A3C Code

This code encompasses the classes we used to build the A3C Network and agent.

1 ResourceGather.main.main main Reference

This is the primary entry point for our program.

Static Public Attributes

- int **max_episode_length** = 300
- bool **load_model** = False
- string **model_path** = './model'
- **flags** = common.parse_args.parse_args()
- **enabled**
- **trace**
- **agent_cls** = agent.Smart
- **input_size**
- **num_actions**
- **max_episodes**
- **q_range**
- **hidden_layer_size**
- **base_explore_rate**
- **min_explore_rate**
- **config** = tf.ConfigProto(allow_soft_placement=True)
- **global_episodes** = tf.Variable(0, dtype=tf.int32, name="global_episodes", trainable=False)
- **optimizer** = tf.train.AdamOptimizer(learning_rate=0.005)
- **master_network** = agent.network.Policy('global', global_episodes, agent.policy_spec)
- **num_workers** = psutil.cpu_count()
- list **workers** = [ ]
- string **name** = "worker_" + str(i)
- **agent_inst** = agent_cls(name, 'global', optimizer, global_episodes, Action_Space(), flags)
- **env**
- **saver** = tf.train.Saver(max_to_keep=5)
- **coord** = tf.train.Coordinator()
- **ckpt** = tf.train.get_checkpoint_state(model_path)
- list **worker_threads** = [ ]
- **t** = threading.Thread(target=(lambda: worker.work(max_episode_length, sess, coord, saver)))

1.1 Detailed Description

This is the primary entry point for our program.

Initial flags are set via parse_args(). The agent is then initialized and the policy_spec is updated. We ensure the map is loaded then reset the TensorFlow graph in case anything was still residing in memory. The TensorFlow device gets configured and we initialize the global episodes to zero, setup the optimizer, and define the number of workers.

Global values are then declared so they can be initialized. The worker list is created and filled with however many workers were defined earlier. For each worker we initialize a StarCraft II environment and fill the worker with the necessary data for it to run.

We then begin a TensorFlow session and initialize worker threads for each worker. The workers will continue to run until all episodes have been completed.

Origin

The documentation for this class was generated from the following file:

- main.py

2 ResourceGather.Action_Space.Action_Space Class Reference

Define the units and their actions.

Public Member Functions

- def __init__ (self)

  Constructs the object.
- def Start_pos (self, obs)

  Defines the start location.
- def check_available_actions (self, obs)

  Takes in the available actions from the observation (should be a list of action_ids) and returns a list of 0's and 1's with respect to our action space.
- def act (self, index, obs, drone_id=None)

  Takes an integer action index (corresponding to the i_th action in the action space) returns 1 if the action is available and can be added to the queue, -1 if not.
- def action_step (self, env_obs)

  This function inerprets the action queue and constructs arguments to be returned.
- def build_Hatchery (self, obs, drone_id)

  Builds a hatchery.
- def build_Gas_Gyser (self, obs, drone_id)

  Builds a gas gyser.
- def build_Spawning_Pool (self, obs, drone_id)

  Queues the building of a spawning pool if one isn't already built or being built.
- def harvest_Minerals (self, obs, drone_id)

  Queues an action to have the selected unit harvest minerals.
- def harvest_Gas (self, obs, drone_id)

  Queues an action to have the selected unit harvest gas.
- def inject_Larva (self, obs, queen_id)

  Queues an action to have the selected queen inject a Hatchery.
- def train_Drone (self, obs, larva_id)

  Queues a train Drone action.
- def train_Overlord (self, obs, drone_id)

  Queues a train Overlord action.
- def train_Queen (self, obs, hatch_id)

  Queues a train Queen action.
- def drone_busy (drone_id)

  Function for checking if drones are doing a non-interuptable task.

Public Attributes

- **busy_units**
- **actionq**
- **pointq**
- **expo_count**
- **action_dict**
- **action_map**
- **top_left**
- **pool_flag**

Static Public Attributes

- dictionary **valid_units**
- bool **pool_flag** = False
- bool **top_left** = True
- dictionary **action_spec**

2.1 Detailed Description

Define the units and their actions.

2.2 Constructor & Destructor Documentation

2.2.1 __init__()

```
def ResourceGather.Action_Space.Action_Space.__init__ (
              self )
```

Constructs the object.

Parameters

self	The object

2.3 Member Function Documentation

2.3.1 act()

```
def ResourceGather.Action_Space.Action_Space.act (
              self,
              index,
              obs,
              drone_id = None )
```

Takes an integer action index (corresponding to the i_th action in the action space) returns 1 if the action is available and can be added to the queue, -1 if not.

Parameters

self	The object
index	The index
obs	The observation
drone_id	The drone identifier

Returns

Returns an integer reward value

2.3.2 action_step()

```
def ResourceGather.Action_Space.Action_Space.action_step (
              self,
              env_obs )
```

This function inerprets the action queue and constructs arguments to be returned.

Parameters

self	The object
env_obs	The environment observeration

Returns

An action call and reward

2.3.3 build_Gas_Gyser()

```
def ResourceGather.Action_Space.Action_Space.build_Gas_Gyser (
              self,
              obs,
              drone_id )
```

Builds a gas gyser.

Parameters

self	The object
obs	The observeration
drone_id	The drone identifier

Returns

The gas gyser.

2.3.4 build_Hatchery()

```
def ResourceGather.Action_Space.Action_Space.build_Hatchery (
              self,
              obs,
              drone_id )
```

Builds a hatchery.

Parameters

self	The object
obs	The observeration
drone_id	The drone identifier

2.3.5 build_Spawning_Pool()

```
def ResourceGather.Action_Space.Action_Space.build_Spawning_Pool (
              self,
              obs,
              drone_id )
```

Queues the building of a spawning pool if one isn't already built or being built.

Parameters

self	The object
obs	The observeration
drone_id	The drone identifier

Returns

Returns false unless the pool has finished constructing.

2.3.6 check_available_actions()

```
def ResourceGather.Action_Space.Action_Space.check_available_actions (
              self,
              obs )
```

Takes in the available actions from the observation (should be a list of action_ids) and returns a list of 0's and 1's with respect to our action space.

0 if the i_th action is not available, 1 if it is available.

Parameters

self	The object
obs	The observation

Returns

list of 0's and 1's of length(num_actions)

2.3.7 drone_busy()

```
def ResourceGather.Action_Space.Action_Space.drone_busy (
              drone_id )
```

Function for checking if drones are doing a non-interuptable task.

Parameters

drone_id	The drone identifier

Returns

True if busy, False otherwise

2.3.8 harvest_Gas()

```
def ResourceGather.Action_Space.Action_Space.harvest_Gas (
              self,
              obs,
              drone_id )
```

Queues an action to have the selected unit harvest gas.

Parameters

self	The object
obs	The observation
drone_id	The drone identifier

2.3.9 harvest_Minerals()

```
def ResourceGather.Action_Space.Action_Space.harvest_Minerals (
              self,
              obs,
              drone_id )
```

Queues an action to have the selected unit harvest minerals.

Parameters

self	The object
obs	The observation
drone_id	The drone identifier

2.3.10 inject_Larva()

```
def ResourceGather.Action_Space.Action_Space.inject_Larva (
              self,
              obs,
              queen_id )
```

Queues an action to have the selected queen inject a Hatchery.

Parameters

self	The object
obs	The observation
queen_id	The queen identifier

2.3.11 Start_pos()

```
def ResourceGather.Action_Space.Action_Space.Start_pos (
              self,
              obs )
```

Defines the start location.

Parameters

self	The object
obs	The observation

2.3.12 train_Drone()

```
def ResourceGather.Action_Space.Action_Space.train_Drone (
              self,
              obs,
              larva_id )
```

Queues a train Drone action.

Parameters

self	The object
obs	The observation
larva_id	The larva identifier

2.3.13 train_Overlord()

```
def ResourceGather.Action_Space.Action_Space.train_Overlord (
              self,
              obs,
              drone_id )
```

Queues a train Overlord action.

Parameters

self	The object
obs	The observation
drone_id	The drone identifier

Returns

{ description_of_the_return_value }

2.3.14 train_Queen()

```
def ResourceGather.Action_Space.Action_Space.train_Queen (
              self,
              obs,
              hatch_id )
```

Queues a train Queen action.

Parameters

self	The object
obs	The observation
hatch↩ _id	The hatch identifier

2.4 Member Data Documentation

2.4.1 action_spec

```
dictionary ResourceGather.Action_Space.Action_Space.action_spec  [static]
```

Initial value:

```
=  {
    'number of actions': _MAX_AVAIL_ACTIONS # Actions should be in a dict or something so we can run len()
       etc. on them
    }
```

2.4.2 valid_units

```
dictionary ResourceGather.Action_Space.Action_Space.valid_units  [static]
```

Initial value:

```
=  {126: ("Move_screen", "Effect_InjectLarva_screen"), #queen
       104: ("Move_screen", "Harvest_Gather_screen", "Build_Hatchery_screen", "Build_SpawningPool_screen",
      "Build_Extractor_screen"), #drone
       151: ("Train_Drone_quick", "Train_Overlord_quick"), #larva
       86: ("Train_Queen_quick")}
```

The documentation for this class was generated from the following file:

- Action_Space.py

3 ResourceGather.Action_Space.ActionEnum Class Reference

Enumerator for actions.

Static Public Attributes

- int **build_Hatchery** = 0
- int **build_Gas_Gyser** = 1
- int **train_Drone** = 2
- int **train_Overlord** = 3
- int **train_Queen** = 4
- int **inject_Larva** = 5
- int **harvest_Minerals** = 6
- int **harvest_Gas** = 7
- int **no_op** = 8

3.1 Detailed Description

Enumerator for actions.

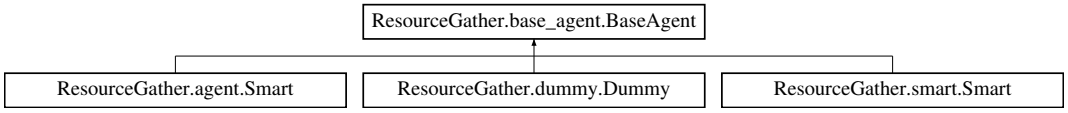
The documentation for this class was generated from the following file:

- Action_Space.py

4 ResourceGather.base_agent.BaseAgent Class Reference

This is the BaseAgent class that PySC2 expects.

Inheritance diagram for ResourceGather.base_agent.BaseAgent:



Public Member Functions

- def __init__ (self, name, parent, optimizer, network, episode, policy_spec, trainer_spec, hyper_params)

  Constructs the object and initializes policy, trainer, and update_local_policy members.
- def train (self, sess, actions, rewards, observations, values)

  This is a wrapper function used to have the trainer train.
- def step (self, sess, observation)

  Wrapper for policy.step.
- def value (self, sess, obs)

  Wrapper for policy.get_values.
- def update_policy (self, sess)

  Makes a call to helper function update_target_graph, which then gets fed into the currently running TensorFlow session.
- def process_observation (self, obs)

  This function raises an error if called.

Public Attributes

- **policy**
- **trainer**
- **update_local_policy**

4.1 Detailed Description

This is the BaseAgent class that PySC2 expects.

It contains functions necessary to run an agent.

4.2 Constructor & Destructor Documentation

4.2.1 __init__()

```
def ResourceGather.base_agent.BaseAgent.__init__ (
              self,
              name,
              parent,
              optimizer,
              network,
              episode,
              policy_spec,
              trainer_spec,
              hyper_params )
```

Constructs the object and initializes policy, trainer, and update_local_policy members.

Parameters

self	The object
name	The name
parent	The parent
optimizer	The optimizer
network	The network
episode	The episode
policy_spec	The policy specifier
trainer_spec	The trainer specifier
hyper_params	The hyper parameters

4.3 Member Function Documentation

4.3.1 process_observation()

```
def ResourceGather.base_agent.BaseAgent.process_observation (
              self,
              obs )
```

This function raises an error if called.

We should not be making a call to this particular process_observation.

Parameters

self	The object
obs	The observation

4.3.2 step()

```
def ResourceGather.base_agent.BaseAgent.step (
              self,
              sess,
              observation )
```

Wrapper for policy.step.

Parameters

self	The object
sess	The sess
observation	The observation

Returns

Returns the results from network.py's step function

4.3.3 train()

```
def ResourceGather.base_agent.BaseAgent.train (
              self,
              sess,
              actions,
              rewards,
              observations,
              values )
```

This is a wrapper function used to have the trainer train.

Parameters

self	The object
sess	The sess
actions	The actions
rewards	The rewards
observations	The observations
values	The values

Returns

Returns the results from network.py's train function

4.3.4 update_policy()

```
def ResourceGather.base_agent.BaseAgent.update_policy (
              self,
              sess )
```

Makes a call to helper function update_target_graph, which then gets fed into the currently running TensorFlow session.

Parameters

self	The object
sess	The TensorFlow session

4.3.5 value()

```
def ResourceGather.base_agent.BaseAgent.value (
              self,
              sess,
              obs )
```

Wrapper for policy.get_values.

Parameters

self	The object
sess	The sess
obs	The obs

Returns

Returns the results from network.py's get_values function

The documentation for this class was generated from the following file:

- base_agent.py

5 ResourceGather.network.Policy Class Reference

The Policy class defines the basic structure of the network such as how the hidden layers are setup.

Public Member Functions

- def __init__ (self, scope, episode, policy_spec, hyper_params)

  Constructs the object.
- def reset (self)

  On an episode reset we want to reset the random explore rate.
- def step (self, sess, obs)

  The network needs to step through iterations with the environment, this is the function to accomplish this.
- def get_value (self, sess, obs)

  Gets the value by running a TensorFlow session and calculating the value as it runs through the graph.

Static Public Member Functions

- def policy_spec (input_size=None, num_actions=None, max_episodes=None, q_range=None, hidden_↩ layer_size=None, base_explore_rate=0.1, min_explore_rate=None)

  Creates a dictionary of policy_spec values to be returned.

Public Attributes

- **hyper_params**
- **network_spec**
- **random_explore_rate**
- **exploration_rate**
- **exploration**
- **input**
- **q**
- **probs**
- **action**
- **value**

5.1 Detailed Description

The Policy class defines the basic structure of the network such as how the hidden layers are setup.

5.2 Constructor & Destructor Documentation

5.2.1 __init__()

```
def ResourceGather.network.Policy.__init__ (
            self,
            scope,
            episode,
            policy_spec,
            hyper_params )
```

Constructs the object.

Defines the structure of the network and sets up the TensorFlow graph. The input layer as well as hidden layers and output are

Parameters

self	The object
scope	The scope
episode	The episode
policy_spec	The policy specifier
hyper_params	The hyper parameters

5.3 Member Function Documentation

5.3.1 get_value()

```
def ResourceGather.network.Policy.get_value (
              self,
              sess,
              obs )
```

Gets the value by running a TensorFlow session and calculating the value as it runs through the graph.

Parameters

self	The object
sess	The session
obs	The observation

Returns

The value.

5.3.2 policy_spec()

```
def ResourceGather.network.Policy.policy_spec (
              input_size = None,
              num_actions = None,
              max_episodes = None,
              q_range = None,
              hidden_layer_size = None,
              base_explore_rate = 0.1,
              min_explore_rate = None )  [static]
```

Creates a dictionary of policy_spec values to be returned.

Parameters

input_size	The input size
num_actions	The number actions
max_episodes	The maximum episodes
q_range	The quarter range
hidden_layer_size	The hidden layer size
base_explore_rate	The base explore rate
min_explore_rate	The minimum explore rate

Returns

Returns a dictionary of the items passed in.

5.3.3 reset()

```
def ResourceGather.network.Policy.reset (
              self )
```

On an episode reset we want to reset the random explore rate.

Parameters

self	The object

5.3.4 step()

```
def ResourceGather.network.Policy.step (
              self,
              sess,
              obs )
```

The network needs to step through iterations with the environment, this is the function to accomplish this.

Runs a TensorFlow session to calculate the action and value that help steer the network and training.

Parameters

self	The object
sess	The session
obs	The observation

Returns

A list containing the action and value.

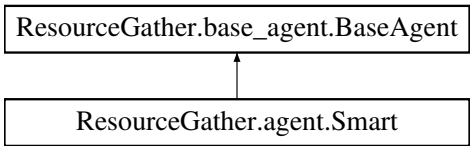
The documentation for this class was generated from the following file:

- network.py

6 ResourceGather.agent.Smart Class Reference

Smart class.

Inheritance diagram for ResourceGather.agent.Smart:



Public Member Functions

- def __init__ (self, name, parent, optimizer, episode, action_space, flags)

  Constructs the object.
- def process_observation (self, obs)

  This is the process observation wrapper function that PySC2 expects to be with the.

Public Attributes

- **action_space**
- **flags**

6.1 Detailed Description

Smart class.

Necessary for PySC2. Updates policy_spec and sets up the action_space and flags members. Also stores the process_observation function which is a necessary component for PySC2.

6.2 Constructor & Destructor Documentation

6.2.1 __init__()

```
def ResourceGather.agent.Smart.__init__ (
              self,
              name,
              parent,
              optimizer,
              episode,
              action_space,
              flags )
```

Constructs the object.

Updates the policy_spec and sets member variables.

Parameters

self	The object
name	The name
parent	The parent
optimizer	The optimizer
episode	The episode
action_space	The action space
flags	The flags

6.3 Member Function Documentation

6.3.1 process_observation()

```
def ResourceGather.agent.Smart.process_observation (
              self,
              obs )
```

This is the process observation wrapper function that PySC2 expects to be with the.

Parameters

self	The object
obs	The observation

Returns

Returns the reward, observation, and if the episode has ended

The documentation for this class was generated from the following file:

- agent.py

7 ResourceGather.network.Trainer Class Reference

This is the trainer class for our network.

Public Member Functions

- def __init__ (self, scope, optimizer, policy, trainer_spec, hyper_params)

  Constructs the object and initializes the TensorFlow graph.
- def train (self, sess, obs, actions, rewards, values)

  The train function is called when the buffer is full or episode quantity met.

Static Public Member Functions

- def trainer_spec (accuracy_coefficient=1.0, advantage_coefficient=10.0, consistency_coefficient=3.0, max↩ _grad_norm=40.0, discount_factor=0.6)

  This is the trainer specifications.

Public Attributes

- **policy**
- **hp**
- **actions**
- **rewards**
- **values**
- **accuracy_loss**
- **consistent_loss**
- **advantage**
- **loss**
- **gradients**
- **var_norms**
- **grad_norms**
- **apply_grads**

7.1 Detailed Description

This is the trainer class for our network.

It defines the TensorFlow graph nodes that will be used to calculate loss, run optimizers, and apply gradients to our network.

7.2 Constructor & Destructor Documentation

7.2.1 __init__()

```
def ResourceGather.network.Trainer.__init__ (
              self,
              scope,
              optimizer,
              policy,
              trainer_spec,
              hyper_params )
```

Constructs the object and initializes the TensorFlow graph.

Parameters

self	The object
scope	The scope
optimizer	The optimizer
policy	The policy
trainer_spec	The trainer specifications
hyper_params	The hyper parameters

7.3 Member Function Documentation

7.3.1 train()

```
def ResourceGather.network.Trainer.train (
              self,
              sess,
              obs,
              actions,
              rewards,
              values )
```

The train function is called when the buffer is full or episode quantity met.

This calls for a TensorFlow session to be ran. It passes in the feed_dict items and returns the fetches list items. TensorFlow will know that it needs to fetch self.loss so it will expect everything necessary to calculate it in the feed_dict.

Parameters

self	The object
sess	The sess
obs	The obs
actions	The actions
rewards	The rewards
values	The values

Returns

A list of lists.

7.3.2 trainer_spec()

```
def ResourceGather.network.Trainer.trainer_spec (
              accuracy_coefficient = 1.0,
```

```
advantage_coefficient = 10.0,
consistency_coefficient = 3.0,
max_grad_norm = 40.0,
discount_factor = 0.6 )  [static]
```

This is the trainer specifications.

Parameters

accuracy_coefficient	The accuracy coefficient
advantage_coefficient	The advantage coefficient
consistency_coefficient	The consistency coefficient
max_grad_norm	The maximum graduated normalize, aka grad clipping value
discount_factor	The discount factor

Returns

Returns a dictionary of the items passed in.

The documentation for this class was generated from the following file:

- network.py

8 ResourceGather.worker.Worker Class Reference

Class for worker.

Public Member Functions

- def __init__ (self, number, main, env, actions, agent, global_episodes, name=None, model_path=None, summary_dir="workerData/", episodes_per_record=10, episodes_for_model_checkpoint=250, buffer_↩ min=10, buffer_max=30, max_episodes=10000)

  Constructs the object and initializes member variables.
- def train (self, rollout, sess, bootstrap_value)

  Workers train when episode buffers are full or after so many episodes.
- def do_actions (self, choice, env_obs)

  Has the environment execute actions.
- def work (self, sess, coord, saver)

  This is the function that the worker spends most of it's time operating.

Public Attributes

- **number**
- **name**
- **model_path**
- **buffer_min**
- **buffer_max**
- **global_episodes**
- **increment**
- **episodes_for_model_checkpoint**
- **episodes_per_record**
- **episode_rewards**
- **episode_real_rewards**
- **episode_lengths**
- **episode_mean_values**
- **max_episodes**
- **summary_writer**
- **main**
- **agent**
- **actions**
- **env**

8.1 Detailed Description

Class for worker.

Workers execute their own environment as well as push and pull values to and from the global network.

8.2 Constructor & Destructor Documentation

8.2.1 __init__()

```
def ResourceGather.worker.Worker.__init__ (
                self,
                number,
                main,
                env,
                actions,
                agent,
                global_episodes,
                name = None,
                model_path = None,
                summary_dir = "workerData/",
                episodes_per_record = 10,
                episodes_for_model_checkpoint = 250,
                buffer_min = 10,
                buffer_max = 30,
                max_episodes = 10000 )
```

Constructs the object and initializes member variables.

Parameters

self	The object
number	The number
main	The main
env	The environment
actions	The actions
agent	The agent
global_episodes	The global episodes
name	The name
model_path	The model path
summary_dir	The summary directory
episodes_per_record	The episodes per record
episodes_for_model_checkpoint	The episodes for model checkpoint
buffer_min	The buffer minimum
buffer_max	The buffer maximum
max_episodes	The maximum episodes

8.3 Member Function Documentation

8.3.1 do_actions()

```
def ResourceGather.worker.Worker.do_actions (
                self,
                choice,
                env_obs )
```

Has the environment execute actions.

Parameters

self	The object
choice	The choice
env_obs	The environment observation

Returns

Reward feedback and the updated environment observation.

8.3.2 train()

```
def ResourceGather.worker.Worker.train (
              self,
              rollout,
              sess,
              bootstrap_value )
```

Workers train when episode buffers are full or after so many episodes.

This function calls agent.train and updates the policy.

Parameters

self	The object
rollout	The rollout
sess	The session
bootstrap_value	The bootstrap value

Returns

Returns the network loss, accuracy, consistency, advantage, grad_norms, and var_norms.

8.3.3 work()

```
def ResourceGather.worker.Worker.work (
              self,
              sess,
              coord,
              saver )
```

This is the function that the worker spends most of it's time operating.

Workers run a session where they reset everything and then run an instance of the environment. The environment will run until the episode is finished, at which point globals are pulled down and the values this worker generated are pushed up.

Parameters

self	The object
sess	The session
coord	The coordinator
saver	The saver

The documentation for this class was generated from the following file:

- worker.py