# A Hierarchical Multiple Classifier Learning Algorithm [*]

Yu-Yu Chou
Numerical Technologies, Inc.
70 West Plumeria Drive,
San Jose, CA 95134, U.S.A.
yuyu@numeritech.com

Linda G. Shapiro
University of Washington
Department of Computer Science and Engineering
Box 352350, Seattle, WA 98195, U.S.A.
shapiro@cs.washington.edu

November 13, 2000

[*]This research was partially supported by the Washington Technology Center and NeoPath, Inc (now TriPath Imaging Inc).

# A Hierarchical Multiple Classifier Learning Algorithm

## Abstract

This paper addresses the classification problem for applications with extensive amounts of data and complex features. The learning system developed utilizes a hierarchical multiple classifier scheme and is flexible, efficient, highly accurate and of low cost. The system has several novel features: 1) It uses a graph-theoretic clustering algorithm to group the training data into possibly overlapping clusters that each represent a dense region in the data space. 2) Component classifiers trained on these dense regions are specialists whose probabilistic outputs are gated inputs to a superclassifier. Only those classifiers whose training clusters are most related to an unknown data instance send their outputs to the superclassifier. 3) Sub-class labeling is used to improve the classification of superclasses. The learning system achieves the goals of reducing the training cost and increasing the prediction accuracy compared to other multiple classifier algorithms. The system was tested on three large sets of data, two from the medical diagnosis domain and one from a forest cover classification problem. The results are superior to those obtained by several other learning algorithms.

## Originality and Contribution

In this paper, we describe a hierarchical multiple-classifier classification scheme. A new way to group the training data using a graph theoretic clustering algorithm is proposed. With the new partitioning mechanism, each data cluster represents a dense region in the data space. Classifiers trained on these dense regions are better able to distinguish local variations in the data. Furthermore, the individual clusters are much easier to manage in terms of computation time and system resources. Since the clusters are not mutually exclusive, borderline data can simultaneously train different classifiers with different class distributions, providing more information for global decision making performed by the second stage classifiers.

The architecture of our hierarchical classifier feeds the results of the most relevant component classifiers to a superclassifier through a gating network. The soft clusters also provide a unique way to control the switches in the gating network. The clustering criteria are used as a pre-filter to determine which classifiers should be applied for an unknown data instance. This not only reduces the training time for the second stage classifiers but also improves the quality of their decisions.

Another novel idea is the use of sub-classes to aid in the classification of the main classes. Contrary to conventional wisdom that more information only adds complexity to the design of a classifier, the additional knowledge employed in our architecture allows the component classifiers to establish better local decision boundaries and hence to achieve higher accuracy.

Our system has proved successful on a practical medical application as well as on another data set with similar characteristics - enormous amounts of data with complicated features. The framework provides a flexible and efficient way to investigate classification problems with similar scales. Different basic classification algorithms can be adapted to enhance the performance depending on the characteristics of the problems. Difficult problems can be broken into pieces to accelerate the processing time without sacrificing performance. The extra feature reduction ability and error instance incorporation can further reduce the computation cost or improve the performance.

# 1 Introduction

The use of machine learning techniques in medical diagnosis applications is common. The automation of pre-screening for cervical cancer [21] [9] [28] [31] is one of the examples. An automated system allows the reduction of clinical costs for a manual examination and eliminates the technician's subjective bias. However, it is a very complicated and difficult task. Many issues need to be taken into considerations when designing a system. These include the calibration of hardware, the reliability of image processing routines, the definition and extraction of object features, and the specificity and sensitivity of classifiers. Furthermore, many of the design considerations are mutually exclusive. For example, a system should have a high sensitivity of detection, so that it can respond to the presence of a very small amount of abnormal cells; on the other hand, an over-sensitive system may cause many false-alarms, which ultimately increase the cost. Therefore, any system design will contain some trade-offs and compromises.

A current system used for automatic pre-screening for cervical cancer examination developed by NeoPath Inc. [31] uses multiple-level probabilistic decision trees to classify the data. The decision trees are hand-carved by the designers and the parameters are fine-tuned to achieve the best performance. It is a tedious job that requires extensive man-power just for a single set of data. The use of machine learning techniques is expected to achieve the following goals:

- Accelerate the training process.
- Automate the training procedure and reduce human interaction.
- Enhance the classification accuracy.

There are several distinguishing characteristics of the data set which affect the classification task greatly:

1. The amount of data is tremendous; each slide can contain thousands of cells. For each laboratory, the number of slides can easily exceed 1,000 per day.
2. Each data instance (cell) is described by a set of sophisticated features. There are more than 300 features for each instance in the current system.
3. There are many different sources of noise existing in the data set. The noise can be induced by technicians' operating differences, focus problems, variations in specimen collection, and other aspects of the data collection procedure.

The aforementioned characteristics make the training and classification task difficult. The problem for stand-alone classification algorithms such as decision trees [32]

[17] and rule-based induction algorithm [7] is that they tended to over-fit the training data and thus produced very poor results on the test data; while neural networks [15] produced less satisfying results than current hand-tuned classifiers yet required a great amount of system resources and training time. Another difficulty lies in the nature of the complicated classes in the problem. There are three levels of classes. Table 1 shows the first two levels of classes, and 142 different classes are associated with the third level. The large number of classes make the decision boundaries complicated, erratic and unstable for standard algorithms.

Multiple classifier learning algorithms have been widely utilized in various machine learning problems [24] [25] [20] [2] [29]. A lot of research has been concentrated on the construction of a good ensemble of classifiers. The output of an ensemble is usually more accurate than the output of its component classifiers provided that 1) the error rate for each component classifier is less than 50% and 2) the errors among the component classifiers have no correlations or small correlations [16] [1].

The pattern recognition (classification) / learning methodology described in this paper is a hierarchical multiple classifier mechanism. It is summarized in the block diagram of Figure 1. The system includes a training phase and a classification phase. In the training phase, the data are clustered into sub-populations, and statistical information for each cluster is computed. The data are then relabeled with the sub-class labels, and a set of component classifiers are constructed for each sub-population. A reduced feature set is produced according to the initial results from the component classifiers that can be used to refine their construction if necessary. A set of identified error instances can be obtained to further refine the training process. After the component classifiers are built, they are applied to the entire data set to produce a new set of data with the additional classifications as new features. The new data set can be considered as a non-linear transformation of the original data set or as a cross-validation data set. A super-classifier is constructed from the new data set and the cluster information computed previously. The process can be repeated for another layer of super-classifiers, recursively. In the classification phase, the process is similar to the training phase. For an unknown data instance, the component classifiers are applied to it to produce the input data for the super-classifier. The final result is produced by the super-classifier according to both the input feature vector and the cluster information.

The survey paper by Dietterich [10] described various algorithms and techniques for multiple classifiers. They can roughly be divided into the following categories:

- Manipulating the training data by sub-sampling or substitution so that each component classifier is trained on a data set with a different class distribution [5] [14] [33]. Our data clustering is related to this approach, but it differs in that the amount of data used to train each component classifier is substantially less than that of other algorithms. The training cost does not increase proportionally as the number of classifi ers increases. Furthermore, the cluster information in our approach serves an additional purpose in the construction of the super-classifi er.

- Manipulating the input features to produce training data with different feature subsets [19].
- Manipulating the target function by grouping classes to produce different training data sets [11]. Our approach also manipulates the target function, but in quite a different way. Instead of grouping the classes, our approach attempts to break classes into fi ner sub-classes. The effect of this approach is discussed in a later section.
- Incorporating randomness into the construction of classifi ers.

Multiple classifi ers also differ in the methods they use to combine the outputs of their component classifi ers. The methods in the literature can be classifi ed as unweighted voting, weighting voting and gating networks. We will compare our system with some of the popular algorithms such as bagging, boosting, the EM algorithm, and others in a later section.

## 2 Data Characteristics and Preparation

Our work was motivated by the need to automate the development of classifi ers for complex medical diagnosis problems with large numbers of features. Our benchmark problem was the classifi cation of cervical smears as normal, abnormal, or artifact, and our goal was to meet or exceed the accuracy of an existent system developed by NeoPath, Inc. The architecture of the Neopath system for *automated cervical smear screening* is illustrated in Figure 2. The processes of image acquisition, segmentation, feature computation and ROI (region of interest) extraction are all completed at the hardware level. The single-cell classifi cation algorithm is the most important one and so is the focus of our study. For training purposes, the extracted cells have been marked by experts with their corresponding classifi cations.

We used two data sets provided by NeoPath to train and test our classifi ers. Although in a normal laboratory environment, the majority of the cells examined are normal, classifi ers trained for data with this type of class distribution have diffi culty locating the abnormalities. The decision boundaries are often skewed or over-generalized to accommodate the descriptions of the majority data, thus ignoring the real focus of the problem. The purpose of pap smear screening is to locate the abnormality among the cells. To enhance the classifi ers' responses to the abnormal cells, the training data were intentionally prepared with more abnormal instances. In the two data sets, the ratio of abnormal cells to normal cells are about 2:1 and 3:1, respectively.

## 2.1 Subclass Labeling

One of the characteristics of the data sets is that there are three levels of classes for target classification. This is quite helpful in designing the component classifiers. The classes of the first two levels are shown in Table 1. Although there are 16 different classifications in the second level, only some of them occur with high enough frequency to be recorded. Table 2 shows the distribution of classes in two sets of data. The unlisted second level classes have no recorded data instances, or the number is negligible.

The objective of the classifiers is to derive the decision boundaries to separate data instances from different classes. For a complicated or noisy data set, the decision boundaries are also very complicated. All of the classification algorithms derive the decision boundaries with the goal of minimizing the misclassification rate of the training data. They also need to keep the decision boundaries smooth (generalized) enough to avoid over-fitting the training data. When data are noisy or complicated and the target consists of only a few classes, the decision boundaries are usually over-generalized and favor classes with larger number of data instances. With the existence of subclasses, the estimated boundaries can be fit closer to the true boundaries, improving the classification accuracy.

For data sets without multiple levels of classes, the sub-class processing is still applicable with additional steps. A graph theoretic clustering algorithm (described in section 3.1.3) can be utilized to produce sub-classes. For data instances of the same class, the graph theoretic clustering algorithm is used to partition the instances into clusters. Each cluster is then assigned a new sub-class label and the classifier can be trained on these new classes. In order to improve the classification accuracy, an important issue is that each sub-class should contain enough instances for the training algorithms. The number of instances of our data as shown in Table 2 is large enough for general learning algorithms such as decision trees or neural networks. However, instances belonging to smaller sub-classes can be treated as exceptions. In this case, an instance-based learning (IBL) algorithm such as the $k$-Nearest Neighbor algorithm can be utilized as a filter to detect these exceptions prior to the execution of the other classification algorithms.

## 2.2 Feature Dimension Reduction

There have been extensive research efforts on the subject of feature subset selection [22] [26] [3]. Due to the advancement of hardware technology and computational ability, more and more features can be exploited to solve difficult classification problems. A large feature set can provide more information on the description of the problem. However, it also draws two concerns: 1) irrelevant features can lead to false decision boundaries by learning algorithms; and 2) irrelevant features and/or highly-correlated features increase the cost of computation without improving the results. Hence the feature reduction process is important not only for reducing the computational expenses but also for increasing the output accuracy.

There are several questions that need to be answered in the process of feature subset selection:

- How many features are necessary to fully describe a data set?
- How can the quality of a feature be measured?
- Is the cost of the extra feature selection process justifi able? Is it worth the effort?

Unfortunately there are no simple nor absolute answers to these questions, especially for diffi cult problems. Usually we rely on heuristic rules to judge the answers to these questions. Some common techniques include greedy search by removing features one at a time; replacement of data with their descriptive statistics; principal components analysis; and the use of classifi cation algorithms in the feature selection process. Here we suggest a heuristic approach based on the application of neural networks.

An important characteristic of neural networks is that by examining the values of weights, we can perform an input feature relevance test to determine which input features are more important. The features with higher summed weights often have a much greater impact on determining the outcome of a classifi cation than those with lower weights. The equation to calculate the input relevance ($R_i$) for the $i$th input feature is $R_i = \frac{\sum_j w_{ij}^2}{\sum_i \sum_j w_{ij}^2}$ where $w_{ij}$ is the weight between input node $i$ and hidden node $j$.

In our hierarchical structure, each component classifi er is trained on a subset of the original training data (see section 3.1), therefore the data size is easily manageable even for neural networks. For each component classifi er, a set of features is selected according to their input relevance with respect to either the number of desired features or the pre-set thresholds. A fi nal feature subset is produced from these feature sets either by intersection, union or weighted average.

Since the training of component classifi ers in our approach is less resource-demanding, we can afford to use all the features available in the data set. Furthermore, in our hierarchical mechanism, the component classifi ers can be treated as non-linear transformations. Each classifi er corresponds to a different transformation. A data instance is transformed into a smaller dimension (the number of total sub-classes) by choosing one or some of these transformations. The super-classifi er is then trained on these transformed data.

## 3 Hierarchical Multiple Classifier Mechanism

The construction of multiple classifi er algorithms has been an active research topic in supervised learning for the past decade. In supervised learning, a set of training examples of the form $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ is given to a classifi er algorithm to solve for an unknown function $y = f(\mathbf{x})$. Each $(\mathbf{x}_i, y_i)$ pair represents an instance in the data space. The vector $\mathbf{x_i}$ records the features extracted from instance $i$; $\mathbf{x}_i = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,n}\rangle$. The $y$ values are drawn from a set of discrete classes $\langle 1, 2, \ldots, K\rangle$. Consider the complete data set (including training data, test data, and unseen data) as a hypothesis space; in the case of classifi cation, the unknown $f$ can be treated as a union of all the decision boundaries. These decision boundaries divide the hypothesis space into some disjoint regions in which each region belongs to one of the output classes. A classifi er is an attempt to approximate the unknown function (decision boundaries) to describe the relationship between the input features and the output classes.

Different classifi er algorithms approximate the unknown decision boundaries with different principles. Some algorithms are excellent at characterizing the details of the training data but are very sensitive to the noise. Some algorithms are less sensitive to noisy data but may over-simplify the decision boundaries. Therefore the errors generated by different algorithms may contribute to different parts of the training data. This scenario may also be achieved by the same algorithm with different settings. The use of multiple classifi ers takes advantage of this characteristic and reduces the errors generated by a single classifi er.

The hierarchical multiple classifi er mechanism described in this paper consists of two major parts: 1) the construction of component classifi ers; and 2) the combination scheme. Figure 3 shows the construction of component classifi ers. The training data are fi rst partitioned into various data clusters, each data cluster representing a sub-population of the original training data. The class distribution of a data cluster is different from that of the original training data. Instead of utilizing the original classes as the target outputs, we assign the sub-classes as the target outputs of each data cluster; the component classifi ers are trained to learn the decision boundaries between these sub-classes. The label "Level II Classes" in the fi gure refers to the assignment of sub-classes, which are $\{$ *Ascus, LSIL, HSIL, Cancer, Repair, Normal, Artifact* $\}$ in the pap smear data. A similar label "Level I Classes" in Figure 4 refers to the original classes: $\{$ *Abnormal, Normal, Artifact* $\}$. For each data cluster, the mean and the variance of the features of its members are computed and stored along with the member instances.

Figure 4 illustrates the combination scheme. After the component classifi ers {C1, C2, ..., CN} are constructed for each data cluster, the component classifi ers are tested on the entire data set, with the unseen instances acting as the cross-validation set. Each component classifi er generates a probabilistic output; they are combined as the input data for the next level classifi er (super-classifi er). A ranking of the data clusters is generated for each data instance. The rank is assigned according to the distance between the data instance and the centroid of a data cluster. The highest rank is assigned to the data cluster whose centroid is the closest to the data instance, and the lowest rank is assigned to the farthest one. These ranks are utilized to control the combination of outputs of the component classifi ers. An additional error classifi er CE which is trained on the error instances generated from the other component classifi ers can also be added to produce additional data for the super-classifi er. In training the super-classifi er, either the original classes "Level I Classes" or the sub-classes "Level II Classes" can be assigned as the target.

## 3.1 Ensembles Construction

As discussed in Section 1, there are four main approaches for constructing the component classifi ers: manipulating the training data, manipulating the input features, manipulating the target function, and incorporating randomness into the algorithms. For the feature subset approach, the relevant features are usually identifi ed by a feature selection process. If a component classifi er is constructed without all the relevant features, the accuracy of this classifi er is likely to decrease. The less relevant the features used, the lower the classifi cation accuracy. If the component classifi ers are constructed with all the relevant features and with different sets of irrelevant features, the errors from different classifi ers are most likely to be highly correlated, since the irrelevant features contribute nothing to the classifi cation. Therefore it is not a very effective approach. The same can be said for the approach to incorporate randomness into the construction of classifi ers. The training cost (time) won't be reduced by the injected randomness but will increase substantially. Furthermore the diversity of component classifi ers induced by the randomness is limited. We believe that among the four general approaches, sub-sampling the training data and manipulating the target function are better ways to build the component classifi ers for complicated mega-data. Our hierarchical system was developed based on these two principles.

There are many different ways to partition the training data. We tried three different methods: random partitioning, K-means clustering, and graph-theoretic clustering.

### 3.1.1 Random Partitioning

Randomly drawing sub-samples is a common approach in algorithms like *bagging* [5]. One characteristic associated with this type of algorithm is the use of repetitions of data instances in each sub-sample. In a problem with voluminous training examples, this is quite a burden in computational cost as well as system resources. In our study, we randomly divided the training data set into $m$ disjoint subsets. The result from our super-classifi er showed no signifi cant difference compared to the *bagging* algorithm, while each partition was quite manageable in terms of memory requirements and CPU time.

One problem with using the random partition is that we have no control over or knowledge about the strength of the trained classifi ers. Each classifi er may be excellent for discriminating a sub-region of the entire hypothesis space; however, a data example is likely to be outside the scope of most of the classifi ers. The erroneous classifi cation contributed by most of the component classifi ers will affect the decision of the super-classifi er and the fi nal results.

### 3.1.2 K-Means Clustering

By clustering similar data instances together, we can have a better realization of the characteristics of each group of data. This realization is helpful in the construction of the super-classifi er. We modifi ed an algorithm proposed in [18], which is based on the K-means and maximin-distance algorithms, to perform the clustering task.

Let a data instance $x$ be represented by an $n$-dimensional feature vector $\langle\ a_1(x), a_2(x), \ldots, a_n(x)\ \rangle$ where $a_r(x)$ denotes the value of the $r$th feature of instance $x$. All of the data instances can be considered as points inside the corresponding $n$-dimensional feature space. The principle of the clustering algorithm is to group similar data points together in terms of their Euclidean distances. The clustering algorithm can be summarized in the following steps:

1. Select two data instances which are as far away as possible from each other in the feature space as follows
   - Randomly select one instance $s$.
   - Find a data instance $a$ which is furthest from $s$.
   - Find a data instance $b$ which is furthest from $a$.
   - The instances $a$ and $b$ are the cluster centers of the initial guess.
2. Assign the remaining data instances to the closest cluster center.
3. Recalculate the cluster centers by averaging the data instances in each cluster..
4. Compute the distance of each data instance to its nearest cluster center, and record the maximum of these distances, $d_{max}$ and the corresponding data instance $X$.
5. Determine whether a new cluster center is necessary as follows:
   - calculate the average distance $d_{avg}$ between the existing cluster centers.

- if $d_{max}$ is greater than $d_{av}/2$, the instance $X$ is assigned as a new cluster center and the process repeats from Step 2.

6. To avoid the data being broken into many small clusters, an additional constraint for the number of allowed clusters is set and the process is stopped when the constraint is met.

A notable characteristic in the process is that the data are clustered based only on the feature vectors. The class distribution is not taken into account in the process. Therefore, each cluster of data represents a region in the feature space. The component classifi er trained for the specifi c data cluster has higher discriminant ability for data in this neighborhood than the data far away from the cluster. It is an advantage we utilize in the combining strategies.

### 3.1.3 Graph-Theoretic Clustering

The aforementioned random-partition and K-means-variant clustering processes are both "hard-partition" methodologies for clustering the data; that is, each data instance only belongs to one cluster and there is no overlaps among the clusters. This is a disadvantage for even a slightly noisy data set, since the noisy data instances will distort the decision boundaries, and the distortion cannot be recovered during the combination (super-classifi er) process.

The third method we tried was a "soft-partition" clustering algorithm based on graph theory. In "soft-partitions" the data instances are allowed to appear in more than one cluster; therefore the boundaries between clusters are not rigidly defi ned but rather blurred. Notice that the boundaries referred to here are not decision boundaries but the dividing paths in the feature space to separate group of data points.

The graph theoretic clustering algorithm [34] used in this work was originally developed for the decomposition of polygonal shapes represented by graphs whose nodes correspond to sampled points along the boundaries of the shapes and whose links represent the relationships between pairs of nodes. Figure 5 (a) shows an example of a simple polygonal shape and (b) shows its corresponding graph in which two nodes are connected by an edge if the corresponding two points are visible to each other through the shape. In the example, there are three compact sub-graphs related to the three regions in the polygon.

The goal of the clustering algorithm is to fi nd the clusters (compact sub-graphs) according to certain well-defi ned parameters. A cluster is a maximal chain of highly compact, highly connected nodes. A conditional density function is defi ned to describe the neighborhood condition of a given node. The conditional density $D(X|Y)$ of node

$X$ given node $Y$ is defined as the number of nodes in the neighborhood of $Y$ which have $X$ as a neighbor. Since the relations between nodes are symmetric, $D(X|Y)$ is actually the number of neighbor nodes common to both node $X$ and $Y$, that is, $D(X|Y) = D(Y|X) = \#(\text{neighborhood}(X) \cap \text{neighborhood}(Y))$. Large groups of nodes with loose relations can be found by using the density function. For each node $X \in S$, a potential region $Z(X, K)$ is defined by $Z(X, K) = \{Y \in S | D(Y|X) \geq K\}$ where $K$ is an integer. Small values of $K$ leads to large and loose regions around the node $X$; large values of $K$ lead to small and tight regions. A region large enough to contain a cluster and yet remain dense must satisfy $Z(X) = Z(X, M)$ where $M = \max\{K | \#Z(X, K) \geq K\}$.

The concept of a *dense region* is defined to identify possible clusters. A *dense region* is a set of nodes and related links that satisfy the following conditions:

1. The nodes in a *dense region* should associate / have relationships with a large enough number of nodes in the same *dense region*.
2. A *dense region* should be highly compact.
3. A *dense region* should have a minimal number of nodes.

A parameter *association* is defined for the first condition. Let $S$ be a set and $R \subseteq S \times S$ be a binary relation on $S$. For any node $N$ in a subset $B$ of $S$, the *association* $A(N|B)$ of node $N$ to subset $B$ is defined as the ratio of the number of nodes in $B$ that are neighbors of $N$ to the total number of nodes in $B$, that is $A(N|B) = \frac{\text{neighborhood}(N) \cap B}{\#B}$. For any subset $B$ of $S$, a parameter *compactness* is defined for the second condition. It is defined as the average association of the nodes of $B$ that $C(B) = \frac{1}{\#B} \sum_{N \in B} A(N|B)$. Three thresholds *minassociation*, *mincompactness*, and *minsize* are given for each condition. For a set of nodes $B \subseteq S$ to be considered as a dense region, it has to satisfy

- $B = \{N \in Z(X) | A(N|Z(X)) \geq minassociation\}$ for some $X \in S$;
- $C(B) \geq mincompactness$;
- $\#B \geq minsize$.

Table 3 lists the neighborhood of each nodes for the example graph in Figure 5(b), Table 4 lists the conditional density and Table 5 lists the $M$, $Z(N)$, and $C(Z(N))$ for each node $N$ in the graph, respectively.

For the thresholds $minassociation = 0.5$, $mincompactness = 0.8$, and $minsize = 3$, the dense regions in Figure 5 are {1, 2, 10, 11, 12}, {3, 4, 9, 10}, {4, 5, 6, 7}, {4, 5, 6, 7, 8}, and {4, 6, 7, 8}. An additional parameter is defined to allow the merging of dense regions with high percentage of overlapping nodes. Given a threshold *minoverlap*, a pair of dense regions $(D_1, D_2)$ are merged if $\#(D_1 \cap D_2)/\#D_1 \geq minoverlap$

or $\#(D_1 \cap D_2)/\#D_2 \geq minoverlap$. The merging process is repeated iteratively until no more merging is allowed. By specifying $minoverlap = 0.7$, the five dense regions are merged and the final clusters are $\{$1, 2, 10, 11, 12$\}$, $\{$3, 4, 9, 10$\}$, $\{$4, 5, 6, 7, 8$\}$.

The graph-theoretic clustering algorithm allows a flexible clustering process. By specifying the four thresholding parameters *minassociation*, *mincompactness*, *minsize*, and *minoverlap*, one can control the compactness, interrelatedness, size, and number of clusters. It also allows the clusters to share common data points. Our implementation allows the user to specify the desired number of clusters by automatically updating the *minsize* parameter. It is also possible to update other parameters to achieve the same effect but this is less desirable.

To apply the graph-theoretic clustering algorithm to partition the data instances into clusters, we represent each data instance in the training set as a data point (node) in the feature space. The next step is to generate the adjacency graph for the data based on the predefined relation. Let $\mathbf{X}$ be a set of training instances and $a_l(\mathbf{x}_i)$ denote the value of the $lth$ feature of instance $\mathbf{x}_i$, for features with continuous value, we define the parameter *closeness* between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ with respect to feature $l$ as

$$Close_l(\mathbf{x}_i, \mathbf{x}_j) = \frac{|a_l(\mathbf{x}_i) - a_l(\mathbf{x}_j)|}{\max'(a_l(\mathbf{X})) - \min'(a_l(\mathbf{X}))}$$

where $\max'$ and $\min'$ are the maximal and minimal functions excluding the values of outliers in $\mathbf{X}$. A binary feature relation $FR_l(\mathbf{x}_i, \mathbf{x}_j)$ exists for the data instances $\mathbf{x}_i$ and $\mathbf{x}_j$ with respect to feature $l$ is defined as

$$\begin{aligned} &\text{discrete :} \\ &FR_l(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } a_l(\mathbf{x}_i) = a_l(\mathbf{x}_j), \\ 0 & \text{otherwise.} \end{cases} \\ &\text{continuous :} \\ &FR_l(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } Close_l(\mathbf{x}_i, \mathbf{x}_j) \leq \delta, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{1}$$

where $\delta$ is the maximal range for two feature values considered to be close. For data instances $\mathbf{x}_i$ and $\mathbf{x}_j$, a binary relation $R_{i,j}$ is defined as

$$R_{i,j} = \begin{cases} 1 & \text{if } \sum_l FR_l(\mathbf{x}_i, \mathbf{x}_j) \geq N, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where $N$ is the minimal number of features that have to satisfy the binary feature relation criterion. The clustering algorithm proceeds after the adjacency graph is computed. Like the two clustering algorithms mentioned before, the graph-theoretic clustering is based solely on the feature vectors. The advantage of this clustering algorithm is that it allows the borderline data instances to be trained for different component classifiers. Therefore the decision boundaries won't be skewed by a specific component classifier in the combining stage. Some of the data instances may not be included in any of the clusters generated by the graph-theoretic clustering algorithm due to the lack of sufficient relations. After the clusters are generated, these data instances will be added into the closest clusters according to the Euclidean distance between the instance and the centroid of the cluster.

## 3.2 Combining Classifiers

The component classifiers are trained to learn the decision boundaries in each data cluster. Since we assign the sub-classes as the outputs, these decision boundaries are more detailed and specific than the ones generated for the original classes. The next step is to utilize the information we gathered, namely the decision boundaries, to form the final classification. Approaches that simply average the decision boundaries are described in section 3.2.1. A better approach that performs piece-wise smoothing on the combined boundaries is described in section 3.2.2. In section 3.2.3 we show that the decision boundaries can be selected effectively and intelligently to form the final decisions due to our unique data partitioning process.

### 3.2.1 Unweighted and Weighted Voting

The simplest way to utilize the results from multiple classifiers is by a voting process called the committee method. In the unweighted scheme, every classifier's output is treated equally and the final classification is simply based on a majority vote. If the classifiers are able to produce class-probability estimates other than a simple classification, an average class probability can be used to decide the outcome. Let $P(f_{C_m}(\mathbf{x}_i) = k)$ denote the probability of data instance $\mathbf{x}_i$ classified as class $k$ by the component classifier $C_m$ where $k \in 1, \dots, K$; the average class probability is $P(f(\mathbf{x}_i) = k) = \frac{1}{M}\sum_{m=1}^{M} P(f_{C_m}(\mathbf{x}_i) = k)$. The final classification for $\mathbf{x}_i$ is then the class with the highest average class probability.

For the weighted scheme, the output of each classifier is treated differently. A common method for weight selection is to assign the weights to be proportional to the accuracy of each individual classifier. Therefore the previous equation can be modified to become $P(f(\mathbf{x}_i) = k) = \sum_{m=1}^{M} W_m P(f_{C_m}(\mathbf{x}_i) = k)$. where $W_m$ is the weight for the output of classifier $m$. Other approaches to obtain the weights include applying learning algorithms to the outputs of component classifiers. A naive Bayes classifier can be used to learn the weights according to Bayes' Theorem.

### 3.2.2 Stacking Cross Validation

Instead of a simple voting procedure for the results from each classifier, an alternative approach is to train a second stage classifier (super-classifier) on the results. Unlike the weighted voting procedure, in which the weights are adjusted according to the overall accuracy of each classifier, a super-classifier can adjust the weights dynamically. Each classifier will have higher weights for the instances it correctly predicted than for those on which it failed. Furthermore, our near-disjoint sub-data set partitioning provides an easy and effective method for cross-validation. The entire data can act as a big cross-validation set for each component classifier, since only a small portion of the data set has been seen by each classifier. This will reduce the over-fitting effect during the training stage.

The super-classifier approach can be recursively applied; that is, different learning algorithms or different settings can be utilized to learn the super-classifier. We can have a set of super-classifiers just as we have a set of component classifiers. Therefore we can have another stage, with a new super-classifier trained in the same way. However, in our experiments, the results usually stabilized after three levels of classifiers.

### 3.2.3 Gating Networks

In this section we describe the novel concepts for training the second-stage classifiers. Recall that the component classifiers are trained on the sub-data sets. Each sub-data set is a cluster of data in the feature space. A classifier trained on a particular region has higher discriminant ability on the data instances closer to the region than those that are far away from the region. Therefore it is natural to take into consideration the location of each data instance with respect to the data clusters.

In the data clustering process, we calculated some information about the clusters as well as the whole data set. The information includes the minimal and maximal values of each feature, the member instances of each cluster, and the means and variances of every features in each cluster. With this information, we can calculate a normalized distance between a data instance and the centroid of a cluster. Let $\mathbf{D} = \{D_m | m = 1 \ldots M\}$ denote a set of $M$ data clusters and $\mathbf{X} = \{\mathbf{x}_i | i = 1 \ldots N\}$ denote a set of $N$ training instances with $L$ features for each instance. The distance between cluster $D_m$

and instance $\mathbf{x}_i$ is

$$
\begin{aligned}
d(m,i) &= \sqrt{\sum_{l=1}^{L} fd_l} \\
\text{where} \quad & \text{continuous:} \quad fd_l = (\frac{me_{m,l} - a_l(\mathbf{x}_i)}{range_l})^2, \\
& \text{discrete:} \quad fd_l = \begin{cases} 0 & \text{if } me_{m,l} = a_l(\mathbf{x}_i), \\ 1 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{3}
$$

In the equation, $me_{m,l}$ denotes the mean of feature $l$ of cluster $D_m$, $a_l(\mathbf{x}_i)$ denotes the $lth$ feature value of instance $\mathbf{x}_i$, and $range_l = \max'(a_l(\mathbf{X})) - \min'(a_l(\mathbf{X}))$ denotes the value ranges of feature $l$ excluding the outliers.

From the normalized distances we compute a rank for each instance with respect to the clusters and the corresponding weights, which are inversely proportional to the distances. Both pieces of information are useful while training the super-classifier. In the previous section we described how different super-classifiers (second-stage classifiers) can be combined in the same way that we combine the component classifiers. By changing the content of the input data, we can train different super-classifiers for further incorporation. The rank is used to choose the input data from the outputs of the component classifiers and the strength of input data can be adjusted by the corresponding weights. Therefore we can learn different super-classifiers by changing the number of component classifiers used to generate the input data and the incorporation of weights. A parameter $k$ is used to determine the number of component classifiers utilized in the super-classifier training. The rank works as a gating switch as shown in Figure 4.

An important characteristic in this combination scheme is that different classification categories are used for different stages of learning. Usually a learning algorithm treats a multi-class problem as a recursive two-class problem. Since the decision boundaries can be very easily and reliably detected for a two-class problem, the learning algorithm can focus on finding the best dividing boundaries first, then break the data into separate regions and repeat the process. The advantage of this methodology is the simplicity and efficiency in dealing with a complicated problem. However, in the complex problem we studied, that methodology did not improve the accuracy of the overall system. Often the decision boundaries can be very accurately determined for the training data, but the same boundaries perform very badly for the test data. While we can adjust the learning algorithm not to over-fit on the training data, the truth is that usually the accuracy of both the training and test sets are sacrificed.

In our methodology, instead of simplifying the problem into a recursive two-class problem, we introduce the concept of sub-class relabeling. While it may seem that we have complicated the problem, in reality we can compensate for the shortcomings of the two-class methodology by keeping complicated decision boundaries and not degrading the performance on the test data. A presumption for the sub-class relabeling process is that there must exist a certain amount of data instances for each sub-class. We can consider the sub-class output as an intermediate representation for a data instance in a sense similar to the output of the hidden nodes in a back-propagation neural network. The function of the super-classifier can be considered as a piece-wise smoothing process to connect the boundaries constructed by different component classifiers. In our results we will demonstrate that the use of sub-class relabeling improves the classification accuracy for both the training data and the test data.

## 3.3 Error Instances Manipulation

One way to improve classification accuracy is to emphasize the learning process on difficult cases. In this work the difficult cases are data instances that are incorrectly classified by the component classifiers. They are also referred to as the error data set. Let $C_1, C_2, \ldots, C_M$ be a set of $M$ component classifiers. An error function is defined for a data instance $\mathbf{x}_i$ with respect to the classifier $C_m$ as $Err(\mathbf{x}_i) = \begin{cases} 1 & \text{if } f_{C_m}(\mathbf{x}_i) \neq y_i\ , \\ 0 & \text{otherwise.} \end{cases}$ where $y_i$ is the correct class for $\mathbf{x}_i$. The error data set can then be defined as $\mathbf{E} = \{\mathbf{x}_i \mid \sum_{m=1}^{M} Err(\mathbf{x}_i) \geq T,\ i \in 1, \ldots, N\}$ where $T$ is a threshold and $1 \leq T \leq M$. By adjusting the threshold we can change our scope for the difficult instances. When $T = 1$ the error set is the largest; since an instance is included in the error set if any one of the component classifiers misclassifies it. The set is smallest when $T = M$; that is, an instance is included only if none of the component classifiers can classify it correctly. If more than half of the component classifiers can correctly classify an instance, it is also likely to be correctly classified by the super-classifier, hence $T = M/2$ is a good choice of the threshold.

There are two ways to utilize the error data set. The first approach is to construct a classifier specifically tailored for the misclassified data instances. The learned classifier $C_E$ can then be treated as one of the component classifiers in the process of super-classifier construction. If we are able to identify instances that are prone to be misclassified, the addition of an error classifier will certainly have great improvement on the overall accuracy. Since we don't have this knowledge, the improvement may not be as significant. In the next section, we will discuss the approach for identifying the possible erroneous data instances.

The second approach for utilizing the error data set is to inject the error instances into the original data clusters. Each error instance is repeated multiple times in the data cluster to which it belongs. It forces a component classifi er to adapt its decision boundaries to accommodate the erroneous instances. There is no need to update the corresponding cluster centroid since the number of additional instances is small compared to the number of total instances in the cluster. Related to this approach is a learning algorithm called *boosting* [14].

## 3.4 Error Instance Detection

An important aspect in the classifi cation task is that when applying a classifi er to an unknown instance, the classifi er simply gives an answer. Although the answers given by some classifi ers can be interpreted probabilistically, the associated probabilities are usually produced according to the statistics of the training data to reflect the overall accuracy hypotheses. For example, for decision tree algorithms, the probability associated with the output class produced by the classifi er can be the classifi cation accuracy of the corresponding leaf node that gives the class estimate. For neural networks, the probability associated with an output class indicates the degree of confi dence for an unknown instance with which it believes that the unknown instance belongs to that class according to the training data. Usually, we have no way to judge if the output of an unseen example given by a classifi er is correct or not.

There are several reasons that we would like to know how a classifi er responds to an unknown example. We are interested not only in the classifi cation output given by the classifi er, but also in how confi dent we can be about the given output. The most obvious reason is to improve the performance of the classifi er. If we can determine which types of examples the classifi er predicts more accurately, we can apply the classifi er only on examples that are likely to be correctly classifi ed; thus its accuracy will increase. For data examples that do not respond well to a particular classifi er, we can either build another classifi er specifi cally for this type of examples or have human experts perform the classifi cation task. This is particularly useful in medical diagnosis problems where high-confi dence diagnoses are important.

The thrust of this section is to have an additional classifi er to try to identify the instances which might be misclassifi ed by the original classifi er. We call this type of instances the *suspicious instances*. The process is illustrated in Figure 6. For the training data, the original classifi ers (denoted as Super Classifi er 1 in the fi gure) will categorize them as either correctly classifi ed or incorrectly classifi ed (Error Instances in the fi gure). Each set is assigned with a new label and an *Error Instance Detection* classifi er is trained based on the new assigned classes. The ideal situation is that the *Error Instance Detection* classifi er will identify an unknown data instance as belonging to one of two sets: one which can be correctly classifi ed by the Super Classifi er 1 with high confi dence and the other which cannot be classifi ed with high confi dence. In the fi gure, they are denoted as "Training Group A" and "Training Group B" where

"Training Group A" is the set that is most likely be correctly classified by the original classifiers. For "Training Group B," we can construct another set of classifiers (denoted as super-classifier 2) to fit the data. The purpose of the *Error Instance Detection* classifier is to model the behavior of the original set of classifiers. For the testing part, the test data are first classified by the *Error Instance Detection* classifier into two groups, like the training data. One will be evaluated by Super Classifier 1 and the other will be either evaluated by Super Classifier 2 or by experts.

## 4 System Evaluation

To demonstrate our system's performance, we applied our methodology on the two data sets received from NeoPath in addition to a forest cover type data set from the UCI Knowledge Discovery in Databases Archive and originally from Colorado State University [4]. We denote them as NeoPath-1, NeoPath-2, and ForestCover respectively.

In data sets NeoPath-1 and NeoPath-2, data instances were acquired with either manually focused or automatically focused lens settings. The number of manual focus instances, auto focus instances, and extracted features are $<$15405, 3720, 323$>$ for NeoPath-1 and $<$20569, 3776, 291$>$ for NeoPath-2. As mentioned in Section 2, the data were prepared with the intention of correctly identifying the abnormal cases from a much larger pool of normal cases. Therefore, there are many more abnormal instances than normal instances in the data set, and most of the abnormal instances were acquired by manually focusing the lens. Since manual focusing produces better images, the extracted instances are less noisy. Two terminologies are used to describe the expected results, namely *sensitivity* - the percentage of abnormal cases classified as abnormal; and *specificity* - the percentage of normal cases classified as normal. Naturally the goal for the classifier is to achieve a high *sensitivity* and a high *specificity* simultaneously. The manually-focused examples are responsible for the *sensitivity* aspect of the designed classifiers, while the automatically focused instances are responsible for the *specificity* aspect.

### 4.1 NeoPath-1

The data set was first randomly divided into two groups: the training set and the test set. The class distributions for both the training set and test set were kept similar to the original class distribution. For each class, 60% of the instances were selected as training data and the remaining 40% as test data. The statistics of the data instance numbers are listed in Table 6.

Since the data set already has sub-classes, we adapted the inherent subclasses as the target outputs for the component classifiers. To construct the component classifiers, we have to cluster the training data first. A relational graph is constructed according to the aforementioned methodology for the training data. The graph-theoretic clustering algorithm is very time-consuming, if there are too many nodes (data instances) in the graph. The computation of the conditional density is on the order of $O(n^3)$ where $n$ is the number of nodes in the graph. To reduce the computational cost, we randomly selected 10% of the training data as the bootstrap set. The graph-theoretic clustering algorithm was applied to this bootstrap set. After the initial clusters were generated, the remaining training data instances were assigned to the closest cluster according to the normalized distance defined in Equation 3. For each cluster, we apply different learning algorithms including neural networks (NevProp), decision trees (C4.5), rule induction (CN2), and a hybrid method (RISE) [12] and selected the back-propagation neural network (NevProp) which had the best results as our backbone algorithm. The decision tree classifier C4.5 was also used occasionally for comparison purposes. In addition to the *sensitivity* and the *specificity* measurements, the accuracy of each abnormal sub-category was calculated.

Table 8 shows the performance of the component classifiers for a 10-cluster partition. The table shows the accuracy percentages of applying the component classifiers on both the training and test data sets. The numbers inside the parentheses in the first column are the number of data instances on which the classifier was trained. Table 9 shows the results for the same partition, but trained by the C4.5 decision tree classifiers.

The parameters of the graph-theoretic clustering algorithm allowed us to specify the desired number of clusters. After the clusters were generated, the training data were partitioned and each subset used to train a classifier. After the component classifiers were trained, a second stage super-classifier was trained on the outputs of the component classifiers. Table 10 and Table 11 show the results of super-classifiers with different settings. In the table, the notation "*X-X*" is used to indicate the formation of component classifiers and super-classifier where $X \in \{S, O\}$. $S$ indicates that the classifier was trained with the sub-classes as the target function; *O* means that the classifier was trained with the original classes as the target function. The first $X$ shows the class category of the component classifiers, while the second $X$ shows that of the super-classifier.

Figure 7 shows the comparisons between our results and NeoPath's results. Our results come from our automatically derived hierarchical classifiers, while NeoPath's results come from their proprietary classifiers which were trained with extensive interactions and fine-tunings by the human experts. The *specificity* and *sensitivity* measurements are calculated over the entire data set, including the training data and test data for comparison reasons. The dashed lines in the figure indicate the results obtained by NeoPath's classifiers. Figure 7 (a) shows the performance for various second stage super-classifi ers; (b) shows the results for injecting the error instance into the data clusters. The error instances, which were obtained according to the settings mentioned in section 3.3, were added to the closest $n$ data clusters and the component classifi ers were retrained. The results from several different clustering settings are shown. In fact, the results for 5, 10, and 14 clusters are not much different. However, when the number of clusters grows to 20, there is a signifi cant decrease in *specificity*. The reason for this phenomenon is that the number of *Normal* instances is far less than the number of *Abnormal* instances, so each of these clusters has only a few *Normal* examples. The decision boundaries for *Abnormal* instances can be properly induced, while the boundaries for *Normal* instances cannot, due to the lack of examples.

In the NeoPath system, the classifi er fi rst performs feature analysis procedures to compute the statistics of features and then selects a subset of features to train the classifi er. However, the process is not fully automatic and requires extensive interactions with human experts. The number of subset features they used is 74, compared to the fullest set of 323. We have mentioned before that our component classifi ers can also be used to perform feature selection tasks. In Figure 9, we show the results of our classifi ers by using the full set of features, NeoPath's subset of features, and the subset of features selected by the component classifi ers. We set the number of features selected by the component classifi ers equal to that of NeoPath's. When using the same subset of features, our classifi ers have almost identical results to NeoPath's. However, the subset of features selected by our component classifi ers performs better than NeoPath's subset of features. And it is clear that the full set of features provides more information and therefore increases the classifi er accuracy.

## 4.2 NeoPath-2

The second data set from NeoPath is similar to the fi rst data set with a slightly different feature set. It is labeled in the same fashion as the fi rst data set. The statistics of this set are listed in Table 7. We applied the same percentages to sample the training set and test set as we did for set 1. The training set is again clustered into 10 clusters. Figure 10 shows the results from the second stage and the third stage super-classifi ers. The weighted rank and the unweighted rank versions are shown in the fi gure. The weighted version is slightly better than the unweighted version, although the difference is small. The dashed lines again indicate the results from NeoPath's classifi ers. Figure 11 shows results from different clustering settings and Figure 12 shows the results for different feature subset settings. More of the results refer to [6]

## 4.3 ForestCover

The forest cover data set contains 581,012 instances, each with 54 features. The paper describing this work specifi es that the fi rst 11,340 records were used for the training set, the successive 3,780 records were used for validation, and the remaining 565,892 records were used for testing. The performance cited in the paper indicates 70% accuracy for backpropagation and 58% accuracy for linear discriminant analysis. The backpropagation network, NeuNet Pro, is a commercial neural network software package [8]. An evaluation version can be obtained from http://www.cormactech.com/neunet. The network was trained on a randomly-selected sample of 32,000 records of randomly-shuffled data and produced an accuracy of 68% on the remainder. We applied the C4.5 decision tree algorithm on the training set and adjusted the parameters according to the performance on the validation set, producing an accuracy of 63.64%. The same approach with the NevProp program produced an accuracy of only 23.96%. For the hierarchical multiple classifi ers, we again partitioned the training data into 10 clusters. The component classifi ers were trained based on the C4.5 algorithm. The validation data was incorporated in the training of the super-classifi er. This produced an accuracy of 70.81% on the test data.

## 4.4 Error Instance Detection

We did not want to construct the *Error Instance Detection* classifi er from the original input data due to the amount of data. Our hierarchical structures provide a remedy to this problem. The *Error Instance Detection* classifi er is trained based on the outputs of the component classifi ers. The super-classifi er can be thought of as an arbitrator to decide the fi nal outcome based on the outputs of the component classifi ers. Since its decision is solely based on the outputs of component classifi ers, it is natural for the *Error Instance Detection* classifi er to use the outputs of the component classifi ers as well.

Table 13 shows preliminary results on the NeoPath-1 data set. The original super-classifi er 1 executed with an accuracy of 83.35% and 73.14% for the training data and test data respectively. The *Error Instance Detection* classifi er separates both the training data and the test data into two groups: A and B. Group A is supposed to be correctly classifi er by super-classifi er 1, while group B is the error instance set. From the table, it can be seen that the classifi cation accuracy of super-classifi er 1 increases for the data in group A. Although the overall accuracy for data in group B by super-classifi er 2 also increases, the accuracy for the test data alone is not as good as with the original classifi er. This suggests that either super-classifi er 2 is over-trained or that some other solution, such as expert inspection, is needed to correctly classify the error instances.

## 4.5 Discussion

The purpose of the component classifiers is to specialize within a region of the data space. However, each component classifier should also maintain proper accuracy for data out of its range. Table 8 shows that although each component classifier was trained with a much smaller number of instances than the entire training set, the overall accuracy is acceptable. It can also be seen that most of the component classifiers do a better job of identifying the *Abnormal* instances, since most of the *sensitivity* measurements are higher than the *specificity* measurements. However, a classifier with extremely high *sensitivity* is not necessary a good thing if its *specificity* is very low. In this case, the classifier is just biased to favor the *Abnormal* instances. Table 9 (using C4.5 instead of NevProp) shows a similar trend to that of Table 8, but the decision tree results are not as good as those from the neural nets.

Table 10 and Table 11 record the results of the super-classifiers. Table 10 shows results in which the target function for both component- and super-classifiers was the sub-class label (S-S), while Table 11 shows results in which the target function for the component classifiers was the sub-class label and the target function for the superclassifier was the main class label (S-O). The most noticeable feature is that both of the *sensitivity* and the *specificity* measurements are much higher than those of the component classifiers. This shows that the super-classifier is actually performing the classification task. The accuracy achieved on the training set is expected to be much higher than for the test set. When sub-classes are used as the target function for the super-classifier, the *Normal* instances are better recognized than the *Abnormal* instances. This can be explained by the behavior of the classification algorithms. For any type of classifiers, the accuracy of a particular target class is affected by the number of training instances associated with that class. The more training instances a classifier has, the better it can predict for that class. Table 6 shows that there are 7,435 *Abnormal* instances and 4,100 *Normal* instances in the training set. There are 5 sub-classes for the *Abnormal* class, but only 2 for the *Normal* class. Therefore, the average number of training instances for each sub-class of the *Normal* class is higher than that of the *Abnormal* class. Since our component classifiers and super-classifier are trained for the sub-classes output in Table 10, this super-classifier performs better on the *Normal* instances. Table 6 also shows that the number of training instances for the sub-classes *Repair* and *Cancer* are about the same, but the results show a much better accuracy on recognizing *Cancer* than *Repair*. The reason is that a *Repair* cell is on the border of *Normal* and *Abnormal*. Damage to a cell can be caused by inflammation, atrophy with inflammation, radiation, intrauterine contraceptive device, or other reasons [27]. Although the inflammation-caused repair can be treated as normal, the radiation-caused repair is not. Therefore the *Repair* instances are categorized as *Abnormal* for further examination.

From Table 11 we can see that the accuracy of *sensitivity* increases quite a bit while still maintaining a good *specificity* accuracy. This demonstrates that the use of different target functions in different stages alleviates the uneven class distribution problem mentioned in the previous paragraph. We prefer the super-classifi ers in this setting for several reasons. The purpose of the screening process is to identify possible abnormalities in a pap smear. A high *sensitivity* is necessary to better protect the examinee, since the sample will be further examined. However, to keep the cost down, a high *specificity* is necessary to avoid unnecessarily re-examination. Therefore the classifi ers with the settings of Table 11 are superior to those of Table 10. Both tables show mixed results with the addition of error classifi ers.

Figure 7 shows the comparison of different results. In the 4 regions divided by the dashed lines, the upper-right region is our goal. In sub-fi gure (a), two of the settings perform better than NeoPath's results. Generally, the classifi ers with different target functions in different stages perform better and are close to or exceed the performance of NeoPath's classifi ers. Sub-fi gure (b) shows the results of injecting the error instances into the original clusters and retraining the component classifi ers. The error instances were added to the closest $n$ clusters. Different second and third stage super-classifi ers were constructed based on the newly trained component classifi ers, and the best results are illustrated in the fi gure. Generally, they are all pretty good results. Some are better than the original super-classifi er without the error instances injection. As $n$ gets larger, the error instances lose their effect on the component classifi ers. Since many of the error instances may be outside the range of a component classifi er, they can decrease the component classifi er's discriminating ability and the overall accuracy.

Figure 10 shows the results for the second NeoPath data set. In sub-fi gure (a) we compared the second stage super-classifi ers. When using the weighted rank, where the output of component classifi ers are not only selected by rank, but also are weighted according to the distance between the instance and the centroid of the corresponding data cluster, the weighted version is better than the unweighted version. Sub-fi gure (b) shows similar results for the third stage super-classifi ers. It can be seen that in this data set, our automatically-derived classifi ers perform much better than the original classifi er. The accuracy of both the *sensitivity* and specifi city increases by more than 5 percents.

Figure 11 shows the effect of different clusters on data set NeoPath-2. For the 3-cluster setting, while each component classifi er has more training data, it loses the characteristic of specializing the decisions in a compact region. Therefore the accuracy is not as good as the other two settings. Figure 12 also shows the effect of different sets of features on data set NeoPath-2. The full set of features provides the best results, but our classifi ers also perform better than the original classifi er with a reduced set of features. The weighted rank setting increases the accuracy, especially for the smaller set of features.

To demonstrate the consistency of the classifi cation accuracy, we selected the NeoPath-

2 data set and performed the hierarchical learning algorithm for 25 repetitions. In each repetition, a different random number was used to generate the training set and test set. In the clustering stage, the parameters were set so that 10 clusters were obtained for each training set. The parameters for the construction of component classifiers were identical for each repetition and so were the parameters for the construction of super-classifier. The super-classifier was constructed with $k = 3$ and unweighted distances. Table 14 shows the mean, variance, maximum and minimum values of these 25 experiments for test set classification accuracy, overall sensitivity and overall specificity. Figure 13 shows the spread of these 25 experiments.

## 5 Comparisons

In this section, we will compare the results of our classifiers with different clustering algorithms and the results of other multiple classifier algorithms.

### 5.1 Clustering

Three different clustering methods were used to construct the component classifiers, namely the random partition clustering, the K-means variant, and the graph-theoretic clustering algorithm. The experiment was evaluated on the NeoPath-2 data set. The parameters of each algorithm were set so that each algorithm generated 10 clusters for the training data. The average number of instances for each cluster is 1,467 for both the random partitioning and the K-means clustering algorithm, and 1,572 for the graph-theoretic clustering algorithm.

Figure 14 shows the results for the test data set from the three different clustering algorithms. The component classifiers were trained with a full set of 291 features. There were 7,007 abnormal instances and 2,666 normal instances in the test set. The figure shows that for overall accuracy, the graph-theoretic clustering is better than the other two algorithms. It is worth noting that the performance of the random partitioning is slightly better than the K-means clustering algorithm. The reason is that for the K-means clustering algorithm, each iteration will generate a new cluster by splitting the clusters previously generated. The clusters generated in later iterations may not have enough instances for the component classifiers to make a meaningful prediction. The fact that it is a “hard” clustering algorithm also prevents the borderline instances from being learned by more than one component classifier, thus reducing the accuracy of component classifiers and the super-classifiers.

Figure 15 shows the results for the same test data set in which the component classifiers are trained on a subset of 74 features. Again the graph-theoretic clustering produces the best results among the three algorithms. However, in this experiment the K-means algorithm performs better than the random partitioning algorithm. This shows that the reduced feature data have a bigger impact on the random partitioning algorithm than on the K-means clustering algorithm.

## 5.2 Multiple Classifiers

We also compare our results with two popular multiple-classifier algorithms *Bagging* and *Boosting* that had been demonstrated with very good performance on various data sets. The *Bagging* algorithm works according to the principle of manipulating the training data with sub-sampling. The *Boosting* algorithm [14] also manipulates the training data but in a different way. It emphasizes the training on the instances that are difficult to learn, that is, the instances that tend to be incorrectly classified. We applied the AdaBoost.M1 algorithm to construct the ensemble of classifiers.

For each replication of *Bagging* or each iteration of *Boosting*, the number of training instances is greater than or equal to that of the original training set. The memory required to carry out the training process of one replication or one iteration by *NevProp* exceeds what our system can provide (128M main memory and 128M swap memory). To accommodate the limitation of our system's resource and *NevProp*, we used only the 74 features provided by NeoPath in the experiments.

Figure 16 shows the results of *Bagging*, *Boosting* and our *Hierarchical* algorithms. There are 10 replications (bags) for the *Bagging* algorithm with an average of 63% of the original training instances kept in each replication. The *Boosting* algorithm converges after 6 iterations (error rate is greater than 0.5 after the 7th iteration); therefore, the final results are calculated according to the 6 classifiers constructed in 6 iterations. The figure shows that our approach performs better than the other two algorithms, furthermore, our algorithm requires much less time to train the classifiers. Figure 17 shows the *sensitivity-specificity* plot of multiple-classifier algorithms and stand-alone algorithms for comparison. The figure shows the measurements on the test data and overall data (including the training data). For stand-alone algorithms, the high *sensitivity* measurements on the overall data are caused by over-fitting the training data. The classification accuracies for the abnormal instances in the test data are *Hierarchical* $>$ *Bagging* $>$ *Boosting*, while the accuracies for normal instances are *Boosting* $>$ *Bagging* $>$ *Hierarchical*. Since there are about five times more abnormal instances than normal instances, the overall accuracies for the test data are *Hierarchical* $>$ *Bagging* $>$ *Boosting*. For more comparison results, refer to [6].

## 6 Conclusions

The multiple-classifier approach has been demonstrated to have equal or superior performance when compared to stand-alone classifiers. Usually the use of multiple classifiers means the increased consumption of system resources and longer training time. For complicated problems or problems with large amounts of data, this may pose some concerns or difficulties in the construction of classifiers.

In this paper, we described a hierarchical multiple-classifier classification scheme, which preserves the strength of the multiple-classifier approach and also manages to reduce some of the problems faced by other multiple-classifier algorithms. In our scheme, the system resource requirements are reduced (only a portion of the training data is needed for each component classifier) and so is the training time. From the results of the NeoPath data, it can be seen that our approach produces better performance than other multiple-classifier algorithms and the classifier produced by NeoPath's experts. Our approach also provides various parameters which allow the users to adjust the classifiers' behaviors to fit their needs. It is also possible to distribute the training of component classifiers into a network for parallel processing to further reduce the training time.

The cluster-based partition for training data is a key factor in designing the component classifiers. The graph-theoretic clustering algorithm we adapted shows very good results on NeoPath's data compared to the K-means-variant clustering algorithm. For different classification problems, other clustering algorithms may need to be investigated for better or faster clustering results. The main classification algorithms used in our study were the NevProp neural net and the C4.5 decision tree. Learning algorithms such as Vector Support Machines [35], Radial-Based Functions [13], Genetic Algorithms [30], and others may provide better performance for different applications. Our approach trains the super-classifier to incorporate the results given by the component classifiers, independent of their individual classification techniques. Other algorithms such as Bayesian networks and Expectation-Maximization algorithms [23] can also be used to combine the outputs of the component classifiers and may provide advantageous results for different applications.

Judging from the results, our classification scheme shows a promising performance with relatively low cost on the system resources and the required human interaction.

## References

[1] Ali, K. M. and Pazzani, M. J. Error Reduction Through Learning Multiple Descriptions. *Machine Learning*, 24(3):173–202, Sep 1996.

[2] Alimoglu, F. and Alpaydin, E. Combining Multiple Representations and Classifi ers for Pen-based Handwritten Digit Recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 637–640, Aug 1997.

[3] Almuallim, H. and Dietterich, T. G. Learning Boolean Concepts in the Presence of Many Irrelevant Features. *Artificial Intelligence*, 69(1–2):279–305, Sep 1994.

[4] Blackard, J. A. Forest Covertype data. ftp://kdd.ics.uci.edu/databases/covertype/covertype.data.gz, 1996. Remote Sensing and GIS Program, Department of Forest Sciences, College of Natural Resources, Colorado State University.

[5] Breiman, L. Bagging Predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[6] Chou, Y. *Hierarchical Multiple Classifier Learning System*. PhD thesis, University of Washington, 1999.

[7] Clark, P. and Niblett, T. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261–283, 1989.

[8] CorMac Technologies Inc. NeuNet Pro v2.1 for Windows. http://www.cormactech.com/neunet/, 1999.

[9] Davis, D. T., Hwang, J. N., and Lee, J. S. J. Improved network inversion technique for query learning application to automated cytology screening. In *Computer-Based Medical Systems. Proceedings of the Fourth Annual IEEE Symposium*, pages 313–320, 1991.

[10] Dietterich, T. G. Machine-Learning Research: Four Current Directions. *AI Magazine*, 18(4):97–136, 1997.

[11] Dietterich, T. G. and Bakiri, G. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[12] Domingos, P. Rule Induction and Instance-Based Learning: A Unifi ed Approach. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 1226–1232. International Joint Conferences on Artifi cial Intelligence, IJCAI-95, Aug 1995.

[13] Dybowski, R. Classifi cation of incomplete feature vectors by radial basis function networks. *Pattern Recognition Letters*, 19:1257–1264, 1998.

[14] Freund, Y.and Schapire, R. E. Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[15] Goodman, P., Rosen, D., Egbert, D., Carlson, D., Hallett, J, and Ju, W. Nevprop version 3. http://www.scs.unr.edu/∼cbmr/, 1998.

[16] Hansen, L. and Salamon, P. Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, Oct 1990.

[17] Haralick, R. M. and Shapiro, L. G. *Computer and Robot Vision*, volume 1, pages 137–138. Addison Wesley, 1992.

[18] Heng, M. H. Image Retrieval using Color and Texture. Technical Report, Department of Computer Science, University of Washington, 1996.

[19] Ho, T. K. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, Aug 1998.

[20] Hu, Y. H., Knoblock, T., and Park, J. M. Nonlinear Committee Pattern Classification. In *Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 568–577, Sep 1997.

[21] Hwang, J. N. and Lin, E. Mixture of discriminative learning experts of constant sensitivity for automated cytology screening. In *Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 152–161, 1997.

[22] John, G. H., Kohavi, R., and Pfleger, K. Irrelevant Features and the Subset Selection Problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, 1994.

[23] Jordan, M. I. and Xu, L. Convergence results for the EM Approach to Mixtures of Experts Architectures. *Neural Networks*, 8(9):1409–1431, 1996.

[24] Kaynak, C. and Alpaydin, E. Multistage Classification by Cascaded Classifiers. In *Proceedings of the 1997 IEEE International Symposium on Intelligent Control*, pages 95–100, Jul 1997.

[25] Kim, J., Seo, K., and Chung, K. A Systematic Approach to Classifier Selection on Combining Multiple Classifiers for Handwritten Digit Recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 459–462, Aug 1997.

[26] Koller, D. and Sahami, M. Toward Optimal Feature Selection. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 284–292, 1996.

[27] Kurman, R. J. and Solomon, D. *The Bethesda System for Reporting Cervical/Vaginal Cytologic Diagnoses*. Springer, 1994.

[28] Lee, J. S. J., Bannister, W. I., Kuan, L. C., Bartels, P. H., and Nelson, A. C. A Processing Strategy for Automated Papanicolaou Smear Screening. *Analytical and Quantitative Cytology and Histology*, 14(5):415–425, Oct 1992.

[29] Maclin, R. and Shavlik, J. W. Combining the Predictions of Multiple Classifi ers: Using Competitive Learning to Initialize Neural Networks. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 1, pages 524–530, Aug 1995.

[30] Mitchell, T. M. *Machine Learning*, pages 249–270. McGraw-Hill, 1997.

[31] Patten, Jr., S. F., Lee, J. S. J., and Nelson, A. C. Neopath, Inc. NeoPath AutoPap 300 Automatic Pap Screener System. *Acta Cytologica*, 40(1):45–52, Jan–Feb 1996.

[32] Quinlan, J. R. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993.

[33] Schapire. R. E. Using Output Codes to Boost Multiclass Learning Problems. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 313–321, 1997.

[34] Shapiro, L. G. and Haralick, R. M. Decomposition of Two-Dimensional Shapes by Graph-Theoretic Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):10–20, Jan 1979.

[35] Vapnik, V. N. *Statistical Learning Theory*. John Wiley & Sons, Inc, 1998.

## List of Figures

## List of Tables

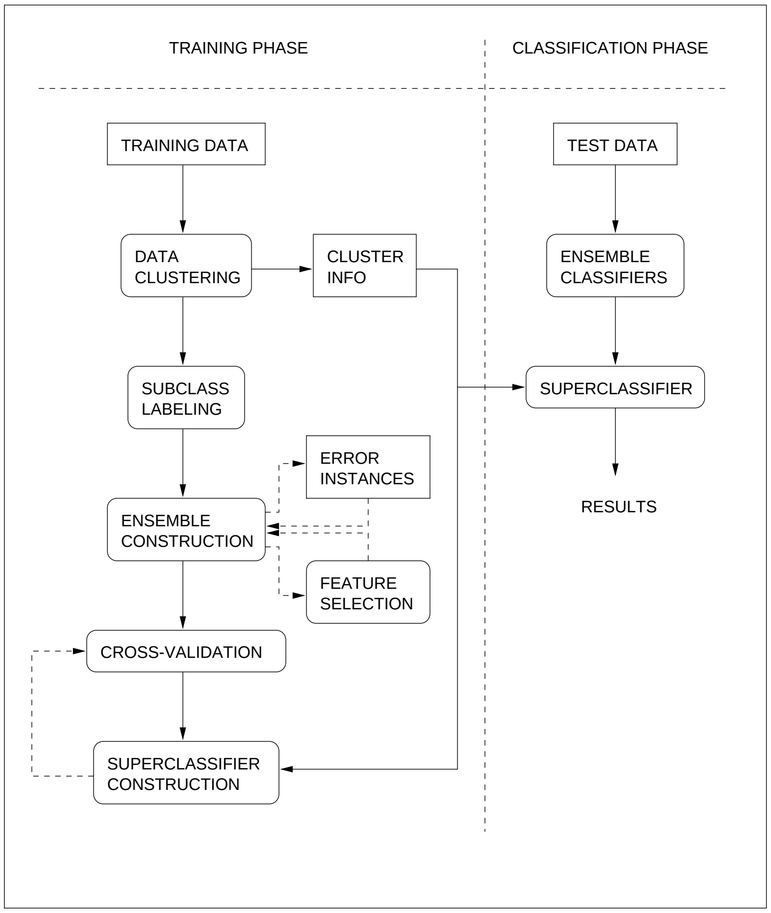


Figure 1: A block diagram of developed approaches in this paper.

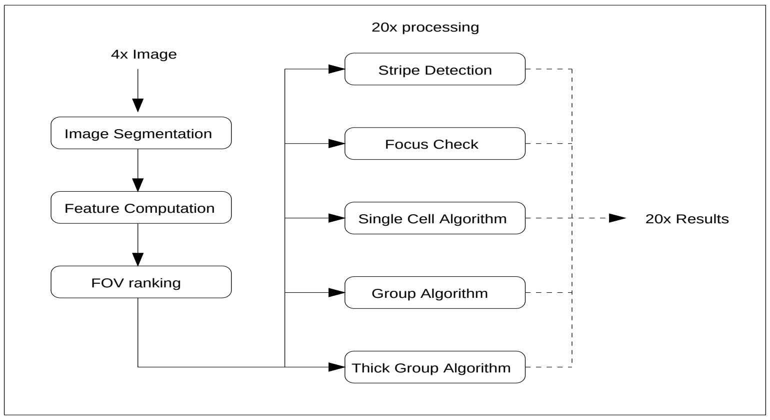


Figure 2: The architecture of NeoPath's AutoPap system

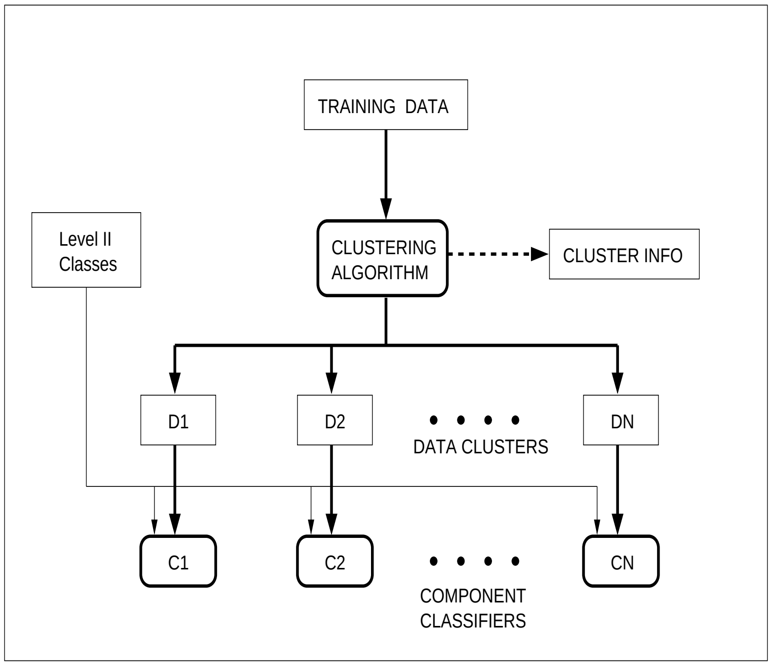


Figure 3: A block diagram of data clustering and component classifi er training.

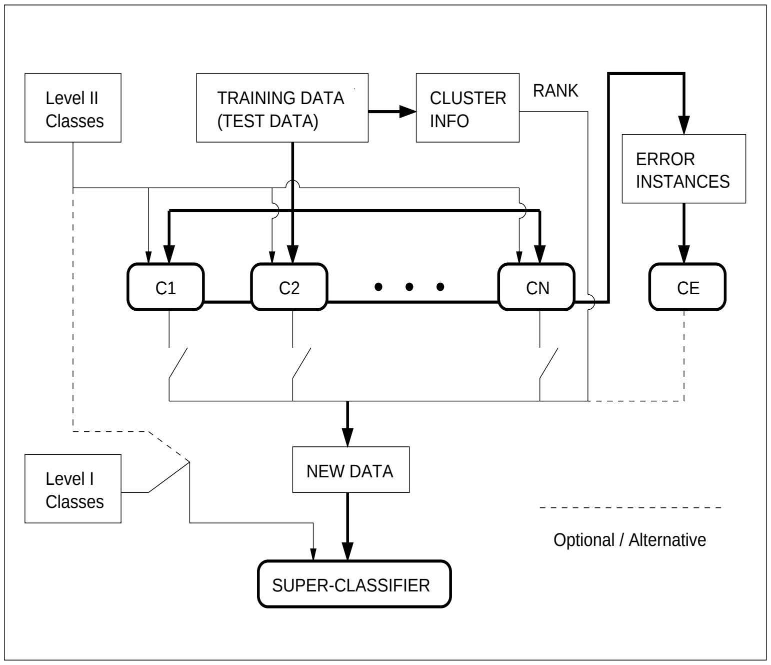
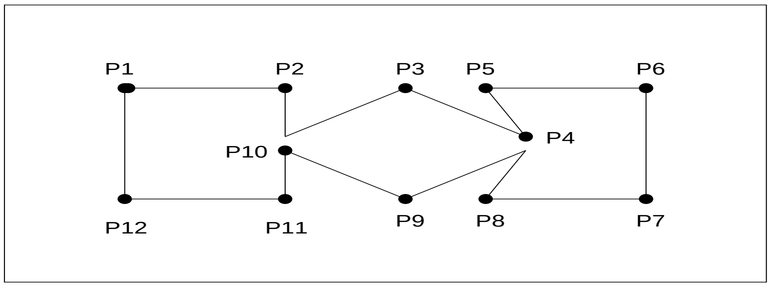
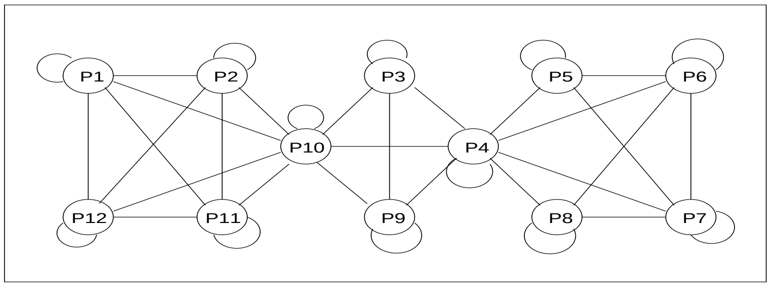


Figure 4: A block diagram of the combination scheme.



(a) simple polygonal shape



(b) corresponding relational graph

Figure 5: A simple example of a decomposable polygonal shape.

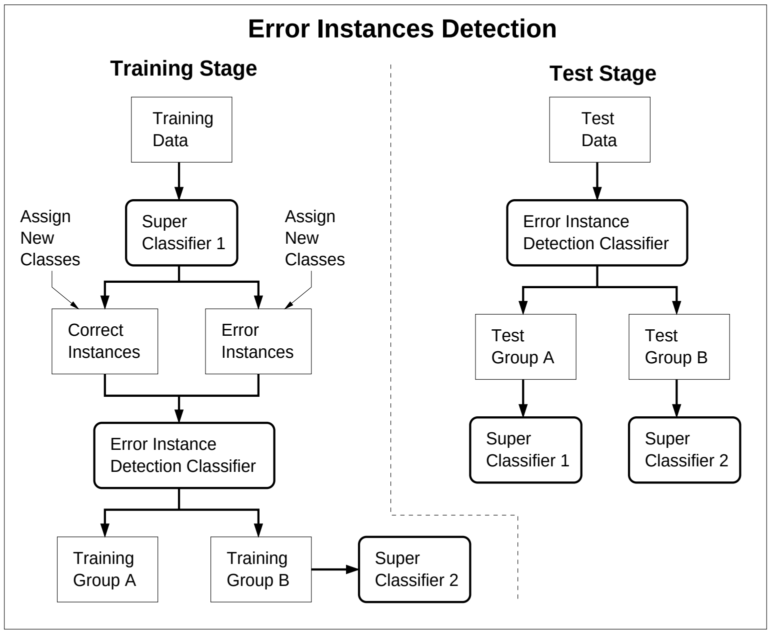
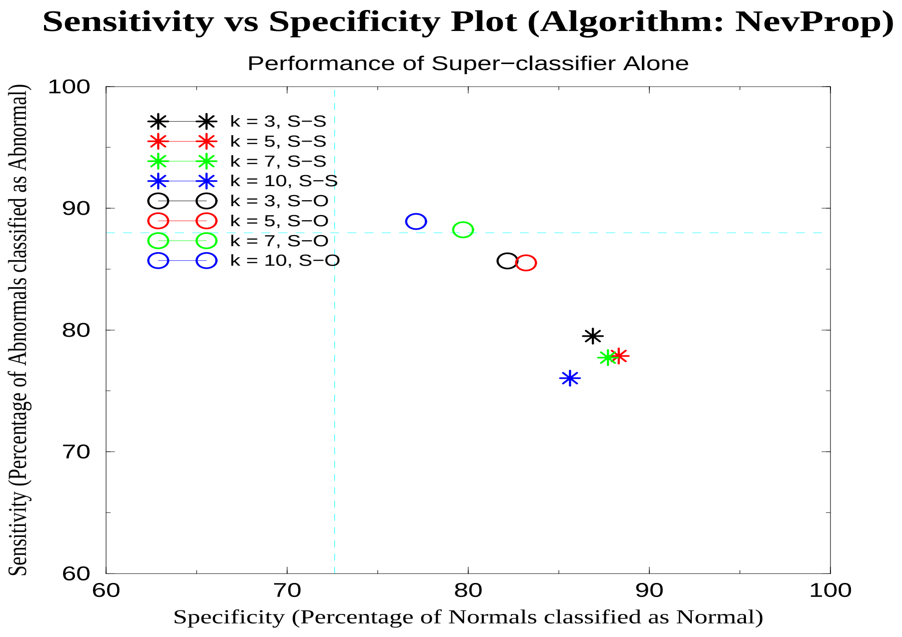
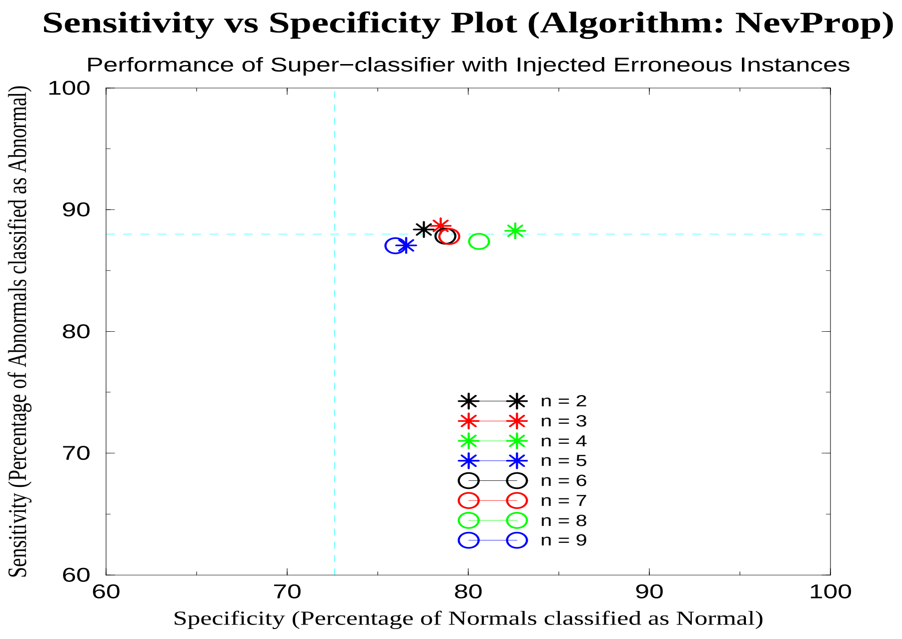


Figure 6: A block diagram for the error instances detection process.



(a) Second stage super-classifi ers



(b) Erroneous instances injection

Figure 7: Comparison for different settings of super-classifi ers on NeoPath-1.

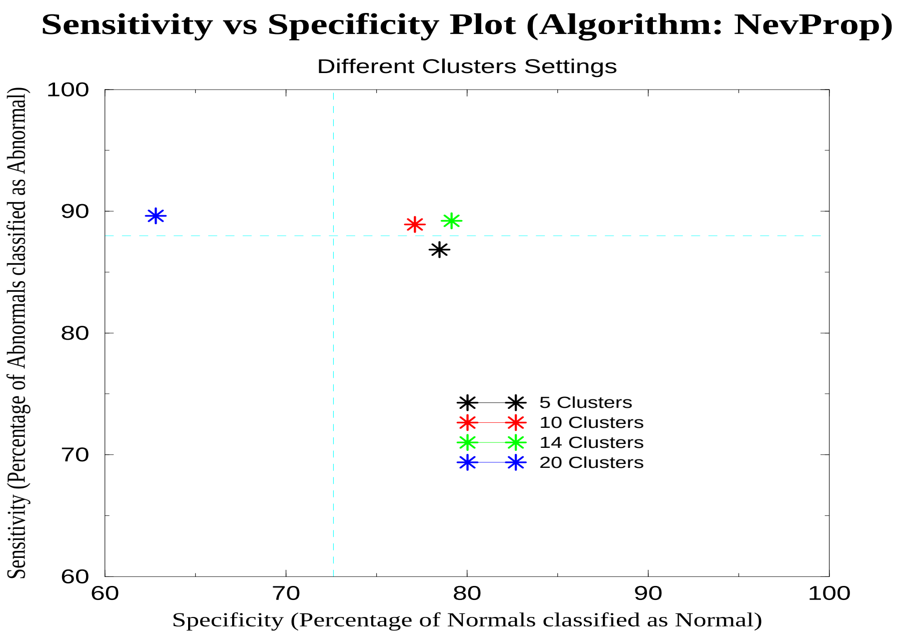


Figure 8: Comparison for different numbers of clusters on NeoPath-1.

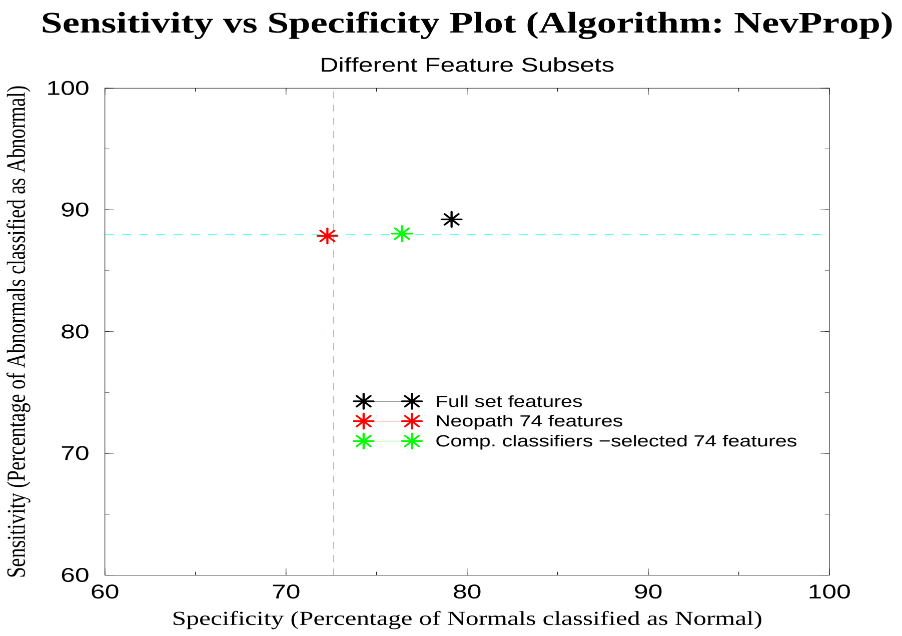
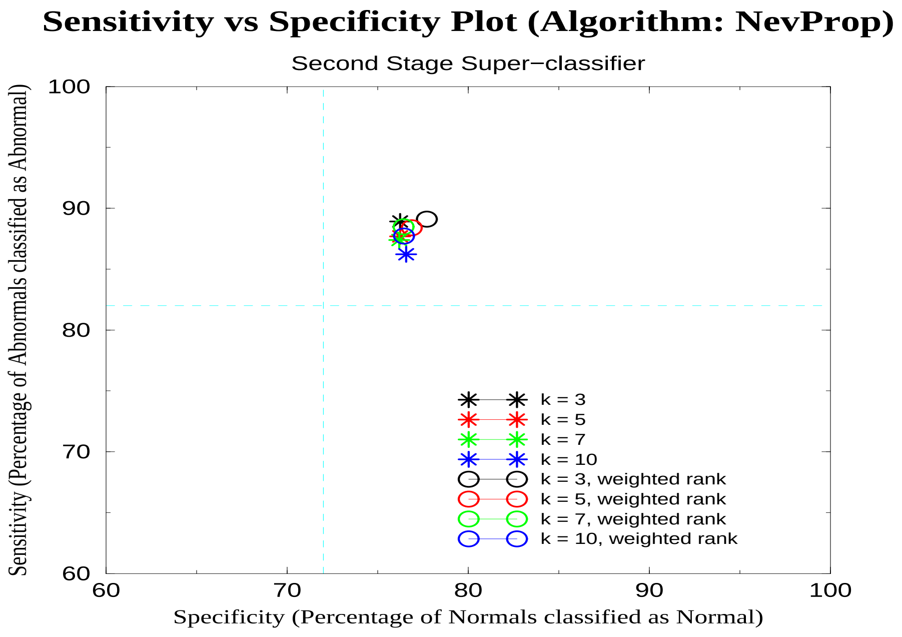
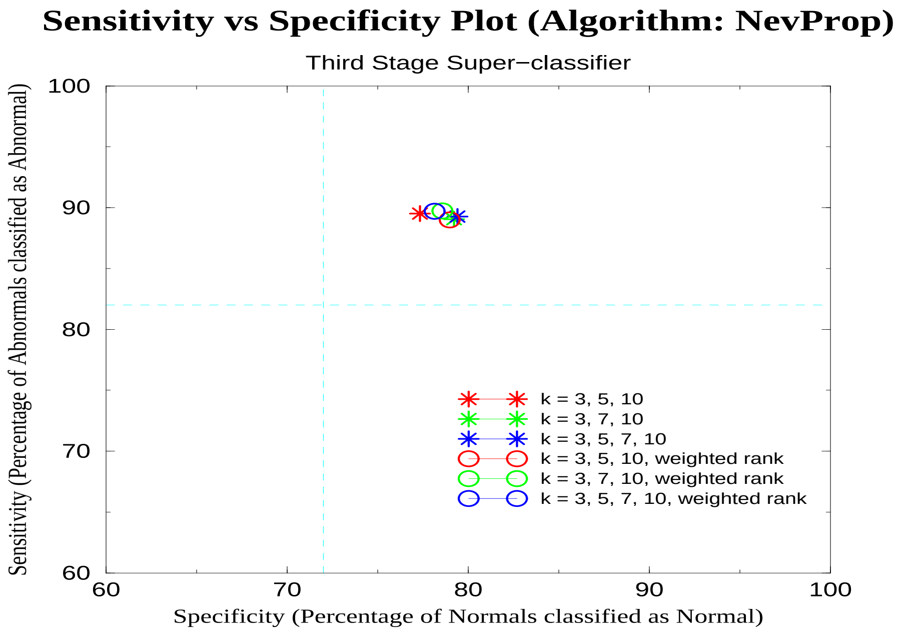


Figure 9: Comparison for different attribute subsets on NeoPath-1.



(a) Second stage super-classifi ers



(b) third stage super-classifi er

Figure 10: Comparison for different settings of super-classifi ers on NeoPath-2.

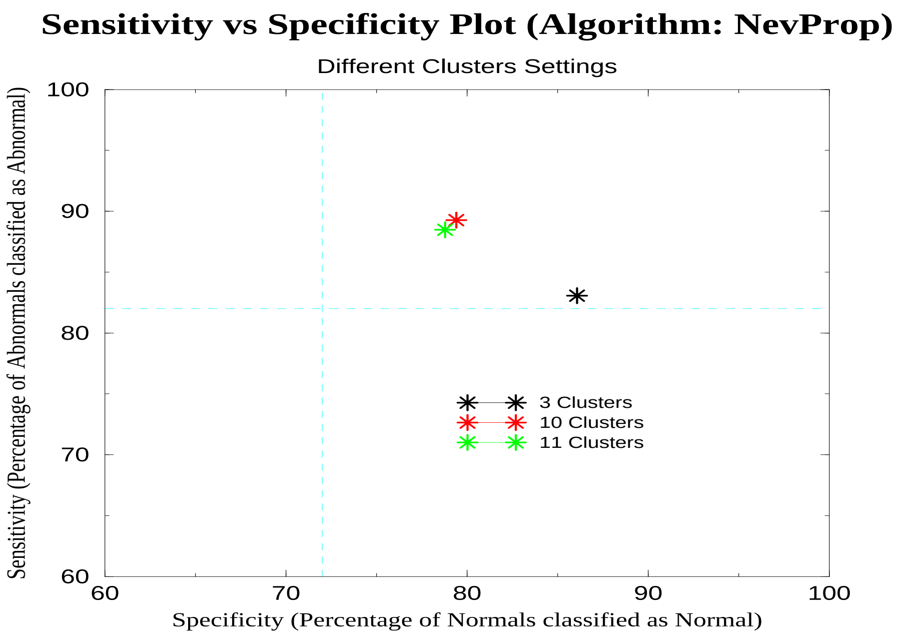


Figure 11: Comparison for different number of clusters settings on NeoPath-2.

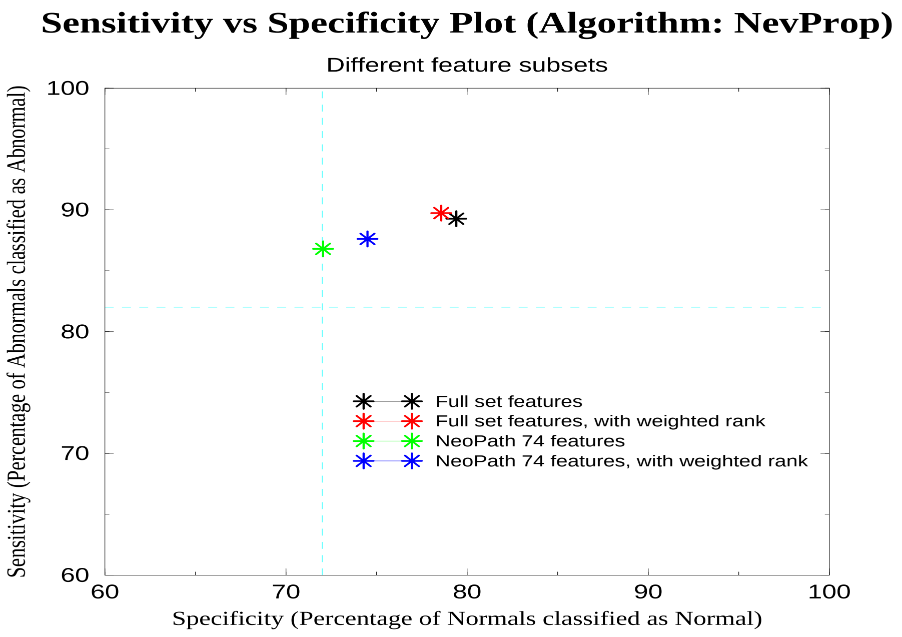


Figure 12: Comparison for different attribute subsets on NeoPath-2.

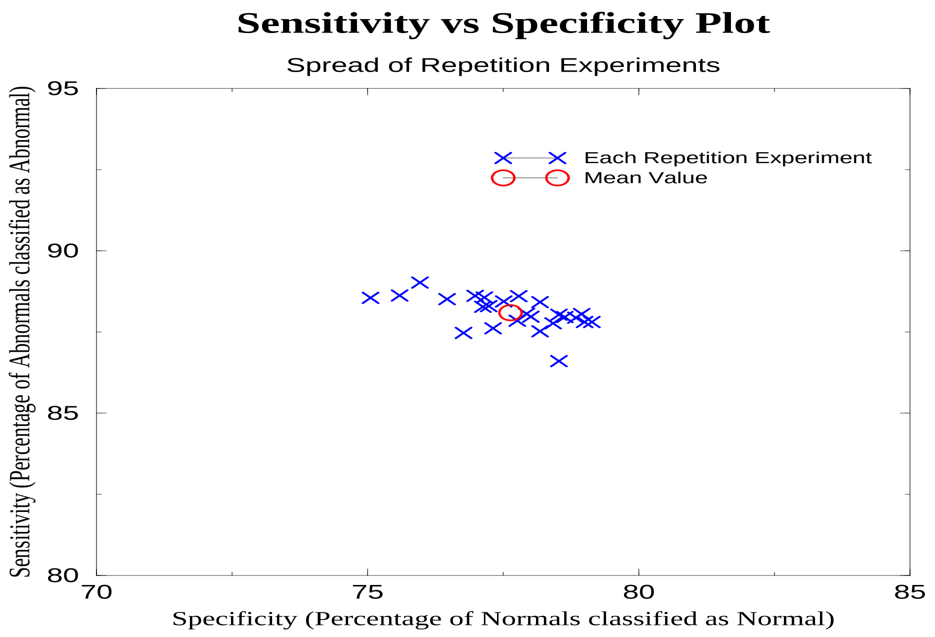


Figure 13: Spread of 25 repetition experiments on NeoPath-2.

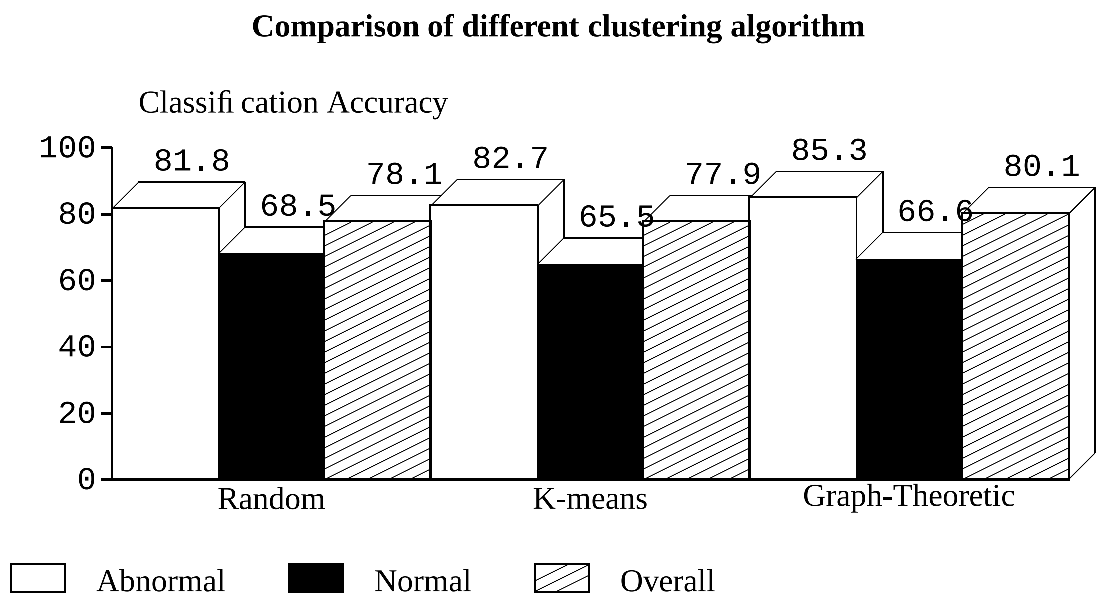


Figure 14: Classification accuracy of the test data set for three different clustering algorithms with a full set of 291 features.

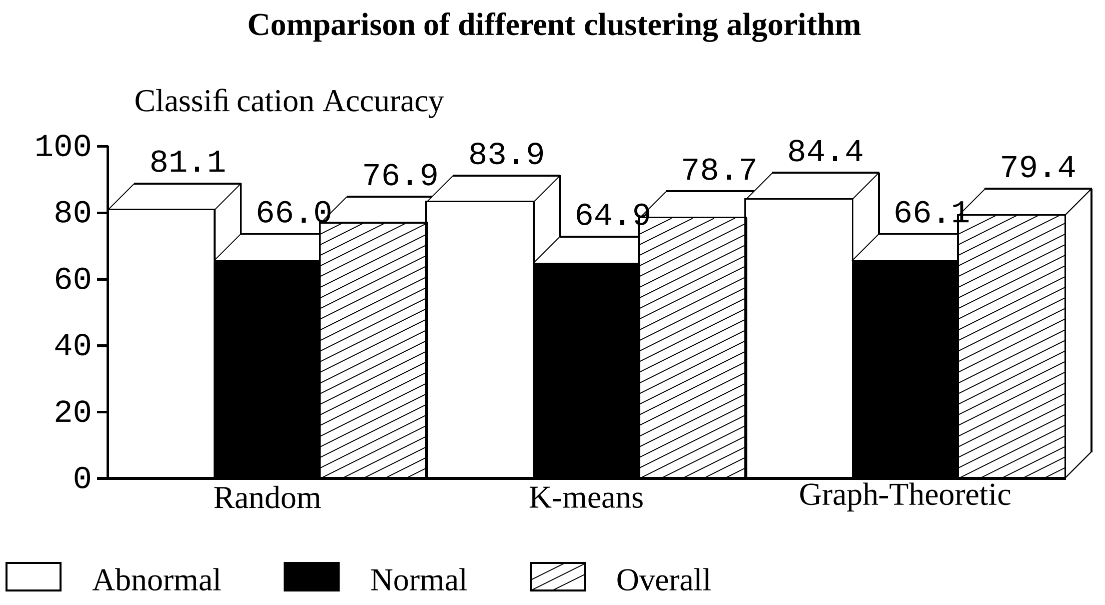


Figure 15: Classification accuracy of the test data set for three different clustering algorithms with a subset of 74 features.

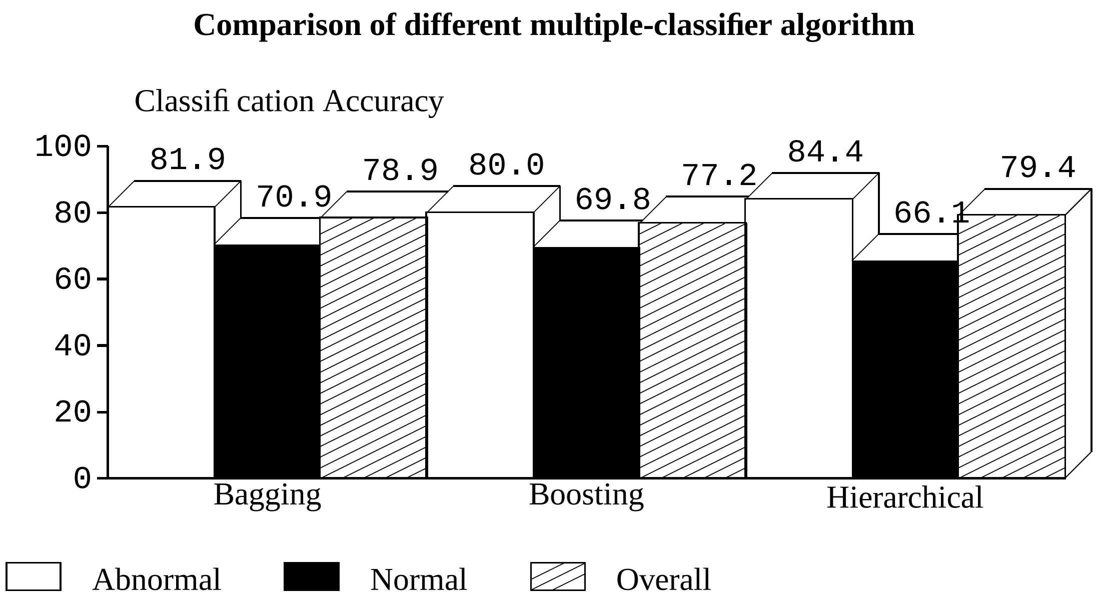


Figure 16: Classification accuracy of the test data set for three different multiple-classifier algorithms with a subset of 74 features.

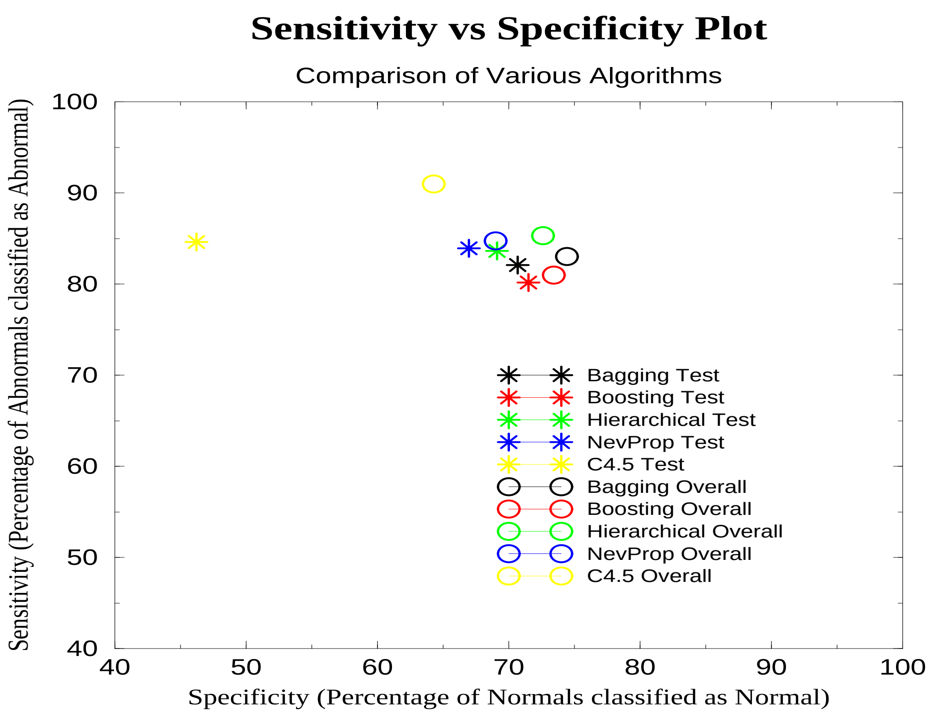


Figure 17: Comparison for different algorithms on NeoPath-2.

Table 1: Class defi nitions for the fi rst two levels.

| *First* | *Second* |
|---|---|
| *Normal* | *Squamous* |
| | *Glandular* |
| | *Other* |
| | *Normal Epithelial* |
| | *Cellular Changes (Repair)* |
| *Abnormal* | *Atypical Squamous (Ascus)* |
| | *Low-grade Squamous Intraepithelial Lesions (LSIL)* |
| | *High-grade Squamous Intraepithelial Lesions (HSIL)* |
| | *Squamous Carcinoma (Cancer)* |
| | *Atypical Glandular* |
| | *Adenocarcinoma* |
| | *Malignant NOS* |
| | *Mixed Abnormal Cells* |
| *Artifact* | *Cellular Artifact* |
| | *Non cellular Artifact* |
| | *Focus Artifact* |

Table 2: Class distributions.

| *First* | *Second* | *NeoPath-1* | *NeoPath-2* |
|---|---|---|---|
| *Abnormal* | *Ascus* | *2625* | *5024* |
| | *LSIL* | *2732* | *3443* |
| | *HSIL* | *3968* | *3229* |
| | *Cancer* | *1533* | *3516* |
| | *Repair* | *1477* | *2404* |
| *Normal* | | *5040* | *3775* |
| *Artifact* | | *1750* | *2954* |

Table 3: Neighborhood nodes for the graph in Figure 5 (b).

| Node | Neighborhood of Node |
|---|---|
| 1 | 1, 2, 10, 11, 12 |
| 2 | 1, 2, 10, 11, 12 |
| 3 | 3, 4, 9, 10 |
| 4 | 3, 4, 5, 6, 7, 8, 9, 10 |
| 5 | 4, 5, 6, 7 |
| 6 | 4, 5, 6, 7, 8 |
| 7 | 4, 5, 6, 7, 8 |
| 8 | 4, 6, 7, 8 |
| 9 | 3, 4, 9, 10 |
| 10 | 1, 2, 3, 4, 9, 10, 11, 12 |
| 11 | 1, 2, 10, 11, 12 |
| 12 | 1, 2, 10, 11, 12 |

Table 4: Conditional densities for the graph of Figure 5 (b).

| $D(X\|Y)$ | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $X\backslash Y$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 5 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 5 | 5 | 5 |
| 2 | 5 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 5 | 5 | 5 |
| 3 | 1 | 1 | 4 | 4 | 1 | 1 | 1 | 1 | 4 | 4 | 1 | 1 |
| 4 | 1 | 1 | 4 | 8 | 4 | 5 | 5 | 4 | 4 | 4 | 1 | 1 |
| 5 | 0 | 0 | 1 | 4 | 4 | 4 | 4 | 3 | 1 | 1 | 0 | 0 |
| 6 | 0 | 0 | 1 | 5 | 4 | 5 | 5 | 4 | 1 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 5 | 4 | 5 | 5 | 4 | 1 | 1 | 0 | 0 |
| 8 | 0 | 0 | 1 | 4 | 3 | 4 | 4 | 4 | 1 | 1 | 0 | 0 |
| 9 | 1 | 1 | 4 | 4 | 1 | 1 | 1 | 1 | 4 | 4 | 1 | 1 |
| 10 | 5 | 5 | 4 | 4 | 1 | 1 | 1 | 1 | 4 | 8 | 5 | 5 |
| 11 | 5 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 5 | 5 | 5 |
| 12 | 5 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 5 | 5 | 5 |

Table 5: The values of $M$, $Z(M)$, and $C(Z(M))$.

| Node $N$ | $M$ | $Z(N)$ | $C(Z(N))$ |
|---|---|---|---|
| 1 | 5 | 1, 2, 10, 11, 12 | 1.00 |
| 2 | 5 | 1, 2, 10, 11, 12 | 1.00 |
| 3 | 4 | 3, 4, 9, 10 | 1.00 |
| 4 | 4 | 3, 4, 5, 6, 7, 8, 9, 10 | 0.59 |
| 5 | 4 | 4, 5, 6, 7 | 1.00 |
| 6 | 4 | 4, 5, 6, 7, 8 | 0.92 |
| 7 | 4 | 4, 5, 6, 7, 8 | 0.92 |
| 8 | 4 | 4, 6, 7, 8 | 1.00 |
| 9 | 4 | 3, 4, 9, 10 | 1.00 |
| 10 | 5 | 1, 2, 10, 11, 12 | 1.00 |
| 11 | 5 | 1, 2, 10, 11, 12 | 1.00 |
| 12 | 5 | 1, 2, 10, 11, 12 | 1.00 |

Table 6: Data instances statistics of the NeoPath-1 data set.

| Set | Focus | Abnormal | Normal[a] | Repair | Ascus | LSIL | HSIL | Cancer |
|---|---|---|---|---|---|---|---|---|
| Train | Manual | 7380 | 1882 | 840 | 2388 | 1577 | 1642 | 933 |
| | Auto. | 55 | 2218 | 53 | 2 | 0 | 0 | 0 |
| Test | Manual | 4860 | 1283 | 545 | 1577 | 1048 | 1090 | 600 |
| | Auto. | 40 | 1407 | 39 | 1 | 0 | 0 | 0 |
| All | Manual | 12240 | 3165 | 1385 | 3965 | 2625 | 2732 | 1533 |
| | Auto. | 95 | 3625 | 92 | 3 | 0 | 0 | 0 |

[a] Includes Normal and Artifact.

Table 7: Data instances statistics of the NeoPath-2 data set.

| Set | Focus | Abnormal | Normal[a] | Repair | Ascus | LSIL | HSIL | Cancer |
|---|---|---|---|---|---|---|---|---|
| Train | Manual | 10526 | 1839 | 1368 | 3019 | 2069 | 1944 | 2126 |
| | Auto. | 83 | 2224 | 80 | 3 | 0 | 0 | 0 |
| Test | Manual | 6943 | 1261 | 892 | 2002 | 1374 | 1285 | 1390 |
| | Auto. | 64 | 1405 | 64 | 0 | 0 | 0 | 0 |
| All | Manual | 17469 | 3100 | 2260 | 5021 | 3443 | 3229 | 3516 |
| | Auto. | 147 | 3629 | 144 | 3 | 0 | 0 | 0 |

[a] It includes Normal and Artifact.

Table 8: Accuracy percentage of component classifi ers with NevProp.

| | | | Sub-classes | | | | |
|---|---|---|---|---|---|---|---|
| Classifi ers | Sensitivity | Specifi city | Repair | Ascus | LSIL | HSIL | Cancer |
| Training Set of NeoPath-1 | | | | | | | |
| 1 ( 799[a]) | 77.2 | 52.8 | 70.1 | 77.9 | 78.7 | 78.9 | 75.0 |
| 2 (1549) | 81.4 | 58.7 | 75.1 | 82.1 | 81.9 | 81.0 | 83.6 |
| 3 ( 524) | 75.5 | 47.4 | 70.7 | 75.7 | 75.2 | 75.9 | 78.8 |
| 4 (1613) | 65.5 | 68.3 | 58.0 | 68.8 | 69.6 | 68.0 | 51.3 |
| 5 ( 473) | 66.4 | 51.9 | 65.8 | 70.4 | 68.0 | 64.5 | 55.8 |
| 6 ( 558) | 76.1 | 51.3 | 74.0 | 75.3 | 77.7 | 78.3 | 72.2 |
| 7 (2081) | 80.5 | 68.8 | 66.6 | 82.5 | 81.7 | 81.4 | 81.7 |
| 8 (1866) | 69.9 | 69.4 | 63.9 | 73.0 | 72.5 | 69.3 | 62.6 |
| 9 (1871) | 72.5 | 72.7 | 57.6 | 74.7 | 73.5 | 72.5 | 77.1 |
| 10 ( 887) | 84.4 | 39.5 | 79.1 | 84.6 | 86.1 | 87.0 | 79.8 |
| Test Set of NeoPath-1 | | | | | | | |
| 1 | 75.3 | 49.0 | 68.0 | 75.8 | 75.5 | 76.7 | 76.5 |
| 2 | 79.8 | 57.3 | 75.9 | 79.4 | 79.9 | 79.1 | 84.3 |
| 3 | 75.2 | 44.7 | 72.1 | 74.3 | 75.9 | 74.5 | 79.3 |
| 4 | 63.1 | 66.2 | 54.8 | 65.9 | 67.3 | 63.8 | 53.8 |
| 5 | 65.5 | 51.9 | 63.4 | 66.6 | 67.5 | 65.8 | 59.7 |
| 6 | 74.5 | 49.5 | 68.0 | 75.0 | 76.7 | 73.2 | 76.2 |
| 7 | 77.4 | 65.7 | 68.5 | 78.9 | 79.0 | 76.4 | 77.8 |
| 8 | 67.4 | 67.3 | 60.1 | 68.3 | 71.7 | 66.9 | 63.2 |
| 9 | 69.9 | 72.9 | 64.4 | 71.4 | 68.1 | 68.4 | 74.5 |
| 10 | 83.3 | 37.1 | 78.9 | 84.5 | 83.8 | 83.0 | 82.0 |

[a] number of instances utilized to train the classifi er

Table 9: Accuracy percentage of component classifi ers with C4.5.

| | | | Sub-classes | | | | |
|---|---|---|---|---|---|---|---|
| Classifi ers | Sensitivity | Specifi city | Repair | Ascus | LSIL | HSIL | Cancer |
| Training Set of NeoPath-1 | | | | | | | |
| 1 | 69.4 | 54.1 | 63.9 | 69.5 | 70.8 | 70.6 | 68.3 |
| 2 | 70.7 | 52.1 | 65.1 | 70.0 | 70.1 | 70.8 | 77.0 |
| 3 | 64.6 | 57.6 | 62.6 | 63.8 | 65.2 | 66.4 | 62.7 |
| 4 | 79.8 | 43.8 | 73.6 | 80.9 | 79.4 | 81.3 | 81.2 |
| 5 | 40.9 | 75.7 | 40.3 | 42.1 | 39.3 | 40.5 | 40.9 |
| 6 | 54.4 | 61.9 | 49.6 | 54.6 | 56.0 | 56.0 | 51.7 |
| 7 | 76.5 | 48.9 | 75.6 | 76.4 | 75.0 | 76.6 | 79.8 |
| 8 | 50.4 | 77.8 | 48.8 | 50.3 | 53.3 | 49.9 | 46.1 |
| 9 | 65.8 | 60.6 | 59.4 | 65.0 | 68.4 | 65.1 | 70.7 |
| 10 | 71.9 | 40.7 | 70.0 | 73.4 | 73.6 | 71.3 | 68.0 |
| Test Set of NeoPath-1 | | | | | | | |
| 1 | 66.6 | 51.1 | 61.5 | 66.5 | 69.1 | 66.1 | 66.3 |
| 2 | 67.7 | 44.3 | 61.5 | 67.5 | 67.7 | 67.8 | 73.0 |
| 3 | 75.7 | 36.9 | 67.6 | 76.1 | 75.4 | 77.6 | 78.0 |
| 4 | 75.7 | 36.9 | 67.6 | 76.1 | 75.4 | 77.6 | 78.0 |
| 5 | 38.8 | 72.7 | 35.1 | 38.8 | 38.8 | 40.5 | 38.2 |
| 6 | 52.4 | 59.2 | 47.8 | 53.4 | 54.3 | 54.9 | 45.3 |
| 7 | 71.4 | 39.9 | 70.5 | 69.8 | 71.1 | 70.7 | 77.3 |
| 8 | 45.4 | 72.9 | 38.5 | 46.3 | 47.9 | 44.7 | 45.2 |
| 9 | 62.1 | 56.6 | 58.4 | 60.8 | 60.7 | 64.2 | 66.5 |
| 10 | 70.3 | 37.6 | 67.6 | 71.0 | 71.5 | 71.1 | 67.3 |

Table 10: Accuracy percentage of super-classifi ers with NevProp.

| Classifi ers | | | | Sub-classes | | | | |
|---|---|---|---|---|---|---|---|---|
| formation[a] | $k^b$ | Sens. | Spec. | Repair | Ascus | LSIL | HSIL | Cancer |
| Training Set of NeoPath-1 | | | | | | | | |
| $S-S$ | $3$ | 83.8 | 90.9 | 63.6 | 87.9 | 85.8 | 85.6 | 83.3 |
| | $3+C_E$[c] | 84.2 | 90.8 | 62.9 | 88.4 | 86.2 | 86.3 | 83.8 |
| $S-S$ | $5$ | 81.4 | 91.5 | 59.9 | 84.9 | 83.8 | 82.7 | 82.9 |
| | $5+C_E$ | 82.0 | 91.0 | 60.5 | 86.0 | 82.6 | 83.1 | 86.4 |
| $S-S$ | $10$ | 78.6 | 88.2 | 64.5 | 80.1 | 80.8 | 78.5 | 81.9 |
| | $10+C_E$ | 79.7 | 86.6 | 64.4 | 81.7 | 81.7 | 79.0 | 83.6 |
| $S-S-S$ | $3,5,10$[d] | 86.3 | 91.7 | 72.3 | 87.9 | 88.7 | 87.1 | 87.5 |
| Test Set of NeoPath-1 | | | | | | | | |
| $S-S$ | $3$ | 73.0 | 80.5 | 61.3 | 73.3 | 75.3 | 73.2 | 76.0 |
| | $3+C_E$ | 73.4 | 79.7 | 61.8 | 73.2 | 76.0 | 73.6 | 76.8 |
| $S-S$ | $5$ | 72.5 | 83.2 | 58.7 | 73.4 | 74.9 | 72.9 | 75.0 |
| | $5+C_E$ | 73.1 | 82.9 | 59.9 | 73.5 | 74.6 | 73.2 | 78.5 |
| $S-S$ | $10$ | 72.1 | 81.4 | 61.8 | 73.7 | 71.8 | 70.3 | 78.3 |
| | $10+C_E$ | 72.5 | 80.5 | 61.3 | 73.0 | 73.8 | 70.6 | 79.3 |
| $S-S-S$ | $3,5,10$ | 75.0 | 78.7 | 63.5 | 75.8 | 76.7 | 74.1 | 79.0 |

[a] $X-X$: two-stage classifi ers (component classifi ers + super-classifi er); $X-X-X$: three-stage classifi ers (comp. + super. + super.); $X \in \{S,O\}$, $S$: sub-classes as target classifi cation; $O$: original classes as target classifi cation [b] Only the outputs from the $k$ closest classifi ers were utilized to construct the super-classifi er. [c] The output of the error classifi er was utilized in the construction of the super-classifi er as well as the outputs from the $k$ closest classifi ers. [d] The third stage super-classifi er was constructed by using the outputs of the second stage super-classifi ers as its input.

Table 11: Accuracy percentage of super-classifiers with NevProp.

| Classifiers | | | | Sub-classes | | | | |
|---|---|---|---|---|---|---|---|---|
| formation | $k$ | Sens. | Spec. | Repair | Ascus | LSIL | HSIL | Cancer |
| Training Set of NeoPath-1 | | | | | | | | |
| $S-O$ | $3$ | 89.5 | 87.8 | 79.8 | 91.1 | 90.7 | 90.6 | 88.5 |
| | $3+C_E$ | 90.7 | 87.7 | 80.6 | 92.7 | 91.6 | 91.4 | 90.4 |
| $S-O$ | $5$ | 88.9 | 88.3 | 80.0 | 89.9 | 90.2 | 89.5 | 89.2 |
| | $5+C_E$ | 91.8 | 86.6 | 84.4 | 92.7 | 92.5 | 92.4 | 91.7 |
| $S-O$ | $10$ | 92.0 | 82.3 | 86.0 | 91.5 | 93.8 | 92.0 | 93.7 |
| | $10+C_E$ | 91.9 | 83.5 | 85.4 | 92.1 | 93.0 | 91.7 | 93.4 |
| $S-S-O$ | $3,5,10$ | 90.1 | 90.0 | 79.1 | 91.8 | 91.2 | 91.5 | 89.4 |
| Test Set of NeoPath-1 | | | | | | | | |
| $S-O$ | $3$ | 79.8 | 73.2 | 69.7 | 81.0 | 81.2 | 79.2 | 81.7 |
| | $3+C_E$ | 80.4 | 72.5 | 72.3 | 80.6 | 82.1 | 79.6 | 82.7 |
| $S-O$ | $5$ | 80.3 | 75.1 | 71.1 | 80.5 | 81.8 | 79.7 | 83.7 |
| | $5+C_E$ | 81.4 | 72.6 | 72.4 | 82.1 | 82.3 | 81.0 | 84.0 |
| $S-O$ | $10$ | 84.1 | 68.9 | 75.2 | 84.9 | 84.7 | 83.4 | 87.7 |
| | $10+C_E$ | 82.9 | 69.4 | 71.7 | 83.2 | 84.7 | 82.6 | 87.0 |
| $S-S-O$ | $3,5,10$ | 79.6 | 74.1 | 67.5 | 80.5 | 80.6 | 79.5 | 83.7 |

Table 12: Performances of various algorithms for the forest cover type data.

| Algorithms | Accuracy |
|---|---|
| Linear Discriminant Analysis | 58% |
| Backpropagation | 70% |
| NevProp | 23.96% |
| C4.5 | 63.64% |
| NeuNet Pro SFAM | 68% [a] |
| Hierarchical Multiple Classifier | 70.81% |

[a] $\approx$ twice of the training records than the other algorithms.

Table 13: Results of error detection classifi er for NeoPath-1

| | Train (11,535) | | Test (7,590) | | |
|---|---|---|---|---|---|
| | Correct | Error | Correct | Error | Overall |
| Super-classifi er 1 | 83.35% | 16.65% | 73.14% | 26.86% | 79.29 |
| | | | | | |
| | Group A | Group B | Group A | Group B | |
| Error Detection | 9676 | 1859 | 6251 | 1339 | |
| | | | | | |
| Group A | Train (9,676) | | Test (6,251) | | Overall |
| Super-classifi er 1 | 98.42% | 1.58% | 73.73% | 26.27% | 88.73 |
| | | | | | |
| Group B | Train (1,859) | | Test (1,339) | | Overall |
| Super-classifi er 2 | 99.35% | 0.65% | 60.87% | 39.13% | 83.24 |

Table 14: Results of repetition experiments for NeoPath-2

| | Accuracy (Test Set) | Sensitivity | Specifi city |
|---|---|---|---|
| Sample Mean | 79.40 | 88.10 | 77.64 |
| Variance | 0.40 | 0.51 | 1.09 |
| Maximum | 80.15 | 89.03 | 79.14 |
| Minimum | 78.70 | 86.61 | 75.06 |