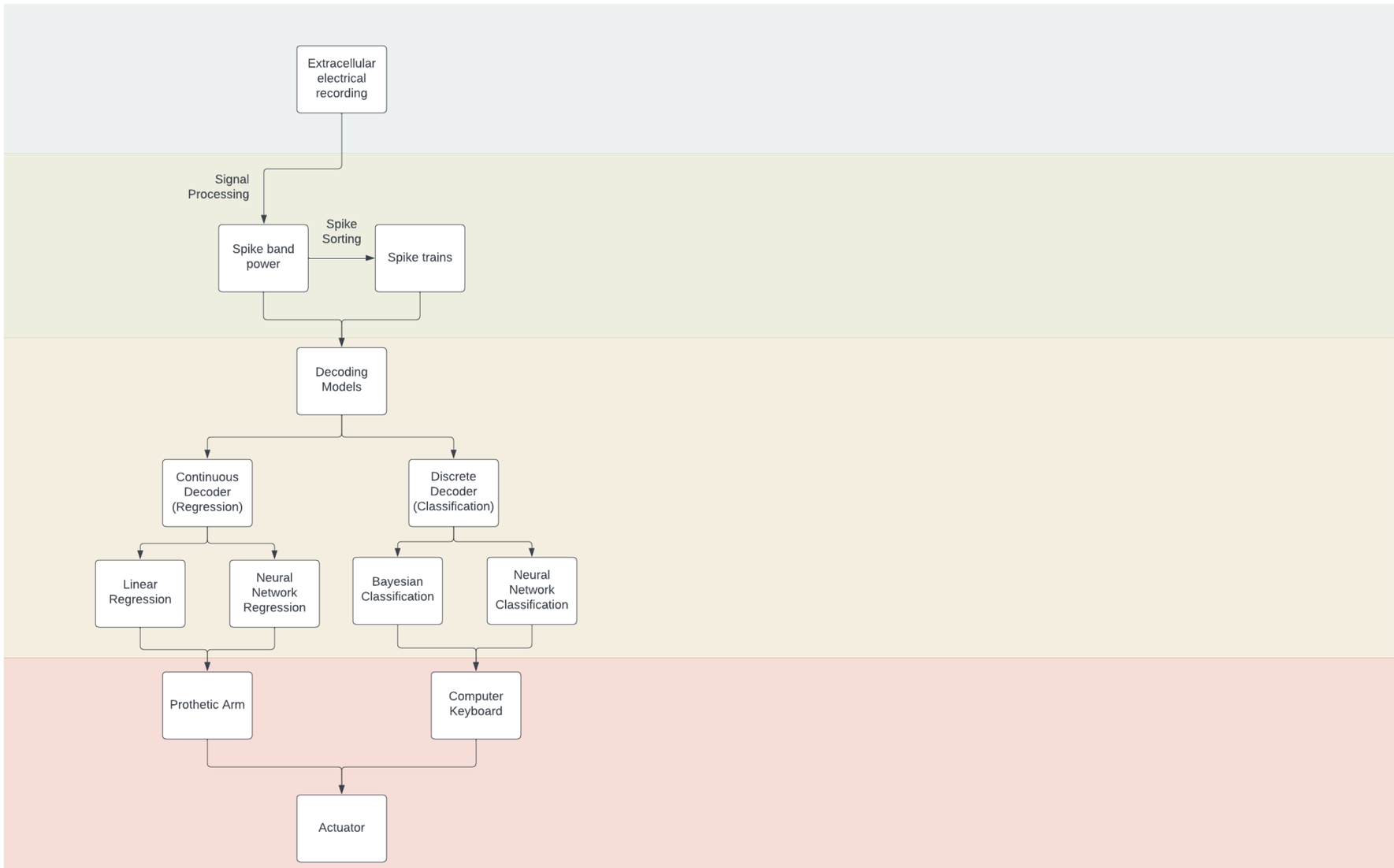
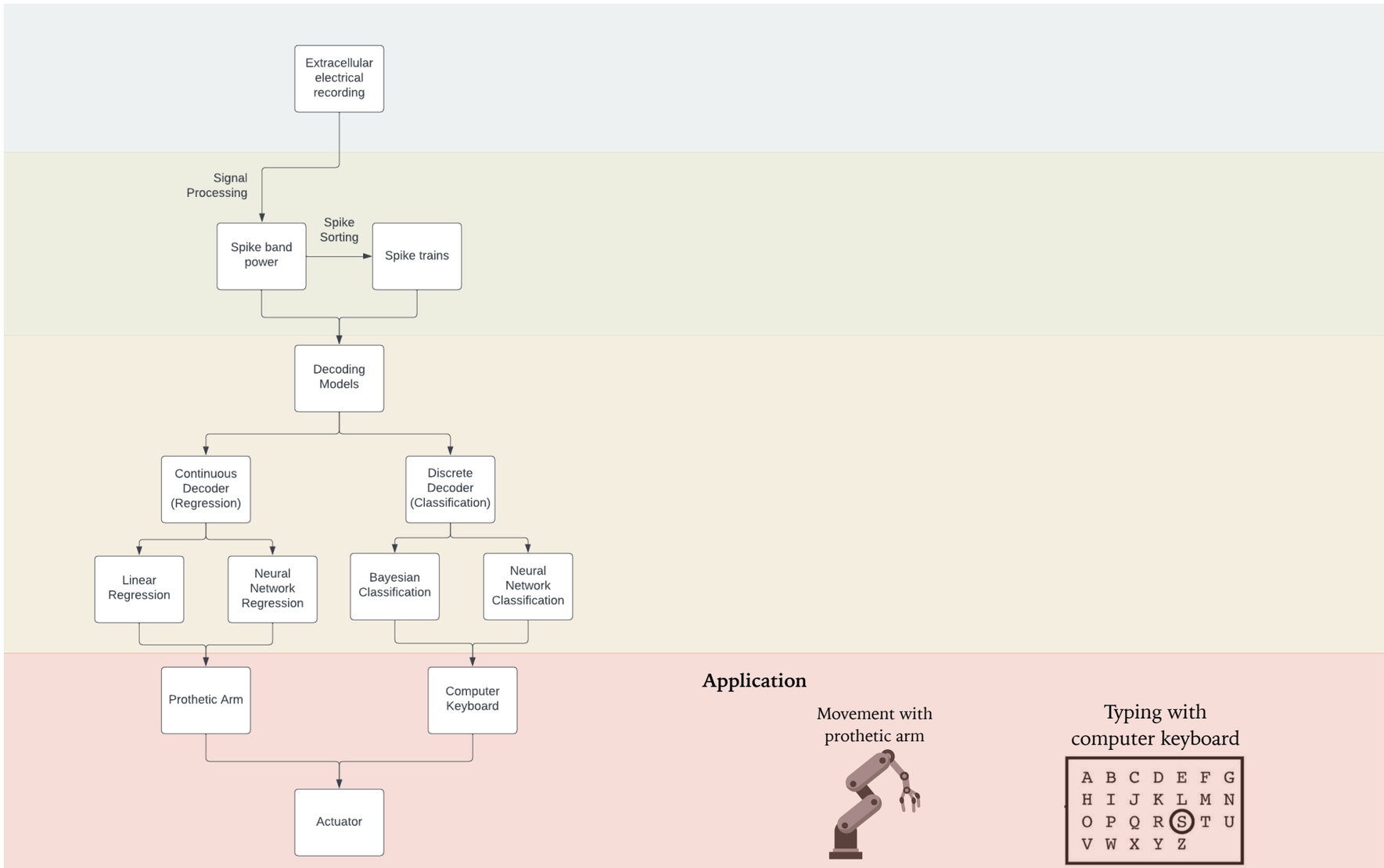
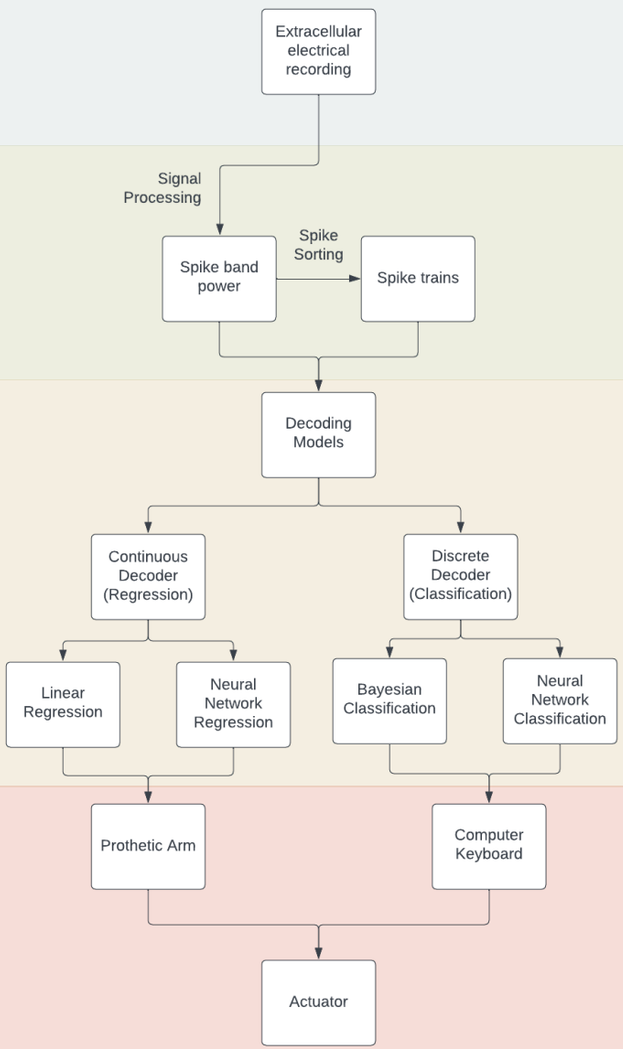
Self-Guided Coding Session



Formulate Decoding Problem Base on Target Application



Formulate Decoding Problem Base on Target Application



Decoding Objective

The **continuous decoder** translates, on a moment-by-moment basis, neural activity into a continuous-valued movement trajectory

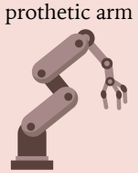
- Precision at each time-point, avoiding accumulated error
- Integrate "feedback" to correct for error at each time-point

The **discrete decoder** act as a classifier to classify the reaching target while the user is preparing and completing the reach.

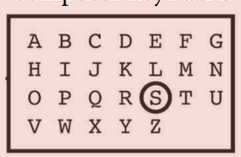
- Decode target with high accuracy as early as possible
- The precise movement trajectory doesn't matter

Application

Movement with prothetic arm



Typing with computer keyboard



Formulate Decoding Problem Base on Target Application

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share J

Table of contents

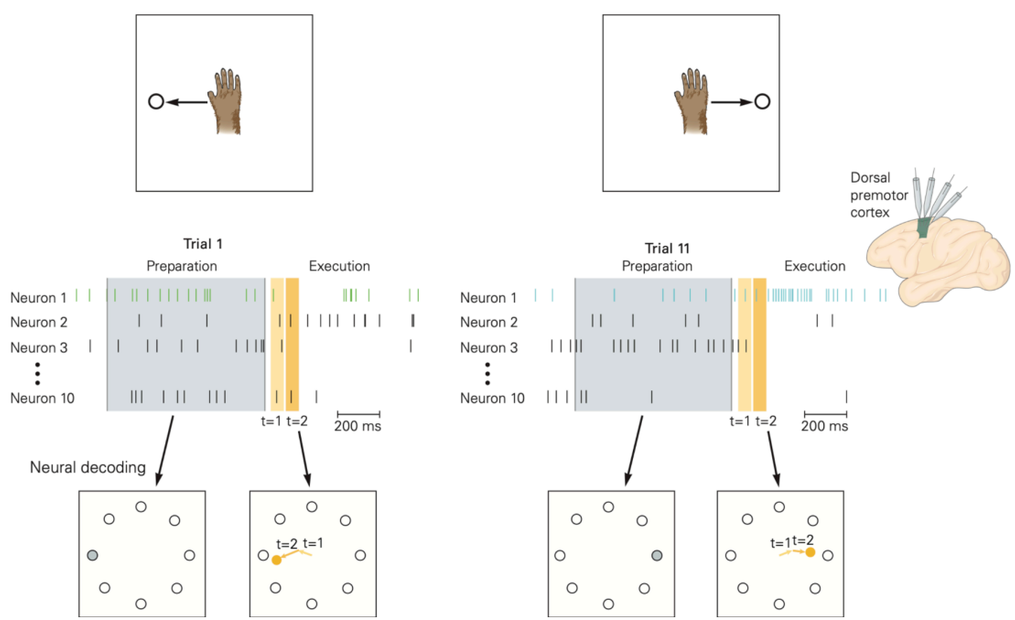
+ Code + Text Reconnect

Formulate the Neural Decoding Problem

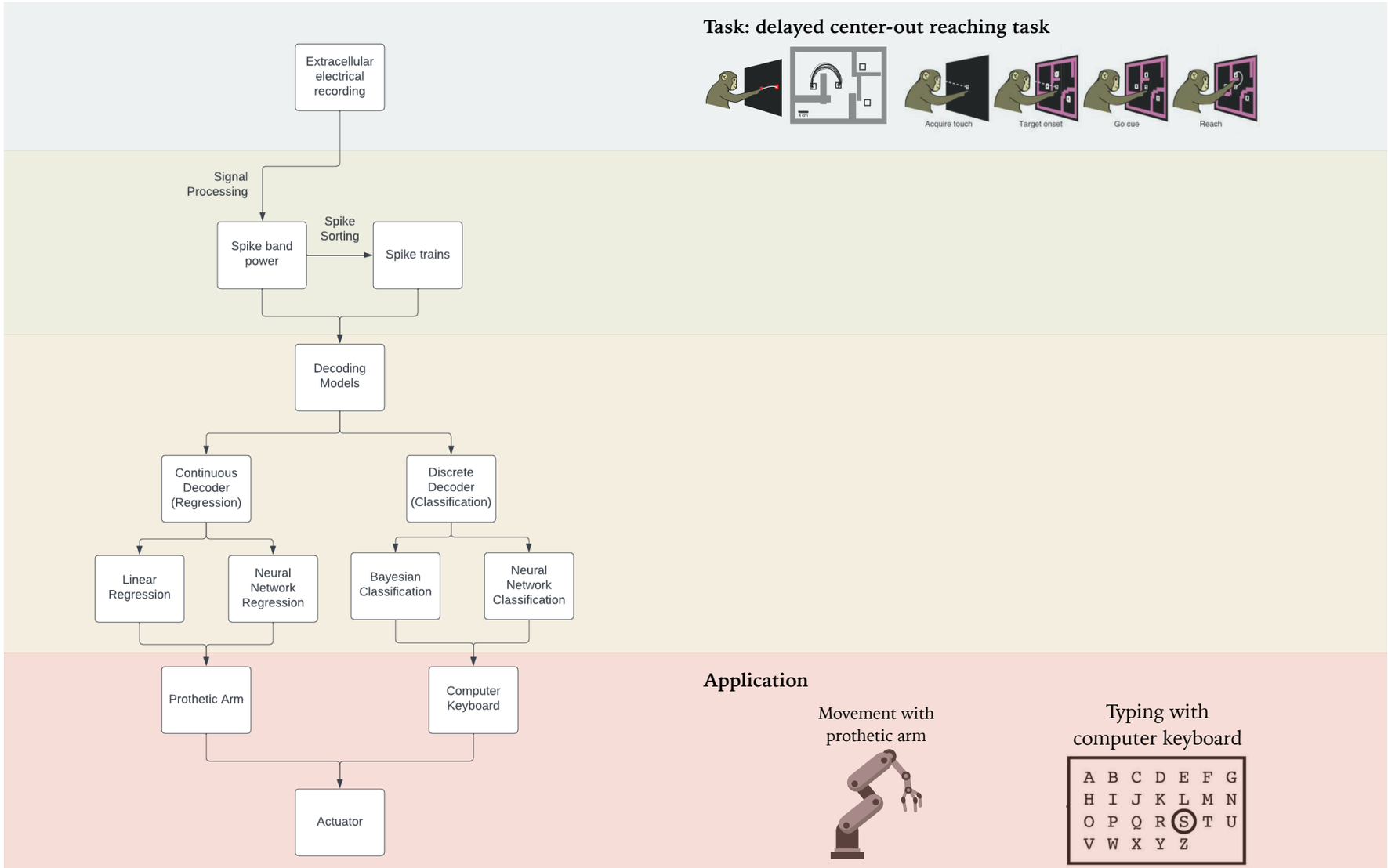
The primary goal of the neural decoding algorithm is to translate the electrical neural activity from the brain into control signals for guiding paralyzed upper limbs, prosthetic arms, and computer cursors to achieve user's movement intent. Neural decoding algorithms can be devised to decode the desired movement trajectory (continuous decoder) or movement target (discrete decoder) based on the pattern of neural activity.

To direct the prosthetic arm through space, the neural decoder must produce movement trajectories as accurately as possible to enact precisely the desired movement. The **continuous decoder** translates, on a moment-by-moment basis, neural activity into a continuous-valued movement trajectory.

In contrast, for neural decoder to direct the cursor for key selection on the keyboard, the percise trajectory moving through the keyboard is less important compare to the speed in decoding the target key. The **discrete decoder**, thus, act as a classifier to classify the reaching target while the user is preparing and completing the reach.



MC_Maze: Delayed Center-out Reaching Task Dataset



MC_Maze: Delayed Center-out Reaching Task Dataset

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share J

Table of contents

+ Code + Text Reconnect

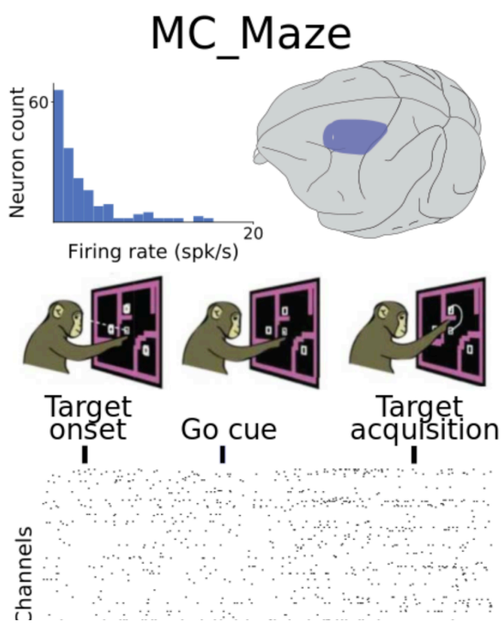
- Neural Decoding
- Tutorial Objectives
- Set Up
  - Install
  - Setting
    - Imports
    - Figure Settings
  - Plotting Functions
  - Helper Functions
- **Delayed Arm Reaching (MC_Maze Dataset)**
  - Task
  - Time Alignment
    - Plan and Movement Neural Activity
    - [500ms Plan Activity, 500ms Movement Activity] Aligned at move_onset_time
  - Behavior Recording
    - Reaching Velocity and Trajectory for one example Condition
  - Neural Recording
    - Single-neuron responses
    - Peristimulus Time Histogram (PSTH)
- Formulate the Neural Decoding Problem
  - Format the Data

Delayed Arm Reaching (MC_Maze Dataset)

The MC_Maze dataset encompasses recordings from sessions in which a macaque performed delayed center-out reaches, capturing neural activity from the primary motor and dorsal premotor cortices.

This dataset, featuring concurrent neural and reaching recordings, offers a means to comprehend the response characteristics of neurons during various reaching behaviors and explore the intricate relationship between this activity and the moment-to-moment reaching movements. This exploration serves as a foundational step toward the development of neural decoding methods, allowing for the extraction of the desired movement trajectory (continuous decoding) or movement target (discrete decoding) based on the neural activity patterns.

This data was provided by Krishna Shenoy, Mark Churchland, and Matt Kaufman from Stanford University, and you can learn more about the task design, data collection, and their analyses of the data in a number of papers, including this (Churchland et al. 2010).



Experiment Task Structure: Plan and Movement Activity

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share J

+ Code + Text

RAM Disk

Table of contents

Time Alignment

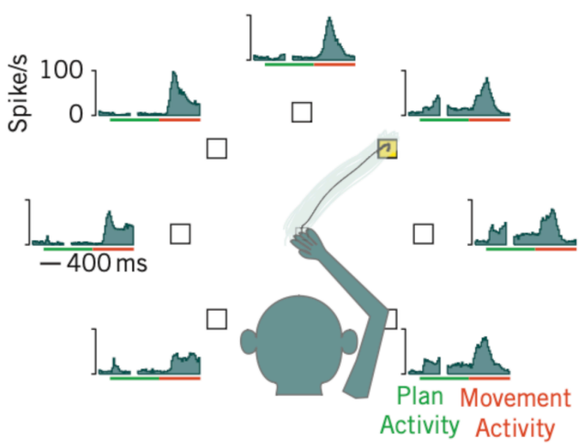
Plan and Movement Neural Activity

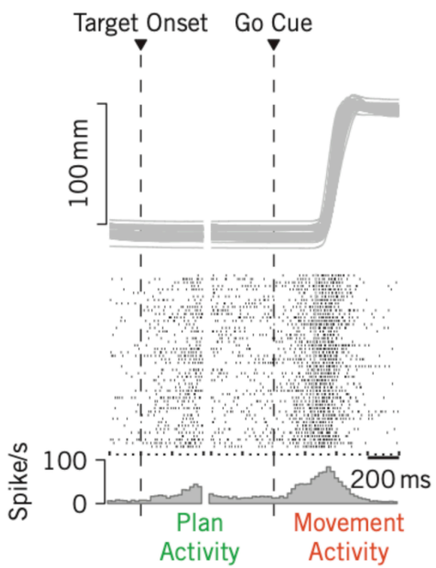
Two types of neural activity are commonly used for driving prosthetic systems: plan and movement activity.

- **Plan Activity**: Plan activity is present before arm movements begin and is believed to reflect preparatory processing required for the fast and accurate generation of movement. Plan activity has been shown to reflect reach target, peak velocity, eye position and attention.
- **Movement Activity**: Movement activity follows plan activity in a delayed reach task. Movement activity often exhibits a moment-to-moment correspondence with the intended trajectory of the hand's movement.

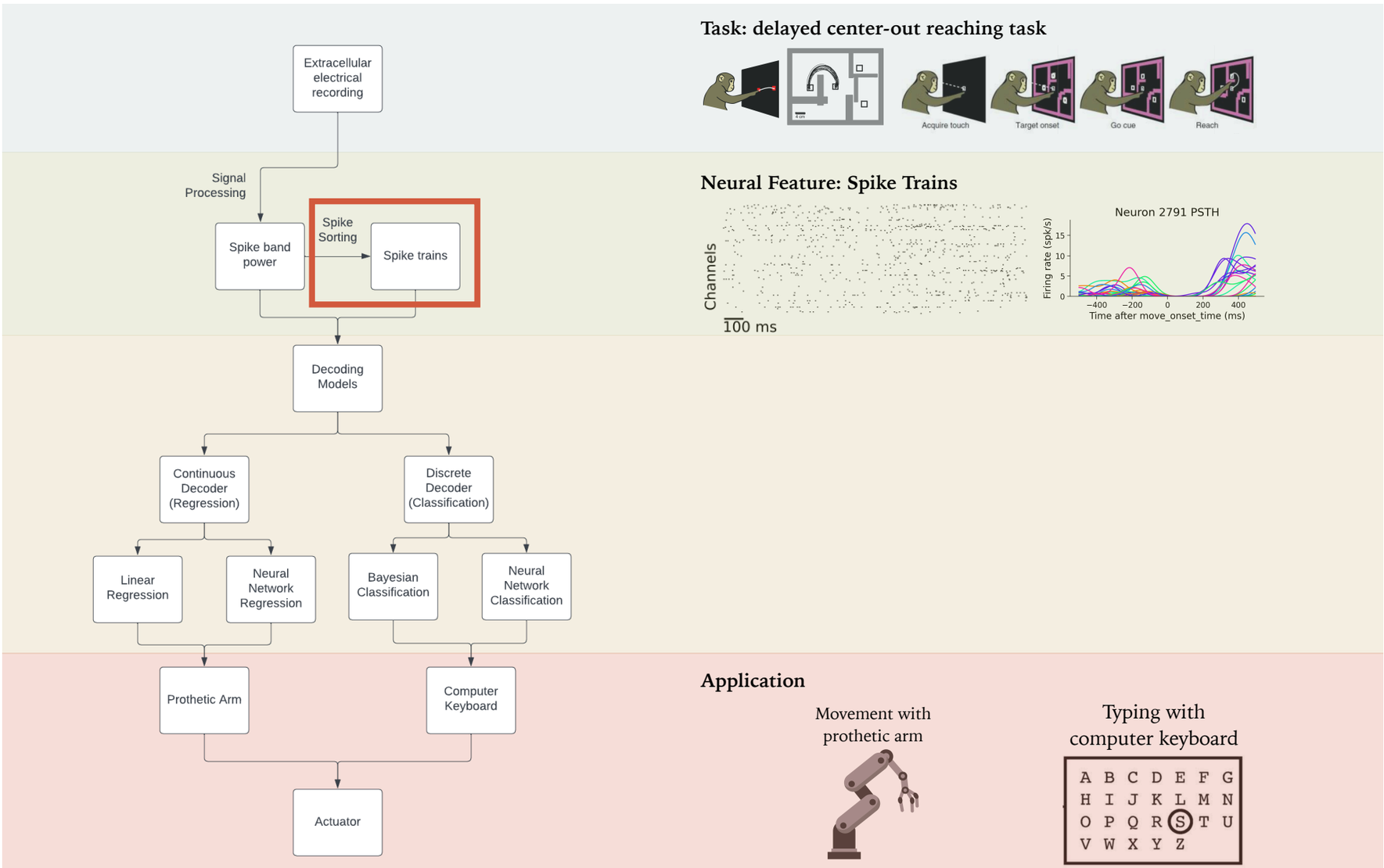
Localization of the neural activity for movement plan and control:

- **PMd (dorsal premotor cortices) neurons** typically exhibit strong plan activity, as well as some movement activity
- **M1 (Primary motor cortex) neurons** typically exhibit the opposite pattern





Neural Feature: Spike Trains



Neural Feature: Spike Trains

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share J

Table of contents

+ Code + Text Reconnect

- Neural Decoding
- Tutorial Objectives
- Set Up
  - Install
  - Setting
    - Imports
    - Figure Settings
  - Plotting Functions
  - Helper Functions
- Delayed Arm Reaching (MC_Maze Dataset)
  - Task
  - Time Alignment
    - Plan and Movement Neural Activity
    - [500ms Plan Activity, 500ms Movement Activity] Aligned at move_onset_time
  - Behavior Recording
    - Reaching Velocity and Trajectory for one example Condition
  - **Neural Recording**
    - Single-neuron responses
    - Peristimulus Time Histogram (PSTH)
- Formulate the Neural Decoding Problem
  - Format the Data

Neural Recording

For these datasets, neural activity was recorded from two Utah arrays: one implanted in the dorsal premotor cortex, which is thought to play a role in movement planning, and one in the primary motor cortex.

This recorded data was spike sorted offline into the provided unit spike times.

- `spikes` | spike times binned at 1 ms

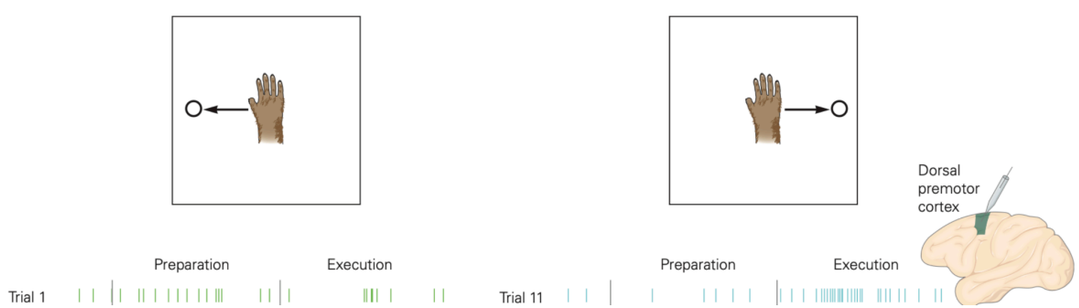
Single-neuron responses

Various aspects of movement (e.g. position, velocity, acceleration, and force) are encoded in the activity of neurons throughout the motor system. Even though our understanding of movement encoding in the motor system is incomplete (mechanism of cortex control for movement), there is usually a relationship between aspects of movement and neural activity. This reliable relationship allows us to estimate the desired movements from neural activity.

Consider the activity of an individual neuron across repeated movements (referred to as 'trials') to the same target. The activity of the neuron can be averaged across many trials to create a spike histogram for each target, often referred to as peristimulus time histogram (PSTH). By comparing the spike histograms for different target, one can characterize how the neuron's activity varies with the movement produced.

Given leftward and rightward arm reaching as an example, the firing rate of the neuron (approximated with spike histogram) shows statistical regularity across the two reach condition and across time.

- Distinct spike patterns for leftward and rightward arm reaching in both preparation and execution time period. On average, a greater level of preparation activity for leftward movements and a greater level of execution activity for rightward movements.
- Consistent moment-by-moment variability in spike patterns across trials of similiar arm reaching indicates correlation between neural activity and arm trajectory across time.



Behavior: Reaching Trajectory and Hand Velocity

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Table of contents

+ Code + Text

RAM Disk

Behavior Recording

In addition to the neural data, cursor position and the monkeys' hand and gaze position were recorded during the experiment, and the hand velocity was estimated offline using the recorded hand position.

- `cursor_pos` | x and y position of the cursor controlled by the monkey
- `eye_pos` | x and y position of the monkey's point of gaze on the screen, in mm
- `hand_pos` | x and y position of the monkey's hand, in mm
- `hand_vel` | x and y velocities of the monkey's hand, in mm/s, computed offline using np.gradient

There are 27 different reach conditions in the MC_Maze dataset. Here, we'll plot the average trajectory per condition to see what typical reaches look like.

5s

```
[35] ## Plot trial-averaged reaches

     # Find unique conditions
     conds = dataset.trial_info.set_index(['trial_type', 'trial_version']).index.unique().tolist()

     # Initialize plot
     fig = plt.figure(figsize=(6, 6))
     ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

     # Loop over conditions and compute average trajectory
     for cond in conds:
         # Find trials in condition
         mask = np.all(dataset.trial_info[['trial_type', 'trial_version']] == cond, axis=1)
         # Extract trial data
         trial_data = dataset.make_trial_data(align_field=data_align_field, align_range=data_align_range, ignored_trials=(~mask))
         # Average hand position across trials
         traj = trial_data.groupby('align_time')[[('hand_pos', 'x'), ('hand_pos', 'y')]].mean().to_numpy()
         # Determine reach angle for color
         active_target = dataset.trial_info[mask].target_pos.iloc[0][dataset.trial_info[mask].active_target.iloc[0]]
         reach_angle = np.arctan2(*active_target[::-1])
         # Plot reach
         ax.plot(traj[:, 0], traj[:, 1], linewidth=0.7, color=plt.cm.hsv(reach_angle / (2*np.pi) + 0.5))
         ax.scatter(traj[-1, 0], traj[-1, 1], color=plt.cm.hsv(reach_angle / (2*np.pi) + 0.5))

     plt.axis('off')
     plt.show()
```

Format Data to Data Tensor

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share J

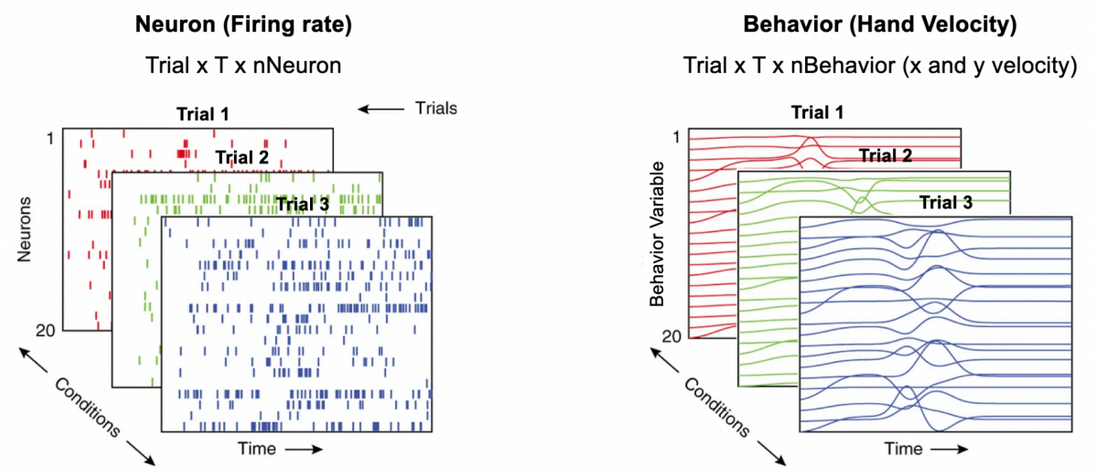
RAM Disk

+ Code + Text

Table of contents

- Dataset)
  - Task
  - Time Alignment
    - Plan and Movement Neural Activity
    - [500ms Plan Activity, 500ms Movement Activity] Aligned at move_onset_time
  - Behavior Recording
    - Reaching Velocity and Trajectory for one example Condition
  - Neural Recording
    - Single-neuron responses
    - Peristimulus Time Histogram (PSTH)
- Formulate the Neural Decoding Problem
- **Format the Data**
- Continuous Decoder (Regression)
  - Machine Learning Regression
    - Ridge Regression
  - Neural Network Regression
    - Recurrent Neural Network
    - RNN(GRU): many-to-many
      - Training RNNRegressor
      - Pre-trained RNNRegressor
      - Evaluate the Model performance

Format the Data




```
[21] # alignment parameter for this dataset
     data_align_field='move_onset_time'
     data_align_range = (-500, 500)
     # Decoding behavior with neural response preceding the behavior
     response_delay = -70

     # Select hand velocity and hand position and behavior data
     behavior_data_fields = ['hand_vel', 'hand_pos']
     # Select smoothed spike train as neural feature
     neuron_data_fields = ['spikes_rate']

     # Size of the time bin in (ms) for each time sample
     bin_width = dataset.bin_width
     # Number of discrete state
     n_state = 6
```

```
[22] ## Load dataset
```

Format Data to Data Tensor

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

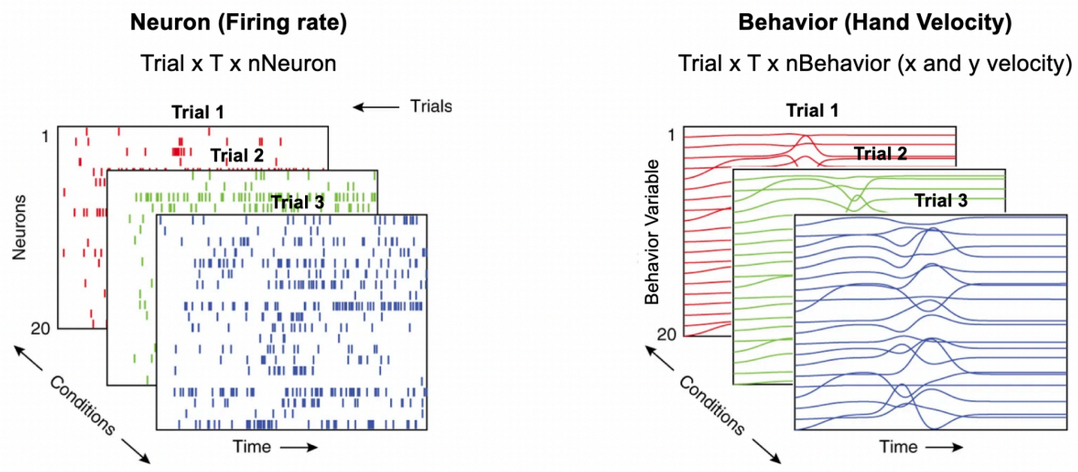
Table of contents

- Dataset)
  - Task
  - Time Alignment
    - Plan and Movement Neural Activity
    - [500ms Plan Activity, 500ms Movement Activity] Aligned at move_onset_time
  - Behavior Recording
    - Reaching Velocity and Trajectory for one example Condition
  - Neural Recording
    - Single-neuron responses
    - Peristimulus Time Histogram (PSTH)
- Formulate the Neural Decoding Problem
- **Format the Data**
- Continuous Decoder (Regression)
  - Machine Learning Regression
    - Ridge Regression
  - Neural Network Regression
    - Recurrent Neural Network
    - RNN(GRU): many-to-many
      - Training RNNRegressor
      - Pre-trained RNNRegressor
      - Evaluate the Model performance

+ Code + Text

RAM Disk

Format the Data




```
[21] # alignment parameter for this dataset
     data_align_field='move_onset_time'
     data_align_range = (-500, 500)
     # Decoding behavior with neural response preceding the behavior
     response_delay = -70

     # Select hand velocity and hand position and behavior data
     behavior_data_fields = ['hand_vel', 'hand_pos']
     # Select smoothed spike train as neural feature
     neuron_data_fields = ['spikes_rate']

     # Size of the time bin in (ms) for each time sample
     bin_width = dataset.bin_width
     # Number of discrete state
     n_state = 6
```

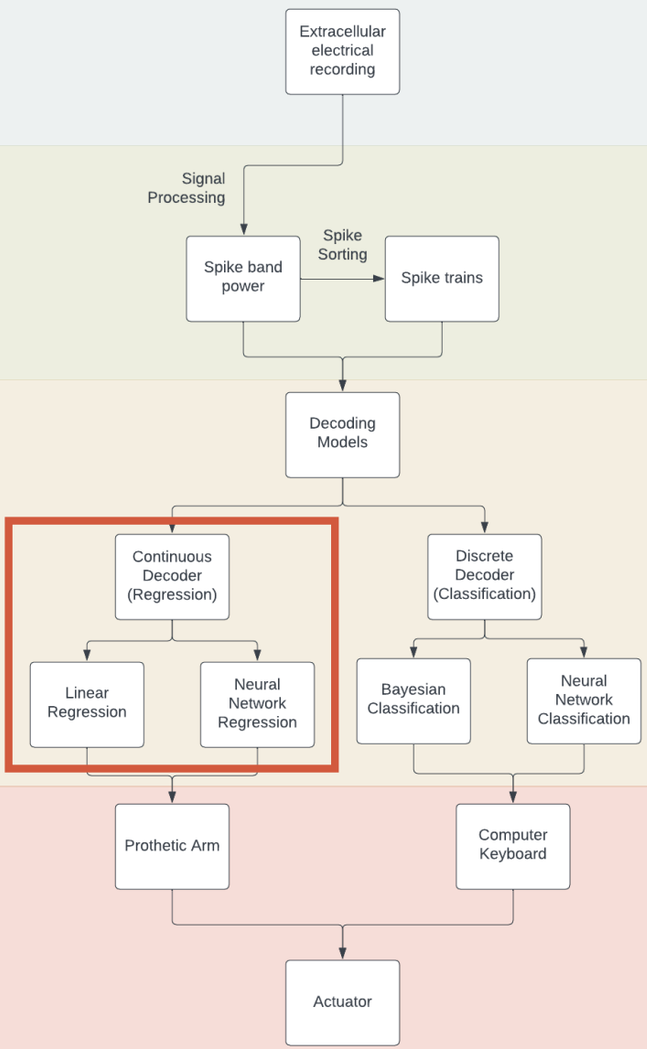
```
[22] ## Load dataset
```

Formatted Data are stored in data_dict

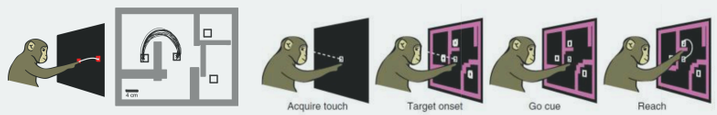
```
print("==============================")
print("Key data fields in data_dict")
print("==============================")

for key in data_dict.keys():
    data_dict[key] = np.vstack(data_dict[key])
    print(f"{key}: {np.shape(data_dict[key])}")
    if key not in ['traj', 'psth']:
      data_dict[key] = data_dict[key][trial_order]
data_dict['trial_id'] = np.array(trial_id)[trial_order]
```

```
==============================
Key data fields in data_dict
==============================
hand_vel: (250, 200, 2)
hand_pos: (250, 200, 2)
cond_label: (250, 2)
cond: (250, 1)
reach_angle: (250, 1)
spikes_rate: (250, 200, 152)
traj: (27, 200, 2)
psth: (27, 200, 152)
state: (250, 1)
```

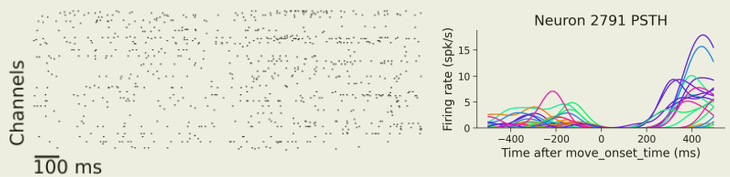
Continuous Decoder: Decoding Hand X and Y Velocity



Task: delayed center-out reaching task

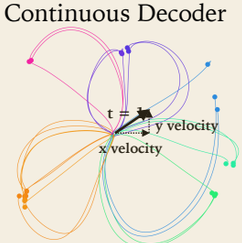


Neural Feature: Spike Trains



Decoding Objective

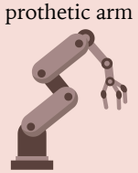
Continuous Decoder



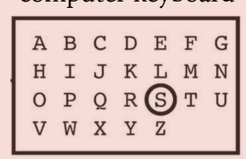
Regression problem: predict the hand x and y velocity in each time bin.

Application

Movement with prothetic arm



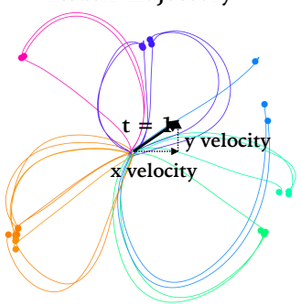
Typing with computer keyboard



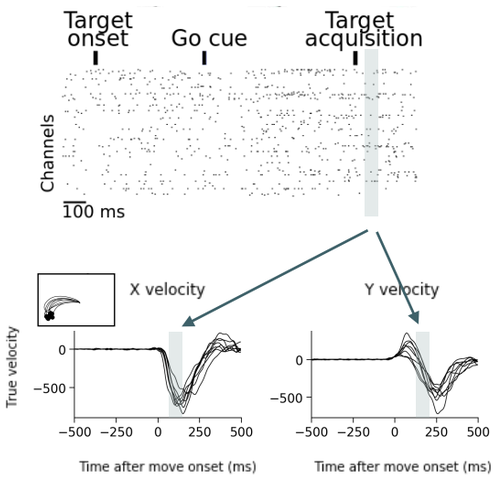
Regression Problem: Decode Hand Velocity With Neural Firing Rate

Regression Problem

Reach Trajectory



Neural Spike Trains

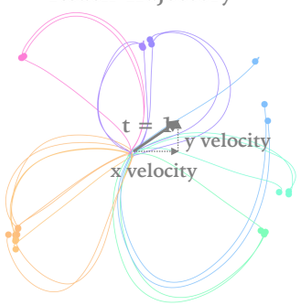


Behavior Hand Velocity

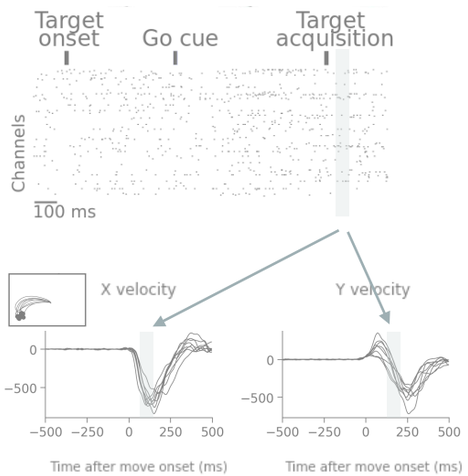
Regression Problem: Decode Hand Velocity With Neural Firing Rate

Regression Problem

Reach Trajectory



Neural Spike Trains

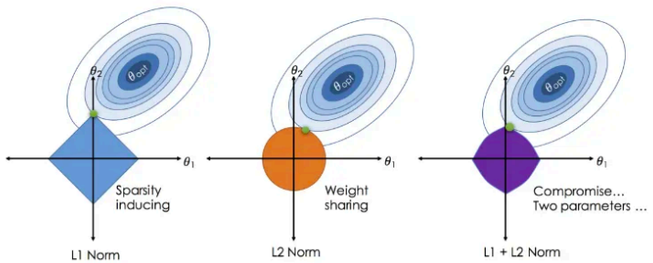


Behavior Hand Velocity

Regression Model

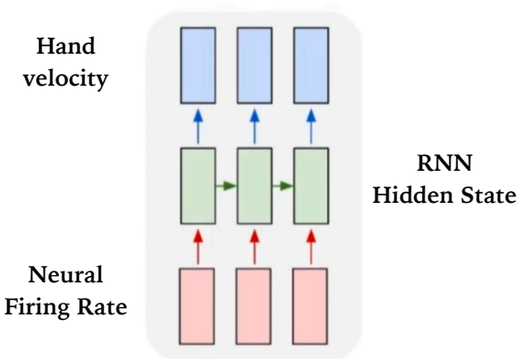
Regularized Linear Regressor

Linear regression with L2 norm penalty for size of the weights to encourage a regression model that generate better at testing time



RNN Regressor

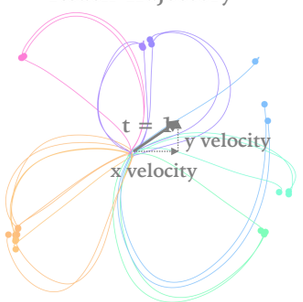
Many-to-many RNN regressor to predict the hand velocity at each corresponding time-point while accumulating trajectory information in the past with the hidden state



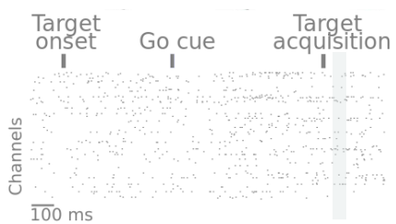
Regression Problem: Decode Hand Velocity With Neural Firing Rate

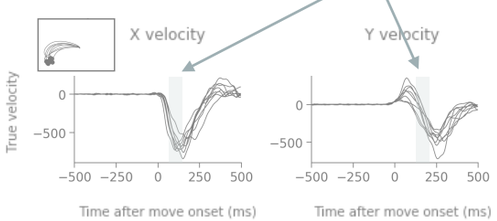
Regression Problem

Reach Trajectory



Neural Spike Trains



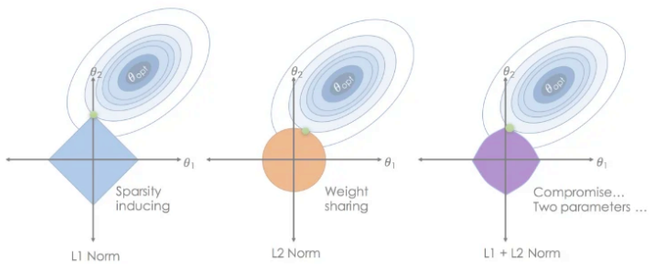


Behavior Hand Velocity

Regression Model

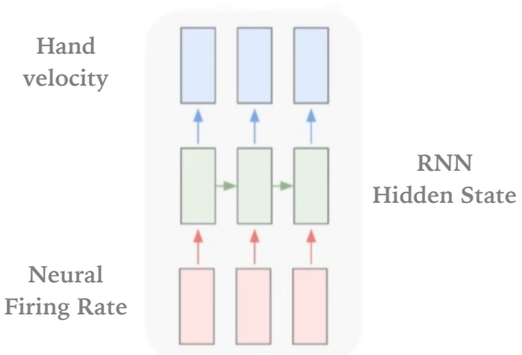
Regularized Linear Regressor

Linear regression with L2 norm penalty for size of the weights to encourage a regression model that generate better at testing time



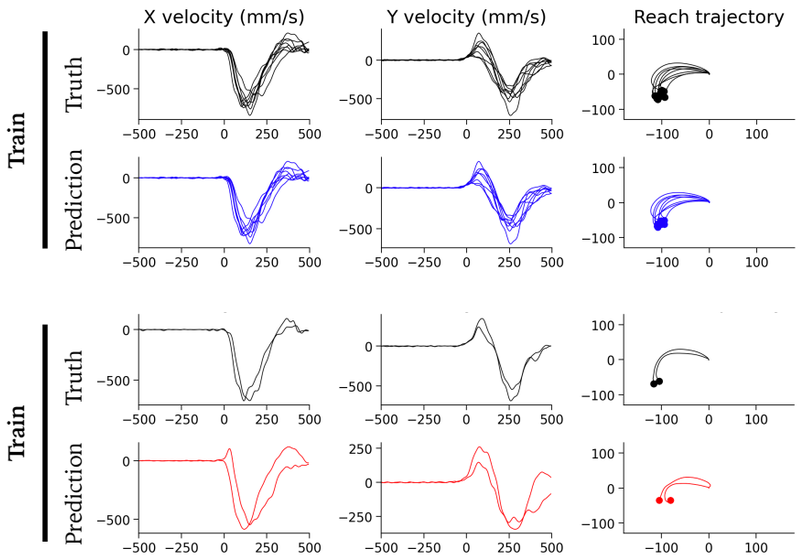
RNN Regressor

Many-to-many RNN regressor to predict the hand velocity at each corresponding time-point while accumulating trajectory information in the past with the hidden state



Model Evaluation

Visualizing decoded hand trajectory (per condition, repeated trials)



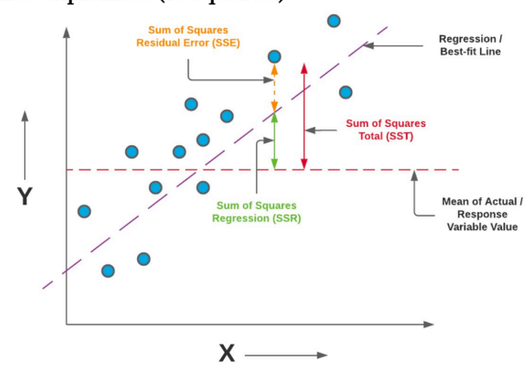
Percentage of variance explained (R-squared)

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

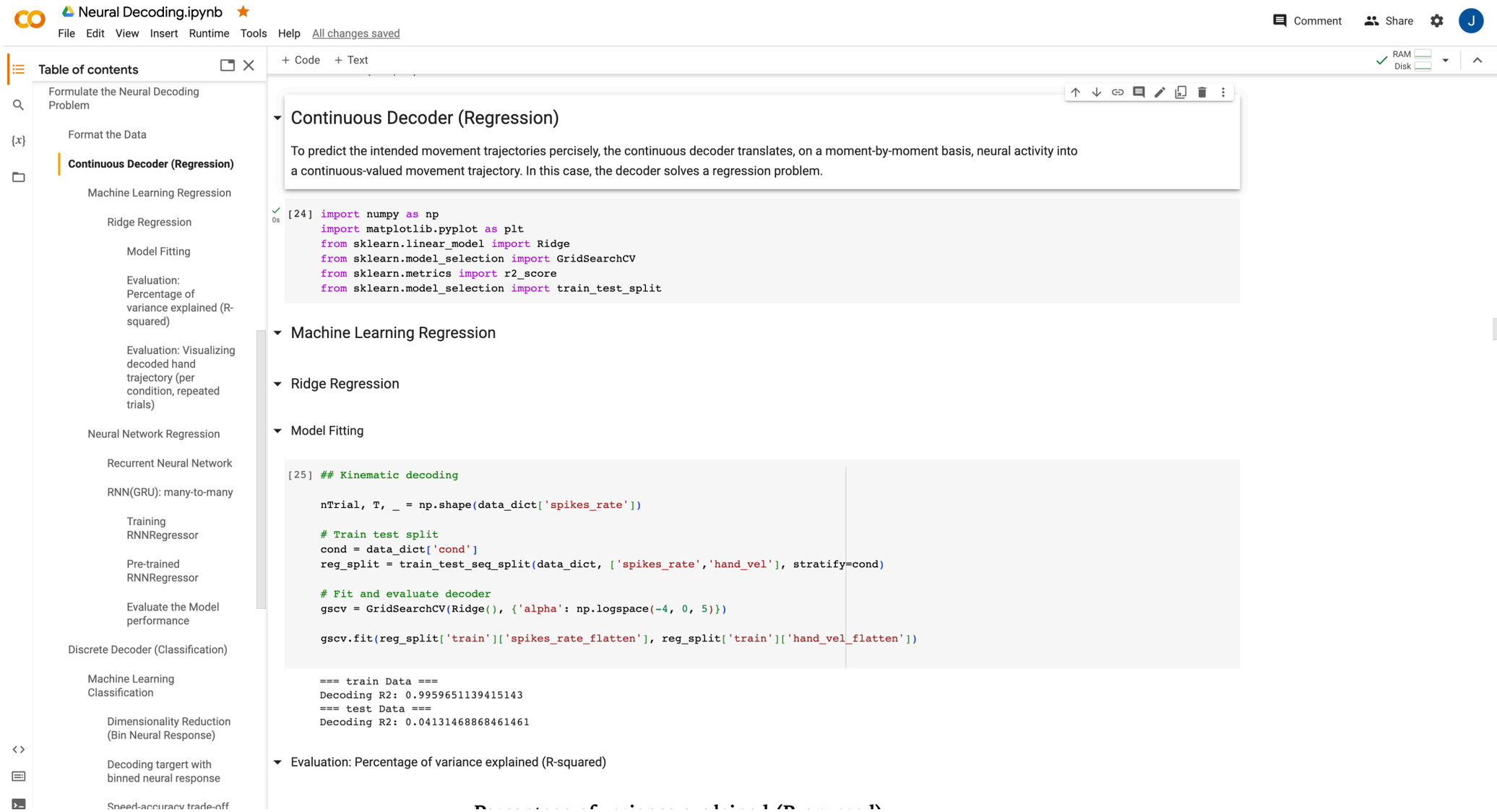
R^2 = coefficient of determination

RSS = sum of squares of residuals

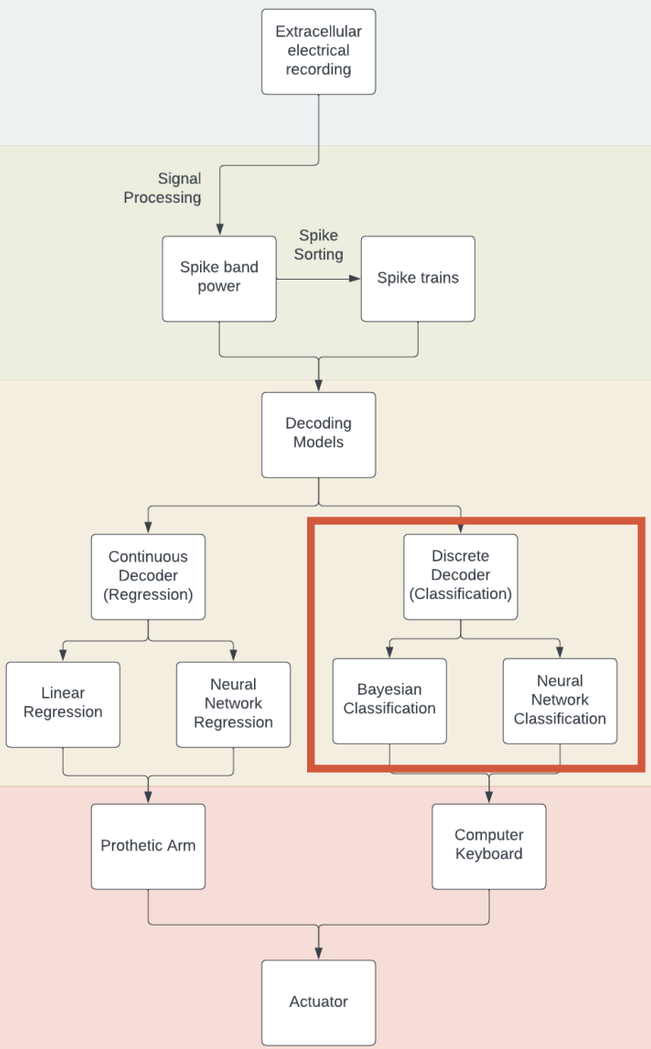
TSS = total sum of squares



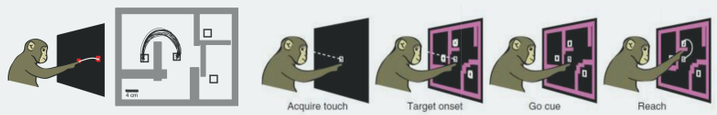
Regression Problem: Decode Hand Velocity With Neural Firing Rate



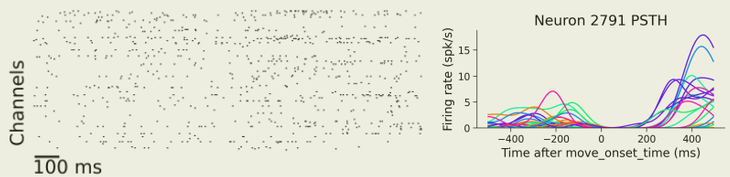
Discrete Decoder: Decoding Reaching Target



Task: delayed center-out reaching task

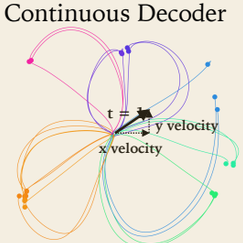


Neural Feature: Spike Trains



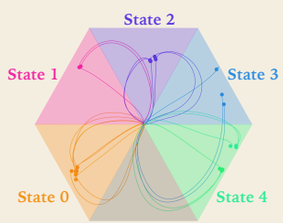
Decoding Objective

Continuous Decoder



Regression problem: predict the hand x and y velocity in each time bin.

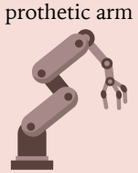
Discrete Decoder



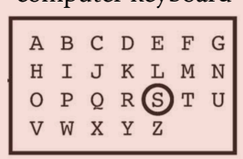
Classification problem: predict the reaching target (discretized reaching angle).

Application

Movement with prothetic arm

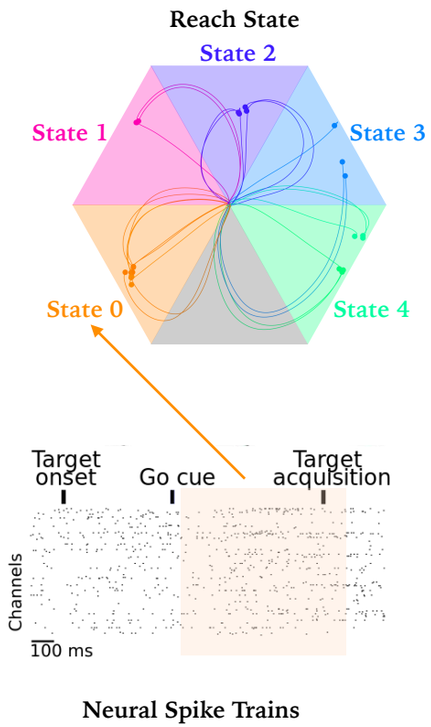


Typing with computer keyboard



Classification Problem: Decode Reaching Target With Neural Firing Rate

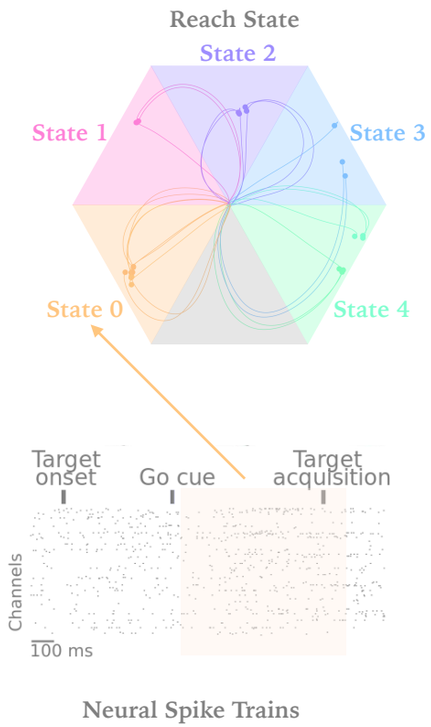
Classification Problem



Neural Spike Trains

Classification Problem: Decode Reaching Target With Neural Firing Rate

Classification Problem



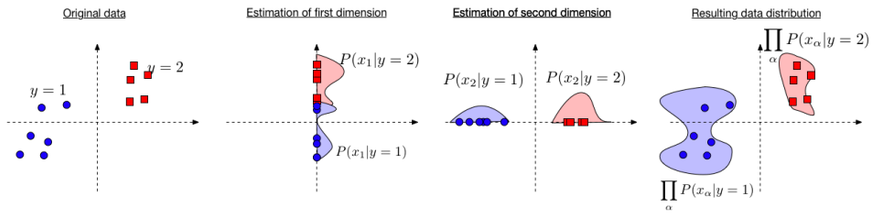
Neural Spike Trains

Classification Model

Naive Bayesian Classifier

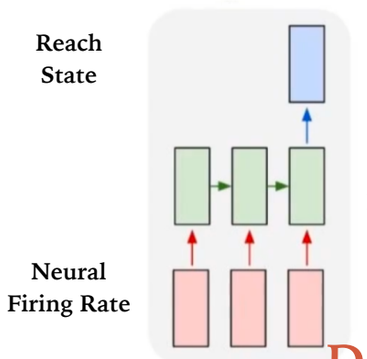
Estimate the class probability base on class prior p(y) and estimation of the class conditioned distribution p(x|y), with naive assumption that each feature dimension is independent given the class label.

$$P(\mathbf{x}|y) = \prod_{\alpha=1}^{d} P(x_\alpha|y), \text{where } x_\alpha = [\mathbf{x}]_\alpha \text{ is the value for feature } \alpha$$



RNN Classifier

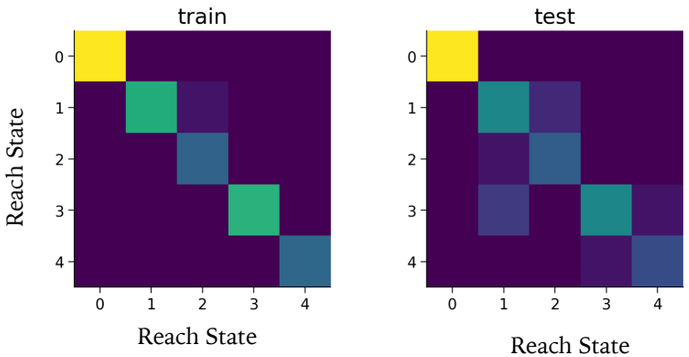
Many-to-on RNN Classifier to predict the reach target by integrating all observed neural firing rate from time 1 to t through the RNN hidden state.



Decoding Challenge

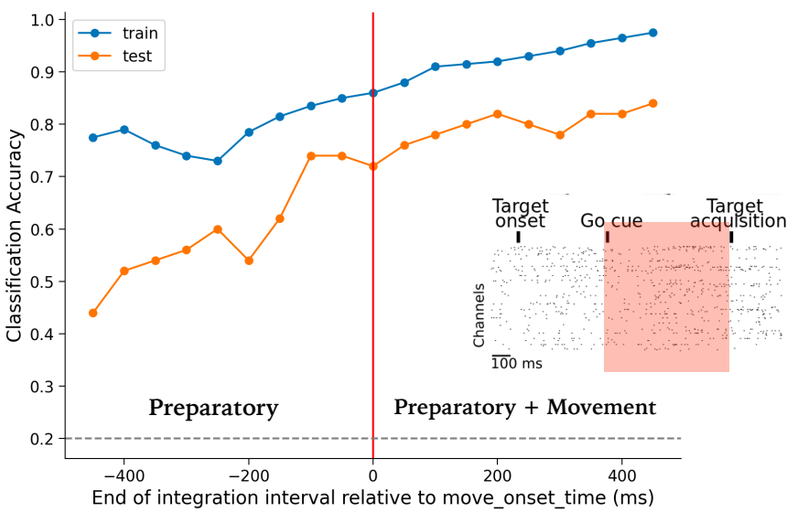
Model Evaluation

Confusion Matrix



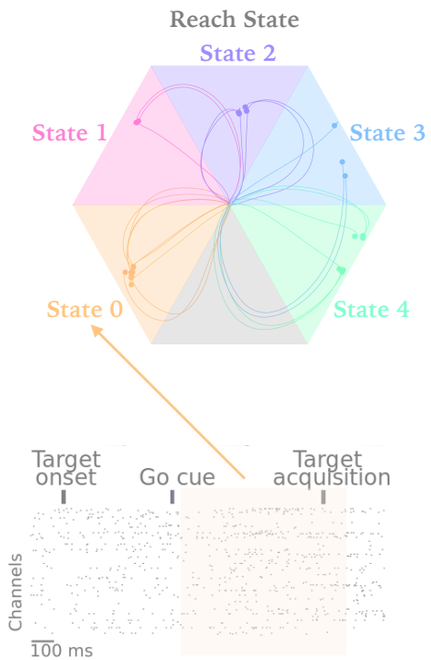
Classification Accuracy

(As a function of integration window)



Classification Problem: Decode Reaching Target With Neural Firing Rate

Classification Problem



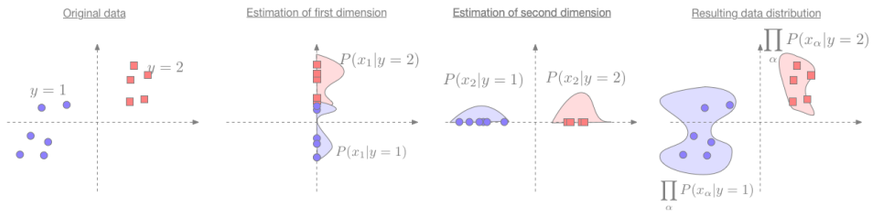
Neural Spike Trains

Classification Model

Naive Bayesian Classifier

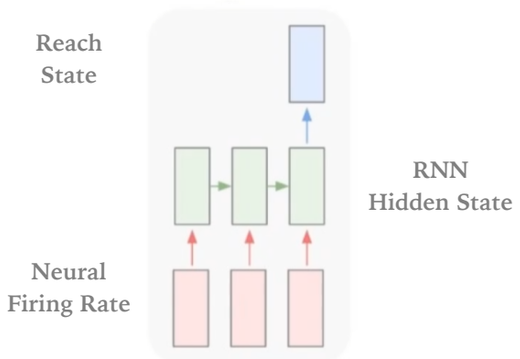
Estimate the class probability base on class prior p(y) and estimation of the class conditioned distribution p(x|y), with naive assumption that each feature dimension is independent given the class label.

$$P(\mathbf{x}|y) = \prod_{\alpha=1}^{d} P(x_\alpha|y), \text{where } x_\alpha = [\mathbf{x}]_\alpha \text{ is the value for feature } \alpha$$



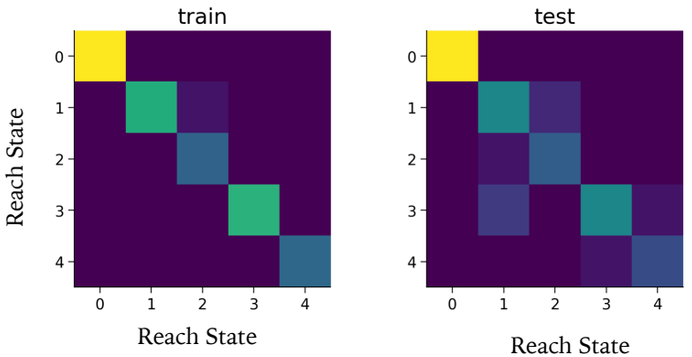
RNN Classifier

Many-to-on RNN Classifier to predict the reach target by integrating all observed neural firing rate from time 1 to t through the RNN hidden state.



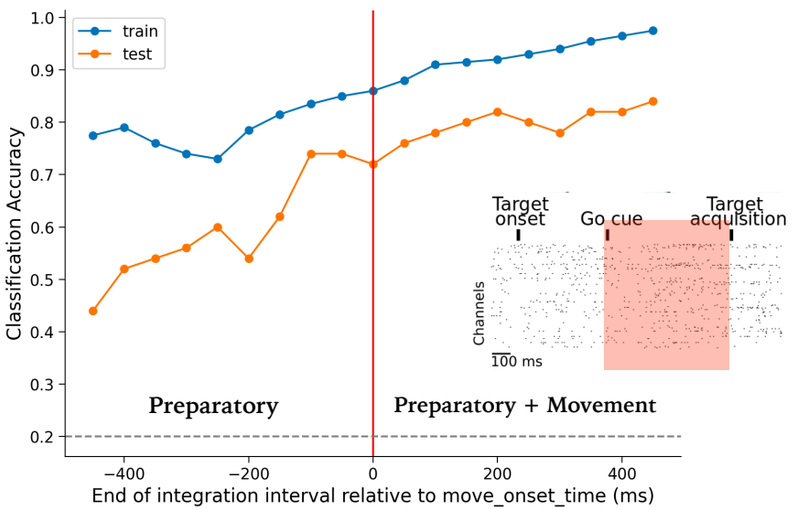
Model Evaluation

Confusion Matrix



Classification Accuracy

(As a function of integration window)



Classification Problem: Decode Reaching Target With Neural Firing Rate

Neural Decoding.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share J

Table of contents

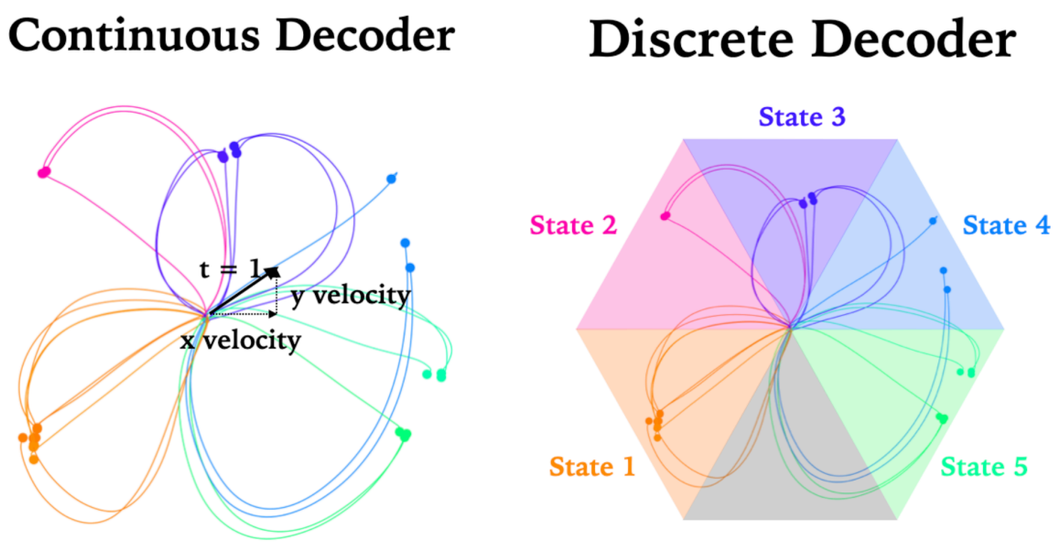
+ Code + Text

RAM Disk

- RNN(GRU): many-to-many
- Training RNNRegressor
- Pre-trained RNNRegressor
- Evaluate the Model performance
- **Discrete Decoder (Classification)**
- Machine Learning Classification
- Dimensionality Reduction (Bin Neural Response)
- Decoding targert with binned neural response
- Speed-accuracy trade-off for decoding target
- Neural Network Classification
- Decoding Challenge - RNN: many-to-one
- Define RNN Model: RNNClassifier
- Define RNN training procedure
- RNNClassifier prediction helper function
- Prepare the Train and Test Dataset
- Define the parameter and start training
- Solution to the Decoding Challenge
- Evaluate the Model performance

Discrete Decoder (Classification)

To predict the intended movement target, the discrete decoder integrate neural activity throughout time to predict the movement target. In this case, the decoder solves a classification problem.



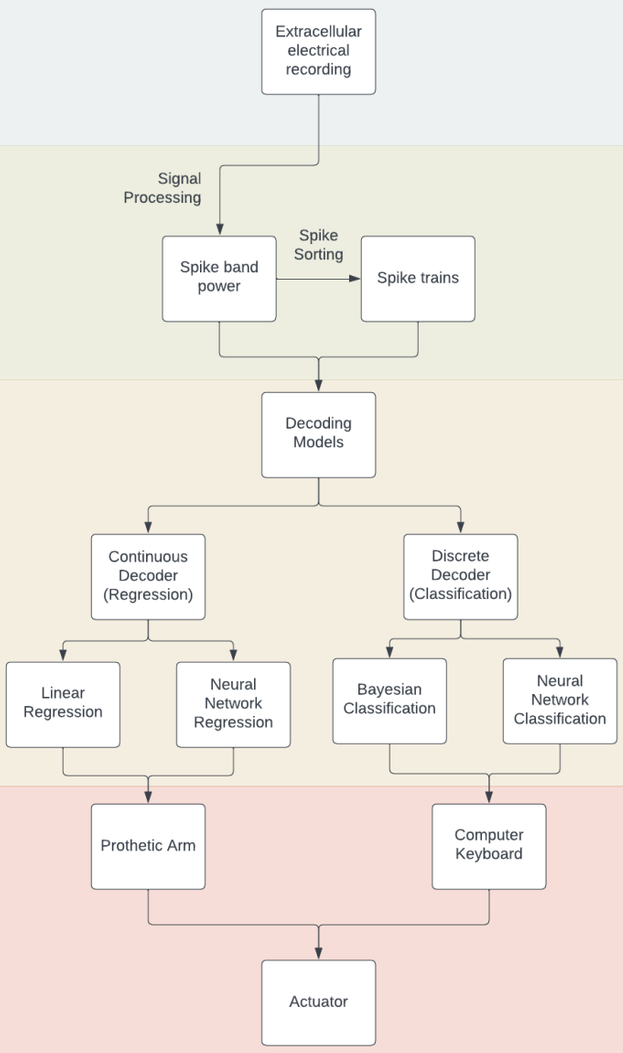
Machine Learning Classification

```
[29] from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import confusion_matrix
     from sklearn.metrics import accuracy_score
```

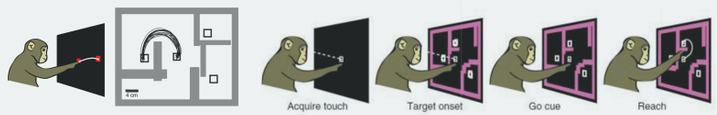
0s

Dimensionality Reduction (Bin Neural Response)

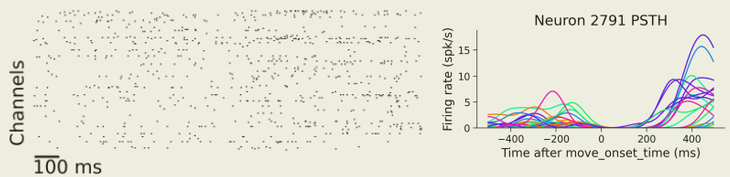
Self-Guided Coding Session



Task: delayed center-out reaching task

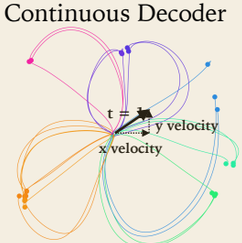


Neural Feature: Spike Trains



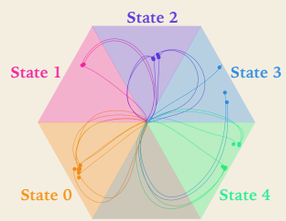
Decoding Objective

Continuous Decoder



Regression problem: predict the hand x and y velocity in each time bin.

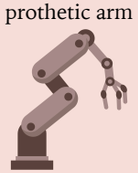
Discrete Decoder



Classification problem: predict the reaching target (discretized reaching angle).

Application

Movement with prothetic arm



Typing with computer keyboard