Data Analytics - Assignment #4

Nishant Kumar (Sr.No: 21495)

October 2023

Visual NeuroScience

1 Perceptual Distance :

The initial assignment problem involves assessing the Relative Entropy distance for various image pairs within the oddball detection experiment. Additionally, you are required to compute both the L1 distance and the Average Search Delay, utilizing the firing rate data of the neurons. The second task involves determining the ratio between the Arithmetic Mean and Geometric Mean of the spreads of the products obtained from the distance metrics.

2 Part 1 & 2: Connection between search delay and distance:

In a dataset, there are 24 columns, and these columns are divided into 4 sets. Each set has 6 groups within it. The dataset represents an experiment involving 24 individuals. Each person was presented with 6 different versions of an oddball-distractor pair. So, in total, there are 144 rows of data in this dataset.

To calculate the average search time for a specific column, we use this formula:

$$AvergeSearchDelay_c = \frac{1}{144}\sum_{i=1}^{144}(SearchData_c^i - BaselineReactionTime)$$

In the equation mentioned, the "i" as a superscript represents data for a specific row, and the "c" as a subscript represents data for a particular column. The baseline time it takes to perform a basic search for this problem is 328 milliseconds.

We're using data from neuron firings to calculate relative entropy and L1 distances. These values are calculated based on definitions explained in our lecture slides. The firing data is organized into 30 columns divided into three sets. Each of the first three sets has 6 groups, and the last set has 12 groups.

Following the instructions, we calculate a total of 18 relative entropy values for the columns in the first three sets. For the last set with 12 columns, we use a more complex search method called a "compound search strategy" to find the distance measurements we need.
Compound Search Strategy: To calculate relative entropy (or L1 distance) values for a pair like Bug-Worm, we need to consider different scenarios. Imagine you have data for a column

where "bug" is the oddball and "worm" is the distractor. There are four possible situations:

- The oddball - "Bug" and the distractor is "Worm."
- The oddball - "Big" and the distractor is "Worm," but they're not flipped.
- The oddball - "Bug," but it's flipped (written backwards), and the distractor is "Worm."
- Oddball - "Bug" and the distractor "Worm" are flipped.

Now, these four cases are equally likely to happen. So, we calculate the relative entropy for each of these cases separately. Then, we find the average of these relative entropy values for that particular column. We repeat this process for all 12 columns, considering the 4 different sets. In total, we'll end up with 12 columns multiplied by 4 sets, giving us $12/4 * 4 = 12$ data points.

Results: In the end, we were able to collect 24 data points on the average search time together with the corresponding relative entropy distance (and L1 distance). According to our theory, the reciprocal of search delay is inversely correlated with perceptual distance. The next two figures show how inverse search time (s^{-1}) and relative entropy distance (as well as s^{-1} and L1 distance) relate to one another.

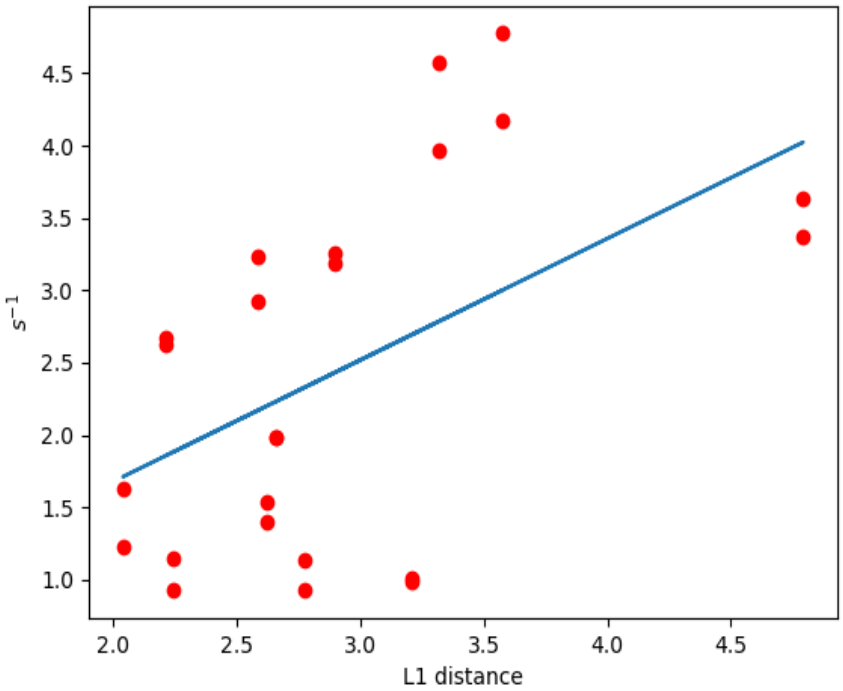

Figure 1: Scatter plot showing the correlation between inverse search delay and L1 distance measure. The blue line represents the best fit by minimizing squared error loss.

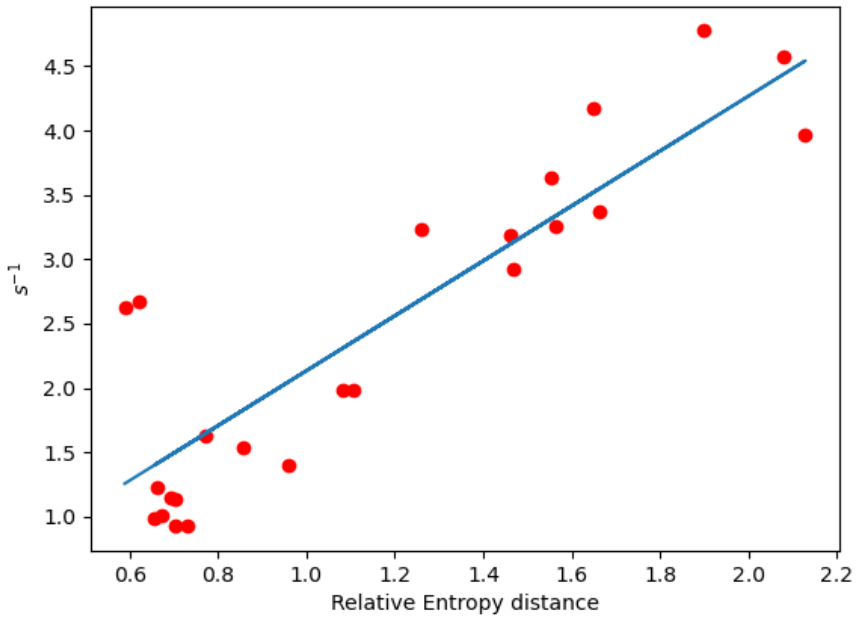

Figure 2: Here is a scatter plot depicting the relationship between the relative entropy distance measure and the inverse search latency. The blue straight line on the plot reduces the squared error loss and provides the best fit for the points.

When you compare the two graphs, it's easy to see that the straight line on the graph is a much better fit when using the relative entropy metric. So, it's pretty clear that relative entropy is a more suitable metric than the l1 distance.

The second problem wants us to compare two ways of measuring the spread of data: one using "search delay" multiplied by "relative entropy" and the other using "search delay" multiplied by "L1 distance." In the first case, the value is **1.129**, and in the second case, it's **1.042**. When we calculate the ratio of the average to the geometric mean (AM/GM ratio), it's lower for the first case. This tells us that **relative entropy** is a better way to measure the spread of data compared to **L1 distance**.

3 Part-III: Fitting Gamma Distribution

In this problem, we're trying to find a good way to describe the time it takes for searches to happen. We want to use something called a Gamma Distribution, which has two parameters: shape (α) and rate (β). Our job is to figure out what values for and work best to represent the search times.

We also need to compare the **cumulative distribution function (CDF)** of the original data with the CDF we get when we only use a part of the search time data. This helps us see how well our Gamma Distribution fits the real search time information.

To tackle the first part of this problem, I randomly chose half of the groups and found the

average and how much the data varies for each of them. This gives us 12 pairs of values (one for the average and one for the variation). I used a tool called Matplotlib to make a visual representation of these values. Here's what it looks like:

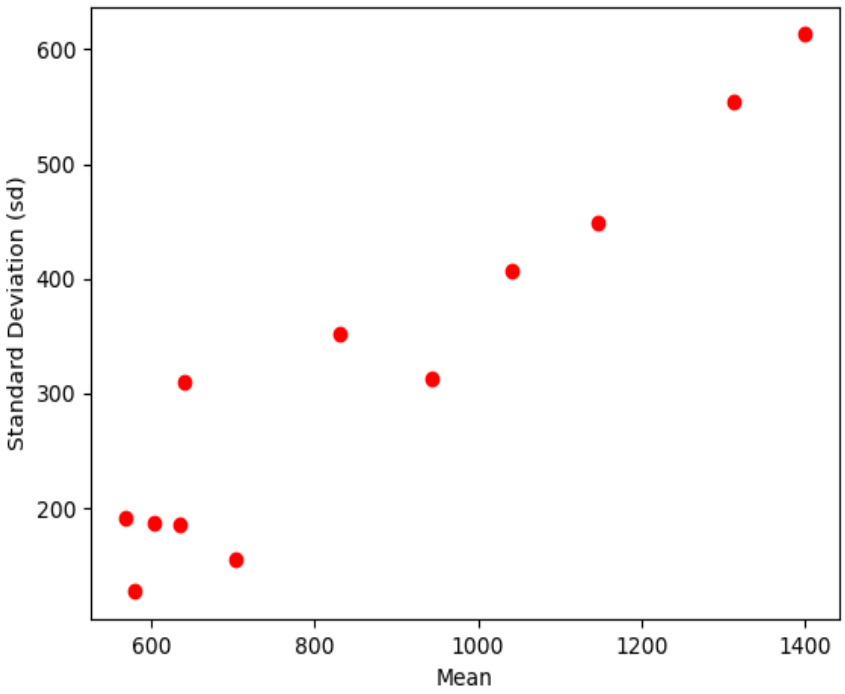

Figure 3: Displays a scatter plot of the average search delay plotted against its standard deviation. The standard deviation fluctuates almost proportionately to changes in the mean, suggesting a linear trend in the plot.

Estimation of Shape parameter: To figure out the shape parameter, we drew a straight line on the previous plot by trying to make it fit as closely as possible. We used a method called *numpy. polyfit* to do this. The steepness of this line tells us the **shape parameter**, and we found it to be **3.478**. So, we can say that α (alpha) = 3.716.

Estimation of Rate parameter: Next, from the 12 groups that still have search time data, I randomly picked half of the samples in each group. Then, I use these selected data points to figure out a parameter called the rate parameter. I do this by trying to find the best estimate using a method that minimizes the difference between the estimated variance and a linear function of the mean. To do this, I use a tool called numpy.polyfit. This is because for the gamma distribution, the Mean = $\frac{\alpha}{\beta}$, and the Variance = $\frac{\alpha}{\beta^2}$. So, the value for $\beta = \frac{Mean}{Variance}$.

Estimated value of **Rate Parameter is 0.00246**

Empirical CDF and Kolmogorov-Smirnov Statistic:
In the third part of the problem, we created a graph called a cumulative distribution function (CDF) using data points we hadn't used before. Additionally, we drew the CDF of a gamma distribution on the same graph, using the shape and rate parameters we previously calculated.

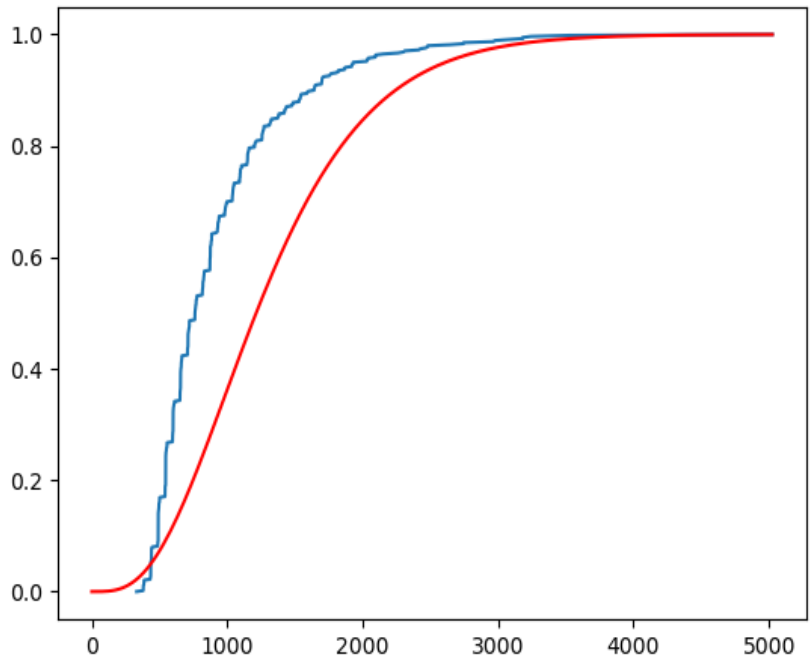

Figure 4: Shows the cumulative distribution function (CDF) of a gamma distribution. The blue curve represents the CDF generated from the data samples we had left. The red curve, on the other hand, represents the CDF of a gamma distribution, using the shape and rate parameters we calculated earlier.

In statistics, the Kolmogorov–Smirnov test (abbreviated as the K–S test or KS test) is a method that helps us measure how different two distributions are from each other. It works by comparing the distribution of our actual data with a known reference distribution or by comparing the distributions of two sets of data.

In our experiment, we wanted to see how well the empirical gamma distribution matched a gamma distribution we calculated using estimated shape and rate parameters. To do this, we used a tool called $scipy.stats.ks_2samp$. The result of this test is a number called the Kolmogorov-Smirnov statistic, which tells us how similar or different the two distributions are.

Kolmogorov statistic:: KStestResult(statistic=0.472, pvalue=8.28681258e-55)

If the K-S statistic is small, it indicates that we can accept the idea that the distributions of the two samples are very similar. So, with a high level of confidence, we can conclude that search delays indeed follow a gamma distribution.

4 Part-IV: Expected Search Time under controlled scenario

The goal of this question is to determine the minimum expected time it would take to find image 0 if it's in an unusual position at location l (hypothesis h = (l, 0, 1)). This is under a

unique condition where the person searching can decide where to start looking at the beginning of each time slot.

Approach: The minimum expected time it takes to find something when you're looking for a specific item (let's call it "oddball" with a value of 0) among a group of other items (distractors with a value of 1) can be estimated using the following inequality.

$$E_{\pi}^{0}[\tau] \geq \frac{D(P_{\pi}^{0} \parallel P_{\pi}^{1})}{max_{\lambda} \sum_{a=1}^{K} \lambda_a D(p_a^0 \parallel p_a^1)}$$

To make the expected search time as short as possible, we should focus on maximizing the denominator of a certain equation. This means we need to find a set of values represented by the vector $\lambda = \{\lambda_a : 1 \leq a \leq K\}$. Here, K stands for the number of actions or choices the subject can make. In this specific problem, the subject has the option to inspect any of the six locations in the image, making K equal to 6.

Since the opponent intentionally varies the oddball's position (it's not entirely random), and the subject can strategically choose where to look, it's advantageous for the subject to avoid revisiting locations they've already checked. In this scenario, if the subject decides where to look without considering their previous choices, they assign a value of $\lambda_a = \frac{1}{5}$ to each of the six possible positions, denoted as $a \in 1, 2, 3, 4, 5, 6$.

However, when the searcher has control, and the adversary follows the strategy mentioned earlier, we use the vector $\lambda = \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1$. By doing this, the denominator of the inequality becomes larger, resulting in a decrease in the expected search time. Consequently, it is more advantageous to remember the places visited before and make decisions based on that information instead of searching randomly without considering previous locations.