Learning Low-Dimensional Latent Dynamics in Neural Population Activity Using Kalman Filtering and Expectation-Maximization

Sara Pirasteh

December 11th 2025

Abstract

Neural population recordings often contain structured low-dimensional dynamics despite being measured through high-dimensional, noisy spike-count features. In this project, I fit a linear Gaussian state-space model (LGSSM) to the neural activity recorded during a handwriting BCI task. I implemented Kalman filtering and Rauch-Tung-Striebel smoothing for inference and used Expectation-Maximization (EM) to learn the dynamics and observation parameters. I evaluated three approaches: (1) Kalman filtering with random parameter initialization, (2) Kalman filtering with PCA-based initialization, and (3) EM learning initialized from random parameters. I measured reconstruction accuracy using normalized RMSE (NRMSE) and compared performance across train/validation/test sentences. The results show that PCA initialization yields strong reconstructions, random initialization can fail dramatically, and EM can substantially improve a poor initialization however, it may converge to worse local optima than PCA. Overall, this project highlights both the usefulness and the challenges of EM for learning latent dynamical models in neural data.

1 Introduction

A central idea in computational neuroscience is that many neurons recorded simultaneously do not evolve independently; instead, they reflect a smaller number of shared underlying factors (e.g. intention, movement planning, or population-level control signals). This motivates modeling neural activity as the output of a latent dynamical system. In brain-computer interface (BCI) settings, such latent models are useful because they provide a compressed representation that can reduce noise, reveal structure, and support decoding.

In this project, I model neural activity using a linear Gaussian state-space model (LGSSM). The model assumes that a low-dimensional latent state x_t evolves linearly over time, and the recorded neural activity vector y_t is a noisy linear function of the latent state. This is the classic setting where Kalman filtering provides optimal inference and EM provides a tractable learning algorithm.

A key practical issue is that EM is not convex; it is sensitive to initialization. Therefore, I compare:

1. Random initialization + Kalman inference
2. PCA initialization + Kalman inference
3. Random initialization + EM learning

The primary outcome of my project is the reconstruction accuracy: given y_t, infer latent x_t and reconstruct $\hat{y}_t$. I also visualize latent trajectories to illustrate what the learned low-dimensional states look like.

2 Dataset and Preprocessing

2.1 Dataset Structure

The neural recordings consist of multi-neuron time series segmented by sentence boundaries, as provided in the original dataset [1]. Each sentence has a variable duration, motivating sentence-aware preprocessing and time-series modeling. The provided dataset contains:

- `neuralActivityTimeSeries`: A long concatenated time series of neural activity of shape (T_{total}, P), where P is the number of recorded features/neurons.
- `numTimeBinsPerSentence`: A vector of sentence lengths, used to split the long array into individual trials/sentences.

Let S be the number of sentences, and let T_s denote the number of time bins in sentence s. After splitting, each sentence becomes:

$$Y^{(s)} \in \mathbb{R}^{T_s \times P}.$$

This structure is important because each sentence is a coherent behavioral episode; that is why it is not recommended to mix or shuffle time bins across sentences in preprocessing.

2.2 Gaussian Smoothing

Neural spike counts (or binned activity) can be sparse and noisy from bin to bin. To reduce high-frequency noise while preserving slower population trends, I applied a Gaussian filter along time:

$$\tilde{Y}^{(s)} = \text{GaussianSmooth}(Y^{(s)}, \sigma),$$

with $\sigma = 1.0$ and smoothing performed independently per neuron (axis 0 is time). Smoothing produces real-valued outputs even when input data are integer counts. This is expected and appropriate for a Gaussian observation model as well as the Kalman filter.

2.3 Downsampling

The dataset is large, and Kalman/EM inference scales linearly with time length. To reduce runtime, I downsampled each sentence by taking every m-th bin (here $m = 5$):

$$\bar{Y}^{(s)} = \tilde{Y}^{(s)}[0 : m : T_s].$$

Downsampling preserves sentence boundaries and reduces sequence length while maintaining temporal order.

2.4 Neuron selection by variance

Because modeling all recorded dimensions can be expensive and time consuming, I selected the top $P' = 30$ neurons by variance computed using the training set:

$$\mathrm{Var}(j) = \mathrm{Var}\left(\{y_{t,j}\}_{t \in \mathrm{train}}\right).$$

This retains neurons with substantial dynamic range and reduces computation. Importantly, neuron selection is computed using training data only to avoid leaking test-set information.

2.5 Train/val/test split

I split sentences into training, validation, and test sets using fixed random seeds for reproducibility:

- Train: 80% of sentences
- Validation: 10%
- Test: 10%

All preprocessing steps (mean subtraction and parameter learning) are based on the training set to avoid leakage.

3 Model: Linear Gaussian State-Space Model

3.1 State and observation equations

The LGSSM is:

$$x_{t+1} = Ax_t + w_t, \qquad w_t \sim \mathcal{N}(0, Q), \tag{1}$$

$$y_t = Cx_t + v_t, \qquad v_t \sim \mathcal{N}(0, R), \tag{2}$$

where:

- $x_t \in \mathbb{R}^k$ latent state ($k = 20$),
- $y_t \in \mathbb{R}^{P'}$ observed neural activity ($P' = 30$),
- A latent dynamics, Q process noise,
- C latent to observed space, R observation noise.

3.2 Mean-centering

Because the observation model is zero-mean (no explicit bias term), I subtract the training mean μ:

$$y_t^c = y_t - \mu.$$

Reconstruction is computed in centered space and then shifted back:

$$\hat{y}_t = \hat{y}_t^c + \mu.$$

This prevents the model from wasting latent capacity to represent a constant offset.

4 Inference: Kalman Filter and RTS Smoother

4.1 Why filtering and smoothing?

Given parameters (A, C, Q, R) and data $y_{1:T}$, the goal is to infer $p(x_t \mid y_{1:T})$.

- The Kalman filter computes $p(x_t \mid y_{1:t})$ using only past and current data.
- The RTS smoother refines this using future evidence, yielding $p(x_t \mid y_{1:T})$.

Smoothing matters for learning (EM) because the M-step uses expected second moments from the full posterior.

4.2 Reconstruction

At each time step, the model predicts the observation as:

$$\hat{y}_t^c = C\hat{x}_t,$$

where $\hat{x}_t$ is the filtered latent mean. This yields a reconstructed neural signal in the same feature space as the original data.

5 Learning: Expectation-Maximization

5.1 Objective

EM maximizes the log-likelihood of observations:

$$\log p(y_{1:T} \mid \theta),$$

but direct maximization is hard due to the latent states. EM alternates:

- E-step: compute expectations under $p(x_{1:T} \mid y_{1:T}, \theta^{old})$ via Kalman smoothing.
- M-step: update θ using closed-form solutions for linear Gaussian models.

5.2 E-step Moments

The smoother provides:

$$\mathbb{E}[x_t], \quad \mathbb{E}[x_t x_t^\top], \quad \mathbb{E}[x_t x_{t-1}^\top].$$

These expected sufficient statistics are used to update A, C, Q, R.

5.3 M-step updates

- A is estimated by regressing x_t on x_{t-1} in expectation.
- C is estimated by regressing y_t on x_t in expectation.
- Q and R are residual covariances under the learned regressions.

EM is guaranteed not to decrease the data likelihood each iteration, but it may converge to a local optimum.

6 Initialization Strategies

6.1 Why initialization matters

Because the latent dimension and parameters create a non-convex landscape, different starting points can lead to different solutions. This is especially important in neural data, where noise can cause many plausible latent explanations.

6.2 Random initialization

I initialized:

- $C \sim \mathcal{N}(0, 1)$,
- $A = 0.5I + 0.1\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$,
- $Q = I$,

which often produces poor reconstructions because C may project the latent state into an arbitrary subspace unrelated to the neural activity structure.

6.3 PCA-based initialization

PCA initialization attempts to discover a good observation subspace immediately. With centered training data X_c, compute SVD:

$$X_c = U\Sigma V^\top.$$

Then:

$$C_{\text{PCA}} = V_{1:k}^\top, \qquad x_t \approx X_c C_{\text{PCA}}.$$

Finally, estimate A and Q by fitting a linear dynamical model in the PCA latent space. This produces strong reconstructions because C_{PCA} is aligned with the dominant variance directions of the data.

6.4 EM initialized from random parameters

I ran EM starting from random initialization to test whether EM could recover structure. Empirically, EM substantially improved reconstruction error compared to pure random initialization, but did not reach the PCA-initialized performance in my runs.

7 Evaluation Metric

7.1 NRMSE

I use normalized root mean squared error:

$$\text{NRMSE} = \sqrt{\frac{\sum_t \|y_t - \hat{y}_t\|_2^2}{\sum_t \|y_t\|_2^2}}.$$

NRMSE is scale-invariant and summarizes reconstruction accuracy across all time steps and features.

8 Results

8.1 Qualitative reconstructions

Figures below show an example test sentence reconstruction for neuron 0 under each method.

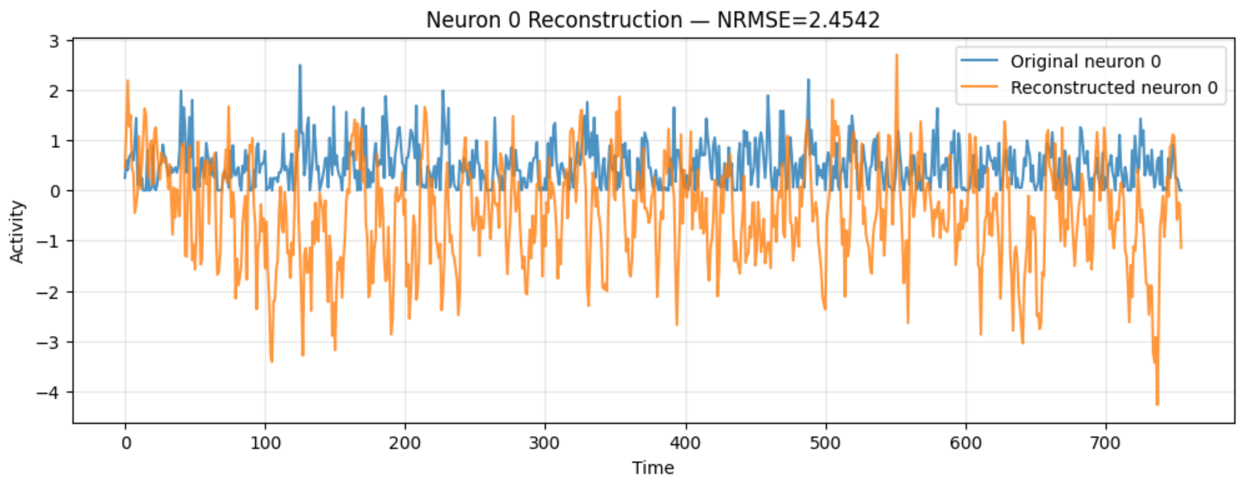

Figure 1: Random init + Kalman: Failure case with large NRMSE (e.g., ≈ 2.45).

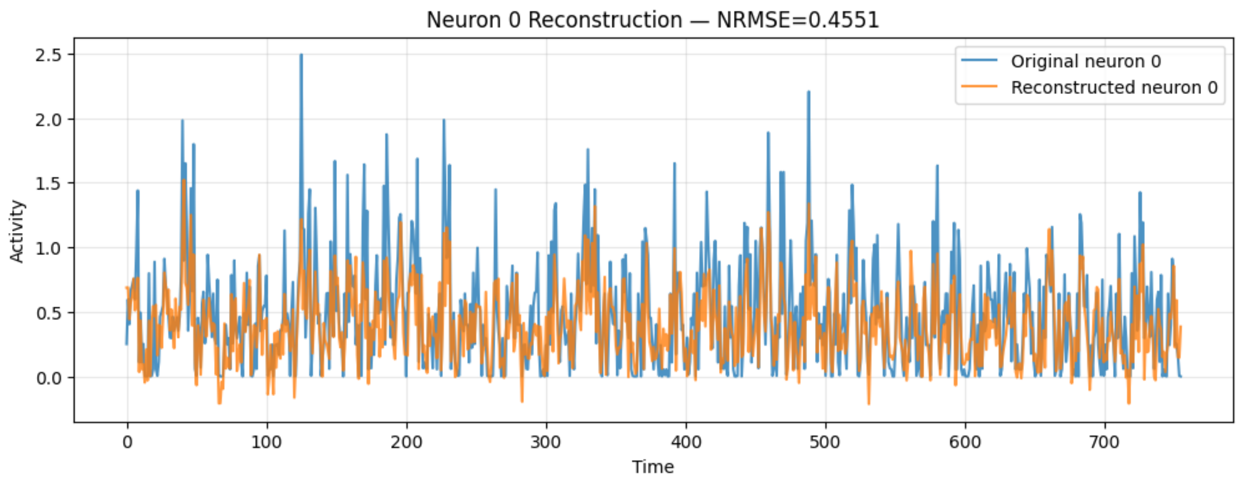

Figure 2: PCA init + Kalman: Strong reconstruction with NRMSE ≈ 0.45.

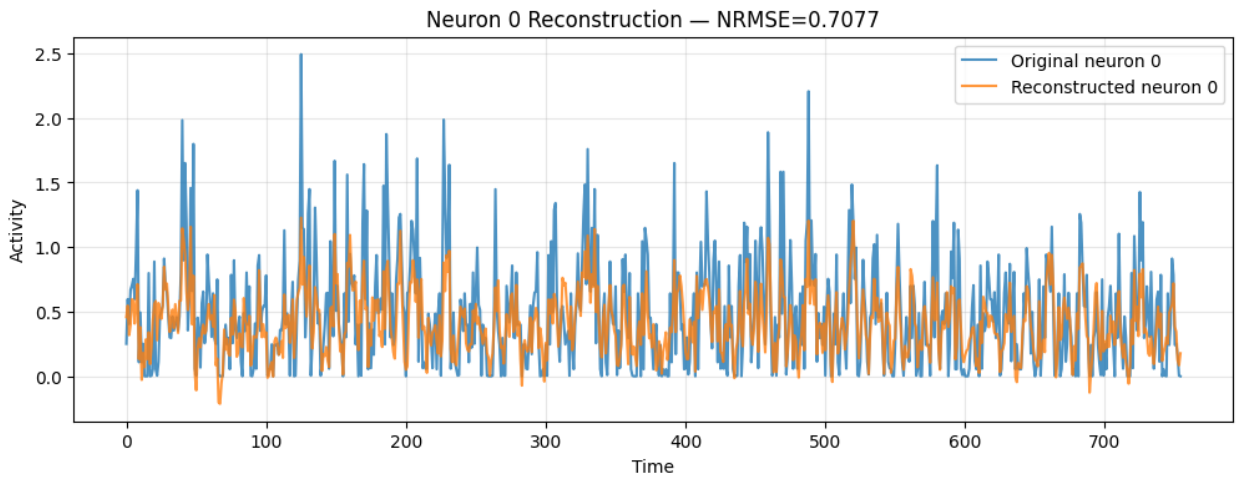

Figure 3: EM Reconstruction improves substantially relative to random init (e.g., NRMSE ≈ 0.70).

8.2 EM convergence

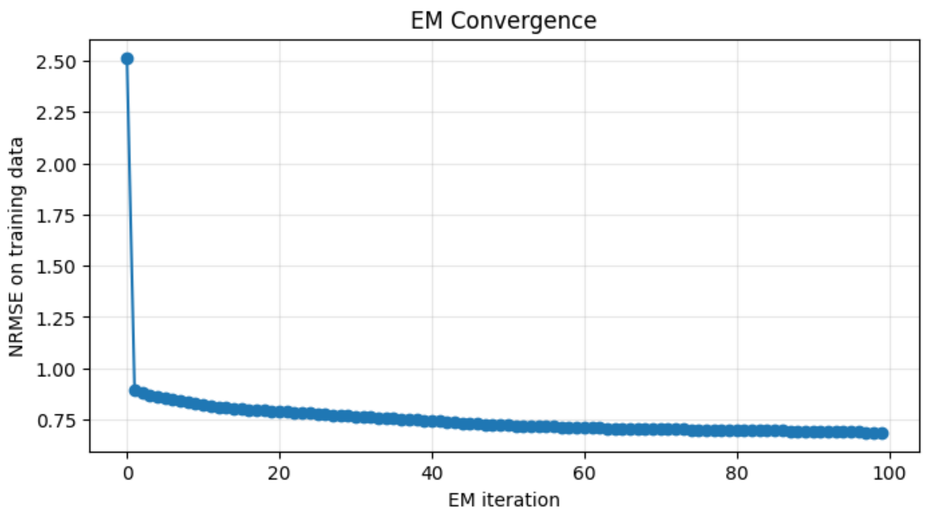

Figure 4: EM training objective proxy (NRMSE vs iteration). Large early improvement followed by gradual stabilization.

8.3 Test-set distribution

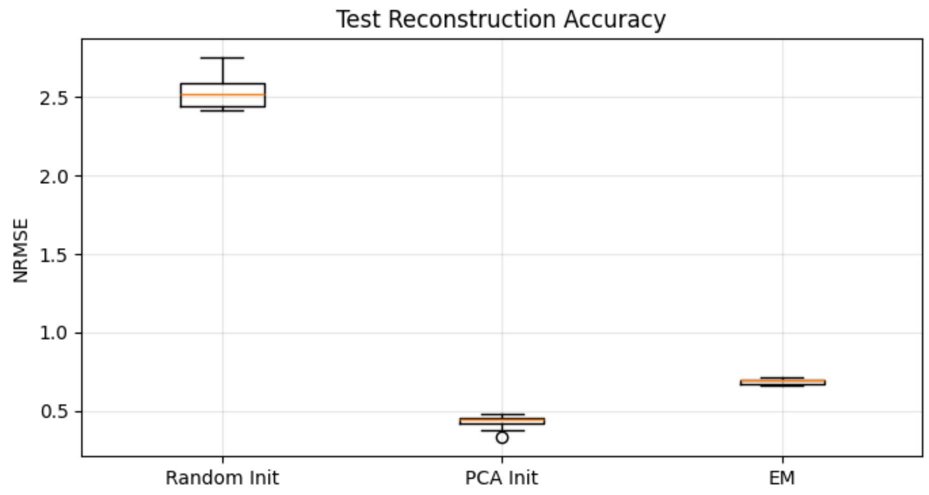

Figure 5: Test reconstruction accuracy across sentences for Random init, PCA init, and EM(Random).

8.4 Interpreting the result pattern

The results follow a consistent story:

- Random init is unstable: C may not align with the data, producing poor reconstructions even if the filter equations are correct.
- PCA init is strong: It immediately captures the main data subspace, so reconstruction is good even without EM.
- EM improves random init: EM learns a better C and dynamics, reducing NRMSE dramatically, but it can remain worse than PCA because it converges to a local optimum.

9 Limitations

This project uses a linear Gaussian model, which is a simplifying assumption. Limitations include:

- Real neural activity may be non-Gaussian and nonlinear.
- The latent dimension k is a hyperparameter; different k changes performance.
- EM is sensitive to initialization and may converge to poor local minima.
- Sentence-to-sentence nonstationarity may violate constant A, C, Q, R assumptions.

10 Conclusion

I implemented Kalman filtering/smoothing and EM learning for a latent dynamical model of neural activity. PCA initialization produced the best reconstructions because it aligns the observation matrix with dominant variance structure. Random initialization can fail badly, but EM can dramatically improve it by learning better parameters. These results show both the strength of latent dynamical models for denoising/reconstruction and the importance of careful initialization when using EM.

Reproducibility Notes

Key settings used in experiments:

- Smoothing: Gaussian filter with $\sigma = 1.0$
- Downsampling: factor $m = 5$
- Selected neurons: top 30 by training variance
- Latent dimension: $k = 20$
- Train/val/test split: fixed random seed
- EM iterations: up to 100, early stopping by tolerance of 10^{-10}

Code Availability

The full implementation of all preprocessing steps, Kalman filtering, RTS smoothing, EM learning, and evaluation is available at: `https://github.com/sarapirast/Learning-Latent-Dynamics-using-Kalman`.

References

[1] Francis R. Willett, Donald T. Avansino, Leigh R. Hochberg, Jaimie M. Henderson, and Krishna V. Shenoy. High-performance brain-to-text communication via handwriting. *Nature*, 593(7858):249–254, 2021.