IE 506: Machine Learning - Principles and Techniques **Jan-May 2024**

Project End-term Report : Statistical Change Detection for Multi-Dimensional Data

Team Name: Machine Learners *Team Members: 23M1508, 23M1512*

Abstract

Our project is based on research paper title "Statistical Change Detection for Multi-Dimensional Data." [1]. This report contain all the details of our project. It is important to know that whether the newly observed data is different from the previous data or not, so it is important to find the distributional change in multi dimensional data. So we will discuss the methodology given in this research paper to find distributional change. We have done code from scratch to work on different data sets which is give in our research paper. We also worked on image dataset (mnist) to examine algorithms.

1 Introduction

This paper considers the problem of building a statistical test for detecting the change of distribution in multidimensional data. Such a test would have numerous applications in a variety of disciplines such as commerce, healthcare, and finance. By monitoring these information about latest data observed in the most recent day we can ask question: Is the distribution of recently observed data different from what has been observed before? So to answer this question we need a distributional change detection method to answer this question.

Motivation Example: One of the examples is to detect the change in financial market pattern. So it is useful to understand the trends. Features Like: Stock prices, trading volumes, Earnings Per Share (EPS), GDP, Interest Rates, Inflation Rates. We have a baseline data set and a recently observed data set. Does it shows a different pattern than what we have observed before? If a change is detected, it might be caused by the potential fraud, or it can identify the upcoming financial crisis.

2 Literature Survey

In this research paper, The detecting changes in distributions of multi-dimensional data sets. The given approach in research paper is start by assuming two sets of independent and identically distributed (i.i.d.) samples, denoted as S and S', drawn from two unknown multi-dimensional distributions F_S and $F_{S'}$ respectively.

Now, Here is briefly walk through the entire test process. First, split the dataset S into two parts: S1 and S2. Then, use the Learn bandwidth algorithm 1 to learn the kernel density estimator model over S1, which helps to find the parameter for the baseline distribution of the data. Next step is to calculate a value called δ for a given pair of S2 and S'. This value δ essentially tells how different S' is from the baseline S. Then, the next step is to determine a critical value called C_{max} using Algorithm 3. This critical value helps to decide whether the observed change in S' is significant or not.

Prior work: For uni-dimensional data, many existed tests, such as K-S test, chi-square test and many more. However, for multi-dimensional data there is little attention. While there exists outlier detection methods

exist for multi-dimensional data, they don't specifically detect distributional mismatches among data sets. Notable statistical tests proposed for this problem include the cross-match test, introduced by Rosenbaum, Paul R.[2], The Cross-Match Test is a statistical method used to compare two multivariate distributions. Another is Kulldorff's test, proposed by Dasu, Tamraparni et al.[3]. This method is based on the concept of relative entropy, also known as the Kullback-Leibler distance, which measures the difference between two nonparametric distributions.

3 Methods and Approaches

In order to compute δ, the density test leverages a kernel density estimator (KDE), a common non-parametric method for learning data distributions. A KDE approximates the data distribution by overlaying many individual kernel functions at points sampled from the distribution being modeled. Each sample contributes density to nearby areas of the data space, smoothing out the distribution. Formally, given data points $x_1, \ldots, x_{|S_1|}$ in dataset S_1, the estimated density at any point s is calculated as:

$$\mathscr{K}_{S_1}(s) = \sum_{x_i \in S_1} \frac{1}{|S_1|} G\left(\Sigma_{x_i}, s - x_i\right) \ldots\ldots\ldots \quad (1)$$

Here, G represents the kernel function, with one kernel centered at each data point in S_1. Σ_{x_i} denotes the bandwidth or spread of the kernel function centered at x_i, typically a positive-definite co-variance matrix for a k-dimensional Gaussian kernel.

Choosing the appropriate bandwidth for each kernel is crucial for the accuracy of the KDE. Unlike common practice, where all kernels share the same bandwidth, we recognize the need for adaptive bandwidths to reflect the varying characteristics of different portions of the data space. To address this, we adopt a variation of maximum likelihood estimation to determine the kernel bandwidths. Instead of directly maximizing the likelihood, we maximize the "pseudo log-likelihood" over all possible choices of Σ_{x_i}, defined as:

$$L = \sum_{x_j \in S_1} \log \left[\sum_{x_i \in S_1 \text{ and } i \neq j} \frac{1}{|S_1| - 1} G\left(\Sigma_{x_i}, x_j - x_i\right) \right] \ldots\ldots\ldots \quad (2)$$

This approach allows us to tailor the kernel bandwidths to the specific characteristics of the data, ensuring a more accurate estimation of the underlying distribution.

First Step : We have the baseline S is randomly partitioned into two halves $S = S_1 \cup S_2$ from our full data. S_1 will be used for modeling and S_2 for testing. We done partitioning of S such that they not contain same index of data. Then we apply equations given below to implement the algorithm 1.

From Equation 3, we calculate the Covariance matrix of i^{th} data point. Let $\omega_1, \ldots, \omega_{|S_1|}$ be the Gaussian kernels centered at $x_1, \ldots, x_{|S_1|}$ respectively. The update rule for the bandwidth of the kernel associated with data point i is:

$$\Sigma_{x_i} = \frac{\sum_{x_j \in S_1} P\left(\omega_i \mid x_j\right) \left(x_j - x_i\right) \left(x_j - x_i\right)^T}{\sum_{x_j \in S_1} P\left(\omega_i \mid x_j\right)} \ldots\ldots\ldots \quad (3)$$

From Equation 4, $P(\omega_i \mid x_j)$ is the probability that x_j is generated by the kernel associated with x_i. This probability is often referred to as the "soft membership" of point x_j in Gaussian component ω_i.

$$\begin{cases} P\left(\omega_i \mid x_j\right) = \frac{p(x_j|\omega_i)}{\sum_{t=1}^{|S_1|} p(x_j|\omega_t)} & i,j = 1,2,\ldots|S_1|, i \neq j \\ P\left(\omega_i \mid x_i\right) = 0 & i = 1,2,\ldots|S_1| \end{cases} \quad \ldots\ldots\ldots (4)$$

From Equation 5 $p(x_j \mid \omega_i)$ is the density of the i^{th} Gaussian kernel at the point x_j. The second line in the equation is given because of the constraints of pseudo log-likelihood objective function.

$$\begin{cases} p\left(x_j \mid \omega_i\right) = \frac{1}{(2\pi)^{k/2}\left|\Sigma_{x_i}\right|^{1/2}} \exp\left(-\frac{1}{2}\left(x_j - x_i\right)^T \Sigma_{x_i}^{-1}\left(x_j - x_i\right)\right) \\ p\left(x_i \mid \omega_i\right) = 0, \quad i,j = 1,2,\ldots|S_1|, i \neq j \quad i = 1,2,\ldots|S_1| \end{cases} \quad \ldots\ldots\ldots (5)$$

Algorithm 1 Learn Bandwidth ($S1$, MaxIteration, Φ)

```
t = 0
while t < MaxIteration do
    Compute density p(x_j|w_i) for all i, j by Equation (5)
    Compute soft membership P(w_i|x_j) for all i, j by Equation (4)
    Compute bandwidth Σ_{x_i} for all i by Equation (3)
    Compute the objective function L_t by Equation (2)
    if (L_t − L_{t−1}) / L_{t−1} < Φ then
        break
    t ← t + 1
```

The above algorithm is Learn bandwidth algorithm which based on the expectation maximization. Since we have to find the distribution of our data for that we have to find parameter which define distribution of our data. Since we find this distribution using kernel density estimate and for this bandwidth is most important parameter for defining the non parametric distribution therefore using this given algorithm 1 on S1, we got data-driven bandwidth for our baseline distribution. Which is the optimum bandwidth.

Second Step : Next, we compute the variance of S2 using the test statistic δ. δ, which is defined as the difference of log probability density of $\mathscr{K}_{S_1}$ at S' and at S_2.

$$\delta = LLH\left(\mathscr{K}_{S_1}, S'\right) - \frac{|S'|}{|S_2|} \times LLH\left(\mathscr{K}_{S_1}, S_2\right)$$

This test statistic have three properties:

1. δ calculated under the null hypothesis is a sample from Δ, which has a normal distribution.
2. The expected value of the mean of Δ is always zero.
3. The variance of Δ can be estimated from the algorithm 2.

The random variable used to produce δ can then be defined below:

$$\Delta = \sum_{i=1}^{|S'|} \log \sum_{x \in S_1} \frac{1}{|S_1|} G\left(\Sigma_x, T_i - x\right) - \sum_{i=|S'|+1}^{|S'|+|S_2|} \frac{|S'|}{|S_2|} \times \log \left\{ \sum_{x \in S_1} \frac{1}{|S_1|} G\left(\Sigma_x, T_i - x\right) \right\}$$

Where,

$$f(T_i) = \log \sum_{x \in S_1} \frac{1}{|S_1|} G(\Sigma_x, T_i - x)$$

$$g(T_i) = \frac{|S'|}{|S_2|} \times f(T_i)$$

Due to the Central Limit Theorem, The Δ has a normal limiting distribution. The mean and variance of Δ is given below:

$$E[\Delta] = 0,\ \mathrm{Var}[\Delta] = \left(|S'| + \frac{|S'|^2}{|S_2|} \right) \sigma^2$$

Where,

$$\sigma^2 = Var[f(T_i]$$

Algorithm 2 EstVar(V, $S2$, estSize, β)

1: Initialize the array Est
2: **for** $t = 1$ **to** estSize **do**
3: Bootstrap resample R from $S2$, where $|R| = |S2|$
4: $Est[t] \leftarrow \frac{|S2|}{|S2|-1} \mathrm{Var}[f(R)]$
end for
5: $V \leftarrow [\text{estSize} \times (1-\beta)]^{th}$ percentile of Est
6: $Var[\Delta] \leftarrow (|S'| + \frac{|S'|^2}{|S2|})V$

Distribution of the population from which δ is coming is to be found. δ is coming from Δ population. distribution of Δ is normal with mean zero and some variance. This variance (Var[Δ]) is found using Algorithm 2.

Third Step : After calculating the variance of S2, We perform a standard null hypothesis test. Depending on the observed δ is significantly far out in the tail of Δ, we either reject the null hypothesis and declare change, or we declare no change since we do not have strong enough evidence to reject the null hypothesis.

After getting the variance from the algorithm 2, we have distribution of Δ using this distribution, we find critical value (C_{max}) of δ using Algorithm 3. Finally, we compare the calculated δ with this critical value C_{max}. We can make Decision whether there's no significant change or significant change in the distribution.

Algorithm 3 CriticalValue(p, $stepSize$)

1: $M = \frac{p}{stepSize} - 1$, M is the number of (α, β) pairs
2: $\alpha_i = i \times stepSize$, $i = 1, 2, ..., M$
3: $\beta_i = p - \alpha_i$, $i = 1, 2, ..., M$
4: Let C be an array of critical values
5: **for** i=1 **to** M **do**
6: Run EstVar(V, $S2$, $estSize$, β_i) to obtain $Var[\Delta]$
7: Find c such that $P(\Delta \leq c) = \alpha_i$, $C[i] = c$.
8: $C_{\max}$ largest value in array C.

3.1 Work Done

Following is the work done under this project:

1. **Dataset Pre-processing:**
   We have worked on the EL Nino (5 Dimensional) Dataset. Initially this had dataset 1,78,000 rows and 12 columns. We removed those data records with missing values and after cleaning we have 93,935 records. We also removed the Spatio - Temporal attributes from the dataset since the changes on these attributes are not meaningful as this was mentioned in our research paper. Finally we got dataset with 93935 rows and 5 columns. From these 93935 data points, required number of data points are randomly sampled and named as dataset S. Sample dataset S is then split into S1 and S2 datasets in which S_1 is used for training and S_2 is used for testing purpose. Inference dataset is S' on which we have to detect the change.
2. **Learning Bandwidth:**
   First, distribution of the original dataset S is found with the help of training dataset S_1. For this, kernel density estimation (kde) is used. Separate gaussian kernel is considered at each of the data points with the assumption that no data point is generated by the kernel centered at itself. All these bandwidths are learned by maximizing the value of pseudo log likelihood that each of the data point in training dataset is coming from distribution of the training dataset. This is done using algorithm 1. Thus we got bandwidth of kernel at each of the data points. Using this, we got density function of the entire distribution of S_1 which represents the distribution of S.
3. **Finding Test Statistic (δ):**
   Test statistic is found using:
   Test statistic (δ) = (log likelihood that S_2 is coming from distribution of S_1) - (log likelihood that S' is coming from distribution of S_1). This is found using equation of δ mentioned earlier. This test statistic is then compared with our estimated threshold value (critical value) to finally detect the change.
4. **Finding Critical Value (C_{max}):** For this, first distribution of the population from which δ is coming is found. δ is coming from Δ population. distribution of Δ is normal with mean zero and some variance. This variance is found using Algorithm 2. After getting this variance, we have distribution of Δ. Using this distribution, critical value (C_{max}) of δ is calculated in such a way that it follows some desired confidence interval. This is done using Algorithm 3.
5. **Inference:** If $\delta < C_{max}$, we declare change in distribution, means distribution of S and S' is not same. Otherwise there is no change in distribution, means distribution of S and S' is same.
   Two way test is used for inference. First change from S to S' is detected, then in second way, change from S' to S is detected (roles of S and S' are swapped in this) and inference decision is made based on results of both the ways. If change is detected in the first way, then we declare change. If no change is detected in the first way, then it is again checked using second way, in the second way if the change is detected then we declare change, otherwise if change is not detected even in the second way then we finally declare that there is no change in the distribution.

6. **Experiments Performed:** Two experiments are performed in order to calculate False positive rate and False negative rate. details of the same are mentioned in 'Experiments' section.

4 Data set Details

Dataset used for our work is 'El-Nino (5-d)' dataset. This is a numerical dataset in '.csv' format. It contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. Initially this dataset had 1,78,000 rows and 12 columns. We removed those data records with missing values and after cleaning we have 93,935 records. We also removed the Spatio -Temporal attributes from the dataset since the changes on these attributes are not meaningful as this is given in our research paper. After removing these spatio - temporal attributes we got 5 dimensional dataset. This dataset is used as a matrix in the research paper and matrix operations are done on it after converting it to a numpy matrix. Source [4] of the dataset is provided in the 'References' section.

5 Experiments

Hardware configuration used for all the experiments:
Intel(R) Xeon(R) CPU E5506, 2.13 GHz, Number of cores = 32, Ram = 128GB

Experiment one: False Positive Rate
This experiment aims to calculate the false positive rate. means, the percentage of times the code detects that there is change in distribution when there is actually no change. Following is the methodology followed for this experiment.

1. **Data Sampling and Instance Creation:** We have sampled N=224 data points with replacement from the data source, constituting an instance. This process was repeated 10 times to generate 10 instances. Each instance is sampled with replacement, ensuring identical distribution across data points. Any split of an instance into two subsets results in subsets with the same generative distribution.
2. **False Positive Rate Estimation:** We have then split each instance evenly. This will give us S and S' datasets each of size 112. Two way taste is performed on this S and S' in each instance. first S is split evenly into S_1 and S_2 using random sampling, each will be of size 56. Algorithm was trained on S_1 and tested using S_2 and S'. For the second way of test, role of S and S' was reversed. Thus, two way change detection test was performed in each instance and number of times the test indicates the change is noted. This number is nothing but number of false positives. since, we already know that S and S' have been sampled from the same dataset and hence there is no change in distribution. False positive rate is estimated by calculating the fraction of times the test refutes the null hypothesis across the 10 instances.
3. **Test Parameters:** estSize = 200 in Algorithm 2. For Algorithm 3: p = 0.04 (due to the two-directional test) and stepSize = 0.002.

Experiment two: False Negative Rate
This experiment aims to calculate the false negative rate (Power). means, the percentage of times the code detects that there is no change in the distribution when change is actually there. Following is the methodology followed for this experiment.

1. **Instance Creation:** Any distribution when undergoes change, it doesn't happen abruptly, it happens gradually. In order to generate most realistic changed distribution (S'), first S' is sampled from original dataset, in that some fraction of noise is added. We have considered two types of noises. those are full

dimensional noise and 1 dimensional noise. In full dimensional noise, we have taken Gaussian noise. it is done by adding Gaussian noises to all the dimensions of some of the sample points (20 % as per research paper) of the selected S'. In 1-d noise we have added noise using 2 methods which are 'add 1-d' and 'scale 1d'. In former, we select any dimension randomly and add standard Gaussian noise in that entire column. In later, we select any dimension randomly and multiply each value of that column by 2. By adding these noises we get three different noisy S' one for full dimensional Gaussian noise, one for 'add 1-d' and one for 'scale 1-d'. The experiment is then run for each of these three noisy S'. each experiment is run for 10 instances. In each of these instance, rows or columns in which noise is to be added are chosen randomly.

2. **False Negative Rate Estimation:** Only testing phase is involved in this experiment since only one way test is done in this experiment. Experiment is run separately for each of the three noise types. In each experiment, 10 instances are there. out of these 10 instances, we count how many times the code detects no change. This will give us number of false negatives since we already know that there is change in the distribution because of the noise addition. Fraction of times the code detects the false negatives out of 10 will give us false negative rate.

6 Results

Inference results:
The code is successfully detecting both gradual as well as abrupt change. Inference decisions are made using two way test.
Experiment 1 results:
False positives according to research paper: 3 in 100 instances.
False positives according to our execution: 0 in 20 instances.
Experiment 2 results:
False negatives for full dimensional Gaussian noise according to research paper: 0 in 100 instances.
False negatives for full dimensional Gaussian noise according to our execution: 0 in 10 instances.
False negatives for scale 1-d noise according to research paper: 2 in 100 instances.
False negatives for scale 1-d noise according to our execution: 0 in 10 instances.

7 Work Done After Midterm review

Working on higher dimensional datasets:
1. Body-fat dataset (15 dimensional)
2. MNIST dataset
3. Boston dataset (16 dimensional)

7.1 Bodyfat dataset:

This is a 15 dimensional dataset with 250 sample points. Authors don't suggest to work on this dataset directly since it has very less number of sample points, instead they've suggested method to increase the number of sample points to 20,000:
a) A sample point is randomly selected from the dataset.
b) Its five nearest neighbour sample points are selected with replacement.

c) Selected point along with its nearest neighbours are averaged to create a new sample point.
d) This process is repeated 20,000 times to create a dataset of 20,000 samples and 15 dimensions.

This newly created dataset is used to train the model. A small dataset S is selected at random from the entire dataset, S is then split into S1 and S2. The inference dataset S' is created according to requirement. If S' is extracted from the original dataset then the algorithm is successfully commenting that there is no change, if we create S' by adding some noise in the dataset extracted from the original dataset, or if we create S' as completely different dataset following some other distribution then the algorithm is successfully detecting the change.

Experiments conducted on Bodyfat dataset:
Two experiments were conducted on the bodyfat dataset.
Experiment 1: This is conducted to calculate the false positive rate of the algorithm. It counts the number of times the algorithm claims that there is change when in reality there is no change.
Experiment 2: This is conducted to calculate the false negative rate of the algorithm. It counts the number of times the algorithm claims that there is no change when in reality there is change.

7.2 MNIST dataset:

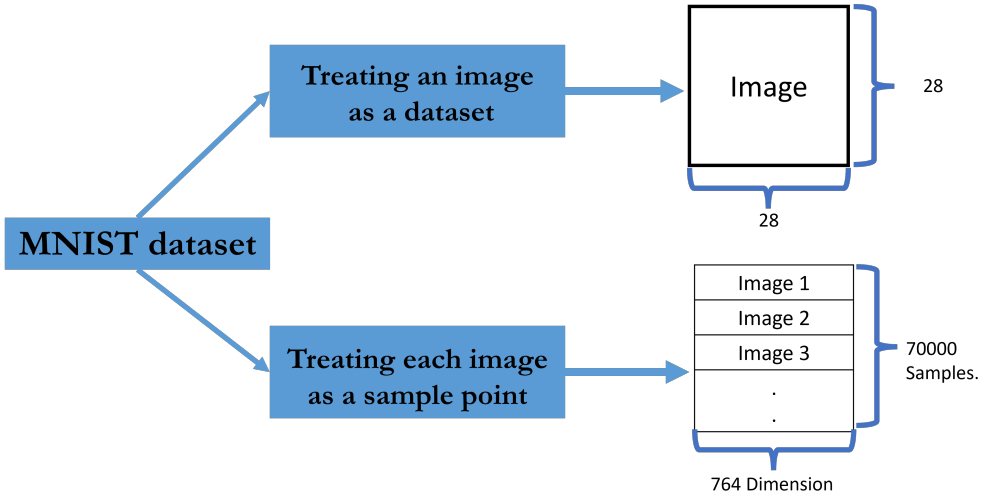


Figure 1: Experiments with two type of dataset formed from image dataset

We have tried two things in this dataset:
a) Treating a single image as a dataset.
b) Treating entire MNIST dataset as our dataset

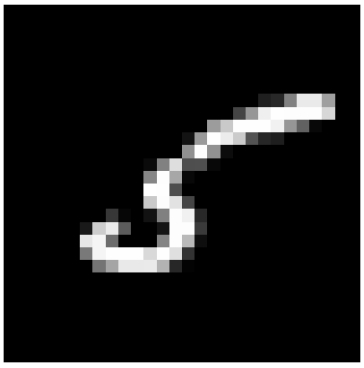
a) Treating a single image as a dataset:
Each image of MNIST dataset is a matrix with 28 rows and 28 columns. From the entire MNIST dataset, after picking one image randomly, we used this image as our dataset and detected the change by adding some noise in the image. Each row of the selected image was treated as a sample point and each column as a dimension. Hence our dataset is a 28 by 28 matrix. For training we didn't split the image into S1 and S2, instead we took entire image matrix as S1 only and again for S2 we took the same image matrix, hence in

image case we considered S1=S2 since splitting the image won't be a good option. For change detection, we have done two things:

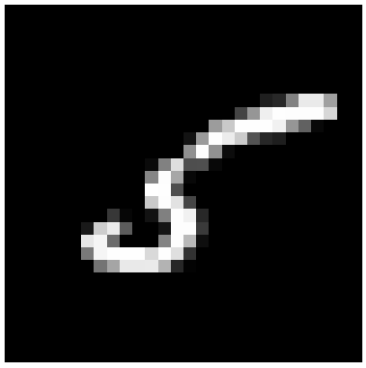
1) Comparing an image with another image from the MNIST dataset.

When we run our algorithm for the given two images below i.e image(a) and image(b) our algorithm also predicted that these two images have same distribution. As we can see that these two given images are same given below.

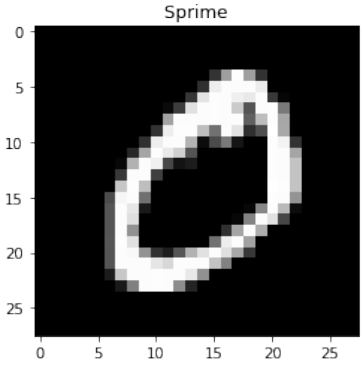
Now When we run algorithm for the two images below i.e image(c) and image(d) our algorithm also predicted that these two images have different distribution. As we can see that these two given images have actually different.



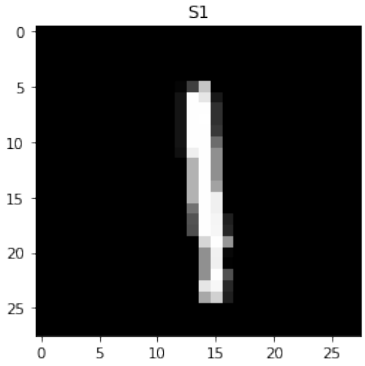
(a) Training image (s1 matrix)



(b) Testing image (Sprime matrix)



(c) Zero digit image

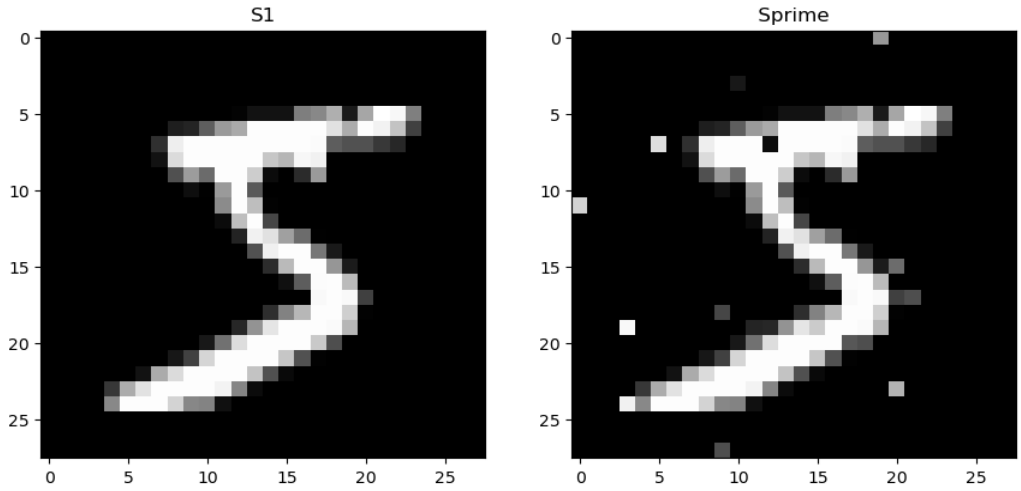


(d) One digit image

Table 1: Images for comparison

2) Comparing an image with the image created by adding some noise to the same image.

For the purpose of change detection, we randomly select one image from the MNIST dataset. This image becomes our dataset, and we introduce changes by adding noise to it as you can see the images below left side images is the original image and right side image is the noise added image. Since each row of the selected image matrix represents a sample point and each column represents a dimension, the entire 28x28 image matrix serves as our dataset (denoted as S1).



(a) Original Image (b) Original image with noise

So we see that algorithm's is correctly detect the similar and dissimilar images based on their distribution, as demonstrated by its predictions for the pairs of images.

b) Treating entire MNIST dataset as our dataset.

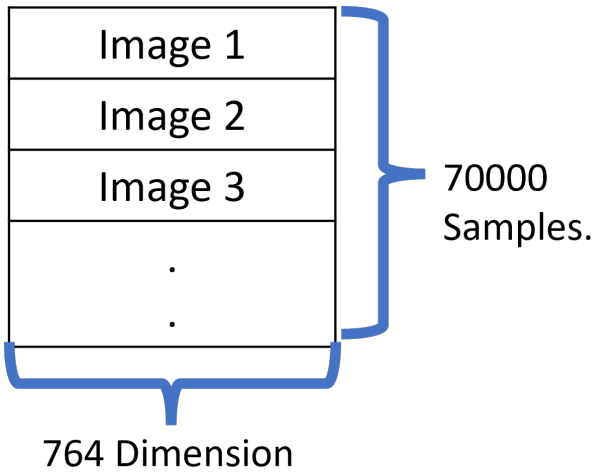


Figure 2: Each image treated as data point

As we can see in the above diagram, Entire MNIST dataset has 70,000 images. We have treated each image as a sample point by flattening it's image matrix. After flattening the matrix of one image, it becomes a sample point with 28*28= 784 dimensions. Hence, our dataset becomes 70,000 by 784 matrix. We tried to detect the change in distribution for this dataset.

Challenges faced:
High dimensionality of the dataset: Maximum dimensional dataset considered by authors in the paper is of 26 dimensions. This algorithm becomes extremely computationally expensive for high dimensions like 784. It becomes extremely slow when run directly.
Things which we tried to make it run:

1. Converting the matrix into binary matrix: Computations on binary matrix can be less expensive that's why we tried this. We decided some greyscale value as a threshold value, values below which were converted to 0 and values above it were converted to 1. Converting the matrix into binary matrix won't change the distribution of the dataset because each unique type of the image will be a unique sample point which will be having all the feature values binary. This binary sample point will have the same number of appearances in the dataset as the actual image had. Hence, converting into binary won't change the distribution of the dataset. Even this thing didn't work, still we couldn't get outputs as the algorithm was still very slow.
2. Removing those columns from the binary matrix which had all zero values: 98 such columns were found after removing those we were left with 686 columns. Algorithm was still very slow.
3. Applying PCA: Knee graph suggested to perform PCA for 49 attributes, It was done. Still algorithm wasn't fast to the acceptable level. Then we performed PCA to make the dataset of 20 attributes. This time it gave the outputs but they were not in the acceptable range, applicability of PCA in this case is the topic of further research.

   From this we acknowledged that the method discussed in the paper is highly computationally expensive for very high dimensional datasets.

7.3 Boston dataset:

This dataset has 506 samples and 20 attributes, authors have suggested to remove some attributes and make it 16 dimensional dataset. It was done as per that. The dataset was then made of 20,000 rows by increasing the number of samples as per the method suggested by authors which was explained in the bodyfat dataset above. Final dataset which we got was 20,000 by 16 matrix. Sampling was done from this to get S. S1 and S2 were made from S. S' was created according to requirement. We conducted the change detection test and we got the expected results.

7.4 Experiments after mid term

We have done both the experiments on Bodyfat dataset. Results are matching with the expected results. Inference tests were done on boston dataset and on mnist dataset, detection results are as expected.

7.5 Results

Inference results:
The code is successfully detecting change when S and S' are same and as well as for when S and S' are not

same. Inference decisions are made using two way test.
It is successfully detecting the change or no change for images in mnist dataset also.
Experiment 1 results:
For bodyfat dataset: False positives according to research paper: 3 in 100 instances.
False positives according to our execution: 0 in 10 instances.

For Boston dataset dataset: False positives according to research paper: 3 in 100 instances.
False positives according to our execution: 0 in 10 instances.

Experiment 2 results:
For Bodyfat dataset dataset: False negatives for full dimensional Gaussian noise according to research paper: 0 in 100 instances.
False negatives for full dimensional Gaussian noise according to our execution: 0 in 10 instances.

False negatives for randomly selected dimensional noise according to research paper: 0 in 100 instances.
False negatives for randomly selected dimensional noise according to our execution: 0 in 10 instances.

False negatives for scale 1-d noise according to research paper: 2 in 100 instances.
False negatives for scale 1-d noise according to our execution: 0 in 10 instances.

8 Possible Future Work

Since for every dataset its covariance matrix is behaving differently so it is important to handle this matrix so to finding the optimal way to handle the covariance matrix is possibly a future work. And finding a way to reduce the time complexity of these algorithm is also good for future work. ANd Proper analysis of whether PCA can be used or not for dimensionality reduction.

9 Conclusion

In this project we implemented the earlier mentioned research paper which focuses on detecting the change in distribution of the multi dimensional data. Sample dataset S is split into S1 and S2 datasets in which S1 is used for training and S2 is used for testing purpose. Inference dataset is S' on which we have to detect the change. Solution approach started with estimating the training data distribution using KDE in which they have considered kernels centered at each of the data points with the crucial assumption that no data point is generated by the kernel centered at itself. All the bandwidths are learned by maximization of pseudo log likelihood that each of the data point in training dataset is coming from distribution of the training dataset. Test statistics was found by: Test statistic = (log likelihood that S2 is coming from distribution of S1) - (log likelihood that S' is coming from distribution of S1). If this test statistic comes out to be more than our calculated threshold value then the change was declared otherwise no change was declared. Results of the paper are as per expectation. The code is successfully able to detect both gradual as well as abrupt change. Two way test verifies the decision made as well. False positive rate and False negative rate are calculated in the Experiment 1 and those are within the expected range, these numbers are better for density test compared to pre-existing kdq tree test and cross-match test. Experiment two tests the ability of the algorithms to detect the gradual change in the data and it is well satisfied by the implemented code.

Similar work on the higher dimensional data is intended in the future.

Our study encompassed a series of experiments aimed at assessing the efficacy of the density test. Upon analysis of the experimental outcomes, it was observed that both the false positive rate and false negative rate fell within acceptable thresholds, as mention in the research paper. However, it is important to acknowledge that while the density test demonstrated satisfactory performance in terms of accuracy, it exhibited limitations in terms of computational efficiency, particularly evident in datasets with dimensions exceeding 30. This underscores the need for further exploration of alternative methodologies to address the computational challenges associated with higher-dimensional datasets.

10 References

[1] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. 2007. Statistical change detection for multi-dimensional data. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). Association for Computing Machinery, New York, NY, USA, 667–676. https://doi.org/10.1145/1281192.1281264, Accessed 1st Feb. 2024.

[2] Rosenbaum, Paul R. "An Exact Distribution-Free Test Comparing Two Multivariate Distributions Based on Adjacency." Journal of the Royal Statistical Society. Series B (Statistical Methodology), vol. 67, no. 4, 2005, pp. 515–30. JSTOR, http://www.jstor.org/stable/3647642, Accessed 30th Mar. 2024.

[3] Dasu, Tamraparni et al. "An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams." (2006). Accessed 30th Mar. 2024.

[4] El Nino Dataset. Kaggle, https://www.kaggle.com/datasets/uciml/el-nino-dataset, Accessed on: 5th Feb 2024.