计算神经建模

Day1

神经计算建模简介

计算神经科学的目标

·用计算建模的方法阐明大脑功能的计算原理

·发展类脑智能的模型和算法

计算神经科学的历史

·1907 LIF model

·**1950s HH model**：单个神经元的模型

·1960s Roll's cable equation: 精细神经元模型，考虑丰富的轴突、树突结构

·1970s Amari, Wilson, Cowan et al.

·1982 Hopfield model (Amari-Hopfield model): 吸引子网络模型，大量物理学背景的研究者进入计算神经领域

·1988 Sejnowski et al. "**Computational Neuroscience**"

理解大脑的三个层次 (David Marr)

- Computational Theory -> Psychology & Cognitive Science -> Human-like Cognitive function
- Representation & Algorithm -> Computational Neuroscience -> Brain-inspired model & algorithm
- Implementation -> Neuroscience -> Neuromorphic computing

大脑处理动态信息

海鞘

计算神经的问题

缺少从神经元出发的能描述大脑高级认知功能的模型

Molecules-Synapses-Neurons-Networks-Maps-System-CNS

神经计算的工具

- Efficiency: high-speed
- Integration: simulation + training + analysis
- Flexibility: accomodate new models
- Extensibility: extensible to new modeling methods (ML)

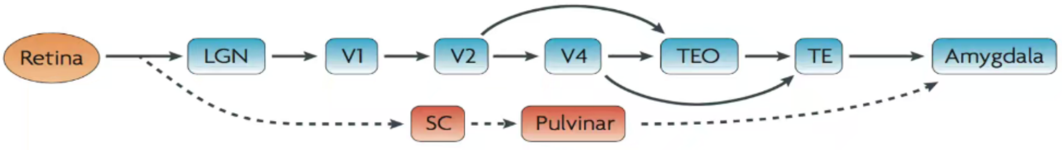
神经计算建模举例

- Image understanding = Image segmentation + object recognition

  The solution of brain: Analysis-by-synthesis 猜测与印证

- The Subcortical pathway

  老鼠看到从小变大的光斑就立刻装死



- Topology first

  视觉系统更敏感于拓扑性质的差异

Python & BrainPy: Programming Basics

Python Basics

NumPy Basics

```python
import numpy as np
# Array
a = np.arange(10).reshape(2, 5)
a.ndim # 2 dimension
a.shape # (2, 5) shape of array
a.size # 10
a.T # transpose
a.dtype # data type

# Array broadcasting
# Two dimensions are compatible when
# 1. They are of equal size
# 2. One of them is 1

# Operations along axes
a = np.ones((2, 3))
a.sum(axis=0) # array{[2., 2., 2.]}
```

BrainPy Introduction

- Dense operators

  Compatible with NumPy, TensorFlow, Pytorch, etc.

- Dedicated operators

  Applies brain dynamics sparse connectivity properties with event-driven computational features.

- Numerical Integrators

  ODE: brainpy.odeint

  SDE: brainpy.sdient

  FDE: brainpy.fdeint

  Delayed differential equations

- Modular and composable

- Object-oriented JIT compilation

BrainPy Programming Basics

- Just-in-Time compilation

  Static compilation converts the code into a language for a specific platform

  Interpreter directly executes the source code

  JIT is compilation that is being done during the execution of a program.

- Object-oriented JIT compilation

  The base class of BrainPy is bp.BrainPyObject, whose methods will be automatically JIT compiled. So the class object must be inherited from bp.BrainPyObject to avoid performing JIT manually.

  All time-dependent variables must be defined as brainpy.math.Variable.

- How to debug?

  Turn off JIT compilation.

- Data operations

- Array
- Variables

  Arrays that are not marked as dynamic variables will be JIT-compiled as static arrays, and modifications to static arrays will not be valid in the JIT compilatoin environment.

```
v.value = bn.arange(10)
```

- Control flows: If-else

  Non- `Variable` -based control statements

```
if condition:
else:
```

  `Variable` -based control statements

```
bm.where(a < 0, 0., 1.)
bm.ifelse(condition, true_branches, false_branches)
```

- Control flows: for loop

  NOT efficiency

```
bm.for_loop(body_fun= , operands= )
```

- Control flows: while loop