loop_bypass

October 13, 2021

1 Supplementary Material to:

2 Blueprint for Next Generation Cyber-Physical Resilience using Defense Quantum Machine - Learning Two-train Example

3 Authors

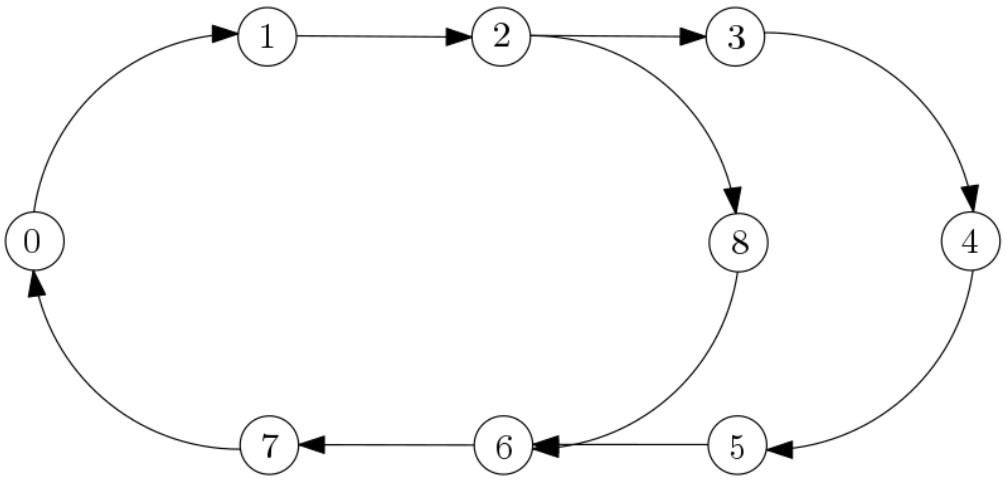
Michel Barbeau
Joaquin Garcia-Alfaro
Version: September 12, 2021
We reused code from: Basic tutorial: qubit rotation
Note: This example runs of version 0.8.0 of Pennylane, which is not the *defaul latest version*. To force installation of that version, in a Linux shell enter:
python -m pip install pennylane==0.8.0

3.1 Two-train scenario

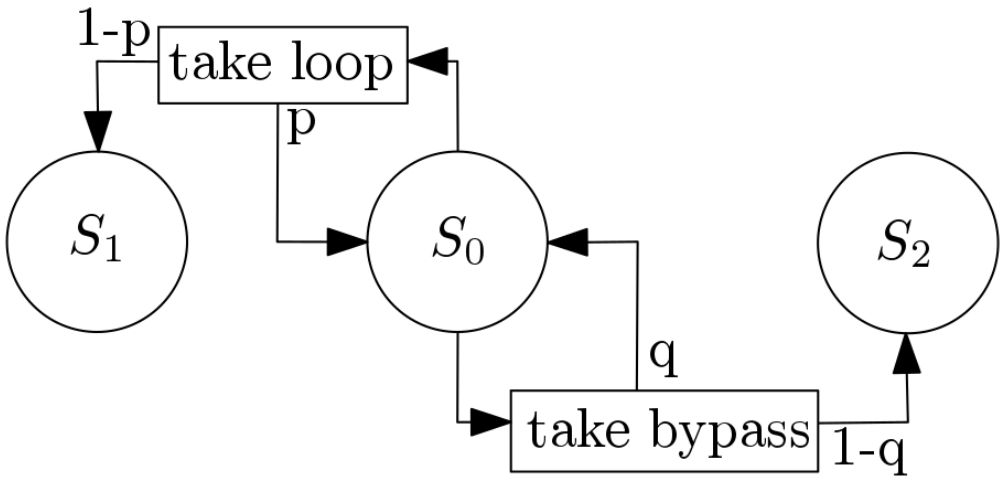
Let us consider the following discrete two-train scenario:

Tracks are broken into sections. Let us assume a scenario where Train 1 is the agent and Train 2 is part of its world. There is an outer loop visiting points 3, 4 and 5, together with a bypass from point 2, visiting point 8 to point 6. Traversal time is uniform across sections. The normal trajectory of Train 1 is the outer loop, while maintaining a Train 2-Train 1 distance greater than one empty section. For example, if Train 1 is at point 0 while Train 2 is at point 7, then the separation distance constraint is violated. The goal of the adversary is to steer the system in a state where the separation distance constraint is violated. When a train crosses point 0, it has to make a choice: either traverse the outer loop or take the bypass. Both trains can follow any path and make independent choices, when they are at point 0.

In the terms of reinforcement learning, Train 1 has two actions available: take loop and take bypass. The agent gets k reward points for a relative Train 2-Train 1 distance increase of k sections with Train 2. It gets $-k$ reward points, i.e., a penalty, for a relative Train 2-Train 1 distance decrease of k sections with Train 2. For example, let us assume that Train 1 is at point 0 and that Train 2 is at point 7. If both trains, progressing a the same speed, take the loop or both decide to take the bypass, then there is no relative distance change. The agent gets no reward. When Train 1 decides to take the bypass and Train 2 decides to take the loop, the agent gets two reward points, at return to point zero (Train 2 is at point five). When Train 1 decides to take the loop and Train 2 decides to



track



track

take the bypass, the agent gets four reward points, at return to point zero (Train 2 is at point one, Train 2-Train 1 distance is five sections).

The representation of the environment is as follows:

The state set is $S = \{0, 1, 2\}$. The action set is $A = \{a_0 = \text{take loop}, a_1 = \text{take bypass}\}$ The transition probability function is defined as $P_{a_0}(0,0) = p$, $P_{a_0}(0,1) = 1 - p$, $P_{a_1}(0,0) = q$ and $P_{a_1}(0,2) = 1 - q$. The reward functions is defined as $R_{a_0}(0,0) = 0$, $R_{a_0}(0,1) = 4$, $R_{a_1}(0,0) = 0$ and $R_{a_1}(0,2) = 2$. This is interpreted as follows. In the initial state 0 with a one-section separation distance, the agent selects an action to perform: take loop or take bypass. Train 1 performs the selected action. When selecting take loop, with probability p the environment goes back to state 0 (no reward) or with probability $1 - p$ it moves to state 1, with a five-section separation distance (reward is four). When selecting take bypass, with probability q the environment goes back to state 0 (no reward) or with probability $1 - q$ it moves state 2, with a three-section separation distance (reward is two). The agent memorizes how good it has been to perform a selected action.

In the sequel, the environment probabilities of zero reward p and q are assumed to be the same.

3.2 PennyLan environment import

```
In [2]: import pennylane as qml
        from pennylane import numpy as np
        import matplotlib.pyplot as plt
```

3.3 Device creation

```
In [3]: dev1 = qml.device("default.qubit", wires=1, shots=1)
```

3.4 Quantum node construction

A variational circuit `W(theta)` is trained, with parameter `theta`. The circuit consists of two gates: an X gate and an Y gate. In this example, there is only state S_0. It is represented by the quantum state $|0\rangle$. Because it is a ground state, the state is not coded explicitly at the input of the circuit. Parameter `theta` is an array of two rotation angles, one for every gate.

```
In [4]: @qml.qnode(dev1)
        def W(theta):
            qml.RX(theta[0], wires=0)
            qml.RY(theta[1], wires=0)
            # return probabilities of computational basis states
            return qml.probs(wires=[0])
```

3.5 Cost model

The actions take loop and take bypass are respectively represented by the quantum states $|0\rangle$ and $|1\rangle$. The variational circuit is trained on the probability of each computational basis state: $|0\rangle$ and $|1\rangle$.

The `cost` function measures the difference between the probablities associated to the variational circuit `W(theta)` and the target probabilities of the quantum states $|0\rangle$ and $|1\rangle$.

```
In [5]: def square_loss(X, Y):
            loss = 0
            for x, y in zip(X, Y):
                loss = loss + (x - y) ** 2
            return loss / len(X)

        def cost(theta, p):
            #  p is prob. of ket zero, 1-p is prob of ket 1
            return square_loss(W(theta), [p, 1-p])
```

3.6 Quantum circuit update

Using optimization, the following code finds the `theta` such that the variational circuit `W(theta)` outputs a quantum state where the probabilities $|0\rangle$ and $|1\rangle$ are respectively p and $1 - p$.

```
In [6]: # initialise the optimizer
        opt = qml.GradientDescentOptimizer(stepsize=0.4)
```

```
        # set the number of steps
        steps = 10
        # set the initial theta value of variational circuit state
        theta = np.random.randn(2)
        print("Initial probs. of basis states: {}".format(W(theta)))

        def update(theta, p):
            # p = probability
            for i in range(steps):
                # update the circuit parameters
                theta = opt.step(lambda v: cost(v, p), theta)
                #if (i + 1) % 10 == 0:
                    #print("Cost @ step {:5d}: {: .7f}".format(i, cost(theta,p)))
            #print("Probs. of basis states: {}".format(W(theta)))
            return theta
```

```
Initial probs. of basis states: [0.61432696 0.38567304]
```

3.7 Reinforcement learning process

Q Learning Reinforcement Learning is used.

Probabilities of quantum states $|0\rangle$ and $|1\rangle$ is proportional to the respective rewards `Q[0]/(Q[0]+Q[1])` and `1 - Q[0]/(Q[0]+Q[1])`.

```
In [45]: # init theta to random values
         theta = np.random.randn(2)
         # number of iterations
         epochs = 100
         # measurement results (probs. of ket zero and ket one)
         M = np.zeros(epochs)
         # environment probability
         p = 0.9 # adversary likely to do the same as agent
         p = 0.0 # adversary unlikely to do the same as agent
         # rewards
         R = [4, 2]
         # Q-value (total reward associated with actions)
         Q = [0, 0]
         # learning rate
         alpha = 0.01
         # main loop
         for i in range(epochs):
             # agent has a random behavior, select random action
             a = np.random.randint(2, size=1)[0]
             # agent has a pattern behavior, most likely take the loop
             #a = 0;
             #num = np.random.random(1)[0]
             #if num > 0.9:
```

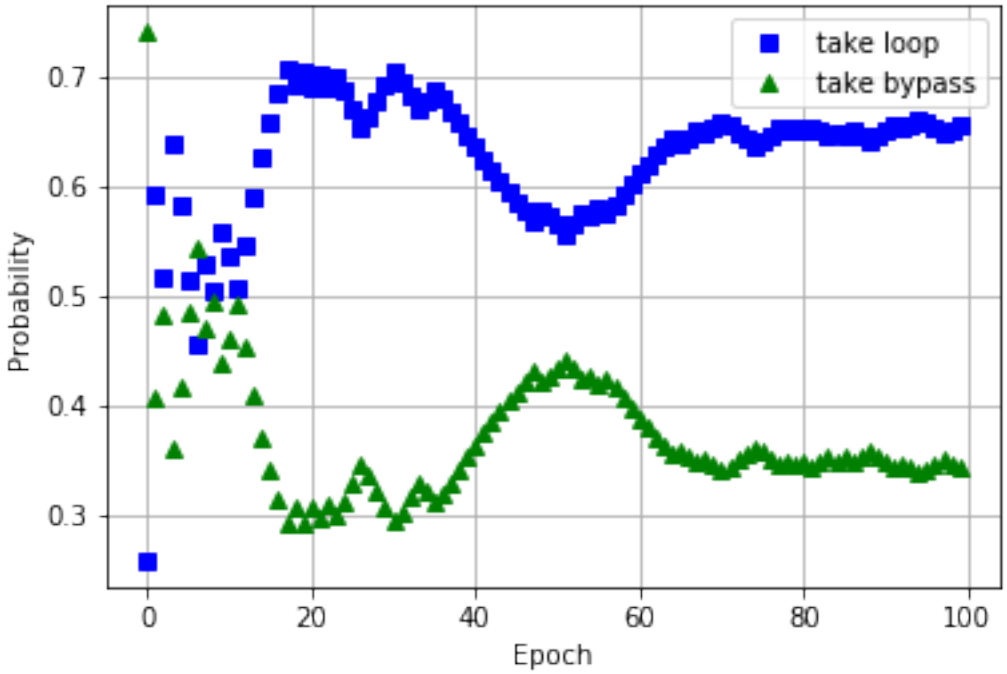
```
            #    r = 1;
            # agent has a pattern behavior, most likely take the bypass
            #a = 1;
            #num = np.random.random(1)[0]
            #if num > 0.9:
            #    r = 0;
            # determine reward
            num = np.random.random(1)[0]
            r = 0
            if num > p:
                r = R[a]
            # Bellman equation
            Q[a] = (1-alpha)*Q[a] + alpha*(r + Q[a])
            print("i: ", i," num: ", num," action: ", a, " reward: ", r, " Q-values: ", Q)
            if (Q[0]+Q[1]) > 0:
                theta = update(theta, Q[0]/(Q[0]+Q[1]))
            M[i] = W(theta)[0]
```

```
i:  0  num:  0.7660289091569452  action:  1  reward:  2  Q-values:  [0, 0.02]
i:  1  num:  0.6615408941707992  action:  0  reward:  4  Q-values:  [0.04, 0.02]
i:  2  num:  0.2466746113257663  action:  1  reward:  2  Q-values:  [0.04, 0.04]
i:  3  num:  0.7363505370675982  action:  0  reward:  4  Q-values:  [0.08, 0.04]
i:  4  num:  0.6986184730653854  action:  1  reward:  2  Q-values:  [0.08, 0.060000000000000005
i:  5  num:  0.6909053769921761  action:  1  reward:  2  Q-values:  [0.08, 0.08]
i:  6  num:  0.07019156256758963  action:  1  reward:  2  Q-values:  [0.08, 0.1]
i:  7  num:  0.752087680104352  action:  0  reward:  4  Q-values:  [0.12000000000000001, 0.1]
i:  8  num:  0.4535136016360577  action:  1  reward:  2  Q-values:  [0.12000000000000001, 0.120
i:  9  num:  0.1776527591072955  action:  0  reward:  4  Q-values:  [0.16, 0.12000000000000001]
i:  10  num:  0.07769326466333237  action:  1  reward:  2  Q-values:  [0.16, 0.14]
i:  11  num:  0.11089427642296668  action:  1  reward:  2  Q-values:  [0.16, 0.16]
i:  12  num:  0.9681559764460075  action:  0  reward:  4  Q-values:  [0.2, 0.16]
i:  13  num:  0.1940035265691964  action:  0  reward:  4  Q-values:  [0.24000000000000002, 0.16
i:  14  num:  0.881540467506921  action:  0  reward:  4  Q-values:  [0.28, 0.16]
i:  15  num:  0.19500779011928748  action:  0  reward:  4  Q-values:  [0.32, 0.16]
i:  16  num:  0.7229852114745271  action:  0  reward:  4  Q-values:  [0.36000000000000004, 0.16
i:  17  num:  0.5250627871445339  action:  0  reward:  4  Q-values:  [0.4000000000000001, 0.16]
i:  18  num:  0.7695918002398373  action:  1  reward:  2  Q-values:  [0.4000000000000001, 0.180
i:  19  num:  0.7795873838121119  action:  0  reward:  4  Q-values:  [0.44000000000000006, 0.18
i:  20  num:  0.9953744482751442  action:  1  reward:  2  Q-values:  [0.44000000000000006, 0.20
i:  21  num:  0.7395567172871411  action:  0  reward:  4  Q-values:  [0.48000000000000004, 0.20
i:  22  num:  0.9032454349134023  action:  1  reward:  2  Q-values:  [0.48000000000000004, 0.22
i:  23  num:  0.44527439084969633  action:  0  reward:  4  Q-values:  [0.52, 0.22000000000000000
i:  24  num:  0.24578654025092617  action:  1  reward:  2  Q-values:  [0.52, 0.24000000000000000
i:  25  num:  0.5687397358360422  action:  1  reward:  2  Q-values:  [0.52, 0.26]
i:  26  num:  0.11863508330902517  action:  1  reward:  2  Q-values:  [0.52, 0.28]
i:  27  num:  0.4942915355813542  action:  0  reward:  4  Q-values:  [0.56, 0.28]
i:  28  num:  0.880540094925226  action:  0  reward:  4  Q-values:  [0.6, 0.28]
i:  29  num:  0.49483148397180265  action:  0  reward:  4  Q-values:  [0.64, 0.28]
```

```
i:  30  num:  0.7947119478394646  action:  0  reward:  4  Q-values:  [0.68, 0.28]
i:  31  num:  0.7199012608846255  action:  1  reward:  2  Q-values:  [0.68, 0.3]
i:  32  num:  0.9188622970277083  action:  1  reward:  2  Q-values:  [0.68, 0.32]
i:  33  num:  0.7448745137387217  action:  1  reward:  2  Q-values:  [0.68, 0.34]
i:  34  num:  0.5100084527755381  action:  0  reward:  4  Q-values:  [0.72, 0.34]
i:  35  num:  0.8896099904852536  action:  0  reward:  4  Q-values:  [0.76, 0.34]
i:  36  num:  0.3488007465251235  action:  1  reward:  2  Q-values:  [0.76, 0.36]
i:  37  num:  0.9957460062669848  action:  1  reward:  2  Q-values:  [0.76, 0.38]
i:  38  num:  0.16834889087575922  action:  1  reward:  2  Q-values:  [0.76, 0.3999999999999999
i:  39  num:  0.5617223863024151  action:  1  reward:  2  Q-values:  [0.76, 0.42]
i:  40  num:  0.8183434641496578  action:  1  reward:  2  Q-values:  [0.76, 0.44]
i:  41  num:  0.6091995224201235  action:  1  reward:  2  Q-values:  [0.76, 0.4599999999999999
i:  42  num:  0.04571468536432277  action:  1  reward:  2  Q-values:  [0.76, 0.48]
i:  43  num:  0.8139599985937443  action:  1  reward:  2  Q-values:  [0.76, 0.4999999999999994
i:  44  num:  0.403117445627298  action:  1  reward:  2  Q-values:  [0.76, 0.5199999999999999]
i:  45  num:  0.852164256524478  action:  1  reward:  2  Q-values:  [0.76, 0.5399999999999999]
i:  46  num:  0.37216436017098775  action:  1  reward:  2  Q-values:  [0.76, 0.5599999999999999
i:  47  num:  0.2104095622191543  action:  1  reward:  2  Q-values:  [0.76, 0.5799999999999998]
i:  48  num:  0.41908827371443547  action:  0  reward:  4  Q-values:  [0.7999999999999999, 0.57
i:  49  num:  0.029388194866140638  action:  1  reward:  2  Q-values:  [0.7999999999999999, 0.5
i:  50  num:  0.22159132346055788  action:  1  reward:  2  Q-values:  [0.7999999999999999, 0.6
i:  51  num:  0.4445236379928028  action:  1  reward:  2  Q-values:  [0.7999999999999999, 0.639
i:  52  num:  0.2278233955250354  action:  0  reward:  4  Q-values:  [0.84, 0.6399999999999999]
i:  53  num:  0.1866064442745763  action:  0  reward:  4  Q-values:  [0.88, 0.6399999999999999]
i:  54  num:  0.2736289876736925  action:  1  reward:  2  Q-values:  [0.88, 0.6599999999999999]
i:  55  num:  0.7429044583101132  action:  0  reward:  4  Q-values:  [0.9199999999999999, 0.659
i:  56  num:  0.9374858365419503  action:  1  reward:  2  Q-values:  [0.9199999999999999, 0.679
i:  57  num:  0.8246079172295852  action:  0  reward:  4  Q-values:  [0.96, 0.6799999999999998]
i:  58  num:  0.8212569505321575  action:  0  reward:  4  Q-values:  [0.9999999999999999, 0.679
i:  59  num:  0.8370799375318702  action:  0  reward:  4  Q-values:  [1.0399999999999998, 0.679
i:  60  num:  0.6169843761587583  action:  0  reward:  4  Q-values:  [1.0799999999999998, 0.679
i:  61  num:  0.17410023627042748  action:  0  reward:  4  Q-values:  [1.1199999999999999, 0.67
i:  62  num:  0.7716887581237862  action:  0  reward:  4  Q-values:  [1.1599999999999997, 0.679
i:  63  num:  0.7186881792035356  action:  0  reward:  4  Q-values:  [1.1999999999999997, 0.679
i:  64  num:  0.887500345404199  action:  0  reward:  4  Q-values:  [1.2399999999999998, 0.6799
i:  65  num:  0.42927148096612333  action:  1  reward:  2  Q-values:  [1.2399999999999998, 0.69
i:  66  num:  0.14020507708866525  action:  0  reward:  4  Q-values:  [1.2799999999999998, 0.69
i:  67  num:  0.4325121698077027  action:  0  reward:  4  Q-values:  [1.3199999999999998, 0.699
i:  68  num:  0.7538087066852512  action:  1  reward:  2  Q-values:  [1.3199999999999998, 0.719
i:  69  num:  0.589872613751951  action:  0  reward:  4  Q-values:  [1.3599999999999997, 0.7199
i:  70  num:  0.6445719194229075  action:  0  reward:  4  Q-values:  [1.3999999999999997, 0.719
i:  71  num:  0.5679141025712394  action:  1  reward:  2  Q-values:  [1.3999999999999997, 0.739
i:  72  num:  0.6627272603128029  action:  1  reward:  2  Q-values:  [1.3999999999999997, 0.759
i:  73  num:  0.7939280578360608  action:  1  reward:  2  Q-values:  [1.3999999999999997, 0.779
i:  74  num:  0.18107263840813959  action:  1  reward:  2  Q-values:  [1.3999999999999997, 0.79
i:  75  num:  0.5162590689802459  action:  0  reward:  4  Q-values:  [1.4399999999999997, 0.799
i:  76  num:  0.026374640687109818  action:  0  reward:  4  Q-values:  [1.4799999999999998, 0.7
i:  77  num:  0.9440466468697397  action:  0  reward:  4  Q-values:  [1.5199999999999998, 0.799
```

```
i:  78  num:  0.818722287437148  action:  1  reward:  2  Q-values:  [1.5199999999999998, 0.819
i:  79  num:  0.43565715349303613  action:  0  reward:  4  Q-values:  [1.5599999999999996, 0.8
i:  80  num:  0.6451089628982771  action:  1  reward:  2  Q-values:  [1.5599999999999996, 0.839
i:  81  num:  0.6572516100993825  action:  0  reward:  4  Q-values:  [1.5999999999999996, 0.839
i:  82  num:  0.6894062662172371  action:  1  reward:  2  Q-values:  [1.5999999999999996, 0.859
i:  83  num:  0.7531872295593329  action:  1  reward:  2  Q-values:  [1.5999999999999996, 0.879
i:  84  num:  0.3841431531028364  action:  0  reward:  4  Q-values:  [1.6399999999999997, 0.879
i:  85  num:  0.9323247679943677  action:  1  reward:  2  Q-values:  [1.6399999999999997, 0.899
i:  86  num:  0.0025218752944096146  action:  0  reward:  4  Q-values:  [1.6799999999999997, 0
i:  87  num:  0.5603851251101524  action:  1  reward:  2  Q-values:  [1.6799999999999997, 0.919
i:  88  num:  0.3248496219873651  action:  1  reward:  2  Q-values:  [1.6799999999999997, 0.939
i:  89  num:  0.817077575711092  action:  0  reward:  4  Q-values:  [1.7199999999999998, 0.9399
i:  90  num:  0.7046655216661141  action:  0  reward:  4  Q-values:  [1.7599999999999996, 0.939
i:  91  num:  0.8871347279805294  action:  0  reward:  4  Q-values:  [1.7999999999999996, 0.939
i:  92  num:  0.17786709160141367  action:  1  reward:  2  Q-values:  [1.7999999999999996, 0.95
i:  93  num:  0.44590224826086944  action:  0  reward:  4  Q-values:  [1.8399999999999996, 0.95
i:  94  num:  0.9561646613113443  action:  0  reward:  4  Q-values:  [1.8799999999999997, 0.959
i:  95  num:  0.5034665765062244  action:  1  reward:  2  Q-values:  [1.8799999999999997, 0.979
i:  96  num:  0.6516404932383221  action:  1  reward:  2  Q-values:  [1.8799999999999997, 0.999
i:  97  num:  0.6388376861693754  action:  1  reward:  2  Q-values:  [1.8799999999999997, 1.019
i:  98  num:  0.6674322589786041  action:  0  reward:  4  Q-values:  [1.9199999999999997, 1.019
i:  99  num:  0.5512214116511054  action:  0  reward:  4  Q-values:  [1.9599999999999995, 1.019
```

In [46]:
```
plt.plot(M, 'bs', label='take loop')
plt.plot(1-M, 'g^', label='take bypass')
plt.xlabel('Epoch')
plt.ylabel('Probability')
plt.grid(True)
plt.legend()
plt.show()
```


take loop
take bypass
Probability
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
Epoch

In [ ]: