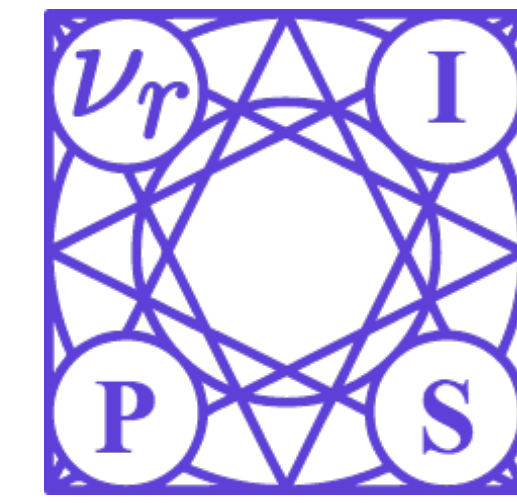




Dynamic Ensemble Modeling Approach to Nonstationary Neural Decoding in Brain-Computer Interfaces

Yu Qi, Bin Liu, Yueming Wang, Gang Pan

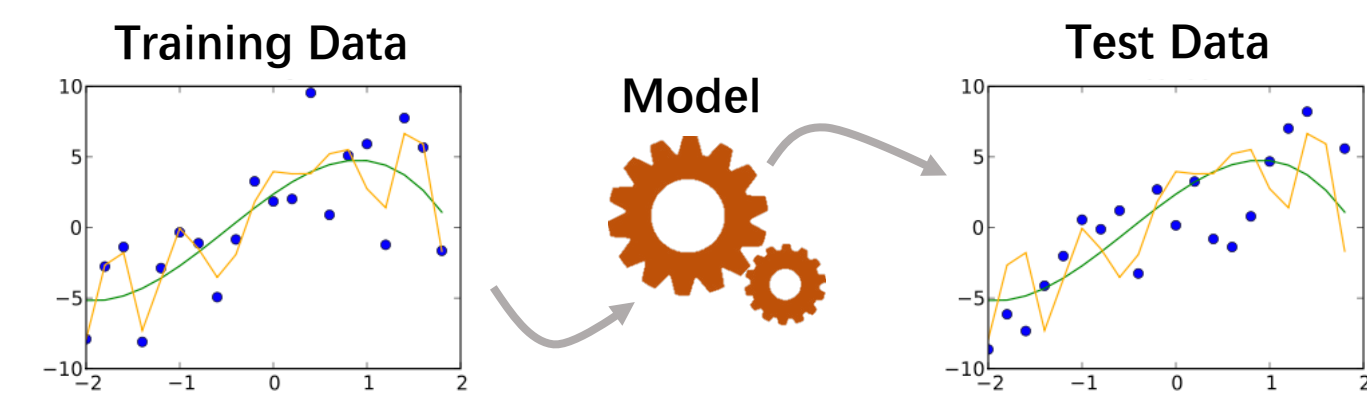
Zhejiang University, Nanjing University of Posts and Telecommunications
qiyu@zju.edu.cn, bins@ieee.org, ymingwang@zju.edu.cn, gpan@zju.edu.cn



NeurIPS
2019

Problem and objective

Problem: Dynamic world and static models

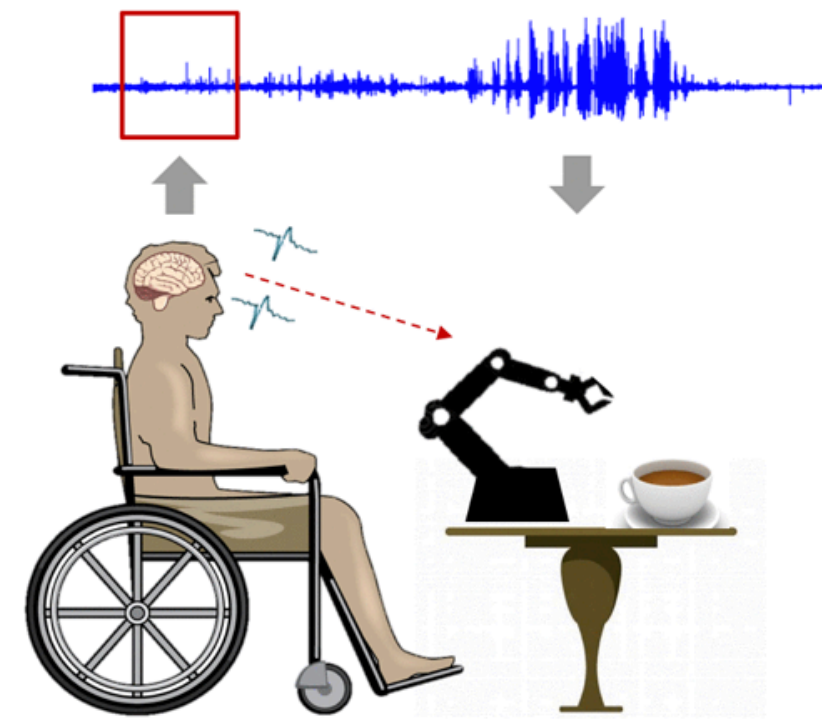
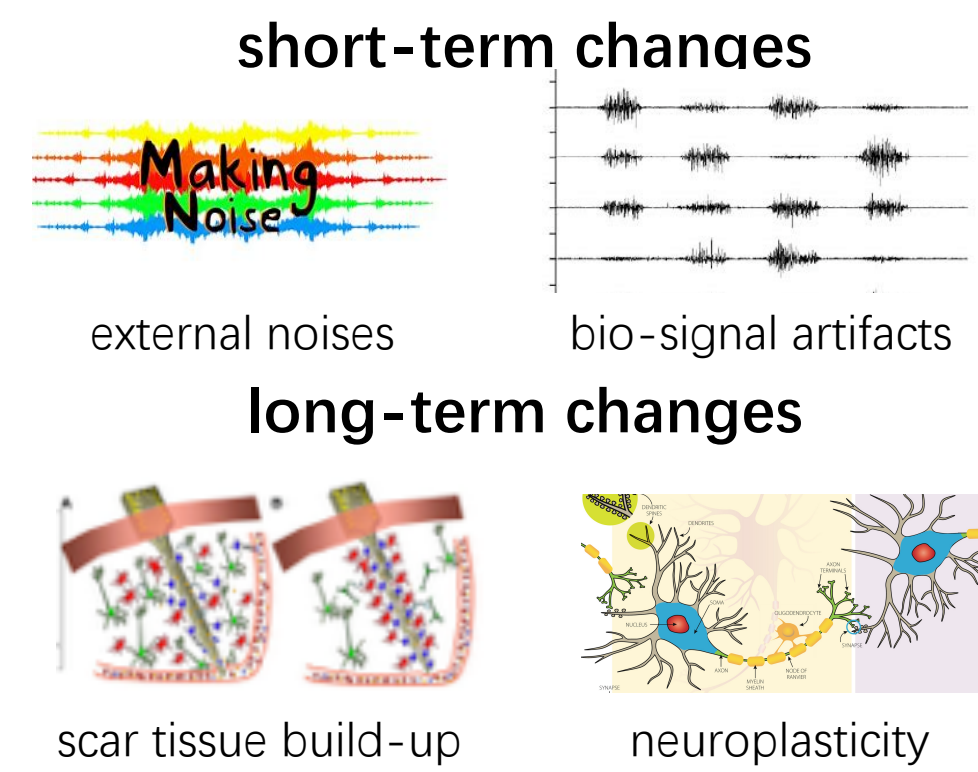


Typical model training pipeline

- use a static model
- assumes the data distribution is fixed and stable in time

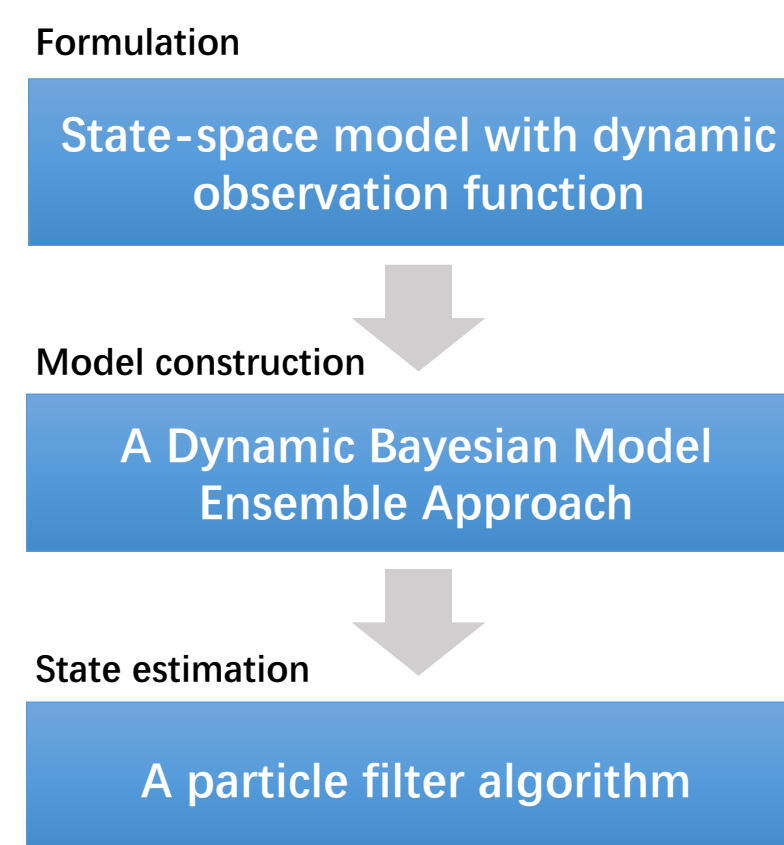
It would fail if the assumption is not satisfied ...

Brain signals are typical nonstationary data

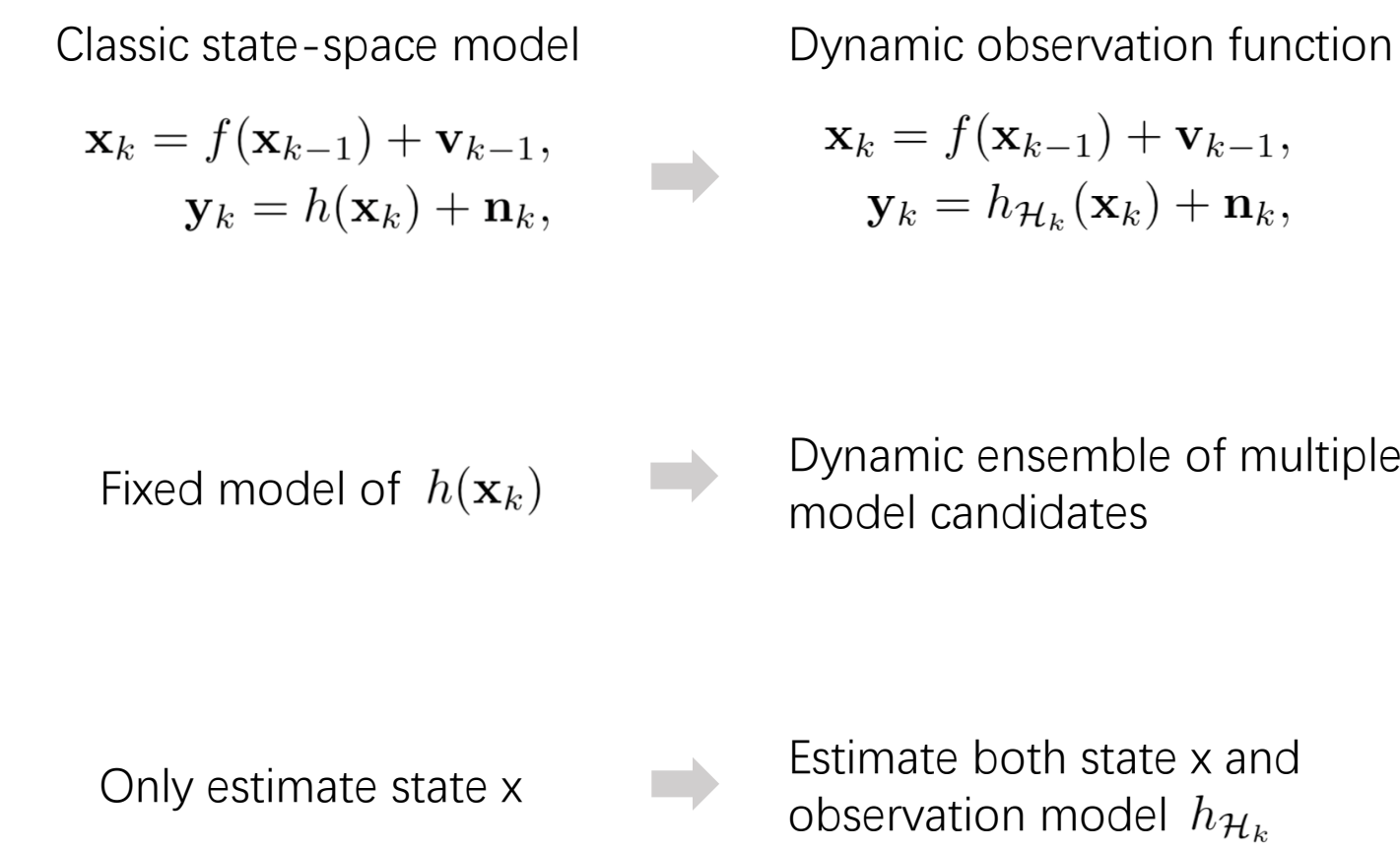


Main insights

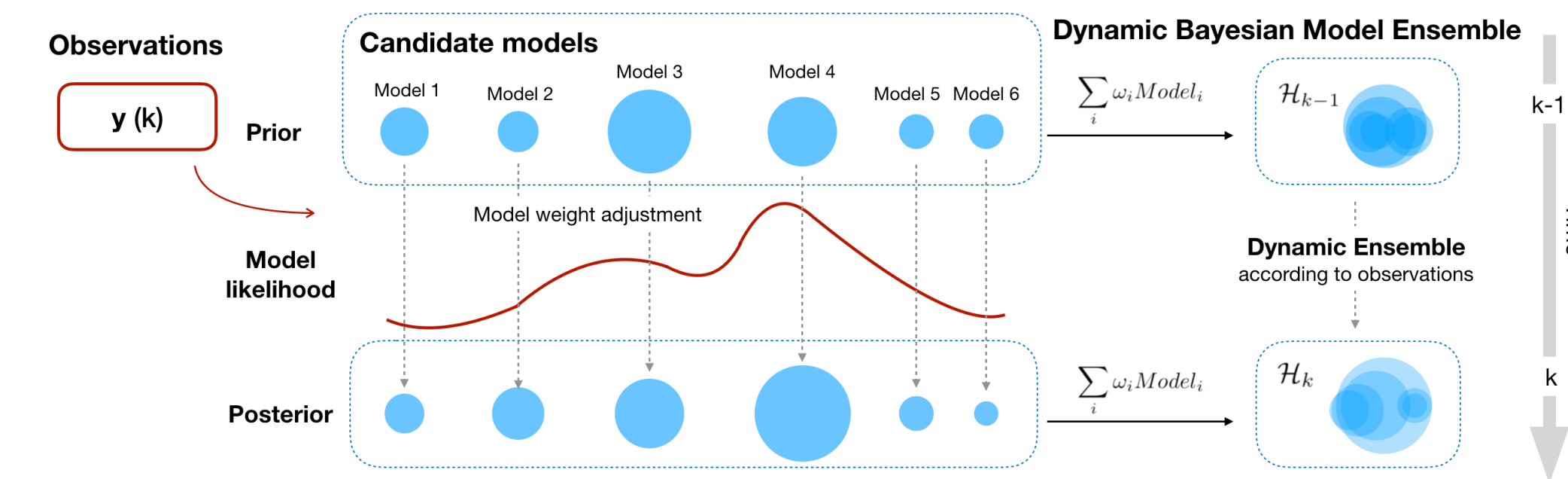
Problems to solve



Proposed solution



Method: Dynamic model ensemble (DyEnsemble)



$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1}, \quad \mathbf{y}_k = h_{\mathcal{H}_k}(\mathbf{x}_k) + \mathbf{n}_k, \quad \Rightarrow \quad p(\mathbf{x}_k | \mathbf{y}_{0:k}) = \sum_{m=1}^M p(\mathbf{x}_k | \mathcal{H}_k = m, \mathbf{y}_{0:k}) p(\mathcal{H}_k = m | \mathbf{y}_{0:k})$$

the posterior probability of \mathbf{x} given $\mathcal{H}_k = m$

the posterior probability of the m -th hypothesis

Step 1. Particle based estimation of $p(\mathbf{x}_k | \mathcal{H}_k = m, \mathbf{y}_{0:k})$

Step 2. Particle based estimation of $p(\mathcal{H}_k = m | \mathbf{y}_{0:k})$

$$\omega_{m,k}^i \propto \omega_{k-1}^i p_m(\mathbf{y}_k | \mathbf{x}_k^i), \quad \sum_{i=1}^{N_s} \omega_{m,k}^i = 1.$$

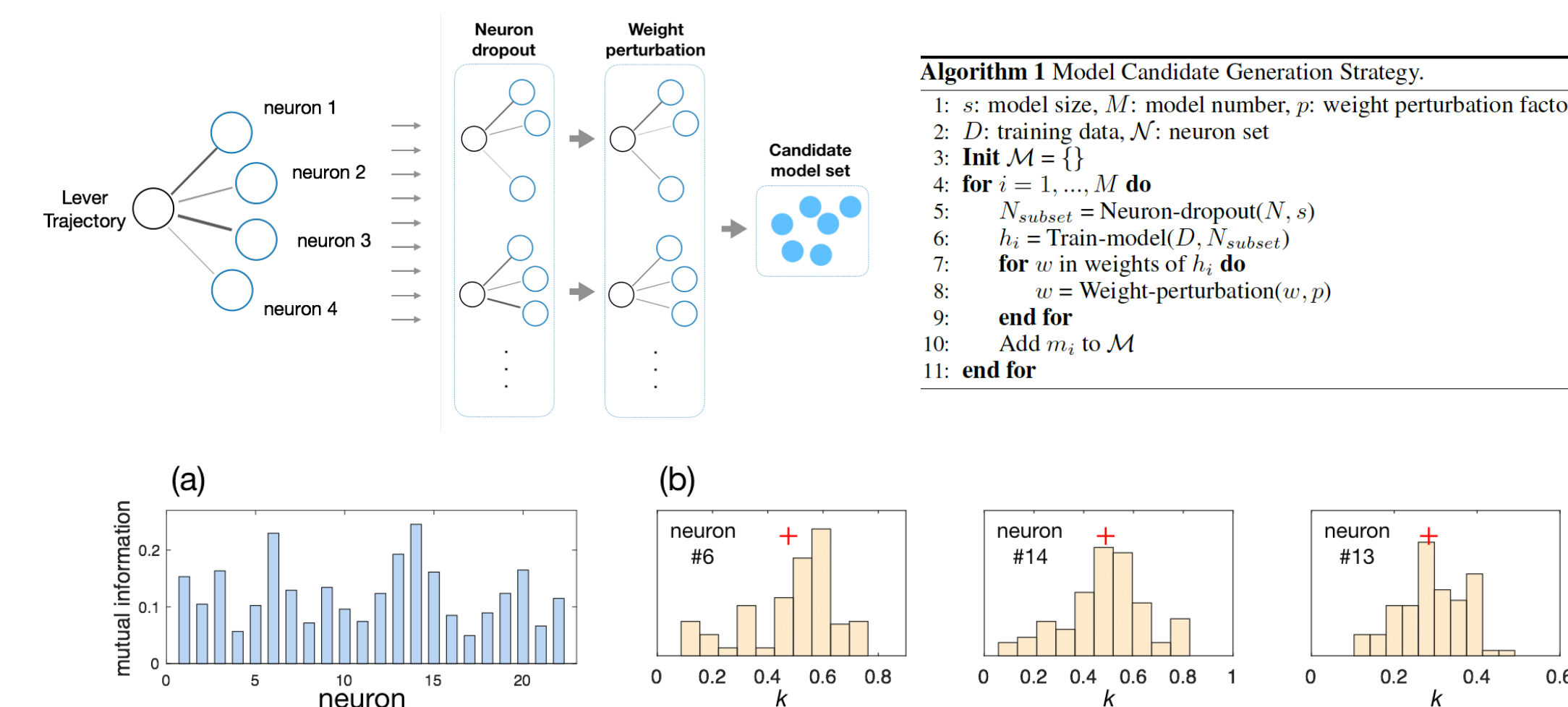
$$p(\mathcal{H}_k = m | \mathbf{y}_{0:k-1}) = \frac{p(\mathcal{H}_{k-1} = m | \mathbf{y}_{0:k-1})^\alpha}{\sum_{j=1}^M p(\mathcal{H}_{k-1} = j | \mathbf{y}_{0:k-1})^\alpha}$$

$$p(\mathbf{x}_k | \mathcal{H}_k = m, \mathbf{y}_{0:k}) \approx \sum_{i=1}^{N_s} \omega_{m,k}^i \delta(\mathbf{x}_k - \mathbf{x}_k^i),$$

$$p_m(\mathbf{y}_k | \mathbf{y}_{0:k-1}) \approx \sum_{i=1}^{N_s} \omega_{k-1}^i p_m(\mathbf{y}_k | \mathbf{x}_k^i)$$

$$p(\mathcal{H}_k = m | \mathbf{y}_{0:k}) = \frac{p(\mathcal{H}_k = m | \mathbf{y}_{0:k-1}) p_m(\mathbf{y}_k | \mathbf{y}_{0:k-1})}{\sum_{j=1}^M p(\mathcal{H}_k = j | \mathbf{y}_{0:k-1}) p_j(\mathbf{y}_k | \mathbf{y}_{0:k-1})}$$

Candidate model generation strategy

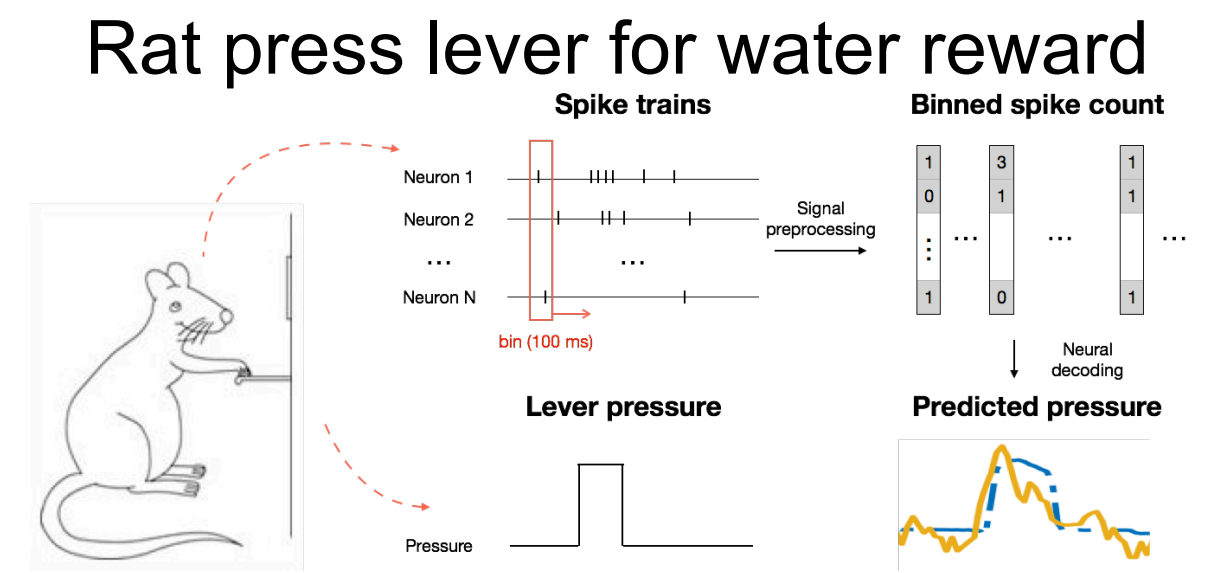


Algorithm 1 Model Candidate Generation Strategy.
1: s : model size, M : model number, p : weight perturbation factor
2: D : training data, N : neuron set
3: **Init** $\mathcal{M} = \{\}$
4: **for** $i = 1, \dots, M$ **do**
5: $N_{subset} = \text{Neuron-dropout}(N, s)$
6: $h_i = \text{Train-model}(D, N_{subset})$
7: **for** w in weights of h_i **do**
8: $w = \text{Weight-perturbation}(w, p)$
9: **end for**
10: Add m_i to \mathcal{M}
11: **end for**

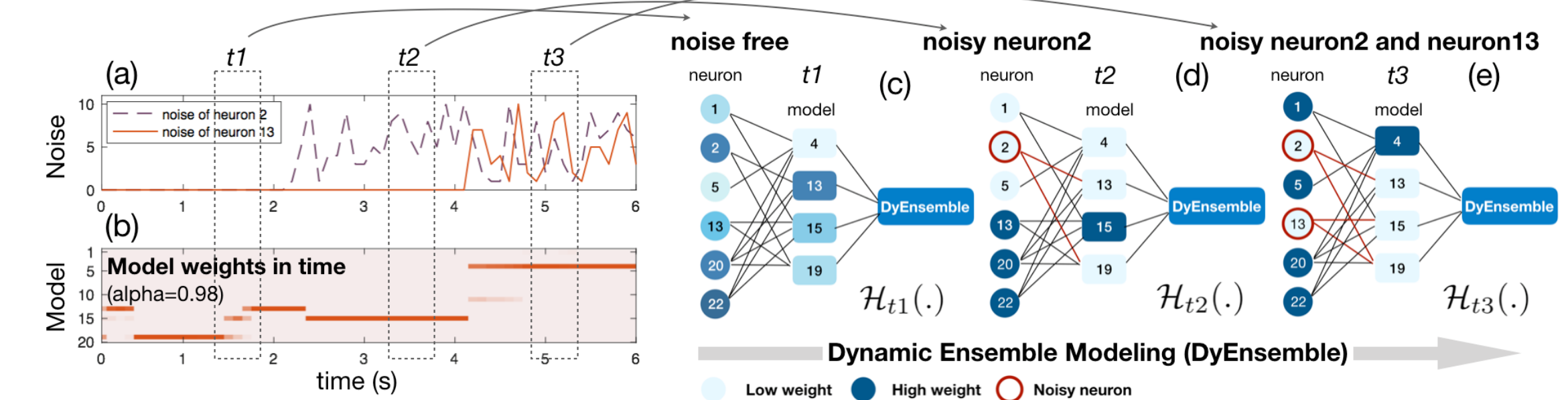
Experiments

Neural signal dataset

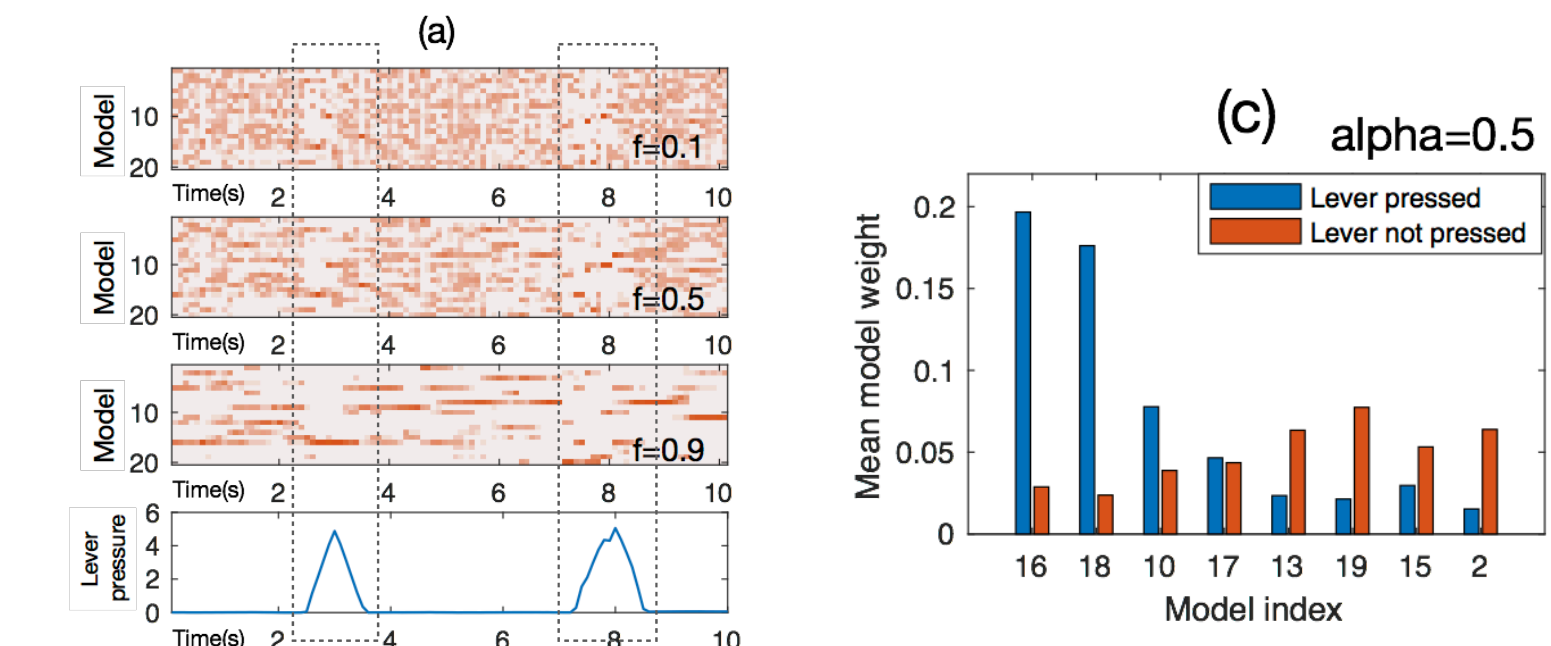
	# neuron	train data	test data
Rat 1	22	200 s	100 s
Rat 2	58	200 s	100 s



Model switching along with changing noises



Model switching along with task behaviors



During lever pressing, only a certain set of candidate models are selected.

Comparison with other approaches

Table 1: Correlation coefficient with different numbers of noisy neurons.

Method	Rat 1			Rat 2		
	Original	Noisy (#2)	Noisy (#4)	Original	Noisy (#2)	Noisy (#4)
Kalman filter	0.777±0.000	0.696±0.012	0.560±0.009	0.798±0.000	0.580±0.039	0.381±0.093
LSTM	0.753±0.017	0.687±0.033	0.617±0.045	0.846±0.021	0.551±0.127	0.338±0.050
Dual decoder	0.779±0.000	0.694±0.010	0.575±0.013	0.803±0.000	0.585±0.025	0.387±0.030
DyEnsemble (w/o P, w/o D)	0.776±0.002	0.684±0.014	0.558±0.009	0.798±0.002	0.579±0.066	0.377±0.155
DyEnsemble (P(0.1), w/o D)	0.780±0.008	0.711±0.004	0.557±0.035	0.780±0.006	0.665±0.024	0.472±0.080
DyEnsemble-2 (P(0.1), D(2))	0.799±0.012	0.735±0.006	0.583±0.090	0.788±0.009	0.633±0.064	0.516±0.092
DyEnsemble-5 (P(0.1), D(5))	0.775±0.015	0.739±0.021	0.671±0.039	0.803±0.009	0.584±0.035	0.596±0.035

* w/o: without; P(k): weight perturbation with $p=k$; D(l): neuron dropout with l neurons dropped.