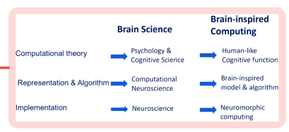
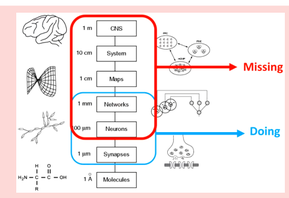
计算神经与BrainPy

计算神经科学概述

- 背景
  - 历史
  - 脑科学的三个层次



  - 神经计算的任务
- 目标与挑战
  - 两大目标
  - 挑战

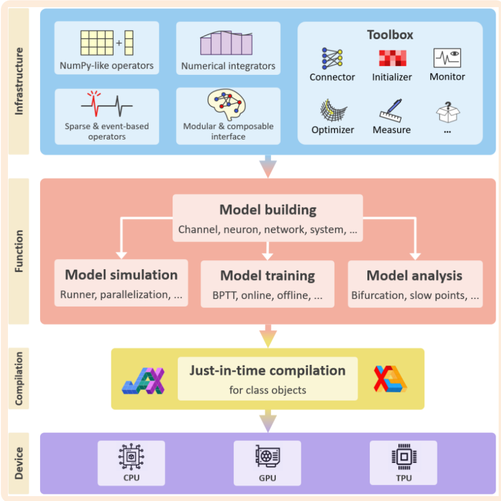


The missing link: a computational model of higher cognitive function

- 建模工具
  - 要求工具
    - 效率
    - 整合
    - 灵活
    - 有扩展性
  - BrainPy
- 建模举例
  - 图像理解
    - 如何提取整体特征
    - 如何从表示空间中形成"整体"假设?
    - 如何feedback从整体到局部

BrainPy简介

- BrainPy Architecture
  - 包装
  - 结构图



BrainPy模块

- 主要特点
  - 稠密运算算子

NumPy 数组运算

```python
import numpy as np
# 创建数组
>>> a =  np.zeros((2, 2))
>>> a
array([[0., 0.],
       [0., 0.]])
# 数组原位操作
>>> a[0] += 5.
# 数组内置函数
>>> a.max()
5.
# 线性代数函数
>>> np.dot(a, np.ones((2, 1)))
array([[10.],
       [ 0.]])
```

BrainPy 数组运算

```python
import brainpy.math as bm
# 创建数组
>>> a = bm.zeros((2, 2))
>>> a
JaxArray(DeviceArray([[0., 0.],
                      [0., 0.]]))
# 数组原位操作
>>> a[0] += 5.
# 数组内置函数
>>> a.max()
DeviceArray(5., dtype=float32)
# 线性代数函数
>>> bm.dot(a, bm.ones((2, 1)))
JaxArray(DeviceArray([[10.],
                      [ 0.]]))
```

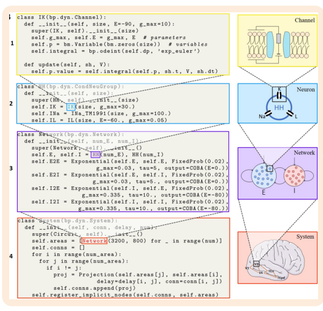
NumPy 随机数运算

```python
# 正态分布
>>> np.random.normal(size=(2, 1))
array([[-0.81543646],
       [ 1.20518382]])
# 均匀分布
>>> np.random.uniform(size=(2, 1))
array([[0.79619099],
       [0.48682065]])
```

BrainPy 随机数运算

```python
# 正态分布
>>> bm.random.normal(size=(2, 1))
JaxArray(DeviceArray([[-0.2434824 ],
                      [-0.16923107]]))
# 均匀分布
>>> bm.random.uniform(size=(2, 1))
JaxArray(DeviceArray([[0.37081087],
                      [0.998075  ]]))
```

  - 专用的算子
  - 数值积分器
  - 模块化和可组合性



  - 面向对象的即时编译

```python
class LogisticRegression(bp.BrainPyObject):
  def __init__(self, dimension):
    super(LogisticRegression, self).__init__()

    # parameters
    self.dimension = dimension

    # variables
    self.w = bm.Variable(2.0 * bm.ones(dimension) - 1.3)

  def __call__(self, X, Y):
    u = bm.dot((1.0 / (1.0 + bm.exp(-Y * bm.dot(X, self.w))) - 1.0) * Y), X)
    self.w.value = self.w - u # in-place update
```

  - 即时编译Just-in-Time compilation
  - 编程基础
    - 数据算子
      - array
        - 静态数组
      - variable
        - 动态变量
      - in-place updating
        - 原地更新
    - 控制流
      - 条件语句
        - if-else
          - brainpy.math.where

```python
a = 1.
bm.where(a < 0, 0., 1.)
```

          - brainpy.math.ifelse

```python
def ifelse(condition, branches, operands):
  true_fun, false_fun = branches
  if condition:
    return true_fun(operands)
  else:
    return false_fun(operands)
```

      - 循环语句
        - for loop

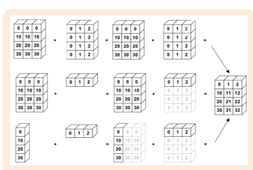
```python
def for_loop_function(body_fun, xs):
  ys = []
  for x in xs:
    results = body_fun(x)
    ys.append(results)
  return ys
```

        - while loop

```python
while cond_fun(x):
    x = body_fun(x)
```

Python和基础模块

- Python
  - 基本元素
  - 数据类型
    - List
    - Tuple
    - Dictionary
  - Class类
    - 属性：attribute
    - 方法：method
- NumPy
  - Array
    - 规约操作
    - Slicing
    - 矩阵操作
    - array broadcasting：



  - Linear algebra

```python
import numpy.linalg

qr              ## Computes the QR decomposition
cholesky        ## Computes the Cholesky decomposition
inv(A)          ## Inverse
solve(A,b)      ##  Solves Ax = b for A full rank
lstsq(A,b)      ##  Solves arg minx ||Ax - b||2
eig(A)          ##  Eigenvalue decomposition
eigvals(A)      ##  Computes eigenvalues
svd(A, full)    ## Singular value decomposition
pinv(A)         ##  Computes pseudo-inverse of A
```

    - 线性代数
  - 傅里叶变换

```python
import numpy.fft
fft  ## 1-dimensional DFT
fft2 ## 2-dimensional DFT
fftn ## N-dimensional DFT
ifft ## 1-dimensional inverse DFT (etc.)
rfft ## Real DFT (1-dim)
```

  - 随机和分布

```python
import numpy.random

rand(d0,d1,...,dn)          ## Random values in a given shape
randn(d0, d1, ...,dn)       ## Random standard normal
randint(lo, hi, size)       ## Random integers [lo, hi)
choice(a, size, repl, p)    ## Sample from a
shuffle(a)                  ## Permutation (in-place)
permutation(a)              ## Permutation (new array)
```

- SciPy

Presented with xmind