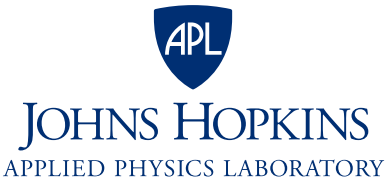


JHU/APL
http://www.jhuapl.edu/

Title:
Lifelong Learning Machines
Preliminary Metrics Documentation

Project Participants:
JHU/APL Test & Evaluation Team

Authors:
Megan Baker
Gautam Vallabha

Date:
November 25, 2019

Contents

Chapter 1

Background

In this document we introduce the Metrics component of the Test and Evaluation Framework (TEF) for Agent-based evaluation *only*. For this preliminary information release, we focus on a high level overview of the system, followed by a brief introduction to the types of syllabi and metrics which are expected, and end with how to get started with the code. Stay tuned for more details in each of these areas!

1.1 Core Capabilities

We remind readers that the purpose of the metrics discussed in this document are to determine whether a system exhibits the Core Capabilities of Lifelong Learning. These metrics may be slightly different from those which might assess a traditional learner. We invite our audience to review the five Core Capabilities whose definitions are below, and refer readers to the BAA for more details.

1. Continual Learning

Ability to handle, or adapt to, changing input distributions (or noise characteristics) within a single task. For an agent-based system, this means that the state transition matrix and reward function are substantively unchanged, while aspects of the environment may change.

2. Adapting to New Tasks

Ability to learn new tasks, without losing knowledge of already-learned tasks. If possible, system should exploit similarities between old and new tasks to improve its learning performance on new tasks. For an agent-based system, this means substantial changes to the state transition matrix or reward function.

3. Selective Plasticity

Ability to process the same input differently depending on task (or goal). Relatedly, the ability to be sensitive to different features of the input depending on task. Note the focus on processing rather than learning per se.

4. Goal-Driven Perception

Ability to incorporate system and task-level constraints (e.g., overall memory use, relative importance of tasks) into the training process.

5. Safety

Ability to incorporate explicit (failsafe) safety constraints into system performance; Ability to detect differences between training and test environments (e.g., anomalous inputs, distributional shifts), quantify the resulting uncertainty in system output, alert a human operator (with details of difference if possible), and safely handle the anomalous situation where possible.

1.2 This Document's Focus

For the purpose of this initial development, we chose to focus our attention on the first two Core Capabilities; Continual Learning and Adapting to New Tasks.

Terminology

Key Concepts
Task: A single abstract capability (or skill) that a performer system must learn
Episode: A concrete instance of a task
Syllabus: A sequence of episodes
Learning Lifetime: One syllabus or multiple syllabi in sequence

Metric Specific Terms for Syllabus Design
Phase: A subcomponent of a syllabus during which either training or evaluation takes place
Block: A unique combination of task and parameters. It is a subcomponent of phase that is automatically assigned by the logging code and does not need to be annotated by the syllabus designer.

Chapter 2

System Architecture

Overview

The Test and Evaluation Framework (TEF) uses the information contained in a syllabus to produce a sequence of episodes for an agent learner. Figure 2.1 depicts a high level overview of the TEF system. The syllabus passes the sequence of episodes to the TEF which sets up an environment and makes it available to the

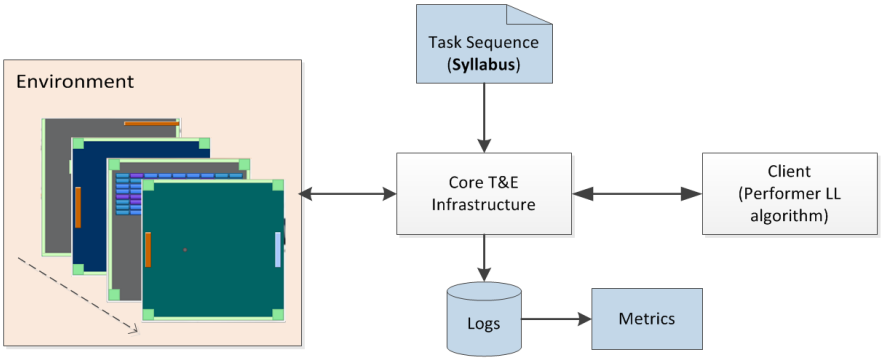


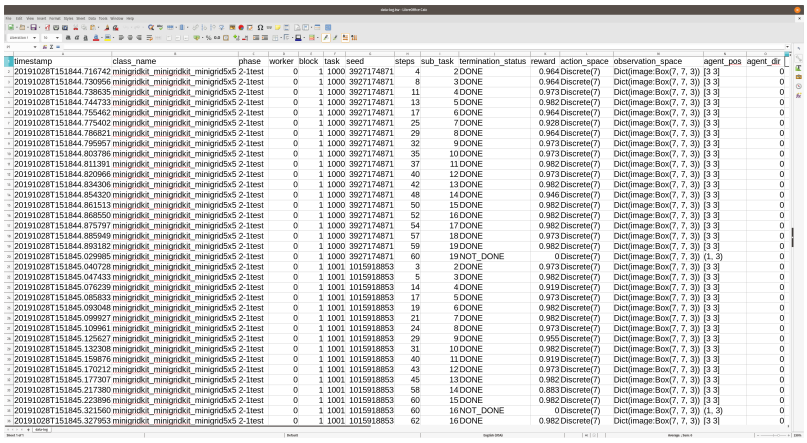
Figure 2.1: This figure describes the production of Metrics from Log files outputted by the Core Test and Evaluation Framework.

Figure 2.2 shows an example syllabus which would evaluate the Continual Learning Core Capability. You can see that only one task is exercised throughout the syllabus, but parameter variation - agent_pos and agent_dir are paramters for this task - is contained throughout the syllabus. Additionally, later Test phases contain previously trained parameters, allowing for a maintenance calculation. Unless otherwise noted, an instance of a syllabus is considered a Lifetime, and metrics are computed on an Agent's performance on episodes within a syllabus.

```
{
   "name": "Test syllabus with MinigridKit episodes",
   "author": "Megan Baker <megan.baker@jhuapl.edu>",
   "contains": "episodes",
   "default_params": {},
   "instructions": [
     {"$phase":  "1.train"},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,1]}, "count": 1000},

     {"$phase":  "1.test"},
     {"$info":  {"disable_updates":  true}},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,1]}, "count": 100},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,3], "agent_dir":2}, "count": 100},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [3,2], "agent_dir":1}, "count": 100},
     {"$info":  {}},

     {"$phase":  "2.train"},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,3], "agent_dir":2}, "count": 1000},

     {"$phase":  "2.test"},
     {"$info":  {"disable_updates":  true}},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,1]}, "count": 100},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,3], "agent_dir":2}, "count": 100},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [3,2], "agent_dir":1}, "count": 100},
     {"$info":  {}},

     {"$phase":  "3.train"},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [3,2], "agent_dir":1}, "count": 1000},

     {"$phase":  "3.test"},
     {"$info":  {"disable_updates":  true}},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,1]}, "count": 100},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [1,3], "agent_dir":2}, "count": 100},
     {"$repeat": { "$episode": "$minigridkit:Minigrid5x5", "agent_pos": [3,2], "agent_dir":1}, "count": 100},
     {"$info":  {}}
   ]
}
```

Figure 2.2: An example syllabus.

When an Agent performs episodes from the syllabus, the TEF will automatically generate logs which are saved in a tab separated file like shown in Figure 2.3. Then, the Metrics code scrapes these logs and extracts the relevant information to produce a set of scores for each metric used to evaluate the Core Capabilty being tested.



timestamp	class_name	phase	worker	block	task	seed	steps	sub_task	termination_status	reward	action_space	observation_space	agent_pos	agent_dir
20191028T151844.716742	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	4	2	DONE	0.964	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.730956	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	8	3	DONE	0.964	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.738635	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	11	4	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.744733	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	13	5	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.755462	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	17	6	DONE	0.964	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.775402	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	25	7	DONE	0.928	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.786821	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	29	8	DONE	0.964	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.795957	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	32	9	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.803786	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	35	10	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.811391	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	37	11	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.820966	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	40	12	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.834306	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	42	13	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.854320	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	48	14	DONE	0.946	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.861513	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	50	15	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.868550	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	52	16	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.875797	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	54	17	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.885949	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	57	18	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151844.893182	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	59	19	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.029985	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1000	3927174871	60	19	NOT_DONE	0	Discrete(7)	Dict(image:Box(7, 7, 3))	(1, 3)	0	
20191028T151845.040728	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	3	2	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.047433	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	5	3	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.076239	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	14	4	DONE	0.919	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.085833	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	17	5	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.093048	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	19	6	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.099927	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	21	7	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.109961	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	24	8	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.125627	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	29	9	DONE	0.955	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.132308	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	31	10	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.159876	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	40	11	DONE	0.919	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.170212	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	43	12	DONE	0.973	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.177307	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	45	13	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.217380	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	58	14	DONE	0.883	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.223896	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	60	15	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	
20191028T151845.321560	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	60	16	NOT_DONE	0	Discrete(7)	Dict(image:Box(7, 7, 3))	(1, 3)	0	
20191028T151845.327953	minigridkit_minigridkit_minigridx5 x 2-1test	0	1	1001	1015918853	62	16	DONE	0.982	Discrete(7)	Dict(image:Box(7, 7, 3))	[3 3]	0	

Figure 2.3: An example log file.

Chapter 3

Syllabus Types

With the Core Capabilties in mind, we move on to intentional design of the syllabi. As a reminder, the calculation of these metrics seeks to answer a question that may differ from traditional reinforcement learning evaluation criteria, and thus, we enforce a semi-rigid structure to these evaluation tasks.

3.1 Syllabus Types by Core Capability

Continual Learning

Syllabus Structure Requirements

---

1. This syllabus consists of a single task with parametric variations
Can involve interpolation of parameters, noisy parameters, etc. but has only one task

2. Each phase consists of a training and optional but recommended evaluation block
Training block is training on one or more parametric variations

---

Metric Evaluation Questions

Note that some of these are computed as a basis for comparison for future eval blocks, etc.

Question: What level of performance does the agent achieve during training?
Metric: Saturation Value

Question: How quickly does it achieve this saturation value?
Metric: Time to Saturation

Question: How quickly does an agent learn during training?
Metric: Normalized Integral of Reward/Time

Question: Can the agent adjust to changes in the environment? How quickly does it recover?
Metric: Recovery Time

Question: Can the agent maintain performance on previously learned parameters after being trained with new ones?
Metric: Performance Maintenance

Question: How does the lifelong learning agent's performance compare to a traditional agent?
Metric: Performance relative to STE

After these metrics are computed, the agent will receive a score for each of these metrics for a Continual Learning syllabus, which will be aggregated to form its Continual Learning score. For now, an average of the scores achieved is reported in each phase/block, but more sophisticated methods may replace these in the future. See below for more details; more information is on its way.

Formal Metric Definitions

1. Saturation Value
Purpose: The saturation value is computed to quantify the maximum maintained value by the agent.
Calculated by: Since multiple rewards may be logged per episode, the mean is taken per episode. Then, the max of the rolling average is taken of the mean reward per episode with a smoothing parameter, s (default, 0.1)
Compared to: Future or single task expert saturation values of the same task

2. Time to Saturation
Purpose: Time to saturation is used to quantify how quickly, in number of episodes, the agent took to achieve the saturation value computed above.
Calculated by: The first time the saturation value (or above) is seen, that episode number is recorded
Compared to: Future times to saturation of the same task

3. Normalized Integral of Reward/Time
Purpose: Taking the Integral of Accumulated Reward over Time allows for a more robust comparison of the time to learn a particular task, taking into account both the shape and saturation of the learning for future comparison. Has limitations; must be normalized by length; only training phases can be compared to each other
Calculated by: Integrating reward over time, then dividing by the number of episodes used to accumulate the reward
Compared to: Future training instances of this metric

4. Recovery Time
Purpose: Recovery time is calculated to determine how quickly (if at all) an agent can "bounce back" after a change is introduced to its environment
Calculated by: After some training phase achieves a saturation value, determine how many episodes, if any, it takes for the agent to regain the same performance on the same task
Compared to: An agent's recovery time is comparable across tasks

5. Performance Maintenance on Test Sets
Purpose: Performance maintenance on test sets is calculated to determine whether an agent catastrophically forgets a previously learned task
Calculated by: Comparing all computed metrics on the train set (saturation values, time to saturation, etc) on the test set and computing the difference in performance

6. Performance relative to STE (training)
Purpose: STE Relative performance assesses whether a lifelong learner outperforms a traditional learner.
Calculated by: Normalizing metrics computed on the lifelong learner by the same metrics computed on the traditional learner

Chapter 4

Metrics Code

4.1 Code Release

L2Metrics code is being released to GitHub! Download it here: *Put a link here*

4.1.1 Overview

In l2metrics/core.py we introduce an abstract Metric class which describes the general format for any Metric you may use or write, whether for Agent or Classification Learners. The most relevant piece of the Metric class is the calculate method, which has the following required arguments: the log data, a phase_info dataframe, and a metrics_dict dataframe. The metrics_dict starts blank and is filled by each metric in its turn, whereas the log data and the phase info are extracted in the MetricsReport constructor from the logs via two helper functions:

l2metrics/util.py - (read_log_data): scrapes the logs and returns a pandas dataframe of the logs and task parameters
l2metrics/_localutil.py - (parse_blocks): builds a pandas dataframe of the phase information contained in the log data

These dataframes are passed along by the MetricsReport to the appropriate metric and thus the Metrics and MetricsReport classes should be utilized in conjuction with each other. Though there are a list of default metrics which the MetricsReport uses for the Core Capabilties being exercised at this time, you may choose to add your own metric to this list by using the add method on MetricsReport. Please see the calc_metrics.py file for more details on how to get started with writing your own custom metric. An extremely simple whole-syllabus-mean is currently implemented as an example to help get you started.

Syllabus and Log Files - Assumptions and Requirements

Please note: Log files should be generated automatically and should require no action on the performer's part. However, Single Task Expert JSON files and/or custom syllabi creation rules must be followed or the L2Metrics code may not work.

1. Log files **must** include logged reward, and the column in the data file **must** be named reward. Without this column, the metrics code will fail. This is assumed to be logged per episode.

2. Single Task Expert Saturation values for each task **must** be included in a JSON file found in $L2DATA/taskinfo/info.json and without this file, the metric "Comparison to STE" cannot be calculated. Further, the task names contained in the JSON file must match the names in the log files exactly. The format for this file will be:

{
"task_name_1" : 0.8746,
"task_name_2" : 0.9315,
...,
"task_name_n" : 0.8089
}

3. Syllabi used to generate the log files **must** include annotations with phase information and shall conform to the below convention. Please see Figure 2.2 for an example

Phase annotation format:

{"$phase": "1.train"}, {"$phase": "1.test"}, {"$phase": "2.train"}, {"$phase": "2.test"}, etc

Structure - Continual Learning:

Consists **only** of a single task with parametric variations exercised throughout the syllabus. Testing phase is optional, but recommended. The purpose of this type of syllabus is to assess whether the agent can adjust to changes in the environment and maintain performance on the previous parameters when new ones are introduced

Getting Started

Get started by first generating some log files. You may do this by the following:
1. Download and install the minigridkit repo, located here: *PUT A LINK HERE*
2. Configure your environment variable $L2DATA to wherever you want your logs to end up.
3. Run minigrid_learnkit/minigrid_train_ppo.py
4. Your logs should appear in $L2DATA\logs\"YOUR_LOG_DIRECTORY"

Then, you should be able to:

5. Pass "YOUR_LOG_DIRECTORY" as the log_dir parameter in the calc_metrics.py file, and you should get an output printed to console that looks something like this:

```
Metric: Average Within Block Saturation Calculation
Value: {'global_within_block_saturation': 0.6201868186374054, 'global_num_eps_to_saturation': 15.833333333333334}

Metric:
Value: {'global_perf': 0.9247070921895495}

Process finished with exit code 0
```

Figure 4.1: Sample output for the example calc_metrics.py .