Preprocessing

Often it is necessary to modify state input tensors before passing them to the reinforcement learning agent. This could be due to various reasons, e.g.:

- Feature scaling / input normalization,
- Data reduction,
- Ensuring the Markov property by concatenating multiple states (e.g. in Atari)

TensorForce comes with a number of ready-to-use preprocessors, a preprocessing stack and easy ways to implement your own preprocessors.

Usage

The

Each preprocessor implements three methods:

1. The constructor (`__init__`) for parameter initialization
2. `process(state)` takes a state and returns the processed state
3. `processed_shape(original_shape)` takes a shape and returns the processed shape

The preprocessing stack iteratively calls these functions of all preprocessors in the stack and returns the result.

Using one preprocessor

```
from tensorforce.core.preprocessing import Sequence

pp_seq = Sequence(4)  # initialize preprocessor (return sequence of last 4 states)

state = env.reset()  # reset environment
processed_state = pp_seq.process(state)  # process state
```

Using a preprocessing stack

You can stack multipe preprocessors:

```
from tensorforce.core.preprocessing import Preprocessing, Grayscale, Sequence

pp_gray = Grayscale()  # initialize grayscale preprocessor
pp_seq = Sequence(4)  # initialize sequence preprocessor

stack = Preprocessing()  # initialize preprocessing stack
stack.add(pp_gray)  # add grayscale preprocessor to stack
stack.add(pp_seq)  # add maximum preprocessor to stack

state = env.reset()  # reset environment
processed_state = stack.process(state)  # process state
```

Using a configuration dict

If you use configuration objects, you can build your preprocessing stack from a config:

```
from tensorforce.core.preprocessing import Preprocessing

preprocessing_config = [
    {
        "type": "image_resize",
        "kwargs": {
            "width": 84,
            "height": 84
        }
    }, {
        "type": "grayscale"
    }, {
        "type": "center"
    }, {
        "type": "sequence",
        "kwargs": {
            "length": 4
        }
    }
]

stack = Preprocessing.from_config(preprocessing_config)
config.state_shape = stack.shape(config.state_shape)
```

The `Agent` class expects a *preprocessing* configuration parameter and then handles preprocessing automatically:

```
from tensorforce.agents import DQNAgent

agent = DQNAgent(config=dict(
    states=...,
    actions=...,
    preprocessing=preprocessing_config,
    # ...
))
```

Ready-to-use preprocessors

These are the preprocessors that come with TensorForce:

Center

class **`tensorforce.core.preprocessing.Center`**

Bases: `tensorforce.core.preprocessing.preprocessor.Preprocessor`

Center/standardize state. Subtract minimal value and divide by range.

Grayscale

class **`tensorforce.core.preprocessing.Grayscale`**(*weights=(0.299, 0.587, 0.114)*)

Bases: `tensorforce.core.preprocessing.preprocessor.Preprocessor`

Turn 3D color state into grayscale.

ImageResize

class **`tensorforce.core.preprocessing.ImageResize`**(*width, height*)

Bases: `tensorforce.core.preprocessing.preprocessor.Preprocessor`

Resize image to width x height.

Normalize

class **`tensorforce.core.preprocessing.Normalize`**

Bases: `tensorforce.core.preprocessing.preprocessor.Preprocessor`

Normalize state. Subtract mean and divide by standard deviation.

Sequence

class **`tensorforce.core.preprocessing.Sequence`**(*length=2*)

Bases: `tensorforce.core.preprocessing.preprocessor.Preprocessor`

Concatenate `length` state vectors. Example: Used in Atari problems to create the Markov property.

Building your own preprocessor

All preprocessors should inherit from `tensorforce.core.preprocessing.Preprocessor`.

For a start, please refer to the source of the Grayscale preprocessor.