Bayesian Decoding at CoSMo

Marius 't Hart

Data: Lee Miller & Jim Rebesco Exercise: Gunnar Blohm Downloaded from: CoSMo Wiki (on March 8th, 2019: Afternoon Tutorial 2 from the two introductory days)

The goal of this assignment is to gain experience using Bayesian statistical methods, such as used for multi-sensory integration, optimal feedback control, Kalman filtering, etc. As an example, we will use neural spike rates from different neurons and apply a Bayesian decoder to infer movement direction.

Basic statistics

In the first part of the tutorial you will see how well individual neurons can decode a movement from those neurons' firing rates.

Please download the data set and familiarize yourself with it. It contains firing rate data of 35 neurons over 205 trials (curtesy of Lee Miller and Jim Rebesco).

For this tutorial, the data is stored as an R data frame:

```
load('data/neuronrates.rda')
```

The first column indicates a movement direction (it is unsure to me if these are eye- or hand movements, or something else), which is one of two directions. And while I don't actually know the movement directions, I labeled them left and right, all the same, to make things a bit more tangible. What is also not clear - but we may guess this - is if these are single-cell recordings, if they are from macaques, and if the spikes rates were recorded simultaneously from all 35 neurons during the same movements. We also don't know what area(s) the neurons were in, or if the recordings were done during preparation of the movement, during the movement, both, or even at some other time.

For the exercise it doesn't matter much, so we can simply imagine what all this data means.

We might be interested in basic distribution of the two kinds of variables: direction and firing rates. Let's have a look at the directions first.

```
summary(neuronrates$direction)
```

```
##  left right
##   101   104
```

The movement directions are almost equally distributed. If we'd guess 'right' all the time, we'd be 50.73% correct, less than a percent above chance. While we might make it equal by removing a few of the right ward trials, it won't matter a lot.

Now, let's get a summary of the other columns:

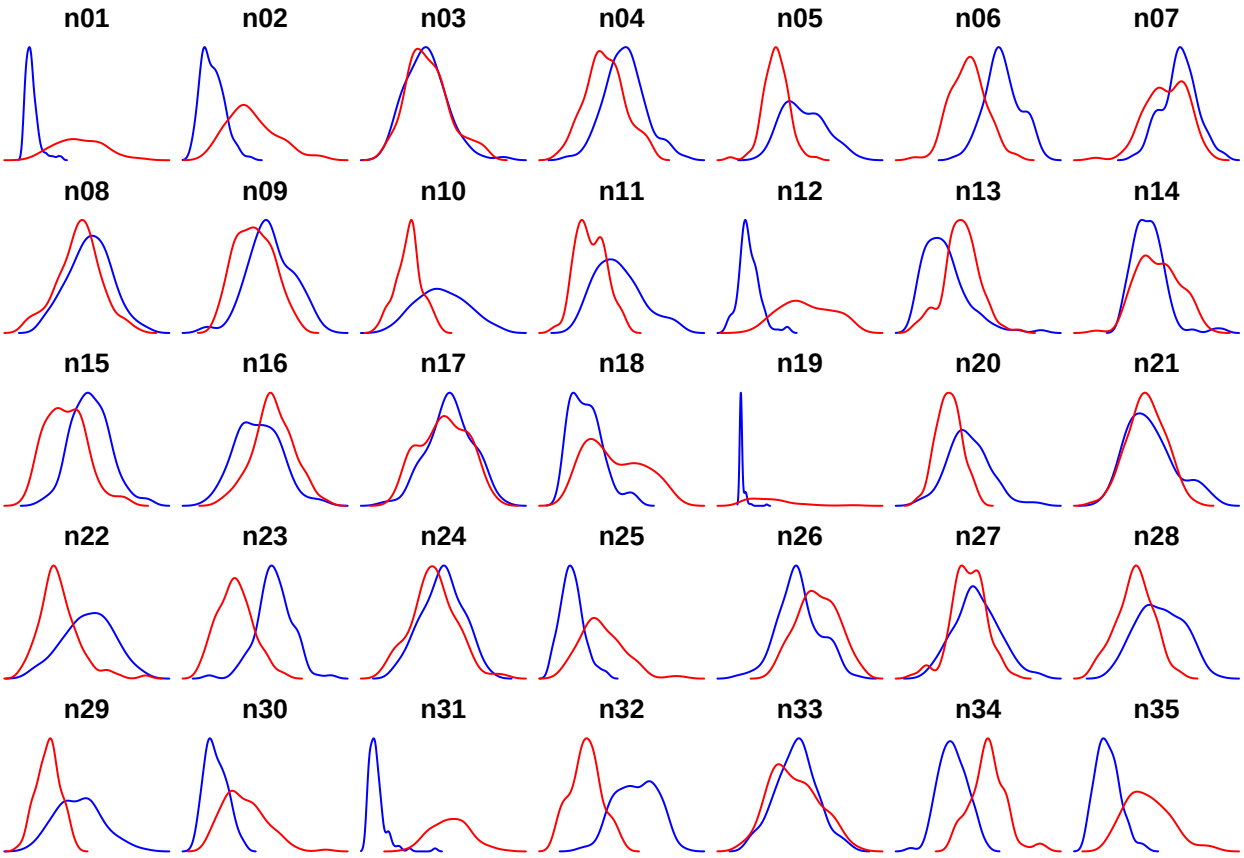
```
summary(as.vector(unlist(neuronrates[2:36])))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   5.584  10.853  15.917  18.663 105.350
```

The minimum spike rate in the data appears to be 0. We can assume this means that some neurons did not fire at all in some intervals. The maximum firing rate is over a 100 (spikes/second?) and it would seem that the data in it's entirety is not really normally distributed.

We plot the data to get a little more familiar with it. We'll do this as two density plots in each subplot, one for left movements (blue), and one for right movements (red). This way we can already get an idea of how well a given neuron can distinguish between left or right movements.

```
par(mfrow=c(5,7),mar=c(0,0,2,0))

for (neuron in sprintf('n%02d',c(1:35))) {
  ld <- density(neuronrates[which(neuronrates$direction == 'left'),neuron])
  rd <- density(neuronrates[which(neuronrates$direction == 'right'),neuron])
  xlim <- c(min(c(range(ld$x),range(rd$x))),max(c(range(ld$x),range(rd$x))))
  ylim <- c(0,max(c(range(ld$y),range(rd$y))))
  plot(-1000,-1000,main=neuron,xlim=xlim,ylim=ylim,xlab='',ylab='',bty='n',ax=F)
  lines(ld$x,ld$y,col='blue')
  lines(rd$x,rd$y,col='red')
}
```



Some neurons fire more when the movement is to the right (2 and 13), and some fire more when the movement is to the left (15 and 23). Most of the distributions look fairly normal. There are also some neurons that seem to provide almost no information (3) or that have very different density curves for left and right movements (19). It seems that even with just the last 7 neurons in this plot (n29 - n35) we should be able to make highly accurate decisions on which directions a bunch of spike rates was taken from.

Let's look at neuron 19 a bit more, to see why it has these density curves.

```
l19 <- neuronrates[which(neuronrates$direction == 'left'),'n19']
r19 <- neuronrates[which(neuronrates$direction == 'right'),'n19']
print(l19)
```

```
##   [1] 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.51282
```

```
##   [9] 0.00000 0.00000 0.00000 0.86957 0.00000 0.00000 0.00000 0.00000
##  [17] 0.00000 0.00000 0.15798 0.20833 0.00000 0.00000 0.42017 0.00000
##  [25] 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
##  [33] 0.00000 0.00000 0.30395 0.00000 0.00000 0.00000 0.00000 0.12453
##  [41] 0.00000 0.00000 0.00000 0.00000 0.30488 0.45045 0.42017 0.00000
##  [49] 0.00000 0.14903 0.18727 0.00000 0.39920 0.00000 0.00000 0.00000
##  [57] 0.00000 0.00000 0.42194 0.35088 0.27397 0.00000 0.00000 0.00000
##  [65] 0.00000 0.39370 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
##  [73] 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
##  [81] 0.00000 0.00000 0.00000 0.00000 0.00000 0.69324 0.43764 0.00000
##  [89] 0.00000 0.00000 0.00000 0.00000 0.00000 2.11270 0.00000 0.00000
##  [97] 0.00000 0.00000 0.00000 0.00000 0.00000
```

```
print(r19)
```

```
##   [1] 3.74330 1.58730 1.90110 3.02010 6.06060 2.21130 3.00750 1.40350
##   [9] 2.01610 1.93050 0.89552 2.17980 1.83490 2.13680 1.54800 1.67790
##  [17] 5.41960 9.06040 3.43640 1.38890 1.29200 1.25390 1.73910 5.38730
##  [25] 3.44830 4.95050 2.88460 3.18840 0.00000 2.65490 3.00430 4.62960
##  [33] 1.91390 6.96520 2.69230 2.45900 3.50880 3.13480 0.80808 1.61290
##  [41] 9.25930 2.30770 3.26090 2.60220 3.39510 3.68420 5.20450 3.01890
##  [49] 4.74780 0.42918 1.98860 2.92890 9.64290 3.84620 2.63740 3.52940
##  [57] 4.06700 5.19480 2.54450 3.14690 6.27060 8.26090 2.62010 2.86890
##  [65] 2.20990 3.11280 4.70590 1.62600 0.00000 0.97087 0.85106 0.29240
##  [73] 0.40323 0.61728 0.89686 1.98020 0.87719 1.20480 0.47847 0.14881
##  [81] 0.19608 0.33333 0.58480 1.10700 1.29030 0.37879 0.89686 0.00000
##  [89] 0.00000 0.75188 0.90090 0.59701 0.40323 2.39520 0.26042 0.37594
##  [97] 3.75000 4.12840 0.42918 1.71430 2.27270 1.03090 0.58140 0.80645
```

```
summary(l19)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.09101 0.00000 2.11270
```

```
summary(r19)
```

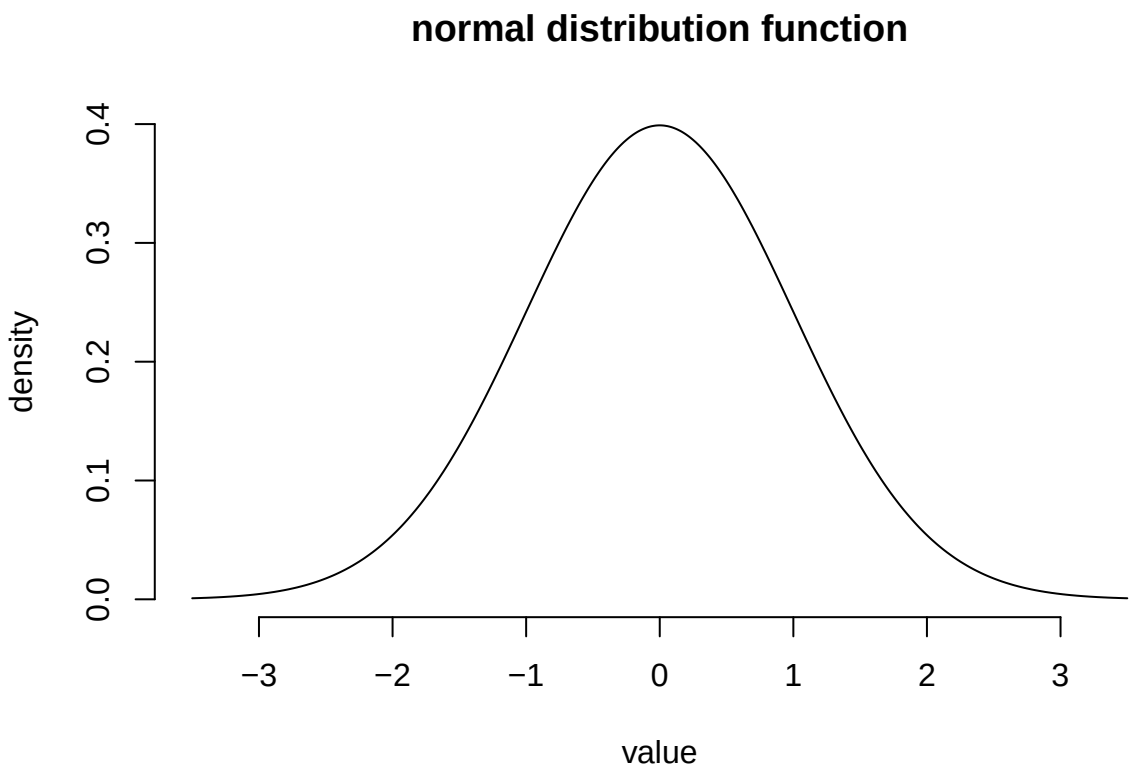
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.8965  2.0764  2.4712  3.2944  9.6429
```

So this neuron is highly selective, with almost no firing for "left" movements, and more distributed firing for "right" movements. Because all the 0 firing rates are on top of each other for left movements, this creates a big peak around 0 for that density plot, while the density plot for right movement spike rates is probably more similar to that of other neurons - but we can't see the y-axis scale as they were left out of the figure to save space.

Identify the firing rate distributions for each neuron (suppose they're Gaussian).

Each neuron will have *two* firing rate distributions: one for left movements, and one for right movements. If we look at the help page for `dnorm()` we see that a normal (or Gaussian) distribution can be described by it's mean and standard deviation. For example, the default mean is 0 and the default standard deviation is 1. This gives a density function like this:

```
plot(seq(-3.5,3.5,.01),dnorm(seq(-3.5,3.5,.01)),type='l',bty='n',
     xlab='value',ylab='density',main='normal distribution function')
```



By getting the mean and standard deviation for every neuron's left and right movement firing rates, we can describe all the data as 35 neurons x 2 movement directions = 70 normal distributions. And each of those normal distributions only need a mean and a standard deviation. Let's put this in a data frame (long-format, R-friendly):

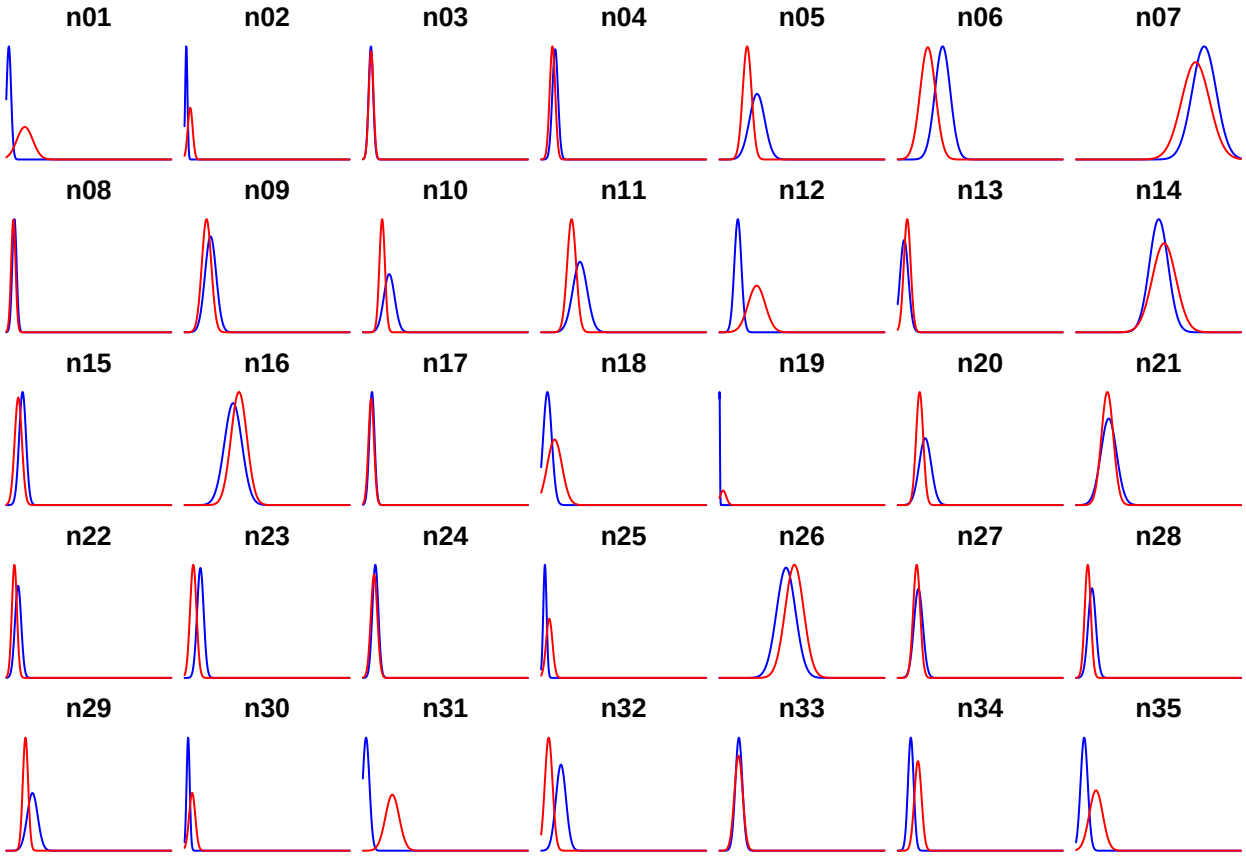
```
# we create some vectors to store everything while going through the data:
neuron <- c() # the neuron's ID
movement <- c() # left or right
mu <- c() # the mean firing rate
sigma <- c() # the standard deviation of the firing rate

for (neuron.ID in sprintf('n%02d',c(1:35))) {
  for (movdir in unique(neuronrates$direction)) {
    spikerates <- neuronrates[which(neuronrates$direction == movdir),neuron.ID]
    neuron   <- c(neuron,   neuron.ID)
    movement <- c(movement, movdir)
    mu       <- c(mu,       mean(spikerates))
    sigma    <- c(sigma,    sd(spikerates))
  }
  # spikerates <- neuronrates[,neuron.ID]
  # neuron   <- c(neuron,   neuron.ID)
  # movement <- c(movement, 'both')
  # mu       <- c(mu,       mean(spikerates))
  # sigma    <- c(sigma,    sd(spikerates))
}

spikedist <- data.frame(neuron, movement, mu, sigma)
```

Compute and plot the likelihoods for each neuron N: p(N|Left) and p(N|Right)

We're supposed to plot likelihoods, and what is meant is probably likelihood *distributions* of *firing rates*.

Since these are probabilities of the neuron having a specific firing rate *given* that the movement was right or left, we don't have to consider much yet. So basically, we're going to create the same plot as the density plot above, but with the normal distributions that we just got.

```
par(mfrow=c(5,7),mar=c(0,0,2,0))

x <- seq(0,110,.1)

for (neuron in unique(spikedist$neuron)) {
  l.mu <- spikedist$mu[which(spikedist$neuron == neuron & spikedist$movement == 'left')]
  l.sigma <- spikedist$sigma[which(spikedist$neuron == neuron & spikedist$movement == 'left')]
  l.dist <- dnorm(x,mean=l.mu,sd=l.sigma)
  r.mu <- spikedist$mu[which(spikedist$neuron == neuron & spikedist$movement == 'right')]
  r.sigma <- spikedist$sigma[which(spikedist$neuron == neuron & spikedist$movement == 'right')]
  r.dist <- dnorm(x,mean=r.mu,sd=r.sigma)

  plot(-1000,-1000,main=neuron,xlim=range(x),ylim=c(0,max(max(l.dist),max(r.dist))),bty='n',ax=F)

  lines(x,l.dist,col='blue')
  lines(x,r.dist,col='red')
}
```



Except maybe for neurons 3, 8 and 33 the distributions don't overlap totally, so we have at least 32 neurons that can tell us something about the direction of movement given the firing rates. And even those 3 will give a little bit of information, that is: they won't make our predictions worse, but they can make them a little bit better.

Do the same for the posterior probabilities. How well do likelihood and posteriors code for movement direction?

Hint: for the marginal, remember you have to sum over all options.

As a reminder, we can look up some information on these types of probabilities on the internet. For example:

P(A|B) = P(B|A) * P(A) / P(B)

For those who like actual equations better:

$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$

Which we can translate to our own situation:

p(right|rate) = p(rate|right) * p(right) / p(rate)

> *"The probability of the movement going to the right, given the observed firing rate is equal to the probability of the probability of the firing rate given that the movement is going to the right times the probability that the movement is going to the right, divided by the probability of observing this firing rate."*

It seems pretty clear how to do this for a single neuron. Of course, it will start to become more interesting if we have more neurons, but let's first do this for all single neurons first. So here's a function that should return the posterior:

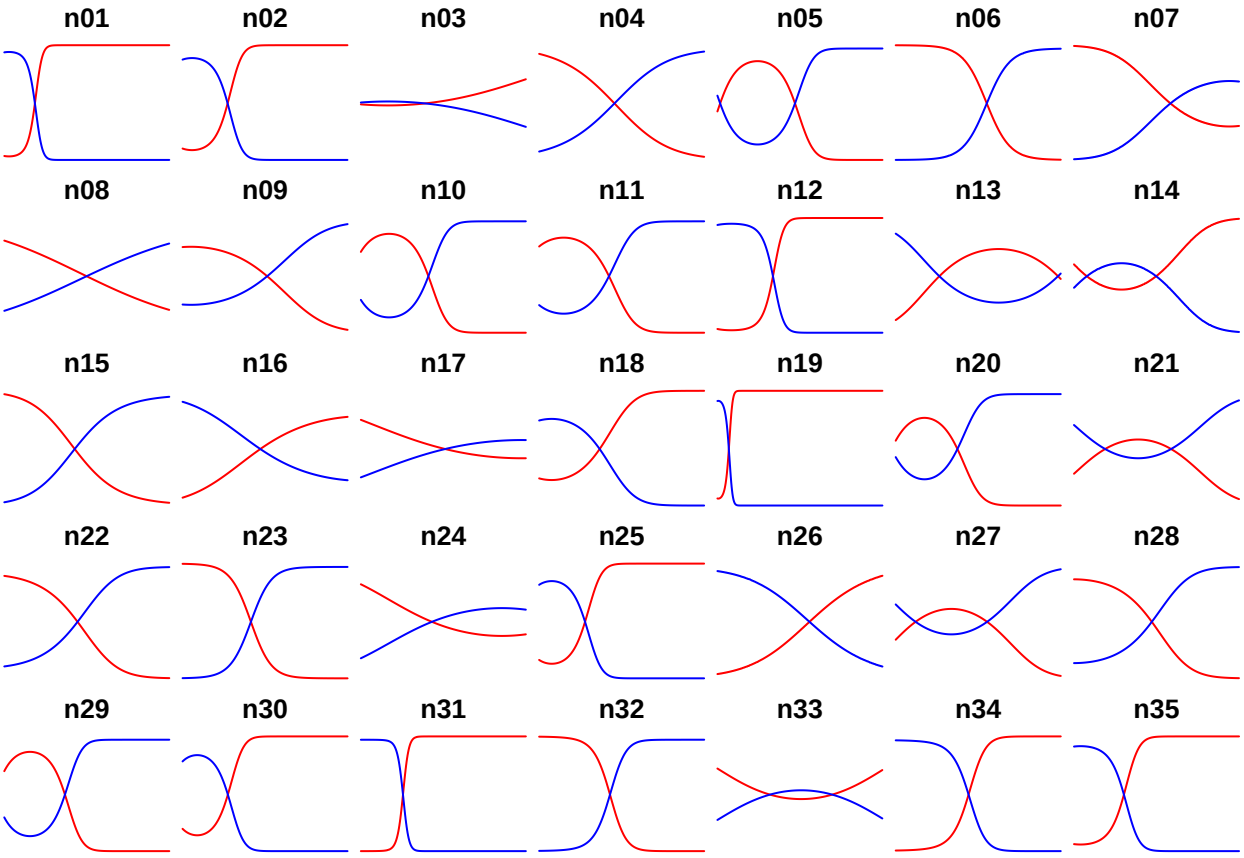
```
probabilityRightNeuron <- function(ratedist,neuron,rate) {

  r.idx   <- which(ratedist$neuron == neuron & ratedist$movement == 'right')
  r.mu    <- ratedist$mu[r.idx]
  r.sigma <- ratedist$sigma[r.idx]

  l.idx   <- which(ratedist$neuron == neuron & ratedist$movement == 'left')
  l.mu    <- ratedist$mu[l.idx]
  l.sigma <- ratedist$sigma[l.idx]

  b.idx   <- which(ratedist$neuron == neuron & ratedist$movement == 'both')
  b.mu    <- ratedist$mu[b.idx]
  b.sigma <- ratedist$sigma[b.idx]

  p_right      <- 104/(101+104)
  #p_right      <- 1

  #p_rate       <- dnorm(rate, mean=b.mu, sd=b.sigma)
  #p_rate       <- dnorm(rate, mean=r.mu, sd=r.sigma) + dnorm(rate, mean=l.mu, sd=l.sigma)
  p_rate       <- (dnorm(rate, mean=r.mu, sd=r.sigma) + dnorm(rate, mean=l.mu, sd=l.sigma))/2
  p_rate_right <- dnorm(rate, mean=r.mu, sd=r.sigma)

  return((p_rate_right * p_right)/p_rate)
}

probabilityLeftNeuron <- function(ratedist,neuron,rate) {

  r.idx   <- which(ratedist$neuron == neuron & ratedist$movement == 'right')
  r.mu    <- ratedist$mu[r.idx]
  r.sigma <- ratedist$sigma[r.idx]

  l.idx   <- which(ratedist$neuron == neuron & ratedist$movement == 'left')
  l.mu    <- ratedist$mu[l.idx]
  l.sigma <- ratedist$sigma[l.idx]
```

```
  p_left      <- 101/(101+104)

  p_rate       <- (dnorm(rate, mean=r.mu, sd=r.sigma) + dnorm(rate, mean=l.mu, sd=l.sigma))/2
  p_rate_left <- dnorm(rate, mean=l.mu, sd=l.sigma)

  return((p_rate_left * p_left)/p_rate)
}
```

Apparently, the probability of observing this firing rate is not well approximated by creating 1 distribution for the neuron based on all data (as I thought) but is better done by **adding** the probabilities of observing the firing rate in the two distributions, and dividing by two (although I wonder if this should be weighted).

Anyway let's plot this for all those neurons, as I tweaked the above function by looking at the figure, and then seeing what went wrong.

```
par(mfrow=c(5,7),mar=c(0,0,2,0))

for (neuron in unique(spikedist$neuron)) {

  xlim <- range(neuronrates[,neuron])

  x <- seq(xlim[1],xlim[2],.01)

  r.y <- probabilityRightNeuron(ratedist=spikedist,neuron=neuron,rate=x)
  l.y <- probabilityLeftNeuron(ratedist=spikedist,neuron=neuron,rate=x)

  plot(-1000,-1000,main=neuron,xlim=xlim,ylim=c(0,1),bty='n',ax=F)

  lines(x,r.y,col='red')
  lines(x,l.y,col='blue')
}
```



The y-axis in all these subplots goes from 0 to 1, and is the probability that the movement was to the right (red curves) or to the left (blue curves), given the firing rate of the neuron (x-axis), where the x-axis is scaled to the range of firing rates for each neuron. So where the curve is higher the chance of a movement to the right is higher. Conversely, when the curve is lower (than 50%) the chance of a movement to the left gets higher.

Neurons with a clear switch from 0 to 1 are perhaps the most informative neurons, e.g. neurons 1 and 6, and many others. Those three neurons we noticed ealrier, because they had largely overlapping normal functions (3, 18 and 33) stand out in this figure as well, as they don't show a clear switch, but instead they are largely flat. We can see a very similar pattern for neurons 17 and 24, and when you look at them in the earlier plots, they have overlapping normal distributions too. Then there is another "class" of neurons that I can distinguish here. Those look like neuron 5, where the chance of a rightward movement first increases with increasing firing rate, but then decreases again. If you look closer at their normal distribution functions, you will see that this can be explained because the distributions overlap a great deal (though not completely) but one distribution clearly has a larger standard deviation than the other.

We still haven't answered this part of the question: *How well do likelihood and posteriors code for movement direction?* This sounds like we need to calculate decoding performance for each neuron. I'm going to assume we will not need to do a leave-one-out type solution, but can just see how many of the movements in the dataset are correctly decoded by the distributions we calculated for each neuron separately. That is, we get the ground truth from the first column of the raw data (`neuronrates$direction`). Then we loop through the neurons, and for each of them we get the probability of a rightward movement given the whole column of actual spike rates for that neuron, and compare it with the ground truth.

```
# we want rightward movements as 1, and leftward as 0:
groundtruth <- as.numeric(neuronrates$direction) - 1

# we will store the decoding performance of every neuron:
```

```r
neuron <- c()
performance <- c()

# loop through neurons
for (neuron.ID in unique(spikedist$neuron)) {

  # get the neurons actual spike rates:
  spikerates <- neuronrates[,neuron.ID]

  # calculate the posteriors
  posteriors <- probabilityRightNeuron(ratedist=spikedist,neuron=neuron.ID,rate=spikerates)

  # decoding is correct when:
  # the posterior (of a right movement) is higher than 0.50 if the movement was to the right
  # and also correct when the posterior is lower than 0.50 and the movement was to the left:
  decodingperformance <- groundtruth == round(posteriors)

  # store the info:
  neuron <- c(neuron, neuron.ID)
  performance <- c(performance, mean(decodingperformance))

}

# put it all in a data frame:
neuronPerformance <- data.frame(neuron, performance)

kable(neuronPerformance)
```

neuron	performance
n01	0.9365854
n02	0.8634146
n03	0.5170732
n04	0.6780488
n05	0.7951220
n06	0.8439024
n07	0.6390244
n08	0.6146341
n09	0.6390244
n10	0.8146341
n11	0.7463415
n12	0.9414634
n13	0.7414634
n14	0.6341463
n15	0.7170732
n16	0.6536585
n17	0.5804878
n18	0.6878049
n19	0.8878049
n20	0.7317073
n21	0.5317073
n22	0.7804878
n23	0.8780488
n24	0.5853659

neuron	performance
n25	0.8195122
n26	0.6878049
n27	0.5853659
n28	0.7024390
n29	0.8097561
n30	0.8097561
n31	0.9902439
n32	0.9024390
n33	0.6097561
n34	0.8634146
n35	0.8780488

```
summary(neuronPerformance$performance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5171  0.6390  0.7415  0.7456  0.8537  0.9902
```

The minimum performance lies at 51% and the maximum at 99% with the mean and median both just below 75%. So there is a lot of variation in how informative the neurons are, but all of them are above chance. This already means that combining them will always be beneficial. In particular it would be interesting to see how well combined performance can be if we use the five "least informative" neurons (these are the ones that are less then 60% accurate).

```
worstfive <- as.character(neuronPerformance$neuron[which(neuronPerformance$performance < .60)])
worstfive
```

```
## [1] "n03" "n17" "n21" "n24" "n27"
```

However, this is the end of the part about individual neurons.

Population decoding: iterative Bayesian (=Kalman)

In this part, we will see how we can make use of the information provided by all neurons' firing rate simultaneously.

Now combine the individual neurons posteriors. Here, previous decoding performance will act as a prior to the next iteration. Note, this is only correct if all observations are statistically independent, which we will assume here.

I'm going to assume we don't have to create new posterior distributions that combine the information from several neurons, but can simply use the posterior distributions we just computed. That is, we can start with a prior of 104 / (101 + 104) that the movement is to the right, and then use likelihood provided by one first neuron to update our beliefs in the posterior. We can then do the same using a second neuron and so on, until we have used all 35 neurons, and then we move on to the next movement / trial.

We can use the neurons in a random order on every trial (and even repeat that several times for a confidence interval?) to see how the decoding performance increases with more neurons in the mix. Since individual neurons are all above chance, combined decoding performance should also be above 50% (and get close to 100%).