from `UCB`
    - ▷ But maybe with social learning, the variance (uncertainty) is reduced based on $\mathbf{Q}^\text{(SoL)}$, e.g with exponential decay

    $$\mathbf{\Sigma} \leftarrow \psi_\text{max}\left[\exp\left(-\mathbf{Q}^\text{(SoL)}\right) \odot \overrightarrow{\rho}\right] \odot \mathbf{V}_{t-1}$$

    This means that the smallest $q_{ij}^\text{(SoL)}$ will have the same uncertainty as v_{ij}, while higher social utility decays such uncertainty. Note: could also use ψ_minmax instead of ψ_max, and also a decaying factor in the exponential

Reward Sampling (ReS)

$$\mathbf{Y} = \mathbf{A} \odot \mathbf{y}' \text{ where } \mathbb{R}^{1\times N} \ni \mathbf{y}' \overset{iid}{\sim} \mathcal{N}\left(\overrightarrow{\mu}^\top \mathbf{A}, \overrightarrow{\sigma}^{2\top} \mathbf{A}\right)$$

Belief Updating (BeU)

$$\begin{cases} \mathbf{M}_t &= \mathbf{M}_{t-1} + \Delta\mathbf{M}_t \\ \mathbf{V}_t &= \mathbf{V}_{t-1} + \Delta\mathbf{V}_t \end{cases} \text{where} \begin{cases} \Delta\mathbf{M}_t &= \mathbf{G}_t^A \odot (\mathbf{Y}_t - \mathbf{M}_{t-1}) \\ \Delta\mathbf{V}_t &= -\mathbf{G}_t^A \odot \mathbf{V}_{t-1} \\ \mathbf{G}_t^A &= \mathbf{G}_t \odot \mathbf{A}_t \\ \mathbf{G}_t &= \dfrac{\mathbf{V}_{t-1}}{\mathbf{V}_{t-1} + \mathbf{\Sigma}} \text{ (element-wise division)} \\ \mathbf{\Sigma} &= \begin{cases} \overrightarrow{\sigma}^2 \mathbf{1}_{1\times N} & \text{if error is task dependent} \\ \sigma_\epsilon^2 \mathbf{1}_{K\times N} & \text{if use free parameter error} \end{cases} \end{cases}$$

- `BMT`$(\overrightarrow{\sigma}^2$ or $\sigma_\epsilon^2)$
- no consideration of drift noise here ξ

Utility Updating

- `UpdateUtility` $\mathbf{Q}_t \leftarrow \mathbf{Q}_t^\text{(AcL)} + \mathbf{Q}_t^\text{(SoL)}$
  Additionally could also consider weighting them like $\mathbf{Q}_t \leftarrow \gamma\mathbf{Q}_t^\text{(AcL)} + (1-\gamma)\mathbf{Q}_t^\text{(SoL)}$

Action Learning (AcL)

$$\mathbf{Q}_t^\text{(AcL)} = \mathbf{M}_t + \beta_\text{u}\mathbf{V}_t$$

- `UCB`(β_u) (like above)
- `MGE`: $\mathbf{Q}_t^\text{(AcL)} = \mathbf{M}_t$ (i.e. $\beta_\text{u} = 0$)
- `VGE`: $\mathbf{Q}_t^\text{(AcL)} = \mathbf{V}_t$

Social Learning (SoL)

- `SetSocial` (SoS) with optional `HomophilyConstruct` $\equiv \mathcal{H}$

$$\mathbf{C}_t = \begin{cases} \mathbf{A}_{t-1} \\ \mathbf{M}_{t-1} \\ \mathbf{Y}_{t-1} \\ \texttt{Argmax}\left(\overrightarrow{\rho}(t-1) \odot \mathbf{Y}_{t-1}\right) \\ \texttt{Argmax}\left(\overrightarrow{\rho}(t-1) \odot \mathbf{Y}^C_{t-1}\right) \\ \texttt{Argmax}\left(\overrightarrow{\rho}(t-1) \odot \mathbf{M}_{t-1}\right) \end{cases} \qquad \left[\longrightarrow \begin{cases} \texttt{CMA} \\ \texttt{EMA} \end{cases}\right]$$

$\mathbf{C}_t^{(h)}$ constructed similarly if using $\mathcal{H}$

$$\mathbf{W}_t = \begin{cases} \mathbf{W}^{(0)} & \text{if predefined} \\ \mathcal{H}\left(\mathbf{C}_t^{(h)}, \beta_h\right) & \text{if using homophily} \end{cases} \qquad \left[\longrightarrow \begin{cases} \psi_1 \text{ or } \psi_1' \\ \psi_2 \text{ or } \psi_2' \end{cases}\right]$$

$$\mathcal{H}\left(\mathbf{C}, \beta_h\right) \overset{iid}{\sim} \text{Bern}(\mathbf{P}_h) \text{ where } \mathbf{P}_h = \begin{cases} \psi_1\left[\left(\mathbf{C}^\top\mathbf{C}\right)^{\beta_h}\right] \\ \psi_1\left[\left(K - \mathbf{C}^\top\mathbf{C}\right)^{-\beta_h}\right] \end{cases}$$

$$\mathbf{S}_t = \mathbf{C}_t\mathbf{W}_t \qquad \left[\longrightarrow \begin{cases} \psi_1 \text{ or } \psi_1' \\ \psi_2 \text{ or } \psi_2' \end{cases} \longrightarrow \begin{cases} \texttt{CMA} \\ \texttt{EMA} \end{cases}\right]$$

- `LearnSocial` (SoL)

$$\begin{aligned} \mathbf{Q}_t^{(\text{SoL})} &= \eta\mathbf{S}_t^\alpha && \text{where shorthand } \eta \leftarrow \eta_s, \alpha \leftarrow \alpha_s \\ &= \eta\left(\mathbf{C}_t\mathbf{W}_t\right)^\alpha && \text{(without normalization or moving averaging)} \\ &= \eta\left[\mathbf{A}_{t-1}\text{Bern}\left(\psi_1\left[\left(\mathbf{A}_{t-1}^\top\mathbf{A}_{t-1}\right)^{\beta_h}\right]\right)\right]^\alpha && \text{(simplest form)} \end{aligned}$$