Contemporary Mathematics

Pattern Recognition Approaches to Solving Combinatorial Problems in Free Groups

Robert M. Haralick, Alex D. Miasnikov, and Alexei G. Myasnikov

Abstract. We review some basic methodologies from pattern recognition that can be applied to helping solve combinatorial problems in free group theory. We illustrate how this works with recognizing Whitehead minimal words in free groups of rank 2. The methodologies reviewed include how to form feature vectors, principal components, distance classifers, linear classifiers, regression classifiers, Fisher linear discriminant, support vector machines, quantizing, classification trees, and clustering techniques.

Free group, automorphism problem, Whitehead method, Pattern Recognition, Classification

Primary 20F28, Secondary 68T10

Contents

1. Introduction

There are many problems in mathematics that can be solved by a search over all possibilities. In combinatorial group theory every classical decision problem has its natural "search" variation. For example, the search version of the Word Problem for a group given by a presentation $\langle X \mid R\rangle$ would ask for a word u from the normal closure of the set of relators R to produce an expression of u as a product of conjugates of elements from $R^{\pm 1}$. The search Conjugacy Problem would require to produce a conjugating element, and the search Membership Problem would ask for an expression of a given u from a subgroup $\langle v_1, \dots, v_k\rangle$ as a product of the generators (and their inverses) $v_1^{\pm 1}, \dots, v_k^{\pm 1}$. In a free group finding a solution of a given consistent equation or determining whether or not a given element has minimal

length in its automorphic orbit (Whitehead minimality problem) are typical search problems. All these problems are recursively enumerable, so, in principal, the total search would produce an answer sooner or later. Unfortunately, in practice the total search could be extremely inefficient: there are not any recursive bounds on time complexity of the search word, conjugacy, or membership problems, and there is no algorithm known to find solutions of equations or determine minimality of a word in a free group in time better than exponential time in the size of the problem. In this paper, we offer some insights to speed up the solving of computational hard problems that arise in certain problem populations of free groups. We will show that with some intelligent reasoning, a significant fraction of problems in the population can be solved much quicker than exponential time.

The classical way algorithms are formulated in search problems is to design a way to perform the associated tree search and try to improve the solution time by fancier data structures, more efficient code, and by heuristics. Our point of view is to improve how we solve these problems using the experience of having solved many such problems from the given population. Thus our problem domain is given by a population of problems. We will sample many problem instances from the population and solve each instance. We will then examine the statistical characteristics in the tree search of how each problem instance was solved and then incorporate this statistical derived knowledge from the experience of solving these problems into a smarter tree search. There are situations in which we may not have to do any tree searching at all and our pattern recognition techniques will do all the work.

The use of this kind of experience may not improve the worst case complexity. But it can dramatically improve the computational complexity on a large fractions of problems from the population. For example, we may discover that 99 percent of the problems can be solved in linear time, 0.9 percent in quadratic time, and the remainder 0.1 percent in exponential time.

There are two dimensions to using this pattern recognition technology. The first dimension is the representation of any problem instance to a fixed dimensional feature vector which captures the information in the problem instance. The second dimension is the use of standard tools in pattern recognition that given the feature vector designates the most probable class or the most probable next successful step in the tree search.

In this paper we discuss briefly several typical pattern recognition techniques and demonstrate some of their applications by the Whitehead's minimality problem. *For notations and relevant results on Whitehead method we refer to the paper* [**4**].

2. General remarks on Pattern Recognition tasks

One of the main applications of Pattern Recognition (**PR**) techniques is classification of a variety of given objects into categories. Usually classification algorithms or *classifiers* try to find a set of measurements (properties, characteristics) of objects, called *features*, which gives a descriptive representation for the objects.

Generally, pattern recognition techniques can be divided in two principal types:

- *supervised learning;*
- *unsupervised learning* (*clustering*).

In supervised learning the decision algorithms are "trained" on a prearranged dataset, called *training* dataset in which each pattern is labelled with its true class

label. If such information is not available, one can use clustering. In this case clustering algorithms try to find *clusters* or "natural groupings" of the given objects. In this paper we use supervised learning pattern recognition algorithms.

Every pattern recognition task of the supervised learning type has to face all of the following issues:

(1) **Obtaining the data.** The *training datasets* can be obtained from the real world or generated by reliable procedures, which provide independent and representative sampled data.
(2) **Feature extraction.** The task of feature extraction is problem specific and requires knowledge of the problem domain. If such knowledge is limited then one may consider as many features as possible and then try to extract the most "significant" ones using statistical methods.
(3) **Model selection.** The model is the theoretical basis for the classifier. A particular choice of the model determines the basic design of the classifier (though there might be some variations in the implementation). Model selection is one of the most active areas of research in pattern recognition. Usually model selection is closely related to the feature extraction. In practice, one may try several standard models starting with the simplest ones or more economic ones.
(4) **Evaluation.** Evaluation of the performance of a particular **PR** system is an important component of the pattern recognition task. It answers the question whether or not the given system performs as required. To evaluate a system one can use various accuracy measures, for example, percentage of correct answers. To get reliable estimates other sets of data that are independent from the training sets must be used. Such sets are called *test datasets*.

    Typically we view a **PR** system as consisting of components 1)-4).
(5) **Analysis of the system.** Careful analysis of performance of a particular classifier may improve feature extraction and model selection. For example, one can look for an optimal set of features or for a more effective model. Moreover, through analysis of the most significant (insignificant) features one may gain a new knowledge about the original objects.

3. Feature Vectors

A feature vector is just a vector of properties about the object of interest, in our case about the mathematical object of interest. The properties are chosen to be ones thought to be relevant to the problem. We will illustrate the selection of features by the Whitehead minimal problem for a free group $F = F(X)$ with basis X.

Let w be a reduced word in the alphabet $\in X^{\pm 1}$. Below we describe features of w which characterize a certain placement of specific words from $F(X)$ in w.

Let $K \in \mathbb{N}$ be a natural number, $v_1, \dots, v_K \in F(X)$ be words from $F(X)$, and $U_1, \dots, U_{K+1} \subseteq F(X)$ be subsets of $F(X)$. Denote by

$$C(w, U_1 v_1 \dots v_K U_{K+1})$$

the number of subwords of the type

$$u_1 v_1 u_2 \dots v_K u_{K+1},$$

where $u_j \in U_j$, which occur in w. For fixed $K, v_1, \ldots, v_K, U_1, \ldots, U_{K+1}$ we obtain a *counting function*

$$w \in F \longrightarrow C(w, U_1 v_1 \ldots v_K U_{K+1}) \in \mathbb{N} \tag{1}$$

The normalized value

$$\frac{1}{|w|} C(w, U_1 v_1 \ldots v_K U_{K+1})$$

is called a *feature* of w and the function

$$w \in F \longrightarrow \frac{1}{|w|} C(w, U_1 v_1 \ldots v_K U_{K+1}) \in \mathbb{R}$$

is called a *feature function* on F. Usually we omit U_i in our notations if $U_i = \emptyset$. If $\bar{C} = (C_1(w), \ldots, C_N(w))$ is a sequence of counting functions like (1) one can associate with w a vector of real numbers:

$$f_{\bar{C}}(w) = \frac{1}{|w|} < C_1(w), \ldots, C_N(w) > \in \mathbb{R}^N$$

which is called a *feature vector*. Every choice of the sequence $\bar{C}$ gives a vector $f_{\bar{C}}(w)$ which reflects the structure of w.

For example, if $a \in X^{\pm 1}$ then $C(w, a)$ counts the number of occurrences of the letter a in w. The feature vector (where for simplicity we assume that the components are written in some order which we do not specify)

$$f_0(w) = \frac{1}{|w|} < C(w, a) \mid a \in X^{\pm 1} >$$

shows the frequencies of occurrences of letters from $X^{\pm 1}$ in w. The feature vector

$$f_1(w) = \frac{1}{|w|} < C(w, v) \mid |v| = 2 >$$

shows the numbers of occurrences of words of length two in w relative to the length of w.

To visualize some structures described by the counting functions above we associate with a given word $w \in F(X)$ a weighted labelled directed graph $\Gamma(w)$. Put $V(\Gamma(w)) = X^{\pm 1}$. For given $x, y \in X^{\pm 1}$ and $v \in F(X)$ we connect the vertex x to the vertex y by an edge with a label v and weight $C(w, xvy)$. Now, with every edge from x to y with label xvy one can associate a counting function $C(w, xvy)$, and vice versa. It follows that every subgraph Γ of $\Gamma(w)$ gives rise to a particular set of counting functions $\bar{C}_\Gamma$ of the type $C(w, xvy)$, and conversely, every set $\bar{C}$ of counting functions of the type $C(w, xvy)$ determines a subgraph $\Gamma_{\bar{C}}$ of $\Gamma(w)$. For instance, the feature mapping f_1 corresponds to the subgraph $\Gamma_1(w)$ of $\Gamma(w)$ which is in a sense a directed version of the so-called *Whitehead graph* of w.

Let U_n be the set of all words in F that are length n. Let W_n be the set of all words in F that are of length n or less. Other relevant features can be defined as follows. Each corresponds to various subgraphs of the graph $\Gamma(w)$:

$$f_2(w) = \frac{1}{|w|} < C(w, x_1 U_1 x_2) \mid x_1, x_2 \in X^{\pm 1} >;$$

$$f_3(w) = \frac{1}{|w|} < C(w, x_1 U_2 x_2) \mid x_1, x_2 \in X^{\pm 1} >;$$

$$f_4(w) = \frac{1}{|w|} < C(w, x_1 U_3 x_2) \mid x_1, x_2 \in X^{\pm 1} >;$$

$$f_5(w) = \frac{1}{|w|} < C(w, x_1 W_1 x_2) \mid x_1, x_2 \in X^{\pm 1} >;$$

$$f_6(w) = \frac{1}{|w|} < C(w, x_1 W_3 x_2) \mid x_1, x_2 \in X^{\pm 1} > .$$

4. Pattern Recognition Tools and Models

There are a variety of pattern recognition tools that are useful for determining a way of making a distinction given a set of feature vectors from one class and a set of feature vectors from another class. For each given class of objects, we sample objects from the class and construct the set of corresponding feature vectors. The pattern recognition technology provides a way of determining a best or near best boundary in the feature space that distinguishes the one class from the other. In this section we will review some of the basic techniques. The reader interested in a fuller discussion may consult general references [**2**],[**5**][**3**],[**6**].

4.1. Principal Components. There are occasions when the feature vectors coming from a class either all lie in a small dimensional flat or most of them lie in a small dimensional flat. Principal component analysis can determine this.

Let $x_1, \ldots, x_N$ be the set of feature vectors sampled from a given class. Define their sample mean μ by

$$\mu = \frac{1}{N} \sum_{n=1}^{N} x_n$$

Define their sample covariance C by

$$C = \frac{1}{N} \sum_{n=1}^{N} (x - \mu)(x - \mu)'$$

Let $t_1, \ldots, t_N$ be the eigenvectors of C with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_N$. Should the feature vectors indeed lie in a small dimensional flat, then there will be a $K < N$ such that $\lambda_k = 0,\ k = K+1, \ldots, N$.

In this case, feature vectors x coming from objects that are in the class can be recognized by testing whether or not

$$||T(x - \mu)|| = 0$$

where T is a $N \times (N - K)$ matrix whose columns are eigenvectors $t_{K+1}, \ldots, t_N$.

Those in the class will have $||T(x - \mu)|| = 0$. $||T(x - \mu)|| > 0||$ is a sure indication that x comes from an object out of the class, but there may be some objects out of the class for which $||T(x - \mu)|| = 0$.

In the case of two classes, we form the matrix T_1 from the zero eigenvalue eigenvectors of the covariance matrix from the class one feature vectors and the matrix T_2 from the zero eigenvalue eigenvectors of the covariance matrix from the class two feature vectors.

Now if $||T_1(x - \mu_1)|| = 0$ and $||T_2(x - \mu_2)|| > 0$, we assign vector x to class one. If $||T_1(x - \mu_1)|| > 0$ and $||T_2(x - \mu_2)|| = 0$, we assign vector x to class two. If $||T_1(x - \mu_1)|| = 0$ and $||T_2(x - \mu_2)|| = 0$, feature vector x comes from an object that is both class 1 and class 2. If $||T_1(x - \mu_1)|| > 0$ and $||T_2(x - \mu_2)|| > 0$ feature vector x comes from an object that is neither class 1 nor class 2.

4.2. Classifying by Distance. Let T_1 and T_2 be defined as before. We form the discriminant function

$$f(x) = ||T_1(x - \mu_1)|| - ||T_2(x - \mu_2)||$$

which measures the difference between feature vector x and the flat associated with class 1 and the flat associated with class 2. The decision rule is to assign vector x to class 1 if $f(x) > \theta$, otherwise assign to class 2. Here, after the discriminant function is defined we determine the value of θ that minimizes the error.

Classifying by distance can also be done with respect to the class means. Here the discriminant function is defined by

$$f(x) = (x - \mu_1)'C_1^{-1}(x - \mu_1) - (x - \mu_2)'C_2^{-1}(x - \mu_2)$$

As before, the decision rule is to assign vector x to class 1 if $f(x) > \theta$, otherwise assign to class 2. Here also after the discriminant function is defined we determine the value of θ that minimizes the error.

4.3. Linear Classifiers. Classifying may be done by a linear decision rule. Here the discriminant function is given by

$$f(x) = v'x$$

where vector v is the weight vector and is the normal to the hyperplane separating the feature space into two parts.

If $f(x) < \theta$ the decision rule is to assign the vector x to class 1 otherwise to class 2.There are a variety of ways to construct the weight vector v. One mthod is by regression. Another is names the Fisher linear discriminant. A third is called the support vector machine approach.

4.3.1. *Regression Classifier.* In the regression classifier, we form a matrix A whose rows are the feature vectors. We form a vector b whose kth component is 0 if the kth feature vector comes from class one and whose kth component is 1 is the kth feature vector comes from class two. We determine the weight vector v as that vector that minimizes $||Av - b||$. The minimizing vector v is given by the normal equation

$$v = (A'A)^{-1}A'b$$

The discriminant function is defined by

$$f(x) = v'x$$

We assign a vector to class one if $f(x) < \theta$ and to class two otherwise. θ is chosen to minimize the error of the assignment.

4.3.2. *Fisher Linear Discriminant.* Fisher's linear discriminant function is obtained by maximizing the Fisher's discriminant ratio, which, as described below, is the ratio of the projected between class scatter to the projected within class scatter.

Let v be the unknown weight vector. Let μ_1 and μ_2 be the class one and two sample means and let C_1 and C_2 be the class one and two sample covariance matrices. Let N_1 be the number of feature vectors in class one and let N_2 be the number of feature vectors in class two. Define the overall mean μ by

$$\mu = P_1\mu_1 + P_2\mu_2$$

where $P_1 = N_1/(N_1+N_2)$ and $P_2 = N_2/(N_1+N_2)$. Then the between-class scatter matrix S_b is given by

$$\begin{aligned} S_b &= \sum_{i=1}^{2} P_i(\mu_i - \mu)(\mu_i - \mu)' \\ &= P_1P_2(\mu_1 - \mu_2)(\mu_1 - \mu_2)' \end{aligned}$$

Define S_w to be the average class conditional scatter matrix, then

$$S_w = \sum_{i=1}^{2} P_iC_i$$

Finally, if we let S designate the scatter matrix of the mixture distribution,

$$S = \frac{1}{N_1+N_2}\sum_{k=1}^{N}[(x_k - \mu)(x_k - \mu)']$$

then

$$S = S_w + S_b$$

In the one dimensional projected space one can easily show that the projected between class scatter s_b and the projected within-class scatter s_w are expressed as

$$s_b = v'S_bv$$

$$s_w = v'S_wv$$

Then the Fisher discriminant ratio is defined as

$$F(v) = \frac{s_b}{s_w} = \frac{v'S_bv}{v'S_wv}$$

The optimum direction v can be found by taking the derivative of $F(v)$ with respect to v and setting it to zero:

$$\nabla F(v) = (v'S_wv)^{-2}(2S_bvv'S_wv - 2v'S_bvS_wv) = 0$$

From this equation it follows that

$$v'S_bvS_wv = S_bvv'S_wv$$

If we divide both sides by the quadratic term $v'S_bv$, then

$$\begin{aligned} S_wv &= \frac{v'S_wv}{v'S_bv})S_bv \\ &= \lambda S_bv \\ &= \lambda P_1P_2(\mu_1 - \mu_2)(\mu_1 - \mu_2)'v \\ &= \lambda\kappa(\mu_1 - \mu_2) \end{aligned}$$

where λ and κ are some scalar values defined as

$$\begin{aligned} \lambda &= \frac{v'S_wv}{v'S_bv} \\ \kappa &= P_1P_2(\mu_1 - \mu_2)'v \end{aligned}$$

Thus we have the weighting vector v as

$$v = KS_w^{-1}(\mu_1 - \mu_2)$$

where $K = \lambda\kappa$ is a multiplicative constant.

The discriminant function is defined by $f(x) = v'x$. The vector x is assigned to class one if $F(x) > \theta$ and to class two otherwise. The threshold θ is set to a value that minimizes the error of the class assignment.

4.3.3. *Support Vector.* Let $z_1, \ldots, z_N$ be the set of N training vectors with corresponding labels $y_1, \ldots, y_N$, a label being +1 for class one and -1 for class two. Consider a hyperplane ribbon R that can separate the training vectors of class one from the training vectors of class two. We represent R by

$$R = \{x \mid 1 \leq w'x \leq 1\}$$

The support vector machine approach seeks to find the widest ribbon so that R separates the vectors of class one from class two.

The distance of the hyperplane $H = \{x \mid w'x = 1\}$ from the origin is $\frac{1}{||w||}$. Therefore the ribbon R has width $\frac{2}{||w||}$. To maximize this width is equivalent to minimizing $||w||$. This minimization must be done under the constraint that $y_k w' z_k > 1$, k= 1,..., K. Define the matrix A by

$$A = \begin{pmatrix} y_1 z_1' \\ y_2 z_2' \\ \vdots \\ y_K z_K' \end{pmatrix}$$

Define the vector b to be a vector of K components all of which have value 1. The support vector approach determines the vector w by minimizing $w'w$ subject to the constraint $Aw > b$. This can be solved by standard quadratic programming methods.

4.4. Quantizing. Let f be a discriminant function. We evaluate $f(x)$ over all the sampled vectors x from class one and from class two to determine the range. We divide the range in a fixed number M of quantizing intervals. The simplest way is called equal interval quantizing. Here the range is divided up into M equal intervals. In each interval the number of sampled vectors coming from class one and coming from class two is determined. The interval is labelled by the class of the majority of the vectors in it.

A vector x having discriminant value $f(x)$ which falls into the mth quantizing interval is assigned to the class that labels the quantizing interval.

Another simple alternative quantizing scheme is to divide the range into intervals each of which have the same number of sampled discriminant values. This is called equal probability quantizing.

A more complex scheme is to divide the discriminant range into M intervals in such a way that the classification error is minimized.

4.5. Classification Trees. The type of classification tree discussed here is a binary tree with a simple discriminant function; thus every nonterminal node has exactly two children [**1**]. During classification, if the node's discriminant function is less than a threshold, the left child is taken; if it is greater than the threshold, the right child is taken. This section describes the design process of the binary tree classifier using a simple discriminant function. There are two methods of expanding a nonterminal node according to the selection of a decision rule for the node. We show how to use an *entropy purity function* to decide what the threshold value should be, and we discuss the relationship of the purity function to the χ^2 test

statistic. We discuss the criteria for deciding when to stop expanding a node and for assigning a class.

Let $x_1, \ldots, x_N$ be the set of vectors in the training set. Associated with each vector is a class label. Let M be the number of classes. Let

$$X^n = \{x_k^n \mid k = 1, \ldots, N^n\}$$

be the subset of N^n training vectors associated with node n. Let N_c^n be the number of training vectors for class c in node n. Since N^n is the total number of training samples in node n, we must have $N^n = \sum_{c=1}^{M} N_c^n$. The decision rule selected for node n is that discriminant function having the greatest purity, a quality we will precisely define later.

Now we define how the decision rule works at node n. Consider the feature vector x_k^n. If the discriminant function $f(x_k^n)$ is less than or equal to the threshold, then x_k^n is assigned to class Ω_{LEFT}^n, otherwise it is assigned to class Ω_{RIGHT}^n. An assignment to Ω_{LEFT}^n means that the feature vector descends to the left child node. An assignment to Ω_{RIGHT}^n means that the feature vector descends to the right child node.

Given a discriminant function f, we sort the feature vectors in the set X^n in an ascending order according to their discriminant function value. Without loss of generality we assume that the feature vectors are sorted in such a way that $f(x_k^n) \leq f(x_{k+1}^n)$ for $k = 1, \ldots, N^n - 1$,. Let w_k^n be the true class associated with the measurement vector x_k^n. Then the set of candidate thresholds T^n is defined by

$$T^n = \left\{ = \frac{f(x_{k+1}^n) - f(x_k^n)}{2} \middle| w_{k+1}^n \neq w_k^n \right\}$$

For each possible threshold value, each feature vector x_k^n is classified by using the decision rule specified above. We count the number of samples n_{Lc}^t assigned to Ω_{LEFT}^n whose true class is c, and we count the number of samples n_{Rc}^t assigned to Ω_{RIGHT}^n whose true class is c.

$$\begin{aligned} n_{L_c}^t &= \#\{k \Big| f(x_k^n) \leq t \text{ and } w_k^n = c\} \\ n_{R_c}^t &= \#\{k \Big| f(x_k^n) > t \text{ and } w_k^n = c\} \end{aligned}$$

Let n_L^t be the total number of samples assigned to Ω_{LEFT}^n and n_R^t be the total number of samples assigned to Ω_{RIGHT}^n, that is,

$$n_L^t = \sum_{c=1}^{M} n_{Lc}^t$$

$$n_R^t = \sum_{c=1}^{M} n_{Rc}^t$$

We define the purity PR_n^t of such an assignment made by node n to be

$$PR_n = \sum_{c=1}^{M} \left(n_{Lc}^t \ln p_{Lc}^t + n_{Rc}^t \log p_{Rc}^t \right)$$

where

$$p_{Lc}^t = \frac{n_{Lc}^t}{n_L}$$

$$p_{Rc}^t = \frac{n_{Rc}^t}{n_R}$$

The discriminant threshold selected is the threshold t that maximizes the purity value PR_n^t. The purity is such that it gives maximum value when the classes of the training vectors are completely separable. For example, consider a nonterminal node having m units in each of three classes in the training sample. If the selected decision rule separates the training samples such that the *LEFT* child contains all feature vectors in one class and the *RIGHT* child contains all the feature vectors in the other two classes, the purity is $0 - 2(m \ln \frac{1}{2}) = -2m \ln 2$. In the worst case, when both the *LEFT* and the *RIGHT* children contain the same number of feature vectors for each class, the purity is $-3(\frac{m}{2} \ln \frac{1}{3}) - 3(\frac{m}{2} \ln \frac{1}{3}) = -3m \ln 3$. Thus we can easily see that the purity value of the former case, where the training samples are completely separable, is greater than the purity value of the latter case, where the training samples are not separable.

The maximization of the purity can also be explained in terms of the χ^2 test of goodness of fit. If a decision node is effective, the distribution of classes for the children nodes will be significantly different from each other. A statistical test of significance can be used to verify this. One test statistic that measures the significance of the difference of the distributions is defined by

$$\chi^2 = \sum_{c=1}^{M} \left(n_{Lc}^t \ln p_{Lc}^t + n_{Rc}^t \ln p_{Rc}^t - N_c^n \ln \frac{N_c^n}{N^n} \right)$$

It has a χ^2 distribution with $M - 1$ degree of freedom. Comparing this equation with PR_n^t, we find that the χ^2 value is just the sum of the purity PR_n^t and some constant value.

Now we discuss the problem of when to stop the node expanding process and how to assign a class to the terminal node. First, it is not reasonable to generate a decision tree that has more terminal nodes than the total number of training samples. Using this consideration as a starting point, we set the maximum level of the decision tree to be $\log_2 N^1 - 1$, which makes the number of terminal nodes less than $\frac{N^1}{2}$, where N^1 is the number of training samples in node 1, the root node. Next, if the χ^2 value is small, the distributions of classes for the children nodes are not significantly different from each other, and the parent node need not be further divided. Finally, when the number N^n of units at node n becomes small, the χ^2 test cannot give a reliable result. Therefore we stop expanding node n when N^n is less than some lower limit. If one of these conditions is detected, then the node n becomes a terminal node.

When a node becomes terminal, an assignment of a label to the node is made. Each terminal node is assigned that class label that is the majority of the class labels of the training vectors in the node.

An alternative decision tree construction procedure uses the probability of misclassification in place of entropy. In this procedure the decision rule selected for the nonterminal node is the one that yields the minimum probability of misclassification of the resulting assignment. To describe the termination condition of a node expansion, we first define type I and type II errors as follows: Let type I error be the probability that a unit whose true class is in Ω_{LEFT}^n is classified as Ω_{RIGHT}^n, and type II error be the probability that a unit whose true class is in Ω_{RIGHT}^n is classified as Ω_{LEFT}^n. Then, if the sample space is completely separable, we would

get zero for both type I and type II errors. Since this is not always the case, we control these errors by considering only those thresholds in the process of threshold selection that give type I error less than ϵ_I and type II error less than ϵ_{II}, where ϵ_I and ϵ_{II} are values determined before we start constructing the decision tree. Next, in the process of expanding a nonterminal node, if we cannot find a decision rule that gives type I error less than ϵ_I and type II error less than ϵ_{II}, which means that the sample space is not separable at a ϵ_I and ϵ_{II} level, we stop expanding this nonterminal node. This process of decision tree construction is repeated until there is no nonterminal node left or the level of the decision tree reaches the maximum level. Assignment of a class to a nonterminal node is done in the same way as in the previous procedure.

As just described, there are two groups of classes at each nonterminal node in the binary decision tree. For the purpose of discussion, we denote all the classes in one group as Ω_{LEFT} and all the classes in the other group as Ω_{RIGHT}. The job of the decision rules discussed here is to separate the left class Ω_{LEFT} from the right class Ω_{RIGHT}. We will employ the same notational conventions used previously. The superscript n denoting the node number will be dropped from the expression if it is clear from the context that we are dealing with one particular node n.

The simplest form for a discriminant function to take is a comparison of one measurement component to a threshold. This is called a *threshold decision* rule. If the selected measurement component is less than or equal to the threshold value, then we assign class Ω_{LEFT} to the unit u_k; otherwise we assign class Ω_{RIGHT} to it. This decision rule requires a feature vector component index and a threshold. Each feature vector component is selected in turn and the set of threshold candidates T of that component is computed. For each threshold in the set T, all vectors in the training set X^n at node n are classified into either class Ω_{LEFT} or class Ω_{RIGHT} according to their value of the selected feature component. The number of feature vectors for each class assigned to class Ω_{LEFT} and to class Ω_{RIGHT} is counted, and the entropy purity is computed from the resulting classification. A threshold is selected from the set of threshold candidates T such that, when the set X^n is classified with that threshold, a maximum purity in the assignment results. This process is repeated for all possible feature components, and the component and threshold that yield an assignment with the maximum purity is selected.

4.6. Clustering. All the previous methods required that the feature vectors be generated for each of the objects to be discriminated and an associated class label be associated with each of the feature vectors. In this section we discuss a method to automatically determine the natural classes, called clusters, to be associated with each feature vector. Feature vectors with the same cluster label are more similar to each other and less similar to feature vectors of a different cluster label. Clustering can be used to help generate hypotheses about natural distinctions between mathematical objects before these distinctions themselves are known.

The most widely used clustering scheme is called K-means. It is an iterative method. It begins with a set of K cluster centers $\mu_1^0, \ldots, \mu_K^0$. Initially these are chosen at random. At iteration t each feature vector is assigned to the cluster center to which it is closest. This forms index sets $S_1^t, \ldots, S_K^t$.

$$S_k^t = \{n \mid ||x_n - \mu_k^t|| \leq ||x_n - \mu_m^t||, m = 1, \ldots, K\}$$

Then each cluster center is redefined as the mean of the feature vectors assigned to the cluster.

$$\mu_k^{t+1} = \frac{1}{\#S_k^t} \sum_{n \in S_k^t} x_n$$

Each iteration reduces the criterion function J^t.

$$J^t = \sum_{k=1}^{K} \sum_{n \in S_k^t} ||x_n - \mu_k^t||$$

As this criterion function is bounded below by zero, the iterations must converge.

5. Recognizing Whitehead Minimal Words in Free Groups

In this section we return to the Whitehead minimal word problem. We first discuss a pattern recognition system based on linear regression, then we show some applications of clustering.

5.1. Linear regression classifier. Data sets. To train a classifier, we must have a training set. To test the classifier we must have an independent test set.

A "random" element w of $F = F(X)$ can be produced as follows. Choose randomly a number l (the length of w), and a random sequence $y_1, \dots, y_l$ of elements $y_i \in X^{\pm 1}$ such that $y_i \neq y_{i+1}^{-1}$, where y_1 is chosen randomly and uniformly from $X^{\pm 1}$, and y_{i+1} is chosen randomly and uniformly from the set $X^{\pm 1} - \{y_i^{-1}\}$. Similarly, one can pseudo-randomly generate cyclically reduced words in F, i.e., words $w = y_1 \dots y_l$ where $y_1 \neq y_l^{-1}$.

To generate the training data set we used the following procedure. For each positive integer $l = 1, \dots, 1000$ we generate randomly and uniformly 10 cyclically reduced words from $F(X)$ of length l. Denote the resulting set by W. Then using the deterministic Whitehead algorithm one can effectively construct the corresponding set of minimal elements

$$W_{min} = \{w_{min} | w \in W\}.$$

With probability 0.5 we substitute each $v \in W_{min}$ with the word $\widetilde{v^t}$, where t is a randomly and uniformly chosen Whitehead automorphism such that $|\widetilde{v^t}| > |v|$ (if $|\widetilde{v^t}| = |v|$ we chose another automorphism t, and so on). Now, the resulting set D is a set of pseudo-randomly generated cyclically reduced words representing the classes of minimal and non-minimal elements in approximately equal proportions. We choose D as the training set.

One remark is in order here. It seems, the class of non-minimal elements in D is not quite representative, since every one of its elements w has Whitehead complexity 1, i.e., there exists a single Whitehead automorphism which reduces w to w_{min} (see [**4**] for details on Whitehead complexity). However, our experiments showed that the set D is a sufficiently good training dataset which is much easier to generate than a set with uniformly distributed Whitehead complexity of elements. A possible mathematical explanation of this phenomena is mentioned in [**4**].

To test and evaluate the pattern recognition methodology we generate several test datasets of different type:

- A test set S_e which is generated by the same procedure as for the training set D, but independently of D.
- A test set S_R of randomly generated elements of $F(X)$.
- A test set S_P of (pseudo-) randomly generated *primitive* elements in $F(X)$. Recall that $w \in F(X)$ is primitive if and only if there exists a sequence of Whitehead automorphisms $t_1 \dots t_l \in \Omega(X)$ such that $x^{t_1 \dots t_l} = w$ for some $x \in X^{\pm 1}$. Elements in S_P are generated by the procedure described in [**4**], which, roughly speaking, amounts to a random choice of $x \in X^{\pm 1}$ and a random choice of a sequence of automorphisms $t_1 \dots t_l \in \Omega(X)$.
- A test set S_{10} which is generated in a way similar to the procedure used to generate the training set D. The only difference is that the non-minimal elements are obtained by applying not one, but several randomly chosen Whitehead automorphisms. The number of such automorphisms is chosen uniformly randomly from the set $\{1, \dots, 10\}$, hence the name.

Features. We use the feature vectors $f_1(w), \dots, f_6(w)$ described in Section 3.

Model. Our model is based on the linear regression classifier described in Section 4.3. For any word w having feature vector $z(w)$ we compute the discriminant function

$$\hat{P}(w) = b' z(w)$$

where b' is the vector of regression coefficients obtained from the training data set D. The decision rule is based on the equal interval quantizing method described in Section 4.4.

Evaluation. Let D_{eval} be a test data set. To evaluate the performance of the given **PR** system we use a simple accuracy measure:

$$A = |\{w \in D_{eval} | \text{minimality of } w \text{ is decided correctly}\}|$$

It is the number of the correctly classified elements from the test set D_{eval}.

Results. Now we present the evaluation of classifiers $\mathbf{P}_f$ on the test dataset S_e when f runs over the set of feature mappings $f_1, \dots, f_6$ mentioned above. By $A(f)$ we denote the accuracy of the classifier $\mathbf{P}_f$. For simplicity we present results only for the free group F of rank 2 with basis $X = \{a, b\}$.

The results of evaluation of the classifiers $\mathbf{P}_i = \mathbf{P}_{f_i}, i = 1, \dots, 6$ on S_e are given in Table 1.

	$A(f_1)$	$A(f_2)$	$A(f_3)$	$A(f_4)$	$A(f_5)$	$A(f_6)$
$\|w\| > 0$	0.954	0.968	0.926	0.869	0.977	0.980
$\|w\| > 4$	0.957	0.969	0.927	0.870	0.977	0.981
$\|w\| > 100$	0.975	0.984	0.947	0.893	0.992	0.994

TABLE 1. Performance of the classifiers $\mathbf{P}_1, \dots, \mathbf{P}_6$ on the set S_e.

One can draw the following conclusions from the experiments:

- It seems, that the accuracy of the classifiers increases when one adds new edges to the graphs related to the feature mappings (though it is not clear what is the optimum set of features);
- The classifier $\mathbf{P}_6$ is the best so far, it is remarkably reliable;
- Very short words are difficult to classify (perhaps, because they do not provide sufficient information for the classifiers);
- The estimated conditional probabilities for $\mathbf{P}_6$ (which come from the Bayes' decision rule) are presented in Figure 1. Clearly, the classes of minimal and non-minimal elements are separated around 0.5 with a small overlap. So the regression works perfectly with the threshold $\Theta = 0.5$.

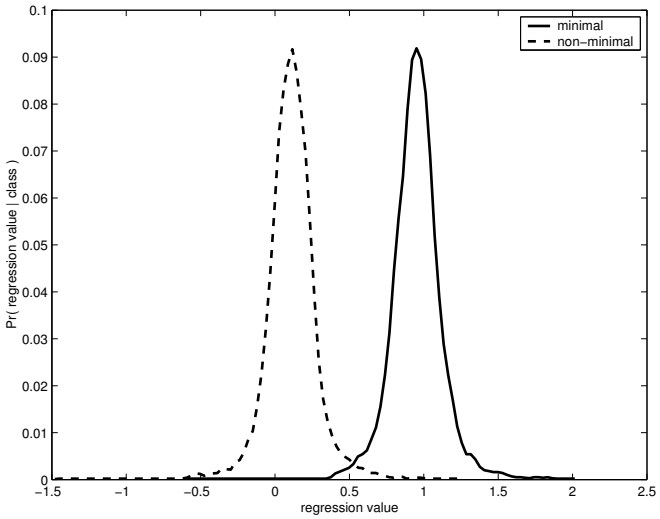

FIGURE 1. Conditional probabilities for $\mathbf{P}_6$.

Looking for the best feature vectors. As we have seen above the performance of a classifier $\mathbf{P}_f$ directly depends on the feature vector $f(w)$ built into it. Sometimes it is possible to reduce the number of features in f maintaining the same level of classification accuracy of, and even find more efficient combinations of the given features. The corresponding procedure is called *feature selection*.

By far, the feature vector f_6 was the most effective. Observe, that f_6 has 60 components (features). To find a better feature vector we used an iterative greedy procedure to select the best vector from the set of all counting functions of the type

$$\{C(w, xvy) \mid x, y \in X^{\pm 1}, v \in F(X), 1 \leq |v| \leq 3\}.$$

It turns out that one of the most effective feature vectors consists only of two counting functions:

$$f^*(w) = \frac{1}{|w|} < C(w, a^{-1}b), C(w, b^{-1}a) > .$$

The results of comparison of $\mathbf{P}_* = P_{f^*}$ with $\mathbf{P}_1$ and $\mathbf{P}_6$ are presented in Table 2.

	$A(f_1)$	$A(f_6)$	$A(f^*)$
$\|w\| > 0$	0.954	0.980	0.987
$\|w\| > 4$	0.957	0.981	0.989
$\|w\| > 100$	0.975	0.994	0.993

TABLE 2. Comparative results for $\mathbf{P}_*$.

5.2. Clustering. In this section we describe one application of the K-means clustering scheme to the Whitehead minimization problem.

In general cluster analysis is used to recover hidden structures in a sampled set of objects. In the Whitehead's minimization method the following *Length Reduction Problem* is of prime interest: given a non-minimal word $w \in F$ find a (*length-reducing*) Whitehead automorphism t such that

$$|wt| < |w|.$$

Below we apply K-means clustering method described in section 4.6 to the Length Reduction Problem. The task is to partition a given set of non-minimal elements into clusters in such a way that every cluster would have a Whitehead automorphism assigned to it which reduces the length of the most words from the cluster. To illustrate performance of the K-means algorithm we take the feature vector function f_2 from Section 3 and the standard Euclidean metric in $\mathbb{R}^4$ (recall that $f_2(w) \in \mathbb{R}^4$ for $w \in F_2$).

To perform the K-means procedure one needs to specify in advance the number K of expected clusters. Since we hope that every such cluster $\mathcal{C}$ will correspond to a particular Whitehead automorphism that reduces the length of elements in $\mathcal{C}$ then the expected number of clusters can be calculated as follows. It is easy to see that the set Ω_2 of all Whitehead automorphisms of the free group $F_2(X)$ with basis $X = \{a, b\}$ splits into two subsets: the set

$$N_2 = \left\{ \begin{pmatrix} a \to ab \\ b \to b \end{pmatrix}, \begin{pmatrix} a \to b^{-1}a \\ b \to b \end{pmatrix}, \begin{pmatrix} a \to a \\ b \to ba \end{pmatrix}, \begin{pmatrix} a \to a \\ b \to a^{-1}b \end{pmatrix} \right\}$$

of Nielsen automorphisms and the set of conjugations. If we view elements of F_2 as cyclic words (i.e., up to a cyclic permutation) then the conjugations from Ω_2 can be ignored in the length reduction problem. Therefore, we would like the K-means algorithm to find precisely 4 clusters.

Let $S \subset F_2$ be a set of non-minimal cyclically reduced words from F_2. We construct the set

$$D =< f_2(w) \mid w \in S >$$

of feature vectors, corresponding to words in S. To start the K-means algorithm one needs to choose the set of initial centers $\mu_t^0, t \in N_2$. Observe, that the algorithm is quite sensitive to this choice of the centers. There are various methods to generate $\mu_t^0, t \in N_2$, here we describe just one of them. Let S' be a sample subset of the set S. For an automorphism $t \in N_2$ put

$$C_t = \{w \in S' \mid |wt| < |w|, \forall r \in N_2 (r \neq t \to |wr| \geq |w|)\}$$

and define

$$\mu_t^0 = \frac{1}{|C_t|} \sum_{w \in C_t} f_2(w) \tag{2}$$

as the initial estimates for the cluster centers (we assume here that the sets C_t are not empty).

The goodness of the clustering is evaluated using a measure R_{max} defined below. Let $\mathcal{C} \subset D$ be a cluster. For $t \in N_2$ define

$$R(t, \mathcal{C}) = \frac{| < v | |t(v)| < |v|, v \in \mathcal{C} > |}{|\mathcal{C}|}.$$

The number $R(t, \mathcal{C})$ shows how many elements in $\mathcal{C}$ are reducible by t. Now put

$$R_{max}(\mathcal{C}) = \max\{R(t, \mathcal{C}) \mid t \in N_2\}.$$

The number $R_{max}(\mathcal{C})$ shows how many elements in $\mathcal{C}_i$ can be reduced by a single automorphism.

Now we perform 4-means clustering algorithm on the set S and compute $R(t, \mathcal{C}_i)$, $t \in N_2$, for each obtained cluster $\mathcal{C}_i$. The results of the experiments are given in Table 5.2 for two different choices of the initial centers (the random choice and the choice according to 2). We can see from the table that the clustering, indeed, groups elements in S with respect to the length reducing transformation.

	random μ_t^0	μ_t^0 estimated by (2)
avg(R_{max})	0.930	1.000
max(R_{max})	1.000	1.000
min(R_{max})	0.743	1.000

TABLE 3. Evaluation of 4-means clustering of the set D.

The experiments show that it is possible to cluster non-minimal elements in F_2, using the standard clustering algorithms, in such a way that:

- every cluster contains elements whose length can be (with a very high probability) reduced by a particular Nielsen transformation;
- the transformation that reduces the length of the most elements from one cluster does not reduce the length of the most elements in another cluster.

This gives a very strong heuristic for choosing a length reducing automorphism for a given word $w \in F_2$.

A simple decision rule which for a given word $w \in F_2$ will predict a corresponding length reducing automorphism t can be defined as follows. Let μ_t be the centers of the clusters, produced by the K-means clustering. Each μ_t corresponds to a particular automorphism $t \in N_2$. For a given non-minimal cyclically reduced word $w \in F_2$ we select an automorphism $t^* \in N_2$ such that

$$\forall t \in N_2 \ (||f_2(w) - \mu_{t^*}|| \ \leq \ ||f_2(w) - \mu_t||)$$

as the most probable length-reducing automorphism for w.

In the conclusion we would like to add that a similar analysis can be used to predict most probable length-reducing automorphisms for words in free groups of ranks n larger then 2. However, the number of the corresponding clusters grows

exponentially with n which increases the error rate of the classification. In this case more careful clustering still could be applied where the clusters correspond to some particular groups of Whitehead automorphism.

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.
[2] S. D. Duda R., Hart P. *Pattern Classification*. John Wiley and Sons Inc., 2001.
[3] K. Fukunaga. *Introduction to Statisical Pattern Recognition*. Academic Press Inc., 1990.
[4] A. D. Miasnikov and A. Myasnikov. Whitehead method and genetic algorithms. Preprint.
[5] E. Patrick. *Fundamentals of Pattern Recognition*. Prentice Hall Inc., 1972.
[6] E. Schalkoff. *Pattern Recognition*. John Wiley and Sons Inc., 1992.

Department of Computer Science, Graduate Center, City University of New York, 365 Fifth Avenue, New York, NY 10016