Multi-agent System and Application

Lecture 5: Value Decomposition

Xiangfeng Wang

Multi-Agent Artificial Intelligence Laboratory,
School of Computer Science and Technology,
East China Normal University

2021 Spring

Table of Contents

Table of Contents

Introduction: Why we need VF

There are two main reasons urge us to adopt value decomposition techniques:

- **Efficient learning**: During learning, the exponentially increasing joint action space and only one global reward make the learning inefficient.
- **Efficient execution**: During execution, the practical scenario lacks communication or only access partial observation which urges agents make decision decentralized.

Introduction: The Key Concepts

Fully cooperative MARL

- $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$.
- $s \in S$ describes the true state of the environment. At each time step, each agent $a \in A \equiv \{1, \ldots, n\}$ chooses an action $u^a \in U$, forming a joint action $\mathrm{u} \in \mathrm{U} \equiv U^n$. This causes a transition on the environment according to the state transition function $P(s' \mid s, \mathrm{u}) : S \times \mathrm{U} \times S \to [0, 1]$ and $\gamma \in [0, 1)$ is a discount factor.
- All agents share the same reward function $r(s, \mathrm{u}) : S \times \mathrm{U} \to \mathbb{R}$.

Introduction: The Key Concepts

- Individual-Global-Max(IGM).

$$\arg\max_{\boldsymbol{u}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{u}) = \begin{pmatrix} \arg\max_{u_1} Q_1(\tau_1, u_1) \\ \vdots \\ \arg\max_{u_N} Q_N(\tau_n, u_N) \end{pmatrix}$$

Introduction: Typical Algorithms

Most of the algorithms are designed based on the IGM to resolve one/both of the two issues. And some of them jumped out of the IGM scope. We will introduce three typical algorithms for each issue separately, including

- **Under IGM**: VDN(AAMAS@2018) $\rightarrow$ QMIX(ICML@2018)
- **Out of IGM**: DCG(ICML@2020)

Table of Contents

Value Decomposition Networks

- The first one restricts individual and joint Q family satisfy IGM.
- Can be used to enable both efficient learning and efficient execution.
- A naive baseline for value factorization

Value Decomposition Networks

Recap and Background:

- *Deep Q-Networks (DQN)*.
- *Independent DQN*: Combines DQN with *independent Q-learning*, in which each agent independently and simultaneously learns it own Q-function.

Value Decomposition Networks

- Main Assumption–Addtivity: The total q value can decompose into the sum of individual q value.
- The additivity is the sufficient but not necessary condition of IGM.

$$Q((h^1, h^2, ..., h^d), (a^1, a^2, ..., a^d)) \approx \sum_{i=1}^{d} \tilde{Q}_i(h^i, a^i)$$

Value Decomposition Networks

- Compared with joint Q learning: lower the joint action space from $\mathcal{O}(\mathcal{A}^N)$ to $\mathcal{O}(N\mathcal{A})$ and enable decentralized execution.
- Compared with independant Q learning: it can take other agents into consideration during training which enable effective coordination learning.

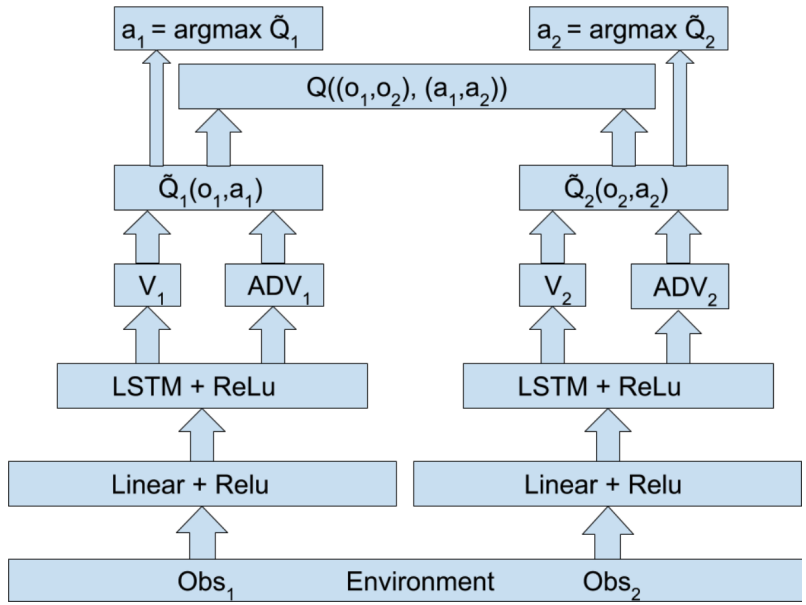


Figure 1: The network architecture of VDN.

Value Decomposition Networks

The cons of VDN can be summarized as follows:

- The additivity restriction is too tight and not every scenario satisfy this condition.
- What if we want to incorporate some auxiliary information during training like MADDPG?

Table of Contents

QMIX

- QMIX exploits how to richer the representation family of VDN under IGM.
- It propose another sufficient but not necessary condition: Monotonicity
- The Additivitity $\subset$ Monotonicity.

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \ \forall a \in A.$$

QMIX

- QMIX adopt a Mixing network to generate weights that satisfy Monotocity.
- The Mixing network also offer ways to incorporate state information in training stage.

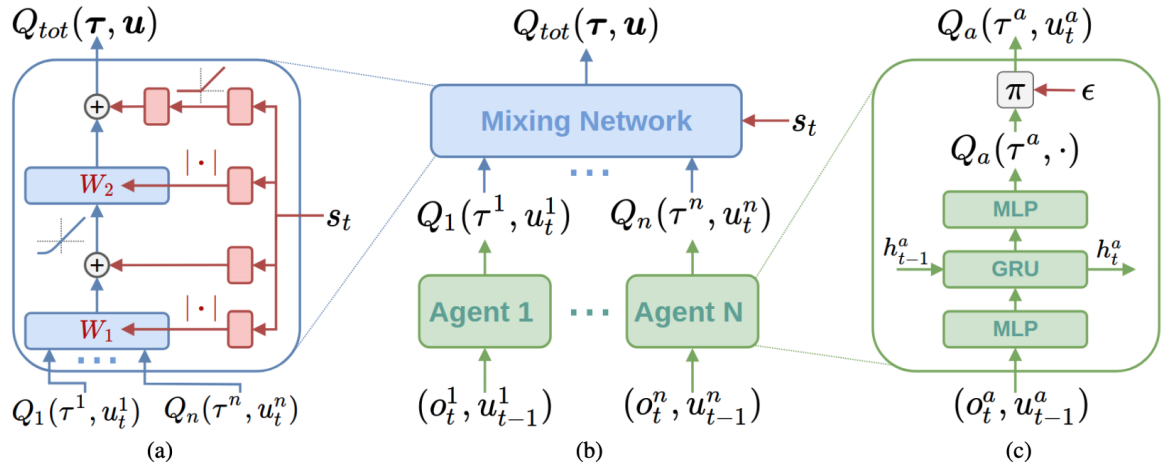


Figure 2: (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure.

QMIX

- It should be noted that the state information is used to generate weights instead of concatenate with individual Q.
- This is because Q_{tot} is allowed to depend on the extra state information in nonmonotonic ways.

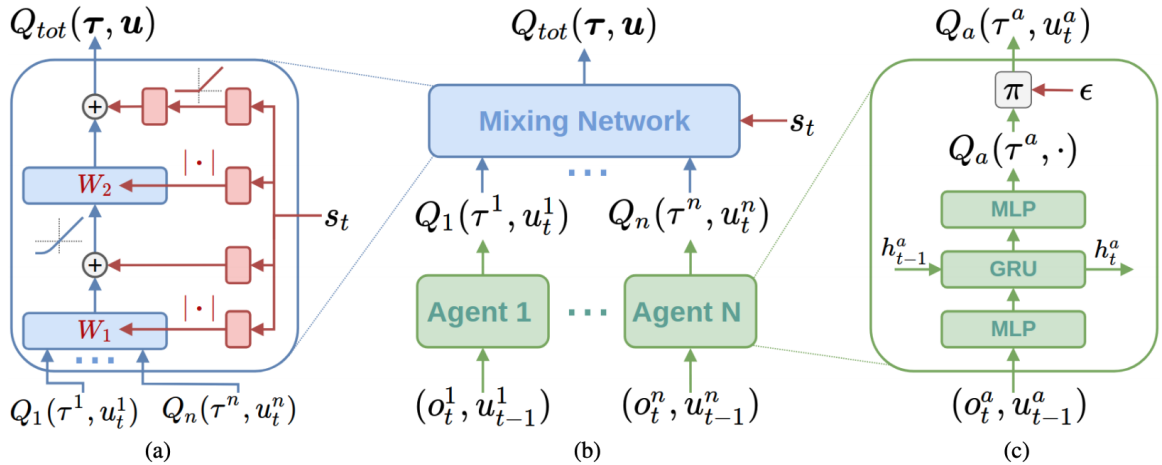


Figure 3: (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure.

QMIX

Both of VDN and QMIX follows the Deep-Q-Learning training procedure with:

- Select action based on the individual Q.
- Calculate loss and backpropagate gradient with the Q_{tot}.

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) \right)^2 \right], \qquad (6)$$

where b is the batch size of transitions sampled from the replay buffer, $y^{tot} = r + \gamma \max_{\mathbf{u}'} Q_{tot}(\boldsymbol{\tau}', \mathbf{u}', s'; \theta^-)$ and θ^- are the parameters of a target network as in DQN. (6) is

Other VF based on IGM

Enrich the representation of QMIX:

- QTRAN(ICML@2019): Exploit achieve full representation of IGM by transform it into constrained optimization, but is hard to scale to complex envs(SCII)
- QPLEX(ICLR@2021): Propose Attention Mixing network and Advantage IGM to achieve full representation of IGM and make it useful in SCII.
- ...

Extensions:

- QR-QMIX(AAMAS@2021): Incorporate Implicit Quantile Network with QMIX to handle the randomness on the reward.
- DOP(ICLR@2021): Introduce the value decomposition techs into Multi agent policy gradient.
- ...

Table of Contents

Nearly Decomposable Value functions

- VDN $\rightarrow$ QMIX $\rightarrow$ QTRAN achieves much richer representation on IGM
- However the IGM or Decetralized Execution itself encounter large variance and hard to approximate the optimal solution on high-level coordination/strong partial observability scenarios.
- NDQ hybrid the communication with QMIX to resolve the above issues.
- Every agent use local observations and the received messages to calculate the individual Q.

$$m_{ij} \sim \mathcal{N}(f_m(\tau_i, j; \boldsymbol{\theta}_c), \boldsymbol{I}),$$

$$Q_j(\tau_j, a_j, m_j^{in}).$$

Nearly Decomposable Value functions

To make the messages be practical and useful, we expect the messages have the following two properties:

- **Expressiveness**: The message passed to one agent should effectively reduce the uncertainty in its action-value function.
- **Succinctness**: Agents are expected to send messages as short as possible to the agents who need it and only when necessary.

They can be implemented with a regularizer:

$$J_c(\boldsymbol{\theta}_c) = \sum_{j=1}^{n} \left[I_{\boldsymbol{\theta}_c}(A_j; M_{ij} | \mathrm{T}_j, M_{(-i)j}) - \beta H_{\boldsymbol{\theta}_c}(M_{ij}) \right],$$

Nearly Decomposable Value functions

- The aforementioned regularizer cannot be optimized directly, NDQ instead optimize its variational lower bound.
- We skip the math part here but provide the alternative loss for the regularizer below:

$$\mathcal{L}_c(\boldsymbol{\theta}_c) = \mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T},j;\boldsymbol{\theta}_c)}\left[\mathcal{CE}\left[p(A_j|\mathbf{T})\|q_\xi(A_j|\mathrm{T}_j, M_j^{in})\right] + \beta D_{\mathrm{KL}}(p(M_{ij}|\mathrm{T}_i)\|r(M_{ij}))\right].$$

- Then the whole loss of NDQ can be calculated as: $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{TD}(\boldsymbol{\theta}) + \lambda\mathcal{L}_c(\boldsymbol{\theta}_c)$ and the $\mathcal{L}_{TD}(\boldsymbol{\theta})$ is the QMIX-like TD-error loss.

Nearly Decomposable Value functions

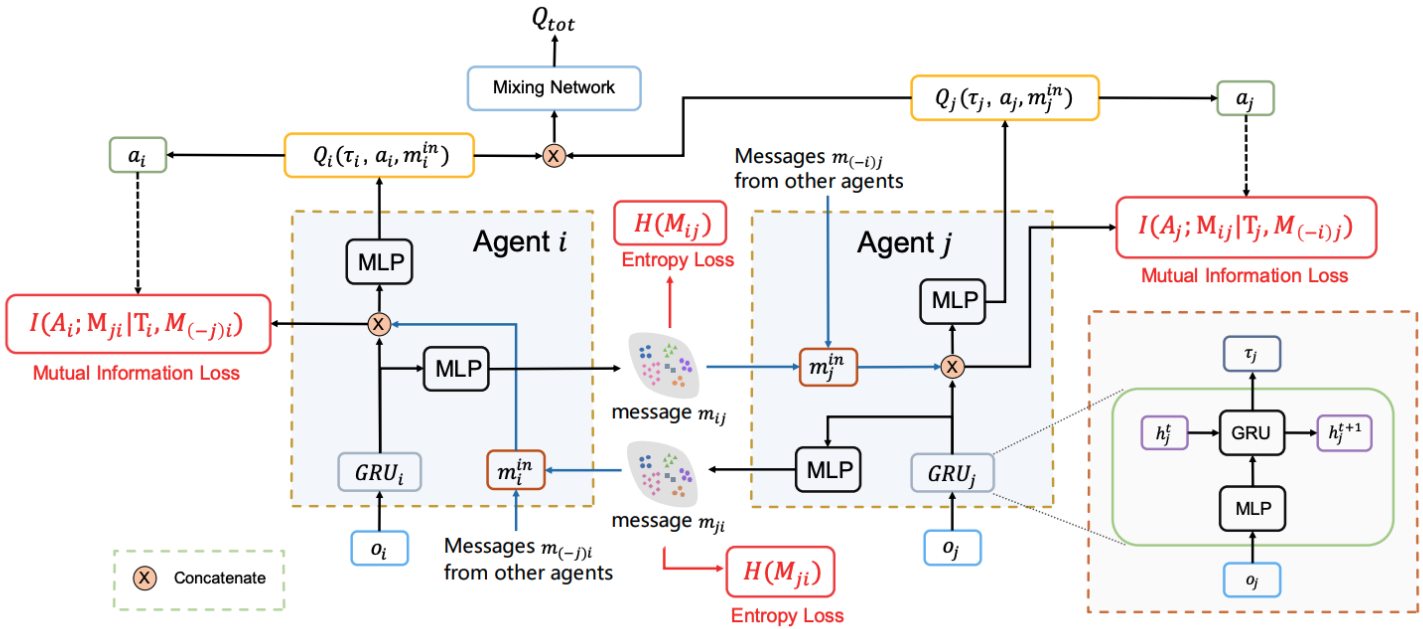


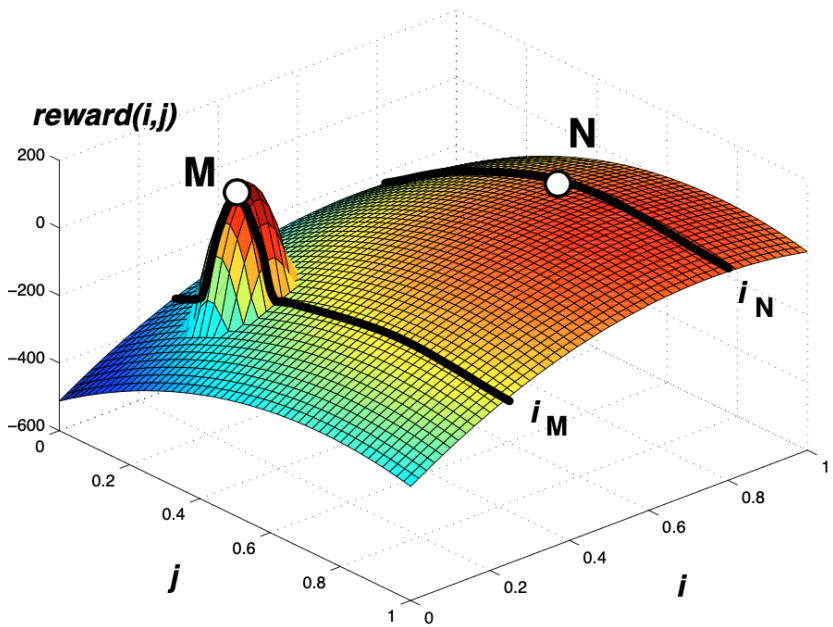
Figure 4: Schematics of NDQ. The message encoder generates an embedding distribution that is sampled and concatenated with the current local history to serve as an input to the local action-value function. Local action values are fed into a mixing network to to get an estimation of the global action value.

Table of Contents

Deep Coordination Graphs

DCG consider another issue brought by IGM: relative over-generalization.

- The axes i and j are the various actions that agents A_i and A_j may perform,
- the axis rewards (i, j) is the joint reward received by the agents from a given joint action $\langle i, j \rangle$.
- However, the average of all possible rewards for action i_M, of agent A_i is lower than the average of all possible rewards for action i_N. Thus, the agents tend to converge to N.



Deep Coordination Graphs

To address the relative over-generalization, DCG adopts represents the value function as a CG with pairwise payoffs and individual utilities.

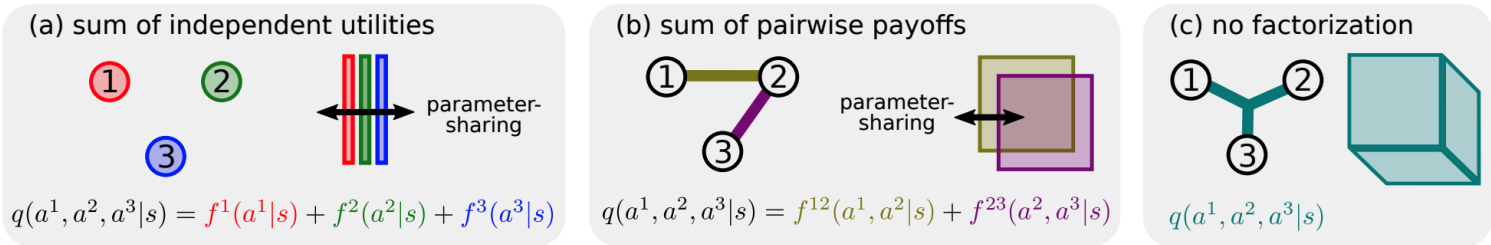


Figure 5: Examples of value factorization for 3 agents: (a) sum of independent utilities (as in VDN) corresponds to an unconnected CG. QMIX uses a monotonic mixture of utilities instead of a sum; (b) sum of pairwise payoffs which correspond to pairwise edges; (c) no factorization (as in QTRAN) corresponds to one hyper-edge connecting all agents. Factorization allows parameter sharing between factors, shown next to the CG, which can dramatically improve the algorithm's sample complexity

Deep Coordination Graphs

- Under the factor graph, we denote utility function as f^i for every agent i and payoff function as f^{ij} for payoff between i and j.

$$q^{\mathrm{CG}}(s_t, \boldsymbol{a}) := \frac{1}{|\mathcal{V}|}\sum_{v^i \in \mathcal{V}} f^i(a^i|s_t) + \frac{1}{|\mathcal{E}|}\sum_{\{i,j\} \in \mathcal{E}} f^{ij}(a^i, a^j|s_t)\,.$$

Deep Coordination Graphs

- The q^{CG} can hard to operate the arg max.
- DCG adopt the loopy belief propagation to approximate the arg max.
- Message Passing for μ for a certain number of iterations,
- then do the approximated arg max:

$$\mu_t^{ij}(a^j) \leftarrow \max_{a^i} \left\{ \frac{1}{|\mathcal{V}|} f^i(a^i|s_t) + \frac{1}{|\mathcal{E}|} f^{ij}(a^i, a^j|s_t) + \sum_{\{k,i\}\in\mathcal{E}} \mu_t^{ki}(a^i) - \mu_t^{ji}(a^i) \right\}. \quad (3)$$

$$a_*^i := \arg\max_{a^i} \left\{ \frac{1}{|\mathcal{V}|} f^i(a^i|s_t) + \sum_{\{k,i\}\in\mathcal{E}} \mu_t^{ki}(a^i) \right\}. \quad (4)$$

Deep Coordination Graphs

In order to scale to large state action space, DCG propose some principles, and we describe some of them as below:

- Restricting the payoffs $f^{ij}\left(a^i, a^j \mid \tau_t^i, \tau_t^j\right)$ to local information of agents i and j only: $f_\theta^i\left(u^i \mid \tau_t^i\right) \approx f_\theta^v\left(u^i \mid \boldsymbol{h}_t^i\right)$ and $\mathrm{RNN}\boldsymbol{h}_t^i := h_\psi\left(\cdot \mid \boldsymbol{h}_{t-1}^i, o_t^i, a_{t-1}^i\right)$
- Sharing parameters between all payoff and utility functions through a common recurrent neural network; $f^{ij}\left(a^i, a^j \mid \tau_t^i, \tau_t^j\right) \approx f_\phi^e\left(a^i, a^j \mid \boldsymbol{h}_t^i, \boldsymbol{h}_t^j\right)$

Deep Coordination Graphs

The above discussions construct the DCG, but how to construct the coordination graph?

DCG	$\mathcal{E} := \left\{ \{i, j\} \middle\vert 1 \leq i < n, i < j \leq n \right\}$
CYCLE	$\mathcal{E} := \left\{ \{i, (i \bmod n) + 1\} \middle\vert 1 \leq i \leq n \right\}$
LINE	$\mathcal{E} := \left\{ \{i, i+1\} \middle\vert 1 \leq i < n \right\}$
STAR	$\mathcal{E} := \left\{ \{1, i\} \middle\vert 2 \leq i \leq n \right\}$
VDN	$\mathcal{E} := \varnothing$

Figure 6: Tested graph topologies for DCG. But for the Networked Systems, incorporate the prior knowledge about graph would be helpful.

Table of Contents

Discussions

- IGM based methods focus on the efficient execution. They often introduce some limitation to the network to ensure the fully decetralized execution can be achieved.
- Out of IGM focus on the efficient learning. They enable communication/belief propogation during execution.