1 Modular policy networks and problem formulations

1. They want to enable transfer across pre-defined discrete degree of variations (DoVs).
2. The DoVs can be different robot morphologies, task goals, etc.
3. w defined to be a world, an instantiation of a DoV.
4. $\mathcal{U}$ defined to be a universe, the set of all possible worlds.
5. They restrict their settings to 2 DoVs, robot and task. R is the number of robots and K is the number of tasks.

1.1 Preliminaries

1. For each world w, defined observations o_w and actions u_w.
2. π_w is a policy for world w. $\pi_w\left(u_w|o_w\right)$ defines a distribution over actions given observations.
3. Objective is to minimize sum of costs $E_{\pi_w}\left(\sum_{t=0}^{T} c\left(o_w, t\right)\right)$.
4. Decompose $o_w = \{o_{w,\mathcal{R}}, o_{w,\mathcal{T}}\}$, where the first term is robot-specific part of the observation and the second is task-specific. Similar decomposition for the state x_w.
5. Decompose cost $c\left(x_w, u_w\right) = c_{\mathcal{R}}\left(x_{w,\mathcal{R}}, u_w\right) + c_{\mathcal{T}}\left(x_{w,\mathcal{T}}\right)$. Action only affects robot-dependent cost term.

1.2 Modularity

1. The problem is to transfer information along values of each DoV while the remaining DoVs change.
2. Define $\pi_{w_{rk}}(u|o)$ to be the policy for world w with robot r performing task k.
3. Let the policy be a distribution parameterized by $\phi_{w_{rk}}(o)$.
4. Decompose $\phi_{w_{rk}}(o)$ into f_r (robot-specific) and g_k (task-specific) modules.

$$\phi_{w_{rk}}\left(o_w\right) = \phi_{w_{rk}}\left(o_{w,\mathcal{T}}, o_{w,\mathcal{R}}\right) = f_r\left(g_k\left(o_{w,\mathcal{T}}\right), o_{w,\mathcal{R}}\right)$$

5. The reason the policy is decomposed in this order is: they expect that the identify of the task would affect the task plan of the policy while the robot configuration would affect the action output.
6. Each robot module and task module have separate sets of parameters
7. This is so that: world w with the same robot instantiation r would reuse the same robot module f_r, while worlds with the same task instantiation would use the same task module g_k.
8. An important point to note is that the output of task module g_k is not fixed or supervised to have a specific semantic meaning. It is a representation that is learnt.
9. Notation-wise, during transfer test phase, given unseen world w_{test}, using robot r_{test} to perform k_{test}, module $f_{r_{\text{test}}}$ and $g_{k_{\text{test}}}$ are composed.
10. They require that some world in the training set must have use robot r_{test} and some other world must have had task k_{test}. But they need not have been trained together.
11. As the number and variety of worlds which use a particular module increases, the module becomes increasingly invariant to changes in other DoV, which is crucial for generalization.

1.3 Architecture and Training

1. $\phi_{w_{rk}}(o)$ is a NN.
2. $\pi_{w_{rk}}(u|o) = N(\phi_{w_{rk}}(o), \Sigma)$ where Σ is learned by state-independent.
3. Several training world are chosen with combinations of robots and tasks, such that every module has been trained on at least one world.
4. The NN is trained using samples from all the worlds synchronously.
5. The objective $\mathcal{L} = \sum_w \mathcal{L}_w$.
6. Regularization: in order to attain zero-shot performance on unseen robot-task combination, the robot and task module must not overfit to any specific robot-task combination.
7. With only a few robots and task (3 and 3), they found overfitting to be an issue.
8. Two regularization schemes are used: limiting the number of hidden units in the network and dropout.