Analysing place cells

Balazs B Ujfalussy

20/11/2017

This is a demo for illustrating place cells. We will analyse a 30 min recording session from a rat running back and forth in a 1.6 m ling linear track, from the laboratory of Gyuri Buzsaki. There are about 120 pyramidal cells recorded simulataneously, half of them are place cells. There are also 20 interneurons.

There are two exercices related to the dataset:

- analysing the coding properties of place cells - how reliable they are? How much information they contain?
- decoding the activity of place cells - can you use the techniques learned in the course to predict the position of the rat from the spike counts?

The source of the data: Grosmark, A.D., Long J. and Buzsáki, G (2016). Recordings from hippocampal area CA1, PRE, during and POST novel spatial learning. CRCNS.org http://dx.doi.org/10.6080/K0862DC5

Paper related to the dataset: Grosmark, A.D., and Buzsáki, G. (2016). Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences. Science 351, 1440–1443.

Load the dataset and observe place cells

We will read a preprocessed data file 'Achilles.RData' - Achilles is the name of the rat. It is a list storing different variables describing the experiment.

```
load('./Achilles.RData')
```

The `summary()` function tells you what variables are encoded int the list `rat`:

```
summary(rat)
```

```
##            Length  Class  Mode
## pos         161524 -none- numeric
## spt        1731934 -none- numeric
## iruns.up        84 -none- numeric
## iruns.down      84 -none- numeric
## PyrIDs         120 -none- numeric
## IntIDs          17 -none- numeric
## MazeRange        2 -none- numeric
```

- The main variables encoded are the position (`pos`), spike times (`spt`). The position is a matrix of two columns: time in seconds and smoothed 1D position of the animal along the linear track. The spt is also a two columnn matrix, time in seconds and the ID of the cell that emitted the spike. (Cell id's refer to the elctrodes the cell was recorded from, so they do not start from 1...)
- Before and after each run the animal stays at the end for a while to consume reward. Up and down runs (left and right, sorry :-)) are associated with different neuronal activity, so they are treated differently. The variable `iruns.down` and `iruns.up` stores the start and the end of the individual runs indexing the rows of the matrix `pos`.
- The variable `PyrIDs` and `IntIDs` stores the name of the (putative) pyramidal cells and interneurons.
- `MazeRange` is the x coordinates associated with the start and the end of each run.

The individual variables in the list can be referred by the $ sign. For example the dimensionality of the position variable can be prompted as `dim(rat$pos)` which returns 80762, 2 meaning that this matrix has 80762 rows and 2 columns.

Now we will load a function that will analyse this data to return the spike counts for each cell on each runs in the function of the (discretised) position.

```
source("PlaceCellFunctions.R")
require(viridis)
```

```
## Loading required package: viridis
```

```
## Loading required package: viridisLite
```

```
pos <- rat$pos
spt <- rat$spt
```

We define the spatial discretization in 5 cm.

```
dx <- 0.05 # cm, resolution
x.breaks <- seq(rat$MazeRange[1], rat$MazeRange[2], by=dx)
x.mids <- round(x.breaks[-1] - dx/2, 3)
```

`act.runs` is an array with the spikes of each cell on each trial with spatial resolution dx. Its dimensions are {number of cells} x {distance} x {trials}. The last neuron is not a true neuron, but stores the time (in seconds) the rat spent at each location at each trial. We only analyse the up runs here.

```
act.runs <- cell.maps.runs(spt, pos, i.runs=rat$iruns.up, dx=0.05, MazeRange=rat$MazeRange, cell.IDs=ra
```

We divide spike count with the occupancy time to get firing rates, and plot the firing rate of all cells in the function of distance.

```
ratemaps.t <- apply(act.runs[1:120,,], c(1,2), sum)
Tmap <- apply(act.runs[121,,], 1, sum)

ratemaps.all <- t(ratemaps.t) / Tmap
matplot(x.mids, ratemaps.all, t='l', lty=1, col=rainbow(120), xlab='x position (m)', ylab='firing rate
```

Next, plot firing rates of two example neurons estimated on individual runs - we still need to divide with occupancy time!

```
par(mfcol=c(1,2)); par(mar=c(4,4,4,4))
image(x.mids, 1:42, act.runs[70,,] / act.runs[121,,], col=viridis(24), xlab='x position (m)', ylab='tri
lines(x.mids, ratemaps.all[,70], col=viridis(3)[3], lwd=2)
axis(4, c(0, 5, 10), c(0, 5, 10))
mtext('firing rate (Hz)', 4, 2, adj=0)
image(x.mids, 1:42, act.runs[97,,] / act.runs[121,,], col=viridis(24), xlab='x position (m)', ylab='tri
lines(x.mids, ratemaps.all[,97], col=viridis(3)[3], lwd=2)
axis(4, c(0, 5, 10), c(0, 5, 10))
mtext('firing rate (Hz)', 4, 2, adj=0)
```

Finally plot the ratemap of all active cells - sorted according to the position of the peak.

```
i.cells.active <- which(apply(ratemaps.all, 2, max) > 5)
N.cells.active <- length(i.cells.active)
ratemaps <- ratemaps.all[,i.cells.active]
ii.maxs <- apply(ratemaps, 2, which.max)
sort.peaks <- sort(ii.maxs, ind=T)$ix
```

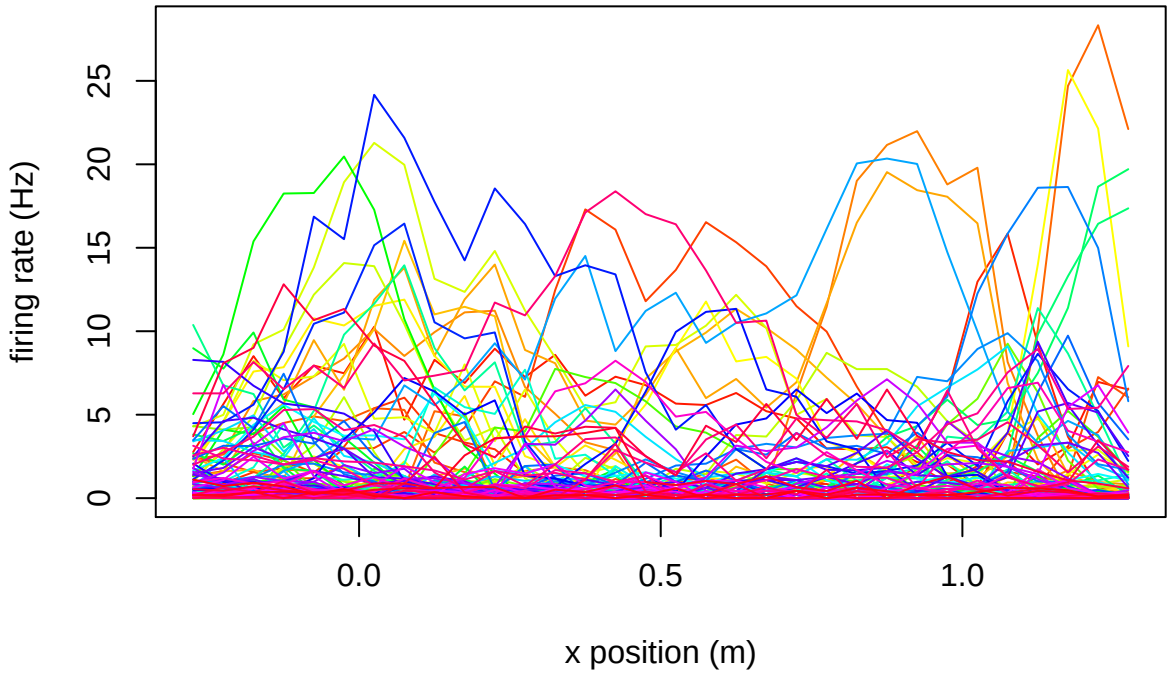

Figure 1: Ratemap of all pyramida cells.

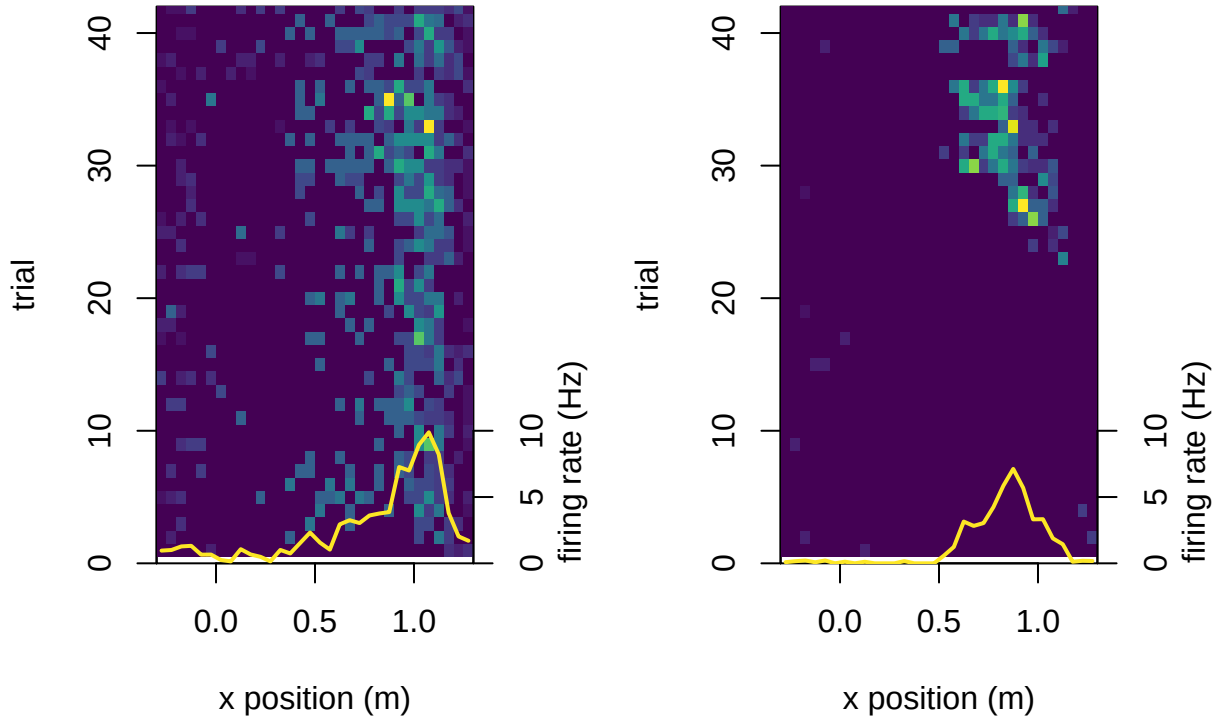

Figure 2: Firing rates on individual trials for two neurons. The left cell looks like a classical place cell, the right starts to fire only after the 30th run.

```
par(mfcol=c(1,2))
matplot(x.mids, ratemaps[,sort.peaks], t='l', lty=1, col=rainbow(60), xlab='x position (m)', ylab='firi
image(x.mids, 1:N.cells.active, ratemaps[,sort.peaks], col=viridis(24), xlab='x position (m)', ylab='ce
```

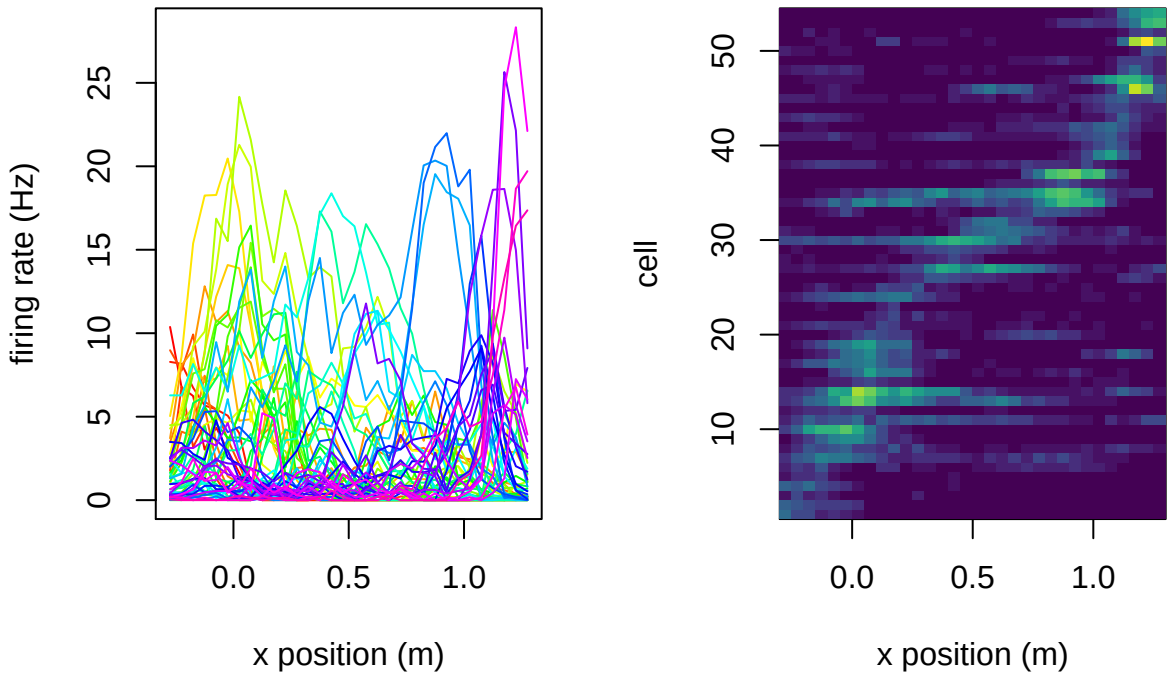

Figure 3: Place cells sorted according to their peak firing rate

Homework 1 - reliability

Calculate the information rate (bits/spike) for all place cells, which is defined in Skaggs et al.: An information theoretic approach to deciphering the hippocampal code. 1993 as:

$I = \sum_x r(x)\, P(x) \log \frac{r(x)}{\hat{r}}$

where $r(x)$ is the firing rate of the cell as the function of position, $\hat{r} = \sum_x P(x)\, r(x)$ is the mean firing rate and $P(x)$ is the estimated probability of being at position x. I is measured in bits/second (if a base 2 logarithm is taken).

Calculate the above quantity for the place cells. Compare it with their activity map. Do you think that cells with higher information rate are more informative about the location of the animal?

What is the information rate of the two cells shown in Fig. 2? Does the difference in their information rate reflets the striking difference in their behaviour? What other measure would you use to discriminate the left and the right cell in Fig. 2? Implement it and show how it works (or describe why it fails)!

Homework 2 - decoding

Now we will do static decoding of place cell activity. First, we need the spike patterns and the position in temporal bins of 0.1 second width. Note, that now we analyse the same dataset in temporal windows. Before we analysed it in spatial windows. The function `pop.act.t` will return the population activity during the run sessions in approximately 100 ms time windows. This will take some time (~1 min on my computer), but you will be able to follow the progress of the algorithm.

```
all.data <- pop.act.t(spt, pos, i.runs=rat$iruns.up, dt=0.1, cell.IDs=rat$PyrIDs[i.cells.active])
```

```
summary(all.data)
```

```
##        Length Class  Mode
## pos      2776 -none- numeric
## popact 149904 -none- numeric
```

The list `all.data` contains two variables: `pos` is a vector, with the x position and `popact` is a matrix with the population activity. Each row in `all.data$popact` is a cell and each column shows the spike counts $\mathbf{s}$ in the given time bin. We have 2776 time points.

We prepare two datasets, a training and a test dataset, each containing the spike counts of the 54 active cells and the location of the animal x. We will randomly select 2000 time points for training and the rest for testing.

```
L.all <- length(all.data$pos)
L.train <- 2000
L.test <- L.all - L.train

i.train <- sample (1:L.all, L.train)
i.test <- which(!(1:L.all %in% i.train))

data.train <- list(pos=all.data$pos[i.train], popact=all.data$popact[,i.train])
data.test <- list(pos=all.data$pos[i.test], popact=all.data$popact[,i.test])
```

We can learn the relationship between $\mathbf{s}$ and x using the training data. Then we can predict the position x from the spikes $\mathbf{s}$. Use one of the learned techniques to predict the position x from the spikes $\mathbf{s}$ in the test data - `data.test`.