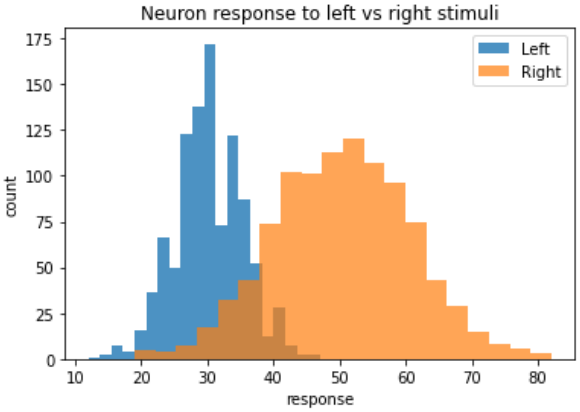
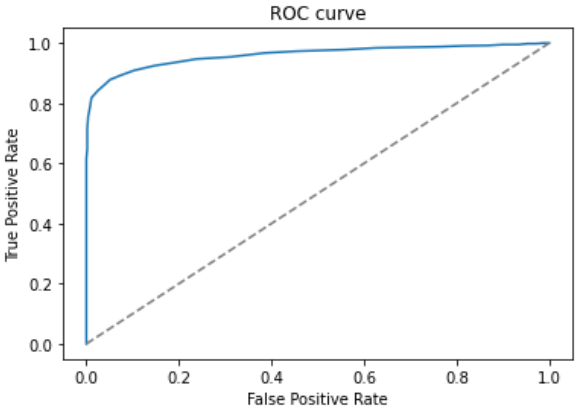
Assignment 5

1) ROC

a. I decided arbitrarily that H0 is “left” and H1 is “right”. The neuron is a great discriminator in my opinion, as can be seen from the ROC curve which is very distant from the indifference diagonal. An approximation of the AUC I got 0.96, almost perfect.



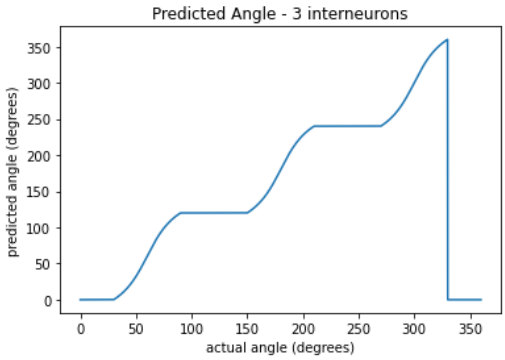


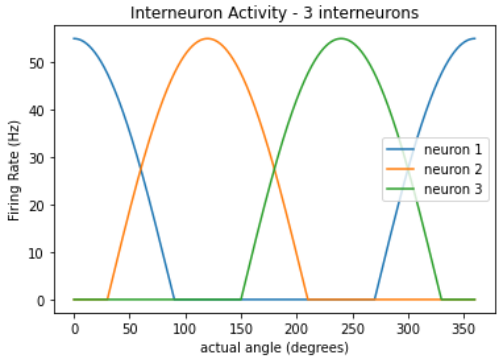
b. The criterion providing about 80% true positives rate (for “right”) is about 41. The false positive rate for this criterion is 11%.

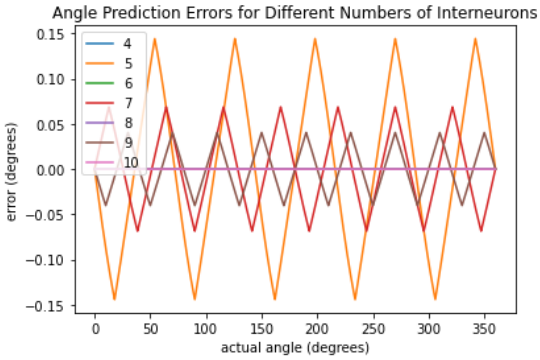
2) Population vector

a. Code Attached.

b. For 3 interneurons, we can clearly see that there are large errors in the prediction (although still not that bad). It is important to remember that the plot ‘wraps around’ the y axis, so the error at 11pi/6 is actually small (prediction is 0 while degree is close to 360). We can see also that more interneurons lead to less errors, and even numbers of interneurons lead to less errors compared to odd numbers of neurons. Obviously the closer the actual angle is to one of the neuron’s preferred angles the less error we have.

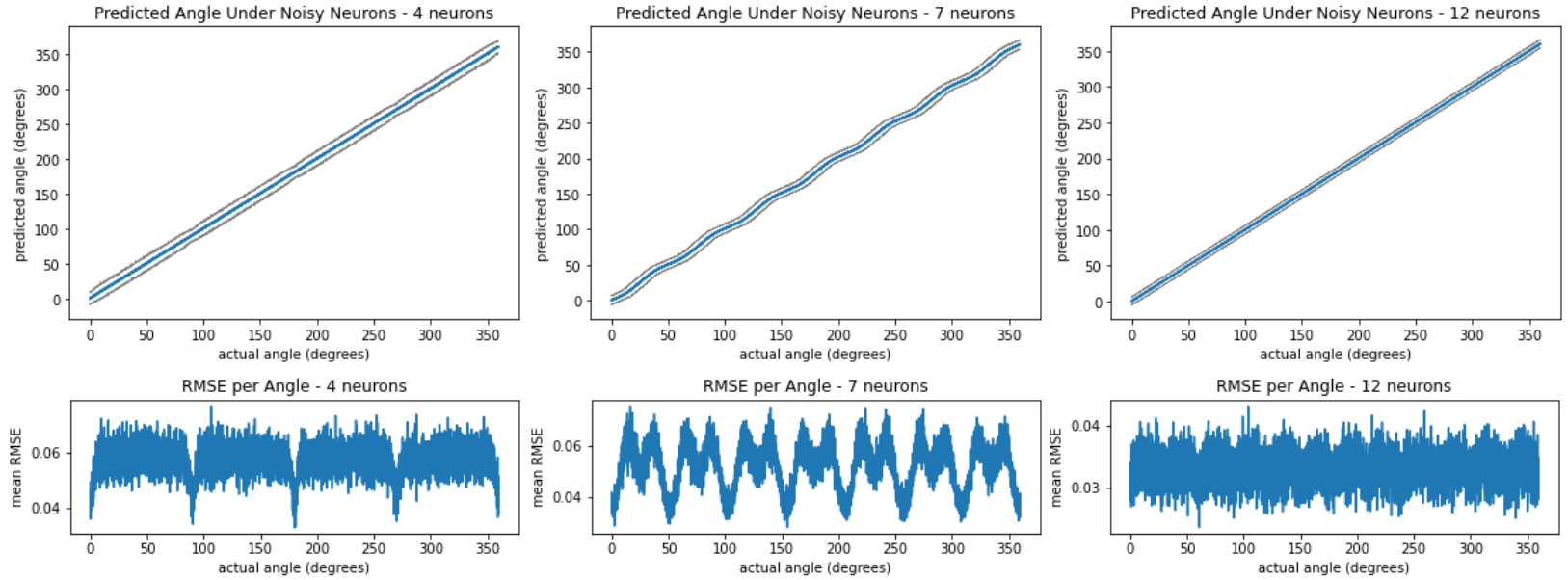






c. I tried with 4, 7, and 12 neurons. The error lines in the plots indicate ±2.5 stds. A perfect predictor is just a diagonal (I didn’t add a line as it made the figure look ugly). As can be seen, the more neurons the less error, generally, but also even number of neurons have

an advantage (I think this is because of interfering phases). Also, lowest RMSE occurs in the favorite angles of the neurons, as was expected.



3) Estimators MAP and ML

a. MAP estimator is maxk for p(x|k)*p(k)
p(time differences = 2, 3, 4, 5, 8 | k)*p(k)
Assuming independence (geometric distribution)
p(2|k)*p(3|k)*P(4|k)*p(5|k)*p(8|k)*p(k)
$(1-k^2)^{2-1}*e^{0.1k}*(1-k^2)^{3-1}*e^{0.1k}*(1-k^2)^{4-1}*e^{0.1k}*(1-k^2)^{5-1}e^{0.1k}*(1-k^2)^{8-1}e^{0.1k}*e^{-0.5k}$
$(1-k^2)*(1-k^2)^2*(1-k^2)^3*(1-k^2)^4*(1-k^2)^7$
$(1-k^2)^{17}$
Easy to see that maxk is 0.

b. Modified the calculation to find new maxk.
p(time differences = 2, 3, 4, 5, 8 , 38| k)*p(k)
Assuming independence and cleaning up a bit
$(1-k^2)^{54}*e^{0.1k}$
take log (preserves monotonicity)
$54*\log(1-k^2) + 0.1k$
Using WolframAlpha, maxk is now = 0.000925

c. Modified the calculation to find new maxk.
p(time differences = 2, 3, 4, 5, 8 | k)*p(k)
Assuming independence and cleaning up a bit
$(1-k^2)^{17}*e^{0.5k}*0.5$
multiply by 2 and take log (preserves monotonicity)
$17*\log(1-k^2) + 0.5k$
Using WolframAlpha, maxk is now = 0.0147

d. MLE estimator is maxk for p(x|k). It's easy to see that omitting the prior is equivalent to using uniform priors, therefore, the maxk will be the same as c, 0.0147.

Appendix – code for 2A

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 10 14:05:03 2022

@author: Benjamin
"""
#%% Assignment 5 - Q2
import numpy as np

#%% A
def roachNeurons(r0, n, ai, a):
# The inputs of the code:
#     r0 - baseline firing rate
#     n - number of interneurons
#     ai - vector of the preferred angle α of each neuron
#     a - real angle
# The output of the code:
#     a_hat - the prediction of the neurons using population vector
    neurons = [np.maximum(r0*np.cos(a-ai[i]),0) for i in range(n)] # firing rate p
    x_hat = np.sum(neurons*np.sin(ai)/r0) # decode x direction
    y_hat = np.sum(neurons*np.cos(ai)/r0) # decode y direction
    a_hat = np.arctan2(x_hat, y_hat) % (2*np.pi) # vector to angle

    return a_hat
```