Change Point Detection Supports Improvement of Threshold Determination

James P. Clark
B.S. Computer Science, American University
4400 Massachusetts Avenue NW, Washington D.C.

Abstract: Threshold Determination is an ever occurring problem in machine learning and suffers from obtaining an optimal solution for various data streams. Most existing determinations are based on a static variable that involves the use of ROC curves and more generally heuristical decisions. This paper provides a solution to improving threshold determination by fusing the ideas of change point detection and previous known threshold solutions to generate a variable threshold. The idea will provide a solid foundation in differentiating a concept drift from an alarm with a low false alarm rate. Comparisons will be made from known threshold determination methods to our proposed change point detection method.

1. Introduction:

The world that we live in today is not static and the volume of data that is absorbed every moment is ever growing. To face the challenges of today efficient and effective tools must be developed to account for changes over time. Typically in today's machine learning environment the data presented is based on historic values of a given time frame.

Although majority of data collected today comes in streams continuously as time passes. According to (Webb et al) the world is dynamic , and the complex distributions that a model models are likely to be non-stationary. In a dynamically changing environment the distribution continually changes leading to what is known as a concept drift. Explained by (Gama et al) a concept drift refers to changes in the conditional distribution of the output (i.e., target variable) given the input (input features), while the distribution of the input may stay unchanged. A simple example of what a concept drift may look like could be normal functionality of a workplace. During the normal working hours there may be lots of work going on that is normal daily routine. At night things change and not much work gets done but it is still considered normal activity.

To deal with managing changes to activity one must provide a solution of how to detect a change has indeed occurred. If a change has occurred it must be determined if it is truly a concept drift or if it is just an outlier disturbing the system. To be able to find when changes to the system happen there is a very powerful statistical tool known as change point detection which can mathematically find when changes to a distribution have been altered. By making use of this tool we can detect whether a change in distribution could be related to an abnormal instance or indeed a concept drift.

This paper provides a novel and simple solution that combines ideas from change point detection and fuses it into a machine learning environment. What this paper proposes is a solution to detecting when changes occur to the structure of data. By being able to detect when changes happen with absolute precision we will be able to differentiate an outlier from a concept drift, with a low false alarm rate. This can allow us to create a varying window size based upon

the use of our change point detection scheme to generate new threshold values. Informed adaptation, which will be explained in section 2.3, could potentially be of greater use and can make handling concept drifts more feasible than previous uses.

2. Related Works:

In this section we cover what exactly a concept drift is and multiple different techniques used today to account for them. A vast majority of ways to handle this tends to be sliding window based algorithms as discussed by (Gama et al). We will give a brief discussion on defining a concept drift and then popular mechanisms that work with them.

2.1 Adaptive Learning

In an environment where the dynamics are constantly changing adaptive learning algorithms provide a simple approach to handling these changes. These particular algorithms can be seen simply as incremental learning. That is as new data is generated over time it is constantly fed into the system and enabling it to continuously learn. These solutions provide examples of how learners can be capable of handling concept drifts.

2.2 Concept Drift

A concept drift, as previously discussed, is when the conditional distribution of the target value changes while the distribution of the input values stay unchanged. To put this into a learning situation we can note P(Y) as the probability distribution over class labels, and P(X) to be the prior probability distribution over covariates. Together we can say that P(Y|X) is the joint probability distribution over objects and class labels.

According to (Webb et al) a definition of a concept must be defined before characterizing a concept drift. Recently a concept has been given a probabilistic definition, defined by (Gama et al), as the prior class probabilities P(Y) and class conditional probabilities P(X | Y). As P(Y) and P(X |Y) determine the joint distribution P(X,Y) and the other way around for the alternative, this is the same as defining a concept as the joint distribution P(X,Y). To put this all together we can add a subscript noting time j as $P_j(X)$ which states the probability at time j. We can say that a concept drift occurs between times j and m if

$$P_j(X) \neq P_m(X).$$

Now that a concept drift has been defined Figure 1 will point out several different types of concept drifts. The first type of drift is a sudden/abrupt change could have the possibility of a sensor being replaced that has a different calibration so although the numbers

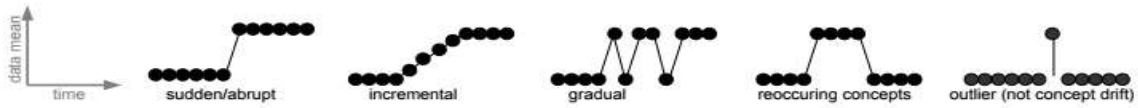


Figure 1

are different the instance is still defined as normal. Incremental is a smooth change to the system which could be seen as the rise in global temperatures over centuries. Most drifts take on these two characteristics in some form but the challenge is to not mix a concept drift with an outlier or a random deviation.

2.3 Adaptive Learning Strategies

To adapt to concept drifts there are several different techniques in place described by (Gama et al). There are two main concepts for deploying these strategies, memory and forgetting. Memory concepts can involve single example or better known as online learning algorithms. These types involve processing one example at a time and updating the system based on that example. It will take the input (x) and make a prediction (y), once the true value of y has been received later it will be able to compute the loss and make an update on the system. This scheme allows it to adapt to concept drifts because it is continually updating the system with each instance, although it tends to be slow at adapting to abrupt changes in the system. Multiple examples bases itself on maintaining a model that uses a given set of recent examples. This requires the use of a sliding window that uses a queue data structure or better known as first-in-first-out structure. What this means is that the algorithm will build an entire new model after every example given. This allows for a system to forget older examples by continuously updating the system on more recent ones.

Sliding windows are based off of multiple examples and can be created in two different ways either a fixed size or a variable size. The size of the window will play a crucial role in how the system will adapt to changes, for example, a large sliding window will perform very well in a time of stability but adapts very slowly to concept drifts. The opposite is true for smaller sizes where it will adapt quickly to a concept drift but performs poorly during a stable period. The problem comes down to how do we account for this and vary the size of the sliding window.

There are two different approaches to window sizes namely blind adaptation and informed adaptation. Blind adaptation doesn't use any sort of detection methods that are searching for a concept drift. This technique typically uses a fixed window size and will periodically retrain the system with the most recent examples. Informed adaptation does the opposite where it looks for a trigger to tell the system something is happening. When a trigger occurs the system can vary the window size if it is able to tell a concept drift is happening.

3. Methodology

The section will provide details on what tools and functions we will need to further explain our algorithm. We will first introduce the basic understanding of how different learning

algorithms work including Autoencoders, One Class Support Vector Machines, and K Nearest Neighbor. After we will explain different types of change point detection schemes and how those can enable us to put them into practice. Lastly there will be a discussion on receiver operating characteristic curves which will wrap all the ideas together.

3.1 Autoencoders

An autoencoder simply put is a feed forward neural network that contains an input layer, an output layer, and one or more hidden layers. The difference between autoencoders and more popular neural networks is that the input size and output size are exactly the same. The fundamental idea of autoencoders is that they are able to simply reconstruct the input given as output.

The connections made between layers of the neural networks go by certain distinct names. Encoder is the connection that is made from input layer to hidden layer, and Decoder is the connection made from hidden layer to output layer. An autoencoder can be summed up similarly as a composition of two maps, which we will note as ϕ and Ψ, where the goal is the minimize the reconstruction error of inputs X:

$$\phi: X \rightarrow F$$
$$\Psi: F \rightarrow X$$
$$\arg \min(\varphi,\psi) \; ||X - (\psi \circ \varphi)X||^2$$

In this specific study the autoencoder will consist of only a single hidden layer. An autoencoder with one hidden layer takes in a single input vector $x \in R^d$ and maps it to $h \in R^P$ (where p is the number of hidden units within the layer) with a nonlinear activation function noted as $\boldsymbol{\Omega}$:

$$h = \boldsymbol{\Omega}(w_1 x + b_1)$$

This process is the encoding process stated previously.

Then the decoding process must be accomplished which maps h to $\bar{\mathbf{x}}$ which is the reconstruction of x produced by the autoencoder:

$$\bar{\mathbf{x}} = (w_2 h + b_2)$$

The reconstruction error can be calculated in a variety of different ways but what is most commonly used what will be used throughout this paper in the mean square error. Autoencoders are trained to minimize this error, which simply means that over time they are trying to output the given input as closely as possible. To define this procedure goes as follows :

$$L(x,w) = L(x, x') = ||x - x'||^2 = ||x - \sigma_2(w'(\sigma 1(wx + b)) + b')||^2.$$

From this equation we can be compute the Empirical risk as:

$$En(w) = 1/n * \sum L(xi,w).$$

The way that an autoencoder can go about minimizing the average error is through weight adjustment. Different error propagation mechanisms and gradient descent optimizers can be applied to accomplish this task. Two different types of gradient descent optimization procedures will be used with back propagation throughout this paper, notably mini-batch gradient descent along with stochastic gradient descent.

1. Mini-batch gradient descent: parameters are updated for each batch of n training examples. The gradient is computed as the average losses over the data points.

$$w(k) = w(k-1) - \eta \partial E_n / \partial w \, (w(k-1))$$

$$= w(k-1) - \eta * 1/n \sum \partial L / \partial w(xi, w(k-1)).$$

2. Stochastic gradient descent: parameters are updated for each a single training example passed through the network:

$$w(t) = w(t-1) - \eta(\partial L / \partial w)(x_t, w(t-1)$$

Where w is the weights and biases, η is the learning rate, L is the loss function of the given training example and E is the empirical risk of n training examples.

3.2 One Class Support Vector Machines

Next on our list of learners is One Class Support Vector Machines (OCSVM) which is a learning tool that can map input data into a high dimensional feature space, given a kernel function. It goes about this by iteratively defining a hyperplane that separates the training data from the origin which at the same time maximizes the distance of the hyperplane from the origin. If familiar with SVM's an OCSVM can be thought of simply as a two-class SVM problem where only one class is used as training data. According to (Heller) the OCSVM solves the following quadratic equation:

$$\min(\tfrac{1}{2} * \sum a_i a_j K(x_i, x_j))$$

with constraints

$$0 \leq a_i \leq 1/vl$$

and

$$\sum a_i = 1$$

where a_i is a lagrange multiplier, v is a parameter that controls trade-off between maximizing the distance of the hyperplane from the origin and the number of data points contained by the hyperplane, I is the number of points in the training data set, and $K(x_i, x_j)$ is the kernel function. A kernel function is used to provide non-linearity when creating the hyperplane, it can take on a variety of forms from sigmoid to rbf.

3.3 K Nearest Neighbor

Compared to OCSVM and Autoencoder the Nearest Neighbor algorithm is by far the simplest to understand. For each training example in the training set we map each example into a space of R^d where d is the number of features. In the testing phase for each testing example we can compute the distance that point is from each of the training points by different distance measurements. With each distance we take k number of distances which are the smallest. Therefore one can conclude in a binary classification setting if the distance tends to be far from a training example then that instance belongs to the positive class and the opposite is true if it is small we can consider that to be negative.

3.4 Thresholds for Predictions and Receiver Operating Characteristic Curves

In order for the Autoencoder, OCSVM, and Nearest Neighbor to distinguish between two different classes they are trained on only normal (negative) instances. Once the learners are trained they have the ability to distinguish normal instances from abnormal ones via the reconstruction error of autoencoders, distance from hyperplane in OCSVM, or distance from training points in Nearest Neighbor, ie abnormal instances will have a larger value. There exists a threshold determination value which acts as a cutoff point between what is abnormal and what is not.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Figure 2

A mathematical procedure that can provide one with an optimal threshold determination comes from Receiver Operating Characteristic curves, better known as ROC curves. As explained in (Drummond 2005) the basis for any evaluation of a classifier's performance are numbers in a confusion matrix, given by Figure 2. The rows of the matrix represent the actual class of an instance where the columns represent the predicted class of each instance. These numbers must be calculated by applying a classifier, to a set of test samples. Dividing the entries in each row by the row total can give an estimate of how efficient the classifier is, or in other words it is the probability the classifier will make a particular prediction given an example from a certain class. These statistics are called the true positive rate, true negative rate, false positive rate, and false negative rate.

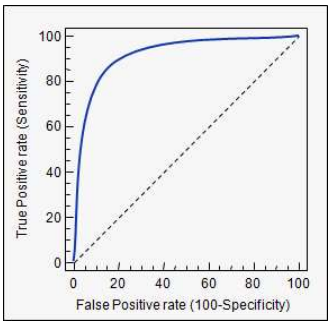


Figure 3

An ROC curve has two dimensions which are sensitivity and specificity, a picture can be seen in Figure 3. Sensitivity or true positive rate can be calculated as:

$$\text{Sensitivity} = TP/(TP + FN)$$

where TP is the true positive value and FN is false negative value. Specificity or false positive rate can be calculated as

$$\text{Specificity} = 1\text{- SPC} = 1 - (TN/TN + FP)$$

where TN is the true negative value and FP is the false positive value. From these two equations we can calculate sensitivity and specificity from varying threshold values. These values can be put onto a graph where sensitivity is the x axis and specificity is the y axis.

To determine an optimal threshold value what we want to do is find the minimum distance from the upper left hand corner which will provide us with a high true positive rate and a low false positive rate. To make it clearer if there exists a threshold value with coordinates (0,1) then that means it was able to predict all positive instances correctly and did not misclassify any. Although this is often not the case we can try to maximize true positive rate and minimize the false positive rate by simple euclidean geometry. To find this value we can use pythagorean theorem for each point generated by this equation

$$F(x,y) = \sqrt{x^2 + y^2}$$

where x is the specificity value and y is the sensitivity. The point with the shortest distance represents a certain threshold value that we will consider our optimal threshold.

3.5 Change Point Detection

The statistical method that is vital to this paper is change point detection. To describe change point detection there must be a setting which contains a set of random variables labeled X_1, X_2, X_3, . , ., .. that undergo abrupt changes to the distribution at change points C_1, C_2, .,.,.. A simple example can be shown in Figure 4 where a gaussian distribution suffers a change in the mean, and the change point is given by the dotted vertical line. Change point detection is capable of giving us at which point has there occurred a change.

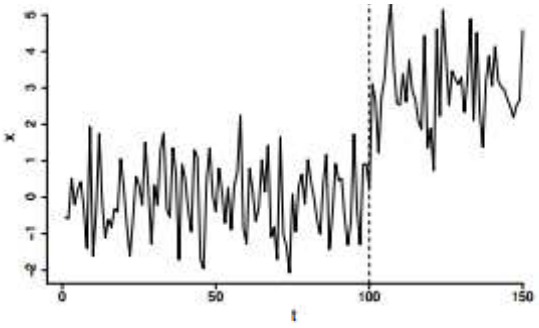


Figure 4

Change point detection can detect changes in two different settings, batch detection and sequential detection. In batch detection one is given a fixed size of n random variables labeled X_1, . , . , X_n and it is tested to see if there exists multiple change points in the sequence. This allows the detection method to look at all variables in the given sequence to determine changes. There also exists sequential detection which is not given a fixed size of variables and instead takes variables in one at a time. From the streaming in data points it will

let the user know if a change has a occurred and when it does it will restart the process over again. This paper will resolve to the former of the choices using batch detection.

The forms that change detection are either Nonparametric or Parametric. In parametric change point detection algorithms it is assumed that the underlying distribution of the data is normal. On the other hand a nonparametric method goes about finding change points when the underlying distribution may be unknown. We will use both methods throughout this paper to determine which is better suited for certain scenarios.

To further explain consider a batch setting where there contains a sequence of n random variables, $X_1,.,., X_n$ which contains at least one change point. If that change point exists at a given time C_0 then the distribution leading up to point C_0 which we will call F_0. After time C_0 there will exist another distribution called F_1 where $F_0 \neq F_1$. The way the change point is determined is by performing a two-sample hypothesis test where the test statistic must be chosen by what the assumption of the underlying distribution is. An example of a test statistic could be a shift in the mean, the variance, or the mean variance. This is typical of the parametric methods, where if we don't want to make any assumptions we can use a nonparametric choice to look for more general changes to the data.

Once a test statistic, D_k has been given the data will be partitioned into contiguous segments at each given point. From each segment the test statistic will be computed and standardized by subtracting the mean and dividing by the standard deviation of each value in the set $D_{1,..}D_k$ and taking the absolute value. It will be determined that there a change has taken place if point $D_n > h_n$ where h_n is a chosen threshold value. This value is calculated by bounding the Type 1 error rate. In other words the null hypothesis of no change is rejected, and the process starts all over again starting from point D_n.

A nonparametric to the approach is far more complex and uses what is known as a Mann-Whitney two sample test. The Mann-Whitney U test is a nonparametric test of the null hypothesis that says it is equally likely that a random selected value from one sample will be less than or greater than a randomly selected value from a second sample. To explain lets assume there exists a set of independent continuous random variables X_1, X_2 .. X_k, ...X_n, where the value k is a change point and the size of the set is n. There exists a U-statistic proposed by (Petit 1979) which is based on the Mann-Whitney two sided test. Let

$$D_{ij} = \text{sgn}(X_i - X_j)$$

Where sgn(x) = 1 if x>0, 0 if x = 0, -1 if x <0 then consider the equation

$$U_{l,T} = \sum_{i=1}^{t} \sum_{j=t+1}^{T} D_{ij}$$

This statistic called the U-Statistic is equivalent to the Mann-Whitney test explained previously. To be clear the U statistic is calculated for all values from 1 to n inclusive.

After having calculated all U statistics the next step in the process is to use another statistic to test out hypothesis of no change or change. Let,

$$K_T = \max \text{abs}(U_{i,T})$$

and then for changes in one direction we will need to involve the statistics.

$$K_T^+ = \max U_{i,T}$$
$$K_T^- = -\min U_{i,T}$$

To be clear the null hypothesis is that, $E(D_{i,j}) = 0$ and the distribution of $U_{i,T,}$ is symmetric around zero, which means that K_T^+ and K_T^- both have the same null distribution, thus no change has occurred. If the distributions are different and K_T^+ is large than there has been a shift down in the series. Also if K_T^- is larger than there has been a shift upwards in the series.

3.6 Change Point Detection + ROC Algorithm

To combine all of the previous ideas we will explain our algorithm for showing that changepoint detection can improve the threshold determination by partitioning the data into different sets of distributions. From the collection of reconstruction errors we can impose our change point detection methods on to the time series. From there we can can create a new threshold value for each given change by using ROC optimal thresholding. This will provide us with a varying threshold determination, based on different window sizes, that will be able to account for changes from negative to positive class values and will also be capable of handling a concept if one does occur. As previously discussed in section 2.3 this can be seen as an informed varying sliding window. The size of the window will vary dependant on where the change points in the system are found.

Algorithm

```
Procedure Test( TotalLosses)
    1.. ChangePoints = FindChangePoints(TotalLosses)
    2.  i = 0
    3. While i <  Number of ChangePoints
    4.          If TotalLosses[ChangePoint[i]] <= Threshold
    5.              Threshold = CalculateROCOptimal(Losses[ChangePoint[i] : ChangePoint[i+1]])
    6.          j = 0
    7.          While j < ChangePoint[i+1] - ChangePoint[i]
    8.                  If Losses[j] <= Threshold
    9.                          Instance = Negative
   10.                  Else
   11.                          Instance = Positive
   12.                  j = j+1
   13.          i = i+1
```

4. Experimental Setup

In this section we will provide details on exactly how the experiment was conducted and which data sets were included in the process.

4.1 Datasets

The data sets that will be included in this study come from artificial data created for this specific experiment and two practical datasets from UCI's Machine Learning Repository. The artificial data will take distinct forms provided in Figure 5. Each of these represent is a shift in the negative class over a given time scale. There will also be random amount of positive class values given at a random time. This will allow us to determine if change point detection is valid. Included in this experiment will be the use of ROC curves to determine the optimal threshold. The two datasets from UCI are namely, Shuttle Dataset and CoverType.

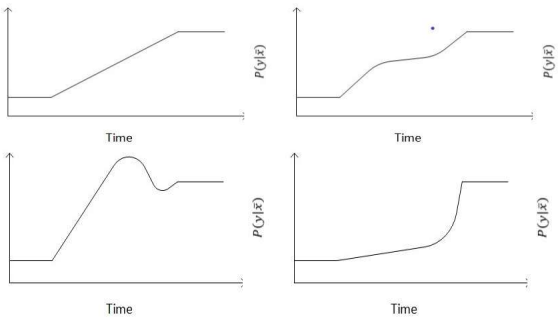


Figure 5

The shuttle data set contains statlog data from NASA that was generated over a time sequence. It's unclear what exactly this dataset represents but it is a classification problem. The set consists of 9 continuous numerical variables, although the first variable is time and the last is the classification. The covertype dataset was collected by the US Forest Service in the Roosevelt National Forest of northern Colorado. The goal is to predict the forest cover type from cartographic variables. The problem is defined by 54 variables of different types being continuous and categorical. The dataset contains 581012 examples. Results given for this dataset, using the first 15120 examples for training and the reminder 565892 for test are: 70% accuracy for backpropagation, 58% for a linear discriminant, and 68.6% for C4.5

4.2 Experimental Settings

We compared the performance of using change point detection with two different static threshold determinations. The threshold determinations tested against are ROC optimal and a heuristic measurement. The ROC optimal was explained previously in section 3.4 and will be used to determine the optimal threshold over the entire testing set. The heuristical measurement is more complicated and is essentially a guess and check method over the entire set.

The heuristic search involved us looking at multiple different statistical measurements most notably variations combing sums of mean and standard deviation. We would start with the value mean + 2* Standard Deviation, because we would just make the assumption that it was distributed normally and that would give us 68-95-99.7 rule to use. From there we would try values lower and high to see if the statistics got any better and would continually do this until we

reached a point that felt we couldn't do any better. We would try to to keep in mind balancing different statistics and not trying to just have one statistic perform well.

The artificial data was created specifically for change point and ROC optimal thresholding. There were no learners involved in this process and the ultimate goal was to show that change point is a viable option under different graphical representations. The way the sets were created was simply generating a function that would mimic the graphical representations in Figure 5. It would involve generating a random number between values we would consider the normal class and increase/decrease those values over time. There would also be random amount of abnormal cases that would have a higher value than the normal classes every 300 steps in time. The total amount of points generated was 10,000. An example would be the normal class being random values between (0, .01) inclusive. The positive class would then we random variables between (.07 - .2). After a given time stamp, say 1000, we could start increasing both of these numbers by a value for example 1.2 and we would see a concept drift emerge.

For the UCI datasets set we implemented an autoencoder using Python and notably Google's tensor flow package. The activation function of the autoencoder was the famous sigmoid function which is given by:

$$F(x) = 1/(1+e^{-x})$$

The number of hidden units was half of the number of features given and truncating the decimal, which given the shuttle data after removing time and class value, was 4 hidden units. The learning rate for this experiment was .05. During the training phase we involved both mini-batch gradient descent and stochastic gradient descent. Mini-batch gradient descent involved batch sizes of 5 and a total of 280,000 total batches. After mini-batch came stochastic gradient descent which involved a total of 4 epochs.

The OCSVM used was implemented using Python's scikit-learn package. We decided to use the sigmoid function, which was previously explained, because it had the best results. The kernel coefficient used was 1/Number of features.

The Nearest Neighbor Algorithm used also comes from Python's scikit-learn package. For a distance measurement we used minkowski distance which is a distance metric in a normed vector space. Minkowski distance is considered to be a generalization of Euclidean distance and Manhattan distance. It can be defined as

$$(\sum|x_i - y_i|^p)^{1/p}$$

where when p = 1 it is Manhattan distance, and when p = 2 it is Euclidean distance. In this given experiment p =2.

The normal class (negative class) in the shuttle data set was class 1 notably because it was the largest class and it spanned the entire time of the data set. This involved 31,108 training examples taken from the training set provided by UCI. The testing set involved 14,500 examples provided in the testing set given. Testing methodology used was to train on a given set of training data and then to test on an entire set of testing data.

For covertype the normal class was also 1 for the same reasons as shuttle as it was the largest class that came up in the testing data. The learners were trained on 1600 instances that were given in the training set. They were then tested on 565912 examples which is the entire testing set given.Testing methodology is the same as previous with a train on the entire training set and test on the entire testing set.

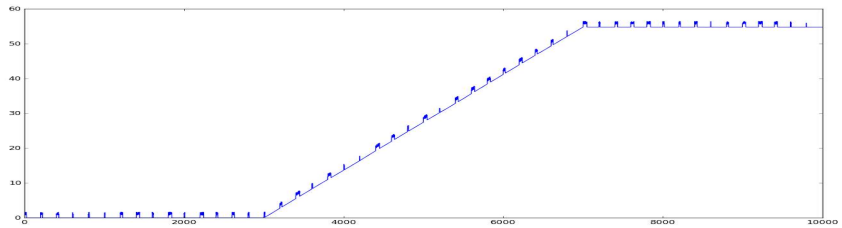
We decided to go with using a test and train method because other known methods such as cross-validation did not make sense in this case. Cross-validation tends to be one of the more popular procedures for performance by randomly splitting the data into varying sizes of training and testing data. Although that wasn't applicable here because mixing the data in that way would change the arrangement of events and we could potentially lose a concept drift.

The change point detection algorithms provided by the R packages come from both parametric and nonparametric families. The parametric algorithm comes from the package ChangePoints and the name of the algorithm is PELT. As explained in section 3.3 a parametric algorithm must be provided a statistical measurement and in this case it will be variance. From the R package CPM the nonparametric algorithm Mann-Whitney will be applied. The value for ARL0, which is the estimation of how many examples will occur until a false positive arises, is the standard 500.

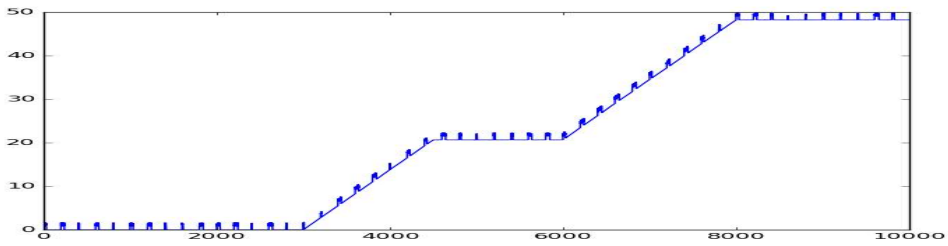
For performance measurements we looked across a variety of results to get a very good idea of how our method was performing. The statistical measurements calculated are Accuracy, Precision, False Positive Rate, and AUC. Accuracy and precision tend to be benchmark performance measures. We found the false positive rate to be of importance since we are making the claim change point will lower false alarm rate. AUC was added just to give the reader a sense of how the learner was performing overall.

5. Experimental Results

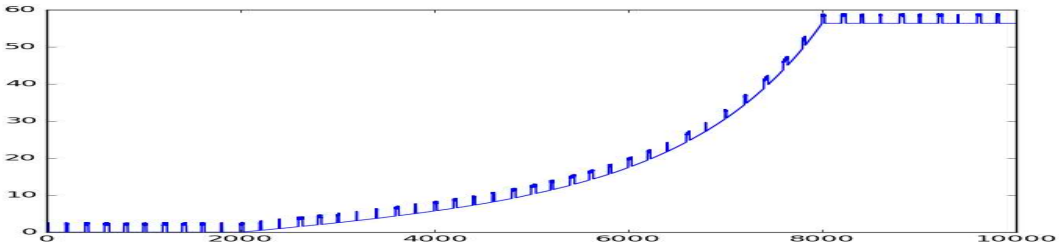
The results of the artificial data set will follow below providing a graph of each which represent possible reconstruction errors of a data set. Below the graph will follow a table of results gathered from the experiment.



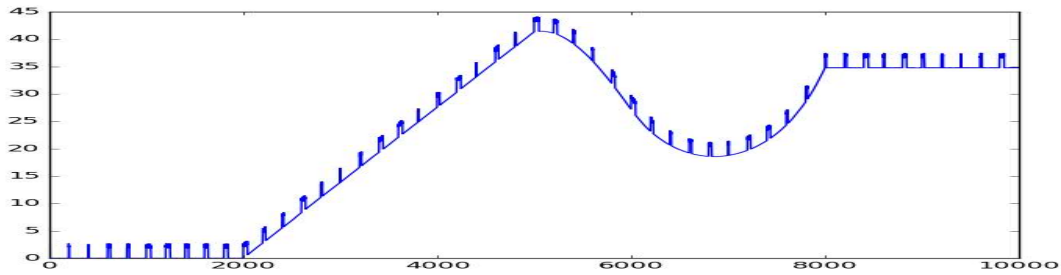
Type	Algorithm	Accuracy	Precision	FP Rate	AUC
Change Point	Mann-Whitney	99.58%	96.83%	.48%	.54
ChangePoint	PELT	13.26%	100%	N/A	.54
ROC - Optimal		51.49%	51.36%	12.61%	.54
Heuristic		88.19%	10.98%	11.98%	.54



Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	99.57%	96.35%	.46%	.57
ChangePoint	PELT	38.98%	62.68%	12.23%	.57
ROC -Optimal		60.36%	55.64%	8.86%	.57
Heuristic		90.09%	15.95%	10.1%	.57



Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	99.65%	97.15%	.39%	.55
ChangePoint	PELT	12.28%	100%	N/A	.55
ROC- Optimal		40.69%	66.12%	11.32%	.55
Heuristic		89.93%	17.99%	10.29%	.55



Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	99.64%	97.02%	.41%	.6
ChangePoint	PELT	32.82%	78.24%	10.12%	.6
ROC - Optimal		48.61%	69.56%	8.39%	.6
Heuristic		88.38%	3.88%	11.67%.	.6

After showing our results on artificial data sets we will now show results on the shuttle and cover type data sets for the Autoencoder, OCSVM, and Nearest Neighbor.

Shuttle - Autoencoder

Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	95.28%	88.05%	3.13%	.91
ChangePoint	PELT	93.39%	93.08%	1.91%	.91
Roc- Optimal		80.78%	86.73%	4.33%	.91
Heuristic		89.68%	53.31%	11.02%	.91

Shuttle - OCSVM

Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	93.53%	94.08%	1.22%	.82
ChangePoint	PELT	88.44%	66.19%	12.24%	.82
Roc - Optimal		74.4%	44.07%	28.37%	.82
Heuristic		79.1%	49.46%	3.63%	.82

Shuttle - Nearest Neighbor

Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	95.52%	97.86%	.46%	.87
ChangePoint	PELT	97.24%	91.05%	2.49%	.87
Roc- Optimal		72.83%	42.43%	30.39%	.87
Heuristic		86.99%	86.22%	1.88%	.87

CoverType - Autoencoder

Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	65.75%	91.11%	39.98%	.69
ChangePoint	PELT	62.66%	69.47%	50.37%	.69
Roc- Optimal		62.94%	59.66%	50.00%	.69
Heuristic		63..27%	89.49%	48.8%	.69

CoverType - OCSVM

Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	62.32%	98.51%	74.85%	.61
ChangePoint	PELT	62.13%	97.89%	72.42%	.61
Roc- Optimal		56.89%	55.01%	55.98%	.61
Heuristic		60.72%	73.87%	53.63%	.61

CoverType - Nearest Neighbor

Type	Algorithm	Accuracy	Precision	FP Rate	AUC
ChangePoint	Mann-Whitney	68.54%	84.83%	38.67%	.73
ChangePoint	PELT	66.27%	72.45%	45.62%	.73
Roc- Optimal		66.25%	65.55%	46.46%	.73
Heuristic		69.04%	82.64%	39.09%	.73

The given results show clearly that in majority of cases change point detection can indeed be a better option than using on a static threshold. Results in the artificial data show that the nonparametric changepoint detection scheme worked better than the parametric but that may be due to the data not fitting in a normal distribution. This shows how parametric change point detection, although extremely powerful, may not always be the best suited tool. In the covertype data set it is noticeable that using an autoencoder along with the nonparametric algorithm was able to have the highest precision percentage and the lowest false positive rate. We must point out though, when it comes to the shuttle data set the parametric algorithm outperforms the nonparametric in certain statistics. For example, when using Nearest Neighbor the parametric algorithm has a higher accuracy but has a lower precision.

Many of these performance measures may depend on whether a false positive rate is of more importance or if overall accuracy is more important. A simple example can come from how HIV tests work, it's better to have a higher false positive than a high false negative. To explain it's better to tell someone they have HIV when in fact they don't than to tell someone they don't when they actually do have to disease and could possibly spread it. These are the types of questions one will have to keep in mind when deciding what is of the utmost importance to a given situation.

6. Conclusion

In this paper we have concluded that change point detection is in fact a viable option when it comes to handling concept drift and changes. It almost always outperforms a static threshold and performs extremely well on time series data. It also has the ability to work on several different types of learners so one may be able to choose what suits the current situation best.

We believe under different conditions we may be able to see a greater gap between using change point and static thresholds dependant on a dataset. The shuttle data set gives only integer values for a time sequence and it isn't specific over how long that data set was collected. If we were to have a huge data set with greater concept drifts we believe that this method will show of even greater value than shown here.

The fall back of this strategy is that it relied heavily on ROC optimal threshold values. In a practical situation this would not be feasible because as explained in section 3.4 to find this threshold one must know the class values it fell in. A possible use could be related to how online learning algorithms update at a later stage once it knows the true value. Although this wouldn't be able to adapt very quickly to constant streaming data.

There are a lot of different avenues that this method could be applied to with some tweaks to how it works. I will be proposing as a senior year capstone project to use sequential change point detection instead of the batch detection as used in this paper. With the use of sequential change point and online learning algorithms with a variable sized sliding window, the sliding window will be able to vary its size depending on when change points are found. It's possible to grow a sliding as we are checking for change points, and once one is found we can then shrink the window to handle adapting to a concept drift. I see this method being highly import to the advancement in being able to learn quickly over time. This will require retraining of the system when concept drifts occur and will be a lot more complex than the given algorithm in this paper.

References

Elkan, Charles, Nearest Neighbor Classification, 2008, University of California San Diego

Drummond, Chris, Holte, Robert C., Cost curves: An improved method for visualizing classifier performance

GAMA, JO˜AO, INDR˙ E ˇ ZLIOBAIT˙ E, ALBERT BIFET, MYKOLA PECHENIZKIY, and ABDELHAMID BOUCHACHIA. 2014. A survey on concept drift adaptation. ACM Computing Surveys (CSUR),46(4):44

Hawkins, Douglas M., Deng, QIQI, A Nonparametric Change-Point Control Chart School of Statistics, University of Minnesota, Minneapolis, MN 55455-0493

Heller, Katherine A., Svore, Krysta M., Keromytis, Angelos D., Salvartore, J. Stolfo, One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses, Columbia University, New York, NY

Pettit, A. N. 1978 A Nonparametric Approach to the ChangePoint Problem, University of Technology, Loughborough, Leics, England

Ross, Gordan J. ,Parametric and Nonparametric Sequential Change Detection in R: The cpm Package University College London

Webb, Geoffrey I., Hyde, Roy, Cao, Hong, Nguyen, Hai Long, Petitjean, Francois, Characterizing Concept Drift