



(51) International Patent Classification:

G06N 20/00 (2019.01)

(21) International Application Number:

PCT/US2021/033107

(22) International Filing Date:

19 May 2021 (19.05.2021)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

63/029,754 26 May 2020 (26.05.2020) US

63/033,304 02 June 2020 (02.06.2020) US

17/323,475 18 May 2021 (18.05.2021) US

(71) Applicant: NEC LABORATORIES AMERICA, INC.

[US/US]; 4 Independence Way, Suite 200, Princeton, NJ 08540 (US).

(72) Inventors: YU, Wenchao; 2110 Fox Run Drive, Plains-

boro, NJ 08536 (US). CHEN, Haifeng; 11 Blackhawk Court, West Windsor, NJ 08550 (US). CHENG, Wei; 5 Arnold Drive, Princeton Junction, NJ 08550 (US).

(74) Agent: BITETTO, James, J.; Tutunjian & Bitetto, P.C.,

401 Broadhollow Road, Suite 402, Melville, NY 11747 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH,

GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) Title: INTERPRETABLE IMITATION LEARNING VIA PROTOTYPICAL OPTION DISCOVERY

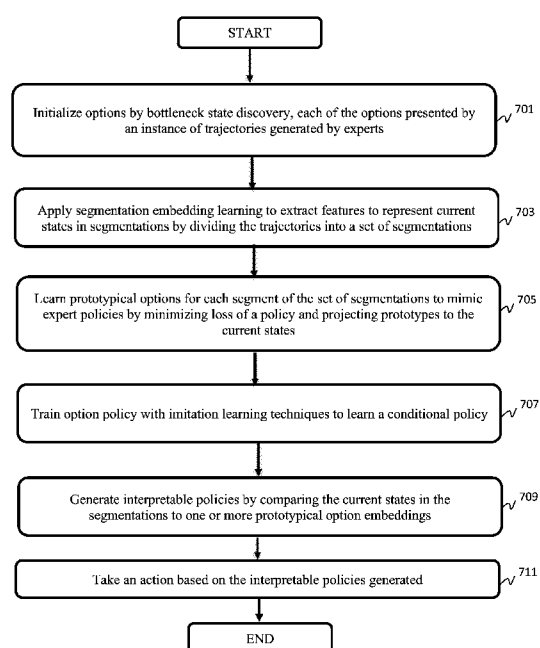


FIG. 7

(57) Abstract: A method for learning prototypical options for interpretable imitation learning is presented. The method includes initializing (701) options by bottleneck state discovery, each of the options presented by an instance of trajectories generated by experts, applying (703) segmentation embedding learning to extract features to represent current states in segmentations by dividing the trajectories into a set of segmentations, learning (705) prototypical options for each segment of the set of segmentations to mimic expert policies by minimizing loss of a policy and projecting prototypes to the current states, training (707) option policy with imitation learning techniques to learn a conditional policy, generating (709) interpretable policies by comparing the current states in the segmentations to one or more prototypical option embeddings, and taking (711) an action based on the interpretable policies generated.

Published:

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

INTERPRETABLE IMITATION LEARNING VIA PROTOTYPICAL OPTION DISCOVERY

RELATED APPLICATION INFORMATION

[0001] This application claims priority to Provisional Application No. 63/029,754, filed on May 26, 2020, Provisional Application No. 63/033,304, filed on June 2, 2020, and U.S. Patent Application No. 17/323,475, filed on May 18, 2021, each incorporated herein by reference herein in their entirety.

BACKGROUND

Technical Field

[0002] The present invention relates to imitation learning and, more particularly, to methods and systems related to interpretable imitation learning via prototypical option discovery.

Description of the Related Art

[0003] Humans have the ability to compose options or skills to solve a complex problem. For example, to treat a COVID-19 patient with a critical condition, an intensive care unit (ICU) doctor needs to compose essential skills such as endotracheal intubation, chest-tube placement, and arterial and central venous catheterization. Discovering the compositional structures from experts' trajectories is beneficial to understand the experts' policy as well as learn a new policy.

SUMMARY

[0004] A method for learning prototypical options for interpretable imitation learning is presented. The method includes initializing options by bottleneck state discovery, each of the options presented by an instance of trajectories generated by experts, applying segmentation embedding learning to extract features to represent current states in segmentations by dividing

the trajectories into a set of segmentations, learning prototypical options for each segment of the set of segmentations to mimic expert policies by minimizing loss of a policy and projecting prototypes to the current states, training option policy with imitation learning techniques to learn a conditional policy, generating interpretable policies by comparing the current states in the segmentations to one or more prototypical option embeddings, and taking an action based on the interpretable policies generated.

[0005] A non-transitory computer-readable storage medium comprising a computer-readable program for learning prototypical options for interpretable imitation learning is presented. The computer-readable program when executed on a computer causes the computer to perform the steps of initializing options by bottleneck state discovery, each of the options presented by an instance of trajectories generated by experts, applying segmentation embedding learning to extract features to represent current states in segmentations by dividing the trajectories into a set of segmentations, learning prototypical options for each segment of the set of segmentations to mimic expert policies by minimizing loss of a policy and projecting prototypes to the current states, training option policy with imitation learning techniques to learn a conditional policy, generating interpretable policies by comparing the current states in the segmentations to one or more prototypical option embeddings, and taking an action based on the interpretable policies generated.

[0006] A method for learning prototypical options for interpretable imitation learning is presented. The method includes dividing a task, by a processor, into a plurality of sub-tasks via a learning policy over options, learning, by the processor, different options to solve each of the plurality of sub-tasks by mimicking expert policy, and fine-tuning the learning policy to learn to take an action based on the task.

[0007] These and other features and advantages will become apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

[0008] The disclosure will provide details in the following description of preferred embodiments with reference to the following figures wherein:

[0009] FIG. 1 is a block/flow diagram of an exemplary option selection mechanism, in accordance with embodiments of the present invention;

[00010] FIG. 2 is a block/flow diagram of an exemplary prototypical option discovery for interpretable imitation learning (IPOD) architecture, in accordance with embodiments of the present invention;

[00011] FIG. 3 is a block/flow diagram of an exemplary method for employing the IPOD architecture of FIG. 2, in accordance with embodiments of the present invention;

[00012] FIG. 4 is a block/flow diagram of an exemplary method for employing the option initialization, segmentation embedding learning, prototypical option learning, and option policy learning components of FIG. 3, in accordance with embodiments of the present invention;

[00013] FIG. 5 is a block/flow diagram of a practical application of the IPOD architecture, in accordance with embodiments of the present invention;

[00014] FIG. 6 is an exemplary processing system for the IPOD architecture, in accordance with embodiments of the present invention;

[00015] FIG. 7 is a block/flow diagram of an exemplary method for executing the IPOD architecture, in accordance with embodiments of the present invention; and

[00016] FIG. 8 illustrates exemplary equations for implementing the IPOD architecture, in accordance with embodiments of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[00017] Imitation learning which mimics experts' behaviors is beneficial to finding meaningful structure or skills in the experts' demonstrations. Despite the superior performance of imitation learning models, they are usually considered as "black-boxes" which lack transparency, limiting their application in many decision-making scenarios, e.g., healthcare and finance. A variety of methods learn a hidden variable of the variation underlying expert demonstrations to construct the structure of expert policy and visualize the changes in the hidden variable. However, post-hoc explanations do not explain the reasoning process of how the model makes its decisions and can be incomplete or inaccurate in capturing the reasoning process of the original model. Therefore, it is often desirable to have models with built-in interpretability.

[00018] The exemplary embodiments address such issues by defining a form of interpretability in imitation learning that imitates human abstraction and explains its reasoning in a human-understanding manner. The exemplary methods employ prototype learning to discover options for built-in interpretable imitation learning. Prototype learning, which drives from the study of human reasoning, is a form of case-based reasoning, which makes decisions by comparing new inputs with a few data instances (prototypes) in, e.g., image recognition, sequence classification, sequence segmentation, etc.

[00019] The exemplary methods discover prototypical options for interpretable imitation learning. The exemplary methods introduce a network architecture referred to as prototypical option discovery (IPOD). Each prototypical option is responsible for explaining a group of variable-length segments of the demonstration trajectories. To learn the prototypical options,

IPOD first learns a policy to break the trajectories into a set of segmentations, which results in K groups of segments for the K prototypical options. IPOD uses LSTM with a soft-attention mechanism to derive segment embedding. For each group of segments, the exemplary methods learn a prototypical contextual policy to take action with states as well as the option embedding, which is determined based on centroids of the segment embedding, as inputs. In this way, the model is interpretable, in the sense that it has a transparent reasoning process when making decisions. For better interpretability, the exemplary methods define several criteria for constructing the prototypes, including option diversity and prediction accuracy.

[00020] The exemplary embodiments introduce an imitation learning framework that learns interpretable policy via prototypical options which include segmentation prototypes. The exemplary embodiments enable learning the prototypical option embedding by weighted segmentation for sparsity and learn the prototypical option's policy by driving the option-relevant information via option embedding. The goal is to learn a new policy π , which imitates the expert behavior by maximizing the likelihood of given demonstration trajectories. Thus, the behavior of an expert agent can be copied to accomplish a desired task.

[00021] Imitation learning refers to learning a policy that mimics the behavior of experts who demonstrate how to perform the given task. The behavior of the expert demonstrator is represented by trajectories $\tau = [s_0, a_0, \dots, s_T, a_T]$, which is a sequence of state action pairs. Imitation learning has various approaches. One approach is behavior cloning (BC), which directly maps from the state to the action. This method usually learns a policy through standard supervised learning. BC does not perform any additional policy interactions with the learning environment, but it suffers from distributional drift. Another approach is inverse reinforcement learning (IRL), which learns a policy by recovering the reward function from demonstrations and with dense reward signals provided from the learned reward function. However, the learned policy is valid only while the learned reward function is valid. Yet another approach is

adversarial imitation learning (AIL), which constrains the behavior of the agent to be approximately optimal with an unknown reward function without explicitly attempting to recover that reward function. However, both AIL and IRL require interacting with the environment for generating the agent's trajectory for comparison with the expert's trajectory. Recently, imitation learning with neural networks efficiently learns a desired behavior in complex environments. However, these methods are usually considered as "black-boxes," which lack transparency. The exemplary methods introduce an interpretable imitation learning framework for more applications of imitation learning, e.g., healthcare, finance, etc.

[00022] An option is a generalization of an action (also known as a skill, sub-policy or a sub-goal). Formally, an option is a three-tuple that includes the start, end probability of an option and the policy of the option. Options offer great potential for mitigating the difficulty of solving complex Markov decision processes (MDPs) via temporally extended actions.

[00023] Interpretable modeling mainly falls into two categories, that is, intrinsic explanation which makes the model transparent by restricting the complexity, e.g., decision tree or case-based (prototype-based) model, and post-hoc explanation, which is achieved by analyzing the model after training, e.g., extracting the importance of states via attention and distilling a black-box policy into a simple structure policy. A set of post-hoc imitation learning was proposed for generating meaningful policy. However, the intrinsic explanation model is sometimes desirable since post-hoc explanations usually do not fit the original model precisely. Prototype learning, which draws conclusions for new inputs by comparing them with a few exemplary cases (e.g., prototypes) belongs to the intrinsic explanation method.

[00024] The options framework models skills as options, which is a closed-loop policy to solve the sub-tasks. For example, picking up an object, jumping, etc. are options, which require a user to take actions over a period of time. An option o includes the following components, that is, its initiation condition, $I_o(s)$, which determines whether o can be executed in state s ,

its termination condition, $\beta_o(s)$, which determines whether option execution must terminate in state s and its closed-loop control policy, $\pi_o(s)$, which maps state s to a low-level action a .

[00025] Prototype theory emerged in 1971 with the work of psychologist Eleanor Rosch, and it has been described as a “Copernican revolution” in the theory of categorization. In prototype theory, any given concept in any given language has a real-world example that best represents this concept. For instance, when asked to give an example of the concept of fruits, an apple is more frequently cited than, a durian. This theory claims that the presumed natural prototypes were central tendencies of the categories. Prototype theory has also been applied in machine learning, where a prototype is defined as a data instance that is representative of all the data. There are many approaches to find prototypes in the data. Any clustering algorithm that returns actual data points as cluster centers would qualify for selecting prototypes.

[00026] The exemplary embodiments introduce the formulation of the prototypical option, which is a kind of option that can be presented by an instance of the trajectories generated by the experts. A prototypical option o includes four components $\langle I_o, \pi_o, \beta_o, g_o \rangle$, that is, an intra-option policy $\pi_o : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, a termination condition $\beta_o : \mathcal{S} \rightarrow [0, 1]$, an initiation state set $I_o \in \mathcal{S}$ and an option prototype g_o .

[00027] Specifically, g_o is defined by sub-trajectories generated by the experts. Given the trajectories of the expert $\tau = \{s_1, a_1, \dots, s_T, a_T\}$, the prototypical option is a set of segments $(g_1, g_2, \dots, g_K)$, where $g_k = s_{v_{m'} : v_m, m'} = m - 1$. Here, $v_m \in [1, T]$ are segment boundary indicator variables with $v_0 = 0$, $v_M = T$, $v_m \geq v_{m'}$, e.g., $g_o = s_{2:4}$, so that $g_o = [s_2, s_3, s_4]$.

[00028] A prototypical option $\langle I_o, \pi_o, \beta_o, g_o \rangle$ is available in state s_t if and only if $s_t \in I_o$. If the option is taken, then actions are selected according to π_o until the option terminates according to β_o . In a prototypical option, g_o is considered as a real-world example to explain the option.

[00029] Options discovery is based on the intuition that it would be easier to solve the long-horizon task from temporal abstraction, e.g., separate or divide the long-horizon task into a set of sub-tasks, and select different options to solve for each sub-task. This intuition informs the steps of the algorithm, that is, breaking or dividing the trajectories into a set of subtasks via learning a policy π_h over options, learning (or discovering) options that could solve these sub-tasks by mimicking the expert' policy, and, once such options are learned, the exemplary embodiments fine-tune π_h to learn to take an option based on the current task.

[00030] Formally, given the trajectories of the expert $\tau = \{s_1, a_1, \dots, s_T, a_T\}$, the goal is to first break or divide trajectories τ into M disjoint segments $(g_1, g_2, \dots, g_M)$, where $g_m = s_{v_{m'}:v_m}, m'$, $s_{v_{m'}:v_m}, m' = (s_{v_{m'}+1}, \dots, s_{v_m})$, $m' = m - 1$. Here, $v_m \in [1, T]$ are segment boundary indicator variables with $v_0 = 0$, $v_M = T$, $v_m \geq v_{m'}$. The segments are grouped into K clusters and learn each cluster's prototypical options, where $G_k = \{g_m\}_{m \in \{0,1,\dots,M\}}$ indicate the m -th group segments.

[00031] The exemplary embodiments leverage prototype learning to introduce an interpretable imitation learning framework by prototypical option discovery, where each prototypical option is responsible for explaining a group of variable-length segments of the demonstration trajectory. As presented in FIG. 2, I2L 200 addresses interpretable imitation learning tasks with steps to learn prototypical options $\langle I_o, \pi_o, \beta_o, g_o \rangle$. To learn the initial state set I_o and the termination condition β_o , the exemplary methods learn a policy $\pi_h(o|s)$ over options to break or divide the trajectories into a set of segmentations, which results in K groups of segments for the K prototypical options. To learn the option prototype g_o , the exemplary methods map each segment into an option embedding $\hat{o}_{v_{m'}:v_m}$ and cluster them to find K central nodes as option prototypes g_o , $o = \{1, \dots, K\}$. As for learning intra-option policy π_o , the

exemplary methods learn a prototypical contextual policy $\pi(o|s, o)$ to take action based on states, as well as the option embedding.

[00032] In options learning (I_o and β_o) step, $\pi_h(ols)$ first constructs a set of admissible options given by: $\mathcal{O}(s_t) = \{o_i | I_{o_i}(s_t) = 1 \cap \beta_{o_i}(s_t) = 0, \forall o_i \in \mathcal{O}\}$. Here the $\mathcal{O}(s_t)$ is updated according to the $\pi_h(o|s)$. IPOD 200, in FIG. 2, determines the $I_{o_i}(s_t)$ and $\beta_{o_i}(s_t)$ by the output of π_h , e.g., o_t , where if $o_t = 1$, $I_{o_i}(s_t) = 1$ and $\beta_{o_i}(s_t) = 0$, otherwise $I_{o_i}(s_t) = 0$ and $\beta_{o_i}(s_t) = 1$. An example of how the agent $\pi_h(o|s)$ selects an option is shown in structure 100 of FIG. 1.

[00033] With regards to learning the policy over options, $\pi_h(ols)$ is learned by choosing the admissible prototypical option. Since the exemplary methods utilize imitation learning to learn the intra-option policy, the reward of $\pi_h(ols)$ is obtained by the selected option π_o which takes primitive actions and receives the reward signal. Thus, the reward of the option is the cumulative reward of the actions taken from a current time to the termination of the option:

$$r_{t:t+\delta} = r_t + \dots + r_{\{t+\delta\}},$$

[00034] where $\delta \in [0, T]$ is the time interval of the option $t + \delta$ is the termination of the option o_t .

[00035] Given the transition $(s_t, o_t, r_{t:t+\delta})$ we update $\pi_h(o|s)$ taking option o_t at state s_t according to policy gradient:

$$\nabla J = \mathbb{E}_s \sim \pi_h[Q(s, \pi_h(o_t|s_t))]$$

[00036] where the option-value $Q(s_t, o_t)$ refers to the expected rewards for an action o_t taken in a given state s_t . Updating options to the policy over options, the above equations show how the exemplary methods can learn the policy π_h over option and use it for selecting options. However, before learning π_h , the exemplary methods must assign appropriate initial parameters to π_h . The exemplary methods segment the trajectories by detecting the bottleneck states within the trajectories. Bottlenecks have been defined as those states which appear frequently on

successful trajectories to a goal but not on unsuccessful ones or as nodes which allow for densely connected regions of the interaction graph to reach other such regions. Informally, bottleneck areas have been described as the border states of densely connected areas in the state space or as states that allow transitions to a different part of the environment. A more formal definition defines bottleneck areas as those states which are local maxima of betweenness, a measure of centrality on graphs, on a transition graph.

[00037] The exemplary methods extract all the states in the trajectories, and use density-based spatial clustering methods (e.g., DBSCAN) to automatically cluster the states into K groups. In the exemplary methods, each state group indicates one option's valid states (where $I_o(s) = 1$). That is, the initial π_h will take that option while it is in these states via behavior cloning.

[00038] In option prototype learning, the exemplary methods aim to learn the option prototype, which is a sub-trajectory or segment generated by the experts. Each option prototype is responsible for explaining a group of variable-length segments of the demonstration trajectory $\mathcal{g}_m$ generated by π_h . Thus, the exemplary methods first initialize K option prototype embedding $\mathbf{o}_k \in \mathbb{R}^n$, $k = \{1, 2, 3, \dots, K\}$ vectors as learnable parameters. Next, the exemplary methods map each group of segments $\mathcal{g}_{m,k}$ individually into a low dimension embedding $\mathbf{g}_{m,k}$ by classifying the segment into the corresponding option's category k . Meanwhile, the exemplary methods learn $\mathbf{o}_k$ by minimizing the distance between $\mathbf{o}_k$ and $\mathbf{g}_{m,k}$. Finally, the exemplary methods consider the segment which has the smallest distance with $\mathbf{o}_k$ as the option prototype of $\mathbf{o}_k$.

[00039] Regarding segmentation embedding learning, the exemplary methods aim to learn a meaningful latent space to represent the segments, where they are clustered (in L2-distance) around semantically similar prototypical options, and the clusters from different classes are well-separated.

[00040] To achieve this, the exemplary methods use a long short-term memory (LSTM) to learn the segment's representation $g_{v_{m':v_m}} = f_\phi(s_{v_{m':v_m}})$ and the embeddings of prototypical option o_k , where $s_{v_{m':v_m}}$, $v_m = t$ indicates the current segment generated by π_h . To force the segment $s_{v_{m':v_m}}$ and the option prototypes to be in the same space, the exemplary methods minimize the distance between $g_{v_{m':v_m}}$ and its closest prototype o_k .

[00041] The optimization problem the exemplary methods aim to solve is:

$$[00042] \quad \mathcal{L}_{emb} = \sum_{m=1}^M \min_{k=1}^K \left\| f_\phi(s_{\{v_{m':v_m}\}}) - o_k \right\|_2^2,$$

[00043] The minimization of $\mathcal{L}_{emb}$ encourages each training segment to have some latent patch that is close to at least one prototypical option. These terms shape the latent space into a semantically meaningful clustering structure.

[00044] Regarding option prototype embedding learning (g_o), since the option prototype embeddings $o_{k=1}^K$ are representations in the latent space, they are not readily interpretable. For interpretability, the exemplary methods assign each prototypical option embedding $o_{k=1}^K$ with their closest segment embedding g in the training set.

[00045] As for learning option prototype embedding, the exemplary methods leverage both supervised learning and imitation learning regarding the effectiveness and interpretability. The exemplary methods attempt to minimize the least square loss between g and o_k , and prevent the learning of multiple similar prototypical options. The exemplary methods use a diversity regularization term that penalizes prototypical options that are close to each other. Meanwhile, the exemplary methods also consider the downstream task (e.g., imitation learning).

[00046] The full objective function of option learning is given as follows:

$$\mathcal{L}_{option} = -\lambda_1 * \mathcal{L}_{IL_{loss}} + \lambda_2 * \sum_{i=1}^K \min_{m=1}^M \|f_{\phi}(s_{v_m':v_m}) - \mathbf{e}_i\|_2^2 + \lambda_3 * \sum_{i=1}^K \sum_{j=i+1}^K \max(0, d_{min} - \|\mathbf{e}_i - \mathbf{e}_j\|)$$

[00047] where the first term is for effectiveness, where an imitation learning objective function is conducted to learn the segment embeddings and option prototype embeddings to mimic expert's policy π_E . $\mathcal{L}_{IM_{loss}}$ (reproduced below) can be any imitation learning method, e.g., a behavior cloning loss or an adversarial imitation learning objective. The second term is for interpretability where an evidence regularization is used to encourage each prototypical option embedding to be as close to an encoded instance as possible. The third term is a diversity regularization term to learn diversified options, where d_{min} is a threshold that classifies or determines whether two prototypes are close or not. d_{min} is set to 1.0 in exemplary embodiments. $\lambda_1, \lambda_2, \lambda_3 \in [0, 1]$ are the weights used to balance the three loss terms.

[00048] Regarding option policy learning π_o , each option o maintains its own policy $\pi_o: \mathcal{S} \rightarrow \mathcal{A}_t$, which is parameterized by its own parameters θ_o . To reduce the parameter complexity, the exemplary methods propose a contextual policy $\pi_{\theta}(a_t | s_t, o_k)$ to learn a conditional policy which is conditioned on both the state and the option, which is shared among all the options.

[00049] The exemplary methods train the option policy $\pi_{\theta}(a_t | s_t, o_k)$ via the traditional imitation learning algorithms defined as $\mathcal{L}_{IL_{loss}}$, e.g., behavior cloning and adversarial imitation learning. The goal of adversarial imitation learning is to minimize the JS divergence between trajectory distribution generated by the expert's policy and the option's policy.

[00050] Note that the exemplary methods use the same policy loss for both option prototypes and option policy, but the exemplary methods only optimize the parameters of option prototypes or option policy for each optimization step.

[00051] Regarding the full objective function, the loss minimized is:

$$\mathcal{L}_{Full} = w_1 \cdot \mathcal{L}_{option} + w_2 \cdot \mathcal{L}_{IL_{loss}} + w_3 \cdot \mathcal{L}_{emb}$$

[00052] where $w_1, w_2, w_3 \in [0, 1]$ are hyper-parameters to balance the weights of the three kinds of loss. As for optimization, the exemplary methods first initialize K groups segments followed by iteratively optimizing $\mathcal{L}_{option} + \mathcal{L}_{IL_{loss}} + \mathcal{L}_{emb}$.

[00053] Therefore, the exemplary embodiments introduce an interpretable imitation learning framework by discovering compositional structure which is called prototypical option discovery imitation learning (IPOD). IPOD constructs prototypical options which embed the skills of experts by an option embedding and an option policy via a prototype learning framework. IPOD generates interpretable agent policies by comparing the state segmentations to a few prototypical option embeddings followed by taking an action based on the option embedding. Unlike seeking a minimal subset of samples as prototypes that can serve as a distillation or condensed view of a data set, the exemplary model of the present invention uses a soft attention mechanism to derive prototypical option embedding from trajectory fragments. The exemplary methods also use the soft attention mechanism to create a bottleneck in the agent, forcing it to focus on option-relevant information.

[00054] FIG. 3 is a block/flow diagram of an exemplary method 300 for employing the IPOD architecture of FIG. 2, in accordance with embodiments of the present invention.

[00055] Prototypical option discovery for interpretable imitation learning (IPOD) proposes to learn prototypical options for interpretable imitation. Each prototypical option is responsible for explaining a group of variable-length segments of the demonstration trajectory. The exemplary methods model each group of segments by computing distances to prototypical option embedding, where prototypical option embedding is a latent variable summarizing the segments. The IPOD model includes the following learning phases.

[00056] At block 303, option initialization takes place:

[00057] The IPOD first initializes the options by bottleneck state discovery methodology. Inspired by previous works on bottleneck state discovery, e.g., frequently visited states, the exemplary methods identify states that connect different densely connected regions in the state space. In order to discover such bottleneck states from expert demonstrations, the exemplary methods use the behavior cloning method with soft attention mechanism to obtain important states with large attention weights. The important states can then be found with DBSCAN clustering. The dense clusters derived from DBSCAN are used for option initialization.

[00058] At block 305, the policy over options learning takes place:

[00059] A prototypical option o includes four components $\langle I_o, \pi_o, \beta_o, g_o \rangle$, an intra-option policy $\pi_o: \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$, a termination condition $\beta_o: \mathcal{S} \rightarrow [0,1]$, an initiation state set $I_o \in \mathcal{S}$, and its option prototype g_o . To select an option in state s_t , $\pi_h(o|s)$ first constructs a set of admissible options given by:

$$\mathcal{O}(s_t) = \{o_i | I_{o_i}(s_t) = 1 \cap \beta_{o_i}(s_t) = 0, \forall o_i \in \mathcal{O}\}$$

[00060] Here the $\mathcal{O}(s_t)$ is updated according to the $\pi_h(o|s)$. IPOD determines the $I_{o_i}(s_t)$ and $\beta_{o_i}(s_t)$ by the output of π_h , i.e., o_t , where if $o_t = 1$, $I_{o_i}(s_t) = 1$ and $\beta_{o_i}(s_t) = 0$. An example of how the agent $\pi_h(o|s)$ selects an option is shown above with respect to $\mathcal{O}(s_t)$.

[00061] $\pi_h(o|s)$ is learned to choose the admissible prototypical option. Since the exemplary methods utilize imitation learning to learn the intra-option policy, the reward of $\pi_h(o|s)$ is obtained by the selected option π_o which takes primitive actions and receives the reward signal. Thus, the reward of the option is the cumulative reward of the actions taken from a current time to the termination of the option: $r_{\{t:t+\delta\}} = r_t + \dots + r_{t+\delta}$, where $\delta \in [0, T]$ is the time interval of the option on-going, and $t + \delta$ is the termination of the option o_t .

[00062] Given the transition $(s_t, o_t, r_{t:t+\delta})$, the exemplary methods update $\pi_h(o|s)$ taking option o_t at state s_t according to policy gradient:

[00063] $\nabla J = \mathbb{E}_s \sim \pi_h[Q(s, \pi_h(o_t|s_t))]$

[00064] where the option-value $Q(s_t, o_t)$ refers to the expected rewards for an action o_t taken in a given state s_t .

[00065] At block 307, prototypical option learning takes place:

[00066] In the second stage, the exemplary methods aim to learn the option prototype, which is a sub-trajectory or segment generated by the experts. Each option prototype is responsible for explaining a group of variable-length segments of the demonstration trajectory g_m generated by π_h . Thus, the exemplary methods first initialize K option prototype embedding $\mathbf{o}_k \in \mathbb{R}^n$, $k=\{1,2,3,...,K\}$ vectors as learnable parameters. Next, the exemplary methods map each group of segment $g_{m,k}$ individually into a low-dimension embedding $\mathbf{g}_{m,k}$ by classifying the segment into the corresponding option's category k. Meanwhile, the exemplary methods learn $\mathbf{o}_k$ by minimizing the distance between $\mathbf{o}_k$ and $\mathbf{g}_{m,k}$. Finally, the exemplary methods consider the segment which has the smallest distance with $\mathbf{o}_k$ as the option prototype of $\mathbf{o}_k$.

[00067] Regarding segmentation embedding learning, the exemplary methods aim to learn a meaningful latent space to represent the segments, where they are clustered (in L2-distance) around semantically similar prototypical options, and the clusters from different classes are well-separated.

[00068] To achieve this goal, the exemplary methods use an LSTM to learn the segment's representation $g_{v_{m'}:v_m} = f_\phi(s_{v_{m'}:v_m})$ and the embeddings of prototypical option $\mathbf{o}_k$, where $s_{v_{m'}:v_m}$, $v_m = t$ indicates the current segment generated by π_h . To force the segment $s_{v_{m'}:v_m}$ and the option prototypes to be in the same space, the exemplary methods minimize the distance between $g_{v_{m'}:v_m}$ and its closest prototype $\mathbf{o}_k$. The optimization problem to be solved is:

$$\mathcal{L}_{emb} = \sum_{m=1}^M \min_{k=1}^K \left\| f_\phi(s_{\{v_{m'}:v_m\}}) - \mathbf{o}_k \right\|_2^2,$$

[00069] The minimization of $\mathcal{L}_{emb}$ encourages each training segment to have some latent patch that is close to at least one prototypical option. These terms shape the latent space into a semantically meaningful clustering structure.

[00070] At block 309, prototypical option embedding learning takes place:

[00071] Since the option prototype embeddings $\mathbf{o}_{k=1}^K$ are representations in the latent space, they are not readily interpretable. For interpretable, the exemplary methods propose to assign each prototypical option embedding $\mathbf{o}_{k=1}^K$ with their closest segment embedding $\mathbf{g}$ in the training set.

[00072] As for learning option prototype embedding, the exemplary methods leverage both supervised learning and imitation learning regarding effectiveness and interpretability. The exemplary methods try to minimize the least square loss between $\mathbf{g}$ and $\mathbf{o}_k$ to prevent learning multiple similar prototypical options. The exemplary methods use a diversity regularization term that penalizes prototypical options that are close to each other. Meanwhile, the exemplary methods also consider the downstream task (imitation learning).

[00073] The full objective function is:

$$\mathcal{L}_{option} = -\lambda_1 * \mathcal{L}_{ILloss} + \lambda_2 * \sum_{i=1}^K \min_{m=1}^M \|f_{\phi}(s_{v_{in}:v_m}) - \mathbf{e}_i\|_2^2 + \lambda_3 * \sum_{i=1}^K \sum_{j=i+1}^K \max(0, d_{min} - \|\mathbf{e}_i - \mathbf{e}_j\|)$$

where the first term is for effectiveness and where an imitation learning objective function is conducted to learn the segment embeddings and option prototype embeddings to mimic expert's policy π_E . The second term is for interpretability where an evidence regularization is used to encourage each prototypical option embedding to be as close to an encoded instance as possible. The third term is a diversity regularization to learn diversified options and d_{min} is a threshold that classifies whether two prototypes are close or not. The exemplary methods set d_{min} to 1.0. $\lambda_1, \lambda_2, \lambda_3 \in [0,1]$ are the weights used to balance the three loss terms.

[00074] At block 311, option policy learning takes place:

[00075] Each option o maintains its own policy $\pi_o: s \rightarrow a_t$, which is parameterized by its own parameters θ_o . To reduce the parameter complexity, the exemplary methods propose a contextual policy $\pi_\theta(a_t|s_t, o_k)$ to learn a conditional policy which is conditioned on both the state and the option, which shares among all the options.

[00076] The exemplary methods train the option policy $\pi_\theta(a_t|s_t, o_k)$ by traditional imitation learning algorithms, e.g., behavior cloning and adversarial imitation learning. The goal of behavior cloning is to mimic the action of the expert at each time step via supervised learning technical. The goal of adversarial imitation learning is to minimize the JS divergence between trajectory distribution generated by the expert's policy and the option's policy.

[00077] Note that the same policy loss is used for both option prototypes and option policy, but the exemplary methods only optimize the parameters of option prototypes or option policy for each optimization step. The exemplary methods can further train the option policy with imitation learning algorithms, e.g., behavior cloning and adversarial imitation learning. The goal of option policy learning is to mimic the segmentations of demonstrations from the experts.

[00078] FIG. 4 is a block/flow diagram of an exemplary method for employing the option initialization, segmentation embedding learning, prototypical option learning, and option policy learning components of FIG. 3, in accordance with embodiments of the present invention.

[00079] Imitation learning with neural networks efficiently learns a desired behavior in complex environments. However, these methods are usually considered as “black-boxes” which lack transparency, limiting their application in many decision-making scenarios. A variety of methods learn a hidden variable of the variation underlying expert demonstrations to construct the structure of expert policy and visualize the changes in the hidden variable. However, post-hoc explanations do not explain the reasoning process of how the model makes

its decisions and can be incomplete or inaccurate in capturing the reasoning process of the original model. Therefore, it is often desirable to have models with built-in interpretability. The exemplary embodiments of the present invention define a form of interpretability in imitation learning that imitates human abstraction and explains its reasoning in a human-understanding manner. The exemplary methods enable prototype learning to discovery options for built-in interpretable imitation learning, which makes decisions by comparing the new inputs with a few data instances (prototypes).

[00080] Regarding the option initialization phase 303:

[00081] At block 401, attention mechanics and behavior cloning are utilized to extract the most important states considered while mimicking the expert's demonstration.

[00082] At block 403, for bottleneck state discovery, DBSCAN is used on the extracted states and the states are automatically clustered into groups.

[00083] Regarding policy over options learning 305:

[00084] At block 411, imitation learning is utilized to learn the intra-option policy, where the reward is calculated by the cumulative rewards from the primitive actions.

[00085] Regarding prototypical option learning 307:

[00086] At block 421, prototypical options are learned via minimizing the loss of the policy and projecting the prototypes to observed states.

[00087] Regarding prototypical option embedding learning 309:

[00088] At block 431, prototypical options are learned via minimizing the loss of the policy and projecting the prototypes to observed states.

[00089] Regarding option policy learning 311:

[00090] At block 441, the option policy is trained with imitation learning algorithms, such as behavior cloning, inverse imitation learning and adversarial imitation learning.

[00091] In summary, the exemplary methods introduce a new architecture, that is, prototypical option discovery for interpretable imitation learning (IPOD). Each prototypical option includes a set of segmentation from experts' trajectories and is embedded by an option policy. The IPOD uses a soft attention mechanism to derive prototypical option embedding from its trajectory fragments. Given a demonstration of the expert, the model matches the segmentations from the demonstration to the learned prototypical options, and makes an action based on the learned prototypical option. The exemplary methods also use the soft attention mechanism to create a bottleneck in the agent, forcing the agent to focus on option-relevant information. In this way, the model is interpretable, in the sense that it has a transparent reasoning process when making decisions. For better interpretability, the exemplary methods define several criteria for constructing the prototypes, including option diversity and accuracy.

[00092] The IPOD considers the prototype learning to discovery options for built-in interpretable imitation learning in accordance with the following as illustrated in FIG. 2. Bottleneck state discovery segments the input trajectories into disjoint segments of variable length by, e.g., density-based clustering methods. Option projection includes representation learning of the segmentations in each cluster, and prototypical option embedding learning. Option refixation takes the low-level actions controlled through the prototypical option embedding and refines each group of segments by matching the segmentation embeddings to prototypical option embeddings.

[00093] FIG. 5 is a block/flow diagram 500 of a practical application of the IPOD architecture, in accordance with embodiments of the present invention.

[00094] In one practical example, a patient 502 needs to receive medication 504. Options are computed for indicating different levels of dosages of the medication 504. The exemplary methods learn a prototypical contextual policy $\pi(a|s, o)$ to take action based on states 506. The IPOD architecture 670 is implemented to enable prototypical option visualization by executing

a reasoning process 555 and evaluating policy performance 557. I2L 670, via the reasoning process 555, can smoothly compose the different options by considering the variant states 506 of the patient 502. In one instance, I2L 670 can chose the low-dosage option for the patient 502. The results 510 (e.g., dosage options) can be provided or displayed on a user interface 512 handled by a user 514.

[00095] FIG. 6 is an exemplary processing system for GBL, in accordance with embodiments of the present invention.

[00096] The processing system includes at least one processor (CPU) 604 operatively coupled to other components via a system bus 602. A GPU 605, a cache 606, a Read Only Memory (ROM) 608, a Random Access Memory (RAM) 610, an input/output (I/O) adapter 620, a network adapter 630, a user interface adapter 640, and a display adapter 650, are operatively coupled to the system bus 602. Additionally, an interpretable imitation learning framework 670 can be employed to execute option initialization 303, policy over options learning 305, prototypical option learning 307, prototypical option embedding learning 309, and option policy learning 311.

[00097] A storage device 622 is operatively coupled to system bus 602 by the I/O adapter 620. The storage device 622 can be any of a disk storage device (e.g., a magnetic or optical disk storage device), a solid-state magnetic device, and so forth.

[00098] A transceiver 632 is operatively coupled to system bus 602 by network adapter 630.

[00099] User input devices 642 are operatively coupled to system bus 602 by user interface adapter 640. The user input devices 642 can be any of a keyboard, a mouse, a keypad, an image capture device, a motion sensing device, a microphone, a device incorporating the functionality of at least two of the preceding devices, and so forth. Of course, other types of input devices can also be used, while maintaining the spirit of the present invention. The user input devices

642 can be the same type of user input device or different types of user input devices. The user input devices 642 are used to input and output information to and from the processing system.

[000100] A display device 652 is operatively coupled to system bus 602 by display adapter 650.

[000101] Of course, the processing system may also include other elements (not shown), as readily contemplated by one of skill in the art, as well as omit certain elements. For example, various other input devices and/or output devices can be included in the system, depending upon the particular implementation of the same, as readily understood by one of ordinary skill in the art. For example, various types of wireless and/or wired input and/or output devices can be used. Moreover, additional processors, controllers, memories, and so forth, in various configurations can also be utilized as readily appreciated by one of ordinary skill in the art. These and other variations of the processing system are readily contemplated by one of ordinary skill in the art given the teachings of the present invention provided herein.

[000102] FIG. 7 is a block/flow diagram of an exemplary method for executing the IPOD architecture, in accordance with embodiments of the present invention.

[000103] At block 701, initialize options by bottleneck state discovery, each of the options presented by an instance of trajectories generated by experts.

[000104] At block 703, apply segmentation embedding learning to extract features to represent current states in segmentations by dividing the trajectories into a set of segmentations.

[000105] At block 705, learn prototypical options for each segment of the set of segmentations to mimic expert policies by minimizing loss of a policy and projecting prototypes to the current states.

[000106] At block 707, train option policy with imitation learning techniques to learn a conditional policy.

[000107] At block 709, generate interpretable policies by comparing the current states in the segmentations to one or more prototypical option embeddings.

[000108] At block 711, take an action based on the interpretable policies generated.

[000109] FIG. 8 illustrates exemplary equations 800 for implementing the IPOD architecture, in accordance with embodiments of the present invention.

[000110] The equations include a loss function for segmentation embedding learning, an objective function, and policy losses.

[000111] As used herein, the terms “data,” “content,” “information” and similar terms can be used interchangeably to refer to data capable of being captured, transmitted, received, displayed and/or stored in accordance with various example embodiments. Thus, use of any such terms should not be taken to limit the spirit and scope of the disclosure. Further, where a computing device is described herein to receive data from another computing device, the data can be received directly from the another computing device or can be received indirectly via one or more intermediary computing devices, such as, for example, one or more servers, relays, routers, network access points, base stations, and/or the like. Similarly, where a computing device is described herein to send data to another computing device, the data can be sent directly to the another computing device or can be sent indirectly via one or more intermediary computing devices, such as, for example, one or more servers, relays, routers, network access points, base stations, and/or the like.

[000112] As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a “circuit,” “module,” “calculator,” “device,” or “system.” Furthermore, aspects of the present

invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

[000113] Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical data storage device, a magnetic data storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can include, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

[000114] A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

[000115] Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

[000116] Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the “C” programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[000117] Aspects of the present invention are described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the present invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks or modules.

[000118] These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer

readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks or modules.

[000119] The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks or modules.

[000120] It is to be appreciated that the term “processor” as used herein is intended to include any processing device, such as, for example, one that includes a CPU (central processing unit) and/or other processing circuitry. It is also to be understood that the term “processor” may refer to more than one processing device and that various elements associated with a processing device may be shared by other processing devices.

[000121] The term “memory” as used herein is intended to include memory associated with a processor or CPU, such as, for example, RAM, ROM, a fixed memory device (e.g., hard drive), a removable memory device (e.g., diskette), flash memory, etc. Such memory may be considered a computer readable storage medium.

[000122] In addition, the phrase “input/output devices” or “I/O devices” as used herein is intended to include, for example, one or more input devices (e.g., keyboard, mouse, scanner, etc.) for entering data to the processing unit, and/or one or more output devices (e.g., speaker, display, printer, etc.) for presenting results associated with the processing unit.

[000123] The foregoing is to be understood as being in every respect illustrative and exemplary, but not restrictive, and the scope of the invention disclosed herein is not to be determined from the Detailed Description, but rather from the claims as interpreted according to the full breadth permitted by the patent laws. It is to be understood that the embodiments

shown and described herein are only illustrative of the principles of the present invention and that those skilled in the art may implement various modifications without departing from the scope and spirit of the invention. Those skilled in the art could implement various other feature combinations without departing from the scope and spirit of the invention. Having thus described aspects of the invention, with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

WHAT IS CLAIMED IS:

1. A method for learning prototypical options for interpretable imitation learning, the method comprising:

initializing (701) options by bottleneck state discovery, each of the options presented by an instance of trajectories generated by experts;

applying (703) segmentation embedding learning to extract features to represent current states in segmentations by dividing the trajectories into a set of segmentations;

learning (705) prototypical options for each segment of the set of segmentations to mimic expert policies by minimizing loss of a policy and projecting prototypes to the current states;

training (707) option policy with imitation learning techniques to learn a conditional policy;

generating (709) interpretable policies by comparing the current states in the segmentations to one or more prototypical option embeddings; and

taking (711) an action based on the interpretable policies generated.

2. The method of claim 1, wherein option initialization includes identifying states from the current states that connect different densely connected regions in a state space.

3. The method of claim 2, wherein a soft attention mechanism is employed to obtain important states with particular attention weights.

4. The method of claim 3, wherein the important states are found with density-based spatial clustering of applications with noise (DBSCAN).

5. The method of claim 1, wherein the bottleneck state discovery divides the trajectories generated by the experts into disjoint segments of variable length by a density-based clustering method.

6. The method of claim 1, wherein each of the options includes an intra-option policy, a termination condition, an initiation state set, and an option prototype.

7. The method of claim 6, wherein the option prototype is defined by a sub-trajectory generated by the experts.

8. The method of claim 1, wherein each of the one or more prototypical option embeddings is assigned with a respective closest segment embedding in a training set.

9. The method of claim 1, wherein the loss is a least square loss.

10. The method of claim 1, wherein a diversity regularization term is employed to penalize one or more of the prototypical options that are close to each other.

11. A non-transitory computer-readable storage medium comprising a computer-readable program for learning prototypical options for interpretable imitation learning, wherein the computer-readable program when executed on a computer causes the computer to perform the steps of:

initializing (701) options by bottleneck state discovery, each of the options presented by an instance of trajectories generated by experts;

applying (703) segmentation embedding learning to extract features to represent current states in segmentations by dividing the trajectories into a set of segmentations;

learning (705) prototypical options for each segment of the set of segmentations to mimic expert policies by minimizing loss of a policy and projecting prototypes to the current states;

training (707) option policy with imitation learning techniques to learn a conditional policy;

generating (709) interpretable policies by comparing the current states in the segmentations to one or more prototypical option embeddings; and

taking (711) an action based on the interpretable policies generated.

12. The non-transitory computer-readable storage medium of claim 11, wherein option initialization includes identifying states from the current states that connect different densely connected regions in a state space.

13. The non-transitory computer-readable storage medium of claim 12, wherein a soft attention mechanism is employed to obtain important states with particular attention weights.

14. The non-transitory computer-readable storage medium of claim 13, wherein the important states are found with density-based spatial clustering of applications with noise (DBSCAN).

15. The non-transitory computer-readable storage medium of claim 11, wherein the bottleneck state discovery divides the trajectories generated by the experts into disjoint segments of variable length by a density-based clustering method.

16. The non-transitory computer-readable storage medium of claim 11, wherein each of the options includes an intra-option policy, a termination condition, an initiation state set, and an option prototype.

17. The non-transitory computer-readable storage medium of claim 16, wherein the option prototype is defined by a sub-trajectory generated by the experts.

18. The non-transitory computer-readable storage medium of claim 11, wherein each of the one or more prototypical option embeddings is assigned with a respective closest segment embedding in a training set.

19. A method for learning prototypical options for interpretable imitation learning, the method comprising:

dividing a task, by a processor, into a plurality of sub-tasks via a learning policy over options;

learning, by the processor, different options to solve each of the plurality of sub-tasks by mimicking expert policy; and

fine-tuning the learning policy to learn to take an action based on the task.

20. The method of claim 19, wherein a soft attention mechanism is employed to obtain important states in the plurality of sub-tasks, the important states found with density-based spatial clustering of applications with noise (DBSCAN).

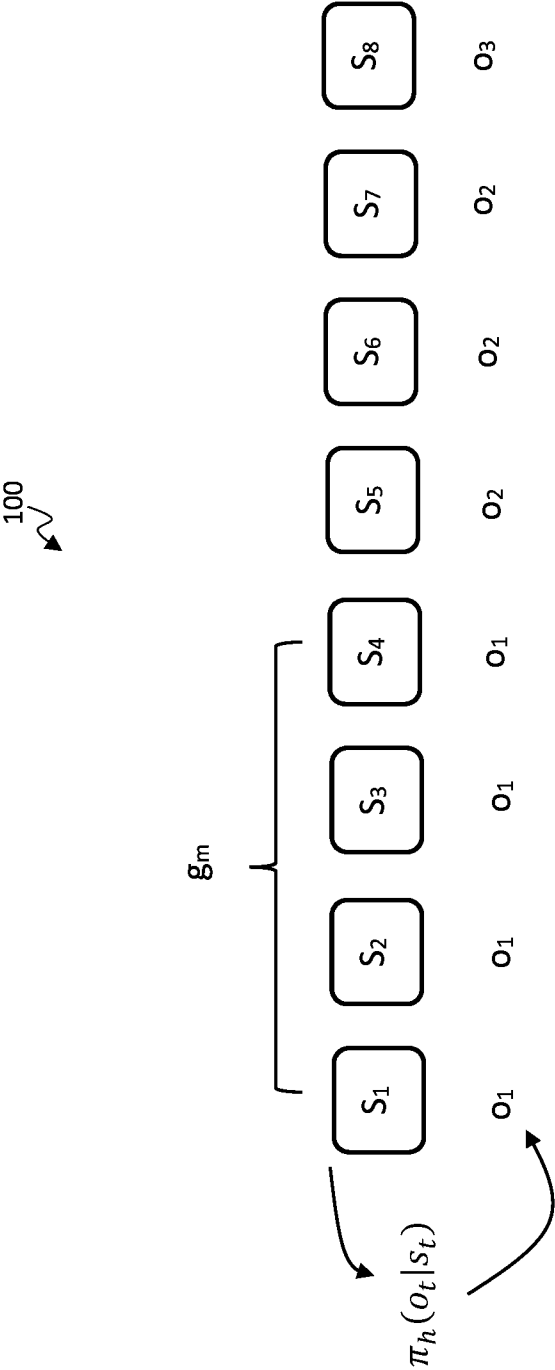


FIG. 1

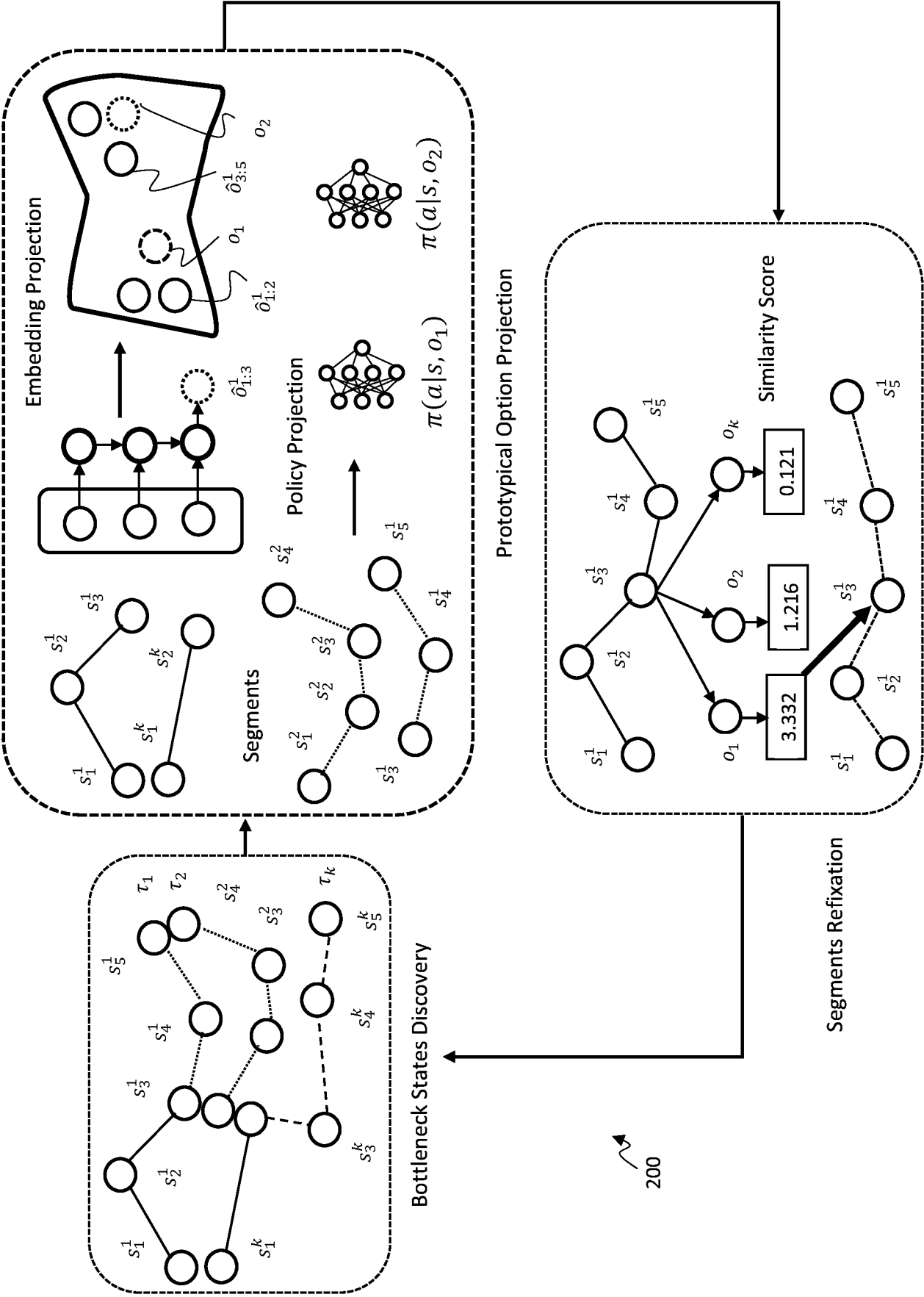


FIG. 2

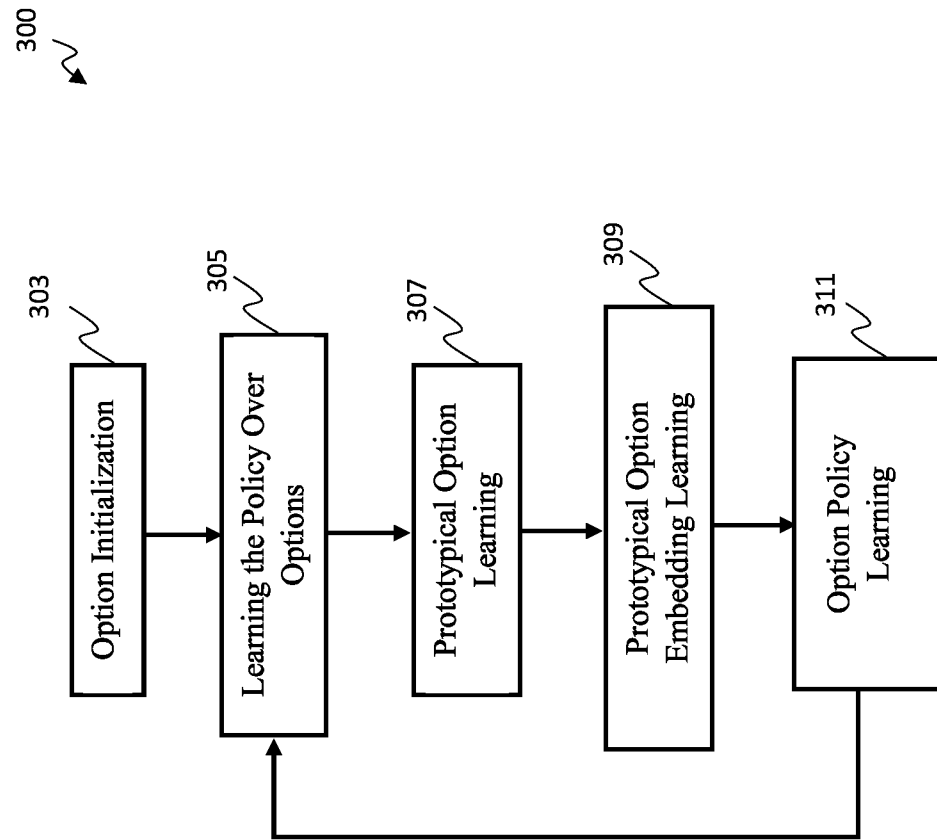


FIG. 3

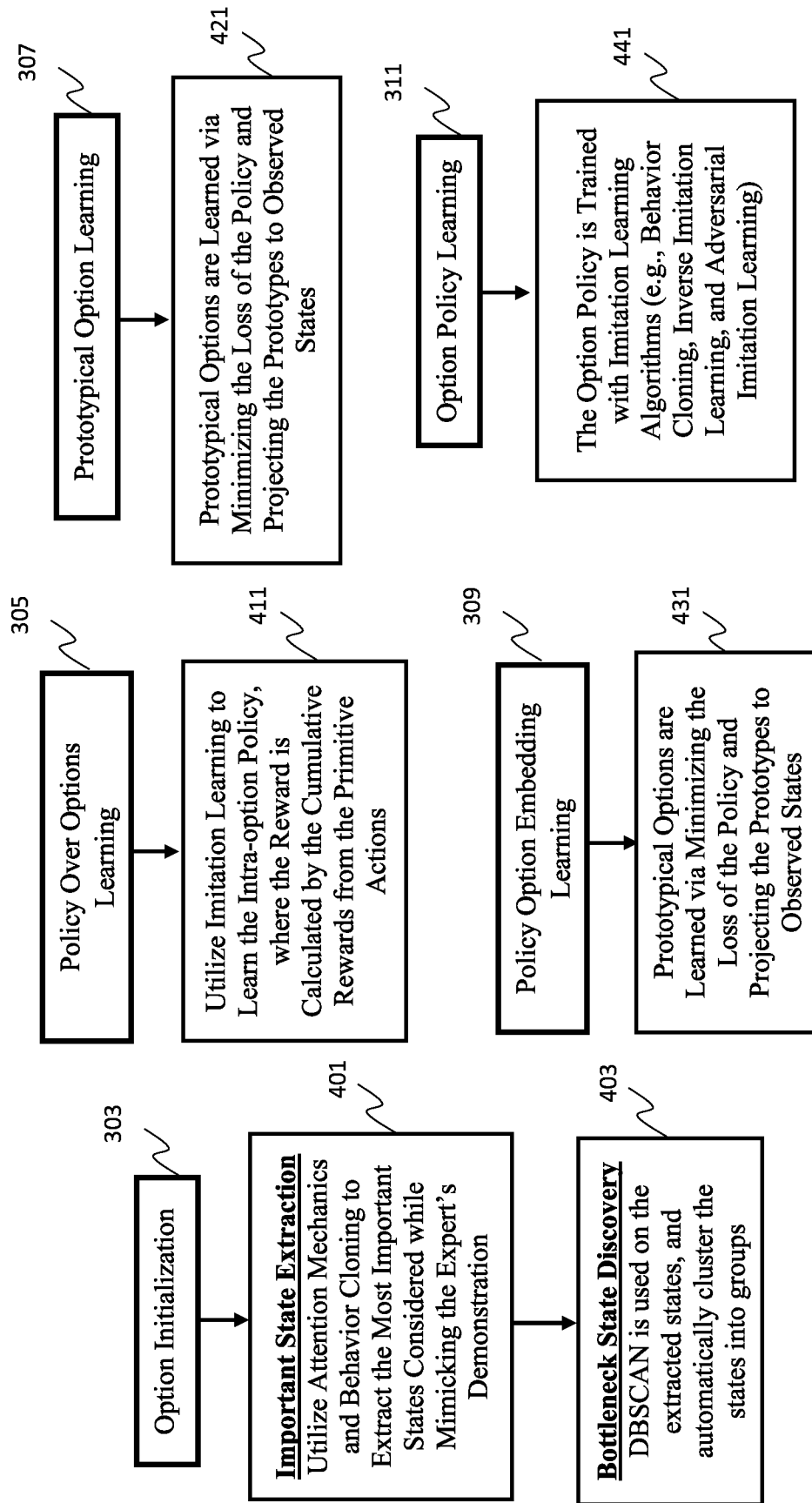


FIG. 4

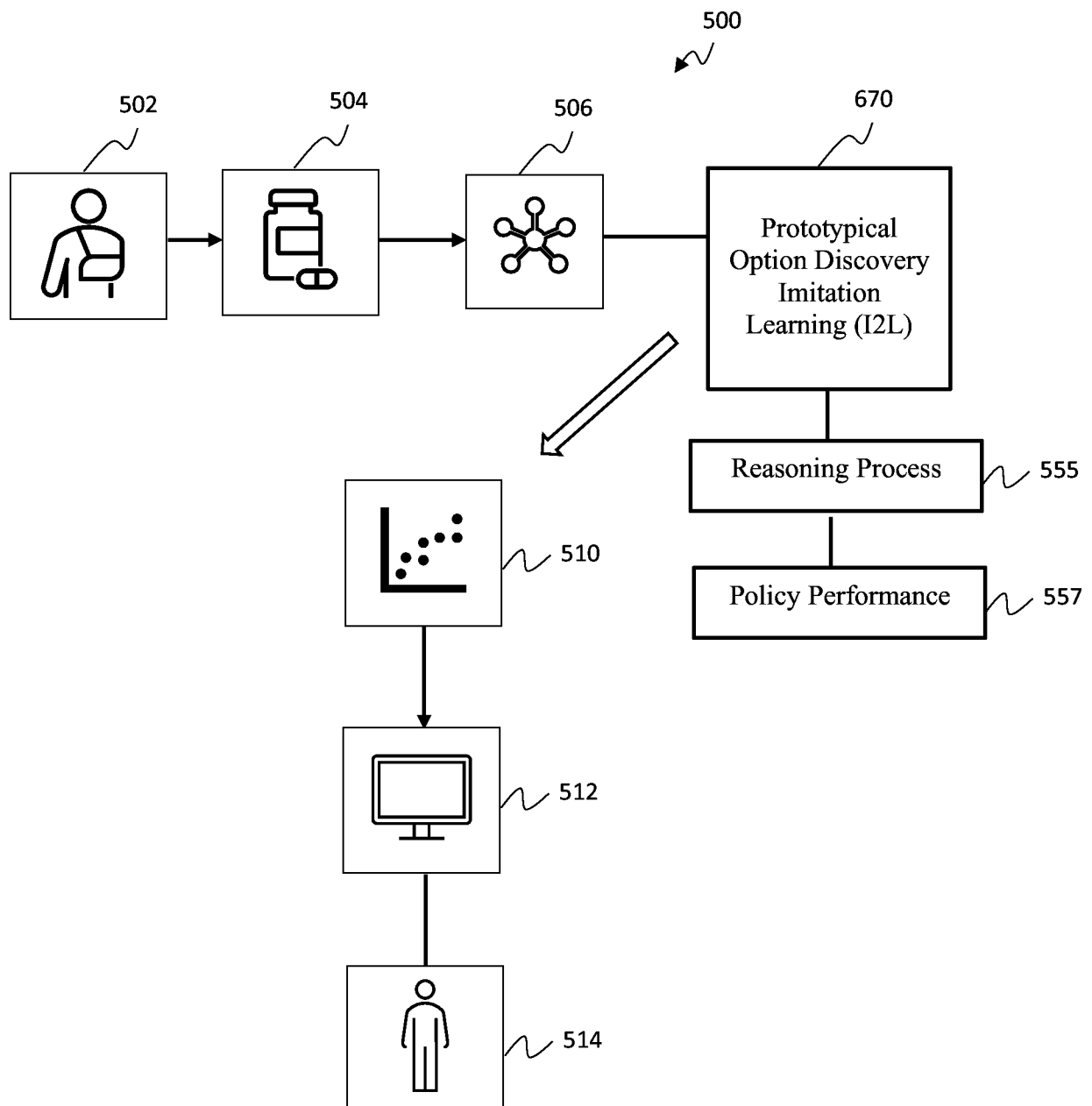


FIG. 5

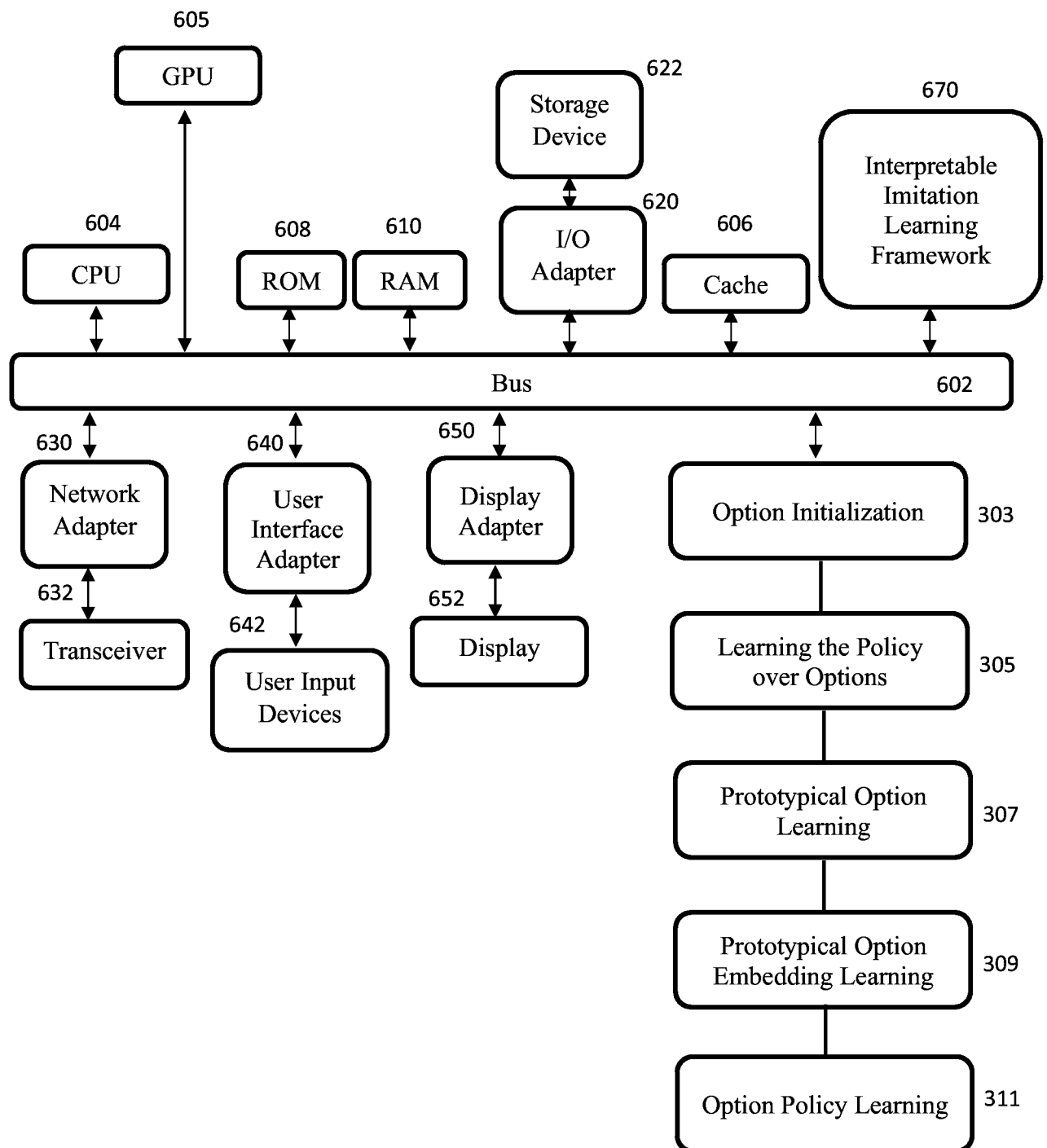
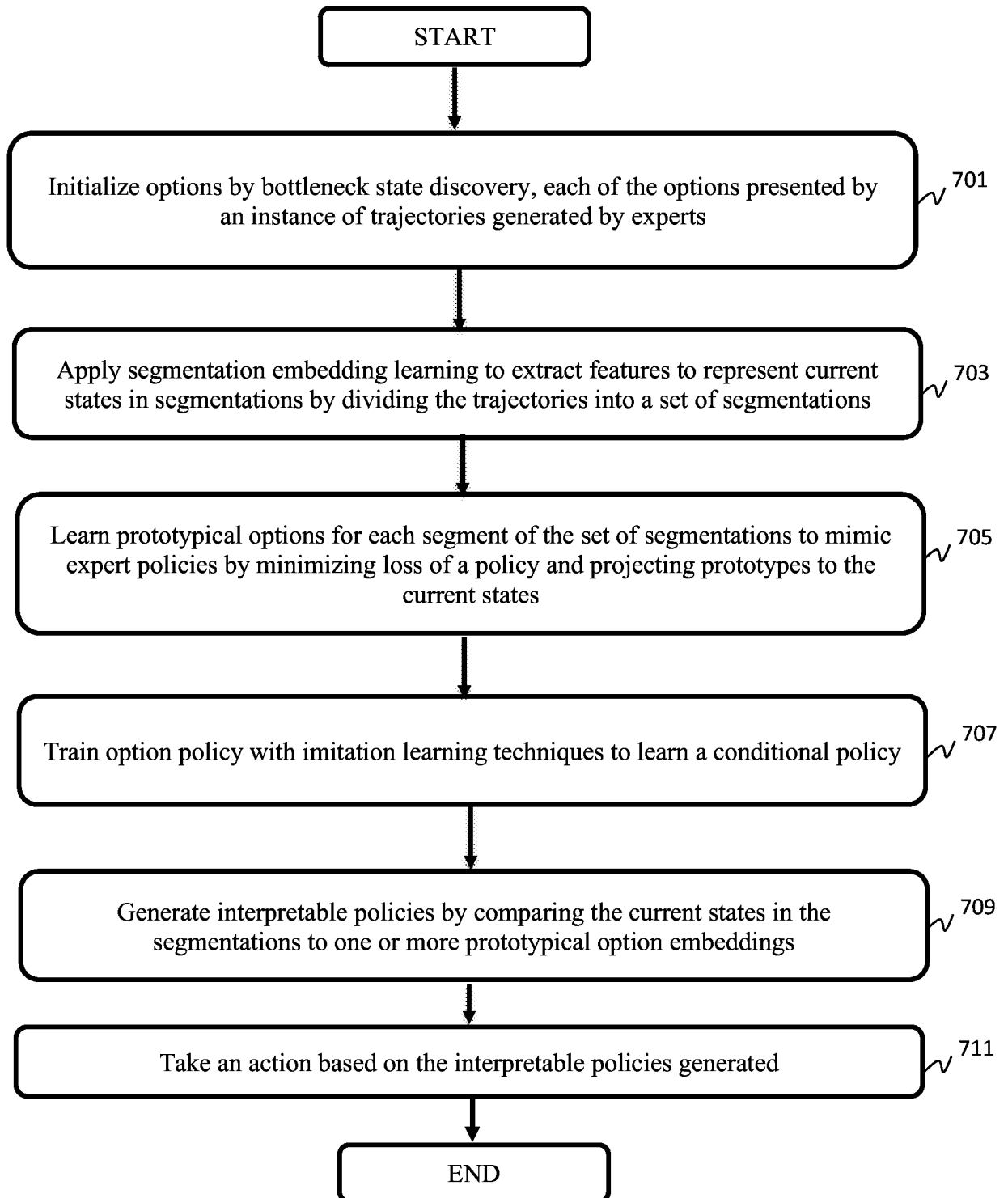


FIG. 6

**FIG. 7**

800

Option Prototype Learning

$$\mathcal{L}_{emb} = \sum_{m=1}^M \min_{k=1}^K \|f_{\phi}(s_{v_{m'}:v_m}) - \mathbf{o}_k\|_2^2$$

Option Prototype Embedding Learning

$$\begin{aligned} \mathcal{L}_{option} = & -\lambda_1 * \mathcal{L}_{IM_{loss}} + \lambda_2 * \sum_{i=1}^K \min_{m=1}^M \|f_{\phi}(s_{v_{m'}:v_m}) - \mathbf{o}_i\|_2^2 + \\ & \lambda_3 * \sum_{i=1}^K \sum_{j=i+1}^K \max(0, d_{min} - \|\mathbf{o}_i - \mathbf{o}_j\|) \end{aligned}$$

Option Policy Learning

$$\begin{aligned} \mathcal{L}_{IMBC} = & \sum_{\mathbf{o}_k \in O} \left[\sum_{\langle s, a \rangle \sim \tau} \left[\sum_{k=1}^K \pi_E(o_k | s) \log \pi_{\theta, O}(o_k | s, \mathbf{o}_k) \right] \right] + \\ \mathcal{L}_{IMAIL} = & \sum_{\mathbf{o}_k \in O} \left[\sum_{\langle s, a \rangle \sim \tau} \left[D_{KL} \left(\rho_{\pi(a|s, \mathbf{o}_k)} \parallel \frac{\rho_{\pi(a|s, \mathbf{o}_k)} + \rho_{\pi_E}}{2} \right) \right. \right. \\ & \left. \left. + D_{KL} \left(\rho_{\pi_E} \parallel \frac{\rho_{\pi(a|s, \mathbf{o}_k)} + \rho_{\pi_E}}{2} \right) \right] \right] \end{aligned}$$

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US2021/033107

A. CLASSIFICATION OF SUBJECT MATTER G06N 20/00(2019.01) According to International Patent Classification (IPC) or to both national classification and IPC				
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) G06N 20/00(2019.01)				
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Korean utility models and applications for utility models Japanese utility models and applications for utility models				
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) eKOMPASS(KIPO internal) & Keywords: bottleneck state discovery, options, segmentation, prototypes, current states, policies				
C. DOCUMENTS CONSIDERED TO BE RELEVANT				
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.		
A	ROY FOX et al. Multi-Level Discovery of Deep Options. arXiv:1703.08294v2. 05 October 2017. Pages 1-19. [retrieved on 2021.08.30]. Retrieved from <URL: https://arxiv.org/pdf/1703.08294.pdf > Sections 3.3-3.4, 4.1-4.3	1-20		
A	AMEET DESHPANDE et al. DISCOVERING HIERARCHIES USING IMITATION LEARNING FROM HIERARCHY AWARE POLICIES. arXiv:1812.00225v2. 05 August 2019. Pages 1-14. [retrieved on 2021.08.30] Retrieved from <URL: https://arxiv.org/pdf/1812.00225.pdf > Section 4.1	1-20		
A	MATTHEW COCKCROFT et al. Learning Options from Demonstration using Skill Segmentation. arXiv:2001.06793v1. January 2020. Pages 1-6. [retrieved on 2021.08.30] Retrieved from <URL: https://arxiv.org/pdf/2001.06793.pdf > Section III	1-20		
A	YUEJUN GUO et al. IBVis: Interactive Visual Analytics for Information Bottleneck Based Trajectory Clustering. Information Theory Application in Visualization. Entropy 2018. 02 March 2018, pages 1-15 Section 4	1-20		
☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.				
<table style="width: 100%; border: none;"> <tr> <td style="width: 50%; vertical-align: top; border: none;"> * Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "D" document cited by the applicant in the international application "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed </td> <td style="width: 50%; vertical-align: top; border: none;"> "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family </td> </tr> </table>			* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "D" document cited by the applicant in the international application "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "D" document cited by the applicant in the international application "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family			
Date of the actual completion of the international search 17 September 2021		Date of mailing of the international search report 17 September 2021		
Name and mailing address of the ISA/KR Korean Intellectual Property Office 189 Cheongsa-ro, Seo-gu, Daejeon 35208, Republic of Korea Facsimile No. +82-42-481-8578		Authorized officer KO, Jae Yong Telephone No. +82-42-481-8212		

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US2021/033107

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SHIRIN SOHRABI et al. State Projection via AI Planning. Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17). 02 February 2017, pages 4611-4617 Pages 4612-4613	1-20