

**Faculty of Engineering & Computer Science**

**Concordia Institute for Information Systems Engineering (CIISE)**

INSE 6180 (Security and Privacy Implications of Data Mining)

FINAL REPORT:

**A Fast Clustering Based Feature Subset Selection Algorithm**

**For High-Dimensional Data**

Submitted to:

**Professor Dr. Nizar Boughila**

**Submitted By**

**Student Name StudentId**

**Kiran Mayie bethi 40092284**

**Venkat Mani deep Chandana 40080924**

**Sahana Shankar 40092026**

**Abstract**

Feature selection involves identifying a subset of the most useful features that produces compatible results as the original entire set of features. A feature selection algorithm may be evaluated from both the efficiency and effectiveness points of view. While the efficiency concerns the time required to find a subset of features, the effectiveness is related to the quality of the subset of features. Based on these criteria, a fast clustering-based feature selection algorithm (FAST) is proposed and experimentally evaluated in this paper. The FAST algorithm works in two steps. In the first step, features are divided into clusters by using graph-theoretic clustering methods. In the second step, the most representative feature that is strongly related to target classes is selected from each cluster to form a subset of features. Features in different clusters are relatively independent, the clustering-based strategy of FAST has a high probability of producing a subset of useful and independent features. To ensure the efficiency of FAST, we adopt the efficient minimum-spanning tree (MST) clustering method. The efficiency and effectiveness of the FAST algorithm are evaluated through an empirical study. Extensive experiments are carried out to compare FAST and several representative feature selection algorithms, namely, ReliefF and CFS. The results, on 35 publicly available real-world high-dimensional image, microarray, and text data, demonstrate that the FAST produces smaller subsets of features.

### Keywords

* Relevant Feature
* Redundant Feature
* T-Relevance
* F-Correlation
* F-Redundancy
* R-featurers

Table of Contents

1. Introduction 3

1.1 Aim

1.2 Problem Identification

1.3 Objectives

3. Motivation 4

4.Literature Review………………………………………………………………………………………………………4

5.Proposed Model Frame Work……………..……………………………………………5

6. Implementation 4

6.1 Introductions 4

6.2 Dataset ………………………………4

6.3 Steps for project Implementation 13

7.Coding and Implementation 13

8.Result Analysis and Conclusion.. 16

9.Related Work………………………………….. 17

10.References……………………………...............……………………………………1

**Chapter -1**

**Introduction**

**1.1 INTRODUCTION TO PROJECT**

Feature subset selection can be viewed as the process of identifying and removing as many irrelevant and redundant features as possible. Our proposed FAST algorithm can effectively eliminate irrelevant features as well as handle redundant features.

Traditionally, feature subset selection research has focused on searching for relevant features. A well-known example is Relief which weighs each feature according to its ability to discriminate instances under different targets based on distance-based criteria function. However, Relief is ineffective at removing redundant features as two predictive but highly correlated features are likely both to be highly weighted. Relief-F extends Relief, enabling this method to work with noisy and incomplete data sets and to deal with multiclass problems, but still cannot identify redundant features.

**1.2 PROBLEM IDENTIFICATION**

The generality of the selected features is limited and the computational complexity is large. For the Filter methods the computational complexity is low, but the accuracy of the learning algorithms is not guaranteed.

The hybrid methods are a combination of filter and wrapper methods by using a filter method to reduce search space that will be considered by the subsequent wrapper. Wrapper methods are computationally expensive and they tend to over fit on small training sets. Of the many feature subset selection algorithms, some can effectively eliminate irrelevant features but fail to handle redundant features yet some of others can eliminate the irrelevant while taking care of the redundant features. Relief is ineffective at removing redundant features as two predictive but highly correlated features are likely both to be highly weighted. Relief-F extends Relief, enabling this method to work with noisy and incomplete data sets and to deal with multiclass problems, but still cannot identify redundant features.

**1.3 AIM**

Irrelevant features, along with redundant features, severely affect the accuracy of the learning machines. Thus, feature subset selection should be able to identify and remove as much of the irrelevant and redundant information as possible. Moreover, good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other. Hence keeping this in mind we should develop a novel algorithm that can effectively deal in handling both redundant and irrelevant features.

**1.4 Objectives**

Good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.

The efficiently and effectively deal with both irrelevant and redundant features, and obtain a good feature subset.

Generally all the three algorithms achieve significant reduction of dimensionality by selecting only a small portion of the original features.

The null hypothesis of the Friedman test is that all the feature selection algorithms are equivalent in terms of runtime.

**Chapter-2**

**Background and Literature Review**

This paper describes efficient methods for exact and approximate implementation of the MINFEATURES bias, which prefers consistent hypotheses definable over as few features as possible. This bias is useful for learning domains where many irrelevant features are present in the training data. We first introduce ReliefF, a new algorithm that exactly implements the MINFEATURES bias. This algorithm is empirically shown to be substantially faster than the FOCUS algorithm previously given in [ Almuallim and Dietterich, 1991 ] .

**2.1Related Work**

Feature subset selection can be viewed as the process of identifying and removing as many irrelevant and redundant features as possible. This is because 1) irrelevant features do not contribute to the predictive accuracy [1], and 2) redundant features do not redound to getting a better predictor for that they provide mostly information which is already present in other feature(s).

Of the many feature subset selection algorithms, some can effectively eliminate irrelevant features but fail to handle redundant features [2], [13], [3], [4], [5], [6], yet some of others can eliminate the irrelevant while taking care of the redundant features [7], [8], [9], [10]. Our proposed FAST algorithm falls into the second group. The below diagram represents working of the algorithm

Traditionally, feature subset selection research has focused on searching for relevant features. A well-known example is Relief [4], which weighs each feature according to its ability to discriminate instances under different targets based on distance-based criteria function. However, Relief is ineffective at removing redundant features as two predictive but highly correlated features are likely both to be highly weighted [11]. Relief-F [3] extends Relief, enabling this method to work with noisy and incomplete data sets and to deal with multiclass problems, but still cannot identify redundant features.

However, along with irrelevant features, redundant features also affect the speed and accuracy of learning algorithms, and thus should be eliminated as well [11], [12], [13]. CFS [08] are examples that take into consideration the redundant features. CFS [08] is achieved by the hypothesis that a good feature subset is one that contains features highly correlated with the target, yet uncorrelated with each other.. Different from these algorithms, our proposed the FAST algorithm employs the clustering-based method to choose features.

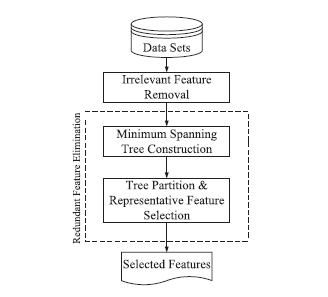
**Chapter -3**

**Proposed Model Frame Work**

**3.1 Framework and Definitions**

Irrelevant features, along with redundant features, severely affect the accuracy of the learning machines [13], [12]. Thus, feature subset selection should be able to identify and remove as much of the irrelevant and redundant information as possible. Moreover, “good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other.” [14]

Keeping these in mind, we develop a novel algorithm which can efficiently and effectively deal with both irrelevant and redundant features, and obtain a good feature subset. We achieve this through a new feature selection framework (shown in Fig. 1)



Architectural view of the project

Which composed of the two connected components of irrelevant feature removal and redundant feature elimination. The former obtains features relevant to the target concept by eliminating irrelevant ones, and the latter removes redundant features from relevant ones via choosing representatives from different feature clusters, and thus produces the final subset.

The irrelevant feature removal is straightforward once the right relevance measure is defined or selected, while the redundant feature elimination is a bit of sophisticated. In our proposed FAST algorithm, it involves 1) the construction of the minimum spanning tree from a weighted complete graph; 2) the partitioning of the MST into a forest with each tree representing a cluster; and 3) the selection of representative features from the clusters.

**3.2 Comparison of Three algorithms**

**Relief Algorithm**

Algorithm

Step 1:- for each training instance a vector of feature values and the class value

Step 2:- n ← number of training instances

Step 3:- a ← number of features (i.e. attributes)

Step 4:- Parameter: m ← number of random training instances out of n used to update W initialize all feature weights

Step 5:- W[A] := 0.0

Step 6:- for i :=1 to m do

Step 7:- randomly select a ‘target’ instance Ri

Step 8:- find a nearest hit ‘H’ and nearest miss ‘M’ (instances)

Step 9:- for A:= 1 to a do

Step 10:- W[A] := W[A] − (diff (A, Ri , H)/m) + (diff (A, Ri , M)/m)

Step 11:- end for

Step 12:- end For

Step 13:- return the vector W of feature scores that estimate the quality of features

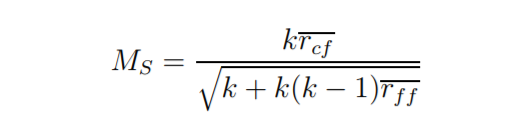
**Strengths**

* It estimates the quality of a given feature in the context of other features
* It makes no assumptions regarding the population distribution or sample size
* The efficiency of the algorithm has been attributed to the fact that it doesn’t explicitly explore feature subsets and because it does not bother trying to identify an optimal minimum feature subset size

**Limitations**

* the original Relief analysis suggests that the algorithm can be fooled by insufficient training cycles
* The original paper also suggests that Relief is fairly noise-tolerant and unaffected by feature interactions
* Relief has also been noted to have a reduced power to identify relevant non-monotonic features

**Correlation feature selection**

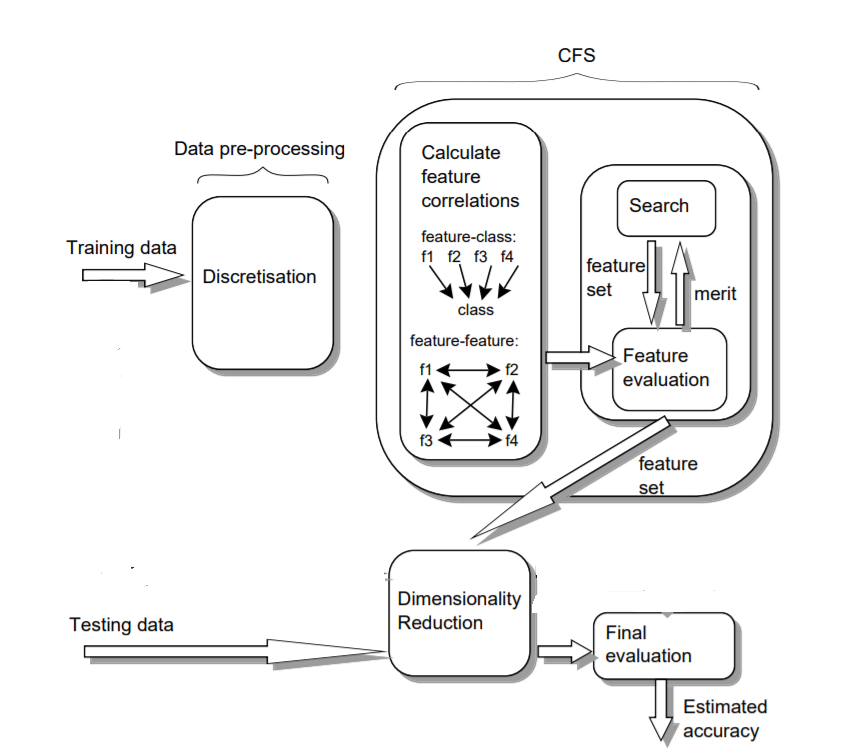
* CFS is a simple filter algorithm that ranks feature subsets according to a correlation based evaluation function.
* The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other.
* Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. Evaluation function is given by, 

Steps of CFS Algorithm:

Step1: Sample training data is first discretized using the method of Fayyad and Irani [FI93], then passed to CFS.

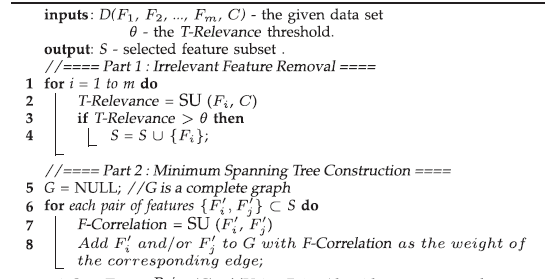
Step2: CFS calculates feature-class and feature-feature correlations and merits is calculated by evaluation function.

* The time complexity of CFS is quite low.
* It deals with small data set.



**Fast Algorithm:**

* In the First step, For a data set with features F1,F2,….Fi and class C, we compute the T-Relevance value for each feature in the first step. The features whose values are greater than a predefined threshold theta comprise the target-relevant feature subset. we heuristically set theta to be the ranked feature for each data set where m is the number of features
* In the second step, we first calculate the F-Correlation value for each pair of features. Then, viewing features as Fi and Fj and as vertices SU(Fi,Fj) and as the weight of the edge between vertices, aweighted complete graph is constructed. As symmetric uncertainty is symmetric further the F-Correlation is symmetric as well, thus is an undirected graph.



**Implementation**

**Data set:**

We have used the diabetics data set which have 5399 tuples and 44 features.

**Diabetic\_data.csv**

**Features:-**

encounter\_id,

patient\_nbr race

gender age

admission\_type\_id

discharge\_disposition\_id

admission\_source\_id

time\_in\_hospital

num\_lab\_procedures

num\_procedures

num\_medications

number\_outpatient

number\_emergency

number\_inpatient

number\_diagnoses

max\_glu\_serum

A1Cresult

Metformin

Repaglinide

Nateglinide

Chlorpropamide

Glimepiride

Acetohexamide

Glipizide

Glyburide

Tolbutamide

Pioglitazone

Rosiglitazone

Acarbose

Miglitol

Troglitazone

tolazamide

examide

citoglipton

insulin

glyburide-metformin

glipizide-metformin

glimepiride-pioglitazone

metformin-rosiglitazone

metformin-pioglitazone

change

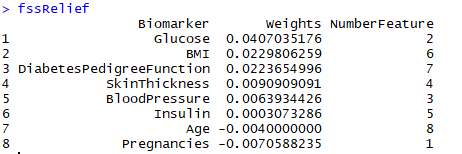
Y

Readmitted

Note: to represent data in CD we have converted all the data into text files

**Relief Algorithm:**

The relief algorithm allocates weights to each and every feature



The main disadvantage of ReliefF was that it was unable to load high dimensional data so we needed to use a low dimensional “Data-1Diabetes.csv”

**Cfs Algorithm**

The Cfs algorithm takes the consideration of only important features

"" "Glucose" "BMI" "Age"

**Fast algorithm**

**Irrelavantdata.csv**(given in CD)

It has the data of all the irrelevant data

**Featuretofeaturecorrelataion.csv**(given in CD)

It checks for feature and feature correlation and assign weights to each and every feature and based on that it has the data of the representative features

**Minimumspannningtree.csv**(given in CD)

Based on the data of feature and feature correlation table it removes all the edges and create a forest which is a representative cluster

**Fast.csv**

It has all the relevant data stored in it

**Code**

if(Fss\_Alg == "CFS")  
    {  
    # CFS  
    set.seed(3233)  
    [start.pt](http://start.pt/)<-proc.time()  
    start\_time<-Sys.time()  
      fss.cfs<-select.forward.Corr(data,disc.method="MDL",attrs.nominal=numeric())  
    cfs.data<-data[,c(fss.cfs)]  
    View(cfs.data)  
    train <- createDataPartition(y,p = 0.7, list = FALSE)  
    training.data <- cfs.data[train,]  
    training.class<-factor(data$Y[train])  
    testing.data <- cfs.data[-train,]  
    testing.class <- data$Y[-train]  
    actual<-sapply(testing.class,FUN = function(x){if(x==1){x<-1}else{x<-0}})  
    trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
    rsvm.train.model <- train(training.data,training.class,method = "svmRadial",  
                             trControl=trctrl,  
                             tuneLength = 10)  
    rsvm.predicted <- predict(rsvm.train.model,testing.data)  
    rsvm<-sapply(rsvm.predicted,FUN = function(x){if(x==1){x<-1}else{x<-0}})  
    cfs.rsvm.acc<-accuracy(actual,rsvm)  
    end\_time<-Sys.time()  
    configuration\_time<-end\_time-start\_time  
    print(configuration\_time)  
    return(cfs.rsvm.acc)  
  }  
  else if(Fss\_Alg == "ReliefF")  
    {  
    start\_time<-Sys.time()  
    if(ncol(data)>7000){  
      data1<-data[,1:2500]  
      fssRelief<-select.relief(data1)  
      data2<-data[,2501:ncol(data)]  
      fssRelief1<-select.relief(data2)  
    }else{  
      fssRelief<-select.relief(data)  
    }  
  
     selected.features<-fssRelief[,3]  
     Relief.rsvm.acc<-data.frame()  
     for(i in 1:7){  
       Relief.rsvm.acc[1,i]<-NA  
     }  
     Relief.rsvm.acc[]  
    for(i in 2:length(selected.features)){  
      Relief.data<-data[,c(selected.features[1:i])]  
      train <- createDataPartition(y,p = 0.7, list = FALSE)  
      training.data <- Relief.data[train,]  
      training.class<-factor(data$Y[train])  
      testing.data <- Relief.data[-train,]  
      testing.class <- data$Y[-train]  
      actual<-sapply(testing.class,FUN = function(x){if(x==1){x<-1}else{x<-0}})  
      trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
      rsvm.train.model <- train(training.data,training.class,method =  
"svmRadial",  
                               trControl=trctrl,  
                               tuneLength = 10)  
      rsvm.predicted <- predict(rsvm.train.model,testing.data)  
      rsvm<-sapply(rsvm.predicted,FUN = function(x){if(x==1){x<-1}else{x<-0}})  
      Relief.rsvm.acc<-accuracy(actual,rsvm)  
    }  
     end\_time<-Sys.time()  
     configuration\_time<-end\_time-start\_time  
       return(Relief.rsvm.acc)  
  
   }  
  
  else if(Fss\_Alg == "FAST")  
 {  
         ## initialization  
    x<-Dataset[,-ncol(Dataset)]  
     y<-Dataset$Y  
     attrs.nominal=ncol(Dataset)  
     Dataset[,ncol(Dataset)]<-as.factor(Dataset[,ncol(Dataset)])  
    target<-y  
    fs<-Dataset  
    is.normalized=TRUE  
    fss <- vector()  
    start.index = 1  
    start\_time<-Sys.time()  
      ## Extract a small grid from dataset  
    input.row.count = nrow(fs)  
    input.column.count = ncol(fs)  
    temp <- input.column.count  
    if (temp < floor(input.row.count/2)) {  
        Grid.size <- temp  
      }else {  
        Grid.size <- floor(input.row.count/2)  
        temp <- temp - Grid.size  
      }  
      stop.index <- start.index + Grid.size - 1  
  
      while (stop.index <= input.column.count)  
      {  
        if(stop.index > start.index)  
        {  
          Grid.data <- fs[, start.index:stop.index]  
          ## Remove NA values and replace with mean column value  
          if(anyNA(Grid.data))  
          {  
            data.withoutNA <- remove.NA(Grid.data)}else{  
              data.withoutNA <- Grid.data  
            }  
          ##Function to normalize data in 0-1 range  
          if(is.normalized){data.normalized<-data.withoutNA}else{  
            data.normalized <- (normlize.data(data.withoutNA))}  
          #CostMatrix <- abs(cor(as.numeric(data.normalized)))  
          CostMatrix <-  
abs(cor(data.normalized[sapply(data.normalized,function(x)!is.factor(x))]))  
          CostMatrix.row <- nrow(CostMatrix)  
          Graph.arcs <- matrix(ncol = 3)[-1, ]  
          for (i in 1:CostMatrix.row) {  
            for (j in 1:i) {  
              if (CostMatrix[i,j] != 0 & CostMatrix[i,j] != Inf &  
![is.na](http://is.na/)(CostMatrix[i,j])) {  
                Graph.arcs <-  
rbind(Graph.arcs,cbind(i+start.index-1,j+start.index-1,CostMatrix[i,j]))  
              }  
            }  
          }  
          nodes <- start.index:stop.index  
          ## Construct a Maximum Spanning Tree  
          tree.arcs<-msTreePrim(nodes,Graph.arcs,start.index)  
          ## removhe the arc with minimum cost  
          tree.arcs.remove <- which(tree.arcs[,3]<0.5)  
          if(length(tree.arcs.remove) != 0){  
            tree.arcs <- tree.arcs[-c(tree.arcs.remove), ]  
            ## Cluster the data  
            Clusters <- Clustering(tree.arcs)}else{  
              Clusters <- list()  
              Clusters[[1]] <- start.index:stop.index  
            }  
          ## Evaluate Weight Score for input subset  
          cluster.count <- length(Clusters)  
          if(cluster.count>0){  
  
              irrdata <-  
data.normalized[,sapply(c(Clusters[[i]]),FUN=function(x){x<-x-start.index+1;return(x)})]  
              Target <-  
target[sapply(c(Clusters[[i]]),FUN=function(x){x<-x-start.index+1;return(x)})]  
              #Run the weight function before executing next statement  
              Weight\_score <- round(Weight(data.normalized,Target),1)  
              feature.selected <- which(Weight\_score==Weight\_score)  
              #fss <- cbind(fss,Clusters[[i]][feature.selected])  
              fss<-append(fss,feature.selected, after = length(fss))  
  
          }  
  
  
        }  
      }  
     FAST.rsvm.acc<-data.frame()  
        for(i in 1:7){  
          FAST.rsvm.acc[1,i]<-NA  
        }  
     for(i in 1:length(feature.selected)){  
       FAST.data<-Dataset[,c(feature.selected[1:i])]  
     }  
    train <- createDataPartition(y,p = 0.7, list = FALSE)  
    training.data <- FAST.data[train,]  
    training.class<-factor(Dataset$Y[train])  
    testing.data <- FAST.data[-train,]  
    testing.class <- factor(Dataset$Y[-train])  
    actual<-sapply(testing.class,FUN = function(x){if(x==1){x<-1}else{x<-0}})  
    trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
      rsvm.train.model <- train(training.data,training.class,method =  
"svmRadial",  
                               trControl=trctrl,  
                               tuneLength = 10)  
    rsvm.predicted <- predict(rsvm.train.model,testing.data)  
    rsvm<-sapply(rsvm.predicted,FUN = function(x){if(x==1){x<-1}else{x<-0}})  
    [FAST.rsvm.cm](http://fast.rsvm.cm/)<-confusion.matrix(actual,rsvm)  
    FAST.rsvm.acc<-accuracy(actual,rsvm)  
    end\_time<-Sys.time()  
    configuration\_time<-end\_time-start\_time  
    return(FAST.rsvm.acc)  
        }  
  }

**Contribution and Discussion**

Like many other feature selection algorithms, our proposed FAST also requires a parameter that is the threshold of feature relevance. Different values might end with different classification results.

In order to explore which parameter value results in the best classification accuracy for a specific classification problem with a given classifier, a 10-fold cross-validation strategy was employed to reveal how the classification accuracy is changing with value of the parameter

As per the paper we have implemented three algorithms to compare the accuracy of the algorithms for finding the representative features so each of us have taken up an algorithm and have implemented it according to the specifications in R programming.

.

**Conclusion**

In this paper, we have presented a novel clustering-based feature subset selection algorithm for high dimensional data. The algorithm involves 1) removing irrelevant features, 2) constructing a minimum spanning tree from relative ones. Each cluster is treated as a single feature and thus dimensionality is drastically reduced.

We have compared the performance of the proposed algorithm with those of the five well-known feature selection algorithms ReliefF, CFS and FAST on the 2 publicly available text data the proposed algorithm obtained the best proportion of selected features, the best runtime, and the best classification accuracy.

**Microarray data**

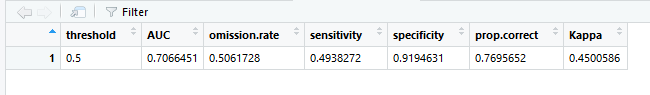
The proportion of selected features has been improved by each of the six algorithms compared with that on the given data sets. This indicates that the six algorithms work well with microarray data. FAST ranks 1 again with the proportion of selected features of 0.71 percent. Of the six algorithms, only CFS cannot choose features.

**Irrelevant feature**

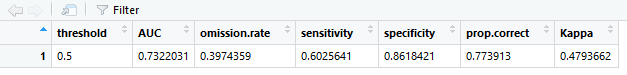
The irrelevant feature removal is straightforward once the right relevance measure is defined or selected, while the redundant feature elimination is a bit of sophisticated. In our proposed FAST algorithm, it involves 1.the construction of the minimum spanning tree from a weighted complete graph; 2. The partitioning of the MST into a forest with each tree representing a cluster; and 3.the selection of representative features from the cluster.

**Difference Between Cfs vs Relief vs Fast**

Relief Algorithm



CFS Algorithm



Fast Algorithm



References

[1]G.H. John, R. Kohavi, and K. Pfleger, “Irrelevant Features and the Subset Selection Problem,” Proc. 11th Int’l Conf. Machine Learning,

pp. 121-129, 1994.

[2] G. Forman, “An Extensive Empirical Study of Feature Selection Metrics for Text Classification,” J. Machine Learning Research, vol. 3, pp. 1289-1305, 2003.

[3] I. Kononenko, “Estimating Attributes: Analysis and Extensions of RELIEF,” Proc. European Conf. Machine Learning, pp. 171-182, 1994.

[4] K. Kira and L.A. Rendell, “The Feature Selection Problem: Traditional Methods and a New Algorithm,” Proc. 10th Nat’l Conf. Artificial Intelligence, pp. 129-134, 1992.

[5] M. Modrzejewski, “Feature Selection Using Rough Sets Theory,” Proc. European Conf. Machine Learning, pp. 213-226, 1993.

[6] M. Scherf and W. Brauer, “Feature Selection by Means of a Feature Weighting Approach,” Technical Report FKI-221-97, Institut fur Informatik, Technische Universitat Munchen, 1997.

[7] R. Battiti, “Using Mutual Information for Selecting Features in Supervised Neural Net Learning,” IEEE Trans. Neural Networks, vol. 5, no. 4, pp. 537-550, July 1994.

[8] M.A. Hall, “Correlation-Based Feature Subset Selection for Machine Learning,” PhD dissertation, Univ. of Waikato, 1999.

[9] H. Liu and R. Setiono, “A Probabilistic Approach to Feature Selection: A Filter Solution,” Proc. 13th Int’l Conf. Machine Learning, pp. 319-327, 1996.

[10] L. Yu and H. Liu, “Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution,” Proc. 20th Int’l Conf. Machine Leaning, vol. 20, no. 2, pp. 856-863, 2003.

[11] D. Koller and M. Sahami, “Toward Optimal Feature Selection,” Proc. Int’l Conf. Machine Learning, pp. 284-292, 1996.

[12] R. Kohavi and G.H. John, “Wrappers for Feature Subset Selection,” Artificial Intelligence, vol. 97, nos. 1/2, pp. 273-324, 1997

[13] M.A. Hall, “Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning,” Proc. 17th Int’l Conf. Machine Learning, pp. 359-366, 2000.

[14] M.A. Hall and L.A. Smith, “Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper,” Proc. 12th Int’l Florida Artificial Intelligence Research Soc. Conf., pp. 235-239, 1999.