CHAPTER 1

1 INTRODUCTION

1.1 INTRODUCTION

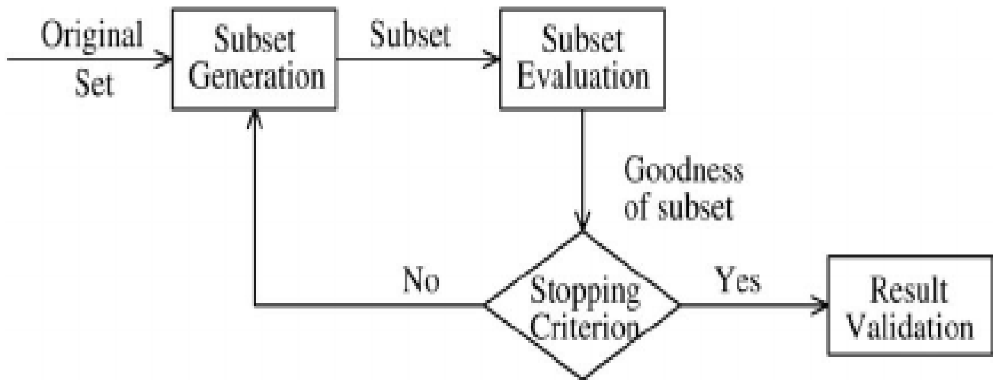
A feature selection algorithm can be seen as the combination of a search technique for proposing new feature subsets, along with an evaluation measure which scores the different feature subsets. The simplest algorithm is to test each possible subset of features finding the one which minimizes the error rate. This is an exhaustive search of the space, and is computationally intractable for all but the smallest of feature sets. The choice of evaluation metric heavily influences the algorithm, and it is these evaluation metrics which distinguish between the three main categories of feature selection algorithms: wrappers, filters and embedded methods.

- Wrapper methods use a predictive model to score feature subsets. Each new subset is used to train a model, which is tested on a hold-out set. Counting the number of mistakes made on that hold-out set (the error rate of the model) gives the score for that subset. As wrapper methods train a new model for each subset, they are very computationally intensive, but usually provide the best performing feature set for that particular type of model.
- Filter methods use a proxy measure instead of the error rate to score a feature subset. This measure is chosen to be fast to compute, while still capturing the usefulness of the feature set. Common measures include the mutual information, the pointwise mutual information, Pearson product-moment correlation coefficient, Relief-based algorithms, and inter/intra class distance or the scores of significance tests for each class/feature combinations. Filters are usually less computationally intensive than wrappers, but they produce a feature set which is not tuned to a specific type of predictive model. This lack of tuning means a feature set from a filter is more general than the set from a wrapper, usually giving lower prediction performance than a wrapper. However, the feature set doesn't contain the assumptions of a prediction model, and so is more useful for exposing the relationships between the features. Many filters provide a feature ranking rather than an explicit best feature subset, and the cut-off point in the ranking is chosen via cross-validation. Filter methods have also been used as a pre-processing step for wrapper methods, allowing a wrapper to be used on larger problems. One other popular

approach is the Recursive Feature Elimination algorithm, commonly used with Support Vector Machines to repeatedly construct a model and remove features with low weights.

- Embedded methods are a catch-all group of techniques which perform feature selection as part of the model construction process. The exemplar of this approach is the LASSO method for constructing a linear model, which penalizes the regression coefficients with an L1 penalty, shrinking many of them to zero. Any features which have non-zero regression coefficients are 'selected' by the LASSO algorithm. Improvements to the LASSO include Bolasso which bootstraps samples, Elastic net regularization, which combines the L1 penalty of LASSO with the L2 penalty of Ridge Regression; and FeaLect which scores all the features based on combinatorial analysis of regression coefficients. These approaches tend to be between filters and wrappers in terms of computational complexity.

In traditional regression analysis, the most popular form of feature selection is stepwise regression, which is a wrapper technique. It is a greedy algorithm that adds the best feature (or deletes the worst feature) at each round. The main control issue is deciding when to stop the algorithm. In machine learning, this is typically done by cross-validation. In statistics, some criteria are optimized. This leads to the inherent problem of nesting. More robust methods have been explored, such as branch and bound and piecewise linear network.



1.2 SUBSET SELECTION

Subset selection evaluates a subset of features as a group for suitability. Subset selection algorithms can be broken up into Wrappers, Filters and Embedded. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded techniques are embedded in and specific to a model.

Many popular search approaches use greedy hill climbing, which iteratively evaluates a candidate subset of features, then modifies the subset and evaluates if the new subset is an improvement over the old. Evaluation of the subsets requires a scoring metric that grades a subset of features. Exhaustive search is generally impractical, so at some implementor (or operator) defined stopping point, the subset of features with the highest score discovered up to that point is selected as the satisfactory feature subset. The stopping criterion varies by algorithm; possible criteria include: a subset score exceeds a threshold, a program's maximum allowed run time has been surpassed, etc.

Alternative search-based techniques are based on targeted projection pursuit which finds low-dimensional projections of the data that score highly: the features that have the largest projections in the lower-dimensional space are then selected.

Search approaches include:

- Exhaustive
- Best first
- Simulated annealing
- Genetic algorithm
- Greedy forward selection
- Greedy backward elimination
- Particle swarm optimization
- Targeted projection pursuit
- Scatter Search
- Variable Neighbourhood Search

Two popular filter metrics for classification problems are correlation and mutual information, although neither are true metrics or 'distance measures' in the mathematical sense, since they fail to obey the triangle inequality and thus do not compute any actual 'distance' – they should rather be regarded as 'scores'. These scores are computed between a candidate feature (or set of features) and the desired output category. There are, however, true metrics that are a simple function of the mutual information.

Other available filter metrics include:

- Class separability
    - Error probability
    - Inter-class distance
    - Probabilistic distance
    - Entropy
- Consistency-based feature selection
- Correlation-based feature selection

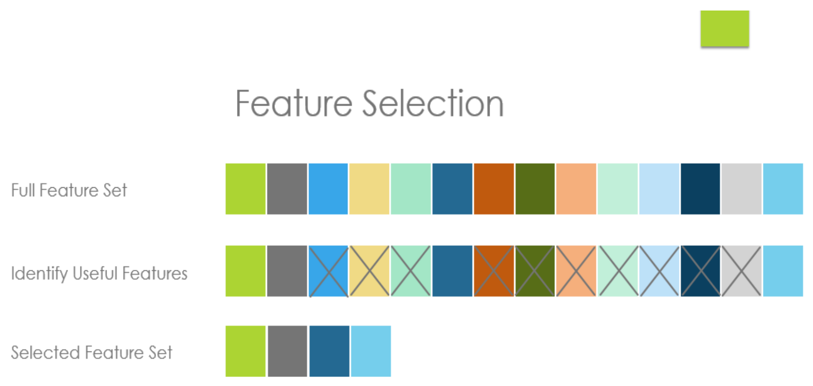
1.3 LITERATURE SURVEY

Feature selection is a problem pervasive in all domains of application of machine learning and data mining: engineering applications, robotics and pattern recognition (speech, handwriting, face recognition), Internet applications (text categorization), econometrics and marketing applications and medical applications (diagnosis, prognosis, drug discovery). Restricting the input space to a (small) subset of available input variables has obvious economic benefits in terms of data storage, computational requirements, and cost of future data collection. It also provides better data or model understanding and even better prediction performance. The process of non-trivial extraction of relevant information that is implicit in the data is known as knowledge discovery in databases, where the data mining phase plays a central role in this process. It has been noted that when large databases are going to get mined, the mining algorithms get very slow, requiring too much time to process the information and sometimes making the problem intractable. One way to attack this problem is to reduce the amount of data before applying the mining process. In particular, the pre-processing method of feature selection applied to the database before mining has shown to be successful, because it eliminates the irrelevant or redundant attributes that cause the mining tools to become inefficient, but preserving the classification quality of the mining algorithm. Feature selection is a important aspect of data preparation. It is also called variable or attributes selection. The data mining modeler need to walk a fine line here. It is very important to limit the number of features going into the classification model to make it less computationally costly, however in the same time the model need to have enough inputs to create good predictions.

The raw data are usually with inferior qualities such as irrelevancy, redundancy and unreliability. Mining raw data for a given task has proved to be deteriorating in efficiency. Algorithm for data mining can be used in dealing with redundant, irrelevant, or missing data. Before the mining process begins, those instances with missing value are considered insignificant to the result and eliminated from the dataset. The most challenging thing is to find out and eliminate redundant as well irrelevant features. As a part of the data preprocessing procedure, feature selection uses certain algorithms to keep only useful features and to remove irrelevant and noisy information, at the same time improving efficiency without significantly reducing the accuracy of the classifier.

Many factors affect the success of machine learning on a given task. The representation and quality of the example data is first and foremost. Nowadays, the need to process large databases is becoming increasingly common. The majority of real-world classification problems require supervised learning where the underlying class probabilities and class-conditionals probabilities are unknown, and each instance is associated with a class label. In real-world situations, relevant features are often unknown a priori. Therefore, many candidate features are introduced to better represent the domain. Theoretically, having more features should result in more discriminating power.

However, practical experience with machine learning algorithms has shown that this is not always the case, current machine learning toolkits are insufficiently equipped to deal with contemporary datasets and many algorithms are susceptible to exhibit poor complexity with respect to the number of features. Furthermore, when faced with many noisy features, some algorithms take an inordinately long time to converge or never converge at all. And even if they do converge conventional algorithms will tend to construct poor classifiers. Many of the introduced features during the training of a classifier are either partially or completely irrelevant/redundant to the target concept; an irrelevant feature does not affect the target concept in any way, and a redundant feature does not add anything new to the target concept. In many applications, the size of a dataset is so large that learning might not work as well before removing these unwanted features.



Recent research has shown that common machine learning algorithms are adversely affected by irrelevant and redundant training information. The simple nearest neighbor algorithm is sensitive to irrelevant attributes, its sample complexity (number of training examples needed to reach a given accuracy level) grows exponentially with the number of irrelevant attributes. Sample complexity for decision tree algorithms can grow exponentially on some concepts (such as parity) as well. The naive Bayes classifier can be adversely affected by redundant attributes due to its assumption that attributes are independent given the class. Decision tree algorithms such as C4.5 can sometimes over fit training data, resulting in large trees. In many cases, removing irrelevant and redundant information can result in C4.5 producing smaller trees. Neural Networks are supposed to cope with irrelevant and redundant features when the amount of training data is enough to compensate this drawback; otherwise they are also affected by the amount of irrelevant information.

Reducing the number of irrelevant/redundant features drastically reduces the running time of a learning algorithm and yields a more general concept. This helps in getting a better insight into the underlying concept of a real-world classification problem. Feature selection methods try to pick up a subset of features that are relevant to the target concept. The necessity of feature selection comes from the famous paradox in pattern classification. The paradox is as follows, when three features are considered, the accuracy of classification should be better than considering only two features. The inclusion of additional features leads to lower rather than higher accuracy. A very large set of features is present. It is difficult to design a classifier because the accuracy can be poor and instable. The presence of huge numbers of features often creates many disturbances for performance of inducing algorithms. In particular, the mix with irrelevant or redundant or noisy features will degrade even more, the classifier's performance accuracy. Suppose that the performance of a subset {f2, f4} is 90%. The goal of the feature selection problem is to find this subset.

Feature subset selection issue is viewed as a dimension reduction problem. In dimension reduction problems, boundary samples (records) take essential part in influencing the results. Two-dimensional feature spaces is reduced to one dimensional feature space. Reduced dimension will enhance the entropy of data because some information will be lost while reducing the dimension. In this way, this would decrease the classification accuracy. In dimension reduction problem, each feature might have incorrectly classified samples. Consequently, an optimal feature subset is a set of related features. It means that other features could correctly classify the samples incorrectly classified by one feature. Boundary samples are incorrectly classified samples of features. So, feature subset selection should be the center of attention on boundary samples. An optimal feature subset is a set of correlated features. It means that the samples incorrectly classified by a feature could be correctly classified by other features. Boundary samples are incorrectly classified samples of features and focus on the feature subset selection.

1.4 Theoretical and Empirical Analysis of Relief and Rrelief:

Relief algorithms are general and successful attribute estimators. They are able to detect conditional dependencies between attributes and provide a unified view on the attribute estimation in regression and classification. In addition, their quality estimates have a natural interpretation. While they have commonly been viewed as feature subset selection methods that are applied in prepossessing step before a model is learned, they have actually been used successfully in a variety of settings, e.g., to select splits or to guide constructive induction in the building phase of decision or regression tree learning, as the attribute weighting method and also in the inductive logic programming. A broad spectrum of successful uses calls for especially careful investigation of various features Relief algorithms have. In this paper we theoretically and empirically investigate and discuss how and why they work, their theoretical and practical properties, their parameters, what kind of dependencies they detect, how do they scale up to large number of examples and features, how to sample data for them, how robust are they regarding the noise, how irrelevant and redundant attributes influence their output and how different metrics influences them.

A problem of estimating the quality of attributes (features) is an important issue in the machine learning. There are several important tasks in the process of machine learning e.g., feature subset selection, constructive induction, decision and regression tree building, which contain the attribute estimation procedure as their (crucial) ingredient. In many learning problems there are hundreds or thousands of potential features describing each input object. Majority of learning methods do not behave well in these circumstances because, from a statistical point of view, examples with many irrelevant, but noisy, features provide very little information. A feature subset selection is a task of choosing a small subset of features that ideally is necessary and sufficient to describe the target concept. To make a decision which features to retain and which to discard we need a reliable and practically efficient method of estimating their relevance to the target concept. In the constructive induction we face a similar problem. In order to enhance the power of the representation language and construct a new knowledge we introduce new features. Typically, many candidate features are generated and again we need to decide which features to retain and which to discard. To estimate the relevance of the features to the target concept is certainly one of the major components of such a decision procedure. Decision and regression trees are popular description languages for representing knowledge in the machine learning. While constructing a tree the learning algorithm at each interior node selects the splitting rule (feature) which divides the problem space into two separate subspaces.

To select an appropriate splitting rule the learning algorithm has to evaluate several possibilities and decide which would partition the given (sub)problem most appropriately. The estimation of the quality of the splitting rules seems to be of the principal importance. The problem of feature (attribute) estimation has received much attention in the literature. There are several measures for estimating attributes' quality. If the target concept is a discrete variable (the classification problem) these are e.g., information gain (Hunt et al., 1966), Gini index (Breiman et al., 1984), distance measure (Mantaras, 1989), j-measure (Smyth and Goodman, 1990), Relief (Kira and Rendell, 1992b), Relief (Kononenko, 1994), MDL (Kononenko, 1995), and also χ 2 and G statistics are used. If the target concept is presented as a real valued function (numeric class and the regression problem) then the estimation heuristics are e.g., the mean squared and the mean absolute error (Breiman et al., 1984), and RRelief (RobnikSikonja ˇ and Kononenko, 1997). The majority of the heuristic measures for estimating the quality of the attributes assume the conditional (upon the target variable) independence of the attributes and are therefore less appropriate in problems which possibly involve much feature interaction. Relief algorithms (Relief, Relief and RRelief) do not make this assumption. They are efficient, aware of the contextual information, and can correctly estimate the quality of attributes in problems with strong dependencies between attributes. While Relief algorithms have commonly been viewed as feature subset selection methods that are applied in a prepossessing step before the model is learned (Kira and Rendell, 1992b) and are one of the most successful preprocessing algorithms to date (Dietterich, 1997), they are actually general feature estimators and have been used successfully in a variety of settings: to select splits in the building phase of decision tree learning (Kononenko et al., 1997), to select splits and guide the constructive induction in learning of the regression trees (RobnikSikonja and Kononenko, 1997), as attribute ˇ weighting method (Wettschereck et al., 1997) and also in inductive logic programming (Pompe and Kononenko, 1995). The RELIEF algorithm is a popular approach for feature weight estimation. Many extensions of the RELIEF algorithm are developed. However, an essential defect in the original RELIEF algorithm has been ignored for years. Because of the randomicity and the uncertainty of the instances used for calculating the feature weight vector in the RELEIF algorithm, the results will fluctuate with the instances, which lead to poor evaluation accuracy.

To solve this problem, a novel feature selection algorithm based on Mean-Variance model is proposed. It takes both the mean and the variance of the discrimination among instances into account as the criterion of feature weight estimation, which makes the result more stable and accurate. Based on real seismic signals of ground targets, experiment results indicate that the subsets of feature generated by proposed algorithm have better performance. Feature selection is an important issue in pattern recognition and machine learning which helps us to focus the attention of a classification algorithm on those features that are the most relevant to predict the class. Theoretically, if the full statistical distribution were known, using more features could improve results. However, in practical a large number of features as the input of induction algorithms may turn them inefficient as memory and time consumers. Besides, irrelevant features may confuse algorithms leading to reach false conclusions, and hence producing even worse results.

So, it is of fundamental importance to select the relevant and necessary features in the preprocessing step. Obviously, the advantages of using feature selection may be improving understandability and lowering cost of data acquisition and handling. Because of all these advantages, feature selection has attracted much attention within the Machine Learning, Artificial Intelligent and Data Mining communities.

In order to solve the problem that results of feature weight selection fluctuated with trained instances in the original RELIEF algorithm, a novel feature selection algorithm based on MeanVariance model is proposed, from which both the mean and variance of the discrimination among instances are considered.

With the mean-variance model, the feature weight estimation method is revised according to the original RELEIF algorithm. Finally, based on real seismic signals of ground targets, experiment results indicate that the subsets of feature generated by proposed algorithm have better performance in target recognition.

CHAPTER 2

2 SOFTWARE AND HARDWARE REQUIREMENTS

2.1 EXISTING SYSTEM

Variable and feature selection have become the focus of much research in areas of application for which datasets with tens or hundreds of thousands of variables are available. These areas include text processing of internet documents, gene expression array analysis, and combinatorial chemistry. The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underling process that generated the data. The contributions of this special issue cover a wide range of aspects of such problems: providing a better definition of the objective function, feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods.

As of 1997, when a special issue on relevance including several papers on variable and feature selection was published (Blum and Langley, 1997, Kohavi and John, 1997), few domains explored used more than 40 features. The situation has changed considerably in the past few years and, in this special issue, most papers explore domains with hundreds to tens of thousands of variables or features New techniques are proposed to address these challenging tasks involving many irrelevant and redundant variables and often comparably few training examples. Two examples are typical of the new application domains and serve us as illustration throughout this introduction. One is gene selection from microarray data and the other is text categorization. In the gene selection problem, the variables are gene expression coefficients corresponding to the abundance of mRNA in a sample (e.g. tissue biopsy), for a number of patients.

A typical classification task is to separate healthy patients from cancer patients, based on their gene expression “profile”. Usually fewer than 100 examples (patients) are available altogether for training and testing. But, the number of variables in the raw data ranges from 6000 to 60,000. Some initial filtering usually brings the number of variables to a few thousand. Because the abundance of mRNA varies by several orders of magnitude depending on the gene, the variables are usually standardized.

In the text classification problem, the documents are represented by a "bag-of-words", that is a vector of dimension the size of the vocabulary containing word frequency counts (proper normalization of the variables also apply). Vocabularies of hundreds of thousands of words are common, but an initial pruning of the most and least frequent words may reduce the effective number of words to 15,000. Large document collections of 5000 to 800,000 documents are available for research. Typical tasks include the automatic sorting of URLs into a web directory and the detection of unsolicited email (spam).

There are many potential benefits of variable and feature selection: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance. Some methods put more emphasis on one aspect than another, and this is another point of distinction between this special issue and previous work. The papers in this issue focus mainly on constructing and selecting subsets of features that are useful to build a good predictor. This contrasts with the problem of finding or ranking all potentially relevant variables. Selecting the most relevant variables is usually suboptimal for building a predictor, particularly if the variables are redundant. Conversely, a subset of useful variables may exclude many redundant, but relevant, variables.

The recent developments in variable and feature selection have addressed the problem from the pragmatic point of view of improving the performance of predictors. They have met the challenge of operating on input spaces of several thousand variables. Sophisticated wrapper or embedded methods improve predictor performance compared to simpler variable ranking methods like correlation methods, but the improvements are not always significant: domains with large numbers of input variables suffer from the curse of dimensionality and multivariate methods may overfit the data.

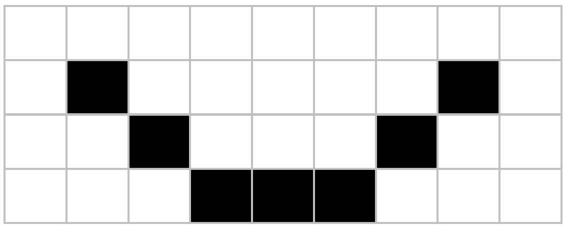
For some domains, applying first a method of automatic feature construction yields improved performance and a more compact set of features. The methods proposed in this special issue have been tested on a wide variety of data sets , which limits the possibility of making comparisons across papers. Further work includes the organization of a benchmark. The approaches are very diverse and motivated by various theoretical arguments, but a unifying theoretical framework is lacking. Because of these shortcomings, it is important when starting with a new problem to have a few baseline performance values. To that end, we recommend using a linear predictor of your choice (e.g. a linear SVM) and select variables in two alternate ways: (1) with a variable ranking method using a correlation coefficient or mutual information; (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates. Further down the road, connections need to be made between the problems of variable and feature selection and those of experimental design and active learning, in an effort to move away from observational data toward experimental data, and to address problems of causality inference.

2.2 PROPOSED SYSTEM

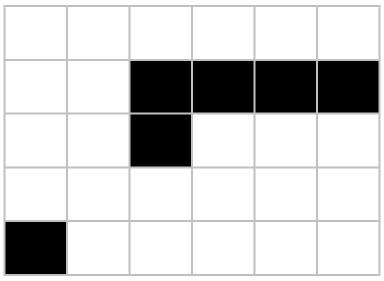
Contextual image classification, a topic of pattern recognition in computer vision, is an approach of classification based on contextual information in images. "Contextual" means this approach is focusing on the relationship of the nearby pixels, which is also called neighborhood. The goal of this approach is to classify the images by using the contextual information.

Similar as processing language, a single word may have multiple meanings unless the context is provided, and the patterns within the sentences are the only informative segments we care about. For images, the principle is same. Find out the patterns and associate proper meanings to them.

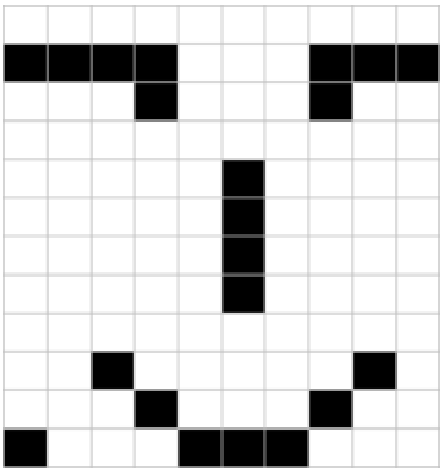
As the image illustrated below, if only a small portion of the image is shown, it is very difficult to tell what the image is about.



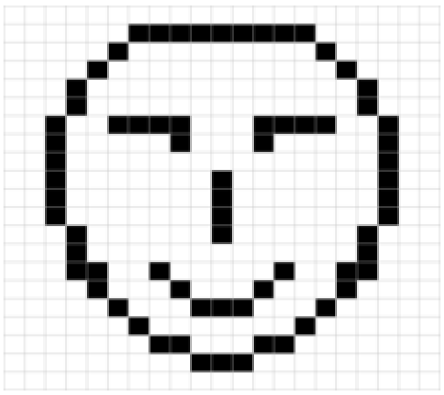
Even try another portion of the image, it is still difficult to classify the image.



However, if we increase the contextual of the image, then it makes more sense to recognize.



As the full images shows below, almost everyone can classify it easily.



Full Image

During the procedure of segmentation, the methods which do not use the contextual information are sensitive to noise and variations, thus the result of segmentation will contain a great deal of misclassified regions, and often these regions are small (e.g., one pixel).

Compared to other techniques, this approach is robust to noise and substantial variations for it takes the continuity of the segments into account.

Functioning as a post-processing filter to a labelled image

This approach is very effective against small regions caused by noise. And these small regions are usually formed by few pixels or one pixel. The most probable label is assigned to these regions. However, there is a drawback of this method. The small regions also can be formed by correct regions rather than noise, and in this case the method is actually making the classification worse. This approach is widely used in remote sensing applications.

Improving the post-processing classification

This is a two-stage classification process:

1. For each pixel, label the pixel and form a new feature vector for it.
2. Use the new feature vector and combine the contextual information to assign the final label to the

Merging the pixels in earlier stages

Instead of using single pixels, the neighbor pixels can be merged into homogeneous regions benefiting from contextual information. And provide these regions to classifier.

Acquiring pixel feature from neighborhood

The original spectral data can be enriched by adding the contextual information carried by the neighbor pixels, or even replaced in some occasions. This kind of pre-processing methods are widely used in textured image recognition. The typical approaches include mean values, variances, texture description, etc.

Combining spectral and spatial information

The classifier uses the grey level and pixel neighborhood (contextual information) to assign labels to pixels. In such case the information is a combination of spectral and spatial information.

Powered by the Bayes minimum error classifier

Contextual classification of image data is based on the Bayes minimum error classifier (also known as a naive Bayes classifier).

Present the pixel:

- A pixel is denoted as x_0.
- The neighbourhood of each pixel x_0 is a vector and denoted as $N(x_0)$.
  - The values in the neighbourhood vector is denoted as $f(x_i)$.
  - Each pixel is presented by the vector

    $\xi = (f(x_0), f(x_1), \ldots, f(x_k))$

    $x_i \in N(x_0); \quad i = 1, \ldots, k$
- The labels (classification) of pixels in the neighbourhood $N(x_0)$ are presented as a vector

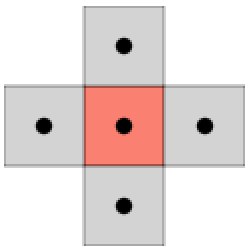
  $\eta = (\theta_0, \theta_1, \ldots, \theta_k)$

  $\theta_i \in \{\omega_0, \omega_1, \ldots, \omega_k\}$

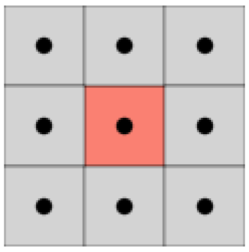
  ω_s here denotes the assigned class.
- A vector presents the labels in the neighbourhood $N(x_0)$ without the pixel x_0

  $\hat{\eta} = (\theta_1, \theta_2, \ldots, \theta_k)$

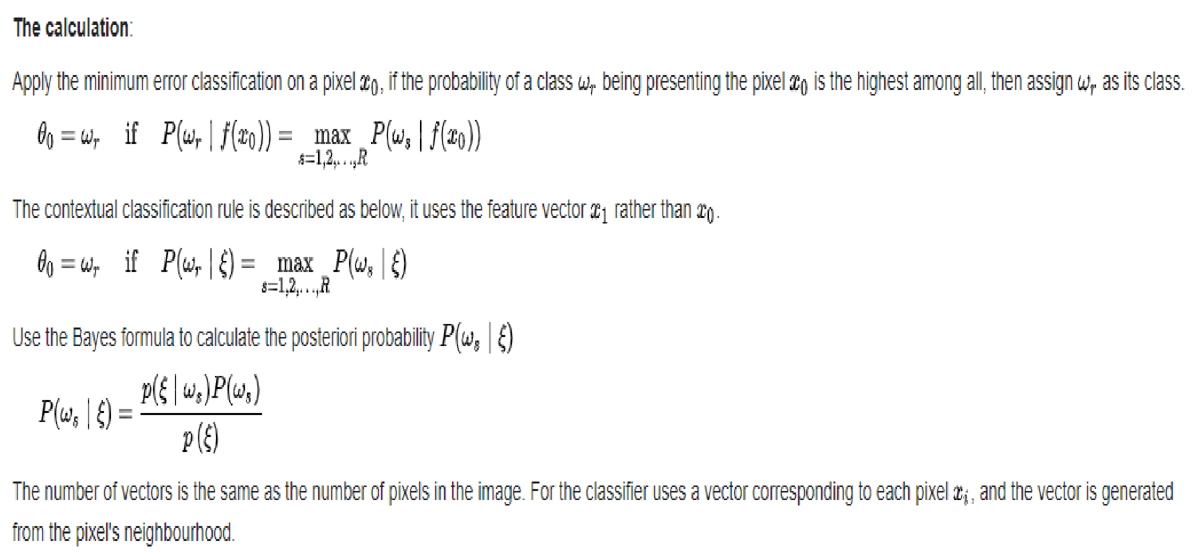
The neighborhood: Size of the neighborhood. There is no limitation of the size, but it is considered to be relatively small for each pixel . A reasonable size of neighborhood would be of 4-connectivity or 8-connectivity (is marked as red and placed in the centre).



4-connectivity neighborhood



8-connectivity neighborhood



The calculation:

Apply the minimum error classification on a pixel x_0, if the probability of a class ω_r being presenting the pixel x_0 is the highest among all, then assign ω_r as its class.

$$\theta_0 = \omega_r \quad \text{if} \quad P(\omega_r \mid f(x_0)) = \max_{s=1,2,\ldots,R} P(\omega_s \mid f(x_0))$$

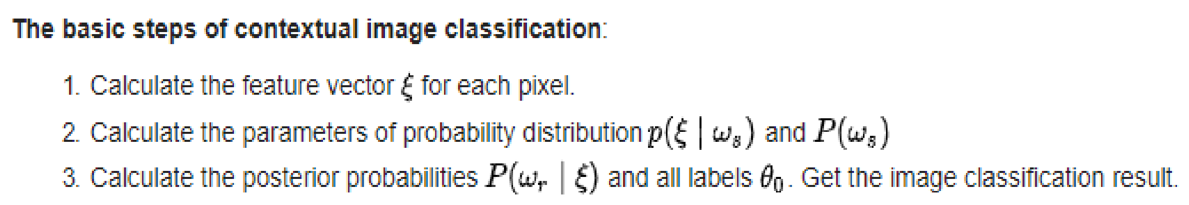
The contextual classification rule is described as below, it uses the feature vector x_1 rather than x_0.

$$\theta_0 = \omega_r \quad \text{if} \quad P(\omega_r \mid \xi) = \max_{s=1,2,\ldots,R} P(\omega_s \mid \xi)$$

Use the Bayes formula to calculate the posteriori probability $P(\omega_s \mid \xi)$

$$P(\omega_s \mid \xi) = \frac{p(\xi \mid \omega_s)P(\omega_s)}{p(\xi)}$$

The number of vectors is the same as the number of pixels in the image. For the classifier uses a vector corresponding to each pixel x_i, and the vector is generated from the pixel's neighbourhood.



The basic steps of contextual image classification:

1. Calculate the feature vector ξ for each pixel.
2. Calculate the parameters of probability distribution $p(\xi \mid \omega_s)$ and $P(\omega_s)$
3. Calculate the posterior probabilities $P(\omega_r \mid \xi)$ and all labels θ_0. Get the image classification result.

Template matching

The template matching is a "brute force" implementation of this approach. The concept is first creating a set of templates, and then look for small parts in the image match with a template.

This method is computationally high and inefficient. It keeps an entire templates list during the whole process and the number of combinations is extremely high. For a pixel image, there could be a maximum of combinations, which leads to high computation. This method is a top down method and often called table look-up or dictionary look-up.

Lower-order Markov chain

The Markov chain also can be applied in pattern recognition. The pixels in an image can be recognised as a set of random variables, then use the lower order Markov chain to find the relationship among the pixels. The image is treated as a virtual line, and the method uses conditional probability.

Hilbert space-filling curves

The Hilbert curve runs in a unique pattern through the whole image, it traverses every pixel without visiting any of them twice and keeps a continuous curve. It is fast and efficient.

Markov meshes

The lower-order Markov chain and Hilbert space-filling curves mentioned above are treating the image as a line structure. The Markov meshes however will take the two-dimensional information into account.

Dependency tree

The dependency tree is a method using tree dependency to approximate probability distributions.

Feature Selection:

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for four reasons:

- simplification of models to make them easier to interpret by researchers/users,
- shorter training times,
- to avoid the curse of dimensionality,
- enhanced generalization by reducing overfitting (formally, reduction of variance)

The central premise when using a feature selection technique is that the data contains some features that are either *redundant* or *irrelevant*, and can thus be removed without incurring much loss of information.*Redundant* and *irrelevant* are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated.

Feature selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points). Archetypal cases for the application of feature selection include the analysis of written texts and DNA microarray data, where there are many thousands of features, and a few tens to hundreds of samples.

A feature selection algorithm can be seen as the combination of a search technique for proposing new feature subsets, along with an evaluation measure which scores the different feature subsets. The simplest algorithm is to test each possible subset of features finding the one which minimizes the error rate. This is an exhaustive search of the space, and is computationally intractable for all but the smallest of feature sets. The choice of evaluation metric heavily influences the algorithm, and it is these evaluation metrics which distinguish between the three main categories of feature selection algorithms: wrappers, filters and embedded methods.

- Wrapper methods use a predictive model to score feature subsets. Each new subset is used to train a model, which is tested on a hold-out set. Counting the number of mistakes made on that hold-out set (the error rate of the model) gives the score for that subset. As wrapper methods train a new model for each subset, they are very computationally intensive, but usually provide the best performing feature set for that particular type of model.
- Filter methods use a proxy measure instead of the error rate to score a feature subset. This measure is chosen to be fast to compute, while still capturing the usefulness of the feature set. Common measures include the mutual information, the pointwise mutual information, Pearson product-moment correlation coefficient, Relief-based algorithms, and inter/intra class distance or the scores of significance tests for each class/feature combinations. Filters are usually less computationally intensive than wrappers, but they produce a feature set which is not tuned to a specific type of predictive model. This lack of tuning means a feature set from a filter is more general than the set from a wrapper, usually giving lower prediction performance than a wrapper. However, the feature set doesn't contain the assumptions of a prediction model, and so is more useful for exposing the relationships between the features. Many filters provide a feature ranking rather than an explicit best feature subset, and the cutoff point in the ranking is chosen via cross-validation. Filter methods have also been used as a preprocessing step for wrapper methods, allowing a wrapper to be used on larger problems. One other popular approach is the Recursive Feature Elimination algorithm, commonly used with Support Vector Machines to repeatedly construct a model and remove features with low weights.

- Embedded methods are a catch-all group of techniques which perform feature selection as part of the model construction process. The exemplar of this approach is the LASSO method for constructing a linear model, which penalizes the regression coefficients with an L1 penalty, shrinking many of them to zero. Any features which have non-zero regression coefficients are 'selected' by the LASSO algorithm. Improvements to the LASSO include Bolasso which bootstraps samples,;[8]Elastic net regularization, which combines the L1 penalty of LASSO with the L2 penalty of Ridge Regression; and FeaLect which scores all the features based on combinatorial analysis of regression coefficients.[9] These approaches tend to be between filters and wrappers in terms of computational complexity.

In traditional regression analysis, the most popular form of feature selection is stepwise regression, which is a wrapper technique. It is a greedy algorithm that adds the best feature (or deletes the worst feature) at each round. The main control issue is deciding when to stop the algorithm. In machine learning, this is typically done by cross-validation. In statistics, some criteria are optimized. This leads to the inherent problem of nesting. More robust methods have been explored, such as branch and bound and piecewise linear network.

Subset selection evaluates a subset of features as a group for suitability. Subset selection algorithms can be broken up into Wrappers, Filters and Embedded. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded techniques are embedded in and specific to a model.

Many popular search approaches use greedy hill climbing, which iteratively evaluates a candidate subset of features, then modifies the subset and evaluates if the new subset is an improvement over the old. Evaluation of the subsets requires a scoring metric that grades a subset of features. Exhaustive search is generally impractical, so at some implementor (or operator) defined stopping point, the subset of features with the highest score discovered up to that point is selected as the satisfactory feature subset. The stopping criterion varies by algorithm; possible criteria include: a subset score exceeds a threshold, a program's maximum allowed run time has been surpassed, etc.

Alternative search-based techniques are based on targeted projection pursuit which finds low-dimensional projections of the data that score highly: the features that have the largest projections in the lower-dimensional space are then selected.

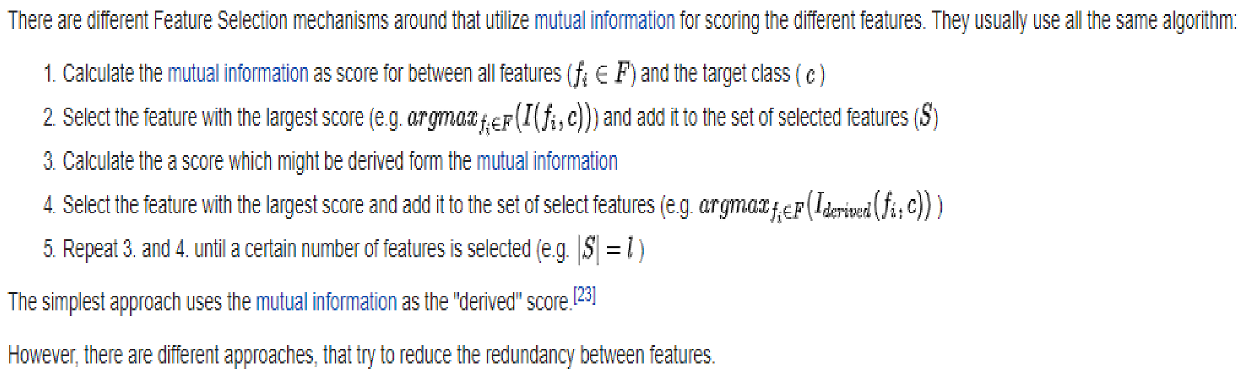
Search approaches include:

- Exhaustive
- Best first
- Simulated annealing
- Genetic algorithm
- Greedy forward selection
- Greedy backward elimination
- Particle swarm optimization
- Targeted projection pursuit
- Scatter Search
- Variable Neighborhood Search

Two popular filter metrics for classification problems are correlation and mutual information, although neither are true metrics or 'distance measures' in the mathematical sense, since they fail to obey the triangle inequality and thus do not compute any actual 'distance' – they should rather be regarded as 'scores'. These scores are computed between a candidate feature (or set of features) and the desired output category. There are, however, true metrics that are a simple function of the mutual information; see here.

Other available filter metrics include:

- Class separability
    - Error probability
    - Inter-class distance
    - Probabilistic distance
    - Entropy
- Consistency-based feature selection
- Correlation-based feature selection


There are different Feature Selection mechanisms around that utilize mutual information for scoring the different features. They usually use all the same algorithm:

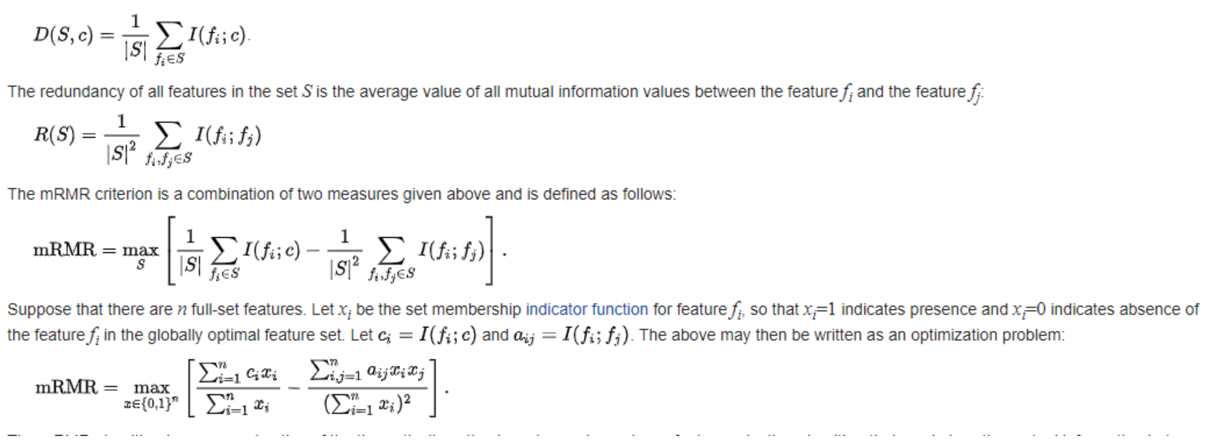
1. Calculate the mutual information as score for between all features ($f_i \in F$) and the target class (c)
2. Select the feature with the largest score (e.g. $argmax_{f_i \in F}(I(f_i, c))$) and add it to the set of selected features (S)
3. Calculate the a score which might be derived form the mutual information
4. Select the feature with the largest score and add it to the set of select features (e.g. $argmax_{f_i \in F}(I_{derived}(f_i, c))$)
5. Repeat 3. and 4. until a certain number of features is selected (e.g. $|S| = l$)

The simplest approach uses the mutual information as the "derived" score.[23]

However, there are different approaches, that try to reduce the redundancy between features.

Minimum-redundancy-maximum-relevance (mRMR) feature selection

Peng *et al.*[24] proposed a feature selection method that can use either mutual information, correlation, or distance/similarity scores to select features. The aim is to penalise a feature's relevancy by its redundancy in the presence of the other selected features. The relevance of a feature set *S* for the class *c* is defined by the average value of all mutual information values between the individual feature f_i and the class *c* as follows:


$$D(S,c) = \frac{1}{|S|}\sum_{f_i \in S} I(f_i;c).$$

The redundancy of all features in the set S is the average value of all mutual information values between the feature f_i and the feature f_j:

$$R(S) = \frac{1}{|S|^2}\sum_{f_i,f_j \in S} I(f_i;f_j)$$

The mRMR criterion is a combination of two measures given above and is defined as follows:

$$\mathrm{mRMR} = \max_S \left[\frac{1}{|S|}\sum_{f_i \in S} I(f_i;c) - \frac{1}{|S|^2}\sum_{f_i,f_j \in S} I(f_i;f_j)\right].$$

Suppose that there are n full-set features. Let x_i be the set membership indicator function for feature f_i, so that x_i=1 indicates presence and x_i=0 indicates absence of the feature f_i in the globally optimal feature set. Let $c_i = I(f_i;c)$ and $a_{ij} = I(f_i;f_j)$. The above may then be written as an optimization problem:

$$\mathrm{mRMR} = \max_{x \in \{0,1\}^n}\left[\frac{\sum_{i=1}^n c_i x_i}{\sum_{i=1}^n x_i} - \frac{\sum_{i,j=1}^n a_{ij} x_i x_j}{(\sum_{i=1}^n x_i)^2}\right].$$

2.3 PURPOSE OF THE PROJECT

When working with databases, there might be some amounts of data that is irrelevant from the data that we require. Feature selection helps by eliminating the data and taking into consideration only the data that is required by us.

A new hybrid model is proposed by combing reliefF algorithm and SVM-RFE method. The new method (we call it reliefF-SVM-RFE) not only performs better than either of the two methods but also costs much less computational time than the wrapper models.

2.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, and outputs. An image is given as an input and 8 images are obtained as output.

2.5 NON-FUNCTIONAL REQUIREMENTS

- EFFICIENCY

The feature selection process is more efficient in our model (by combining 2 algorithms) than using the feature selections algorithms individually.

- PERFORMANCE

Experimental results demonstrate that with the feature subset from our proposed reliefF-SVM-RFE method a better classification performance can be achieved.

2.6 SOFTWARE REQUIREMENTS

LANGUAGE: MATLAB 9.5

TECHNOLOGY: MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most uses of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M – files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

Following are the basic features of MATLAB −

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.

- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

The MATLAB system consists of five main parts

- **Development Environment**:

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and command window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

- **The MATLAB Mathematical Function Library**:

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms.

- **The MATLAB Language**:

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.

- **Graphics:**

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

- **The MATLAB Application Program Interface (API):**

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

Various toolboxes are there in MATLAB for computing recognition techniques, but we are using **IMAGE PROCESSING** toolbox.

GRAPHICAL USER INTERFACE (GUI):

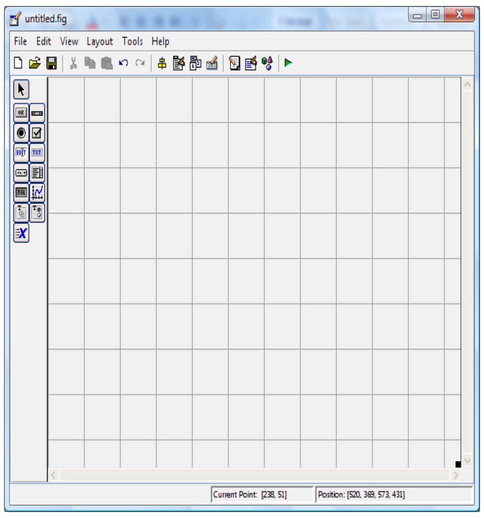
MATLAB's Graphical User Interface Development Environment (GUIDE) provides a rich set of tools for incorporating graphical user interfaces (GUIs) in M-functions. Using GUIDE, the processes of laying out a GUI (i.e., its buttons, pop-up menus, etc.) and programming the operation of the GUI are divided conveniently into two easily managed and relatively independent tasks. The resulting graphical M-function is composed of two identically named (ignoring extensions) files:

- A file with extension .fig, called a FIG-file that contains a complete graphical description of all the function's GUI objects or elements and their spatial arrangement. A FIG-file contains binary data that does not need to be parsed when he associated GUI-based M-function is executed.
- A file with extension .m, called a GUI M-file, which contains the code that controls the GUI operation. This file includes functions that are called when the GUI is launched and exited, and callback functions that are executed when a user interacts with GUI objects for example, when a button is pushed.

To launch GUIDE from the MATLAB command window, type

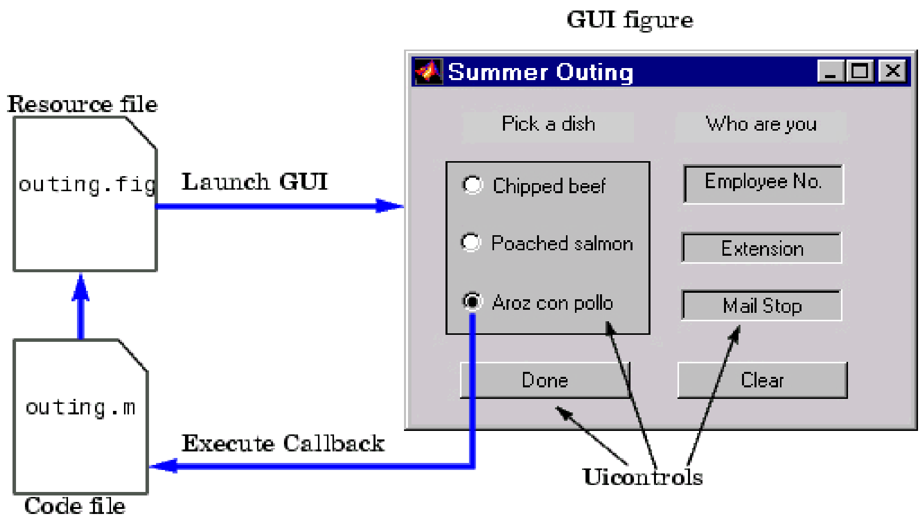
```
guide filename
```

Where filename is the name of an existing FIG-file on the current path. If filename is omitted, GUIDE opens a new (i.e., blank) window.

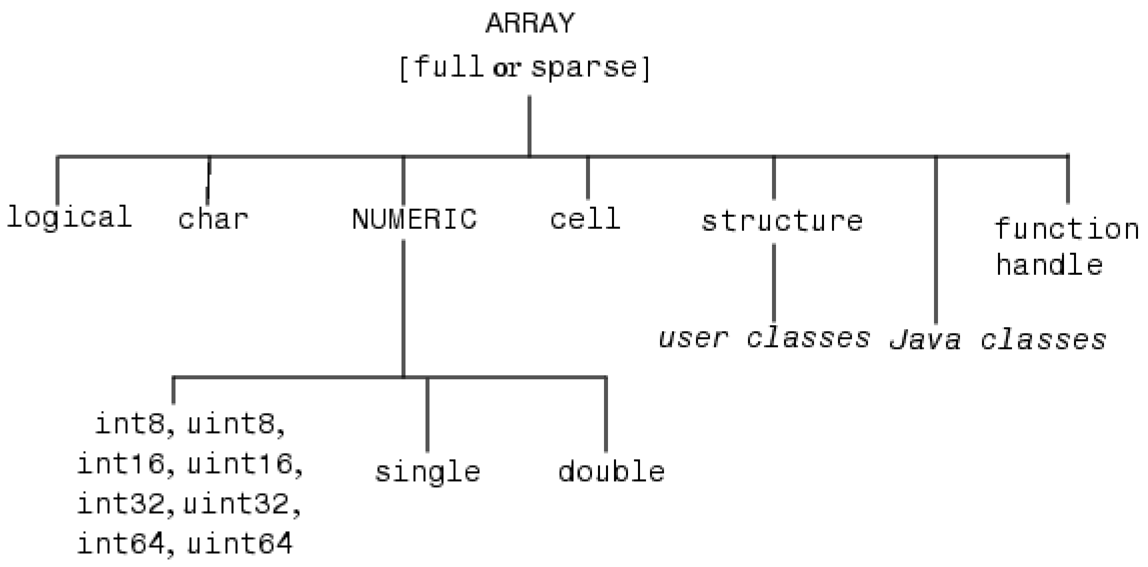

untitled.fig
File Edit View Layout Tools Help
Current Point: [238, 51]
Position: [520, 369, 573, 431]

A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components that enable a user to perform interactive tasks. The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed.

GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders just to name a few. GUIs created using MATLAB tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots.



Data Types in MATLAB are as follows:



FUNCTIONS:

MATLAB provides a large number of standard elementary mathematical functions, including abs, sqrt, exp, and sin. Taking the square root or logarithm of a negative number is not an error; the appropriate complex result is produced automatically. MATLAB also provides many more advanced mathematical functions, including Bessel and gamma functions. Most of these functions accept complex arguments. For a list of the elementary mathematical functions, type

- help elfun

For a list of more advanced mathematical and matrix functions, type

- help specfun
- help elmat
-

Some of the functions, like sqrt and sin, are built-in. They are part of the MATLAB core so they are very efficient, but the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in M-files. You can see the code and even modify it if you want. Several special functions provide values of useful constants.

Pi	3.14159265...
I	Imaginary unit, √-1
I	Same as i
Eps	Floating-point relative precision, 2^{-52}
Realmin	Smallest floating-point number, 2^{-1022}
Realmax	Largest floating-point number, $(2-\varepsilon)2^{1023}$
Inf	Infinity
NaN	Not-a-number

Operators in MATLAB:

Operators have familiar precedence rules as in most of the programming languages.

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division (described in "Matrices and Linear Algebra" in Using MATLAB)
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order
+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Left division (described in "Matrices and Linear Algebra" in Using MATLAB)
^	Power
'	Complex conjugate transpose
( )	Specify evaluation order

2.7 HARDWARE REQUIREMENTS

Operating System:

- Windows 7, Windows 8 or Windows 10

Hardware:

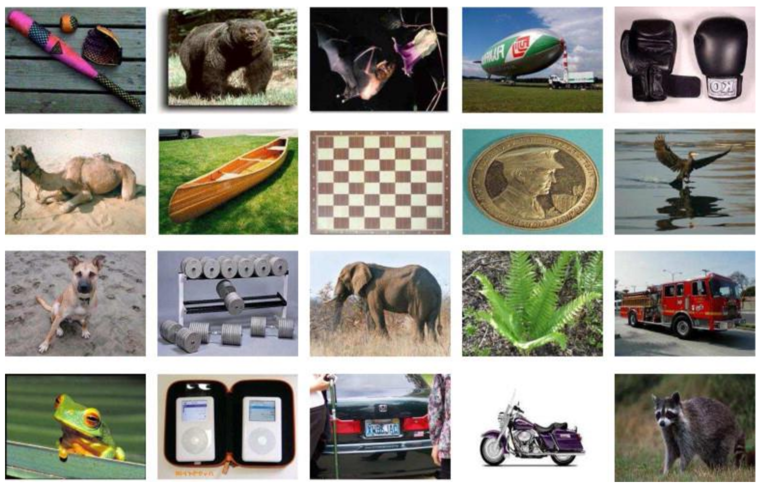
- Processor (CPU) with 2 gigahertz (GHz) frequency or above
- A minimum of 4 GB of RAM
- Monitor Resolution 1024 X 768 or higher
- A minimum of 5-8 GB of available space on the hard disk
- Internet Connection Broadband (high-speed) Internet connection with a speed of 4 Mbps or higher
- Keyboard and a Microsoft Mouse or some other compatible pointing device

Graphics:

Hardware accelerated graphics card supporting OpenGL 3.3 with 1GB GPU memory is recommended.

2.8 DATASET

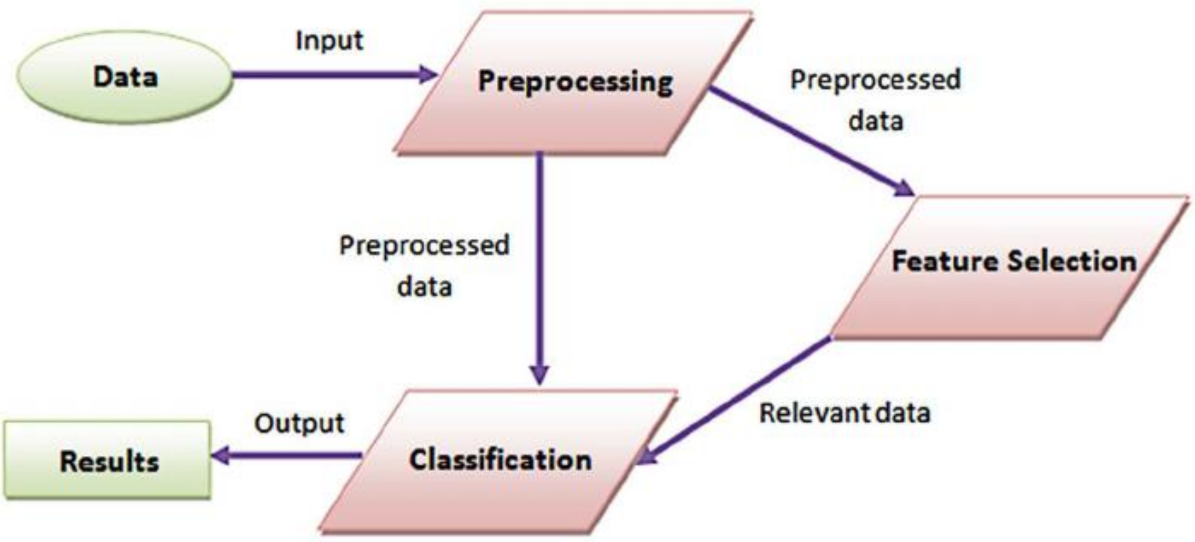
The dataset that we are using is Caltech-256 image dataset.
It is a dataset of 256 object categories consisting of 30607 images.



CHAPTER 3

3 DESIGN

3.1 SYSTEM ARCHITECTURE



3.2 OBJECTIVES

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

3.3 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information.
- Create document, report, or other formats that contain information produced by the system.

CHAPTER 4

4 IMPLEMENTATION

4.1 WORKING MODEL

Our model basically consists of 3 major steps:

i. relieff algorithm
ii. SVM-RFE algorithm
iii. Combining steps i and ii

4.2 RELIEFF ALGORITHM

Relieff is an algorithm developed by Kira and Rendell in 1992 that takes a filter-method approach to feature selection that is notably sensitive to feature interactions. It was originally designed for application to binary classification problems with discrete or numerical features. Relief calculates a feature score for each feature which can then be applied to rank and select top scoring features for feature selection. Alternatively, these scores may be applied as feature weights to guide downstream modeling. Relief feature scoring is based on the identification of feature value differences between nearest neighbor instance pairs. If a feature value difference is observed in a neighboring instance pair with the same class (a 'hit'), the feature score decreases. Alternatively, if a feature value difference is observed in a neighboring instance pair with different class values (a 'miss'), the feature score increases. The original Relief algorithm has since inspired a family of Relief-based feature selection algorithms (RBAs), including the Relief algorithm. Beyond the original Relief algorithm, RBAs have been adapted to (1) perform more reliably in noisy problems, (2) generalize to multi-class problems (3) generalize to numerical outcome (i.e. regression) problems, and (4) to make them robust to incomplete (i.e. missing) data.

To date, the development of RBA variants and extensions has focused on four areas; (1) improving performance of the 'core' Relief algorithm, i.e. examining strategies for neighbor selection and instance weighting, (2) improving scalability of the 'core' Relief algorithm to larger feature spaces through iterative approaches, (3) methods for flexibly adapting Relief to different data types, and (4) improving Relief run efficiency.

Their strengths are that they are not dependent on heuristics, they run in low-order polynomial time, and they are noise-tolerant and robust to feature interactions, as well as being applicable for binary or continuous data; however, it does not discriminate between redundant features, and low numbers of training instances fool the algorithm.

Take a data set with *n* instances of *p* features, belonging to two known classes. Within the data set, each feature should be scaled to the interval [0 1] (binary data should remain as 0 and 1). The algorithm will be repeated *m* times. Start with a *p*-long weight vector (W) of zeros.

At each iteration, take the feature vector (X) belonging to one random instance, and the feature vectors of the instance closest to X (by Euclidean distance) from each class. The closest same-class instance is called 'near-hit', and the closest different-class instance is called 'near-miss'. Update the weight vector such that

$$W_i = W_i - (x_i - \mathrm{nearHit}_i)^2 + (x_i - \mathrm{nearMiss}_i)^2$$

Thus, the weight of any given feature decreases if it differs from that feature in nearby instances of the same class more than nearby instances of the other class, and increases in the reverse case.

After *m* iterations, divide each element of the weight vector by *m*. This becomes the relevance vector. Features are selected if their relevance is greater than a threshold *τ*.

Kira and Rendell's experiments showed a clear contrast between relevant and irrelevant features, allowing *τ* to be determined by inspection. However, it can also be determined by Chebyshev's inequality for a given confidence level (*α*) that a *τ* of 1/sqrt(α*m) is good enough to make the probability of a Type I error less than *α*, although it is stated that *τ* can be much smaller than that.

Relief was also described as generalizable to polynomial classification by decomposition into a number of binary problems.

Reliable probability estimation

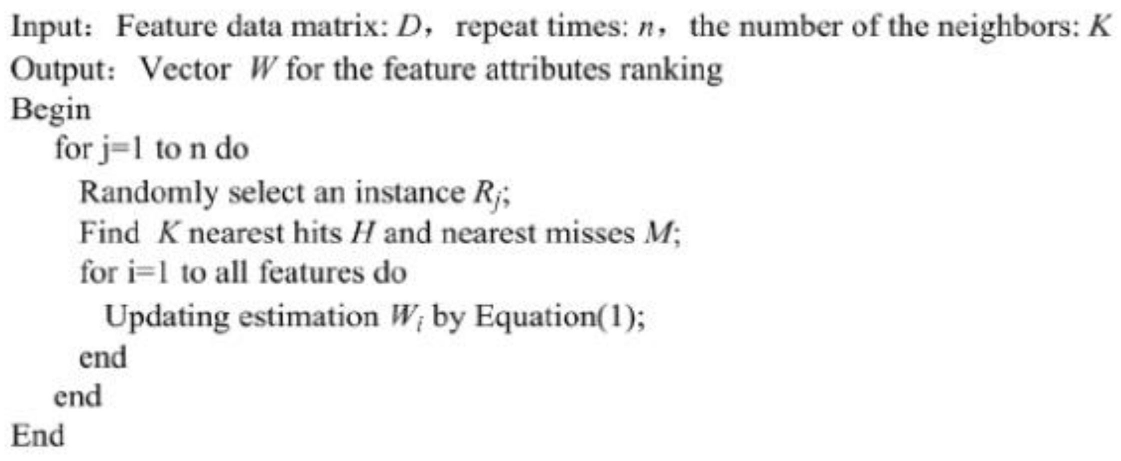
Rather than repeating the algorithm *m* times, implement it exhaustively (i.e. *n* times, once for each instance) for relatively small *n* (up to one thousand). Furthermore, rather than finding the single nearest hit and single nearest miss, which may cause redundant and noisy attributes to affect the selection of the nearest neighbours, Relief searches for *k* nearest hits and misses and averages their contribution to the weights of each feature. *k* can be tuned for any individual problem.

Incomplete data

In Relief, the contribution of missing values to the feature weight is determined using the conditional probability that two values should be the same or different, approximated with relative frequencies from the data set. This can be calculated if one or both features are missing.

Multi-class problems

Rather than use Kira and Rendell's proposed decomposition of a multinomial classification into a number of binomial problems, Relief searches for *k* near misses from each different class and averages their contributions for updating W, weighted with the prior probability of each class.



```
Input: Feature data matrix: $D$, repeat times: $n$, the number of the neighbors: $K$
Output: Vector $W$ for the feature attributes ranking
Begin
   for j=1 to n do
     Randomly select an instance $R_j$;
     Find $K$ nearest hits $H$ and nearest misses $M$;
     for i=1 to all features do
       Updating estimation $W_i$ by Equation(1);
     end
   end
End
```

4.3 SVM-RFE ALGORITHM

The high-performance implementations of machine learning algorithms have been enhanced by recent developments in programmable, highly parallel Graphics Processing Units (GPUs). In this paper, a solution which is based on Support Vector Machines Recursive Feature Elimination (SVM-RFE) is proposed to eliminate gene redundancy so that the compact gene subsets are provided from a given input data set. We use the microarray data set involving ovarian cancer as the sample input data set. 253 (tissue samples) x 15154 (genes) expression values are contained in the ovarian data set. The total memory working set size of this application is 300MB. The code designed in the paper is specifically written for the parallel GPUs application.

The Support Vector Machines are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. They can also be considered a special case of Tikhonov regularization. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

Feature selection is an important part dealing with machine learning technology. SVM-RFE is recognized as one of the most effective filtering methods, which is based on a greedy algorithm that only finds the best possible combination for classification without considering the differentially significant features between the classes. To overcome this limitation of SVM-RFE, the proposed approach is tuned to find differentially significant features along with notable classification accuracy. This package is able to enumerate the feature selection of any two-dimensional (for binary classification) data such as a microarray etc. This vignette explains the use of the package in a publicly available microarray dataset.

As we mentioned before, relief is a general and successful feature attributes estimator and is able to effectively provide quality estimation of features in problem with dependencies between feature attributes. However, relief does not explicitly consider the role of classifier in feature extraction. On the other hand, the SVM-RFE algorithm totally takes the importance of classifier into account, it is computationally expensive. Thus, it could be helpful to integrate the weight vector i W from relief into SVM-RFE method. Therefore, we devise a new ranking criterion as follows:

$$G_i = W_i + C_i .$$

where i C is the ranking indicator from SVM-RFE method. It should be pointed out that before the sum operation of i W and i C those two weight vectors should be normalized to [0, 1], which can eliminate the error caused by two different dimensions. Thus, the i^{th} G is the final ranking indicator for the i^{th} feature in our image classification.

Steps Involved are:

a) Train a data model
b) Compute a ranking of features
c) Remove the feature with the Worst Rank

Ranking Criterion:

If the number of ballots ranking A as the first preference is greater than the number of ballots on which another candidate B is given any preference, then A's probability of winning must be no less than B's.

This criterion is trivially satisfied by rank ballot methods which require voters to strictly rank all the candidates (and so do not allow truncation). The Borda count is usually defined in this way.

Woodall has called the Plurality criterion "a rather weak property that surely must hold in any real election" opining that "every reasonable electoral system seems to satisfy it." Most proposed methods do satisfy it, including Plurality voting, IRV, Bucklin voting, and approval voting.

Among Condorcet methods which permit truncation, whether the Plurality criterion is satisfied depends often on the measure of defeat strength. When *winning votes* is used as the measure of defeat strength in methods such as the Schulze method, Ranked Pairs, or Minimax, Plurality is satisfied. Plurality is failed when *margins* is used. Minimax using *pairwise opposition* also fails Plurality.

When truncation is permitted under Borda count, Plurality is satisfied when no points are scored to truncated candidates, and ranked candidates receive no fewer votes than if the truncated candidates had been ranked. If truncated candidates are instead scored the average number of points that would have been awarded to those candidates had they been strictly ranked, or if Nauru's modified Borda count is used, the Plurality criterion is failed.

The rank of A is the largest order of any non-zero minor in A. (The order of a minor is the side-length of the square sub-matrix of which it is the determinant.) Like the decomposition rank characterization, this does not give an efficient way of computing the rank, but it is useful theoretically: a single non-zero minor witnesses a lower bound (namely its order) for the rank of the matrix, which can be useful (for example) to prove that certain operations do not lower the rank of a matrix.

A non-vanishing p-minor ($p \times p$ submatrix with non-zero determinant) shows that the rows and columns of that submatrix are linearly independent, and thus those rows and columns of the full matrix are linearly independent (in the full matrix), so the row and column rank are at least as large as the determinantal rank; however, the converse is less straightforward. The equivalence of determinantal rank and column rank is a strengthening of the statement that if the span of n vectors has dimension p, then p of those vectors spans the space (equivalently, that one can choose a spanning set that is a *subset* of the vectors): the equivalence implies that a subset of the rows and a subset of the columns simultaneously define an invertible submatrix.

4.4 COMBINING BOTH THE ALGORITHMS

As we mentioned before, reliefF is a general and successful feature attributes estimator and is able to effectively provide quality estimation of features in problem with dependencies between feature attributes.

However, reliefF does not explicitly consider the role of classifier in feature extraction. On the other hand, the SVM-RFE algorithm totally takes the importance of classifier into account, it is computationally expensive.

Thus, it could be helpful to integrate the weight vector from reliefF into SVM-RFE method. Therefore, we devise a new ranking criterion.

4.5 CODE

4.5.1 Selecting an image from the dataset

```
function data
 b='.jpg';
 c1='.bmp';
fid = fopen('database.txt', 'w+');
for i=1:82
    a=num2str(i);
   filename=strcat(a,b);
    fprintf(fid,'%s\r',filename);
end
fclose(fid);
```

4.5.2 To represent the selected image on an axis

```
function Input_im_Callback(hObject, eventdata, handles)
% hObject    handle to Input_im (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file,path] = uigetfile('*.jpg','Pick an Image File');
if isequal(file,0) || isequal(path,0)
    warndlg('User Pressed Cancel');
else
    a = imread(file);
    axes(handles.axes12);
    imshow(a);
    title('Input Image');
    handles.a = a;
end
```

4.5.3 Search for the matching images

```
function search_image_Callback(hObject, eventdata, handles)
LL1 = handles.LL1;
LL2 = handles.LL2;
LL3 = handles.LL3;
fid = fopen('database.txt');
resultValues_Gc = [];      % Results matrix...
resultNames_Gc = {};
i = 1;                 % Indices...
j = 1;
while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break, end       % Meaning: End of File...
    database_image = imread(imagename);
    database_image = imresize(database_image,[256 256]);
    YUV_DB = rgb2ycbcr(database_image);
   y = YUV_DB(:,:,1);
   u = YUV_DB(:,:,2);
   v = YUV_DB(:,:,3);
   s1 = get(handles.popupmenu1,'value');
      switch s1
         case 1
         [ll1 lh1 hl1 hh1] = dwt2(y,'haar');
         [ll2 lh2 hl2 hh2] = dwt2(u,'haar');
         [ll3 lh3 hl3 hh3] = dwt2(v,'haar');
         case 2
         [ll1 lh1 hl1 hh1] = forward_lift(y,2);
         [ll2 lh2 hl2 hh2] = forward_lift(u,2);
         [ll3 lh3 hl3 hh3] = forward_lift(v,2);
      end
    [r c]=size(LL1);
    for m=1:r
       for n=1:c
    Dist1(m,n) = abs((LL1(m,n)-ll1(m,n))+(LL2(m,n)-ll2(m,n))+(LL3(m,n)-ll3(m,n)));
       end
    end
   resultValues_Gc(i) = sum(sum(Dist1));
   resultNames_Gc(j) = {imagename};
   i = i + 1;
   j = j + 1;
end
fclose(fid);
```

4.5.4 Ranking the images

```
[sortedValues1, index1] = sort(resultValues_Gc);
name = resultNames_Gc;
for i = 1:10
        a = imread(cell2mat(name(index1(i))));
        if i==1
        axes(handles.axes2);
        imshow(a);
        end
        if i==2
        axes(handles.axes3);
        imshow(a);
        end
        if i==3
        axes(handles.axes4);
        imshow(a);
        end
        if i==4
        axes(handles.axes5);
        imshow(a);
        end
        if i==5
        axes(handles.axes6);
        imshow(a);
        end
        if i==6
        axes(handles.axes7);
        imshow(a);
        end
        if i==7
        axes(handles.axes8);
        imshow(a);
        end
        if i==8
        axes(handles.axes9);
        imshow(a);
        end
        if i==9
        axes(handles.axes10);
        imshow(a);
        end
        if i==10
        axes(handles.axes11);
        imshow(a);
        end
end
```

4.5.5 Feature selection based on wavelength

```
function quary_feature_Callback(hObject, eventdata, handles)
inp_im = handles.a ;
inp_im = imresize(inp_im,[256 256]);
YUV = rgb2ycbcr(inp_im);
axes(handles.axes13);
imshow(YUV);
title('YUV image');
s1 = get(handles.popupmenu1,'value');
Y = YUV(:,:,1);
U = YUV(:,:,2);
V = YUV(:,:,3);
switch s1
   case 1
   [LL1 LH1 HL1 HH1] = dwt2(Y,'haar');
   [LL2 LH2 HL2 HH2] = dwt2(U,'haar');
   [LL3 LH3 HL3 HH3] = dwt2(V,'haar');
   case 2
   [LL1 LH1 HL1 HH1] = forward_lift(Y,2);
   [LL2 LH2 HL2 HH2] = forward_lift(U,2);
   [LL3 LH3 HL3 HH3] = forward_lift(V,2);
end
dec1 = [LL1 LH1;HL1 HH1];
dec2 = [LL2 LH2;HL2 HH2];
dec3 = [LL3 LH3;HL3 HH3];
figure,imshow(dec1,[]);
title('Wavelet Transform for Y Component');
figure,imshow(dec2,[]);
title('Wavelet Transform for U Component');
figure,imshow(dec3,[]);
title('Wavelet Transform for V Component');
handles.LL1=LL1;
handles.LL2=LL2;
handles.LL3=LL3;
guidata(hObject, handles);
helpdlg('Process Completed');
```

4.6 EXPERIMENTAL RESULTS

Our experimental dataset, named dataset-Caltech, includes six categories of images which is randomly picked from Caltech-256 database. Half of the images of each category are used for training and the others for testing.

In the experiment on dataset-Caltech, all the 75 feature descriptors as discussed above are first extracted for each image from dataset-Caltech, and then a 3268-dimension feature space can be obtained. In the second step, we consider each feature component in the feature space as a single feature, finally a compact and effective subset with about 150 feature components will be obtained.

It can be seen that our proposed algorithm performs the best among others on dataset-Caltech, which indicates the effectiveness of our new ranking criterion. When about 150 feature components are selected, our proposed relief-SVM-RFE method achieves the highest average classification accuracy of 96.14%, while the wrapper model SVM-RFE method achieves 94.43% and the filter model relief algorithm only achieves an accuracy of 93.27%. The results further demonstrate that our proposed hybrid model relief-SVM-RFE is superior to either the filter type relief method or the wrapper type SVM-RFE model.

CHAPTER 5

5 SOFTWARE TESTING

Testing is an important component in software life cycle. It aids in finding and repairing uncovered errors so that the software does not pose any problems to the vendor. Since the system is developed using an object-oriented approach, class testing is done at unit level and functional testing is done at system level.

Testing objectives include:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.

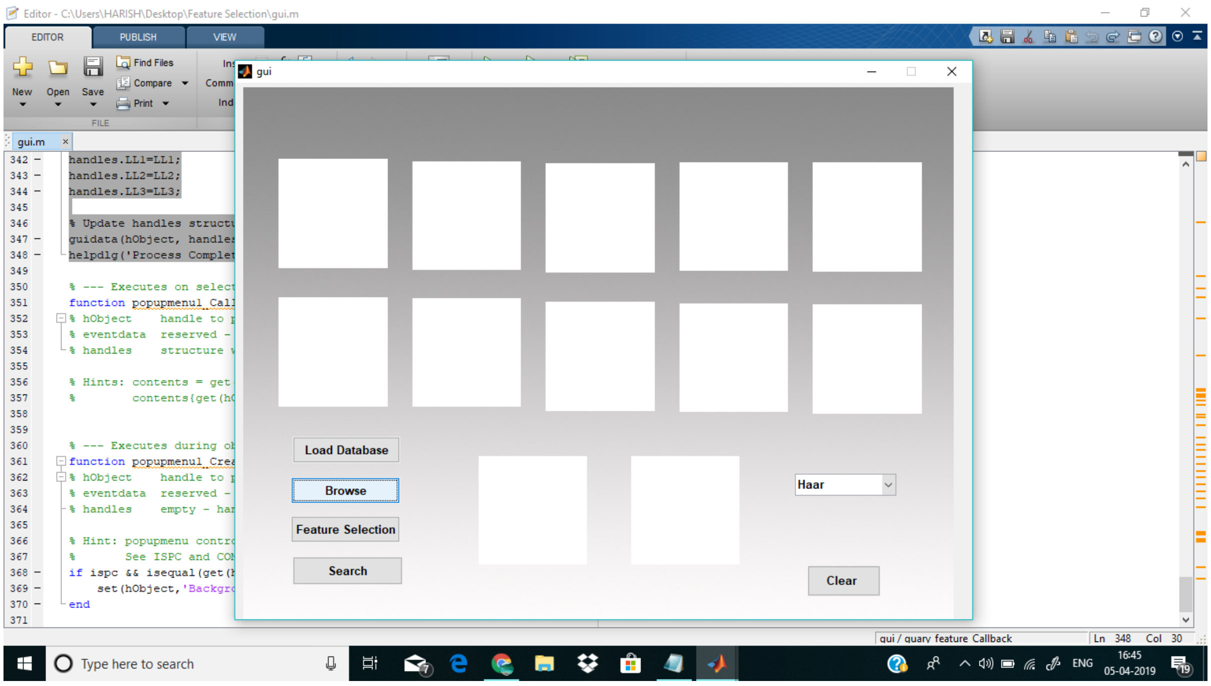
Testing is a schedule process carried out by the software development team to capture all the possible errors, missing operations and also a complete verification to verify objective are met and user requirement are satisfied. The design of tests for software and other engineering products can be as challenging as the initial design to the product itself.

5.1 TEST CASE

A test case in software engineering is a set of conditions or variables under which a tester will determine if a requirement or use case upon an application is partially or fully satisfied. It may take many test cases to determine that a requirement is fully satisfied. The following steps are to be followed to design the test cases.

- Each test case should be uniquely identified and explicitly associated with the class to be tested.
- The purpose of the test should be stated.
- Each test case should be uniquely identified and explicitly associated with the class to be tested.
- The purpose of the test should be stated.

i. Loading the database into the GUI.



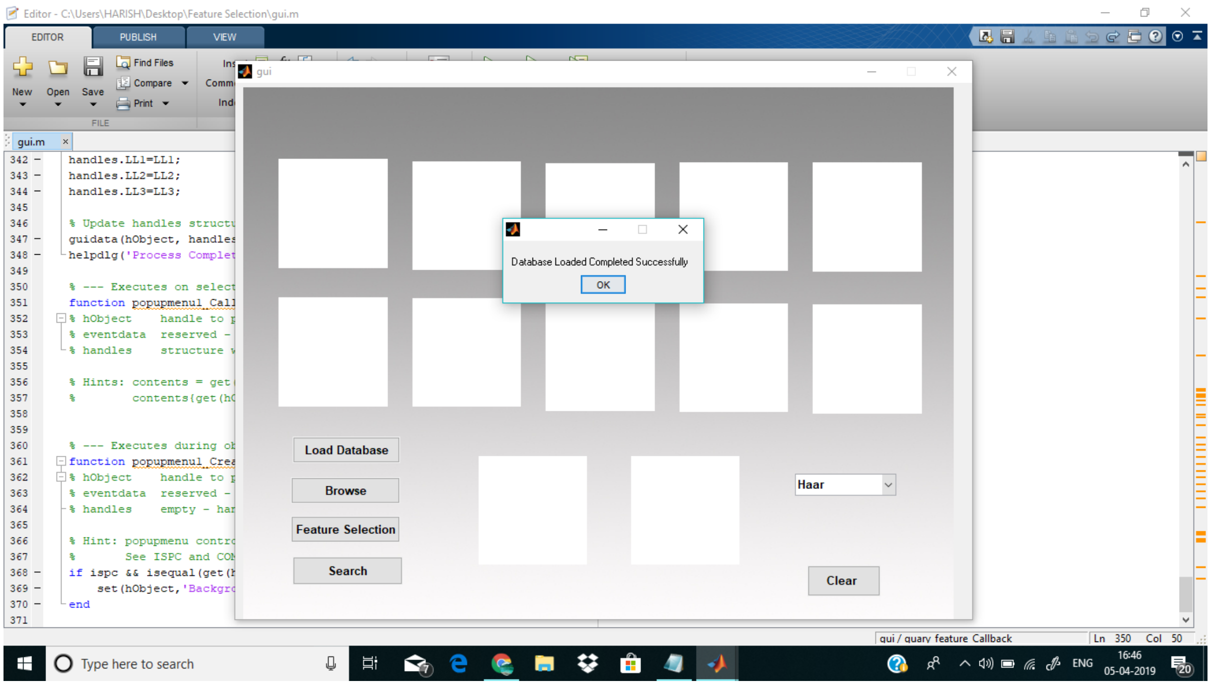
Editor - C:\Users\HARISH\Desktop\Feature Selection\gui.m

```
handles.LL1=LL1;
handles.LL2=LL2;
handles.LL3=LL3;

% Update handles structu
guidata(hObject, handles
helpdlg('Process Complet

% --- Executes on select
function popupmenu1_Call
% hObject    handle to p
% eventdata  reserved -
% handles    structure w

% Hints: contents = get
%        contents{get(hO

% --- Executes during ob
function popupmenu1_Crea
% hObject    handle to p
% eventdata  reserved -
% handles    empty - han

% Hint: popupmenu contro
%       See ISPC and COM
if ispc && isequal(get(h
    set(hObject,'Backgro
end
```

gui / quary feature Callback Ln 348 Col 30

16:45 05-04-2019

ii. Database successfully loaded.



Editor - C:\Users\HARISH\Desktop\Feature Selection\gui.m

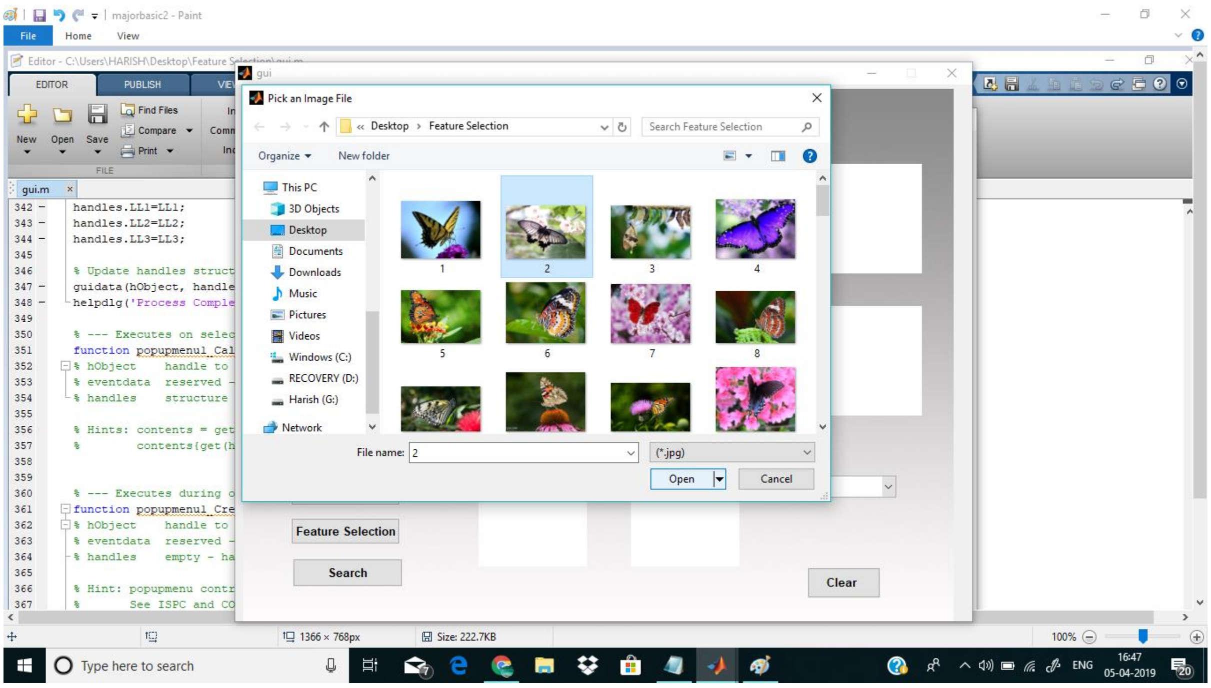
```
handles.LL1=LL1;
handles.LL2=LL2;
handles.LL3=LL3;

% Update handles structu
guidata(hObject, handles
helpdlg('Process Complet

% --- Executes on select
function popupmenu1_Call
% hObject    handle to p
% eventdata  reserved -
% handles    structure w

% Hints: contents = get
%        contents{get(hO

% --- Executes during ob
function popupmenu1_Crea
% hObject    handle to p
% eventdata  reserved -
% handles    empty - han

% Hint: popupmenu contro
%       See ISPC and COM
if ispc && isequal(get(h
    set(hObject,'Backgro
end
```

Database Loaded Completed Successfully

gui / quary feature Callback Ln 350 Col 50

16:46 05-04-2019

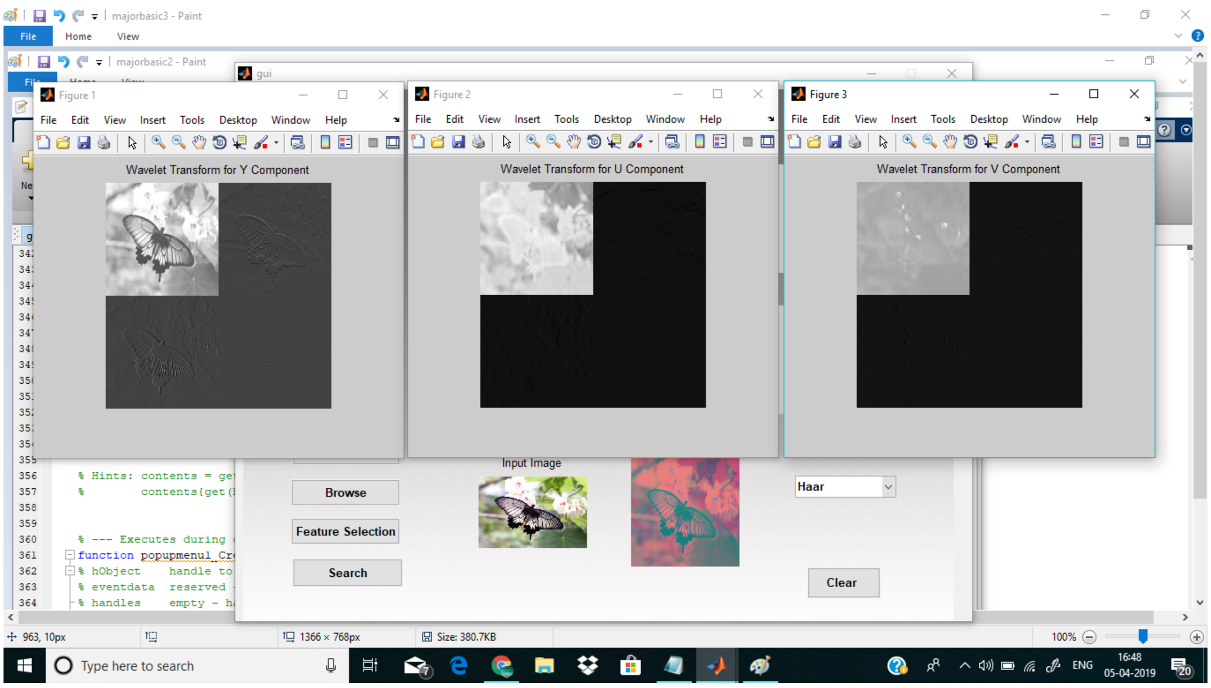
iii. Select an image from the database.



```
handles.LL1=LL1;
handles.LL2=LL2;
handles.LL3=LL3;

% Update handles struct
guidata(hObject, handle
helpdlg('Process Comple

% --- Executes on selec
function popupmenu1_Cal
% hObject    handle to
% eventdata  reserved -
% handles    structure

% Hints: contents = get
%        contents{get(h


% --- Executes during c
function popupmenu1_Cre
% hObject    handle to
% eventdata  reserved -
% handles    empty - ha

% Hint: popupmenu contr
%       See ISPC and CO
```

File name: 2
(*.jpg)

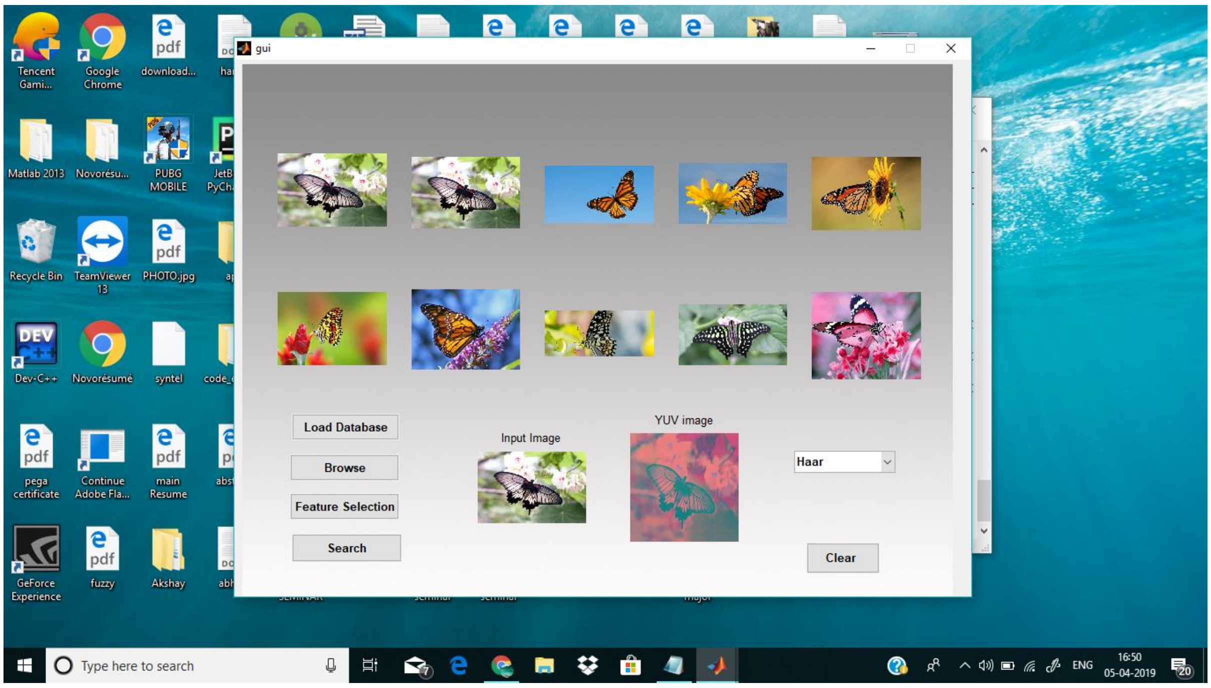
iv. Performing Feature Selection



```
% Hints: contents = ge
%        contents{get(

% --- Executes during
function popupmenu1_Cr
% hObject    handle to
% eventdata  reserved
% handles    empty - h
```

Haar

v. Displaying the output



CHAPTER 6

6 CONCLUSIONS

In this paper, an improved and novel strategy combining relief and SVM-RFE algorithm is proposed to select a compact feature subset performing well in image classification. In this method, the relief method is employed to filter out many noisy feature components and obtain an effective candidate feature subset, then the new evaluation criterion integrating those from the relief method and the SVM-RFE is applied to select the final feature components. Experimental results demonstrate that with the feature subset from our proposed relief-SVM-RFE method a better classification performance can be achieved. However, some future work is needed to further improve the performance of the selection algorithm. For example, much more image descriptors can be extracted\ for images before processing or even a more efficient and less computational time-consuming evaluation criterion can be devised. It is well recognized that the issue of effectively feature selection and accurately image classification still remains to be open.

CHAPTER 7

7 REFERENCES

[1] Guyon, I. and Elisseeff, A. (2003) An Introduction to Variable and Feature Selection. The Journal of Machine Learning Research, 3, 1157-1182.

[2] Song, D.J. and Tao, D.C. (2010) Biologically Inspired Feature Manifold for Scene Classification, IEEE Transaction on Image Processing, 19, 174-184.

[3] Marko, R.S. and Lgor, K. (2003) Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning, 53, 23-69.

[4] Zhang, Y., Ding, C. and Li, T. (2008) Gene Selection Algorithm by Combining ReliefF and mRMR. BMC Genomics, 9, S27. http://dx.doi.org/10.1186/1471-2164-9-S2-S27

[5] Suykens, J.A.K. and Vandewalle, J. (1999) Least Squares Support Vector Machine Classifiers. Neural Processing Letters, 9, 293-300.http://dx.doi.org/10.1023/A:1018628609742

[6] Guyon, I., Weston, J., Barnhill, S., et al. (2002) Gene Selection for Cancer Classification Using Support Vector Machines. Machine Learning, 46, 389-422. http://dx.doi.org/10.1023/A:1012487302797

[7] John, G.H., Kohavi, R., Pfleger, K. (1994) Irrelevant Features and the Subset Selection Problem. ICML, 94, 121-129.

[8] Yang, J., Yu, K., Gong, Y., et al. (2009) Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, 1794-1801.

[9] Van De Sande, K.E.A., Gevers, T. and Snoek, C.G.M. (2010) Evaluating Color Descriptors for Object and Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32, 1582-1596. http://dx.doi.org/10.1109/TPAMI.2009.154

[10] Chun, Y.D., Kim, N.C. and Jang, I.H. (2008) Content-Based Image Retrieval Using Multiresolution Color and Texture Features. IEEE Transactions on Multimedia, 10, 1073-1084. http://dx.doi.org/10.1109/TMM.2008.2001357

[11] Carlin, M. (2001) Measuring the Performance of Shape Similarity Retrieval Methods. Computer Vision and Image Understanding, 84, 44-61. http://dx.doi.org/10.1006/cviu.2001.0935