openai / gym

Branch: master gym / examples / agents / cem.py Find file Copy path

futurely Fix deprecated Monitor API for the cem example (#453) bb9bbce on 2 Jan

6 contributors

102 lines (86 sloc) 3.61 KB

```python
from __future__ import print_function

import gym
from gym import wrappers
import logging
import numpy as np
try:
    import cPickle as pickle
except ImportError:
    import pickle
import json, sys, os
from os import path
from _policies import BinaryActionLinearPolicy # Different file so it can be unpickled
import argparse

def cem(f, th_mean, batch_size, n_iter, elite_frac, initial_std=1.0):
    """
    Generic implementation of the cross-entropy method for maximizing a black-box function

    f: a function mapping from vector -> scalar
    th_mean: initial mean over input distribution
    batch_size: number of samples of theta to evaluate per batch
    n_iter: number of batches
    elite_frac: each batch, select this fraction of the top-performing samples
    initial_std: initial standard deviation over parameter vectors
    """
    n_elite = int(np.round(batch_size*elite_frac))
    th_std = np.ones_like(th_mean) * initial_std

    for _ in range(n_iter):
        ths = np.array([th_mean + dth for dth in  th_std[None,:]*np.random.randn(batch_size, th_mean.size)])
        ys = np.array([f(th) for th in ths])
        elite_inds = ys.argsort()[::-1][:n_elite]
        elite_ths = ths[elite_inds]
        th_mean = elite_ths.mean(axis=0)
        th_std = elite_ths.std(axis=0)
        yield {'ys' : ys, 'theta_mean' : th_mean, 'y_mean' : ys.mean()}

def do_rollout(agent, env, num_steps, render=False):
    total_rew = 0
    ob = env.reset()
    for t in range(num_steps):
        a = agent.act(ob)
        (ob, reward, done, _info) = env.step(a)
        total_rew += reward
        if render and t%3==0: env.render()
        if done: break
    return total_rew, t+1

if __name__ == '__main__':
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)

    parser = argparse.ArgumentParser()
    parser.add_argument('--display', action='store_true')
    parser.add_argument('target', nargs="?", default="CartPole-v0")
    args = parser.parse_args()
```

```python

    env = gym.make(args.target)
    env.seed(0)
    np.random.seed(0)
    params = dict(n_iter=10, batch_size=25, elite_frac = 0.2)
    num_steps = 200

    # You provide the directory to write to (can be an existing
    # directory, but can't contain previous monitor results. You can
    # also dump to a tempdir if you'd like: tempfile.mkdtemp().
    outdir = '/tmp/cem-agent-results'
    env = wrappers.Monitor(env, outdir, force=True)

    # Prepare snapshotting
    # ----------------------------------------
    def writefile(fname, s):
        with open(path.join(outdir, fname), 'w') as fh: fh.write(s)
    info = {}
    info['params'] = params
    info['argv'] = sys.argv
    info['env_id'] = env.spec.id
    # ------------------------------------------

    def noisy_evaluation(theta):
        agent = BinaryActionLinearPolicy(theta)
        rew, T = do_rollout(agent, env, num_steps)
        return rew

    # Train the agent, and snapshot each stage
    for (i, iterdata) in enumerate(
        cem(noisy_evaluation, np.zeros(env.observation_space.shape[0]+1), **params)):
        print('Iteration %2i. Episode mean reward: %7.3f'%(i, iterdata['y_mean']))
        agent = BinaryActionLinearPolicy(iterdata['theta_mean'])
        if args.display: do_rollout(agent, env, 200, render=True)
        writefile('agent-%.4i.pkl'%i, str(pickle.dumps(agent, -1)))

    # Write out the env at the end so we store the parameters of this
    # environment.
    writefile('info.json', json.dumps(info))

    env.close()

    logger.info("Successfully ran cross-entropy method. Now trying to upload results to the scoreboard. If it breaks, you can al
    gym.upload(outdir)
```