B. TECH. PROJECT REPORT

On

Machine Learning Algorithms for Constraint Environment

BY

Keshav Goyal(150001014)

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2018

Machine Learning Algorithms for Constraint Environment

A PROJECT REPORT

Submitted in partial fulfillment of the
requirements for the award of the degrees
of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Keshav Goyal(150001014)

Guided by:

Dr. Abhishek Srivastava, Associate Professor

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2018

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **Machine Learning Algorithms for Constraint Environment** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering completed under the supervision of **Dr. Abhishek Srivastava, Associate Professor, Computer Science and Engineering,** IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Keshav Goyal

15001014

CERTIFICATE BY BTP GUIDE

It is certified that the above statement made by the students is correct to the best of my knowledge.

Dr. Abhishek Srivastava
Associate Professor
Department of Computer Science and Engineering
IIT Indore

PREFACE

This report on **"Machine Learning Algorithms for Constraint Environment"** is prepared under the guidance of **Dr. Abhishek Srivastava.**

In this project hybrid fuzzy neural network has been used for classification of datasets into classes. Optimized compression algorithm has been used to compress the data sets while procuring comparable accuracy. Compression algorithm is implemented using MATLAB. Machine Learning model is implemented using python. There are various applications of this project. This analysis proves to be useful for predictions in constraint environment using IOT devices effectively and conveniently.

Keshav Goyal
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

Acknowledgements

I would like to thank my B-Tech project supervisor **Dr. Abhishek Srivastava** for their guidance and constant support in structuring the project and their valuable feedback throughout the course of this project. Their overseeing the project meant there was a lot that I learnt while working on it. I thank them for their time and efforts.

I am grateful to **Mr. Arun Kumar** without whom this project would have been impossible. He provided valuable guidance with the Mathematics involved in the project.

I am really grateful to the Institute for the opportunity to be exposed to systemic research especially **Dr. Abhishek Srivastava'**s Lab for providing the necessary hardware utilities to complete the project. Lastly, I offer my sincere thanks to everyone who helped me complete this project, whose name I might have forgotten to mention.

Keshav Goyal
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

Abstract

A new method is presented to solve the problem of forest fire detection. Hybrid fuzzy neural network is being used to classify the data points into classes. IOT devices are being used for all the real time computation. But the computation power of IOT devices is limited. We cannot directly execute machine learning algorithms on IOT devices due to which we need to preprocess the dataset. An optimized compression algorithm is being used to compress the dataset. The prototype locations are optimized through Adam optimizer and deterministic annealing algorithms. The optimized algorithm also includes an initialization strategy which aims to provide the maximum classification rate on the training set with the minimum number of prototypes. Results of experiments shows that the modified compression algorithm is more efficient than the previous algorithm.

Contents

List of figures

Chapter 1: Introduction

A forest fire is simply an uncontrolled fire that is wiping out large fields and areas of land. Forest fires have harmful effects on wildlife and forest vegetation. Forest fires depletes a lot of natural resources. These forest fires leads to increment in carbon dioxide, carbon monoxide in atmosphere.

Forest fire prediction is one of the major applications of using machine learning algorithms in constraint environment. By predicting forest fire, we can control the fire before it spreads in larger area, by which we can reduce the loss of life and environmental resources. To detect the forest fire, we capture four main components of the environment using IOT sensors. We use Humidity, Temperature, Smoke, IR sensitivity to predict the forest fire. After successful capturing of data, we used Hybrid Fuzzy Neural network to classify the data into 3 classes. First class characterizes that the chances of fire are higher i.e. close of 100, second class characterizes that the chances of fire are very slim i.e. close to 0, third class characterizes that the fire is ambiguous. We only have IOT devices in the forest to predict the forest fire. But the computation power of IOT devices are not enough to fulfill the requirements.

We need a compression algorithm using which we can reduce the real time computations while procuring comparable accuracy. In the beginning we generate prototypes using classified dataset with the help of K-Means algorithm. Now we need to optimize the positions of each and every prototype. We choose an error function which depends on the position of prototypes. We are using iterative optimization algorithms to reach the local/global minima of the error function. By which the position of prototypes gets optimized. This compression algorithm is main motive of our project. The compression algorithm has already been implemented. But the positions of prototypes were not optimized in algorithm. Therefore, we are using adam optimizer and deterministic

annealing to optimize the positions of prototypes. After optimizing the position of prototypes, we can remove the inactive prototypes, hence minimize the computational power required.

Chapter 2: Literature Review

There are various research papers already published on forest fire prediction project. But they chose different strategy/model to solve the problem. There are various research papers already published on compression algorithm which are used to compress the data points.

2.1 Model

IOT sensors in the forest records data of the environment and send the data to the cloud through a wireless network. Cloud does all the processing and send the results to the nearest fire extinguishing department.

2.1.1 IOT Sensors

IOT devices are used to monitor the environment and send the recorded data to cloud over wireless network. The sensors used are Temperature, Smoke, Photoelectric Transducer, Anemometers(wind). They used the above components for prediction of forest fire.

2.1.2 Cloud Engine

Cloud Engine is a managed service that allows developers to build and bring superior machine learning models to production. Cloud Engine receives data from IOT devices and run the Machine Learning model to predict the results. Afterwards it sends the results to forest fire extinguishing department over wireless network.

2.2 Compression Algorithm

This algorithm is used to reduce the dataset to fewer data points. Afterwards these data points can be used in place of the original dataset for with comparable accuracy.

2.2.1 Neighborhood Components Analysis

In this paper they propose a novel method for learning a Mahalanobis distance measure to be used in the KNN classification algorithm. The algorithm directly maximizes a stochastic variant of the leave-one-out KNN score on the training set. It can also learn a low-dimensional in car embedding of labeled data that can be used for data Visualization and fast classification. Unlike other methods, our classification model is non-parametric, making to assumptions about the shape of the class distributions or the boundaries between them. The performance of the method is demonstrated on several data sets, both for metric learning and linear dimensionality reduction.

2.2.2 Stochastic Neighbor Embedding

They describe a probabilistic approach to the task of placing objects, described by high-dimensional vectors or by pairwise dissimilarities, in a low-dimensional space in a way that preserves neighbor identities. A Gaussian is centered on each object in the high-dimensional space and densities under this Gaussian (or the given dissimilarities) are used to define a probability distribution over all the potential neighbors of the object. The aim of the embedding is to approximate this distribution as well as possible when the same operation is performed on the low-dimensional "images of the objects. A natural cost function is a sum of Kullback-Leibler divergences, one per object, which leads to a simple gradient for adjusting the positions of the low-dimensional images. Unlike other dimensionality reduction methods, this probabilistic framework makes it easy to represent each object by a mixture of widely separated low-dimensional images. This allows

ambiguous objects, like the document count vector for the word "bank", to have versions close to the images of both "river" and "finance" without forcing the images of outdoor concepts to be located close to those of corporate concepts.

2.2.3 Finding prototypes for nearest neighbor classification

A new method was presented to find prototypes for a nearest neighbor classifier. The prototype locations are optimized through a gradient descent and a deterministic annealing process. The proposed algorithm also includes an initialization strategy which aims to provide the maximum classification rate on the training set with the minimum number of prototypes.

2.2.4 Inactive prototypes elimination algorithm

Most research efforts in Empirical Concept Learning have been devoted to nominal attribute spaces. Using numerical ordered spaces rises some specific problems, especially for Top-Down algorithms. It builds a Flexible Matching function based on a Decision Tree issued from a classical Top-Down Induction algorithm. Empirical tests show that NFDT performs better than a Classical Decision Tree method, especially when noise is present and that it can be used to evaluate the quality of concept descriptions issued from a Decision Tree.

2.3 Drawbacks

The previous model had major drawbacks due to which it was not efficient and convenient to use. From the above described research papers on compression algorithm second last is major motivation behind the compression algorithm of our project. But there are some limitations of algorithm described in the research paper.

- It is difficult to send data to cloud engine from the forest, since we cannot guarantee reliability in network connection. Due to which it is not efficient and can take time, meanwhile fire can spread in larger area.
- The previous model is not cost effective. Because it takes a lot of money to set up a cloud machine with high computing power. The stochastic algorithm used is not optimal.
- It is not storage optimal and the required computation power is greater.

Chapter 3: Our Approach

There are various real life problems, we can solve by using machine algorithms in constraint environment. Forest fire prediction is one of the them. This chapter is about the algorithm we are using to solve the problem of forest fire prediction. In the beginning we collect and classify our own dataset. Afterwards we use the compression algorithm to reduce the required computing power.

3.1 Dataset Collection

We collect the dataset using IOT sensors. We use 3 different types of sensors to capture the surrounding data. We are capturing 4 components of the climate to predict the forest fire i.e. temperature, humidity, smoke, IR sensitivity.

1) Temperature and Humidity sensor- It uses a capacitive humidity sensor and a thermistor to measure the surrounding temperature.
2) IR Flame sensor- This sensor is specifically designed to respond to 4.3μm light emitted by hydrocarbon flames.
3) Smoke sensor(MQ7)- This gas sensor detects the concentrations of CO in the air and output its reading as an analog voltage.

3.2 Classification Model

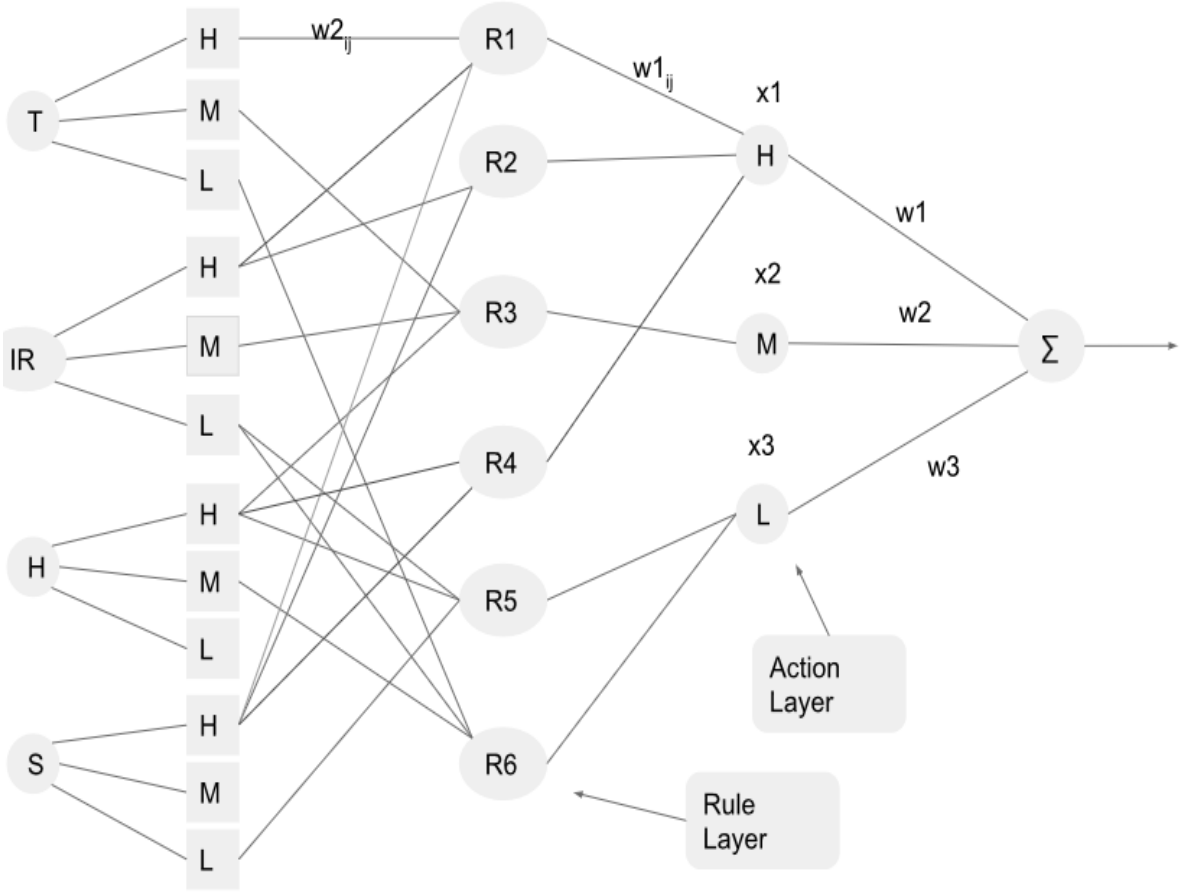

Figure 1: Fuzzy Neural Network Model using for classification

We are using Hybrid Neuro-Fuzzy System for classification and training purposes. A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing training data points.

There are 5 layers in our Hybrid Neuro-Fuzzy System.

i. Input Layer
ii. Fuzzification Layer
iii. Fuzzy Rule Layer
iv. Action Layer
v. Output Layer

3.2.1 Input Layer

This layer of our model has 4 nodes for each attribute. This layer of fuzzy neural network model takes raw input and forwards the value of each attribute to the next layer.

3.2.2 Fuzzification Layer

Fuzzification is the process of changing a scalar value into fuzzy value. We are using 3 linguistic variable for each input and output attributes i.e.{ High, Medium, Low}. We are using gaussian membership function for fuzzification. Initially the membership functions are spaced equally over the weight space, although if any expert knowledge is available this can be used for initialization. In order to maintain the semantic meaningfulness of the memberships contained in this layer of connections some restrictions are placed on adaptation. When adaptation is taking place the centers are limited remain within equally sized partitions of the weight space.

3.2.3 Rule Layer

This layer consists of Fuzzy rules. The connections from the previous layer to this rule layer are used to perform pre-condition matching of fuzzy rules. The layer is also expandable in that nodes can be added to represent more rules as the network adapts. For ex. IF Temperature is High and smoke is High and IR is High THEN Fire is High.

3.2.4 Output Layer

The output layer performs a modified center of gravity defuzzification. Defuzzification is the process of converting a fuzzified output into a single crisp value with respect to a fuzzy set. Among the commonly used defuzzification strategies, the Center of Gravity (COG) method yielded the best result. Adapting the output membership functions would mean moving the centers, but the requirement that the membership degrees to which a particular output value belongs to the various fuzzy labels must always sum up to one, is always satisfied. For each center, there is a constraining band (partition) where this value can move to.

3.2.5 Backpropagation

The Back-propagation algorithm looks for the minimum value of the error function in weight space using a technique called the gradient descent. We compute for each weight and update the weight iteratively. A modified backpropagation algorithm is used for the purpose of rule adaptation. This adaptation mode can be suitable for systems where the membership functions to be used are known in advance or where the implementation is constrained by the problem in some way. We compute $\frac{\partial E}{\partial w}$ for each weight and update the weight iteratively.

Error Function = $\frac{1}{2}\sum_{i=1}^{n}\left(\acute{Y} - Y\right)^2$

Ý- Expected Error

Y-Actual Error

Gradient Descent Algorithm

$w_{i+1} = w_i - \alpha\frac{\partial E}{\partial w}$

Chapter 4: Compression Algorithm

After the classification part we are left with millions of data points. But we only have Arduino device in the forest for prediction purposes. Arduino has about 2KB of SRAM and 32KB of flash memory. Practically it is impossible for Arduino to predict the results in no time with such large dataset. We need to compress the dataset to optimal number of prototypes so that all the required computations can be done on Arduino alone.

4.1 K-Means Algorithm

K-Means is used to group data points into clusters. We are using K-Means to generate fix number of prototypes for each class. K-Means algorithm uses iterative refinement to get a final result. The algorithm takes number of clusters K and dataset as input. It first generates K centroid randomly. Each data point is assigned to its nearest centroid based on Euclidian distance between centroid and data point. In the next step centroid are recomputed. This is done by taking mean of all data points belonging to the same cluster. Both of the above steps are executed simultaneously until we get the optimal results. This algorithm is guaranteed to converge to a result.

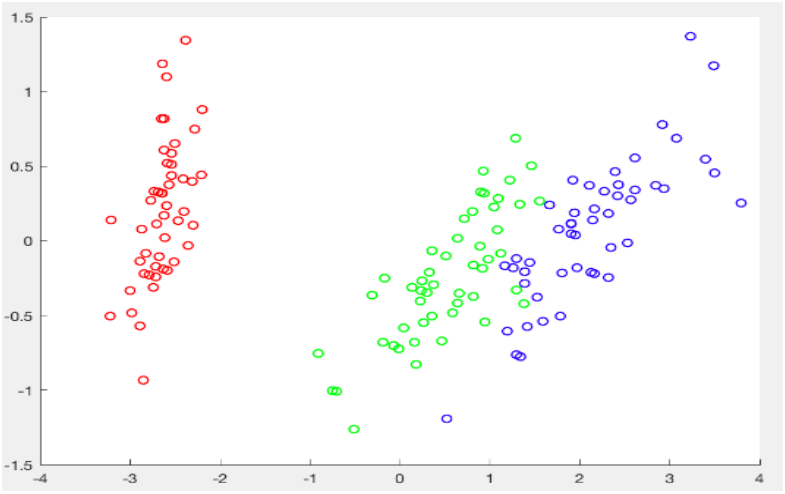

Figure 2: Data points of each class in the dataset

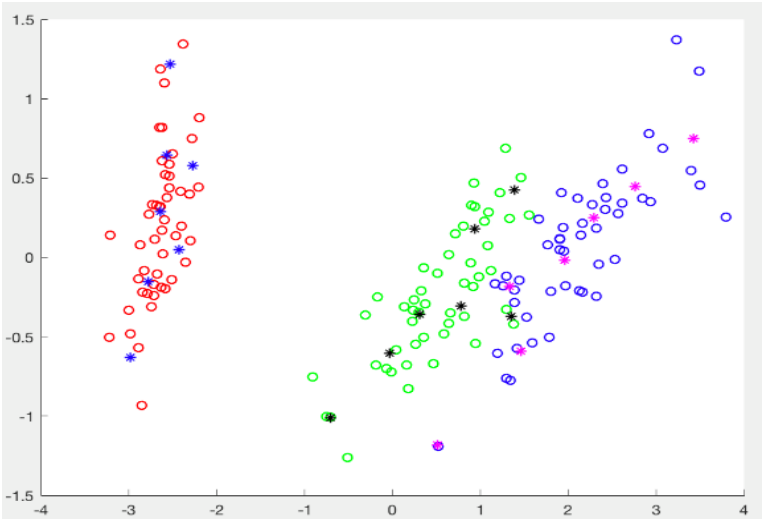

Figure 3: Prototypes generated using K-Means and data points of each class.

Data points with same color represents a class. There are 3 classes in the dataset. First picture represents the dataset. Second picture also includes the prototypes generated by K-Means algorithm.

4.2 Error Function

Observed probability- This is a Gibbs distribution for all the possible allocations of a pattern to one of n clusters defined by the prototypes. Here T represents temperature. It calculates the probability that a data point belongs to i_{th} cluster from the same class.

$$Z_i(X) = \frac{exp[-d_i^2/T]}{\sum_k exp[-d_k^2/T]}$$

Desired Probability- Here d_i represents the Euclidian distance between data point and prototype. Here C(X) represents the class of the cluster.

$$Z_i^*(X) = \begin{cases} 0, & \text{If } i \notin C(X) \\ \left(\frac{exp[-d_i^2/T]}{\sum_{k \in C(X)} exp[-d_k^2/T]}\right), & \text{Otherwise} \end{cases}$$

Error Function – When T decreases the probabilities turn out to be more biased towards the Euclidian distance between prototype and data points.

$$E(X) = \frac{1}{2} T \sum_i (Z_i^*(X) - Z_i(X))^2$$

4.3 Adam Optimiser

We are using Adam optimiser to reach the optimum minima of the error function. Adam is an efficient stochastic optimization technique. Adam iteratively updates the position of the prototypes until the error function converges. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

4.4 Deterministic Annealing

Annealing involves the gradual lowering of the temperature of a system. Annealing process starts at high T, for which all the probabilities of Y_i, are more or less similar. Same things can be said about desired probabilities $\acute{Y}_i$. Each data point thus influences all the prototypes of its class and the prototypes of other class in similar way. During training gradual decrease in T increases the bias in favour of the nearest prototype belonging to the correct class. Thus, at lower temperature we obtain a drastic classifier, and error function E measures the total error rate of classification on the training set. Therefore, this process distributes the prototypes in each class while minimising the total error rate of classification. Another property of annealing is that it helps to avoid local minima in function optimisation.

4.5 Inactive prototype elimination algorithm

Elimination algorithm is used to remove the inactive prototypes. After applying optimization algorithm some of the prototypes becomes inactive meaning we can maintain the same accuracy after removing them. This algorithm removes a number of prototypes from the dataset if their elimination does not decrease the consistency of the

resulting description on the training set. The principle of this algorithm is to iteratively remove a prototype from the competitive process, starting from smallest (in terms of error) to the largest ones. Here error represents the difference between accuracy before and after removing a particular prototype. We keep on removing the prototypes until error <= ε where ε parameter specifies the requested quality level of a decision to be taken. ε is to be chosen between 0 and 1.

4.6 Algorithm

1. Apply PCA to reduce the dimensionality of dataset.
2. Prototypes=KMeans(D,K)
3. Set temperature=2000
4. Set gradient Step at small value, momentum to 0.9, deltaP1=0, deltaP2=0, decay rate 1=0.999, decay rate 2=0.9, momentum=0.9,Error=0,PrevError=0
5. For every datapoint of dataset
   Zo = observedProbability(Dataset,Prototypes,temperature,datapoint)
   Zd = desiredProbability(Dataset,Prototypes,temperature,datapoint)
   Error = Error + deltaCostFunction(Dataset,Prototypes,Zo,Zd,datapoint)
   deltaE = deltaE + deltaCostFunction(Dataset,Prototypes,Zo,Zd,datapoint)
   Error = Error/(length(Dataset))
   (Prototypes,deltaP1,deltaP2) = Adam(Prototypes,deltaE,deltaP1,deltaP2,iteration)
   Set PrevError = Error, Error = 0, deltaE=0
   Iteration = iteration +1
6. Repeat step 5 until abs(Error-PrevError)>=10^{-4}.
7. temperature = temperature*0.9.
8. Repeat step 5,6,7 sequentially until temperature>20.
9. Prototypes = Elimination_algo(Prototypes,Dataset).

4.7 Benefits

Model described in the Chapter 2 is not efficient because we cannot guarantee high speed internet in forest. We have removed cloud engine from our model. We have improved the compression algorithm discussed in chapter 2. The position of the prototypes is also optimal as compared to the previous algorithm.

- All the real time computations are done on the IOT devices. IOT devices are much cheaper than cloud engine due to which our model is convenient to use.
- By optimizing the location of prototypes we can eliminate some of the prototypes thus we can use the storage of IOT devices more effectively.
- By reducing the number prototypes, we minimize the required computing power for our predictions.
- By optimizing the position of prototypes we improved the accuracy up to certain extent.

Chapter 5: Results

No. of datapoints	No. of prototypes taken initially	No. of reduced prototypes of christine's algorithm	Acc. of christine's algorithm	No. of reduced prototypes of our approach	Acc. of our algorithm
500	10	8	95.8%	7	96.2%
2000	16	13	96.5%	10	96.6%
5000	30	24	97%	15	97.3%
10000	40	30	96.6%	18	96.8%

Number of data points column represents the total number of data points in each dataset. Number of prototypes taken initially represents the total number of prototypes generated using K-Means algorithm. Our compression algorithm is inspired from christine's algorithm. Third column represents the total number of prototypes left after applying christine's algorithm. Fourth column represents the accuracy achieved by prototypes, after applying christine's algorithm. Fifth column represents the total number of prototypes left after using our algorithm. Similarly, last column represents the accuracy achieved after applying our algorithm.

Each of the dataset has 2 classes. The dimensionality of each data point of the dataset is 2.

Chapter 6: Conclusion and future work

In the field of nearest neighbor classifiers, a lot work has already been done since 90s. The work is also related to recent studies in Machine Learning. Some of the previous works are based on the selection of a subset of training set. But we are using prototype based approach which involves generation of prototypes instead of selection of subset of training set. This approach has already been implemented before but the stochastic optimization algorithm used was not optimal. We have used Adam optimizer which prove to be more efficient in this case from the results. By optimizing the location of prototypes we have eliminated some of the prototypes thus we can use the storage of IOT devices more effectively. By reducing the number prototypes, we have minimized the required computing power for our predictions. By optimizing the position of prototypes we also have improved the accuracy up to certain extent. We can reduce several thousands of data points to hundreds of prototypes while maintain comparable accuracy. Therefore, all the real time computations can be done using IOT devices. We will create a distributed network of IOT devices to solve the problem of forest fire prediction.

References

[1] Decaestecker C (1997) Finding prototypes for nearest neighbor classification by means of gradient descent and deterministic annealing. Pattern Recogn 30(2):281–288.

[2] Hinton, Geoffrey & Roweis, Sam. (2003). Stochastic Neighbor Embedding.

[3] Kusner, M.J., Tyree, S., Weinberger, K.Q., & Agrawal, K. (2014). Stochastic Neighbor Compression. ICML.

[4] Van de Merckt, Thierry. (1992). NFDT: A System that Learns Flexible Concepts Based on Decision Trees for Numerical Attributes.. 322-331.

[5] Diederik P. Kingma & Jimmy Ba (2014). Adam: A Method for Stochastic Optimization.

[6] http://deeplearning.stanford.edu/wiki/index.php/PCA.

[7] https://www.datascience.com/blog/k-means-clustering.

[8] https://en.wikipedia.org/wiki/Wildfire.

[9] http://www.scholarpedia.org/article/Fuzzy_neural_network.

[10] https://www.pcboard.ca/flame-sensor-module.

[11] https://www.sparkfun.com/products/10167.

[12] https://www.pololu.com/product/1482.

[13] Pratik M. Parekh, Dimitrios Katselis, Carolyn L. Beck & Srinivasa M. Salapaka (2015). Deterministic annealing for clustering: Tutorial and computational aspects. American Control Conference (ACC), , 2906-2911.

Appendix A:

- To record the dataset, we need to setup Arduino and sensors. We need to build a circuit and write a code in order to capture the dataset.
- Principal component analysis algorithm is used to reduce the dimensionality of the dataset. It finds the directions of maximum variance in high dimensional data and projects the dataset onto lower dimensional subspace while retaining most of the information.