EE638 Project Report

Shashwat Shukla
150260025

All code for this project was written in Python 2.7 and can be accessed at the following Github repository:
https://github.com/ShashShukla/Place-Cell-Decoding-from-Spike-train-data

Introduction

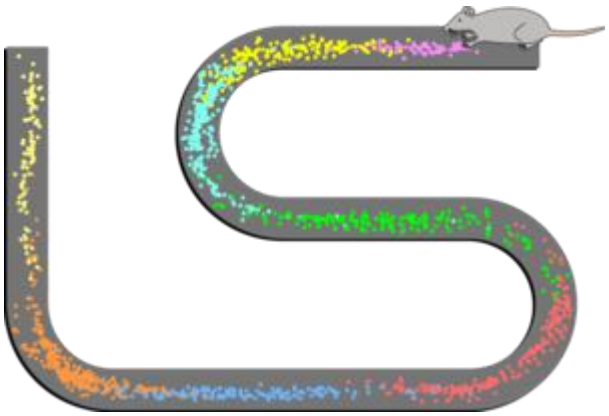
The human brain has over a trillion neurons, with thousands of synaptic connections per neuron. Neurons communicate with each other via spikes (i.e action potentials). It is still unclear what aspects of the spike-train carry useful information for communication and how incoming sensory information modulates spike-train outputs. Furthermore, neurons have been found to fire stochastically, often with Poisson statistics (over short enough intervals). One dominant theory is that the mean-rate of Poisson firing varies as a function of the input to the neuron. This then leads to the modeling of spike-trains as a point process and the use of Point-process theory and inhomogeneous Poisson processes in studying their behavior.

In recent years, the number of neurons that we can record from in live animals such as rats and monkeys while they perform various cognitive tasks has increased substantially, with head-mounted microdrive arrays offering up to 500 simultaneous neuron recording channels. Given this spike-train data, we would like to infer/decode the "state" of the organism. This has applications in Computational Neuroscience and in applied fields such as Brain-Computer Interfacing. Inference on such large data-streams is difficult with conventional Point-process models. The paper by Brown et al. [1] proposes an efficient State-space model instead, that allows for computationally tractable decoding from spike-train data. The model has an observation model that belongs to the Exponential family of distributions, making analytic approaches more tractable. Further work has been done using this State-space framework over the last two decades [2][3] using other models from the Exponential family, but will not be pursued here.

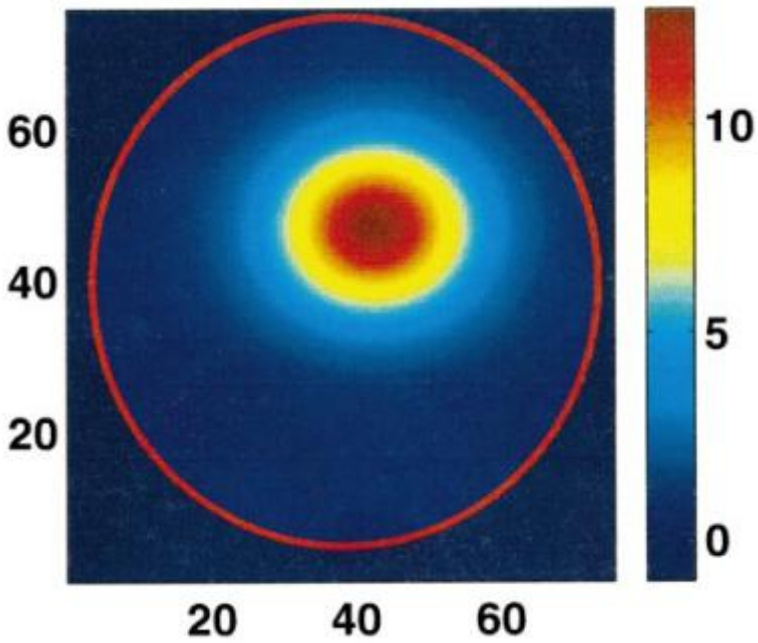
Problem formulation

In particular, Brown et. al propose to decode the spatial position of a rat from the spike-train recordings from CA1 place cells in the hippocampus of the rat. A place cell is a kind of pyramidal neuron in the hippocampus that becomes active when an animal enters a particular location in its environment. The firing of place cells is timed in relation to local theta waves, a process termed phase precession. Hence they model the instantaneous firing rate of the inhomogeneous Poisson output of these place cells as a function of the animal's position in the environment and phase of the theta rhythm, which is thus the observation model. Note that they assume a product form for the contributions of spatial location and theta phase, and hence don't model the interdependent nature of observed phase precision. It is to be noted that the spatial response of the place cells is close to Gaussian. Further, they found that the contribution from the theta rhythm is not as significant as that from spatial location and hence in this project, **I will only be modeling the spatial response of the place cells**. This is then the observation model. They further model the animal's path during

foraging as a Gaussian random walk, which is then the state-process model. With this state-process and observation model, they then propose a method of decoding, in which they derive a nonlinear, recursive causal filter algorithm for predicting the animal's position from place cell ensemble firing patterns. The method they derive is a specific instance of the **Iterated Extended Kalman Filter (IKEF)**.



Spatial firing patterns of 8 place cells recorded from the CA1 layer of a rat. Dots indicate positions where action potentials were recorded, with color indicating which neuron emitted that action potential. [4]



Pseudocolor map depicting the receptive field of a particular place cell [1]

The following is hence the formulation of the problem stated formally:

Let (0, T] be the foraging time of the animal and $x(t) = [x_1(t), x_2(t)]'$ be the 2x1 vector denoting the animal's position at time t. Further, let there be C place cells in total and let t_i^c denote the spike recorded from cell c at time t_i in (0, T].

State equation:

As noted before, the path of the rat is modeled as a Gaussian random walk. Clearly, the covariance matrix scales with the time interval Δ_k (analogous to how in 1D random walks we have $\langle x^2 \rangle = Dt$). This leads to the following state equation:

$$x(t_k) - x(t_{k-1}) \sim N(0, W_x(\Delta_k))$$
$$W_x(\Delta_k) = \begin{bmatrix} \sigma_{x1}^2 & \rho\sigma_{x1}\sigma_{x2} \\ \rho\sigma_{x1}\sigma_{x2} & \sigma_{x2}^2 \end{bmatrix} \Delta_k$$

Observation equation:

The receptive fields of the place cells are modeled as Gaussian. This Gaussian function is then the instantaneous firing rate for the inhomogeneous Poisson process. Hence the firing rate of a particular place cell (indexed by c) at time t is given as:

$$\lambda^c(t \mid x(t), \xi^c) = \exp\left\{\alpha_c - \frac{1}{2}(x(t) - \mu_c)' W_c^{-1}(x(t) - \mu_c)\right\}$$

Here $\mu_c(t) = [\mu_{c,1}(t), \mu_{c,2}(t)]'$ is the 2x1 vector whose components are the x_1 and x_2 coordinates of the place field center, α_c is the location intensity parameter, $\xi^c = [\alpha_c, \mu_c, W_c]$. W_c is a scale matrix given as:

$$W_c = \begin{bmatrix} \sigma_{c,1}^2 & 0 \\ 0 & \sigma_{c,2}^2 \end{bmatrix}$$

Let $I(t_k) = [I_1(t_k), I_2(t_k), .. I_C(t_k)]'$ be the vector of indicator variables for the C place cells for time t_k. That is, $I_c(t_k)$ is 1 if there is a spike at t_k from cell c and 0 otherwise. Now, for a Poisson process, the waiting time for the next event/spike is given as an exponential function. That is, for a homogeneous Poisson process, if a spike was emitted at time t=0, then the probability density for the next spike occurring at time $t=\tau$ is given as $p(\tau) = \lambda \cdot \exp(-\lambda \cdot \tau)$. Here they simulate the process by binning it into time-intervals, each of size Δ_k. Any spike in $(t_k - \Delta_k, t_k)$ is assigned to t_k. Further, Δ_k is assumed to be small enough that only utmost one spike is emitted in a time-bin. Hence the probability of a spike at t_k is then the integral of the pdf over $(t_k - \Delta_k, t_k)$. Further note that they take the instantaneous firing rate to be approximately constant and equal to $\lambda^c(t \mid x(t), \xi^c)$ over this interval of length Δ_k. This then yields the following probability of a spike at t_k:

$$f(t_k) = \int_{t_{k-1}}^{t_k} p(t)dt \approx p(t_k)\Delta_k = \lambda(t_k)\Delta_k \cdot \exp(-\lambda(t_k)\Delta_k)$$

Note that without this assumption, we would be dealing with an inhomogeneous Poisson process, for which the pdf is given as:

$$p(t_k) = \lambda(t_k) \cdot \exp\left(- \int_{t_{k-1}}^{t_k} \lambda(t) dt \right)$$

This integral would not have an analytical expression in general, as is true here as well, for the particular Gaussian form of $\lambda^c\left(t \mid x(t), \xi^c\right)$.
In their formulation, they have written this as $f(t_k) = \lambda^c(x(t_k)) \cdot \exp\left(-\lambda^c(x(t_k))\right)$, meaning that they have taken $\lambda^c(x(t_k)) = \lambda(t)\Delta_k$, which is to say that $\lambda^c\left(t \mid x(t), \xi^c\right)$ is to be understood as the instantaneous firing rate multiplied by Δ_k. Hence they have absorbed a Δ_k in their formulation into the $\lambda^c\left(t \mid x(t), \xi^c\right)$ term itself. With this understanding, and noting that the responses of each of the C place cells is taken to be independent of the others, the observation equation for the model is:

$$f(I(t_k) \mid x(t_k), t_{k-1}) = \prod_{c=1}^{C} \left[\left\{ \lambda^c(x(t_k)) \right\}^{I_c(t_k)} \cdot \exp\left\{ -\lambda^c(x(t_k)) \right\} \right]$$

Decoding using IKEF

We first define some more notation. As defined in the previous section, let $I(t_k) = [I_1(t_k), I_2(t_k), .. I_C(t_k)]'$ be the vector of indicator variables for the C place cells for time t_k. Further, let $\zeta^k = [I_1(t_k), I_2(t_k), .. I_C(t_k)]'$ be the set of indicator variables until the k-th timestep. Then we have the posterior density for $x(t_k)$ as:

$$f\left(x(t_k) \mid \zeta^k\right) = \frac{f\left(x(t_k) \mid \zeta^{k-1}\right) \cdot f\left(I(t_k) \mid x(t_k)\right)}{f\left(I(t_k) \mid \zeta^{k-1}\right)}$$

Further, the one-step prediction density is given as:

$$f\left(x(t_k) \mid \zeta^{k-1}\right) = \int f\left(x(t_k) \mid \zeta^{k-1}\right) \cdot f\left(x(t_k) \mid x(t_{k-1})\right) dx$$

We will subsequently use the Laplace approximation to approximate the posterior for $x(t_k)$ as a Gaussian to yield a causal estimate, using the one-step prediction for $x(t_k)$ along with the observation model. The one-step prediction is obtained by propagating the causal estimate for $x(t_{k-1})$ through the state transition equations (it can be shown using the result for the marginal of a multivariate Gaussian that this is equivalent to computing the integral given above for the one-step prediction density. As the state-transition equations are linear, this yields a Gaussian again. Using this one-step prediction, we can compute the causal estimate of $x(t_k)$ via the Laplace approximation, with the required computation being the maximisation of a non-linear function. We perform this maximisation using the Newton Conjugate Gradient method, which is a second order method. In this way, we can iteratively compute causal estimates for the $x(t_k)$.

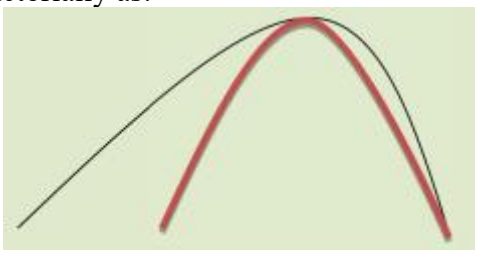
Stated mathematically, let the causal estimate for $x(t_{k-1})$ that uses samples upto timestep k-1 be denoted by $\hat{x}(t_{k-1} \mid t_{k-1})$. Then for the one-step prediction we have:

$$E\left(x(t_k) \mid \zeta^{k-1}\right) = \hat{x}(t_{k-1} \mid t_{k-1})$$
$$\text{var}\left(x(t_k) \mid \zeta^{k-1}\right) = W_x(\Delta_k) + W(t_{k-1} \mid t_{k-1}) = W(t_k \mid t_k)$$

For the Laplace approximation:

$$f\left(x(t_k)\mid \zeta^k\right) \propto f\left(x(t_k)\mid \zeta^{k-1}\right)\cdot f\left(I(t_k)\mid x(t_k)\right)$$
$$= \frac{1}{2\pi\left|W(t_k\mid t_{k-1})\right|^{1/2}} \times \exp\left\{-\frac{1}{2}\left(x(t_k)-\hat{x}(t_k\mid t_{k-1})\right)'\cdot W^{-1}(t_k\mid t_{k-1})\cdot\left(x(t_k)-\hat{x}(t_k\mid t_{k-1})\right)\right\} \times \prod_{c=1}^{C}\left[\left\{\lambda^c\left(x(t_k)\right)\right\}^{I_c(t_k)}\cdot \exp\left\{-\lambda^c\left(x(t_k)\right)\right\}\right]$$

This function is a product of a Gaussian and exponential distribution, both of which are log-concave functions. Hence this function, which is their product, is also log-concave and thus has only one mode and hence it is reasonable to approximate the posterior as a Gaussian centered around this mode. This is called the Laplace approximation, depicted pictorially as:



The covariance matrix is taken as the inverse of the Hessian $\nabla^2 \log f\left(x(t_k)\mid \zeta^k\right)$, computed at the mode. We compute the first and second derivatives of the function as:

$$\nabla \log f\left(x(t_k)\mid \zeta^k\right) = -W^{-1}(t_k\mid t_{k-1})\cdot\left(x(t_k)-\hat{x}(t_{k-1}\mid t_{k-1})\right) - \sum_{c=1}^{C} I_c(t_k) W_c^{-1}\left(x(t_k)-\mu_c\right)$$
$$+ \sum_{c=1}^{C}\exp\left\{\alpha_c - \frac{1}{2}\left(x(t_k)-\mu_c\right)' W_c^{-1}\left(x(t_k)-\mu_c\right)\right\}\cdot W_c^{-1}\left(x(t_k)-\mu_c\right)$$

$$\nabla^2 \log f\left(x(t_k)\mid \zeta^k\right) = -W^{-1}(t_k\mid t_{k-1}) - \sum_{c=1}^{C}\left[I_c(t_k) - \lambda^c\left(x(t_k)\right) + \lambda^c\left(x(t_k)\right) W_c^{-1}\left(x(t_k)-\mu_c\right)\left(x(t_k)-\mu_c\right)'\right] W_c^{-1}$$

The mode $\hat{x}(t_k\mid t_k)$ is then computed by maximising the funcion f by setting:

$$\nabla \log f = 0$$

This non-linear equation is solved using the Newton-Conjugate-Gradient algorithm, wherein a function f(x) is fit to a locally quadratic form:

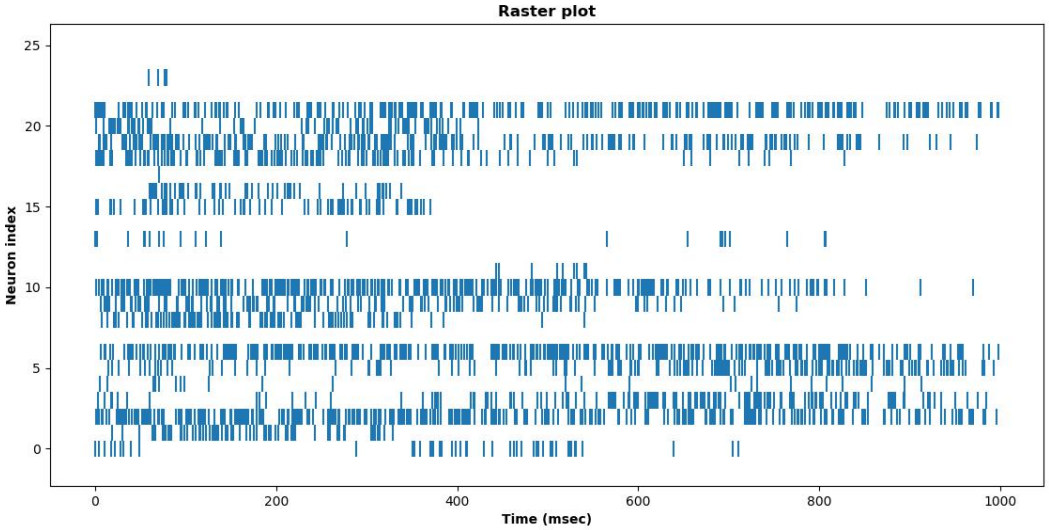
$$f(x) \approx f(x_0) + \nabla f(x_0)\cdot(x-x_0) + \frac{1}{2}(x-x_0)' H(x_0)(x-x_0)$$

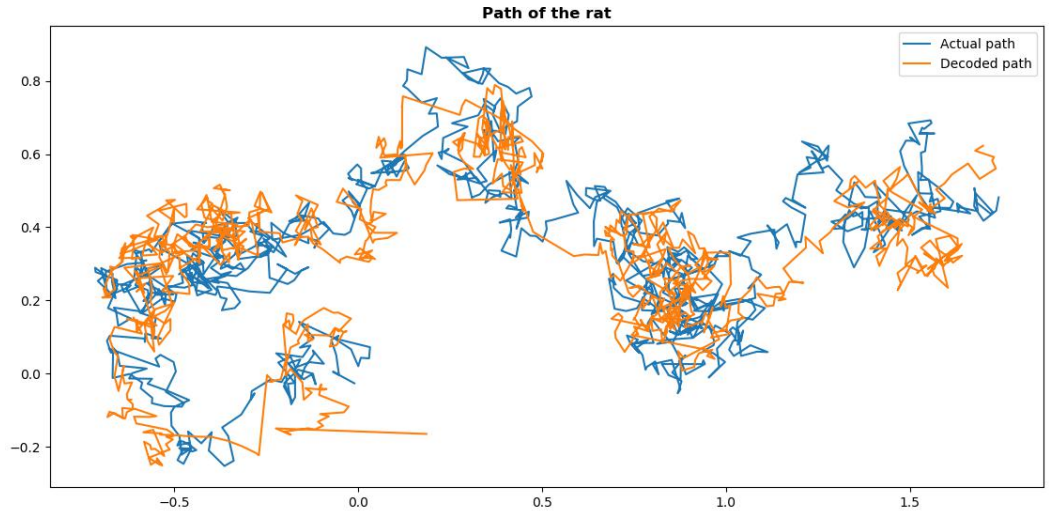
Convergence is quadratic, and typically only 2-4 iterations are needed. It can be seen from the equations above that the inverse of the scale matrices are the fixed components of the weights on the place cell means and reflect the geometry of the place fields. Place cells whose scale matrices have small scale factors—highly precise fields—will be weighted more in the new position prediction.

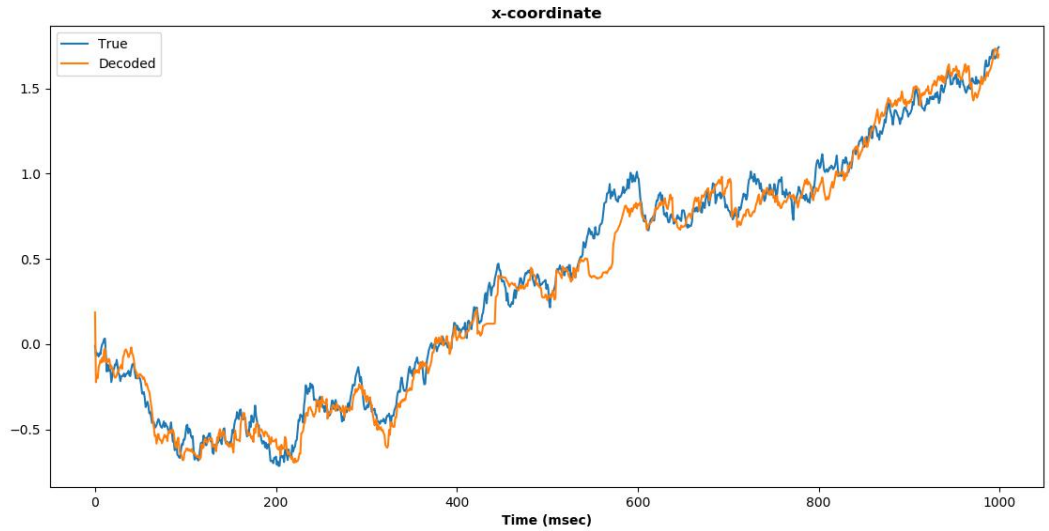
Simulation results

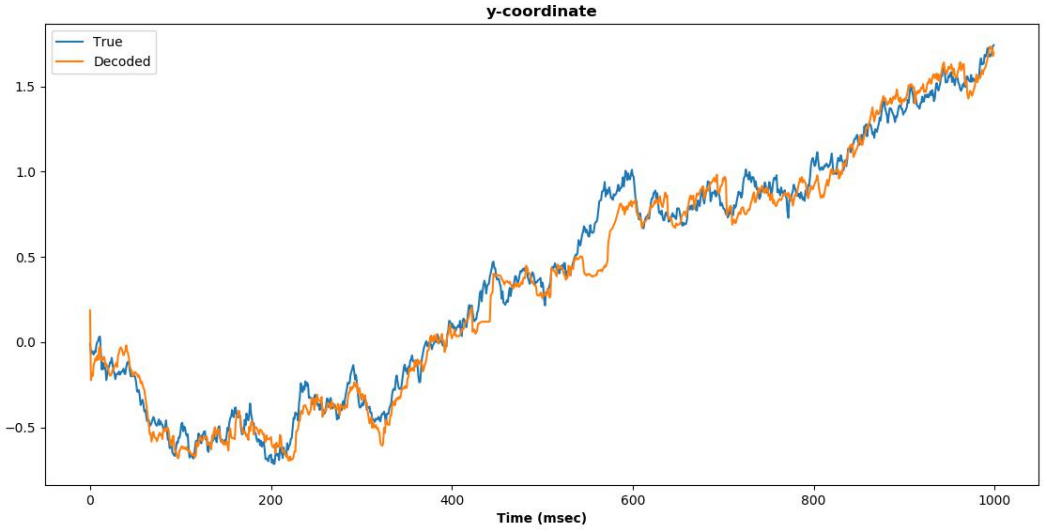
I simulated the formulation discussed above in Python 2.7, with 25 place cells and 1000 time-steps. Despite the small number of place cells, the algorithm was largely able to track the position of the rat. The results for two runs are shown below. The raster plot shows the spike-train output from the place cells, which is followed by a comparison of the actual and decoded paths of the rat.

Run 1:

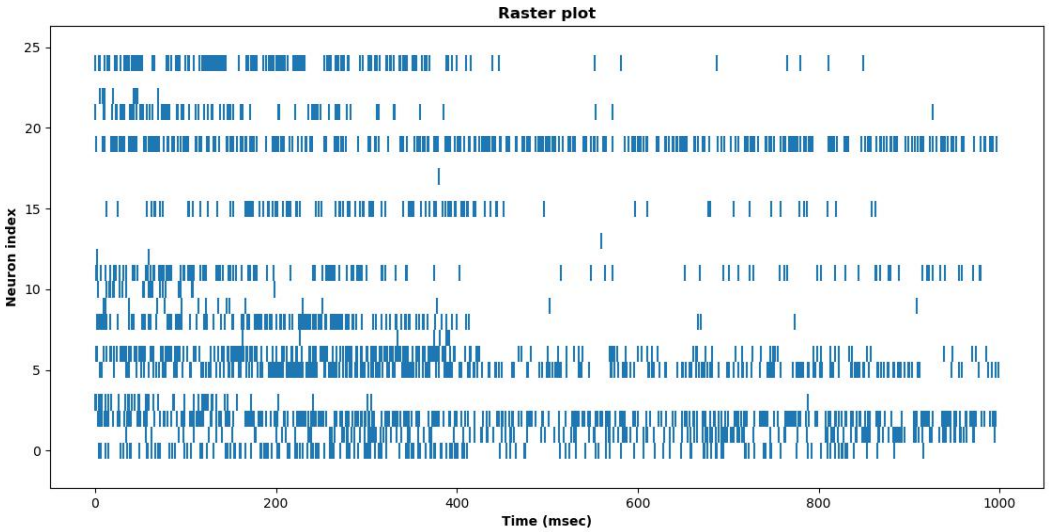

Raster plot
Neuron index
Time (msec)

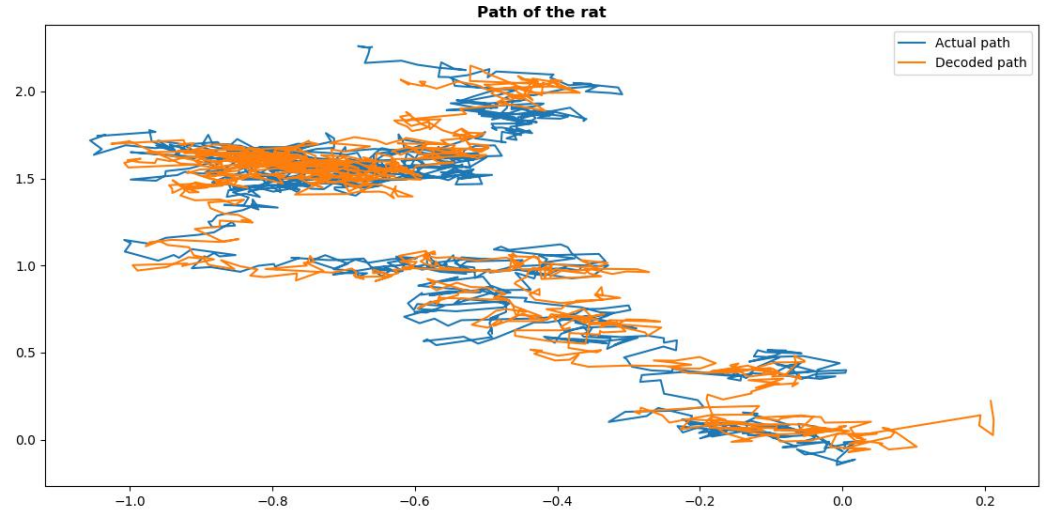

Path of the rat
Actual path
Decoded path

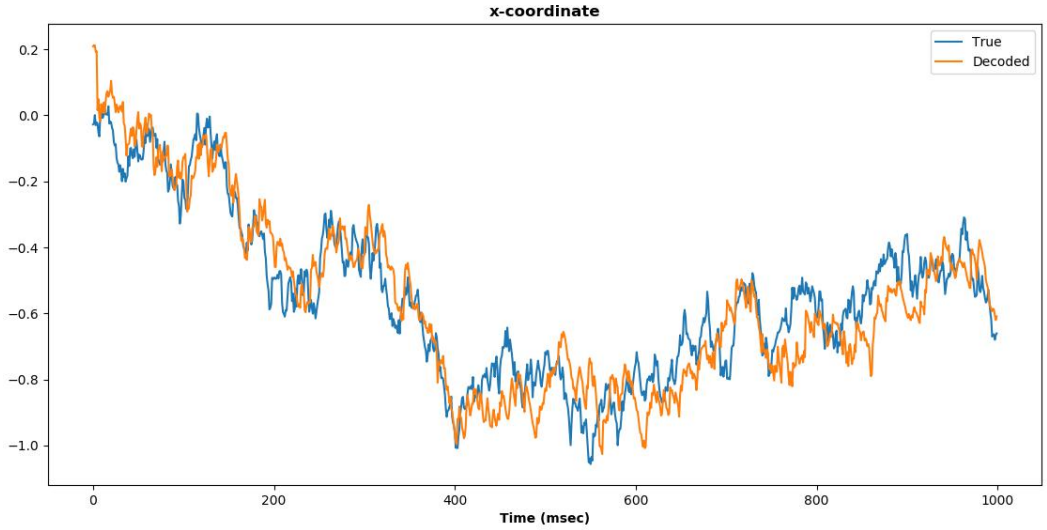

x-coordinate
True
Decoded
Time (msec)

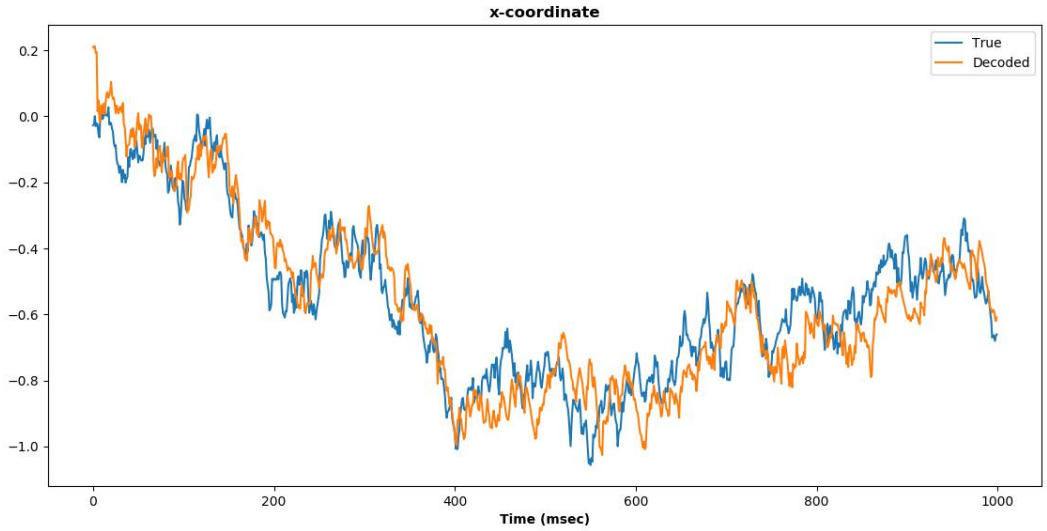

y-coordinate
True
Decoded
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
800
1000
Time (msec)

Run 2:


Raster plot
Neuron index
25
20
15
10
5
0
0
200
400
600
800
1000
Time (msec)


Path of the rat
Actual path
Decoded path
2.0
1.5
1.0
0.5
0.0
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2





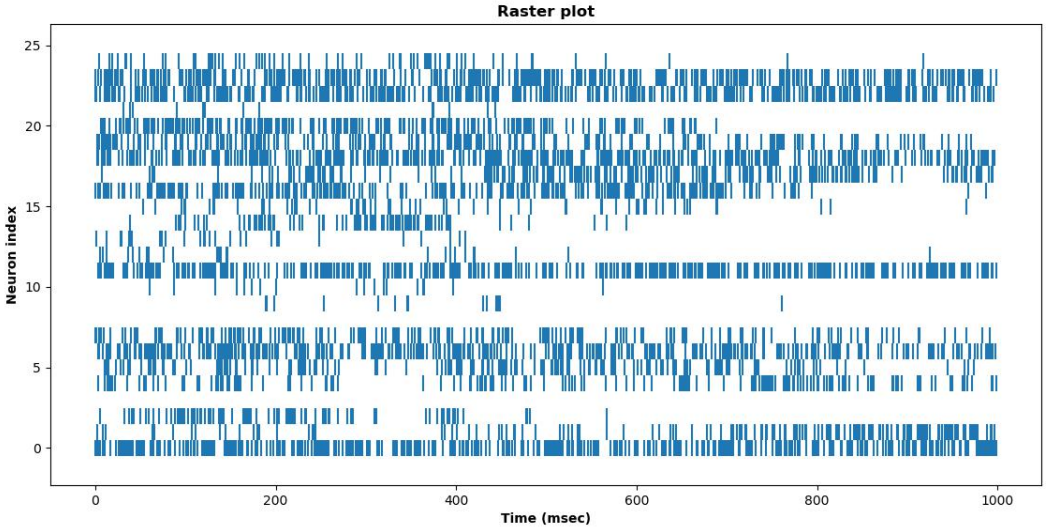
Decoding using Particle Filter

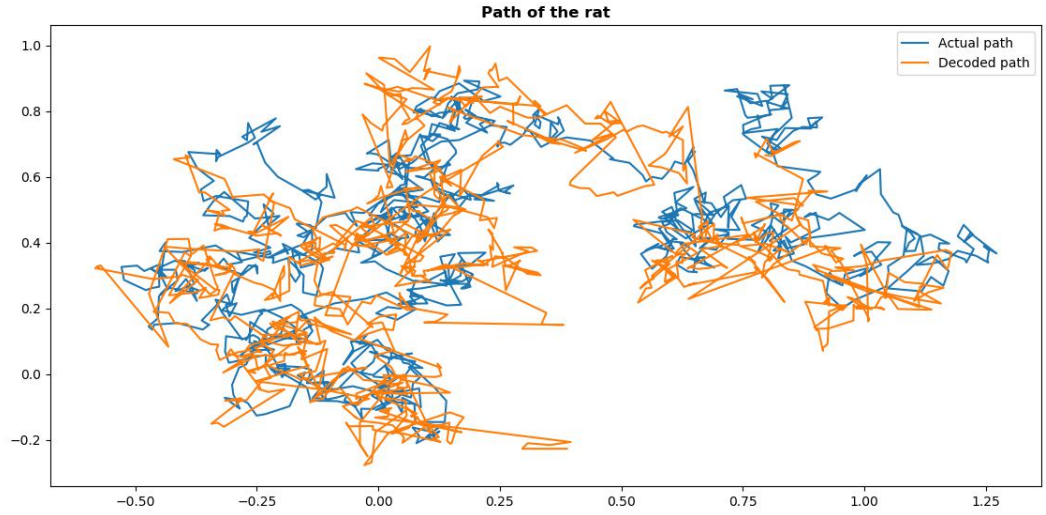
A Sequential Importance Resampling Particle filter (SIR-PF) is a specific instance of a Sequential Monte Carlo method that uses Importance sampling and particle resampling methods. As in other Monte Carlo methods, the probability distribution of the current state (at time t – 1) is represented by a set of samples. These samples are called particles. To obtain the probability distribution of the state at the following time point, it suffices to propagate the probability through the equations formulated above. The idea of importance sampling is to sample from a different distribution, and to then reweight these samples so that their weighted sum is an approximation to the desired distribution. However, as these weights are computed sequentially, the variance of the estimates increases exponentially with the number of timesteps. Resampling is a way to combat this by sampling new particles from the current estimated posterior distribution. Here I have implemented simple multinomial resampling at each step, as in [5]. For technical details of importance sampling and resampling methods please refer [6][7].

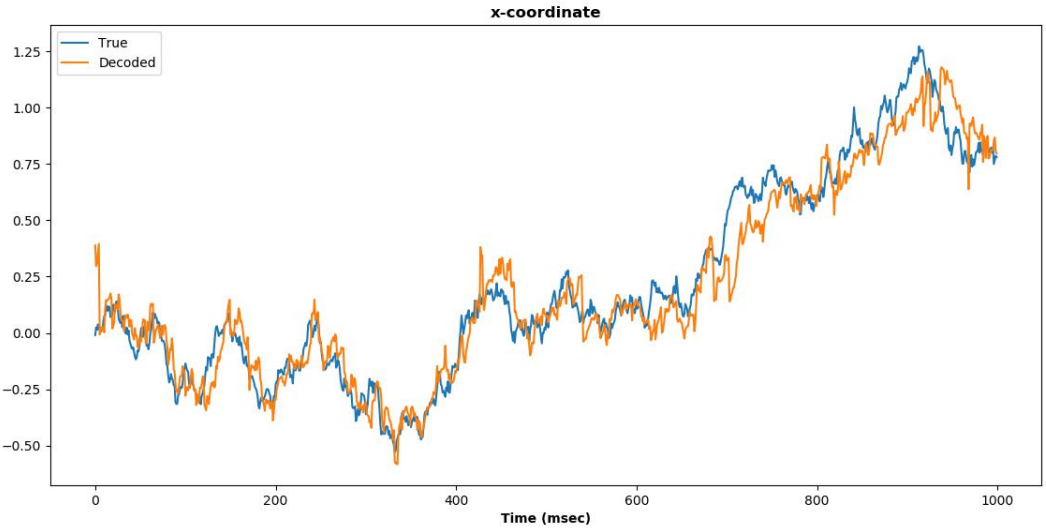
Simulation results

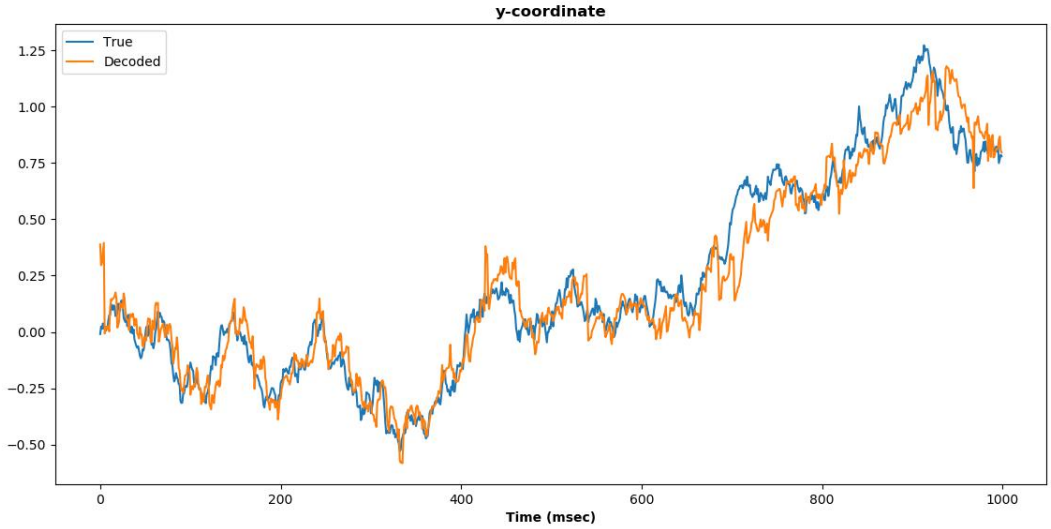
As for the IEKF, I simulated the SIR-PF in Python 2.7, with 25 place cells and 1000 time-steps. The number of particles used was 1000.

Run 1:

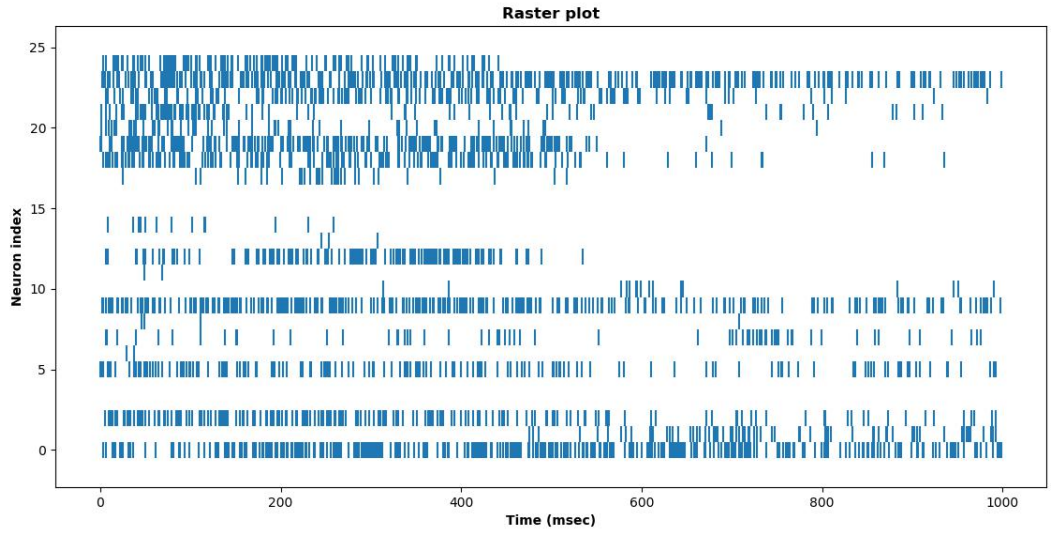


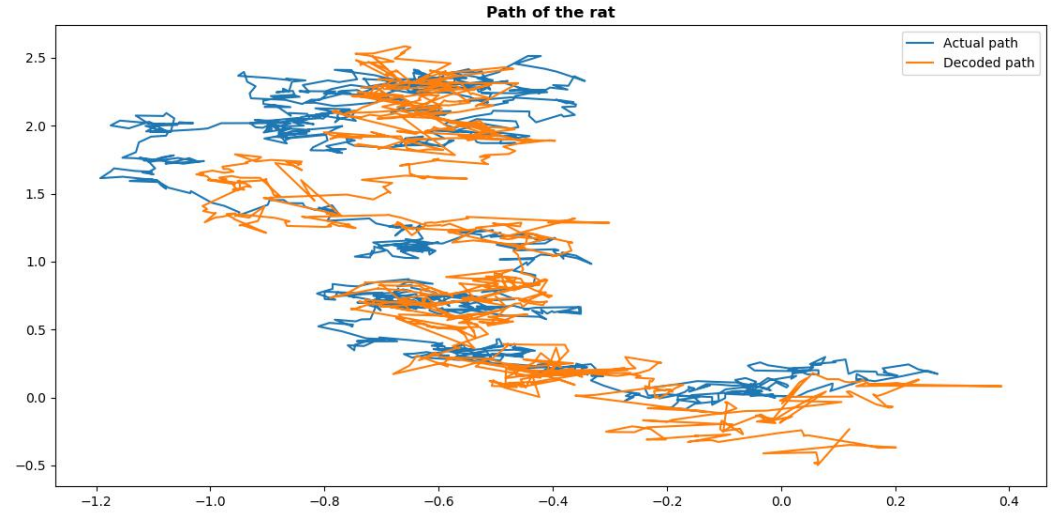


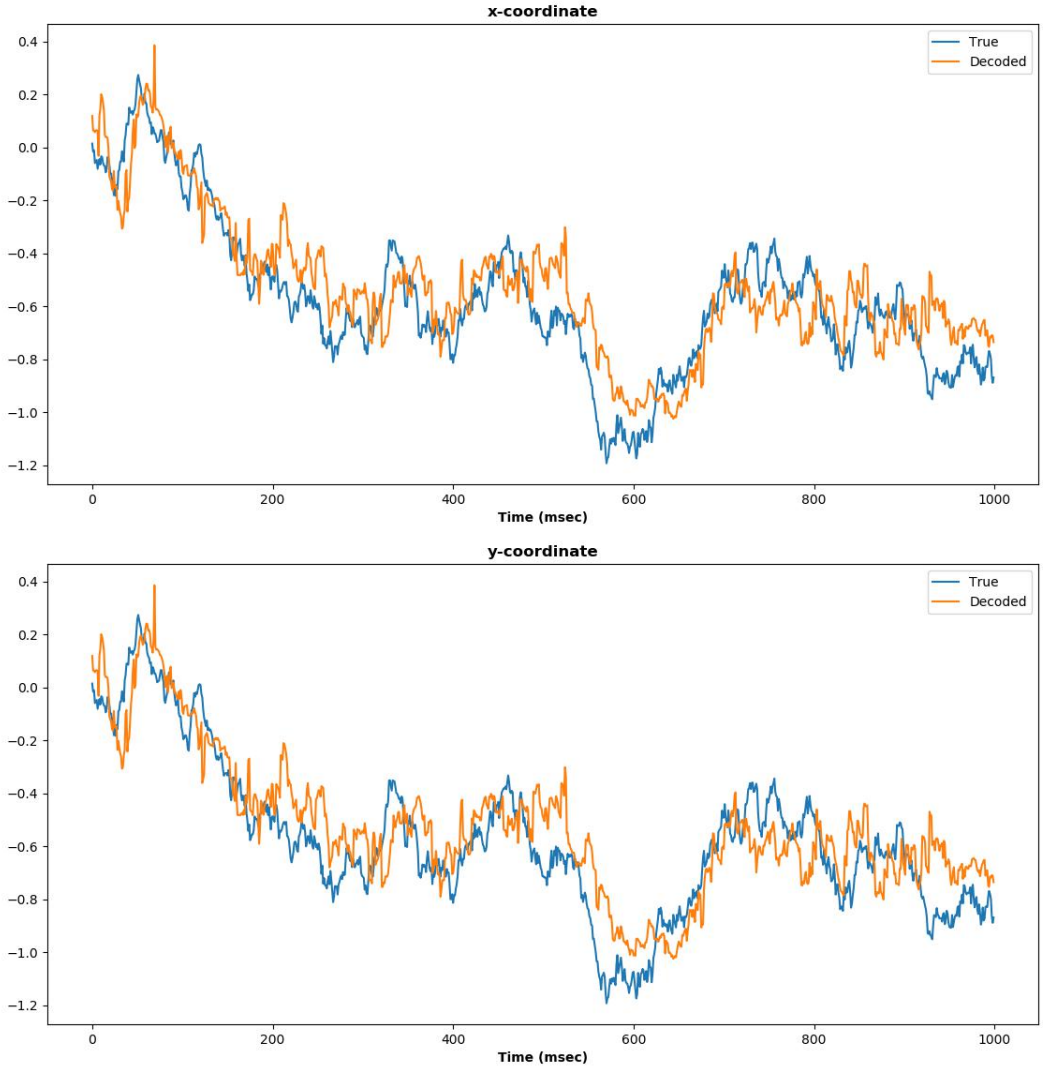



y-coordinate
True
Decoded
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
200
400
600
800
1000
Time (msec)

Run 2:

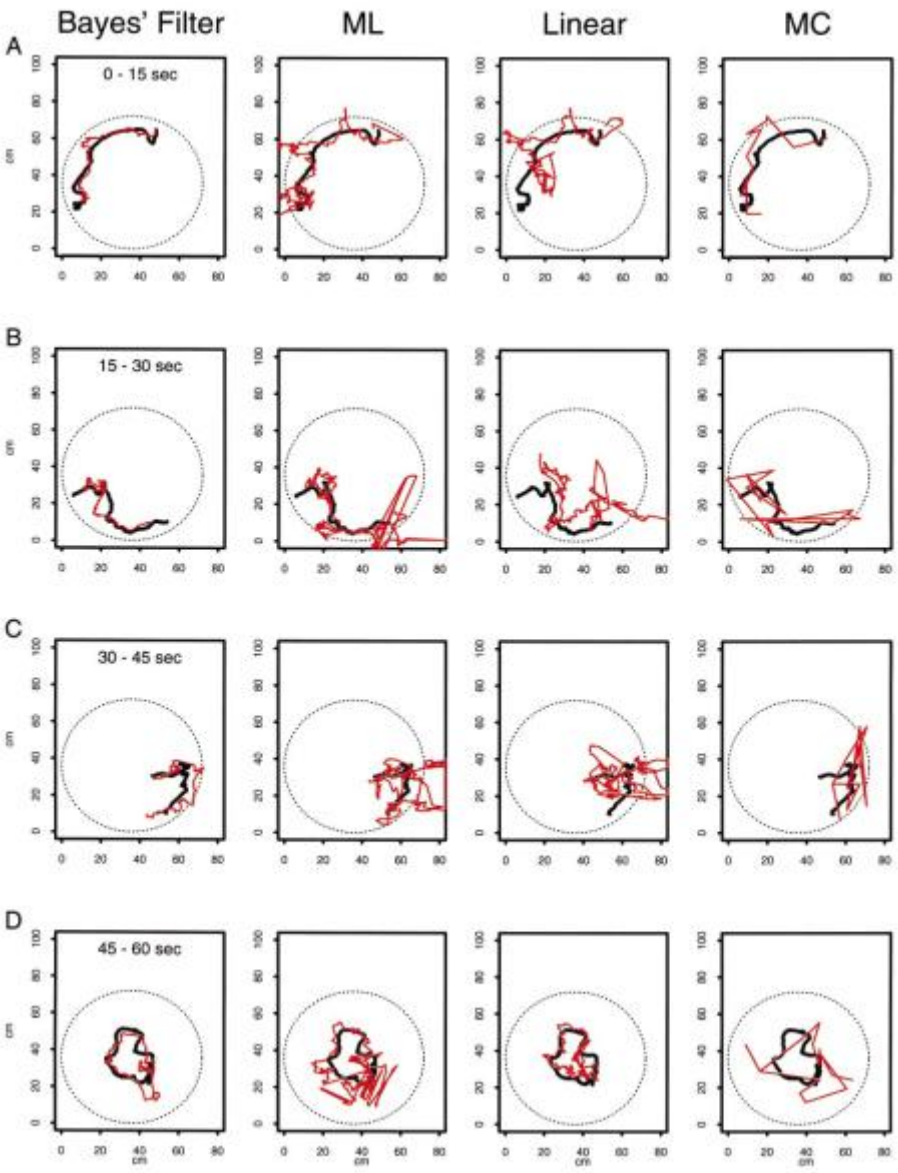

Raster plot
Neuron index
25
20
15
10
5
0
0
200
400
600
800
1000
Time (msec)


Path of the rat
Actual path
Decoded path
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.2
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4



Comparison with other methods

Brown et al. [1] compared their proposed IEKF decoder with other Non-Bayesian decoders: Maximum likelihood (ML) decoding (taking non-overlapping 1ms time-intervals), optimal linear decoding and Maximum Correlation (MC) decoding. For these filters, the position estimate is computed from the place cell firing patterns during a short time window. The Bayes' filter relies both on prior and new information, whereas the non-Bayes' algorithms use only current information. It is to also be noted that because this ML algorithm uses a 1 sec time window, it is not the ML algorithm that would be derived from the Bayes' filter by assuming an uninformative prior probability density. The local linear decoding algorithm uses no information about previous position or the probability of a place cell firing to determine the position prediction. It simply weights the place cell centers by the product of the number of spikes in the time interval and the inverse of the scale matrices. If no cell fires, there is no position prediction. Because the algorithm uses no information about the place cell firing propensities, it is expected to perform less well than either the Bayes or the ML algorithms. The following figure from [1] depicts the decoded paths (in red) using the various methods considered:



The position predictions of the ML and linear decoding algorithms agreed qualitatively with the true path; however, both had many large erratic jumps in the predicted path. The large errors occurred in a short time span because these algorithms have no continuity constraint and because the number of spikes per cell in a 1 sec time window varied greatly because of the low intrinsic firing rate of the place cells. On the other hand, because these algorithms lack a continuity constraint, they could adapt quickly to changes in the firing pattern, which suggested abrupt changes in the animal's velocity. The MC algorithm path predictions were also erratic and more segmental in structure because this algorithm estimated the animal's position in nonoverlapping 1 sec intervals. Note that the random walk is a weak continuity assumption made to implement the Bayes' filter as a sequential algorithm. This constraint is one reason the path predictions of the IEKF algorithm closely resembled the true path.

While the IEKF is better than the EKF due, it often has the problem of getting stuck in local minima, and Brown et al. also report that the IEKF fails to sometimes track the rat's position after rapid movements. A proposed solution to overcome these

limitations is to use a particle filter [5], which is a sequential Monte Carlo method. In [5] they report that the EKF seems to get stuck for long periods of time after rapid movements, which might be due to the posterior of the state having multiple local minima. The EKF, which performs a local search, gets stuck in a local minimum for some number of time steps, until by chance a path is created towards the global minimum, but it could also be due to a failure of the model to capture the super-Gaussian changes in position from frame to frame. The particle filter that I have implemented here replicates these findings.

Conclusions and Future work

The Iterated Extended Kalman filter was formulated and successfully used to decode the rat's position from a small number of place cell recordings. The validity of these approaches was verified via simulations. These methods were compared to other decoding algorithms. In future work, the assumption of an inhomogenous Poisson process can further be relaxed by including a refractory period. For the particle filter, more sophisticated resampling methods can be used. It can also be noted that the decoding stage proposed here is the E-step in the EM algorithm, if we wished to also estimate the responses of the place cells, and hence this framework can be extended to jointly decode and estimate responses of neurons.

References

1. Brown, Emery N. and Frank, Loren M. and Tang, Dengda and Quirk, Michael C. and Wilson, Matthew A., A Statistical Paradigm for Neural Spike Train Decoding Applied to Position Prediction from Ensemble Firing Patterns of Rat Hippocampal Place Cells, Journal of Neuroscience 15 November 1998, 18 (22) np; DOI: https://doi.org/10.1523/JNEUROSCI.18-22-j0001.1998
2. Shimazaki, H., Amari, S. ichi, Brown, E. N., & Grün, S. (2012). State-space analysis of time-varying higher-order spike correlation for multiple neural spike train data. PLoS Computational Biology, 8(3). https://doi.org/10.1371/journal.pcbi.1002385
3. Shimazaki, H. (2015). Approximate Inference for Time-Varying Interactions and Macroscopic Dynamics of Neural Populations. Nature Neuroscience, 18(5), 736–743. https://doi.org/10.1038/nn.3979
4. https://neuwritesd.org/2013/07/02/the-hippocampus-going-the-distance-beyond-space-and-time/
5. Brockwell, A. E. (2004). Recursive Bayesian Decoding of Motor Cortical Signals by Particle Filtering. Journal of Neurophysiology, 91(4), 1899–1907. https://doi.org/10.1152/jn.00438.2003
6. http://www.almoststochastic.com/2013/08/sequential-importance-sampling.html
7. https://www.seas.harvard.edu/courses/cs281/papers/doucet-johansen.pdf