Network Architecture and Policy Learning Methods

We use a single end-to-end architecture that combines the two sources of information, [CSP*17]. These two information sources are divided into two modules, a state processing module, which we describe next, and a policy learning module, which we describe below.

As defined above, our states are defined by \( s_t = (I, M_t) \) where \( M_t \) is the first-person view of the environment at iteration \( t \). Each frame is fed through a convolutional neural network to create a convolutional representation of the state, \( x_M = f(M_t, \theta_{conv}) \in \mathbb{R}^{C \times H \times W} \), where \( d \) denotes the number of feature maps, i.e., the intermediate representations in the convolutional network. This is shown in the upper module of Figure 2.

\[
M_{GA}(x_I, x_M) = M(h(x_I)) \odot x_M = M(a_I) \odot x_M
\]

We use Minecraft as a platform to examine task-oriented language grounding.

**Attention and Hadamard Products**

In addition, these instruction embeddings are passed through a fully-connected layer with a sigmoid activation function to create an attention vector, \( a_I = h(x_I) \in \mathbb{R}^d \). Furthermore, each element of the attention vector is expanded into a \( H \times W \) matrix to match the size of our state-image, which is then multiplied element-wise with the output of the CNN (i.e., we use the Hadamard product of the two embeddings):

\[
M_{GA}(x_I, x_M) = M(h(x_I)) \odot x_M = M(a_I) \odot x_M
\]

**Policy Learning Module**

The multimodal fusion unit, \( M_{GA} \) is provided to the policy learning module. We use imitation learning to learn the optimal policy. In particular, an oracle is implemented that finds the target object’s location described by the natural language instructions. It then orients itself in the direction facing the longest corridor to the object and finds the shortest path through an \( A^* \) path to the object. The imitation learning module is trained with a single fully connected layer to estimate the policy function. Alternatively, we also utilize a Stein policy gradient, a particle based variational method that controls the gradient variance, [DW16].

Instead of optimizing for a single policy, Stein variational policy gradient searches for a distribution \( q(\theta) \) to optimize the expected return using variance reduction methods. Concretely, we formulate our variational optimization objective as

\[
\max \left\{ \mathbb{E}_\theta [J(\theta)] + \alpha \mathbb{E}_q [H(q)] \right\}
\]

where the second term provides entropy of the variational distribution \( q \), and \( \alpha \) is a “temperature” parameter that controls the rate of exploration. Its optimal value is given by

\[
q(\theta) \propto \exp \left\{ \frac{1}{\alpha} J(\theta) \right\} q_0(\theta).
\]

This variational objective has the advantage of acting as a regularizer/control variate, and a fast particle based simulation mechanism.

References

