ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/331599344

A company is collecting and storing data from over 1000 gas turbines. They store the value about 200 sensors on each turbine every minute and want to keep this for five years. One...

Preprint · March 2019

DOI: 10.13140/RG.2.2.26212.22405

CITATIONS
0

READS
17

1 author:

Sunil Mandhan
Hitachi, Ltd.

7 PUBLICATIONS **4** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Detection of startup and shutdown events in gas turbines. View project

Project Creating a compiler from scratch - simple enough to replicate and understand. View project

All content following this page was uploaded by Sunil Mandhan on 08 March 2019.

The user has requested enhancement of the downloaded file.

1. Problem/Purpose

The goal is to write a program that finds each startup in a dataset based on the training data that is provided.

This program is to output the date and time for each turbine startup in the data. SAX must be used for data reduction. After the data has been reduced, each word in the data should be inspected and determined if there is a startup in that word by using a k-nearest neighbor algorithm.

3. Assumptions

1. It is assumed that if a startup event is detected in a window then the next startup should not be looked in the next immediate window starting with the second SAX symbol of the first window. It should be looked in the next window after the current window ends.
2. SAX algorithm is implemented the excel sheet manually using the formula. The data values from the training data and test data excel sheets are copied in to the source code manually.
3. Turbine test data and training data are from the same turbine.

4. Results

1. Working source code and program is produced.
2. There are total 6 startups in the data. Time for each startup event is reported.
3. Questions to be asked about the assignment are answered.

5. Conclusions

This assignment helped in understanding the handling of big data programmatically and provided many challenges such as:

1. Normalizing the data to be scaled for the SAX levels.
2. Implementing the efficient SAX algorithm.
3. Implementing the efficient k-nn algorithm.

Along with this, the assignment provided a way to look at the work done in the assignment to answer some questions that are related to performance improvements and alternative approaches.

Table of Contents

List of Figures

List of Tables

1. PROBLEM STATEMENT

A company is collecting and storing data from over 1000 gas turbines. They store the value about 200 sensors on each turbine every minute and want to keep this for five years. One thing they would like to do with this data is look back over the data to find all occurrences of interesting events. An example of an interesting event is a turbine startup.

The goal is to write a program that finds each startup in a dataset. This program is to output the date and time for each turbine startup in the data. SAX must be used for data reduction. After the data has been reduced, each word in the data should be inspected and determined if there is a startup in that word by using a k-nearest neighbor algorithm.

The turbine data to be analyzed for startups consists of one week of turbine speed data (TNH) from one turbine. TNH is the speed at which the turbine is rotating. This data is given once per minute.

Training data that consists of examples of startups and non-startups is also given. Each training example consists of sixty data points and is labeled as a shutdown (0), startup (1), or other condition (2). This data is used by a nearest neighbor algorithm that determines if the current word of the turbine data is a startup or not. DataLabel

The program should read in the data, call the SAX algorithm, and then call the nearest neighbor algorithm.

For deliverables, following items are to be submitted:

1) Working source code and program.
2) List of the times for all startups in the turbine data
3) A written description that describes the work and answers the following questions
    - What is the time needed for this algorithm to run (assuming n is the size of the turbine data, t is the size of the training data, and m is the number of training examples)?
    - Ways to speed this up (for big datasets)?
    - What values were used for the SAX parameters segment, alphabet, and word size? Why are these good values?
    - What, if any, additional training examples did you create?
    - How does your nearest neighbor algorithm determine similarity?
    - Do your sliding windows overlap, if so by how much?

2. RESULTS AND ANALYSIS

2.1 Results

Results are prepared based on the deliverables requirements.

1. Working source code and program and related files
   Please refer to the appendix A for the source code and other details.

2. List of the times for all startups in the turbine data
   There were total 6 startups detected in the test data, details are given below in table 1.

Table 1 – Startup index and detection time

Startup index	Startup time
1	Sun Jan 01 23:01:00
2	Mon Jan 02 23:01:00
3	Tue Jan 03 23:01:00
4	Wed Jan 04 22:56:00
5	Thu Jan 05 22:56:00
6	Fri Jan 06 22:51:00

2.2 Analysis

There were two algorithms used when developing the program and source code. These two algorithms are:

1. SAX algorithm
2. K-nn algorithm
3.

These are explained below in ample details.

2.2.1 *SAX Algorithm:*

To get the SAX representation of the given time series two parameters had been used.

Segment size was chosen as 5 and hence the word size became 12 as each training example had 60 data points. The alphabets were chosen as 5 in numbers.

First the time series was normalized using the mean and standard deviation of the entire time series training data.

The mean of the series came out to be: 53.4528

Standard Deviation was: 43.5532 43.5532

The normalization was done using the following formula:

Normalized data = (data - mean) /standard_deviation

Then the average was taken over a non-overlapping window of size 5. This converted each row of the training data to a length of 12.

Then in order to ensure almost equally probable symbols within a SAX word one should use a sorted list of breakpoints as shown below.

It was decided to use four break points at {-.84, -.25, .25, .84}.

Based on these breakpoints, the distance between possible two SAX symbols' combinations was calculated. These distances are mentioned in the Table 2 below.

Table 2 – distance between two SAX symbols

SAX Symbol	**a**	**b**	**c**	**d**	**e**
a	0	0	.59	1.09	1.68
b	0	0	0	.59	1.09
c	.59	0	0	0	0.59
d	1.09	.59	0	0	0
e	1.68	1.09	.59	0	0

Once the averaging was done in windows of 5 then each average was converted to symbols based on the above breakpoints.

For the testing data also averaging over 5 was done and each SAX word length was chosen to be 12. To get the next SAX word the window moved by 1 SAX symbol which means 5 data points. So the new SAX test word will have 11 symbols same as the previous test word left shifted and a new symbol at the end of the word.

The flow chart for the SAX algorithm is given below in Figure 1.

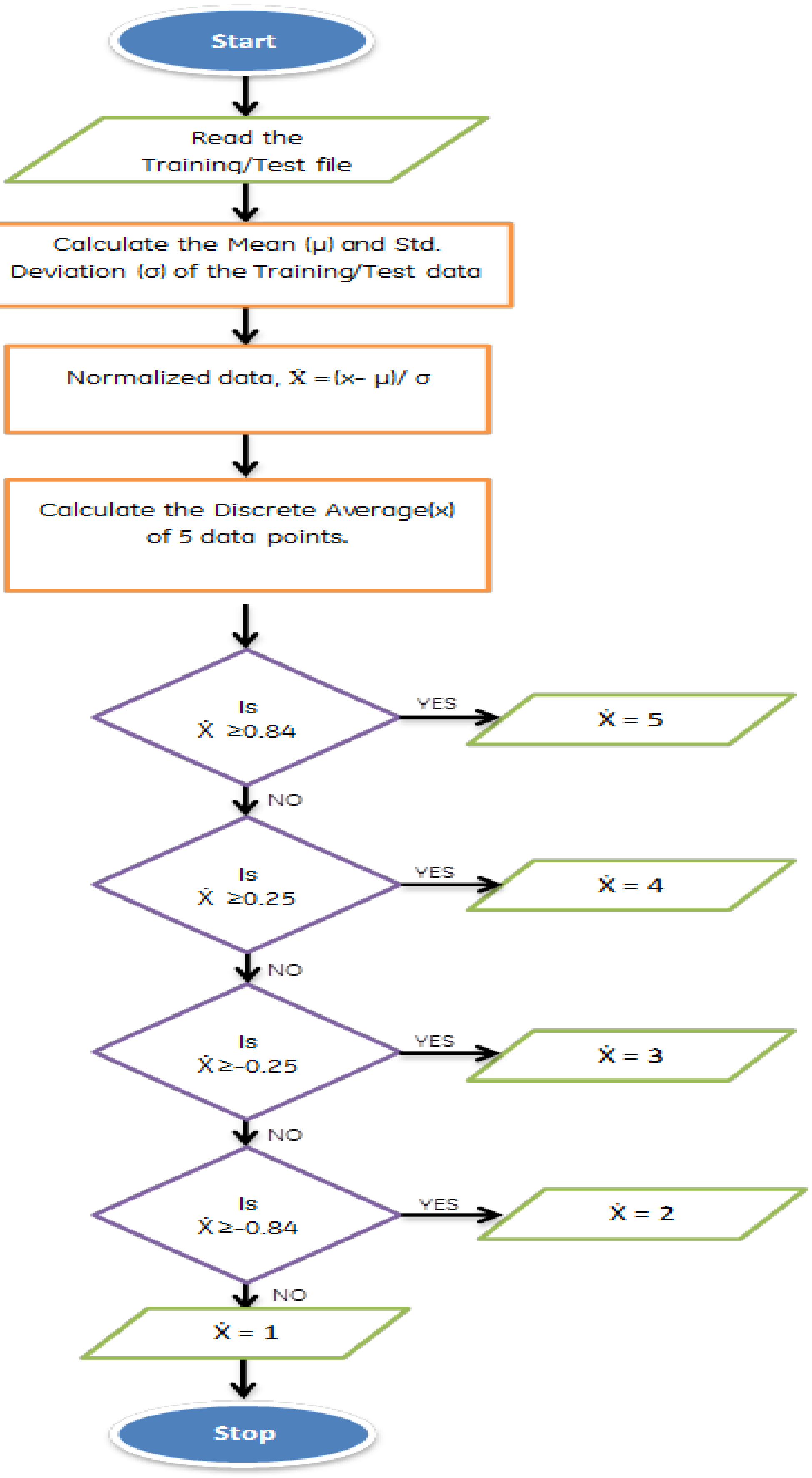
Start
Read the Training/Test file
Calculate the Mean (μ) and Std. Deviation (σ) of the Training/Test data
Normalized data, Ẋ = (x- μ)/ σ
Calculate the Discrete Average(x) of 5 data points.
Is Ẋ ≥0.84
YES
Ẋ = 5
NO
Is Ẋ ≥0.25
YES
Ẋ = 4
NO
Is Ẋ≥-0.25
YES
Ẋ = 3
NO
Is Ẋ≥-0.84
YES
Ẋ = 2
NO
Ẋ = 1
Stop

Figure 1 – SAX flow chart

2.2.2 *K-Nearest Neighbor Algorithm:*

The distance between two SAX words could be calculated in terms of the SAX levels that have been chosen. For the distances between two SAX symbols, please refer to the Table 2 in section 2.2.1.

But it was decided to calculate distance in a simpler and efficient way which is as follows.

Code each symbols as a=1, b=2, c=3 and so on as given below in Table 3.

Table 3 – SAX symbols and corresponding numeric values

SAX symbol	**Corresponding numeric value**
a	1
b	2
c	3
d	4
e	5

Distance between two SAX words is calculated as:

Distance (SAX1, SAX2) = $\sum_{i=1}^{Wordlength} abs(SAX1[i] - SAX2[i])$.

For each test example find 5 training examples that were closest to it. The final class of the test sample is predicted as the class that majority of its neighbor belongs to.

The flow chart for this algorithm is given below in Figure 2.

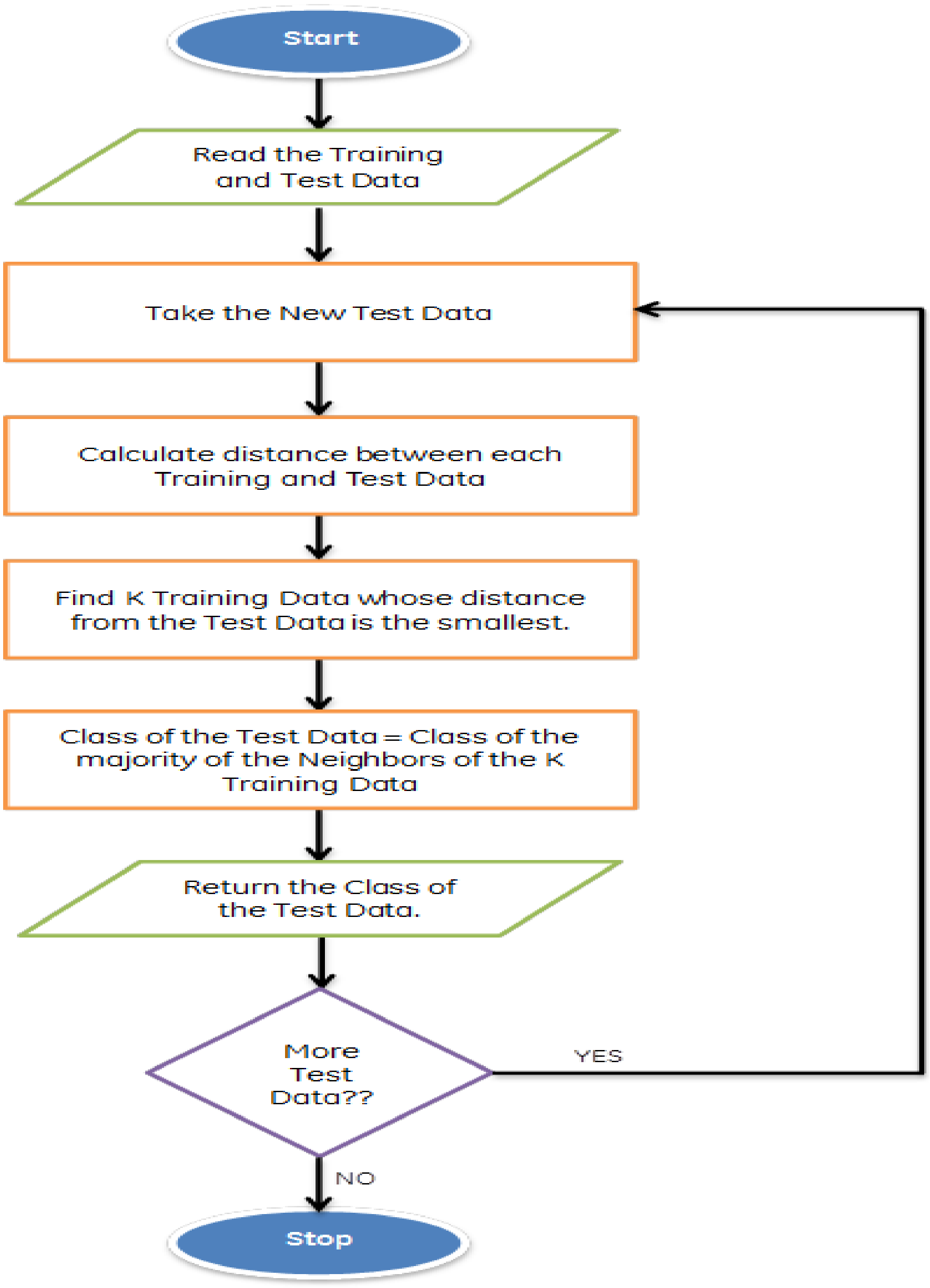
Start
Read the Training and Test Data
Take the New Test Data
Calculate distance between each Training and Test Data
Find K Training Data whose distance from the Test Data is the smallest.
Class of the Test Data = Class of the majority of the Neighbors of the K Training Data
Return the Class of the Test Data.
More Test Data??
YES
NO
Stop

Figure 2 – flow chart for k-nn algorithm

2.3 Questions to answer:

Question
What is the time needed for this algorithm to run (assuming n is the size of the turbine data, t is the size of the training data, and m is the number of training examples)?

Answer
SAX: According to the flowchart, SAX consists of five distinct parts namely, finding the mean of training data, finding the standard deviation, normalizing the data, reducing data size by taking an average of 5 data points and finally classifying data into SAX levels.

Each activity requires basic operation like comparison, addition, multiplication or division which we assume can be done in unit time by the processor. Hence total time for converting the training data (size t) to SAX format and test data (size n) is **5t + 5n.**

K-nn: According to flowchart, K-nn involves calculating distance of W length SAX word with m training sets each of length W. This takes a time of W*m, assuming distance lookup takes unit time. Then sorting a length m array of distances takes m^2 time. This operation is run n-W times on the test data. The final operation is finding K nearest neighbors for each window and classifying it as a startup or non-startup.

So total run time of K-nn is **(W*m + m^2 + K+2)*(n-W)**

W=12, t=22*60, m=22, n=10066

Question
Can you think of ways to speed this up (for big datasets)?

Answer
For big data, iSAX can be used instead of SAX.
Segment size can be increased since its large data and better compression can be achieved.
The algorithm should run in in linear or even sub-linear time. So in KNN, sorting algorithm chosen should be linear (i.e n) in execution and not n*log (n) which is generally used.

Question
What values did you use for the SAX parameters segment, alphabet, and word size? Why are these good values?

Answer
Segment=5, alphabet=5, and word size=12.

Since the TNH value varies from 0 to 100 and there is a linear increase or decrease during startup and shutdown, alphabets were chosen so that each regional could capture the increase or decrease. Also 5 min average was chosen because the data pattern doesn't make any significant change over 5 min window lengths.

Question

What, if any, additional training examples did you create?

Answer:
We didn't create any extra training samples. We thought that the given samples are enough for the detection.

Question
How does your nearest neighbor algorithm determine similarity?

Answer:
This answer can easily be inferred from the section 2.2.2 and Figure 2.

Question
Do your sliding windows overlap, if so by how much?

Answer
Sliding windows were overlapping. When a startup was detected in a window suppose from w1, w2, ..., wn then the next startup was also detected in w2, w3, ..., wn+1 window. To void this , it was decided to skip the current window and strat at the next window like wn+1, wn+2, ..., w2n, and so on.

3. DISCUSSION

This assignment helped in understanding the development of the algorithms for the big data manipulation. We needed to find the efficient algorithm to deal with the big data to find the matching pattern in that data.

Some of the design trade-offs that came into play while implementing the program were:

1. Efficiency of the program
2. Simplicity of the algorithms

Some of the major issues and solutions recommended are given in the section 2.3 with question and answer format.

Appendix A – Source code and other details

The source code is attached along with this report with the file name given below.

TurbineStartupEventFinder.zip

The source code is written in java programming language and can be opened in eclipse tool.

The SAX converted test data and training data is also attached along with the report in the folder with the name given below.

Dataset

View publication stats