PsychRNN

An open source Python package for cognitive task modeling using recurrent neural networks
How is PsychRNN useful?

- **Cognitive tasks**
  - Flexible Framework for defining cognitive tasks
  - Examples
    - Random Dot Motion (RDM)
      - Perceptual discrimination
    - Vibrotactile Delayed Discrimination
      - Romo et al., 1999
      - Parametric working memory

- **Biological Constraints**
- **Recurrent Neural Net backend**
  - Useful model
  - Different types (RNN, LSTM)
Experimentalists

- Abstraction of Neural Nets
- Intuitive Task Definitions

Theorists

- Reduced Barrier to entry
- Modularity
- Curriculum Learning
- Biological Constraints

- Flexibility of Framework
Vibrotactile Delayed Discrimination Task

- Traditionally
  - Two vibration frequencies
  - Two stimulus periods
  - After, monkey decides: higher or lower?

- Our Task
  - Two channels
    - As in Machens et al., 2005
    - Channels have opposite stimuli during stimulus period

RDM: Random Dot Motion Task

- Traditionally
  - Two directions
  - Dots moving in both directions
  - %Coherence in direction

- Our Task
  - Two channels
  - Coherence: channel value

Biological constraints

- Dale’s Law
- Connectivity
  - Input
  - Recurrent
  - Output
Modularity

Merriam-Webster Definition: “constructed with standardized units or dimensions for flexibility and variety in use”

- Abstracted
  - Flexible
  - Easy to Use
- Object-Oriented
  - Customizable
for i in range(3):
    for j in range(3):
       romo = rmo.Romo(dt = 10, tau = 100, T = 2000, N_batch = 1, stim_duration_1 = (i+1)/3, stim_duration_2 = (j+1)/3)
        gen = romo.batch_generator()
        x,_,_ = next(gen)
Package Structure

PsychRNN Backend

Loss
Accuracy

PsychRNN Task Object

Trained Neural Network

Synaptic Weights
Output
State Variables
Task Definition

Class your_new_class(Task):

    def generate_trial_params(self, batch, trial):
        ...
        produces trial specific params for your task
        ...

    def trial_function(self, t, params):
        ...
        specifies conditional network input, target output and loss mask for task at time t
        ...

only requires knowledge of numpy and python

Task Input-Output Description

Trial Parameters

PsychRNN Task Object
Backend Selection

RNN

LSTM

User Defined Network

PsychRNN Backend

Biological Constraints
Number of Units

Requires TensorFlow Knowledge

Training Parameters

Trained Neural Network
Curriculum Learning

- Observation: people learn best in stages
  - Eg: Learning chess
- So do machines
  - Bengio et al. 2009
  - Learns faster
  - Quality of local minima
  - Enables learning of more complicated tasks
Curriculum Learning: Beyond Speedup

- Animals trained in stages on cognitive tasks
  - Work intensive
  - Often not addressed
- Speed up animal training?
- How do different curricula affect results?
Curriculum Learning Before

For each stage in learning:

- Instantiate Network
- Train Network
- Save Weights
- Destroy Network
What does curriculum learning look like now?

- Instantiate Network
- For each stage in learning:
  - Train Network
- Save Weights
- Destroy Network

```
from psychrnn.backend.curriculum import Curriculum
from psychrnn.tasks import rdm as rd

rdms = [rd.RDM(dt = 10, tau = 100, T = 2000, N_batch = 128, coherence=((10 - i)/15)) for i in range(10)]

curriculum = Curriculum(rdms)
train_params['curriculum'] = curriculum
model.train(rdms[0].batch_generator(), train_params)
```
Why PsychRNN?

How does PsychRNN compare to alternative options?
Advantages over Alternatives

Pycog

- Modularity
- Curriculum Learning
- Supported Backend
  - PsychRNN: Tensorflow
  - Pycog: Theano

Machine Learning Frameworks (TensorFlow, etc)

- Lower barrier-to-entry
- Focus on the science
- Enables exploration
  - Easy to tweak / try something new
- Clean, standardized, easy-to-understand code
Thank You!

Questions?