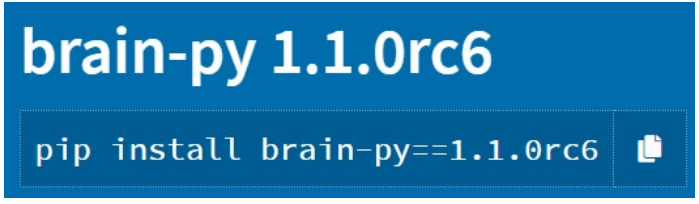
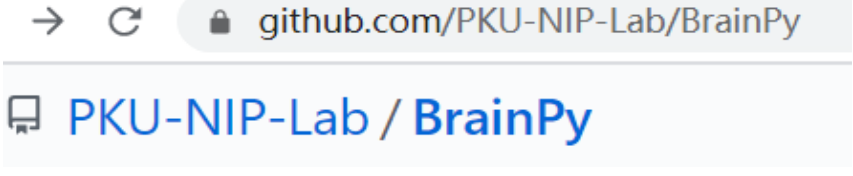
Implement a continuous-attractor network (CANN) with BrainPy

Install BrainPy

方法一：pypi



方法二：GitHub




```
1 import brainpy as bp
2 import brainpy.math as bm
```

The continuous-attractor network to implement

$$\tau \frac{du(x,t)}{dt} = -u(x,t) + \rho \int dx' J(x,x') r(x',t) + I_{ext}$$

$$r(x,t) = \frac{u(x,t)^2}{1 + k\rho \int dx' u(x',t)^2}$$

$$J(x,x') = \frac{1}{\sqrt{2\pi}a} \exp\left(-\frac{|x-x'|^2}{2a^2}\right)$$

$$I_{ext} = A \exp\left[-\frac{|x-z(t)|^2}{4a^2}\right]$$

The two equations can be rewritten as (when dx' small)

$$\tau \frac{du(x,t)}{dt} = -u(x,t) + \rho \sum_{x'} J(x,x') r(x',t) dx' + I_{ext}$$

$$r(x,t) = \frac{u(x,t)^2}{1 + k\rho \sum_{x'} u(x',t)^2 dx'}$$

- $u(x,t)$: the synaptic input at time t of the neurons whose preferred stimulus is x
- $r(x,t)$: the corresponding firing rate
- ρ: the neural density
- τ: the synaptic time constant

Consider a one-dimensional continuous stimulus x. The value of x is in the range of (−π, π].

Three tasks:

- Population coding
- Template matching
- Smooth tracking

Wu, Si, Kosuke Hamaguchi, and Shun-ichi Amari. "Dynamics and computation of continuous attractors." Neural computation 20.4 (2008): 994-1025.
Mi, Yuanyuan, et al. "Spike frequency adaptation implements anticipative tracking in continuous attractor neural networks." NeurIPS 27 (2014): 505-513.

Implement the CANN with BrainPy

```python
class CANN1D(bp.NeuGroup):
  def __init__(self, num, tau=1., k=8.1, a=0.5, A=10., J0=4., z_min=-bm.pi, z_max=bm.pi, **kwargs):
    super(CANN1D, self).__init__(size=num, **kwargs)

    # parameters
    self.tau = tau  # The synaptic time constant
    self.k = k  # Degree of the rescaled inhibition
    self.a = a  # Half-width of the range of excitatory connections
    self.A = A  # Magnitude of the external input
    self.J0 = J0  # maximum connection value

    # feature space
    self.z_min = z_min
    self.z_max = z_max
    self.z_range = z_max - z_min
    self.x = bm.linspace(z_min, z_max, num)  # The encoded feature values
    self.rho = num / self.z_range  # The neural density
    self.dx = self.z_range / num  # The stimulus density
```

Necessary parameters

```python
def dist(self, d):
    d = bm.remainder(d, self.z_range)
    d = bm.where(d > 0.5 * self.z_range, d - self.z_range, d)
    return d
```

Function for the distance in the ring

$$J(x, x') = \frac{1}{\sqrt{2\pi}a} \exp\left(-\frac{|x - x'|^2}{2a^2}\right)$$

```python
def make_conn(self, x):
    assert bm.ndim(x) == 1
    x_left = bm.reshape(x, (-1, 1))
    x_right = bm.repeat(x.reshape((1, -1)), len(x), axis=0)
    d = self.dist(x_left - x_right)
    Jxx = self.J0 * bm.exp(-0.5 * bm.square(d / self.a)) / \
          (bm.sqrt(2 * bm.pi) * self.a)
    return Jxx
```

Function for the connection

```python
def get_stimulus_by_pos(self, pos):
    return self.A * bm.exp(-0.25 * bm.square(self.dist(self.x - pos) / self.a))
```

Function for stimulus

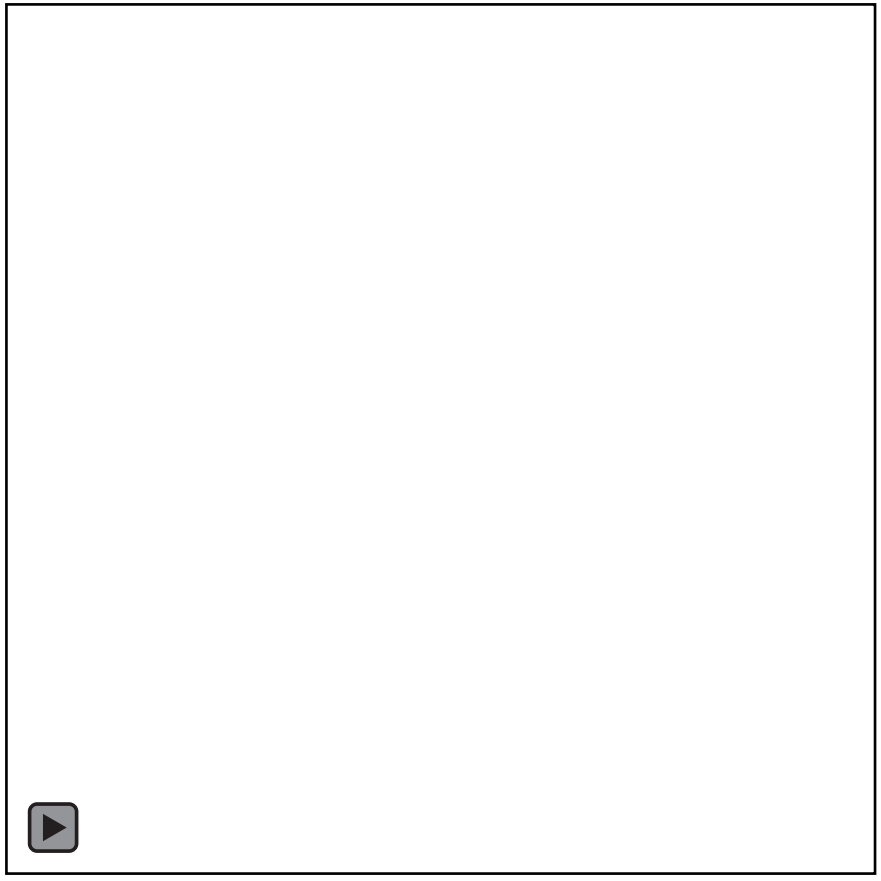
$$I_{ext} = A \exp\left[-\frac{|x - z(t)|^2}{4a^2}\right]$$

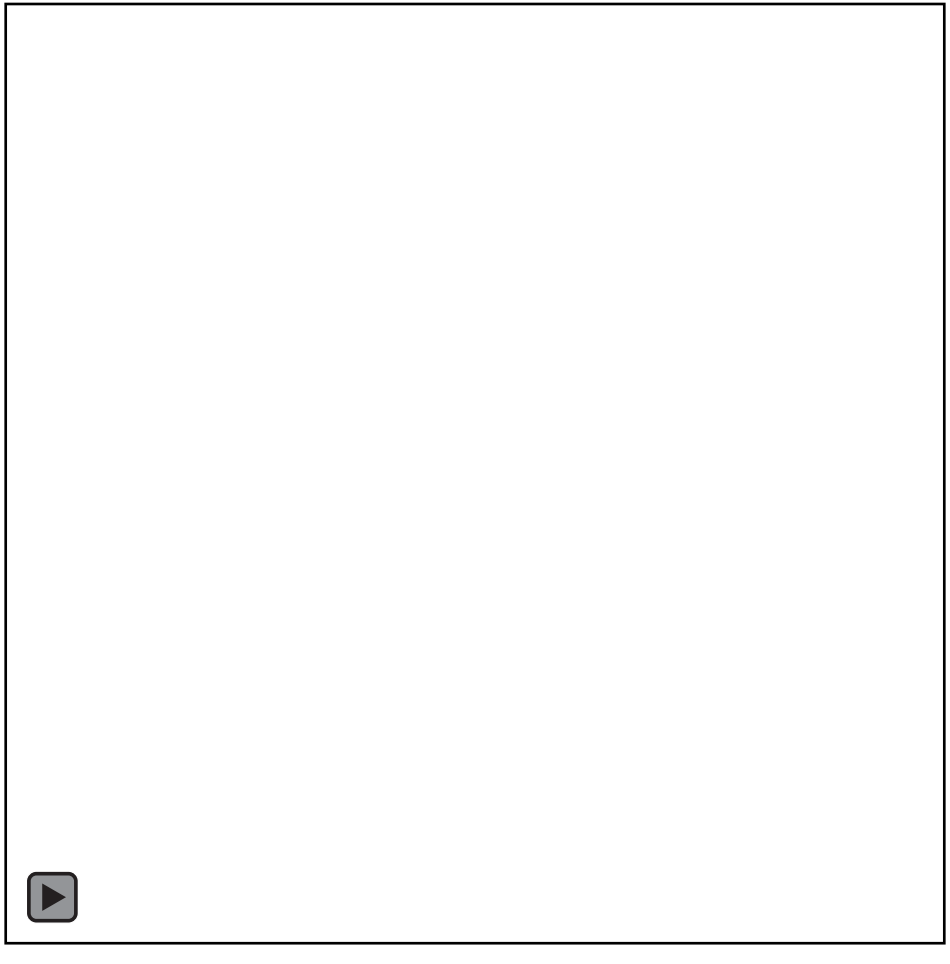
Update Function

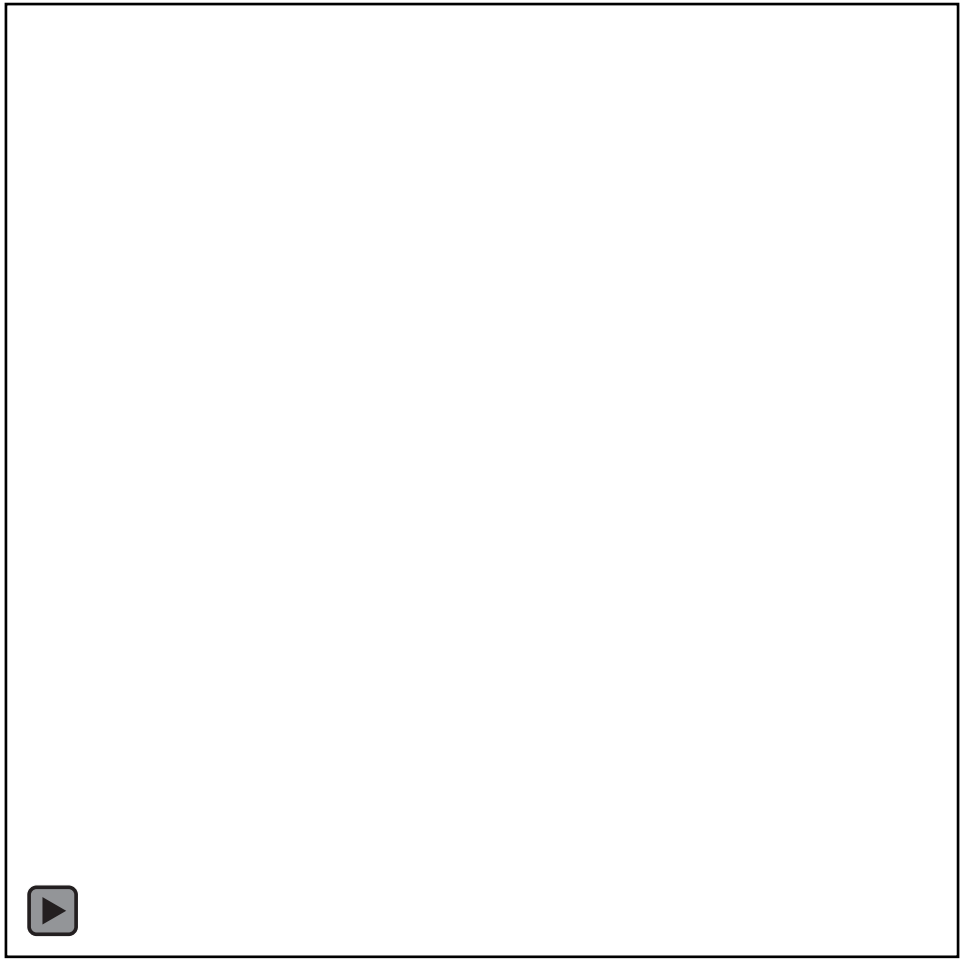
```
45    @bp.odeint(method='rk4')
46    def integral(self, u, t, Iext):
47      r1 = bm.square(u)
48      r2 = 1.0 + self.k * bm.sum(r1)
49      r = r1 / r2
50      Irec = bm.dot(self.conn_mat, r)
51      du = (-u + Irec + Iext) / self.tau
52      return du
53
54    def update(self, _t, _dt):
55      self.u[:] = self.integral(self.u, _t, self.input)
56      self.input[:] = 0.
```

$$r(x,t) = \frac{u(x,t)^2}{1 + k\rho \sum_{x'} u(x',t)^2 dx'}$$

$$\tau \frac{du(x,t)}{dt} = -u(x,t) + \rho \sum_{x'} J(x,x') r(x',t) dx' + I_{ext}$$







参考资料

- https://brainpy-examples.readthedocs.io/en/latest/cann/Wu_2008_CANN.html

课后作业 Homework

1. 阅读文献 《2008 - Dynamics and Computation of Continuous Attractors》

2. 安装Anaconda Python环境(https://docs.anaconda.com/anaconda/install/)

3. 安装 BrainPy == 1.1.0rc6 (https://pypi.org/project/brain-py/1.1.0rc6/)

4. 【**提交作业，代码 + report (配图)**】实现上述PPT中 CANN 模型的3个tasks。

5. 以word或pdf的格式提交报告。将“代码+报告”打包以“姓名_学号_作业二报告.zip”命名，在11月16日24:00前发送到邮箱chutianhao@stu.pku.edu.cn