Computational Neuroscience Homework

HW4: Neural Decoding

# Exercise 1) Analysis of neuron in MT

## A) For 15% correlated motion of stimulus (i.e. easy task), you found =(38,4), and =(17,4). Generate the sample response of the neuron for 1000 trials where up and down stimuli are given half and half. Plot the histogram of firing rate in all trials as in Fig 1.

By generating the sample response of the neuron for both up and down stimuli, we used ‘randn’ function. The plotted result is as below:

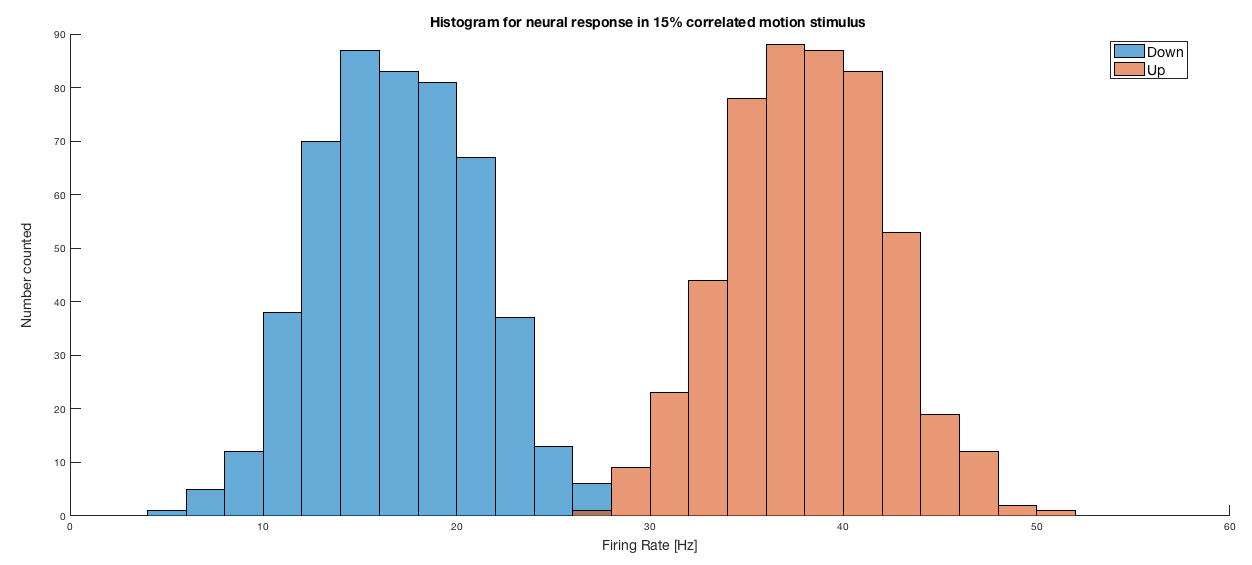


Figure . The histogram of the sample response of neuron for both up and down stimuli (15% correlated). The neural response is easily discriminated in this case.

As confirmed in Figure 1, the neural firing rate for two different conditions were easy to be distinguished. To support my claim, we can use the discriminability index, which is defined as

Here, we know that =38, =17, and =4. Thus, 21/4=5.25, which agrees with my claim that the neural firing rate for up and down stimuli is easy to be discriminated from each other.

Now, the question is, whether we really need to collect about 500 samples for each case. If we are fine with 200 samples, collecting another 300 samples would be a waste of time.

Some would suggest methods to find the minimum number of samples needed for representing the distribution, such as measuring its normality or fitting p-value. However, this requires prior knowledge about the distribution the sample follows. Let’s just think for a moment. In reality, can we actually have prior knowledge about the sample’s distribution? No. Thus, we might have to find a different strategy.

The mean and variance of the sample is one of the most well-known quantity that represents the distribution, and if the number of the sample is ‘enough’ to represent the sample’s mean and variance, those parameters should converge to some value. Thus, my suggestion would be to find the number of samples where the sample’s mean and variance starts to fix to some point. In other words, if we declare as the mean value of samples and as the variance of samples, then

and

Then, I arbitrarily defined a quantity such as

When the parameter becomes lower than some threshold, we instantly stop to collect samples.[[1]](#footnote-1)

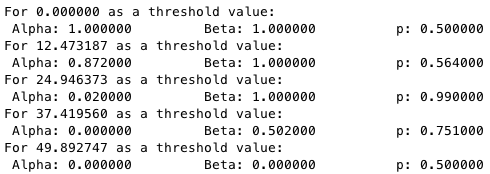
Applying this method, I counted the number of the samples collected for the up-response. This case, the average samples collected for 40 trials were as below:



When we ran the code several times, the exact number always changed but it was mostly somewhere near 140. Thus, we can expect that 140 samples for both up and down stimuli might be enough, though the exact logic for deciding the threshold value should be further discussed.

## B) Now choose 5 arbitrary numbers as your threshold z of firing rate for decoding in above case. Implement a code to calculate the “hit rate” and “false alarm rate” , as in the lecture note. Show your estimated , and the probability of correct answer , for all z values above.

When we have to choose only 5 arbitrary numbers as the threshold value of firing rate, it would be better to show how and changes as we change the threshold value. Thus, I collected 5 threshold values, starting from 0, and ending up with the value where the firing rate is maximized. The interval between each threshold value was set as a constant, using ‘linspace’ function. Below is the execution result:



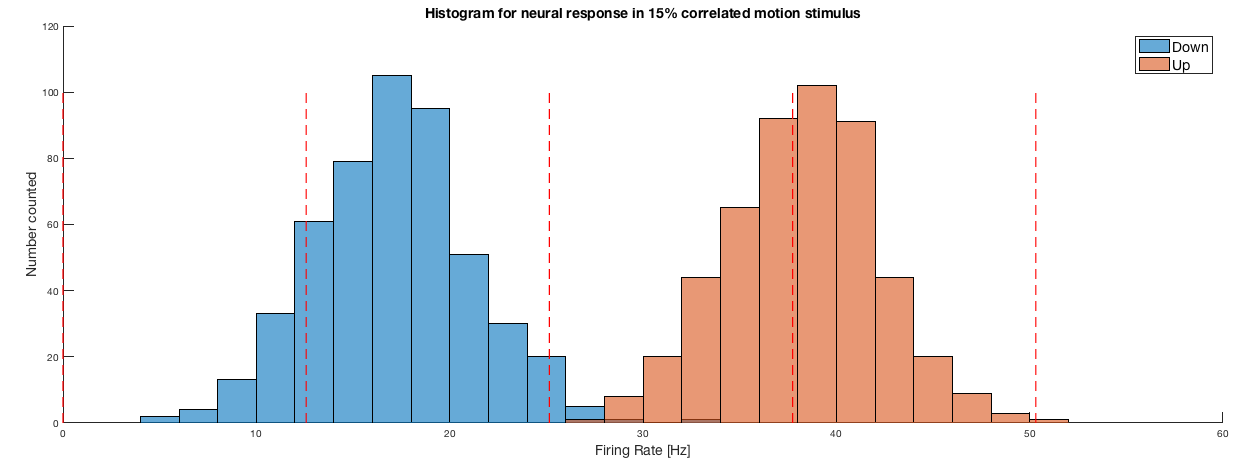


Figure . The histogram of the neural response firing rate for both down and up stimuli. The red lines are the threshold values set in the ‘Prob1b.m’ code.

As we can confirm from the execution result, for very low threshold, both alpha and beta becomes 1, so the probability becomes 1/2. Also, when we set the threshold too high, the alpha and beta value becomes 0, and again the probability becomes 1/2. To have the probability of discrimination to be maximized, beta should be 1, whereas alpha should be 0. In this case, the probability was maximized for the third value, and this seems quite obvious when we look at Figure 2. (The third threshold value is closest to the boundary between two different responses.)

## C) For 1% correlated motion of stimulus (i.e. hard task), you found =(22,5), and =(19,5). Repeat the process in a) and b) for this condition

Using the same method used in a) and b), we can obtain the result as below:

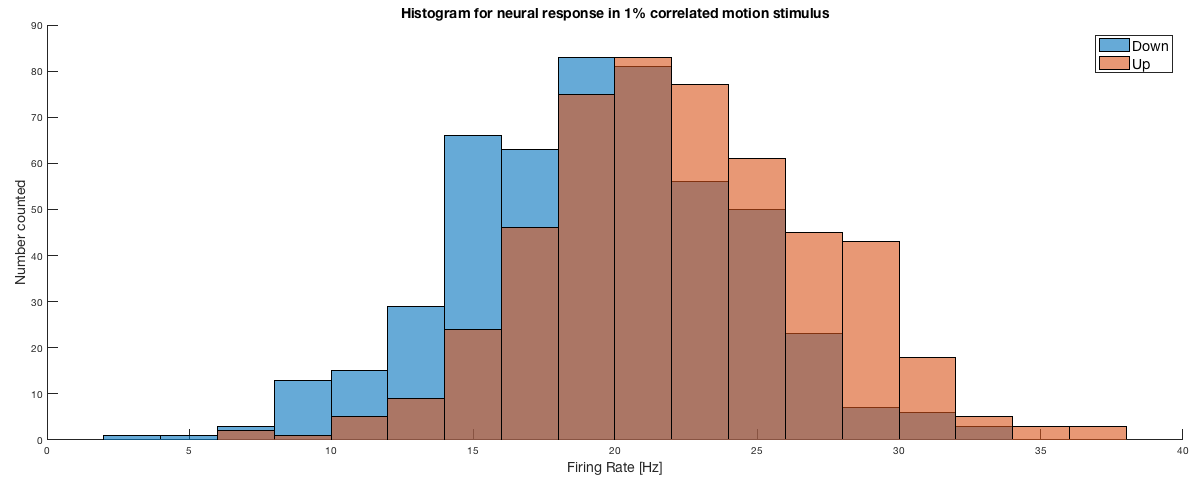


Figure . The histogram of the sample response of neuron for both up and down stimuli (1% correlated). The neural response is easily discriminated in this case.

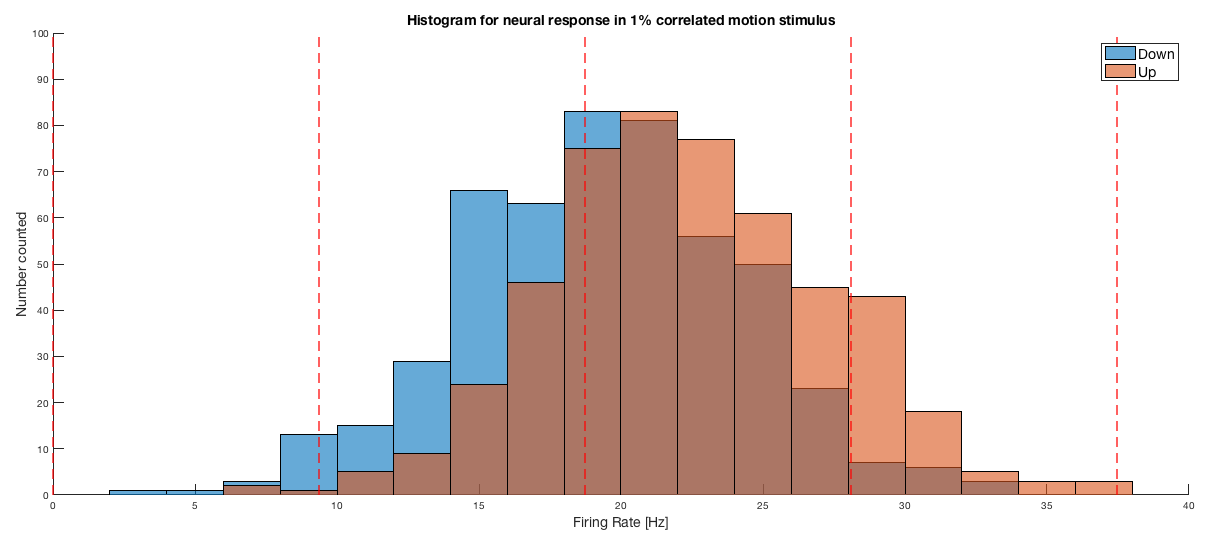
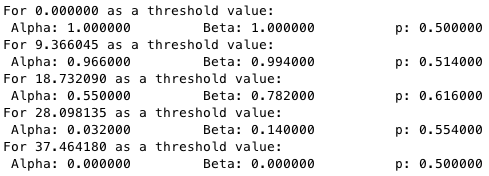


Figure . The histogram of the neural response firing rate for both down and up stimuli. The red lines are the threshold values set in the ‘Prob1c.m’ code.

Also, the alpha, beta, probability for each threshold value is:



In this case, as seen in Figure 3 and 4, the neural response for up and down stimuli was not significantly different. The probability was 0.5 when threshold value was too low or too high as in the earlier case. In the third threshold value, the probability was maximized compared to other threshold values, but it did not exceed 0.9. This is because the neural response for up and down stimuli isn’t discriminated well as in the previous example. (Here, =22, =19, and =5. Thus, 3/5=0.6, which is much lower than the previous case.)

## D) plot the ROC curve for above two cases, 15% and 1% correlated motions. For this, you may try as many z values as you want.

Now, by using many threshold z values, we collect alpha and beta values for both 1% and 15% correlated motions, and plot them in a ROC curve as below:

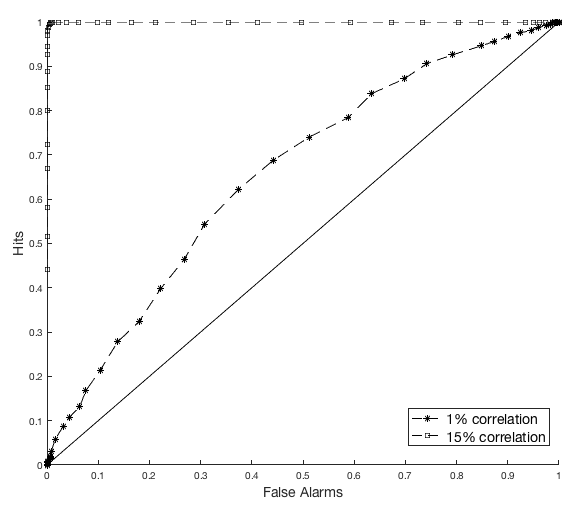


Figure . ROC curve for both 1% correlation and 15% correlation. The area below the curve is nearly 1 in the 15% correlation motion, whereas it is much lower, but at-least higher than 1/2 in the 1% correlation motion.

The probability of getting the correct answer is

Thus, the area below the ROC curve is the probability of getting the correct answer. For the case of 15% correlation, the probability is nearly 1. However, it is much lower in the case of 1% correlation, but at-least it is higher than 1/2. Here, we can confirm the fact that 1% correlation motions are much harder to be distinguished compared to 15% correlation.

## E) Choose the optimal value of z for 15% correlated motion. Explain your method to make this.

As we have seen in ‘Prob1a.m’, we have seen that the average and variance of the sample starts to converge when approximately 140 samples are collected. Based on this fact, we can ambitiously say that we can approximate the distribution the sample follows when we collect 140 samples. Assuming that we ‘know’ the sample follows a normal distribution, let’s first find the variance and the average of the up response and down response. Then, the normal distribution would be , Then, when we set the z value where summation of

, which is double of the probability becomes maximized. Then, for the rest of the task, we can use this threshold value to classify whether the stimulus was up or down for samples further obtained. The accuracy of this method for 1000 samples further collected is as below:

../../../../../../Screen%20Shot%202016-05-18%20at%208.26.13%20PM.png

Thus, we can say that this method is quite efficient and reasonable to select the threshold value for discrimination.

# 2) Neural decoding of V1 neuron

## A) Plot the tuning curve of a sample neuron with . For N=100 sample neurons of different (randomly chosen), use the population vector method to decode the stimulus direction, for actual direction of stimulus is 60 .

The tuning curve of a neuron with would be as below:

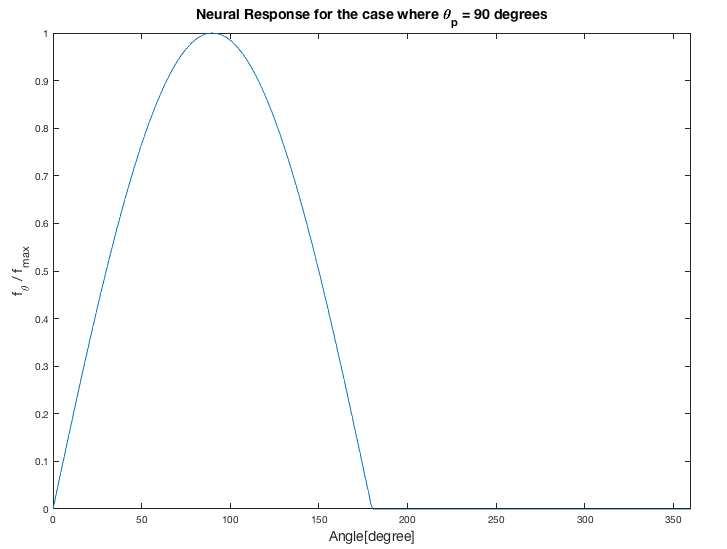


Figure . The tuning curve of the neuron with = when the neural response function is .

Then, using 100 randomly sampled neurons, (In other words, 100 randomly selected ) we can use the below method to decode the actual direction:

Here, to make this easier, I used the complex domain: based on the fact that

would be , and we can simply add them to determine the direction. Below is the execution result of ‘prob1a.m’, based on this method.



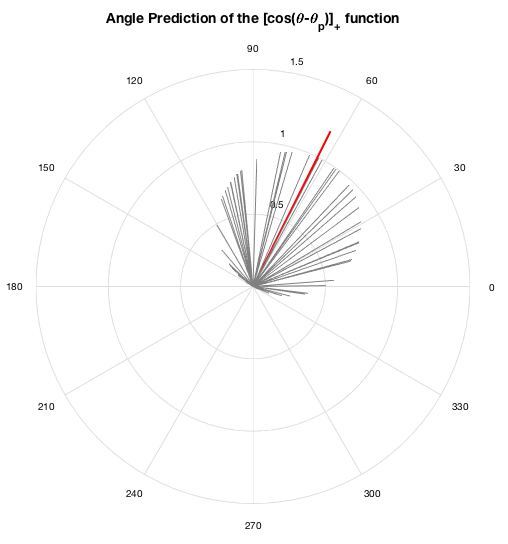


Figure . The angle prediction in the case where the response function is . The gray lines are the directional vectors of each neuron, and the length of each gray line is proportional to its normalized response. The red line is the predicted direction which is obtained by summing up all response vectors. The length of the red line is fixed to 1.2, to visualize it with the gray line.

The predicted angle was 63.47 degrees, which was very similar to the real direction. (error rate: 5.783%)

## B) Repeat the process in a) when the tuning curve is sharpened as . Explain any changes in the result and discuss how the nonlinearity of neural response can improve neural population decoding.

Now we repeat the same process as in a) for a sharpened tuning curve. First of all, the tuning curve of a neuron with is,

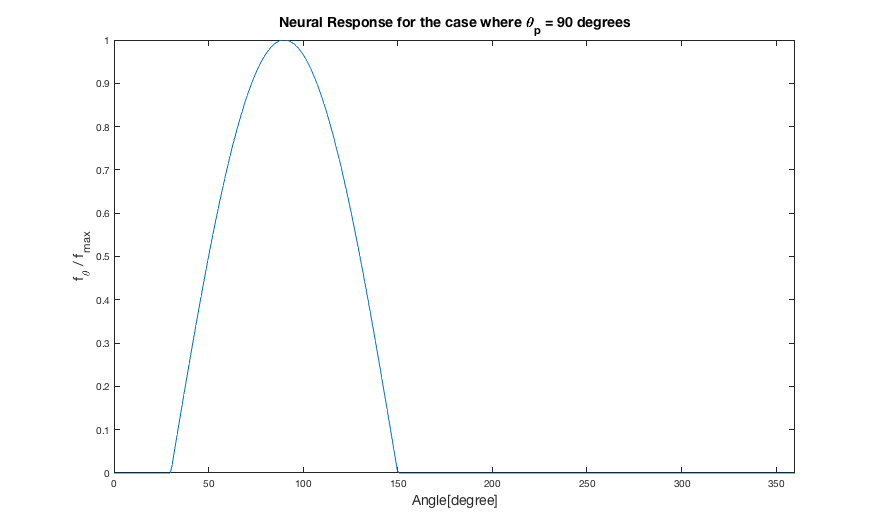


Figure . The tuning curve of the neuron with = when the neural response function is

The tuning curve seems to be narrowed in this case, compared to .

Now let’s look at the direction decoding of this sharpened neurons:

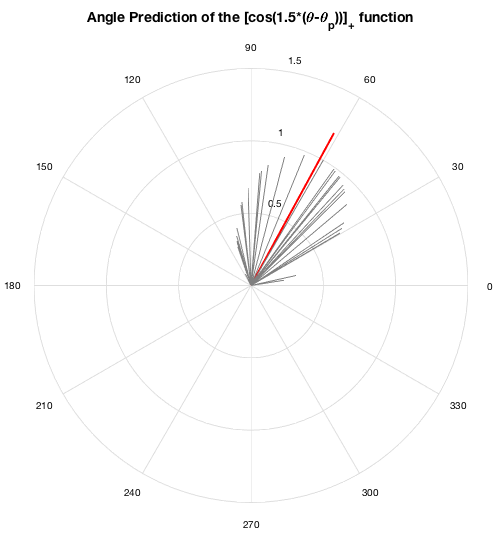
**

Figure . The angle prediction in the case where the response function is . The gray lines are the directional vectors of each neuron, and the length of each gray line is proportional to its normalized response. The red line is the predicted direction which is obtained by summing up all response vectors. The length of the red line is fixed to 1.2, to visualize it with the gray line.

Figure/%5BProb2b%5D%20Predicted%20Angle.png

This case, the predicted angle was 61.44 degrees, which was even better compared to the earlier case. (error rate: 2.4%) However, this didn’t feel to be enough for supporting the claim that ‘narrower response function is better than wider response function’. To make my assumption to be supported with more solid evidence, I measured the average root-mean-squared error for 1000 trials each, varying the width from 180 degrees ( case) to less than 40 degrees:[[2]](#footnote-2)

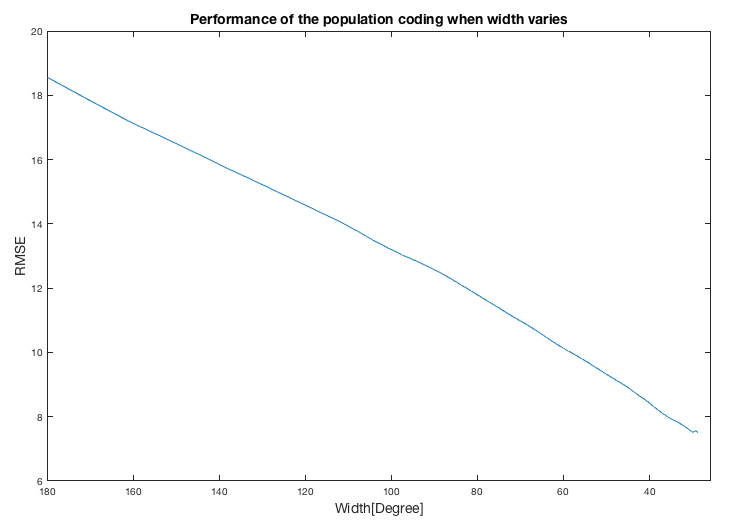


Figure . The average RMSE depending on the width of the tuning curve of neurons. As the width gets narrower, the RMSE decreases, which would mean that they have a ‘better performance’ in predicting the direction of the stimulus.

As seen in Figure 10, as the width gets narrower, the RMSE decreased to some level. This means that it is ‘easier’ to predict what the direction of the stimulus is using the same number of neurons when the width of the tuning curve gets narrower.

Now, let’s focus on the importance of ‘non-linearity’. The neural response of the neuron in V1 is not linear: they only activate in some range of angles, and intuitionally speaking, we can say that this is ‘narrowing’ the tuning curve’s width. Thus, neurons to specifically fire in some certain range makes an improvement in decoding the direction of the stimuli, and also the fact that the firing rate shows a ‘cosine’ curve rather than a constant function (neurons have same response when they are activated) gives more specificity of the neuron to an angle. Thus, as the width of the tuning curve gets narrower, the specificity of each neuron to an angle would increase, and this would improve the system to have a better angle prediction.

However, another question that came up to me was that whether ‘narrowing’ the tuning curve is the only way to improve prediction. Unfortunately, the tuning curve of the neuron is the physiological property of the neuron itself, not what we can control. Would there be any other way to predict the direction of the stimuli?

To solve this question, I added two different methods: First, rather than adding the directional vector with the length being proportional to the normalized firing rate, I decided to change the length of the vector to be the square of the normalized firing rate. This enables us to give higher weight in directions with high responses, whereas lower weight in directions with low responses. Second, If the neural response was lower than the threshold, they were eliminated in summation. Using this method, we compared our performance with neural decoding using original method with and tuning curve.[[3]](#footnote-3)

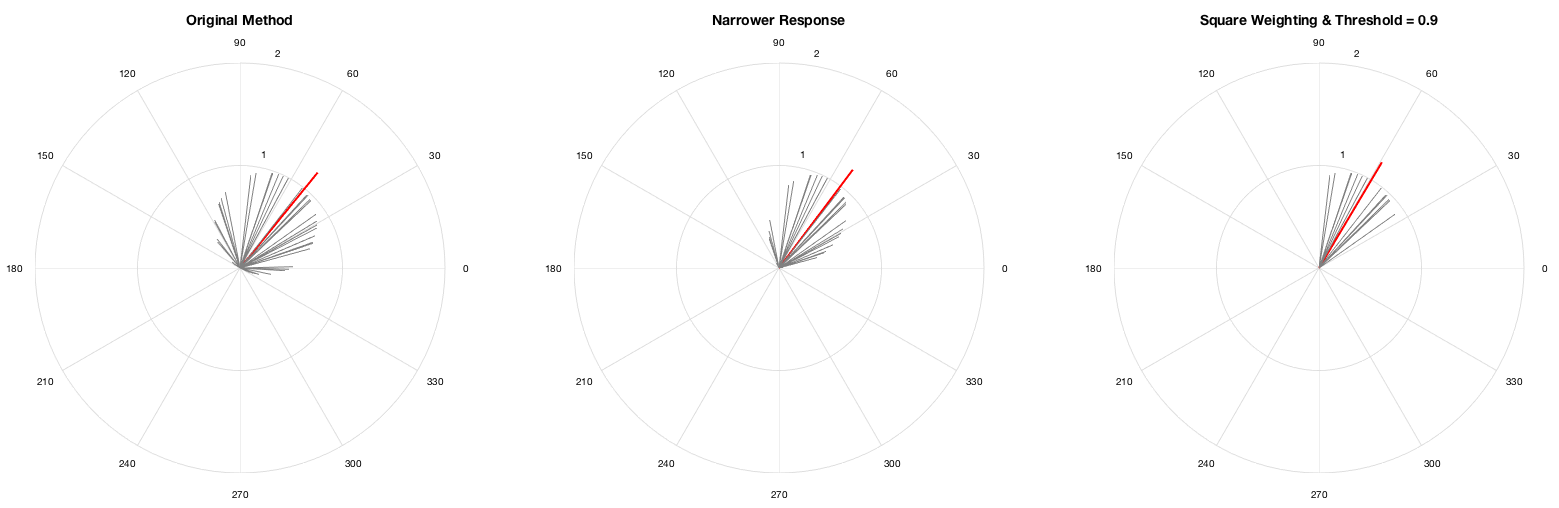


Figure . The angle prediction in the case where the response function is and original method is used (left), where the response function is and original method is used (middle), and the case where response function is and modified method is used (right).

Figure/%5BProb2b%5D%20Performance%20of%20methods.png

Figure . Average root mean square error for original response function, narrowed response function both using original decoding method, and original response function using modified decoding method. (square weighting + threshold method: threshold = 0.9) The RMSE is lower in the modified decoding method compared to both response function using original decoding method, which means there is possibility for developing new decoding methods which can ‘efficiently’ improve prediction accuracy.

1. In my code, this threshold value is decided after running the code many times. The threshold was set at the value where the number of samples for each trial had a low variance, but no precise logic supports how to decide the threshold. Thus, further discussion and study are required. [↑](#footnote-ref-1)
2. This is done in the ‘Prob2b\_Discussion2.m’ code. [↑](#footnote-ref-2)
3. This is done in the ‘Prob2b\_Discussion.m’ code. [↑](#footnote-ref-3)