

# CPSC532W: Probabilistic Programming, Homework 5

Namrata Dekka

Code for this assignment can be found here: <https://github.com/namratadeka/cpsc532W/tree/main/CS532-HW5>.

```
class Env(dict):
    def __init__(self, params=(), args=(), outer=None):
        self.update(zip(params, args))
        self.outer = outer
    def get(self, var):
        return self[var] if (var in self) else self.outer.get(var)

class Procedure(object):
    '''A user-defined function.'''
    def __init__(self, params, body, env):
        self.params, self.body, self.env = params, body, env
    def call(self, *args):
        return evaluate(self.body, Env(self.params, args, self.env))

def standard_env():
    "An environment with some Scheme standard procedures."
    env = pmap(penv)
    env = env.update({'alpha' : ''})

    return env

def evaluate(exp, env=None): #TODO: add sigma, or something
    if env is None:
        env = standard_env()

    if isinstance(exp, str):
        if env.get(exp) is not None:
            return env.get(exp)
        return exp
    elif not isinstance(exp, list):
        return torch.tensor(exp).float()
    op, *args = exp
    if op == 'if':
        (test, conseq, alt) = args
        exp = (conseq if evaluate(test, env) else alt)
        return evaluate(exp, env)
    elif op == 'sample':
        evaluate(args[0], env)
        dist = evaluate(args[1], env)
        return dist.sample()
    elif op == 'observe':
        evaluate(args[0], env)
        dist = evaluate(args[1], env)
        obs = evaluate(args[2], env)
        return obs
    elif op == 'fn':
        params, body = args
        return Procedure(params, body, env)
    else:
        proc = evaluate(op, env)
        vals = [evaluate(arg, env) for arg in args]
        return proc(*vals)

return
```

Figure 1: The HOPPL evaluator code.

## 1 Program 1 (Test cases)

All test cases passed.

```
('normal', 5, 1.4142136)
p value 0.29122091125030425
('beta', 2.0, 5.0)
p value 0.8637021996153093
('exponential', 0.0, 5.0)
p value 0.03243910141859374
('normal', 5.3, 3.2)
p value 0.9875058073791319
/home/namrata/projects/cpsc532W/CS532-HW5/primitives.py:
, it is recommended to use sourceTensor.clone().detach()
True), rather than torch.tensor(sourceTensor).
'sqrt': lambda alpha, x: torch.sqrt(torch.tensor(x)),
('normalmix', 0.1, -1, 0.3, 0.9, 1, 0.3)
p value 0.7932149371520858
('normal', 0, 1.44)
p value 0.438355243966025
All probabilistic tests passed
```

Figure 2: Screenshot showing passing all probabilistic tests.

```
FOPPL Tests passed
/home/namrata/projects/cpsc532w
, it is recommended to use sour
True), rather than torch.tensor
'sqrt': lambda alpha, x: torc
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
All deterministic tests passed
```

Fig

Figure 3: Screenshot showing passing all deterministic tests.

## 2 Program 2

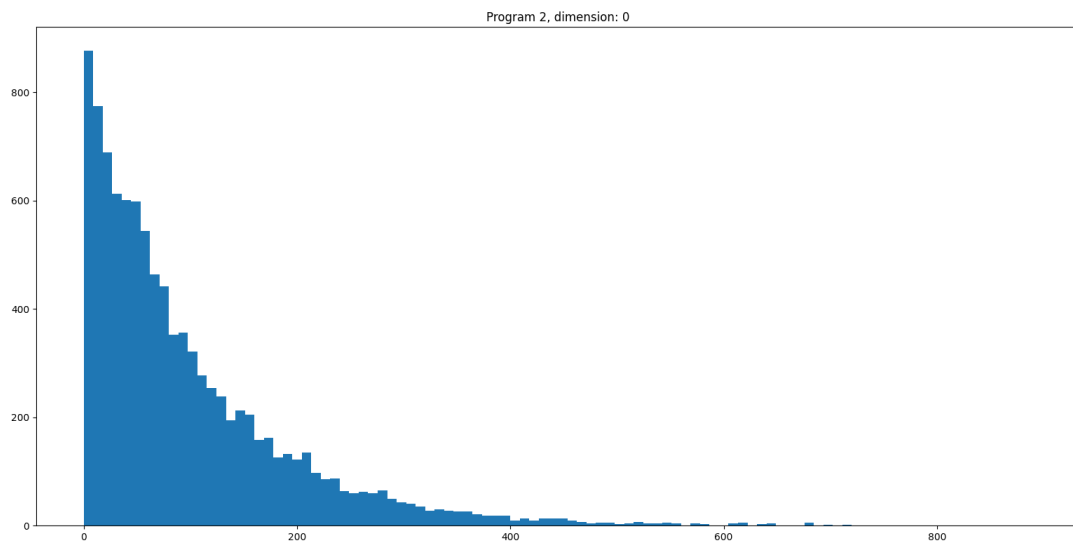


Figure 4: Prior histogram for Program 2. Mean=98.4549, Variance=9513.9033

## 3 Program 3

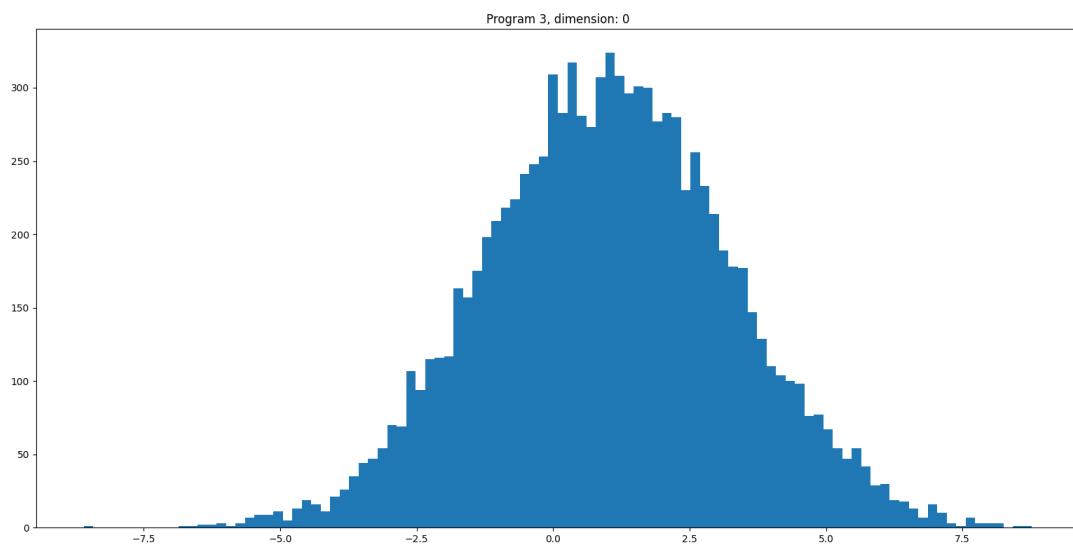


Figure 5: Prior histogram for Program 3. Mean=1.0260, Variance=5.0382

## 4 Program 4



Figure 6: Heatmaps for prior means and variances for Program 4.