INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

A Study of Various Representations

Using

NEXTPITCH: A Learning Classifier System

by

Francine Federman

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1996

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/18/96
Date

[signature] Jacqueline J. Jones
Chair of examining Committee

9/18/96
Date

[signature]
Executive Officer

Dr. D. L. Scarborough

Dr. K. Ebcioglu
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

A Study of Various Representations Using NEXTPITCH: A Learning Classifier System

By

Francine Federman

Advisor: Dr. Jacqueline A. Jones

Our model, NEXTPITCH, a learning classifier system using genetic algorithms, inductively learns to predict the next note in a nursery melody. Just as a listener develops expectations of what is to follow based on what has been heard, NEXTPITCH models human music learning by developing the rules that represent actual pitch transitions in the melody.

The focal point of this research is to compare and analyze different representations of specific features of Western tonal music within the construct of a learning classifier system. The rationale for the specific note representations merges ideas from learning classifier systems and genetic algorithms with concepts espoused by the music cognition community.

The areas of study addressed are representation of music, classifier format and the number of classifiers to use. Our results are correlated by analyses of

variance statistical routines. Information theory is used to partition melodies into classes so that we may examine the applicability of the results from one set of melodies to another.

Acknowledgments

This dissertation is dedicated to all those women who precede me and to all those women who will succeed me. Family, especially children, are our mainstay, but it is our dreams and their fulfillment that give life its meaning. Follow your dreams!

To all my colleagues, friends, and family: Without each of you graciously offering your talents, I could not have completed this task.

Thank You:

Joseph Driscoll, for friendship and insights. Without each, I would not have produced this document.

Marco Morazan, whose friendship and knowledge filled in the blanks when I had many blanks to fill in.

To Betty Miller, Brooklyn College, and Louis Pisha, C.W. Post, librarians who made my literary searches possible.

Paula Farbman and Larry Ballereau for transposing the music.

Lois Tepper for holding my hand and sharing your knowledge of statistics.

Jackie Jones and Don Scarborough, my advisors at C.U.N.Y. Your ideas and hurdles permitted me to evolve into a researcher.

Susan Dorchak, for your friendship during this endeavor.

To my colleagues at C. W. Post and Brooklyn College for all your inspiration and endorsement.

To the administration at C .W. Post, for your financial support and computer resources. Without either, there would have been no "iterations."

To the administration at the Computer Science Dept. at C.U.N.Y. and the C.U.N.Y. Research Foundation for your funding.

Sharon Salz, a true friend keeping me sane in the day-to-day world.

Frances Rosenzweig. We will succeed in the face of adversity.

Ron Hochstein. Without your hardware loan, this final document would not have been completed.

And in the final analysis, it all comes to family support:

Shelly Braunstein, for all your early morning hand-holding conversations. "We are our own worst enemy."

Susan Blumberg, for all your anytime conversations; anything is possible if we have perseverance.

To my Mom, Evelyn Federman, for all her everlasting guidance (morning, noon, and night). WE DID IT!

To my brother, Ronald Federman, for letting me know nothing good comes without hard work.

To my husband Glenn, throughout all my ups and downs. Thanks for being Glenn "you never let me down."

And finally, to my children, Meri and Kenny. It wasn't always easy living with a "determined" Mom. Thank you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of this Dissertation

The intention of our research is to compare and analyze different representations of specific features of Western tonal music. The representations are tested in a system, NEXTPITCH, that models the learning of familiar melodies by a learning classifier system using a genetic algorithm as the learning mechanism (Holland, et al., 1986). NEXTPITCH adapts Goldberg's Simple Classifier System to the domain of music (Goldberg, 1989).

NEXTPITCH uses nursery melodies as input, where the condition of a classifier represents a possible two note input sequence and the action of a classifier represents a predicted next pitch. The processing of NEXTPITCH allows it to "learn" the consecutive note transitions within each three-note sequence. The output of the system is a report that measures the performance of predicting the next note over a period of time.

The literature states that users of classifier systems rely on ad hoc methods for choosing parameter settings (Miller and Forrest, 1989) and representations of the information being learned. We study a sample of encodings of music for the purpose

of providing information for future research in the disciplines of artificial intelligence and music cognition. The goal of our work is to determine which of the representations is superior.

This paper addresses three topics in the field of learning classifier systems (LCS): the representation of music, the classifier format, and the number of classifiers to use. Our research tests five representations of pitch that are carefully selected based on ideas from LCS and music psychology. All representations encode the features of pitch name along with octave and accidental information. Then, all representations are supplemented by encoding the relational properties between two notes: contour, duration and interval. But it is the way the relational properties are joined to the principal feature of pitch that emphasizes the difference between complete representations. As a second focus, this research looks at the behavior of an LCS where the conditions in the classifiers do not all have the same format, contrasting it with traditional systems where the conditions of a classifier have the same format. In addition, we study the appropriate number of classifiers to use for these tests.

1.2 Outline of this Dissertation

Chapter 2 provides background knowledge on the task of music learning as well as machine learning. An overview of classifier systems and genetic algorithms is provided.

Chapter 3 reviews the current literature with emphasis on computer-based systems applied to the domain of music. The general issue of representation and the

encoding of music are discussed.

In Chapter 4, NEXTPITCH--the core of this research--is explained. When relevant, the descriptions of our earlier models, NEXTNOTE and NEXTSONG--implementations of a Simple Classifier System (SCS) to music learning--are reviewed.

In Chapter 5, the problem of music representation as interpreted by this research is presented. All representations are defined and the rationale for choosing each is offered.

In Chapter 6, the methodology of this research is delineated. There are four phases: Parameter Selection, Evaluating the Model, Applicability of Learning, and Size of Classifier Set Impact.

Chapter 7 states the results from all testing procedures that are pertinent to our analysis. The ensuing discussion utilizes the analysis of variance statistical routines (ANOVA) enabling us to analyze the performance of the variables (i.e. pitch representations, classifier structure, and melody) as well as the interactions within the results.

Finally, Chapter 8 furnishes the conclusion where the significance of this dissertation and related future research directions are discussed.

Chapter 2

Background

2.1 Introduction

Artificial intelligence is an area of computer science that simulates the function of the human brain with programs that operate on values that represent the knowledge the mind acquires. One technique used in these programs, the *production system,* uses rules to reason with.

Music perception is an area that exemplifies human learning; therefore it is reasonable to have computer systems model music learning. Of the many ways available to implement music learning, the production system seems to be one of the best. Classifier systems, a specific type of production system, can be used to model the inductive learning of music.

2.2 Induction and the Learning of Music

Inductive Learning

Induction is defined by Holland (Holland, et al., 1986) as encompassing all inferential processes that expand knowledge in the face of uncertainty. Induction can be defined as learning without previously knowing the rules when exposed to a new

situation in the environment.

As applied to *music learning*, induction is learning the rules by listening to a melody and determining if each note fits the melody or the rhythm. The environmental input is the repetition of the melody. What is inductively learned is to anticipate what is to follow from what has been heard.

Why Music?

Sequential pattern learning is an important area of human inductive learning. One type of sequential pattern learning, found in many intelligence tests, is letter sequence prediction (e.g. ajbkcl...). Simon & Kotovsky's (1963) pattern induction program determines pattern from a sequence of letters.

One example of sequential patterns which is not often studied is music. The musical counterpart of Simon & Kotovsky's pattern induction program is the listener who induces musical pattern either from hearing a sequence of sounds or from reading sheet music. By extending Simon & Kotovsky's 1963 model, Simon & Sumner (1993) examined the theory that music consists of patterns. Simon & Sumner defined music pattern as multidimensional, consisting of melody, harmony, rhythm, and form, each of which in turn may have many dimensions.

Music differs from other sequential learning because it consists of several related attributes (e.g., pitch and duration) and it has a hierarchical structure (e.g., it is divided into phrases which combine to form larger phrases). Thus, music presents a complex example of sequential learning.

The Music Task and Induction

A music listener learns the musical invariants of his culture; those invariants transcend the details of any particular piece. For example, in Western tonal music, a listener expects to hear a piece that develops within a stable tonal framework, has some particular metric structure, and has a particular harmonic structure. A listener has expectations, at any point in the piece, about what is to follow. That is, based on the notes just heard, a limited set of notes can follow.

The induction problem can be summarized as one of how a listener learns to develop expectations of future musical events based on what has been heard so far. The model in this research, NEXTPITCH, predicts each note in a sequence of notes (after the first) and maintains a set of rules that represent the melody that is being learned.

2.3 Learning and Production Systems

Machine learning is an area of computer science where, by using the techniques of modeling, one can simulate the human learning process. Working computer systems can adapt to new situations and learn from them, simulating both the way cognitive tasks are performed and the way new concepts are built.

One technique by which machine learning programs simulate human logic is the "if/then" rule (if p, then q). This rule type allows for a representation of the basic structure of human reasoning and problem solving.

Production Systems

One of the major architectures for problem solving is the *production system*. A basic production system consists of a global data base, a set of production rules in the form "IF <condition> THEN <action>," and a set of control structures. Early computer production systems include Anderson's (1983) ACT, and Laird's (1983) SOAR; both were systems built to investigate different production system architectures.

Music learning, a subset of human learning, can be modeled by a production system. Sloboda (1985) states, in reference to the production system, that as a technique to simulate music learning "there is no other theoretical framework currently available which allows such detailed formalization of cognitive processes but at the same time supports many general but more vague conceptualizations and observations about learning."

Production systems are excellent systems to simulate cognitive abilities. Both production systems and humans have the ability for self-modification. Both will automatically apply many rules to achieve a hierarchy of goals. Production systems have a working memory that simulates human memory. Just as human memory is altered by learning, the production system has the capability to alter its working memory as rules evolve from representing general ideas to more specific ideas.

2.4 Overview of Classifier Systems

A classifier system is a specific type of production system. The learning classifier system (LCS) is an extension of a classifier system that uses a genetic

algorithm (GA) as its optimization algorithm. John Holland derives the foundation for genetics-based machine learning (GBML), where he defines schemata and discusses natural adaptation, robustness, fixed representation, and genetic operators (Holland, 1992). GBML is the mathematical basis for an LCS.

An LCS uses fixed-length strings called classifiers or production rules; each classifier is coupled with a measure of the classifier's usefulness, called strength. All classifiers are syntactically meaningful in the alphabet in which they are encoded. The input from the environment is processed against these classifiers; the system finds classifiers that match and rewards those that do. The output action from the system is produced by the best classifiers.

Basics of a Learning Classifier System

An LCS consists of three distinguishable components: the performance system, the credit assignment system, and the rule discovery system. The performance system interacts directly with the environment and has parallel rule activation. The credit assignment system or *apportionment of credit* (AOC) allows the system to evaluate and learn the relative value of the different rules. The *bucket brigade algorithm* (BBA) is one common AOC algorithm used. Finally, the rule discovery system is where the system "learns" new rules. It is here that the GA is used to generate rules and replace less useful rules.

Performance System

The performance system includes a set of condition/action rules, the classifiers. The classifiers are of the form <$condition_1$, $condition_2$,..., $condition_n$> <message>. In this research "$_n$" may equal only two or three. The condition of a classifier is a finite string from the coded alphabet, i.e. (0, 1), plus a wild card (#). The message or action is a finite string of the same alphabet with no wild card. A sample classifier is "010#1001##:01001" where "$_n$" is 2 and $condition_i$ is length 5.

Input is encoded in the same way as the conditions, but without the wild card. The input interface responds to the environment by trying to find conditions that match the input, and it then encodes information about the current state of the environment into finite length messages that represent possible actions. Those classifiers that have their conditions met then post their action to the message list. The message list or working memory contains all current messages generated by the input interface. The output interface decodes the best messages into actions that are applied to the environment.

General Operational Cycle

The general operational cycle of a genetics-based machine learning system as implemented using a LCS can be stated as follows (Smith and Goldberg, 1990):

1. Use the input interface to code the current environmental signal and place the resulting messages on the message list.
2. Determine the set of classifiers that are matched by the current messages.

3. Resolve conflicts between matched classifiers to determine the set of active classifiers.
4. Purge the message list.
5. Add the active classifiers' messages to the message list.
6. Generate action signals to the environment.
7. If a reward is present, allocate credit.
8. Repeat.

One execution of this sequence is called an "iteration." The GA is activated periodically after a specified number of operational cycles. The actions of the AOC subsystem are implemented in steps 3, 4, 5, and 7.

Learning can be introduced into a classifier system by two different methods: (1) as a result of experience, modifying additional parameters associated with classifiers and (2) actually changing the contents of the classifier list.

Apportionment of Credit

The first learning mechanism above is included by the AOC system where matching classifiers update their own statistical parameters. Each classifier keeps track of its strength (numerical value of usefulness) and its bid to become active. The bid is proportional to the strength (bid = strength $\times$ specificity). The specificity is the count of the number of 1's and 0's in the condition section of the classifier, where a higher specificity is assumed to depict a more relevant classifier. For example, classifier 1 is "1##0#10#1#:01101" and has a specificity of 5; classifier 2 is "101#01##10:11001" and has a specificity of 7; assuming that both match the input, classifier 2 is more

specific and is therefore preferable. Learning occurs when classifiers which have matched and are successful become stronger and are thus more likely to have their messages posted.

Bucket Brigade Algorithm

The classifier's role in achieving success from the environment is implemented by the BBA. When a classifier matches the input, it qualifies to participate in an auction to post its message. The more fit classifiers (higher strength and higher specificity) have an excellent chance of posting their messages. When a classifier is activated, it must pay an amount determined by the clearinghouse to all those classifiers previously active that were responsible for posting messages that led to its selection; thus the name "bucket brigade."

The reinforcement procedure rewards the classifier that is responsible for the system's achieving the correct behavior, thereby increasing the classifier's strength.

System Evaluation

The LCS with a GA is a stochastic system whose performance depends on chance. For example, two factors that may affect performance are the particular set of randomly generated classifiers initially chosen and the particular random number seeds used. Thus, in order to evaluate whether one execution of an LCS is better than another, we must evaluate whether the differences in the observed performances are due to chance. The literature states that results for LCS's are obtained by averaging

five to ten runs (Riolo, 1989; Roberston and Riolo, 1988; Janikow, 1993). We run various instantiations of our model many times (discussed in Chapter 6) and also use the tools of statistics to evaluate whether observed differences are statistically reliable (see Chapter 7).

2.5 Overview of Genetic Algorithms

The second learning mechanism is activated by the inclusion of the genetic algorithm. Genetic algorithms (GA), developed by J. Holland and colleagues, are search procedures based on natural genetics. The GA is theoretically proven to be robust through a search space. The first step in any GA problem is to encode the parameters as a fixed-length string. The GA searches for strings that are similar; therefore the encoding selected is vital. Holland (1992) defines a *schema* as a similarity template describing a subset of strings that have similarities at certain positions (bits). The classifiers in an LCS are an example of schemata.

Execution Cycle of a Genetic Algorithm

The basic execution cycle of a GA (Goldberg, 1989) is:

1. Select pairs of highly fit schemata.
2. Apply genetic operators to produce new strings.
3. Replace the weakest of the population with the new strings.

In each cycle, the GA is working with a population that is closer to optimum. Thus, the GA is guided towards areas of the search space that will lead to the improvement of

those strings that are represented. The classifier system in our research uses this GA with some enhancements.

The mathematical justification for the GA is stated by the Fundamental Theorem of GA or the Schema Theorem. The theorem states that *building blocks*--short, low order (those with a small number of fixed positions), above average schemata (highly fit relative to the population)--receive exponentially increasing trials in subsequent generations, where a generation is one complete pass through the GA execution cycle previously stated.

2.6 Genetic Algorithms in a Learning Classifier System

There are various implementations of genetic algorithms and each step in the basic execution cycle of a GA has its share of flexibility. We adhere to the GA suggested in Goldberg (1989) in his discussion of genetic-based machine learning.

Selection in Genetic Algorithms

The GA devised by Goldberg applies its procedure to classifiers randomly selected from those classifiers that have the highest strength and are thus highly fit. Several methods can be used to select pairs of parent classifiers in Step 1 of the execution cycle of a GA. In our research, "stochastic sampling with replacement," also known as "roulette wheel selection," is the method used (Goldberg, 1989). A roulette wheel is created with slots sized according to a parent's fitness in proportion to the average population fitness. When a parent classifier is needed, a "spin" determines

which is selected. A child classifiers replace unfit classifiers, as explained below.

Genetic Operators

The three fundamental operators used in Step 2 of the execution cycle of a GA are *reproduction*, *crossover*, and *mutation*. The most highly fit classifiers get to reproduce. *Reproduction* duplicates copies of classifiers in proportion to their fitness. The more fit will receive more copies. Simple *crossover* is applied by first selecting two highly fit parent classifiers and then choosing a random position on which to split. All positions after the split point are swapped between parents and the strings are reunited to create new offspring. *Mutation* is applied bit by bit to all classifiers at random.

Replacement in Genetic Algorithms

In Step 3 of the execution cycle of a GA, replacing the weakest portion of the population, the issue is the size of the population to be replaced. The method used in Goldberg (1989) uses a value called the "proportion to select" to define the percentage of the population to replenish. When an individual classifier is created, the individual selected to die is the most similar to it from a subset of the population as defined by the "crowding factor" (Goldberg, 1989). This subset is created by selecting at random a portion of the population, and choosing the least fit members of that portion.

2.7 Simple Classifier System Model

Our adaptation of a learning classifier system (LCS) to music is based on the simple classifier system (SCS) written by Goldberg (1989). An SCS is a subset of an LCS. The SCS is Goldberg's implementation of genetic-based machine learning in its most basic form without all the enhancements and advancements that one can use to tailor the system. This SCS differs from an LCS in the following manner: the message list is simplified so that it contains only the environmental messages generated by the input interface; the BBA is minimized so that payment is given only to currently active classifiers and not to previously active classifiers, and the GA is modified to limit the number of classifiers to replace.

The conditions for replacement of classifiers by the GA are low performance and similarity to the new classifiers created. Similarity is computed by counting those positions that match. Those classifiers with low performance which are more similar to the classifiers being inserted into the population are replaced.

2.8 Summary

Music learning is an area often studied by cognitive psychologists. It has been suggested that one way for a computer system to model this learning is by a production system. One specific type of production system—an LCS using GA--can be written to simulate the task of music induction. Our SCS, NEXTPITCH, is such a system.

Our research, to investigate the representation of music within the context of an LCS, is definitely needed. As of the date of this writing, there is no LCS system

written that learns music. Music representation, classifier format, and population size in an LCS do indeed warrant further investigation.

Chapter 3

Review of Current Literature

3.1 Introduction

In the current literature, there is no learning classifier system (LCS) that implements the domain of music. The closest application of an LCS has been the task of letter sequence prediction. Both letter sequence prediction and music learning are examples of sequential pattern learning; therefore principles from one may influence the other.

The research previously performed with music uses genetic algorithms (GA), and neural networks (NN). There is no uniform representation for music in these other computer-based systems. Not only does each system use a different encoding [including binary codes; binary-coded decimal (BCD), Gray code, integers, and real numbers], but each also represents different features of the music (including pitch, octave, duration and circle of fifths).

3.2 Learning Classifier Systems

Riolo (1986, 1988) created a classifier system to predict letter sequences. He uses binary coding to represent the alphabet. Vowels and consonants are separate

types, where vowels are type one, represented as "01" and consonants are type two, represented as "10". The type is joined with the representation for the alphabet where "a" and "b" have a decimal value of 1 and a binary coding of "00001", yielding the representation for "a" as "0100001" and "b" as "1000001".

Zhou (1991, 1987) adapts Riolo's classifier system to include conceptual learning. Zhou investigates the problem of representing relationships in an LCS. Using Riolo's representation for the alphabet, Zhou relates the letters by encoding the concepts of "prior" and "next" in binary. The concepts represent the distance between letters in a sequence (i.e., If "A" then "D"; the value of "next" is three and is encoded as "00011"). He creates and saves "chunks" of knowledge in the concept encodings, by grouping classifiers with common bit positions.

Schuurmans and Schaeffer (1989) discuss the representational difficulties of a classifier system. They speculate that current systems requiring multiple classifiers cannot adequately represent general concepts between classifiers. There exists no way in classifier representation to enforce the concept that message bits of multiple classifiers that match identical "#" bits have the same value. Continuing with this reasoning, Shu and Schaeffer (1989) introduce a Variable Classifier System that attempts to correct this problem. Variables are positions in a classifier containing the symbol "*", and are used to achieve high level symbolic representations. They permit relating information within a classifier by matching corresponding fields in a message, thereby parameterizing common features.

3.3 Genetic Algorithms

Other research has been done using genetic algorithms to determine which representation, binary or Gray code, is superior. Caruana and Schaffer (1988) compare binary coding to Gray coding using a genetic algorithm system. They use five functions, displaying a variety of characteristics that DeJong (1975) implemented to study GA performance; they also append one of their own to represent a test environment of six problems in function minimization. They find that when the task is to discover a best solution, Gray coding only statistically outperforms binary code on two of the functions and on the other four functions it is not statistically different from binary. They conclude that Gray code is often superior to, and not worse than, binary coding.

Horner, et al., (1991), uses genetic algorithms in music composition. He uses a technique called "thematic bridging," defined as the transformation of an initial musical pattern consisting of a specific number of notes to some other final pattern over a specified duration. Operations of change are chosen from the following: delete a note, rotate the pattern, and/or mutate a specific note. It is the GA that optimizes the sequence of operations that the thematic bridging applies. The operations are represented by real numbers between 0.0 and 1.0. Notes are not encoded in this system. The features of music that the operations modify include discrete pitch, amplitude or octave, and duration.

McIntyre (1994) uses Goldberg's Simple Genetic Algorithm to generate four-part Baroque harmony. The system uses a predefined melody as input, generates the

other three harmonies, and transposes and encodes the notes to the key of C major. It uses integers between 0-28 to represent the notes in treble and bass clef (using an alphabet of size 29). The results were favorable, with nearly every evolution of the genetic algorithm producing a believable harmony.

Laine, et al. (1994), uses a genetic algorithm system to compose new music in the musical style of the original material. The genetic algorithm is used as an optimizing technique to select the best mathematical function to describe the style of the original music. The musical input is a series of numbers that represent the pitches of a piece of music transposed to C major. A single musical line is represented without dynamics or other additional information; it is divided up by the smallest rhythmic unit in the piece.

3.4 Neural Networks

Neural networks, although similar in some ways, are significantly different from classifier systems. "Connectionist models (neural networks) learn exclusively from revision of connection strengths between elementary units" (Holland, 1986). In contrast, LCS's have the ability to generate new rules. We include ideas from the following neural network systems in our research:

Mozer (1994), uses a neural network to compose music by representing pitches with Sheppard's (1982) definitions of pitch height, chroma circle, and circle of fifths (to be defined). The encodings he uses represent pitch height as a real number, while chroma circle and circle of fifths are each represented by a six position binary value,

similar to Gray code, where the representation preserves the distance relationship among tones.

Todd (1991), trains a neural network to learn a musical style. He encodes to binary the twelve chromatic categories of each octave. The binary code applied is based on a localist scheme where each unit represents a pitch, emphasizing the characteristic of pitch adjacency. Using this code, each pitch representation is equally similar to every other (i.e. an A is "001," a B is "010," and a C is "100;" A and B, B and C, C and A are similar in just one position). He includes the features of pitch and duration to represent the melody input that is transposed to the key of C major.

Bharucha (1991) developed a neural network system, MUSACT, that extracts chords from tones and determines the key from the chords. MUSACT extracts the pitch height from the pitch class in different network layers to learn the relationships of chords and harmonic intervals. The model encodes the twelve chromatic pitch classes as 0, 1, ... , 11 with the tonic always at 0.

Miller, et al. (1992), compares two psychologically-based approaches to modeling the listener's development of the hierarchical representations of the metric structure of a melody. This work is based on the theories of Lerdahl and Jackendoff. The first approach, BEATS, is rule-based, and the second approach, BEATNET, is a one-layer cooperative/competitive network. Both approaches are concerned with determining the relationship of the metric structure in the melody. In BEATS, the period and phase are tied together; given a period, there is only one phase. In BEATNET, period and phase are independent, and the tempo information helps to resolve differences. The input to both BEATS and BEATNET is a symbolic

representation of note durations.

3.5 Population Size Issue

We consider three perspectives from the current literature that address population size (number of schemata or number of classifiers) and performance with GA and LCS's. The first reference considers a GA system where a formula for the optimal population size of strings is given. The next two references discuss the population size of classifiers in LCS's that use GA.

Goldberg (1985), derives an expression for the optimal population size of strings when taking into account only a GA system. He reports that the optimal population size is represented by the formula $pop = 1.65 \times 2^{0.21 \times length}$, where *length* is the number of positions in the strings. This formula suggests using very large populations for long strings and, conversely, small populations for short strings. For example, a population containing strings of length 30 would yield an optimal population size of 130 strings; in contrast, a population with strings of length 60 would yield an optimal population of size 2,389 strings.

Other researchers study population size of classifiers with various learning mechanisms, including GA (Robertson, 1988; Robertson and Riolo, 1988). It was found that "increasing population size increases performance, although with diminishing marginal returns after some point" (Robertson, 1988).

Shu (1992), examines initial populations and their size in relation to performance within an LCS. Her findings supplement the previous ideas, stating that

"there is a population size limit for each problem after which the performance decreases" (Shu, 1992).

3.6 Related Literature

Other research using computer systems to model music represent music including the features of pitch, accidental and duration.

Ebcioglu's (1992) expert system, CHORAL, is an algorithmic model whose task is to harmonize a four-part chorale in the style of J. S. Bach. He develops a new programming language, BSL, to perform this task. The knowledge base includes rules from multiple viewpoints, such as chord skeleton and the melodic lines of the individual parts. Each view emphasizes different characteristics of music, including pitch, accidental, duration, and melodic patterns.

Widmer (1992) uses an *explanation based learning* system written in Prolog to model music perception and intelligent musical learning where the harmonization of chords is the principal focus. He uses a human teacher to supply samples to guide the learning process. When the program constructs a plausible explanation for why a chord is good, then, if the teacher agrees, the explanation is stored. The technical knowledge base defines the basic music concepts of notes, intervals, as well as additional properties such as pitch and duration.

3.7 Summary

After investigating the current literature, we have not found any uniform

method of representing music in computer-based analysis. It seems that representations are chosen to meet the needs, demands, and limitations of the system being implemented. Apparently, binary representations are a favorite, but there does not appear to be any agreement on the features to include or the type of encoding to use.

Since we want to investigate the representation of music, we use an SCS, a simple adaptation of an LCS. Riolo (1995) suggests using a system with minimal complexity to study the effects of representation. The following chapter discusses our interpretation of the implementation of music learning using an SCS with GA, and Chapter 5 discusses the various representations of music that were tested in the system.

Chapter 4

SCS Model of Music Learning

4.1 Introduction

Our initial adaptation of music learning to Goldberg's SCS (1989) was a program called NEXTNOTE. Its initial task was to learn simple melodies by first learning to predict the pitch of the next note in a musical sequence. In our second model, NEXTSONG, we expanded the representation of music to include various additional features.

After carefully considering the inadequacies in both NEXTNOTE (Jones and Federman, 1993) and NEXTSONG (Federman and Jones, 1995), we formed the ideas for improvement which led to NEXTPITCH. NEXTPITCH remedies the problems in the previous systems by using various encodings for the notes, by using an increased set of classifiers, and by making a change in classifier structure.

NEXTNOTE, NEXTSONG, and NEXTPITCH learn nursery melodies, a subset of Western tonal music. Nursery melodies are simple in that there are usually no key changes in the melody, and they also include some pattern of phrase repetition. Thus, using nursery melodies makes the task of studying music learning more manageable.

In all of our music classifier systems, there is a common backbone that describes the nature of our adaptation of an SCS/LCS to our task. The classifiers represent possible note transitions in a piece of music. The environment is a piece of music, and the input interface is the encoding of the input notes. The message list contains the encoded representation of the input notes. The output interface picks the best match and predicts the next note. A reward is given for the correct prediction.

4.2 NEXTPITCH: The Model

NEXTPITCH, a revision of the earlier models, is modified to incorporate several different representations of notes and classifier formats. NEXTNOTE limited its learning to one aspect of the structure of music--pitch; in contrast, NEXTSONG included the additional features of contour and note length, thus enabling us to decide on the music features to include in NEXTPITCH (note name, octave, accidental, contour, duration, and interval). NEXTSONG modified the representation of pitch as well as the classifier format, thereby leading us to the changes in NEXTPITCH.

Similar to its predecessors, NEXTPITCH organizes the problem of learning by forming rules based on what has been heard. Like people, the model induces the rules of music by "listening" and "observing" the repetition of the melody. As the number of repetitions of a melody increases, the rules become more directed and the ability "to know" the melody is heightened.

Input to NEXTPITCH

NEXTPITCH has as its input the same nursery melodies as NEXTSONG. To maximize learning, the full length melodies "Yankee Doodle," "Old MacDonald," and "London Bridge" are transposed so that they are all in the key of C in 4/4 time. The symbolic encoding of the input notes contains the six features of note name, octave, accidental, contour, duration and interval. A complete description of the input encoding appears in Appendix A.

Processing of NEXTPITCH

NEXTPITCH follows the processing of NEXTSONG with some modifications based on our experience from the previous models. The principal processing follows the general operational cycle of the genetic-based machine learning system (Section 2.4; an "iteration" is one pass through all the steps), starting with the generation of random classifiers.

We tested various numbers of classifier because of our experiences in NEXTNOTE. The size of the population of random classifiers varies between 100, 200 or 400 classifiers depending on the testing procedure (to be discussed in Chapter 6). The set of 100 random classifiers does not permit the variability necessary for longer input with the encoding used. Melodies with 14 notes were learned more easily than melodies with 27 notes; the shorter input has a greater chance of matching the classifiers. Therefore, to improve learning, our choice was either to increase the number of classifiers or change the encoding.

The input notes are repeated in an infinite loop. Each classifier has two conditions, where each condition represents one note (the exact note representation is discussed in Chapter 5). Since the classifier's action represents the possible next note, the entire classifier represents a three note transition. Thus, the rules learned represent the pitch transitions based on the preceding two pitches. This is in contrast to our earlier model, NEXTNOTE, where the classifier had only one condition representing one note. NEXTPITCH more accurately simulates the human listener (most listeners can predict the note that follows a sequence of notes) by making its prediction based on a series of notes.

As each input note is read, the processing continues with the classifiers competing to match the previous two-note input and, to post their message. The classifier whose condition best matches the input (the winner) gets to predict the next note. When the prediction is correct, the AOC distributes a fraction of the reward among all classifiers that were both active and also had the correct condition/action (Riolo, 1988).

We realized that in NEXTSONG the GA had a greater effect on learning than in NEXTNOTE. It seems that it is easier for an LCS to learn with the five-bit pitch representation used in NEXTSONG than with the six-bit representation used in NEXTNOTE. This discovery led us to investigate various note representations called *whole-pitch* and *pitch-plus* in NEXTPITCH and will be discussed more fully in the analysis of NEXTPITCH.

In NEXTPITCH, we set the GA parameters to increase the GA's involvement.

This was incorporated by a change in the processing that activates the GA every 100 iterations and replaces the population of classifiers with a proportion to select of 30% and a crowding factor of size 3. Also to encourage learning, crossover is applied to all parents. The model uses a mutation rate of 10% on all classifiers.

Output of NEXTPITCH

The output report of NEXTPITCH measures how well the system succeeds at predicting the next note. The report is generated every 500 passes through the melody. The statistics in the output report include the iteration count (the number of notes processed), percent of predictions correct from the start, and the percent of correct predictions in the last three times through the entire melody. The report shows the gradual improvement or learning with time (Appendix B).

Representation of Music

The representation of music will be fully discussed in the following chapter.

4.3 Summary

NEXTPITCH emerged as a result of the inadequacies found in NEXTNOTE and NEXTSONG. We improved the representation to include more music features. We believe that the change in pitch representation helped to alleviate the problem of poor learning in NEXTNOTE when using 100 classifiers.

We found that adding more conditions does not hamper learning, and we

experimented with changing the classifier makeup by varying the number of conditions along with the consistency of the condition. We also saw that the number of classifiers can affect learning, which we investigated further, as discussed in the ensuing sections.

Chapter 5

Understanding the Problem of Representation

5.1 Introduction

The number of ways to represent music is limitless. We organized the problem of music representation using our experiences with our earlier models, the LCS literature, and music research. Our hypothesis is that changing the representation of the notes will change the performance of the system. To this end, the model was tested with numerous different note representations. Each musical note has multiple features; among them are dynamics, timbre, duration, and pitch, which consists of note name, octave, and accidental. Since notes are grouped into phrases and motives, the relationship between notes is important. The issue becomes which features to represent, how to represent them, and how they affect the performance of the system.

A rationale is offered for choosing each representation and testing the format of the classifiers. The majority of GA-based systems use binary alphabets, and a justification for this alphabet is also furnished.

5.2 Rationale for Note Representations

Each representation merges ideas from LCS and music research. The five

representations used are Binary Code Whole-pitch, Gray Code Whole-pitch, Binary Code Pitch-plus, Gray Code Pitch-plus, and Circle of Fifths Pitch-plus. The two *whole-pitch* representations simply assign a sequential number to each note, whereas the *pitch-plus* representations assign numbers using a more psychologically motivated scheme. This classification describes how our research encodes the pitch information; the whole-pitch representations encode all pitch information using a single number, while the pitch-plus representations subdivide the pitch information before encoding. The pitch-plus representations attempt to depict how people perceive pitch, allowing the LCS to simulate human learning. This research attempts to determine whether the simply numerical representation or the cognitively-based representation does better.

Before we begin, let's define some terms. A binary alphabet consists only of digits 0 or 1. Binary coding uses base 2 to assign numbers to each value and then encodes those numbers using the binary alphabet. The Gray code representation also uses a binary system with the binary alphabet; however, when going from one coded group to the next, only one bit in the group changes. The circle of fifths orders pitches by using the numerical interval of a fifth between neighbors.

Why Use a Binary Alphabet?

Goldberg (1989) states that the most representative encoding for LCS and GA consists of a binary alphabet. The following discussion uses Holland's (1992) notion of schema to support this idea (Section 2.5).

Counting Schemata

Holland (1992) suggests that for adaptive plans (i.e. GA) it is more desirable to have schemata with many positions deciding among a few attributes than to have a few positions with a range of many attributes. We compare two different base alphabets where the goal is to select an alphabet that contains the maximum number of schemata per bit of information. The schemata are developed by extending an alphabet. For example, to develop schemata from a binary alphabet (0,1), extend it to a ternary alphabet (0, 1, #), where a # is a "don't care," and then create strings from the ternary alphabet.

Using counting methods, we can compare different base alphabets to see which is best to use for a given problem. For example, let's compare a binary alphabet (V_1) with a ten-element alphabet (V_2). A binary alphabet (V_1) with a string of length 20 can represent approximately the same number of items as a ten-element alphabet (V_2) with a string of length 6. V_1 can represent 2^{20} or 1,048,576 distinct items, and V_2 can represent 10^6 or 1,000,000 distinct items. But the number of schemata in the two alphabets is extremely different. Including the "don't care" as a third element in V_1 and an eleventh element in V_2, V_2 has only 11^6 or 1,771,561 distinct schemata, while V_1 has 3^{20} or 3.48×10^9 distinct schemata. It is obvious a binary alphabet has substantially more schemata than a 10 element alphabet and is therefore a better choice.

Matching Schemata

It is clear that a binary alphabet has more schemata than a ten element alphabet; now we see how this affects schemata matching. A string matches a schema if all bits

at each location match: 1 matches 1; 0 matches 0, and "*" matches 0 or 1. Considering an exact alphabet match as one instance and the "don't care" match as one instance, each position can match in two ways. Thus, using the alphabets from the previous section, a specific string can match 2^{20} or 1,048,576 schemata in V_1 and 2^6 or 64 schemata in V_2, because in V_1 each of the 20 positions can match in two ways and in V_2 each of the six positions can match in two ways. A string has a more likely chance of matching in the binary alphabet than in the ten-element alphabet because the binary alphabet contains more schemata for the same number of items and will allow for more adaptation.

Ideas from LCS & GA

To reiterate, the binary alphabet has more schemata, and therefore a schema has a greater chance of matching another schema. In the SCS model, all this is of importance when the GA performs replacement. The replacement of similar population members occurs; the least fit classifiers are replaced by more fit classifiers that are the most similar to them. The better the chance of matching the least fit to the more fit classifier, the better the chance of substituting a more meaningful classifier.

Consequently, of the available codings, the binary alphabet is most often selected for use in LCS and GA. We use two examples of a binary alphabet: binary coding and Gray coding.

Binary Coding

In this research, we use two binary coding representations, each with different motivations. The Binary Code Whole-pitch directly encodes the twelve chromatic steps of a scale by sequentially numbering them based on their proximity in the scale. In comparison, the Binary Code Pitch-plus is a more musically-oriented representation, encoding just the note name and then joining it to octave and accidental information.

Gray Coding

Other GA researchers have suggested that a Gray coding is certainly no worse, and often superior to, binary coding (Carauann and Schaffer 1988). Thus three representations use Gray code, again with different motivations. The Gray Code Whole-pitch directly encodes the twelve chromatic steps of a scale, while the Gray Code Pitch-plus is similar to Binary Code Pitch-plus but encodes just the note name. The third Gray code representation, Circle of Fifths Pitch-plus, Gray codes pitches according to their perceived proximity.

Ideas from Music Cognition

To allow this LCS to parallel the human learning of Western tonal music more closely, ideas from music cognition studies are included. All of the representations in this research have some musical basis, but it is the pitch-plus representations that are more psychologically motivated. We discuss how different approaches in music perception influenced our representations and the features of music we include.

Music Perception

There are two theories that comprise the backbone of our note representations. The first, based on Sheppard (1982), emphasizes the way people perceive the similarity of pairs of pitches. Sheppard theorizes that there is a multidimensional representation of pitch using pitch height, the chroma circle and the circle of fifths. Pitch height refers to how pitches differ by octave, the chroma circle refers to ordering pitches that are a tonal half-step apart in the scale, and the circle of fifths groups pitches by key, emphasizing the importance of the interval of a fifth in music perception. Several representations in our study use Sheppard's ideas: (1) all pitch-plus representations allow for the pitch height dimension, (2) Gray coding the circle of fifths allows neighboring pitches to be closely represented, and (3) the chroma circle is used in the whole-pitch representations.

The second theory we consider refers to Dowling's (1993) observation of the way people perceive and come to learn the music of their culture. He suggests that pitch categories (classes) are programmed into the brain just by listening to music. There are twelve pitch classes, C, C$^{\#}$, D, D$^{\#}$, E, F, F$^{\#}$, G, G$^{\#}$, A, A$^{\#}$, and B. In his studies, Dowling (1993) uses melodies spanning only those classes in the C major scale (C, D, E, F, G, A, and B). Two of our representations use these pitch classes: Binary Code Pitch-plus and Gray Code Pitch-plus.

Table 1 summarizes this research's representations in relation to the music perception ideas discussed in this section by delineating which topic applies to each representation.

Table 1: Summary of Pitch Information

Representation	Pitch Class	Pitch Height	Chroma Circle	Circle of Fifths
Binary Code:				
Whole-pitch			X	
Pitch-plus	X	X		
Gray Code:				
Whole-pitch			X	
Pitch-plus	X	X		X

Note Relationships

Other music theorists say it is the relationship between successive notes that permits music learning. Both adults and children use contour and intervals to recognize familiar melodies (Andrew and Dowling, 1991; Dowling and Harwood, 1986; Trehub, et al., 1985). Other researchers say that the duration of notes and the rhythmic contours are important to music acquisition (Davidson, McKernon and Gardner, 1981). The representations in this study include the relational qualities of contour, duration and interval between two notes, together with the pitch attributes of note name, octave, and accidental to represent a subset of features from Western tonal music (see Section 5.4).

5.3 Rationale for LCS Format Testing

This model also experiments with different classifier formats. Classifier systems traditionally use the same format for each condition (Goldberg, 1989; Riolo,

1986, 1988). Other researchers are experimenting with changing this format. Shu and Schaeffer (1989) introduce a Variable Classifier System (VCS) that can be used to move information between conditions so that the relationship between conditions can be captured. The VCS also has conditions of varying lengths (Shu, 1992).

In music perception, the relation between notes is an important feature. Therefore, it is logical to add a condition that represents only the relation between notes. This allows contrasting the traditional approach--where all conditions have the same content--and the updated approach. In the updated approach, the format of the conditions differs; two conditions have the same content, consisting of pitch information, and a third condition has a different content, consisting of only the relational information.

5.4 Note Representations Defined

This section defines the different note representations used in this thesis. There are two categories of encodings: those that represent pitch information and those that represent relational information. This research tests five different encodings for pitch information and one encoding for the relational information. The manner in which the classifier includes the relational information is tested in two ways.

5.4.1 Pitch Information Representations

The music features included in all representations are pitch name, accidental and octave. All pitch information representations encode notes in a number of octaves

around middle C; the octave starting with middle C is called the third octave. All representations use five bits to encode pitch; however, the way these five bits are broken up differs in the various representations.

Whole-pitch Representations

The whole-pitch representations use all five bits as a group to encode the pitch information. The five bits encode pitch name, accidental, and octave as a single number. This code allows for a range of 30 pitches, with two encodings for rests. All notes of the chromatic scale are sequentially numbered, starting with $G^{\#}$ below middle C as zero, A as one, and continuing up for 30 notes to $C^{\#}$ in the fifth octave. This numbering labels middle C as 4 and $C^{\#}$ in the fifth octave as 29, with rests as 30 and 31. There are two whole-pitch representations: Binary Code and Gray Code Whole-pitch.

Binary Code Whole-pitch

Binary Code Whole-pitch converts the decimal value of each pitch to its five-bit binary equivalent (Table 2). The Binary Code Whole-pitch encoding thus symbolizes middle C (note 4) as "00100."

Gray Code Whole-pitch

This representation uses a Gray code to assign a five-bit binary value to each pitch (Table 2). Gray coding encodes values so that, in going from one pitch to the

next, only one bit in the code group changes. Therefore neighboring pitches are separated by a one-bit distance in the encoded representation. The Gray Code Whole-pitch encoding thus symbolizes middle C as "00110."

Table 2: Whole-pitch Codes

Pitch	Decimal Value	Binary Code	Gray Code
G♯2	0	00000	00000
A 2	1	00001	00001
A♯2	2	00010	00011
B 2	3	00011	00010
C 3	4	00100	00110
C♯3	5	00101	00111
D 3	6	00110	00101
D♯3	7	00111	00100

Pitch-plus Representations

The pitch-plus representations use three bits to represent pitch name and supplement those with two more bits, one bit to represent the accidental and one to represent the octave, yielding a representation that totals five bits. Each encodes the note name, so we have a pitch "class" for the name of the note, where C represents C or C♯, and the accidental is represented separately. The accidental bit represents a natural as "0" and a sharp as "1" (scales are selected so there are no flats). The octave bit represents a pitch in the third octave as "0," and a pitch in the fourth octave as "1." The familiar melodies tested here only span one octave; therefore a representation that

allows only two octaves is sufficient. The pitch-plus representation subdivides the five bits of "01110" as follows: bits 1-3 or "011" represents note name; bit 4 or "1" is the accidental; and bit 5 or "0" is the octave. There are three pitch-plus representations: two use pitch class, and one uses the circle of fifths.

Table 3: Pitch Class Codes

Pitch	Decimal Value	Binary Code	Gray Code
C	0	000	000
D	1	001	001
E	2	010	011
F	3	011	010
G	4	100	110
A	5	101	111
B	6	110	101
Rest	7	111	100

Pitch Class Codes

In these two representations, the seven pitches of the C major scale are used to define the pitch class representations. The pitch class encodings assign decimal numbers to each pitch class (Table 3). Then the decimal number is interpreted in binary in two ways, in binary code and in Gray code. In Binary Code Pitch-plus, middle C (octave 3) is "00000" and F♯3 is "01110." In Gray Code Pitch-plus, middle C (octave 3) is "00000" and F♯3 is "01010" (Table 5).

Circle of Fifths Pitch-plus

This representation orders pitch classes by the circle of fifths. The circle of fifths reflects the relationship between pitches where pitches are separated by the interval of a fifth. This relationship also groups tones that are in the same key. The decimal numbering represents F as 0; C as 1; G as 2; D as 3; A as 4; E as 5; B as 6, and rest as 7 (Table 4). These values are given a binary Gray code: F is 000; C is 001; G is 011; D is 010; A is 110; E is 111; B is 101, and a rest is 100. This encoding thus symbolizes middle C as "00100" and in the same octave F# as "00010" (Table 5).

Table 4: Circle of Fifths Code

Circle of Fifths	Decimal Value	Gray Code
F	0	000
C	1	001
G	2	011
D	3	010
A	4	110
E	5	111
B	6	101
Rest	7	100

Pitch Information Summary

Table 5 contrasts the five pitch information representations for the notes C and C# in the third octave, F and F# in the third octave, and C and C# in the fourth octave.

Table 5: Sample Representations

	C 3	C♯ 3	F 3	F♯ 3	C 4	C♯ 4
Whole-pitch						
Binary Code	00100	00101	01001	01010	10000	10001
Gray Code	00110	00111	01101	01111	11000	11001
Pitch-plus						
Binary Code	00000	00010	01100	01110	00001	00011
Gray Code	00000	00010	01000	01010	00001	00011
Circle of Fifths	00100	00110	00000	00010	00101	00111

5.4.2 Relational Information

Music is not just a series of isolated notes; instead the patterns are formed by adjacent notes. This research looks at how, by encoding this relationship, learning can be affected. The model encodes the relations of contour, duration and interval between two adjacent notes; this encoding supplements each of the pitch representations to yield a more descriptive representation. All the relational information bits use Gray code.

Contour

Contour is defined as the pattern of ups and downs of pitch from note to note in a melody. This feature is represented by a two bit Gray code where "00" means no change; "01" means up; "10" means down; and "11" means that there was no prior note information.

Duration

Duration is defined as the correlation of time between two notes. This feature is represented by a two-bit Gray code where "00" means they are the same length; "01" means the current note is longer, "10" means the current note is shorter, and "11" means there is no prior note information.

Table 6: Interval Encoding

Interval	Decimal Value	Gray Code 3 Bit
Major 2nd	0	000
Unison	1	001
Major 3rd	2	011
Perfect 5th	3	010
Perfect 4th	4	110
Minor 3rd	5	111
Major 6th	6	101
Minor 2nd	7	100

Interval

Interval is the pitch distance between two notes. The eight most common intervals used in nursery melodies are sequentially numbered according to their frequency of use. A major 2nd is represented as 0, unison as 1, major 3rd as 2, perfect 5th as 3, perfect 4th as 4, minor 3rd as 5, major 6th as 6, and a minor 2nd as 7. This interval is represented by a three-bit Gray code (Table 6). Thus a major 2nd becomes

"000", unison "011", major 3rd "011", perfect 5th "010", perfect 4th "110", minor 3rd "111", major 6th "101", and a minor 2nd becomes "100."

5.4.3 Complete Representations

The complete representation for a note is either five bits or twelve bits depending on the format of the classifier being tested. We assign each format a mnemonic name.

There are two classifier formats tested herein. In the first format, each classifier has two conditions. Two divisions reflect this case. In the first division, each of the two conditions consists only of the five-bit pitch representation. We call this division "P_2" reflecting $pitch_1$ and $pitch_2$ in $condition_1$ and $condition_2$ respectively (Fig. 1). In the second division, each of the two conditions consists of a five-bit pitch representation supplemented by seven bits of relational information, yielding twelve bits. The relational information represents the relationship between the previous note and the current note. We call this division "P_2R_3". P_2 reflects the pitch in each of the two conditions, and R_3 reflects the supplemental relational information which actually represents the relationship between three notes (the previous note, $condition_1$'s note and $condition_2$'s note--Fig. 1). In each condition of P_2R_3, bits 1-5 represent pitch, bits 6-7 represent the contour, bits 8-9 represent the duration, and bits 10-12 represent the interval. Suppose you have a sequence of notes A, C, and D and a classifier representing the sequence C D. In this case, $pitch_1$ is C and the seven bits of relational information in $condition_1$ shows the relationship between A and C. Similarly, $pitch_2$ is

D and the seven bits of relational information in $condition_2$ shows the relationship between C and D.

Figure 1: Classifier Formats

$Pitch_2$ "P_2"

$Condition_1$	**$Condition_2$**
01010	01010
pitch	pitch

$Pitch_2$ $Relation_3$ "P_2R_3"

$Condition_1$				**$Condition_2$**			
01010	01	10	011	01010	01	10	011
pitch	contour	duration	interval	pitch	contour	duration	interval

$Pitch_2$ $Relation_2$ "P_2R_2"

$Condition_1$	**$Condition_2$**	**$Condition_3$**		
01010	01010	01	10	011
pitch	pitch	contour	duration	interval

In the other classifier format, each classifier has three conditions with different content. In this test, $condition_1$ has the same content as $condition_2$ and each consists of only the five-bit pitch representation. $Condition_3$ contains the seven bits of information representing the relation between notes 1 and 2. We call this division "P_2R_2". Here P_2 reflects the pitch in each of the first two conditions and R_2 reflects the supplemental relational information between two notes in $condition_3$ (Fig. 1). In P_2R_2, bits 1-2 of $condition_3$ are the contour, bits 3-4 are duration, and bits 5-7 are interval.

In all test divisions, the action represents the predicted next note and contains only the appropriate five-bit pitch information. The representation is the same as the pitch representation in the conditions: whole-pitch or pitch-plus.

5.5 Summary

This model tests various ways of representing pitch (note name, octave and accidental) along with the relational attributes of contour, duration, and interval. It uses a binary alphabet to allow for the most representative schema.

The note representations use both binary coding and Gray coding along with two pitch information representations: *whole-pitch* and *pitch-plus*. *Whole-pitch* is the direct encoding of the chromatic steps while *pitch-plus* is the more psychologically /musically-oriented representation.

The model also tests another method of enabling an LCS to be more representative of the domain: different classifier formats. This research tests this by using a condition that symbolizes the relationship between two other conditions.

Chapter 6

The Methodology

6.1 Introduction

The research was divided into four phases of testing: Parameter Selection, Evaluating the Model, Applicability of Learning, and Size of Classifier Set Impact. In Parameter Selection, we establish the parameters for the LCS. The parameters consist of the random number seeds, the classifier population size, and the initial classifiers. The primary testing phase, Evaluating the Model, is the data collection phase to determine how different representations affect LCS learning. In the Applicability of Learning procedure, we attempt to reproduce the results obtained in the earlier phases on a broader collection of melodies. In the Size of Classifier Set Impact phase, we vary the number of classifiers in the population and determine the effect on the various classifier formats.

As previously stated, NEXTPITCH is a modified version of NEXTNOTE and NEXTSONG, earlier systems that adapt Goldberg's (1989) Simple Classifier System to the task of music learning. In NEXTPITCH, the environment procedures of the performance subsystem are rewritten to allow for the new representations and classifier formats.

6.2 Parameter Selection

The Parameter Selection procedure uses the P_2R_3 classifier format with the Gray Code Whole-pitch representation on the three nursery melodies "Old MacDonald," "Yankee Doodle," and "London Bridge." We select P_2R_3 because it has a standard classifier format (conditions of the same format) that contains the most information of all the three classifier formats. The Gray Code Whole-pitch representation is chosen because of its similarity to the encoding used in NEXTNOTE to test the value of Gray code.

In every execution of NEXTPITCH, one complete test execution is 300,000 iterations, where an iteration represents the processing of one three-note group.

Processing for Random Number Variance

The testing procedure begins by determining the variance due to the random number generator and establishing the optimal size of the classifier population as follows:

1. Using ten different random seeds, we generate ten classifier sets, each with a population of 400 classifiers. Also, we save the next random number generated after each set of classifiers is created so that there are now ten saved random numbers from which we pick the seeds for future runs.

2. Five numbers are randomly selected from the ten previously saved numbers. These five numbers serve as the random seeds for each
of five parallel test executions of NEXTPITCH. The literature shows that in other studies results for LCS testing are averaged

over at least five runs (Booker, 1989; Riolo, 1991; Janikow, 1993; Carse and Fogarty, 1994). Results from these five parallel tests will be averaged to meet this criterion.

3. For each song, for each of the five random numbers, we execute NEXTPITCH as follows:
   a. using each classifier set with population of 400 classifiers.
   b. using each classifier set, but decreasing the population to include the first 200 classifiers.
   c. using each classifier set, but decreasing the population to include the first 100 classifiers.

Selection of Classifier Populations

From each of the executions of NEXTPITCH in Step 3, we determine the best performance of each run. The best performance is defined as the percentage of correct predictions during the last three times through the melody. Then a "set average" of the best performance for each classifier set, for each song, is calculated for the five random number tests. Finally, an overall average is computed from these set averages for each classifier set and for each song. The detailed results are presented in Appendix C (Section 1), separated by each of the three melodies and each of the three classes of population sizes.

Next, the optimal population size is decided by selecting the population that has the best overall average. Table 7 summarizes the overall averages for each melody for each population size. It is evident that the classifier population of 400 has the highest success rate for all melodies, with "London Bridge" at 60.18%; "Old MacDonald" at 58.2%, and "Yankee Doodle" at 51.38%. Therefore, we opt to use the classifier population of 400 for the subsequent phases of testing.

Table 7: Parameter Selection

Overall Averages of Best Performance

	Population Size		
	100	**200**	**400**
London Bridge	36.70%	45.58%	60.18%
Old MacDonald	36.48%	45.72%	58.20%
Yankee Doodle	38.48%	44.24%	51.38%

Of the ten classifier sets that contain 400 classifiers we determine which gave the best and worst results for each melody. Table 8 specifies the minimum, or worst, performance as well as the maximum, or best, performance for each of the original melodies accompanied by the number of the classifier set that produced the results. There are four sets (1, 2, 5, and 6) which produced the six results. These four sets are then promoted to the next phases of testing.

Table 8: Population Size of 400:

Minimum/Maximum Performance For Each Melody

	Minimum	**Set #**	**Maximum**	**Set #**
London Bridge	34%	2	83%	6
Old MacDonald	34%	6	96%	2
Yankee Doodle	30%	5	76%	1

Parameter Selection is now complete with the successful selection of classifier sets numbered 1, 2, 5, and 6 from the classifier sets of size 400.

6.3 Evaluating the Model

After the classifier sets are established, we continue to evaluate the model by running NEXTPITCH through one complete test execution of 300,000 iterations for the following:

> For each of the three melodies "Old MacDonald," "Yankee Doodle," and "London Bridge,"
>
> > For each of the four classifier sets selected from Parameter Selection,
> >
> > > For each of the five random numbers,
> > >
> > > > For each of the five representations: Binary Code Whole-pitch, Gray Code Whole-pitch, Binary Code Pitch-plus, Gray Code Pitch-plus, and Circle of Fifths Pitch-plus.

All the test runs described above are executed for each of the following three classifier formats:

1. "P_2": Base line with pitch information only. There is no relational information. The classifier set has structure $Condition_1$, $Condition_2$/Action (C_1, C_2 / A), where C_1 equals C_2 and includes only pitch information.
2. "P_2R_3": Representation with both pitch and relational information. The classifier set has structure $Condition_1$, $Condition_2$/Action (C_1, C_2 / A), where C_1 equals C_2 and includes both pitch information and all relational information.
3. "P_2R_2": Representation where conditions of the classifier are different. The classifier set has structure $Condition_1$, $Condition_2$, $Condition_3$/Action (C_1, C_2, C_3 / A), where C_1 equals C_2 and includes only pitch information and C_3 includes only relational information.

The processing encompasses 900 runs: three melodies * four classifier sets * five random numbers * five representations * three classifier formats.

6.4 Applicability of Learning

The purpose of this procedure is to determine if the results obtained for the various representations can be applied to other melodies or whether they are specific to the tested melodies. To do this, we partition the original melodies based on their information content and then select three new melodies that are in the same partition as each of the original melodies. We run the same 900 tests using the three new melodies.

Overview of Information Theory

"Information theory" (Winston, 1993) says you can calculate the information content of a datum based on what Winston calls *disorder* or *the amount of uncertainty*. This theory can be adapted to music using the notion of musical expectancy. What is expected to be the next note of a sequence is based on what is in the previous sequence. As the chance of predicting the next note increases, there is less uncertainty (disorder) and, consequently, less information is transmitted by each note.

The traditional symbol for disorder, H, measures the amount of information of a *message*. Assuming there is no previous knowledge available, each *message* is a note and $H = log_2 (n) = log_2 (1/p)$, where n is the number of possible events and p is the

probability of any one of the equiprobable events (e.g. one pitch occurring out of all equiprobable pitches).

After training, the classifier population in an LCS will come to be more representative of the notes that actually occur in a melody. The LCS learns to expect the various pitches of a melody in proportion to their probability of occurrence. The amount of information conveyed by any particular pitch depends on the degree to which it is expected or unexpected, where expected events convey little information and unexpected events convey more. A weighted average of the information conveyed by each possible event (i.e. note) is computed,

$$H = \Sigma (p_i \times log_2 1/p_i)$$

where the sum is over the i possible events.

If there is even more knowledge of the frequency of occurrence based on previous notes, the p_i represents the conditional probability of pitch i, i.e. p(pitch i| pitch j is the previous note). In this case there is even less uncertainty (disorder).

Information Theory and The Model

In this research, the information content of a melody is determined by the predictability/unpredictability of the melody in combination with the predictions of the LCS, where the amount of knowledge about the previous notes affects these predictions and the H value.

The *H* value is different for each classifier format. $\mathbf{P_2}$ represents only pitches with no relational features, $\mathbf{P_2R_3}$ represents relational features from three notes and $\mathbf{P_2R_2}$ represents relational features from two notes.

We calculate the average amount of information per note of the base melodies--"Old MacDonald," "Yankee Doodle," and "London Bridge"--on the basis of the following:

1. relative frequency of each pitch
2. the probability of each pitch based on the properties of the previous note
3. the probability of each pitch based on the properties of the previous two notes
4. the probability of each pitch based on the properties of the previous three notes

Partitioning into Melody Classes

To reiterate, the *H* value is a measure of the certainty/uncertainty of a note in relation to a specific melody. The more expected a note is, the less information it conveys and a lower its *H* value. The melodies with lower *H* values are more predictable and are easier for an LCS to learn; conversely, melodies with higher H values are less predictable and harder to learn.

Using an auxiliary program, we computed the average amount of information per note of the original melodies where the notes are equiprobable P_e; the relative frequency of each pitch is P_i; the probability of each pitch based on the properties of the previous note is $P_{(i/j)}$; and the probability of each pitch based on the properties of

the previous two notes is $P_{(i/jk)}$ (Table 9). We selected matching melodies based on P_i through $P_{(i/jk)}$ rather than any larger sequences, because, as more notes are considered, it becomes more difficult to find other melodies that closely match the original melodies.

Table 9: Partitioning of Melodies into Classes

		Average Information Content Per Note (*H*)			
Melody	**# Notes**	**Equiprobable**	**One Note**	**Two Notes**	**Three Notes**
		P_e	P_i	$P_{(i/j)}$	$P_{(i/jk)}$
Class 1					
Old MacDonald	26	4.7	2.408	1.226	0.167
Twinkle, Twinkle	28	4.8	2.471	1.231	0.183
Class 2					
London Bridge	24	4.5	2.288	1.287	0.364
Farmer in the Dell	24	4.5	2.235	1.286	0.459
Class 3					
Yankee Doodle	30	4.9	2.439	1.406	0.313
Auld Lang Syne	28	4.8	2.496	1.469	0.472

We computed the H values of many melodies and selected the three that were the most similar. "Twinkle, Twinkle, Little Star" has values similar to "Old MacDonald"; thus both are placed in Class 1. "Farmer in the Dell" has values similar

to "London Bridge," placing both in Class 2. "Auld Lang Syne" has values similar to "Yankee Doodle," and both are placed in Class 3. For all melodies in a class for $P_{(i)}$ and $P_{(i/j)}$ and also for Class 1 $P_{(i/jk)}$, the correlation was accurate to the nearest tenth.

Appendix F lists a sampling of melodies with their respective values. Table 9 summarizes Appendix F, showing the partitioning of the original melodies (shown in italics) and the new melodies selected because they have comparable *H* values. This table shows the *H* values of various notes along with the melodies that are found to be in each class. Class 1 has the overall lowest *H* value for $P_{(i/jk)}$, Class 2 follows with both $P_{(i)}$ and $P_{(i/j)}$ having lower values than Class 3.

Applicability of Learning Processing

Using the three new melodies, we repeat the "Evaluating the Model" processing with the established random number seeds and classifier set populations. This involves repeating the 900 tests for the new melodies.

6.5 Size of Classifier Set Impact

Additional runs of the model were made to test whether 400 classifiers was the optimal population size. According to Robertson (1988), there is a point where the size of the population will not measurably increase performance when using classifiers of the same length.

The Size of Classifier Set Impact procedure is similar to Evaluating the Model: NEXTPITCH executes for 300,000 iterations, uses the same original melodies, the five

random numbers, all note representations, and all three classifier formats. The differences in this phase are that we limited the testing to classifier population set 1 only, and we decreased the size of the population to 100 and 200 classifiers. That is, population set 1 was executed twice for each of the previous tests (three original melodies × random numbers × five note representations × three classifier formats), once for population size 100 and once for population size 200. This was a total of 450 additional tests added to the 225 tests for the population size of 400 classifiers.

6.6 Summary

Each of the four procedures of testing relates to the other. The first procedure--Parameter Selection--establishes the base line parameters for the subsequent phases. The melodies in the second phase--Evaluating the Model--provide the partitioning so that additional melodies can be found to meet the criteria for the third phase of testing--Applicability of Learning. The second phase is the mainstay of this research and is repeated in the third phase. The final phase--Size of Classifier Set Impact--attempts to relate classifier population size to classifier format.

The methodology described provides the results from the Evaluating the Model and Applicability of Learning procedures that can be used to determine the overall superior note representation and the best overall classifier format.

Chapter 7

Analysis of NEXTPITCH's Results

7.1 Introduction

The ensuing discussion organizes the results from the three testing phases Evaluating the Model, Applicability of Learning, and Size of Classifier Set Impact. All phases execute the same program; they differ only in their data. The first two phases use different sets of melodies, whereas the third phase uses the original melodies, but limits itself to classifier set 1. The details of the results offered for consideration in this chapter are found in Appendices E, G, and K respectively.

After briefly discussing the results, we analyze our findings in the remainder of the chapter. Our interpretation of the results are guided by these inquiries:

1. Which runs are the minimum/maximum performers of this research?
2. Does NEXTPITCH support documented findings comparing performance and size of classifier set?
3. Which type of representation, binary code or Gray code, is superior? Do these results confirm the results in the literature?

4. Which type of representation--simple numeric or psychologically motivated representation of pitch--is favored? Does the circle of fifths representation perform as well as the other representations?

5. Which model format of classifier condition (same or different format) is preferable? Does a change in structure affect performance? Does the grouping of relational information affect performance?

6. Do the melody classes affect performance?

7. Are the results from Evaluating the Model phase applicable to the results from Applicability of Learning phase? (Are the results applicable to other melodies?)

These questions are answered in the order of their significance.

Attempting to answer some of these questions led us to modify the data being analyzed and to perform additional tests. In attempting to answer question 1, "Which runs are the minimum/maximum performers?", we first considered our entire set of results from the Evaluating the Model and Applicability of Learning procedures (also referred to as the "original" results). We observed a bias toward the Binary Code Whole-pitch and the P_2R_3 classifier formats within the original data. Because the classifier sets were chosen based on those which had the minimum and maximum performance in the first set of runs (Section 6.2), our results span a wide range of values. Because of this knowledge, we then restricted our data set to a subset of values we define as "trimmed results."

When trying to answer question 2, "Does the size of the classifier set affect performance?", we noticed that the original choice of 400 classifiers might not have been the best for all formats. Therefore, further runs were performed in the Size of

Classifier Set Impact phase. The results showed a relationship between classifier set size, classifier length, and performance.

Since an LCS using GA is a stochastic system, we need to evaluate whether the observed differences in tests are statistically reliable. To assist in exploring questions 3 through 7, we employed a fully factorial analysis of variance (ANOVA). The factors included were classifier format, note representation, and melody class. We observed that P_2R_2 classifier format outperforms the other formats (Fig. 2), melody class 1 outperforms classes 2 and 3 (Fig. 4 and Fig. 7), pitch-plus outperforms whole-pitch, and that there is no superiority between binary and Gray code.

Terminology

A clarification of terminology used in the subsequent discussion is necessary at this point. A "test" or "run" is defined as one execution of NEXTPITCH using one specific classifier format, one note representation, one melody, one classifier population, and one random number seed. In all tables and figures, the "best performance results" for each run are based on the percent of correct predictions the last three times through each melody. For reporting purposes, "All Formats" refers to treating the classifier formats as one unit. [The three formats are P_2 (C_1, C_2 / A, where C_1 and C_2 include only pitch information), P_2R_3 (C_1, C_2 / A, where C_1 and C_2 include both pitch information and all relational information), and P_2R_2 (C_1, C_2, C_3 / A, where C_1 and C_2 include only pitch information and C_3 includes only relational information).] Melody classes refer to the three groupings based on disorder (Section 6.4).

In all histograms, the y-axis represents the number of runs. The bins on the x-axis labeled "percent correct" each include tests whose best performance is within a range of values (i.e. the 50% bin includes values from 49.5% to 59.5% inclusive).

All runs were performed on Dynex P5, Pentium 60Mhz processors, using Borland Turbo Pascal 5.5. The typical run for a population size of 100 classifiers consumed 10 minutes of CPU time, for 200 classifiers 15 minutes of CPU time, and for 400 classifiers 30 minutes of CPU time. This accumulates to 450 hours for the Evaluating the Model and Applicability of Learning phases, respectively and to 93.75 hours for the Size of Classifier Impact phase.

7.2 NEXTPITCH's Results

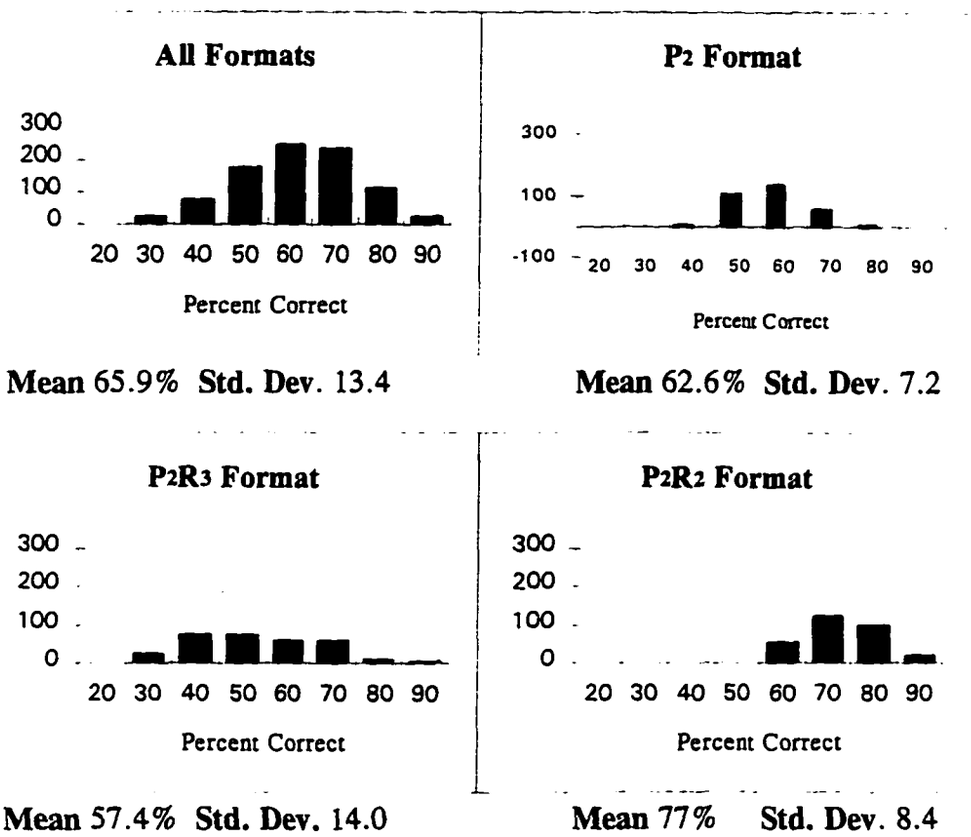
As a start, we used histograms to organize the results of the runs within each of the two phases of testing, Evaluating the Model and Applicability of Learning. This section discusses the raw results prior to trimming; the trimmed results are discussed in Section 7.3. The general insights gained from these histograms led us to the subsequent ANOVAs. The most significant result is that P_2R_2 format appears to be superior to the other formats.

Evaluating the Model--Raw Results

Charts in Appendix E show the results from the Evaluating the Model phase divided up according to note representation, classifier format, and melody titles. Fig. 2 organizes these results by classifier formats. The first histogram represents the best

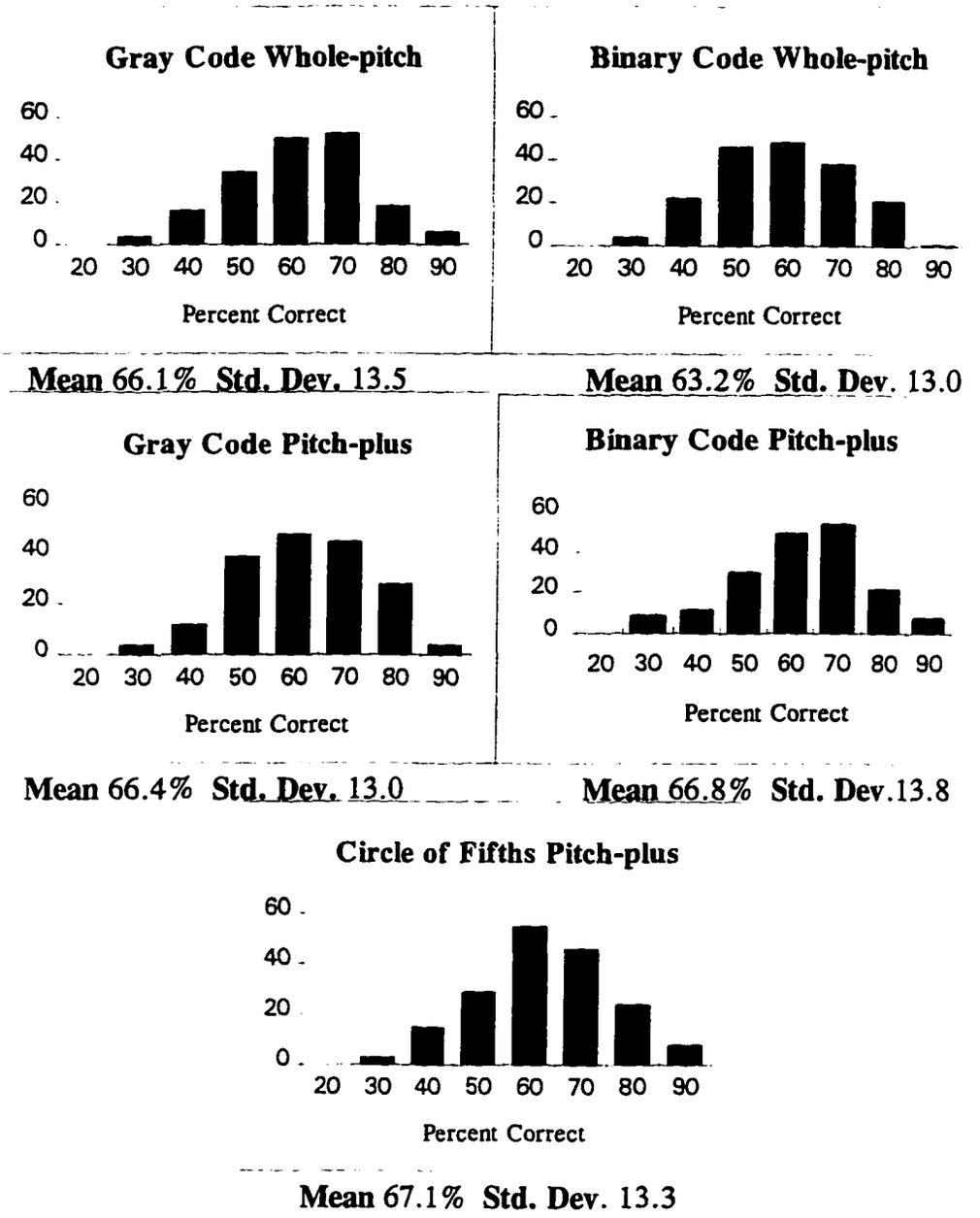
performance results for all classifier formats, followed by the results separated by classifier format. The results show the number of runs in which the model achieved each percent correct, ranging from 30% to 90%. Two ideas from Fig. 2 warrant further analysis. The first is that the P_2R_2 format appears to be superior to the other formats. Secondly, P_2R_3 demonstrates a flat distribution, showing a bias in the results towards this format. Trimming was used to eliminate this bias, see Section 7.3.

Figure 2: Evaluating the Model's Best Performance Results by Classifier Format



y-axis is the number of tests

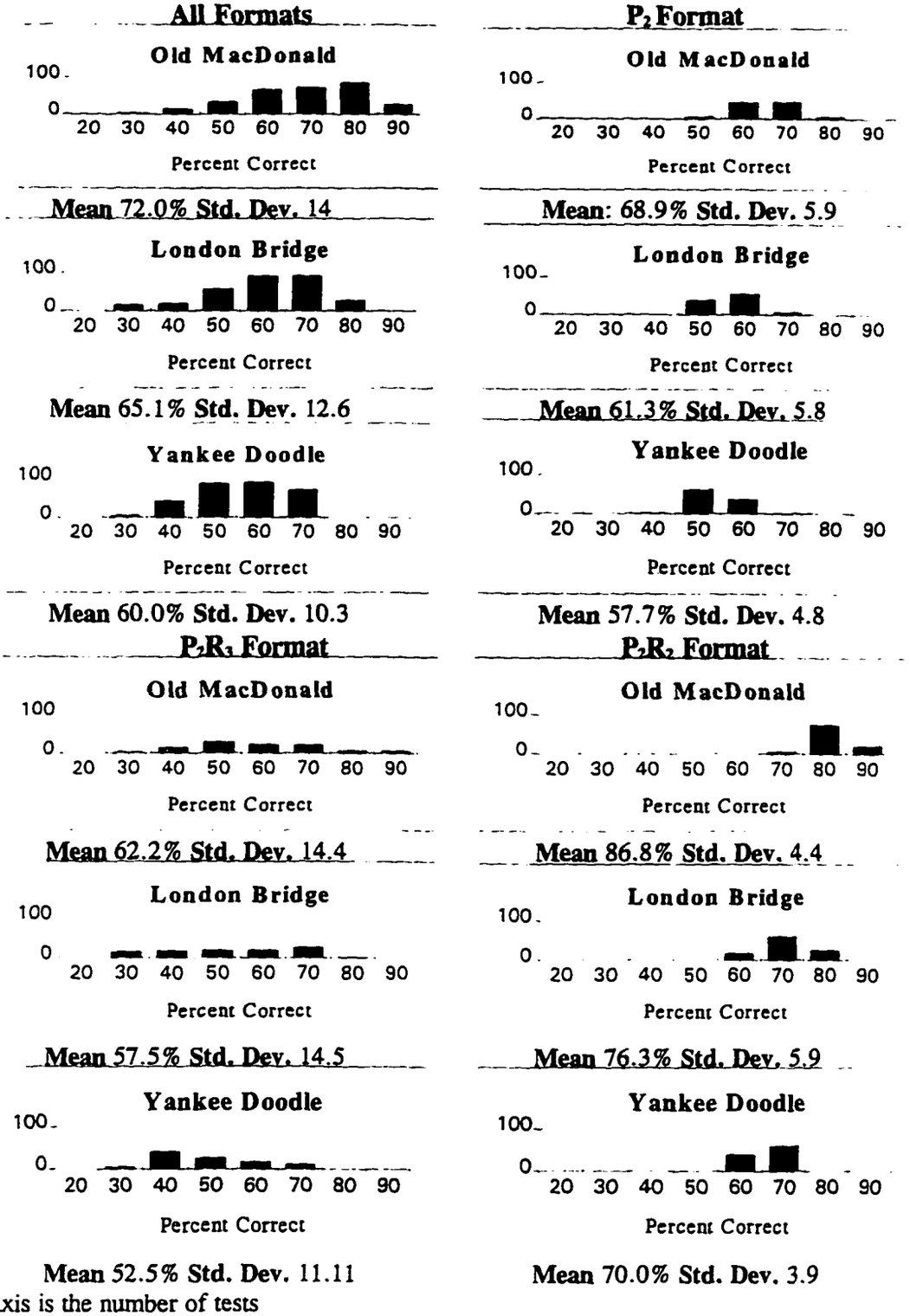
Figure 3: Evaluating the Model's Best Performance Results by Note Representation for All Classifier Formats



y-axis is the number of tests

Note representation doesn't appear to make much difference. Appendix F organizes the results by note representation for all classifier formats separated by melody. Fig. 3 collapses the data across melodies (based on Appendix F) and

Figure 4: Evaluating the Model's Best Performance Results by Melody Class



y-axis is the number of tests

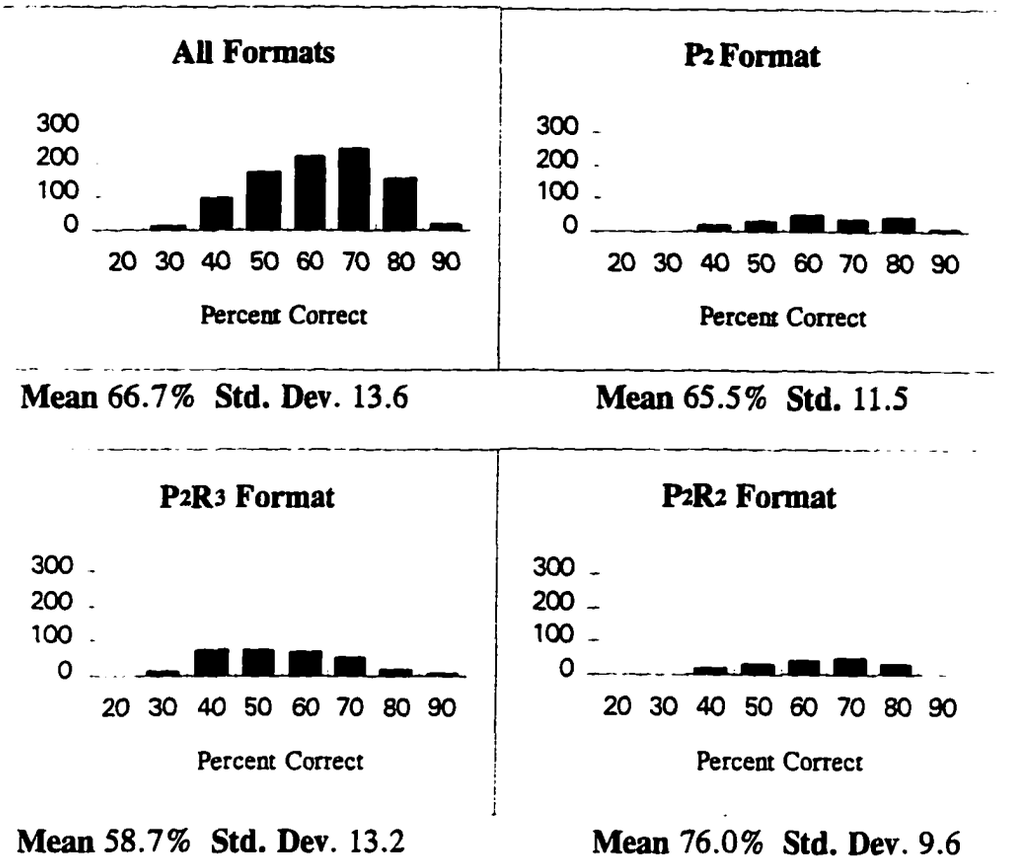
organizes the results by note representation for all classifier formats. All note representations have results ranging from 30% to 90% correct. Therefore, further analysis (Section 7.5) becomes necessary in order to confirm which is the superior note representation.

There appears to be a correspondence of melody class to performance. Class 1 (most predictable) does better than 2 or 3, as hypothesized. A summary of this phase's best performance by melody title for all classifier formats and each individual format is shown in Fig. 4. "Old MacDonald" is in Class 1, "London Bridge" is in Class 2, and "Yankee Doodle" is in Class 3. We see that Class 1 does better than Class 2 and Class 2 does better than Class 3. This supports our hypothesis that the melody class with the lower H value has less disorder and is more predictable. In Section 7.7, we will show that these results are statistically meaningful.

Applicability of Learning Results

Applicability of Learning results do support phase 1's results. Phase 2, the Applicability of Learning, tests whether the results from Evaluating the Model can be generalized to other melodies. The results from this phase are listed by note representation for each classifier format in Appendix G. Fig. 5 summarizes the results for all classifier formats, followed by the individual formats' results. It is not possible, by inspection of Fig. 5, to order the classifier formats by performance; however, further analysis (Section 7.11) shows that these results support phase 1.

Figure 5: Applicability of Learning's Best Performance Results by Classifier Format

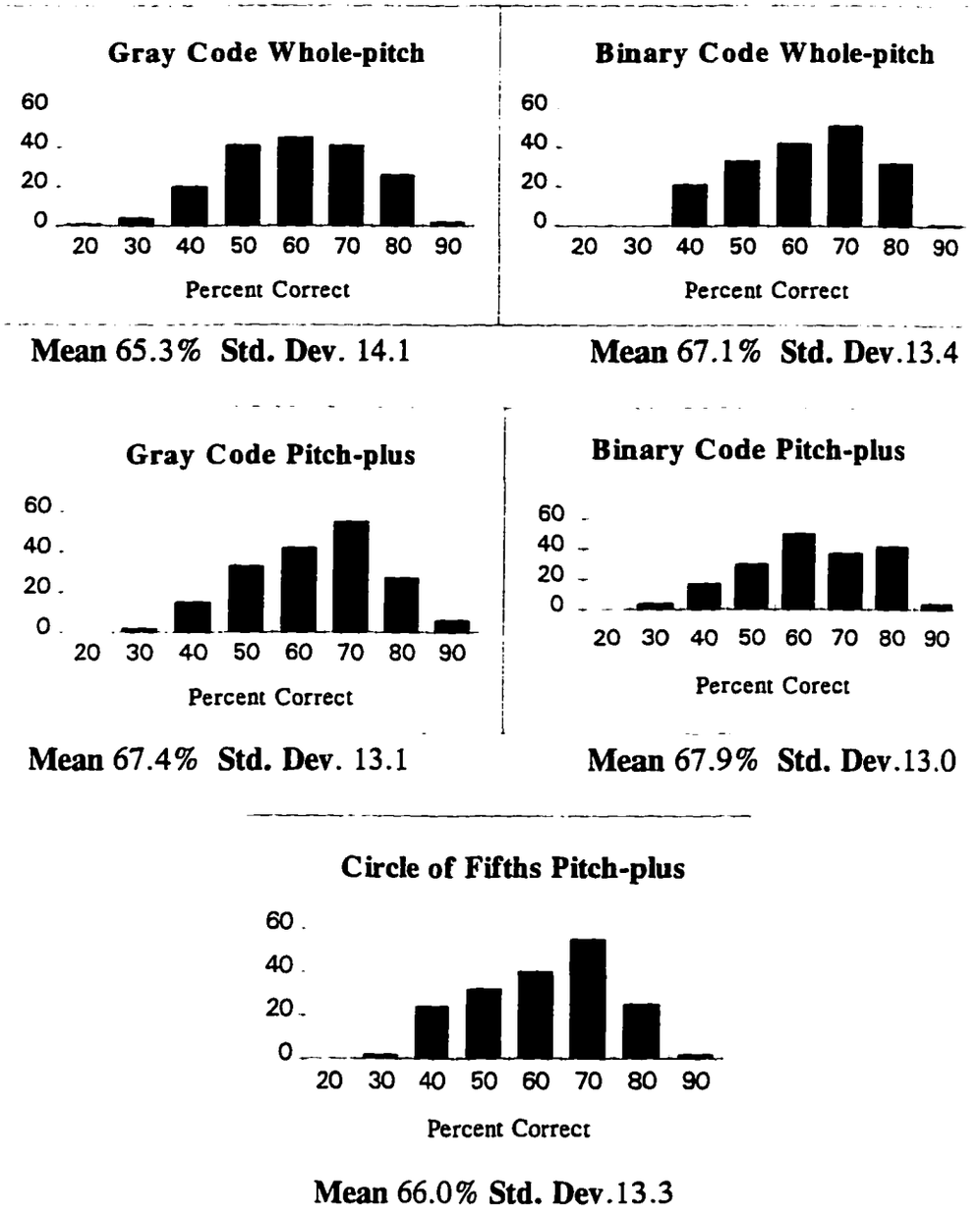


y-axis is the number of tests

Fig. 6 shows this phase's results organized by note representation for all classifier formats, collapsing the data in Appendix H across melody. As in the first phase, it is not possible to determine one preferred note representation.

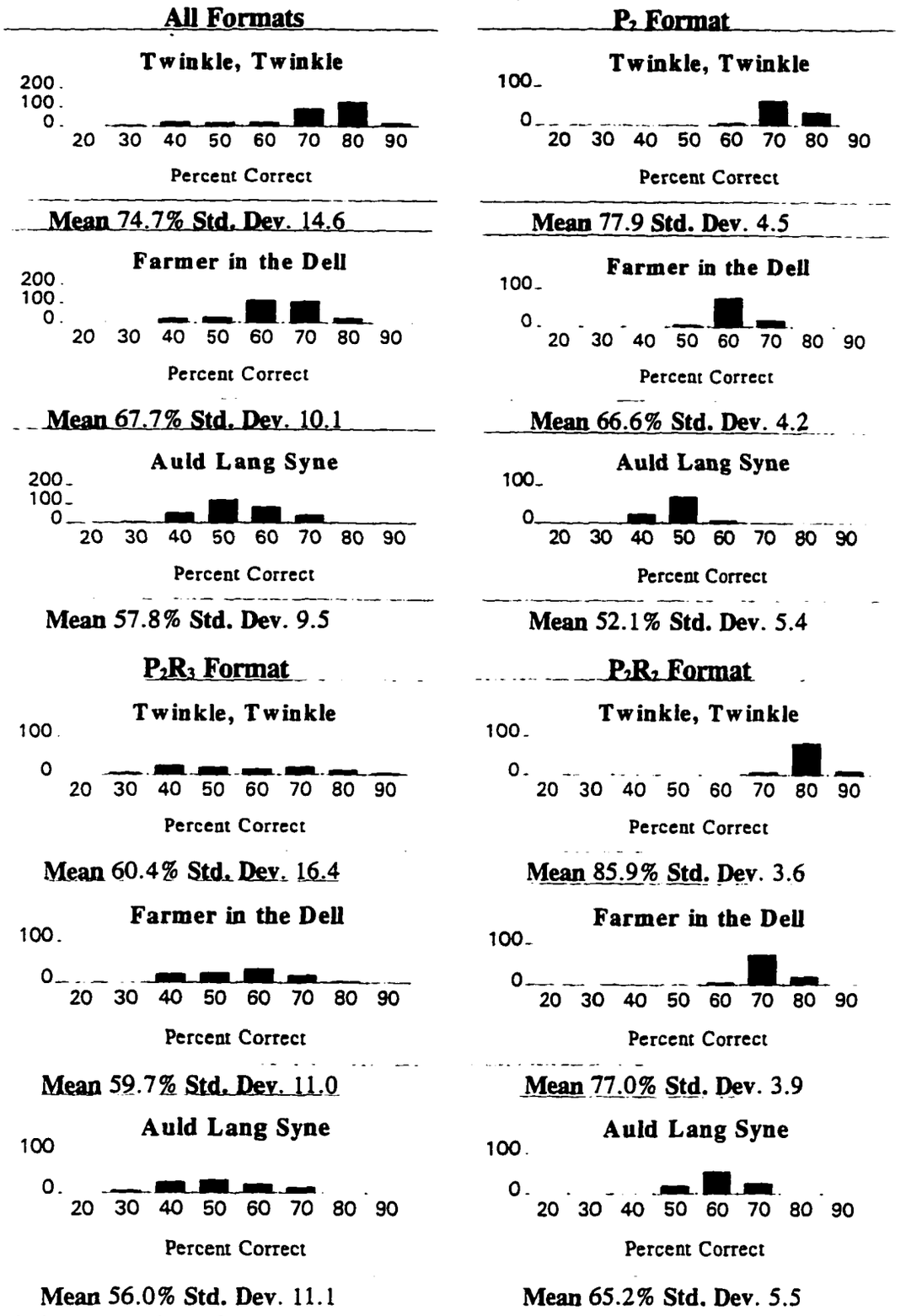
The Applicability of Learning results support the partitioning of the melodies. From Fig. 7, it appears that, whether considering all classifier formats or each individual format, Class 1 ("Twinkle, Twinkle Little Star") is superior to Class 2 ("Farmer in the Dell"), and that both outperform Class 3 ("Auld Lang Syne").

Figure 6: Applicability of Learning's Best Performance Results by Note Representation for All Classifier Formats



•y-axis is the number of tests

Figure 7: Applicability of Learning's Best Performance Results by Melody Class



y-axis is the number of tests

7.3 Minimum/Maximum Performers

To answer our first question, "Which runs are the minimum/maximum performers of this research?", we discuss the minimum and maximum performers from two perspectives: the original data and the trimmed data. The original data demonstrated a bias toward the formats that we used to select the parameters. Therefore, the data was trimmed to eliminate the highs and lows. Trimming the data eliminated the bias.

Minimum/Maximum Performers in Original Data

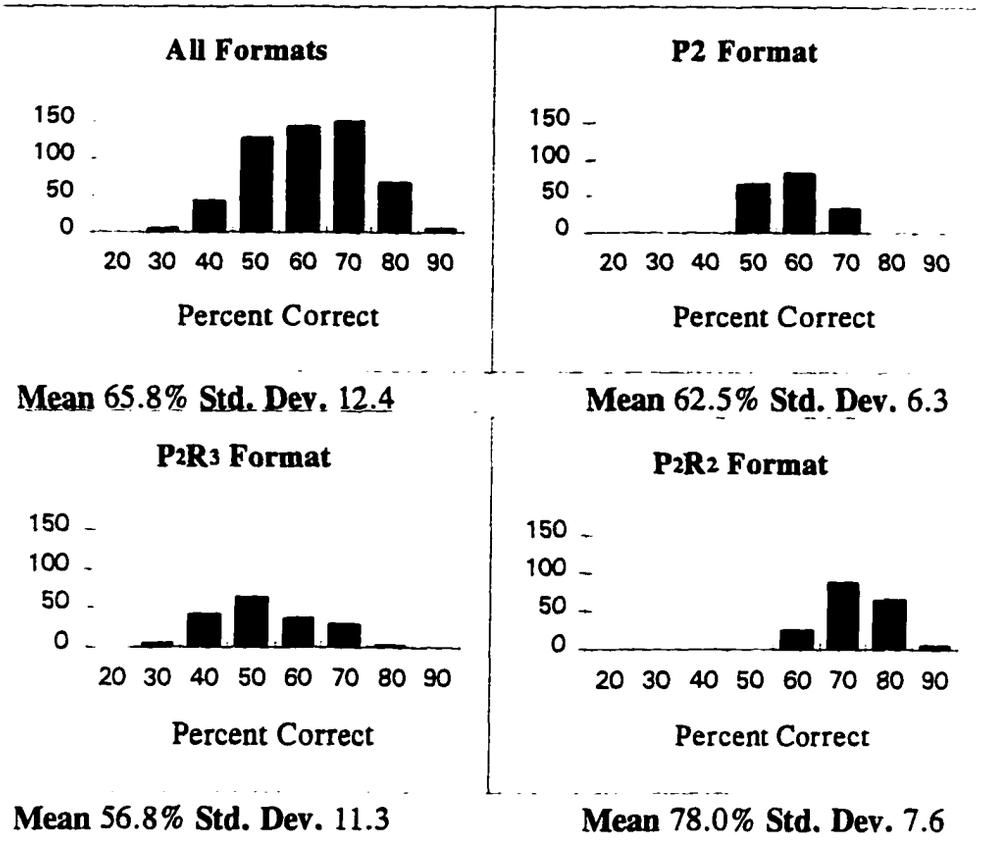
We can determine the minimum and maximum performers within the original data using the data shown in Appendix I. This appendix summarizes the results from both the Evaluating the Model and Applicability of Learning phases, showing average, minimum, and maximum best values for each melody, for each note representation and for each classifier format. There were 900 tests. The eight minimum results of all untrimmed tests range from 25% correct to 34% correct, where all eight were format P_2R_3. Five were Gray Code Whole-pitch and three were Binary Code Pitch-plus. The four maximum results, ranging from 95% to 96% correct (where the next highest was 92%) were also all in P_2R_3: two of these four used Gray Code Whole-pitch, and one each used Binary Code Pitch-plus and Circle of Fifths Pitch-plus.

The raw data showed that P_2R_3 as well as Gray Code Whole-pitch exhibited values at the extremes, indicating a bias. The bias occurs because the Gray Code

Whole-pitch P_2R_3 tests supplied the foundation used in establishing our parameters. To alleviate this partiality, we "trimmed" the results.

Figure 8A: Trimmed Values by Classifier Format

Evaluating the Model



Trimmed Results Description

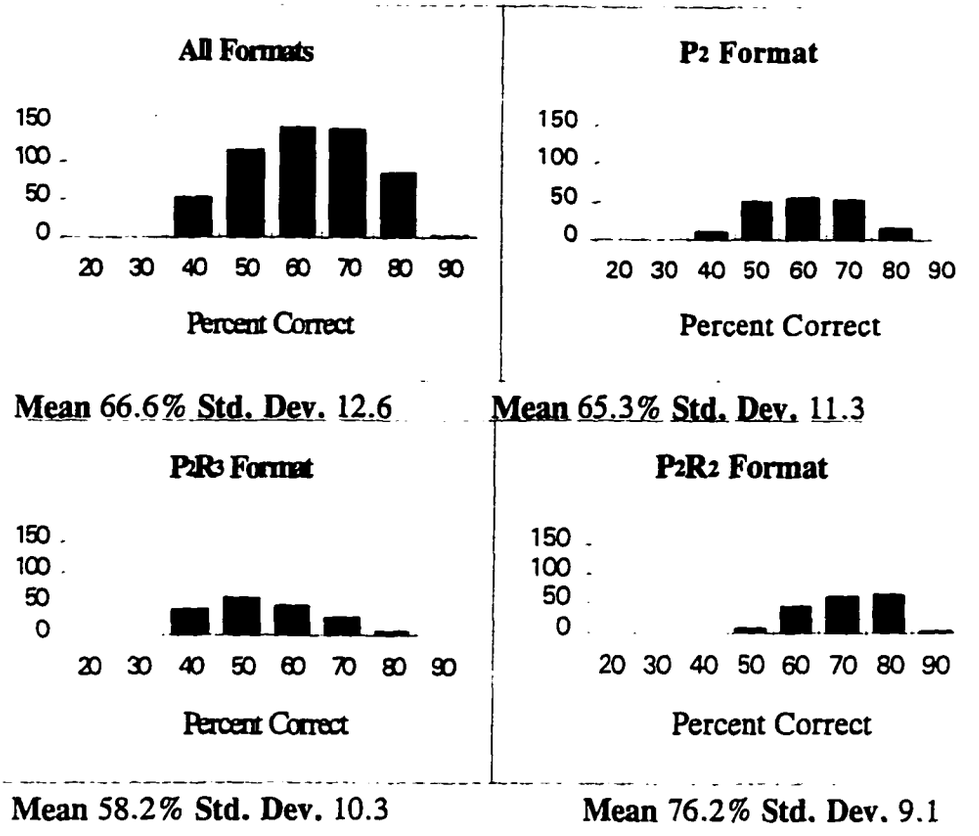
We trimmed the results by grouping the five random number runs for each specific population set, classifier format, note representation, and melody. The minimum and maximum values for each group were dropped and the middle three values retained. Appendix J lists the trimmed results by note representation for each population set, melody, and classifier format. Trimming the results reduces the number

of data values from 1,800 (each representing the best performance of one run from the original data set) to 1,080 data values, where the distribution is 216 data values for each of the five note representations, 360 data values for each of the three classifier formats, and 180 data values for each of the six melodies.

Fig. 8 details the trimmed values by classifier format for the two phases, Evaluating the Model (Fig. 8A) and Applicability of Learning (Fig. 8B). In both instances, the P_2R_2 format outperforms the others. To prove that this was statistically significant, we include classifier format in the ANOVA.

Figure 8B: Trimmed Values by Classifier Format

Applicability of Learning



* y axis is the number of tests

Minimum and Maximum Performers in the Trimmed Data

Using the trimmed results, we obtain seven minimum results, ranging from 37% to 39% correct. All are still from format P_2R_3; however, five are now Binary Code Pitch-plus and one is Gray Code Pitch-plus. Of the seven maximum results, ranging from 91% to 92% correct, six are now P_2R_2 and only one is P_2R_3. Of the seven, three are in Gray Code Pitch-plus, two in Binary Code Pitch-plus, one in Circle of Fifths Pitch-plus, and one in Gray Code Whole-pitch. The bias toward P_2R_3 and Gray Code Whole-pitch is no longer present.

7.4 Analysis of Variance Methodology

The analysis in Section 7.2 shows that the classifier format and melody class have an effect on performance and that we cannot determine the superior note representation without further analysis. Since an LCS is a stochastic system, a statistical analysis is necessary to evaluate its overall performance. For these reasons, we performed the various ANOVAs discussed in this section. The objective of our analysis was to evaluate the relative performance of classifier format, note representation and melody class rather than the absolute performance.

To use an ANOVA, the data must follow a normal distribution. Although the individual data values are not normally distributed (Fig. 8), the Central Limit Theorem asserts that distributions of means of samples tend to conform to the normal distribution, even if the distributions from which the samples were drawn are not normal. ANOVA analyzes the means of the data, which are normally distributed.

All the ANOVAs use the trimmed data set (Section 7.3) as input. The literature supports this idea of selecting a subset of results for an ANOVA (Schaffer, et al., 1989). The reason for comparing only the middle values is that the "F test" in an ANOVA is sensitive to differences in the whole range of values. Thus, we interest ourselves less with the endmost values and more with the mean results within a group.

The ANOVA will identify those effects found to be significant ($p \lessapprox 0.05$, where p is probability) while also leading to meaningful insights. Furthermore, according to established practices, when the F test has $p < 0.05$, one may reject the null hypothesis. For this investigation the null hypotheses are:

1. There is no difference between population sizes.
2. There is no difference between binary code and Gray code.
3. There is no difference between whole-pitch and pitch-plus.
4. There is no difference between classifier formats.
5. There is no difference between melody class.

These hypotheses correspond to inquiries 2, 3, 4, 5, and 6 respectively (Section 7.2). To evaluate the appropriate size of the classifier set, question 2, we included classifier format, population size, and melody class in an ANOVA which we call "ANOVA P" (discussed in Section 7.6). To examine issues 2, 3, 4, and 5, we perform two independent ANOVAs. Both include the values for Binary Code Whole-pitch, Binary Code Pitch-plus, Gray Code Whole-pitch; however, "ANOVA A" incorporates Gray Code Pitch-plus while "ANOVA B" incorporates Circle of Fifths Pitch-plus (Table 13). This division enables us to compare the effects of Gray Code Pitch-plus and Circle of Fifths Pitch-plus independently. To help us determine which

note representation is best, we subdivide this variable into two variables that represent its characteristics. The first characteristic is binary/Gray code and the second is whole-pitch/pitch-plus. The means of the trimmed data set are compared by the two ANOVAs (ANOVA A and ANOVA B) as follows:

1. Common to both:
   Classifier format: consisting of P_2, P_2R_3, and P_2R_2
   Melody Class: grouping the six melodies by the three classes

2. *ANOVA A*: Binary Code Whole-pitch, Binary Code Pitch-plus Gray Code Whole-pitch, and Gray Code Pitch-plus
   Binary Code/Gray Code: grouping the four note representations by binary code vs. Gray code
   Whole-pitch/Pitch-plus: grouping the four note representations by whole-pitch vs. pitch-plus

3. *ANOVA B*: Binary Code Whole-pitch, Binary Code Pitch-plus, Gray Code Whole-pitch, and Circle of Fifths Pitch-plus
   Binary Code/Gray Code: grouping the four note representations by binary code vs. Gray code
   Whole-pitch/Pitch-plus: grouping the four note representations by whole-pitch vs. pitch-plus

To explore whether the results are applicable from one phase to another, question 7, we perform four additional ANOVAs. ANOVA A (Table 15) and ANOVA B (Table 16) are each executed twice: once for each set of means from the trimmed data set from phase 1, Evaluating the Model (ANOVA A/C and B/C), and phase 2, Applicability of Learning (ANOVA A/D and B/D).

We define ANOVA terminology. The denominator of the F ratio, the within groups mean square, is an estimate of the average variability in the groups (i.e. classifier format, melody class, note representation, and population size). The values for the denominator are as follows: ANOVA P is 33.4, ANOVA A is 48.1, ANOVA B is 49.6, ANOVA A/C is 42.6, ANOVA A/D is 36.4, ANOVA B/C is 44.5, and ANOVA B/D is 37.1. We adopt the conventional format for reporting the results. For example, "effect" $F(2, 27) = 4.98$, $p < 0.05$; where "effect" is the variable under consideration, 2 equals the degrees of freedom for the numerator of the F ratio, 27 equals the degrees of freedom for the denominator of the F ratio, 4.98 is the value of the ratio, and $p < 0.05$ indicates that the probability, p, of obtaining such a large F ratio is less than 0.05 ($p < 0.05$ indicates a significant result).

7.5 Note Representations

To determine the overall best note representation, an additional ANOVA, ANOVA P, was performed for note representation on the trimmed data. The result showed no significant difference in the means: note representation $F(4,1075) = 1.12$, $p > 0.05$. Table 10A lists the trimmed means for each note representation; the means differ by less than 4%. Therefore, we cannot statistically claim that any one of our note representations is superior to any other. This result led us to investigate the ways of encoding the note representation (binary code and Gray code) and pitch information (whole-pitch and pitch-plus) and their effect on performance, discussed in the ensuing sections.

Table 10: Summary of Trimmed Means for Note Representations

A: Means by Overall Note Representation

Gray Code Whole-pitch	65.6%
Binary Code Pitch-plus	67.3%
Binary Code Whole-pitch	65.0%
Circle of Fifths Pitch-plus	66.1%
Gray Code Pitch-plus	66.8%

B: Means by Note Representation and Classifier Format

	Classifier Format		
	P_2R_3	P_2	P_2R_2
Gray Code Whole-pitch	55.4%	64.3%	77.1%
Binary Code Pitch-plus	59.4%	64.5%	77.9%
Binary Code Whole-pitch	56.8%	62.5%	75.8%
Circle of Fifths Pitch-plus	57.9%	63.9%	76.7%
Gray Code Pitch-plus	58.2%	64.5%	77.9%

C: Means by Note Representation and Melody Class

	Melody Class		
	1	2	3
Gray Code Whole-pitch	71.7%	66.8%	58.2%
Binary Code Pitch-plus	74.4%	67.2%	60.1%
Binary Code Whole-pitch	74.3%	63.6%	57.1%
Circle of Fifths Pitch-plus	73.4%	66.4%	58.6%
Gray Code Pitch-plus	73.7%	67.4%	59.4%

When considering classifier format and melody class, we see trends similar to those in the untrimmed data. Table 10B summarizes the means organized by classifier format. That format P_2R_2 is the leader, P_2 next, and P_2R_3 is the worst is clear across all

note representations. Also, Table 10C shows that the performance by melody class is maintained for all note representations. Class 1 does best, followed by Class 2 and lastly Class 3. We will discuss this in Section 7.7.

7.6 NEXTPITCH and Population Size

Question 2 asks whether the size of the classifier population has an effect on performance. To determine if a population of 400 classifiers was the optimal population size, additional runs were made (Section 6.5). We took classifier set 1 for all representations, classifier formats, and original melodies, and ran it for three different classifier set sizes (100, 200, and 400) for a total of 675 runs, which were then trimmed to 405. In Appendix K, the trimmed results of these runs are listed for set 1 organized by population size, note representation, classifier format, and melody.

The average (mean) trimmed performance and the standard error of the means are listed in Table 11 by population size separated by note representation and classifier format, collapsed across the melody classes. The data shows the following trends:

1. For P_2, as the population size increases, the performance decreases.
2. For P_2R_3, as the population size increases, so does the performance.
3. For P_2R_2, as the population of classifiers increases from 100 to 200, the performance either increases or levels off with one exception. In contrast, when the population increases from 200 to 400, a leveling off occurs except when considering Binary Code Pitch-plus.

Table 11: Average Trimmed Performance by Population Size, Note Representation, and Classifier Format for Classifier Set 1

	Population Size					
	100		200		400	
Format	Mean	Standard Error	Mean	Standard Error	Mean	Standard Error
Gray Code Whole-pitch						
P_2	75%	2.0	67%	2.0	62%	1.7
P_2R_3	35%	1.4	48%	2.0	52%	2.8
P_2R_2	76%	1.5	76%	2.6	77%	2.3
Binary Code Whole-pitch						
P_2	69%	3.5	62%	2.9	61%	2.9
P_2R_3	35%	1.7	44%	1.2	56%	2.0
P_2R_2	72%	1.9	76%	2.7	76%	2.7
Gray Code Pitch-plus						
P_2	72%	2.5	69%	2.4	61%	1.4
P_2R_3	38%	1.9	39%	2.8	56%	4.0
P_2R_2	74%	4.0	80%	2.4	79%	2.5
Binary Code pitch-plus						
P_2	73%	2.7	69%	2.5	66%	1.9
P_2R_3	38%	3.4	41%	1.8	52%	3.3
P_2R_2	78%	2.6	72%	3.3	78%	2.8
Circle of Fifths Pitch-plus						
P_2	73%	2.0	67%	1.7	64%	1.5
P_2R_3	34%	1.4	43%	1.4	52%	2.7
P_2R_2	67%	3.9	78%	1.7	78%	2.7

These results directly agree with Shu's (1992) findings that, for any population, there is a size limit where the test population can reach peak performance. In our sample, illustrated in Table 11, we see that P_2 format reaches its most effective performance at a population size of 100 and then begins to decrease. P_2R_2 begins to level off as population size increases and P_2R_3's performance increases (meaning the population size at which it reaches peak performance is 400 or above).

The optimal population size is clearly correlated to the length of the classifier. P_2 with a classifier length of 15 can be considered the easiest problem, P_2R_2 is next having a classifier length of 22, while P_2R_3, with a classifier length of 27, is the hardest problem. Goldberg (1996) explains that holding the number of iterations constant in an SCS means that, as the population size increases, each classifier is evaluated less. NEXTPITCH executes all three classifier formats for 300,000 iterations. The chances of a particular classifier being chosen by the GA decreases as the population size increases (i.e., 1/100 is more likely than 1/200). Therefore, in an easy problem such as P_2 , larger populations are "penalized." The system cannot zoom in on the more meaningful classifiers. In contrast, in the harder problem such as in P_2R_3 , the GA may require the larger population in order to create good building blocks (highly fit schemata). In the larger populations, there is more variability in the classifiers and the harder problem (i.e., the one which uses longer classifiers) needs this variability so that it has more combinations within which to match and find the more meaningful classifiers. The easy problems do not need this "extra" variability; it hampers these problems.

The results from these reduced runs (limited to classifier set 1) agree with the results from the full set of tests. Population size matters and is correlated to the length of the classifier. Classifier format matters. Even though P_2 does its overall best with a population of 100 classifiers, it is still outperformed by P_2R_2 at its worst. This is seen in Table 12, where our population findings for classifier format are collapsed across note representations. The results of ANOVA P on classifier format, population size, and melody class agree with our other findings (forthcoming sections). The values for ANOVA P are: classifier format $F(2, 404) = 1084.2$, $p < 0.05$, population size $F(2,404) = 17.49$, $p < 0.05$, and melody class $F(2,404) = 103.7$, $p < 0.05$. There is a difference in these areas, and these results hold up statistically: the hypothesis that population size does affect performance is confirmed. Our findings concur with the literature, that each problem has an appropriate population size.

Table 12: Summary of Average Population Size by Classifier Format for Classifier Set 1

	Population Size					
	100		200		400	
Format	Mean	Standard Error	Mean	Standard Error	Mean	Standard Error
P_2	72%	1.2	68%	1.0	62%	.9
P_2R_3	36%	.9	42%	.9	53%	1.3
P_2R_2	73%	1.4	76%	1.2	77%	1.1

7.7 Melody Class 1 / Melody Class 2 / Melody Class 3

From Section 7.2, it is obvious that melody class affects performance. The ANOVAs performed concur with these findings. The value for the F ratios are ANOVA A melody class $F(2, 863) = 328.5$, $p < 0.05$ and ANOVA B melody class $F(2,863) = 324.2$, $p < 0.05$. The means by melody class are shown in Table 13.

Table 13: Summary of Trimmed Means by Melody Class

	Melody Class		
	1	2	3
ANOVA A	73.5%	66.3%	58.7%
ANOVA B	73.5%	66.0%	58.5%

Table 14: Summary of Means by Melody Class and Classifier Format

A: ANOVA A Means by Melody Class and Classifier Format

	Melody Class		
	1	2	3
P_2	73.8%	63.3%	54.7%
P_2R_3	60.2%	58.4%	53.7%
P_2R_2	86.6%	77.1%	67.8%

B: ANOVA B Means by Melody Class and Classifier Format

	Melody Class		
	1	2	3
P_2	73.4%	63.7%	54.2%
P_2R_3	60.7%	57.6%	53.8%
P_2R_2	86.3%	76.8%	67.5%

The means for ANOVA A and ANOVA B are listed by classifier format and melody class in Tables 14A and 14B respectively. P_2R_2, Melody Class 1 is the best performer and P_2R_3, Melody Class 3 is the worst performer. P_2R_2 is the highest performing classifier format across melody class with P_2 and P_2R_3 following. It is also obvious that Melody Class 1 is the best overall in each classifier format, followed by Melody Classes 2 and 3.

The fact that Melody Class 1 is superior was predicted by its having the lowest *H* value for $P_{(i/j)}$ and $P_{(i/jk)}$; Melody Class 2 had the next higher *H* values for $P_{(i/j)}$ and $P_{(i/jk)}$, and Melody Class 3 has the highest *H* values for $P_{(i/j)}$ (Section 6.4). In other words the lower the *H* value, the less uncertainty, and the less information content per bit. As the *H* value increases, performance decreases, because the melody has become less predictable.

An LCS learns to expect the various pitches of a melody in proportion to their probability of occurrence. After training, the classifier population comes to be more representative of the notes that actually occur. Two factors may affect this probability, thereby affecting the *H* value. The first factor is based on the repetition of a sequence of notes. For example, consider the two-note sequence A, C followed by a next note D; the more often this sequence occurs in the melody, the lower the *H* value. The second factor is determined by the type of pitch transitions in the melody sequence. Again consider the above sequence (A, C, and next note D); the transitions are identified by two endpoints (AD). A unique transition exists when, given a note sequence (AC), there is only one possible next note (D), and a non-unique transition

exists when there is more than one possible next note. The more unique transitions there are in a melody, the lower the *H* value. The system will easily learn these unique transitions, whereas the classifiers that represent the non-unique transitions will receive conflicting reinforcement. The note sequences of the melodies in Melody Class 1 have more unique transitions than Class 2 melodies, and both have more than Class 3 melodies ("Old MacDonald" and "Twinkle, Twinkle, Little Star" in Class 1 each have one non-unique transition, "London Bridge" in Class 2 has three non-unique transitions, "Farmer in the Dell" in Class 2 and "Yankee Doodle" in Class 3 each have four non-unique transitions, and "Auld Lang Syne" in Class 3 has six non-unique transitions).

7.8 P_2 / P_2R_3 / P_2R_2 Classifier Formats

As we discussed in Section 7.2, the classifier formats have the most distinct influence on performance. Clearly, P_2R_2 format is the best, followed by P_2 format, and then P_2R_3, as seen in Table 15. The significance of F for both ANOVAs are $p < 0.05$ indicating that there is a difference between classifier formats. The value for the F ratios are ANOVA A classifier format $F(2, 863) = 604.3$, $p < 0.05$ and ANOVA B classifier format $F(2,863) = 573.9$, $p < 0.05$.

P_2R_2 is clearly the best classifier format we see in this experiment, while P_2R_3 is clearly the worst performer. We believe that the poor performance of P_2R_3 is due to its having the longest classifier. In section 7.6, we have shown a correlation between classifier length and the number of classifiers. That becomes relevant here. Results

indicate that format P_2R_3 needs a larger population of classifiers to perform adequately. In contrast, P_2's performance decreases overall as population size increases because it has too many classifiers. It suffers from premature convergence (to a local maximum), and it is not able to correct itself on any regular basis within the constraint of 300,000 iterations. This leaves P_2R_2 whose performance seems to be leveling off with the highest value at 400 classifiers.

Table 15: Summary of Trimmed Means by Classifier Format

	Classifier Format		
	P_2R_3	P_2	P_2R_2
ANOVA A	54.4%	63.9%	77.2%
ANOVA B	57.4%	63.8%	76.9%

The discussion of length is also studied by Robertson and Riolo (1988). They compare a two-bit representation and a five-bit representation, showing that learning is about twice as fast for the shorter representation, and that the longer representation does not achieve the same maximum as the shorter within the same number of iterations. This idea directly relates to our results. P_2R_3 is the longest classifier format, with 27 bits. For this format, there are 2^{27} or 7.62×10^{12} distinct schemata (Section 5.2); obviously this is a very large search space. The population size of 400 classifiers can not adequately represent the search space, with the result that there may be many nonmeaningful classifiers in any population. The GA has the difficult task, that is of finding the more fit classifiers within the time constraint of 300,000 iterations. Therefore, to permit the GA to introduce more variability through crossover and

mutation, more iterations are most likely necessary. It would be interesting to run P_2R_3 for more than 300,000 iterations to see if it would surpass P_2R_2's performance.

In our system, it is the relational information that defines a sequence of notes and makes a unique pattern (Section 7.7). Both P_2R_2 and P_2R_3 include relational information, whereas P_2 does not. The P_2 format is harder to learn when the note sequences are not unique because it lacks the relational information found in the other formats; that relational information is necessary to define a unique pattern.

The relational information included in two of our classifier formats defines a sequence of notes. This additional information can transform a non-unique transition into a unique transition. For example, consider one two-note sequence A, C, where the next note is D and another two-note sequence A, C, where the next note is B. Without the relational information between A and C (condition$_3$ of C_1, C_2, C_3 / A in the P_2R_2 format), the AC sequence is non-unique. However, if C and A are the same length in the ACD sequence and different lengths in the ACB sequence, then the representations differ (Section 5.4.2). This would affect P_2R_2's performance because it is easier to learn unique transitions; therefore, P_2R_2 would do better. Similarly, including the relational information in condition$_1$ and condition$_2$ of C_1, C_2 / A (P_2R_3 format) would affect P_2R_3's performance.

As seen previously, P_2R_2 outperforms P_2. On the other hand, it may be that, with additional classifiers, the P_2R_3 format could reap the benefit of the additional relational information found in this format and eventually do better. From another viewpoint, P_2R_2 contains the most relevant information of all three classifier formats. P_2 completely lacks its relational information, and P_2R_3 contains too much information.

P_2R_3 has information about the previous note that is not helpful in the current system because the present bucket brigade algorithm does not allow for rule chaining that would foster learning.

To address the issue of why the classifier format matters, we now look at the GA operator of crossover. This operator can have a major effect on the classifier format by either creating a more meaningful classifier or disrupting a useful classifier. Remember that the split point is randomly decided. For the purposes of illustration, we can restate the P_2R_3 format (C_1, C_2/A) as P_1R_1, P_2R_2/A and similarly, the P_2R_2 format (C_1, C_2, C_3/A) as P_1, P_2, R_2 /A, where each P_i is five bits and each R_i is seven bits, explaining the relationship P_{i-1}-P_i. Disruption occurs when the changes mean that the relational information (R_i) no longer represents the relationship between P_{i-1} and P_i or when it destroys a correct P_{i-1} to P_i sequence. P_2R_3 and P_2R_2 have 23 and 16 crossover points respectively. Disruption by crossover can occur at any point within the classifier. When the system starts out, the classifiers are mostly not fit, and crossover is necessary to introduce more variability, thus creating more fit classifiers; however, in the latter stage, when the classifiers are more fit, the crossover could degrade the system by substituting less meaningful classifiers.

In the latter stage, many classifiers may already represent the correct sequence within the music. In order for the system to learn successfully is has to keep the meaningful correspondences within a classifier together. As one example, assume that P_1 and P_2 describe a correct sequence and R_2 does not represent this P_1P_2 relationship. Then the only way to keep P_1 together with P_2 is if crossover occurs in R_2 of P_1R_1, P_2R_2/A (P_2R_3) or in R_2 of P_1, P_2, R_2/A (P_2R_2). This can occur 7/23 =.30 times in P_2R_3

and 7/16 = .44 in P_2R_2. Therefore, crossover has a better chance of creating more meaningful classifiers in P_2R_2 than in P_2R_3 when given a correct P_1P_2 sequence.

This shows that crossover influences P_2R_2's performance when the pitches are kept together. It would be worthwhile in the future to investigate a classifier format that incorporates this idea. This new format could be defined as C_1, C_2/A, where C_1 includes only pitch information and C_2 includes only relational information (P_1P_2, R_1R_2/A).

Another way the GA can affect performance is through the values of its parameters (i.e. percent of crossover, percent of mutation, and how often to apply the GA). In choosing parameters for NEXPITCH we had to weigh two things: the preference for setting optimal performance for each classifier format vs. at the same time keeping as much constant as possible from trial to trial. Therefore, we chose parameters that worked well in previous models. The P_2R_3 and P_2 formats might have a better performance if the GA parameters are optimized for each of these respective formats. The objective of fine tuning these parameters is to increase learning.

We conclude that classifier format does influence performance. The inclusion of relational information allows for more learning, but we can't determine whether the higher performance is due to the change in structure or the change in content or both.

7.9 Binary Code and Gray Code

Question 3 asks whether binary code or Gray code is superior. To determine the superior representation, we compare the categories of binary code to Gray code by

collapsing the note representations within ANOVA A and ANOVA B.

The F values for both ANOVAs were large, indicating that the means of the two groups, binary code and Gray code, are not statistically different. The value for the F ratios are: ANOVA A binary/Gray code $F(2, 863) = 0.02$, $p > 0.05$ and ANOVA B binary/Gray code $F(2,863) = 0.34$, $p > 0.05$. Referring to Table 16, the difference between the means within each ANOVA is less than .2%.

Table 16: Summary of Trimmed Means by Binary Code vs. Gray Code

	Binary Code	Gray Code
ANOVA A	66.1%	66.2%
ANOVA B	66.1%	65.9%

These findings agree with Caruana and Schaffer (1988) who state that Gray code is not statistically different from binary code although Gray code may sometimes actually be superior. In Section 3.3 we see that the existing literature prefers Gray code to binary code, but there is little experimental evidence to support this. In our research, Gray code performed slightly better for ANOVA A where the four note representations are directly comparable, yet for ANOVA B, the binary code performed marginally better.

For statistical analysis, our population is considered small; therefore, we conclude there is insufficient empirical evidence for us to state categorically whether Gray coding or binary is superior for this purpose.

7.10 Whole-pitch and Pitch-plus

One of the questions of the research was whether a psychologically motivated representation, pitch-plus (PP) would do better than a simple numeric representation, whole-pitch (WP) (Section 5.2). The means show that it does, although the F test results differ. (Recall that ANOVA A uses Gray Code Pitch-plus data and ANOVA B uses Circle of Fifths Pitch-plus data.) The means supported our hypothesis, but they are only statistically significant in ANOVA A and not in ANOVA B. In ANOVA A, WP/PP $F(2,863) = 13.7$, $p < 0.05$, showing that there is a difference between whole-pitch and pitch-plus. Contrarily, in ANOVA B, the F test WP/PP is $F(2, 863) = 8.45$, $p > 0.05$, which tells us that the means are not significantly different. It can be seen in Table 17 that, for both analyses, pitch-plus outperforms whole-pitch. A further breakdown of Table 18 shows that the means of Gray Code Pitch-plus (ANOVA A), Circle of Fifths Pitch-plus (ANOVA B), and Binary Code Pitch-plus (both ANOVAs) confirms pitch-plus' outperformance of whole-pitch.

Table 17: Summary of Trimmed Means by Whole-pitch vs. Pitch-plus

	Whole-pitch	Pitch-plus
ANOVA A	65.3%	67.0%
ANOVA B	65.3%	66.7%

These statistics show that our more psychologically motivated representations pitch-plus, outperform the non-psychologically motivated representations, whole-pitch. This supports the literature which states that representations which are correlated to the domain knowledge perform better (Caruana and Schaffer, 1988).

Table 18: Summary of Trimmed Means by Gray Code / Binary Code vs. Whole-pitch / Pitch-plus

		ANOVA A	ANOVA B
	Whole-pitch	**Pitch-plus**	**Pitch-plus**
Gray Code	65.6%	66.8%	66.1%
Binary Code	65.0%	67.3%	67.3%

Why does pitch-plus outperform whole-pitch? It is our contention that pitch-plus outperforms whole-pitch because the whole-pitch representations are harder to learn than the pitch-plus representation. Recall that whole-pitch represents a note by five bits, incorporating all information, while pitch-plus separates three bits of note, one bit octave, and one bit accidental. The purpose of the GA is to apply its operators with the goal of yielding more meaningful classifiers. The crossover operator breaks up classifiers and reforms them. When this operator is applied to pitch-plus representations, there is less chance of disruption because the three bits for note name will more likely stay together during crossover and keep a meaningful classifier. In contrast, in the five bits used by whole-pitch, there is a higher chance of disruption of the note representation.

In addition, the mutation operator changes a bit in a classifier at random. When it is applied to the pitch-plus representation, it has a higher likelihood of yielding a more fit classifier than when applied to the whole-pitch representation. If the three bits for note name are correct, a mutation on either bit 4 (accidental) or bit 5 (octave) may yield a better classifier, or (if the two bits are correct) then a mutation on bits 1-3 may lead to a better classifier. However, in whole-pitch, a mutation on a bit may create a classifier that is completely different, because the five bits represent a single value. If

the mutation is on either one of the high order bits (bits 1-3) or one of the low order bits (bits 4-5), then there is a major change in the note representation. For example, a mutation on bit 2 of pitch representation "00001" leads to "01001". In Binary Code Whole-pitch, this would be a change from A in the second octave to F in the third octave producing an octave change and an unrelated note, therefore a less meaningful classifier. In contrast, using the previous example, in Binary Code Pitch-plus this would be a change from C in the fourth octave to E in the fourth octave, producing no octave change, therefore a more relevant classifier. As a second example, a mutation on bit 4 of "10001" leads to "10011". In Binary Code Whole-pitch, this would be a change from $C^{\#}$ in the fourth octave to $D^{\#}$ in the fourth octave, a completely different note; however, in Binary Code Pitch-plus this would be a change from G in the fourth octave to $G^{\#}$ in the fourth octave, only a change in accidental, therefore producing a more meaningful classifier.

7.11 Correlation of Results to Other Melodies

Evaluating the Model (phase 1) used one set of melodies; Applicability of Learning (phase 2) ran the same tests using a second set of melodies. To answer question 7, "Are our results from Evaluating the Model phase statistically similar to the results from the Applicability phase?" four more ANOVAs were run (Section 7.4). The results in phase 1 were applicable to phase 2 for classifier format and melody class. A summary of the ANOVAs, allowing us to compare the results, appears in Tables 19 and 20. We briefly discuss the results using the following four areas: P_2 vs.

P_2R_3 vs. P_2R_2, Melody Class 1 vs. Melody Class 2 vs. Melody Class 3, Binary Code vs. Gray Code, and Whole-pitch vs. Pitch-plus.

The classifier format statistics are comparable when going from one set of melodies to another set. The F test results for all four ANOVAs for classifier format were consistent with the previous results (Section 7.8). These tests therefore confirm that there is a difference between classifier formats. From Tables 19A and 20A, we clearly see that P_2R_2 is the best performer in both cases, followed by P_2 and then P_2R_3.

Phase 2 results, like phase 1, show Melody Class 1 the leader and Melody Classes 2 and 3 the lesser performers. The F tests show the same $p < 0.05$ for all corresponding ANOVAs. The data illustrated in Tables 19B and 20B support this, confirming that melody class is a significant factor in learning.

Tables 21 and 22 summarize for ANOVA A and ANOVA B the means by melody class and classifier format for Evaluating the Model (Tables 21A and 22A) and Applicability of Learning phases (Tables 21B and 22B), which are ANOVA C and ANOVA D respectively. All means correlate between ANOVA C and ANOVA D. P_2R_2 is the maximum performer, and P_2 outperforms P_2R_3 except when considering Melody Class 3 in Applicability of Learning (Tables 21B and 22B, ANOVA D).

The melody that creates the exception, "Auld Lang Syne" (Table 9, Section 6.4) has the highest H value of all our melodies. This melody has the most non-unique transitions (Section 7.7) and needs more relational information to resolve conflicts. P_2 does not include the relational properties, whereas P_2R_3 includes the relational properties, allowing this melody to perform better in this format.

Table 19: Summary of ANOVA A/C and ANOVA A/D

A: Means by Classifier Format

	Classifier Format		
	P_2R_3	P_2	P_2R_2
ANOVA A/C Evaluating the Model	56.6%	62.1%	78.0%
$F(2, 431) = 416.0, p < 0.05$			
ANOVA A/D Applicability of Learning	58.3%	65.7%	76.4%
$F(2, 431) = 324.6, p < 0.05$			

B: Means by Melody Class

	Melody Class		
	1	2	3
ANOVA A/C Evaluating the Model	72.0%	64.9%	59.8%
$F(2, 431) = 126.3, p < 0.05$			
ANOVA A/D Applicability of Learning	75.1%	67.6%	57.6%
$F(2, 431) = 302.0, p < 0.05$			

C: Means by Whole-pitch vs. Pitch-plus

	Whole-pitch	Pitch-plus
ANOVA A/C Evaluating the Model	64.4%	66.7%
$F(1, 431) = 12.7, p < 0.05$		
ANOVA A/D Applicability of Learning	66.2%	67.4%
$F(1, 431) = 4.6, p > 0.05$		

D: Means by Gray Code / Binary Code vs. Testing Procedure

	Evaluating the Model		Applicability of Learning	
	Whole-pitch	Pitch-plus	Whole-pitch	Pitch-plus
Gray Code	65.9%	66.4%	65.3%	67.3%
Binary Code	63.0%	67.0%	67.1%	67.5%

E: Means by Binary Code vs. Gray Code

	Binary Code	Gray Code
ANOVA A/C Evaluating the Model	65.0%	66.1%
$F(1, 431) = 3.4, p > 0.05$		
ANOVA A/D Applicability of Learning	67.3%	66.3%
$F(1, 431) = 3.1, p > 0.05$		

Table 20: Summary of ANOVA B/C and ANOVA B/D

A: Means by Classifier Format

	Classifier Format		
	P_2R_3	P_2	P_2R_2
ANOVA B/C Evaluating the Model	56.4%	62.5%	77.9%
$F(2, 431) = 395.1, p < 0.05$			
ANOVA B/D Applicability of Learning	58.3%	65.0%	75.9%
$F(2, 431) = 304.2, p < 0.05$			

B: Means by Melody Class

	Melody Class		
	1	2	3
ANOVA B/C Evaluating the Model	72.2%	64.6%	60.0%
$F(2, 431) = 124.0, p < 0.05$			
ANOVA B/D Applicability of Learning	74.7%	67.4%	57.1%
$F(2, 431) = 308.8, p < 0.05$			

C: Means by Whole-pitch vs. Pitch-plus

	Whole-pitch	Pitch-plus
ANOVA B/C Evaluating the Model	64.4%	66.8%
$F(1, 431) = 13.6, p < 0.05$		
ANOVA B/D Applicability of Learning	66.2%	66.6%
$F(1, 431) = 0.05, p > 0.05$		

D: Means by Gray Code / Binary Code vs. Testing Procedure

	Evaluating the Model		Applicability of Learning	
	Whole-pitch	Pitch-plus	Whole-pitch	Pitch-plus
Gray Code	65.9%	66.8%	65.3%	65.6%
Binary Code	63.0%	67.0%	67.1%	67.5%

E: Means by Binary Code vs. Gray Code

	Binary Code	Gray Code
ANOVA B/C Evaluating the Model	65.0%	66.3%
$F(1, 431) = 4.0, p > 0.05$		
ANOVA B/D Applicability of Learning	67.3%	65.5%
$F(1, 431) = 10.0, p > 0.5$		

Table 21: Summary of Means by Melody Class and Classifier Format for ANOVA A

A: ANOVA A/C Evaluating the Model

	Melody Class		
	1	2	3
P_2	68.9%	60.2%	57.2%
P_2R_3	60.0%	57.9%	51.8%
P_2R_2	87.0%	76.6%	70.4%

B: ANOVA A/D Applicability of Learning

	Melody Class		
	1	2	3
P_2	78.7%	66.4%	52.1%
P_2R_3	60.3%	59.2%	55.6%
P_2R_2	86.3%	77.6%	65.2%

Table 22: Summary of Means by Melody Class and Classifier Format for ANOVA B

A: ANOVA B/C Evaluating the Model

	Melody Class		
	1	2	3
P_2	69.2%	60.9%	57.5%
P_2R_3	60.6%	56.6%	52.0%
P_2R_2	86.9%	76.4%	70.4%

B: ANOVA B/D Applicability of Learning

	Melody Class		
	1	2	3
P_2	77.6%	66.5%	50.9%
P_2R_3	60.7%	58.5%	55.6%
P_2R_2	85.7%	77.2%	64.6%

Hence, we conclude that our melody class results are applicable to other melodies.

One result from phase 1 was not confirmed by phase 2. Phase 1 showed pitch-plus outperforms whole-pitch. From Table 19C, the F test for ANOVA A/C is WP/WW $F(1,431) = 12.7$, $p < 0.05$ and from Table 20C, the F test for ANOVA B/C is WP/WW $F(1,431) = 13.6$, $p < 0.05$. However, though phase 2 means showed the same trend, the results were not statistically significant. From Table 19C the F test for ANOVA A/D is WP/WW $F(1,431) = 4.6$, $p > 0.05$, and from Table 20C the F test for ANOVA B/D is WP/WW $F(1,431) = 0.05$, $p > 0.05$. Similar values are seen in Tables 19D and 20D where the means show that pitch-plus outperforms whole-pitch.

When considering binary code and Gray code, the results are not statistically significant in either phase 1 and phase 2. Tables 19E and 20E show the corresponding results, confirming overall that the difference between binary code and Gray code was not statistically significant.

7.12 Analysis of Classifier Content

At this point in our analysis we focus on an additional question: "What do the classifiers look like?" We would like to see what representations NEXTPITCH uses to learn with: do classifiers represent more general information (with don't cares "#") or do the classifiers reflect pattern matching (each position matches an exact encoded value). One test was selected from trimmed performers with the best performance of 92%. That test was rerun and the population of classifiers was saved. The selected

test used P_2R_2, Gray Code Pitch-plus, "Old MacDonald", classifier set 1 and random number 4. The seven classifiers that had the maximum strength when the run achieved 92% appear in Table 23 and are numbered 1 through 7. We discuss the classifiers from three perspectives. The first two analyze the classifier with respect to the P_1 and P_2 positions. The third analyzes the R_2 positions.

Table 23: The Contents of the Best Classifiers

Number	P1	P2	R2	Action
1	#0# 11	1#1 10	#0 ## ###	110 10
2	#00 11	1## 10	#0 ## ###	110 10
3	0## ##	#10 01	00 01 101	011 00
4	0## 1#	#10 01	00 01 101	011 01
5	0## 11	111 #0	10 #1 1#1	011 00
6	#10 1#	001 #1	11 0# #01	100 10
7	#0# #1	110 10	#1 ## #0#	100 11

* in P_1 and P_2 bits 1-3 are note name, bit 4 is accidental, bit 5 is octave

** in R_2 bits 1-2 are contour where "00" means no change; "01" is up; "10" is down; and "11" means no prior information, bits 3-4 are duration where "00" is same length; "01" means current note is longer; "10" current note is shorter; and "11" means no prior information, bits 5-7 are interval, where "000" is a major 2nd, "001" is a unison, "011" is a major 3rd, "010" is a perfect 5th, "110" is a perfect 4th, "111" is a minor 3rd, "101' is a major 6th, and "100" is a minor 2nd.

The first group of classifiers we consider consists of classifiers 1, 2, 3, and 7. If any "#" in P_1 or P_2 is replaced by one of its possible values (i.e. "0" or "1") then

these classifiers do represent two-note transitions in the melody. Then some substitutions produce a sequence of note names that are correct for this piece of music but the accidental or octave may not be correct. For example, using classifier 1 (Table 23), and replacing the first three "#" with "0", "0", and "1" respectively, produces the two-note transition of $C^{\#}$ in the fourth octave followed by G in the third octave. The correct sequence in the melody is C in the fourth octave followed by G in the third octave. Therefore, only the accidental bit in P_1 is incorrect. This type of replacement leads to similar results in classifiers 2, 3 and 7.

The next set of classifiers numbered 4, 5, and 6 cannot with any replacement represent a two-note sequence. For example, using classifier 6, the P_2 values "001#1" could become "00101" or D in the fourth octave, a valid P_2 in this melody. The same reasoning can be applied to the P_1 bits. Part of this type of classifier already represents a note in the melody; thus the GA has less to learn.

We now address the third perspective, the R_2 positions. Some values produced by replacing the "#" in R_2 represent actual relational information that is meaningful in the melody. For example, using classifier 7, replacing the interval bits in R2 (bits 5-7: "#0#") can yield "000" a major 2^{nd}, "001" a unison, or "101" a major 6^{th}, all of which are valid in this melody. As a second example, classifier 1 with all "#" in the interval positions can represent any interval. These classifier represent more general information. They represent relations; rather than the exact note sequence. That is why they are successful, because these intervals occur frequently in the melody.

Some classifiers represent two-note sequences while others represent relations between notes. Further exploration and longer runs are necessary to determine whether

pattern matching or relational information is ultimately more successful.

7.13 Summary

Our findings support the view that there is an ideal population size that is determined by the classifier length. Shorter classifiers need smaller populations; larger classifiers need larger populations.

Classifier format does affect performance. P_2R_2 format outperforms P_2 and P_2R_3. The length of a classifier influences the results. We also showed that including relational information is more advantageous than not including it.

Melody class based on H values definitely influences performance in all classes. Melody Class 1 outperforms Melody Classes 2 and 3.

We found that psychologically motivated representations do better. It appears pitch-plus representation outperforms whole-pitch although this was not statistically significant in all phases of testing. This was confirmed in phase 1 and not in phase 2 and further research will be needed. We agree with the literature which states that representations that are more descriptive of the domain are preferred over those that are not.

We did not find any one note representation to be statistically the best. Also, we confirm that neither binary code or Gray code is superior.

For the most part, there is a correlation between the results from the Evaluating the Model phase to the Applicability of Learning phase, thus showing that our results can be transferred and apply to other melodies.

Chapter 8

Conclusions

8.1 Significance

As a general inductive learning model, the learning classifier system using genetic algorithms emphasizes efficient low-level learning where no previous domain knowledge is necessary. The LCS is a viable model that can be applied to music learning. The system learns by the internal generation of new rules that, in turn, compete with current rules. New knowledge is gained by building on the system's current knowledge base.

This dissertation studies a variety of topics in the field of LCS: the number of classifiers to use, LCS classifier format, and the representation of music. We are interested in the "best performance" results of all our tests, and employ this average in all our analyses. As a supplemental focus, we utilize *information theory*, enabling us to evaluate nursery melodies so that we may partition the melodies into classes.

Our model, NEXTPITCH, is a simple classifier system without all the "bells and whistles" of classifier systems currently in use. We specifically chose to use a simple system so that we can identify and report the results of our ideas.

The appropriate number of classifiers, or population size, depends on the problem being examined. Our findings concur with the literature stating that each problem has an optimal population size after which the performance decreases. A major contributing factor is the length of the classifiers. The length of a classifier affects the disruption caused by crossover.

As to the change in LCS structure, we find that including more information within the classifier (by determining the relational properties between two notes) improves performance when contrasted with omitting it.

This research demonstrates that information theory (*H* value) can be used as a way to predict the performance of an LCS system. The lower the *H* value, the less information conveyed and more predictability exists; therefore, the better the performance.

When the issue is representation, the musically oriented/psychologically motivated representations are often superior to the other representations discussed in this thesis, so other researchers should use representations which do this.

We also find that there is no dominance between binary and Gray code. Therefore, researchers can use whichever is most convenient.

The representation chosen for a learning system plays a role in the performance of that system. Our foremost contribution to artificial intelligence is how to represent the many characteristics of music. This thesis shows that researchers intending to represent music should use a representation which separates pitch from accidental and octave and at the same time include the

relational information between notes.

8.2 Future Research

On a concluding note, there are areas of study that warrant further attention. We limit many variables. Among these are number of iterations, random number seeds, population size, the melodies, AOC parameters, frequency with which the GA is applied, and mutation and crossover probabilities. These variables can be modified in many ways in further studies. This paper reflects initial findings; however, it would be enlightening to see how these results compare with results from a study using a larger search space, or to perform a fully factorial design on all the variables.

One specific issue needing further investigation is the population size. Additional tests on larger size populations will allow us to determine P_2R_3 format's maximum performance and increase our knowledge of the relation between performance and classifier length.

Another important area is the GA parameters. Our research keeps them constant throughout our study; it would be interesting to see how, if at all, their variation would influence results.

We use three melody classes with two melodies for each class. Further study using melodies with different *H* values--reflecting various complexities within music--would therefore define new melody classes. This will no doubt serve to

prove the maintainability of our results across note representation and classifier format.

In another vein, we need to evaluate further the effect of changing the LCS structure on performance. Keeping the classifier length constant and choosing representations that change the way the classifier bits are divided up (i.e. ordering the representational information differently or including other relational information or note characteristics) would help us to investigate the relationship between structure and LCS performance.

As a final thought, the SCS in this thesis considers representations using binary alphabets. A modification to NEXTPITCH enabling it to manipulate classifiers using a different base alphabet might yield a more sophisticated LCS and would indeed prove informative.

Appendix A

Sample Input to NEXTPITCH

"Farmer in the Dell": note name, accidental, octave, duration, contour, interval

g 3##0, c 4+ +4, c 4-00, c 4+00, c 4-00, c 4+00, d 4-+2, e 4+ +2,

e 4-00, e 4+00, e4-00, e 4+00, g 4-+4, g 4-00, a 4-+2, g 4+-2,

e 4--4, c 4+-3, d 4-+2, e 4+ +2, e 400, d 4+-2, d 4-00, c 4+-2

Description of input encoding:

1. **Note name** is the pitch name consisting of "a, b, c, d, e, f, g" or "r" for a rest.
2. **Accidental** is either a " " for a natural or a "1" for a sharp (all flats are interpreted as the corresponding sharp).
3. **Octave** where the third octave is represented as a "0" and the fourth octave is a "1".
4. **Duration** where "0" means they are the same length; "+" means the current note is longer; "-" means the current note is shorter; and "#" means there is no prior note information.
5. **Contour** where "0" means no change; "+" means up; "-" means down; and "#" means that there was no prior note information.
6. **Interval** where a major 2nd is "2", unison is "0", major 3rd is "3", perfect 5th is "5", perfect 4th is "4", minor 3rd is "7", major 6th is "6", and a minor 2nd is "8".

Appendix B

Sample Output of NEXTPITCH

Sample Output Report where the columns represent: iteration count (the number of notes processed), percent of predictions correct from the start, and the percent of correct predictions in the last 3 times through the entire melody respectively:

Iteration Count	% Correct From Start	% Correct Last 3 Times
90	0.0444	0.0444
180	0.0500	0.0556
270	0.0519	0.0556
360	0.0583	0.0778
450	0.0578	0.0556
540	0.0611	0.0778
630	0.0603	0.0556
720	0.0653	0.1000
810	0.0716	0.1222
900	0.0767	0.1222
990	0.0808	0.1222

Appendix C: Parameter Selection Results

Gray Code Whole-pitch, P_2R_3

C.1 Detailed Results London Bridge

Population Size 100

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	33	30	33	41	38	35.0
1	29	44	38	34	45	38.0
2	33	25	30	50	33	34.2
3	33	51	33	29	33	35.8
4	43	33	43	34	43	39.2
5	33	37	34	33	33	34.0
6	43	29	43	47	38	40.0
7	34	33	37	33	38	35.0
8	33	51	38	33	33	37.6
9	38	43	33	34	43	38.2
					Overall	36.7

Population Size 200

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	34	47	33	65	33	42.4
1	55	79	61	34	43	54.4
2	75	34	33	43	33	43.6
3	75	33	33	38	47	45.2
4	33	33	33	38	41	35.6
5	38	61	43	43	37	44.4
6	61	61	65	51	34	54.4
7	34	43	51	40	23	38.2
8	51	61	69	43	47	54.2
9	43	61	33	33	47	43.4
					Overall	45.5

Gray Code Whole-pitch, P_2R_3

London Bridge

Population Size 400

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	51	69	48	69	55	58.4
1	65	36	75	51	51	55.6
2	34	61	56	65	65	56.2
3	43	56	38	61	65	52.6
4	56	45	65	44	70	56.0
5	73	79	75	69	77	74.6
6	45	83	77	66	52	64.6
7	52	79	59	79	75	68.8
8	47	48	65	56	43	51.8
9	56	65	65	65	65	63.2
					Overall	60.1

C.2 Detailed Results Old MacDonald

Population Size 100

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	39	67	39	53	38	47.2
1	51	34	30	26	30	34.2
2	26	35	26	34	28	29.8
3	25	39	42	47	26	35.8
4	26	44	42	47	43	40.4
5	21	26	34	30	26	27.4
6	80	43	34	42	30	45.8
7	43	26	30	29	39	33.4
8	41	34	42	34	30	36.2
9	26	23	48	46	30	34.6
					Overall	36.4

Gray Code Whole-pitch, P_2R_3

Old MacDonald

Population Size 200

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	47	38	51	38	69	48.6
1	47	51	46	47	33	44.8
2	38	51	38	56	46	45.8
3	47	47	60	56	57	53.4
4	60	55	47	42	47	50.2
5	61	33	57	42	64	51.4
6	43	38	60	42	33	43.2
7	29	34	30	30	34	31.4
8	48	39	42	61	42	46.4
9	44	42	43	30	51	42.0
					Overall	45.7

Population Size 400

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	39	43	41	67	58	49.6
1	51	51	73	44	51	54.0
2	58	73	42	96	58	65.4
3	47	42	51	67	76	56.6
4	71	64	42	92	84	70.6
5	56	75	47	47	67	58.4
6	56	64	43	34	60	51.4
7	60	64	51	42	39	51.2
8	71	47	79	62	79	67.6
9	80	48	58	56	44	57.2
					Overall	58.2

Gray Code Whole-pitch, P_2R_3

C.3 Detailed Results Yankee Doodle

Population Size 100

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	41	37	37	37	37	37.8
1	18	37	37	41	37	34.0
2	23	30	37	36	37	32.6
3	41	47	47	41	44	44.0
4	30	37	47	37	34	37.0
5	37	41	37	37	37	37.8
6	37	40	38	48	30	38.6
7	55	37	37	38	37	40.8
8	37	31	38	42	30	35.6
9	55	51	39	44	44	46.6
					Overall	38.4

Population Size 200

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	30	37	41	34	21	32.6
1	52	44	41	48	41	45.2
2	47	44	44	55	62	50.4
3	51	47	44	76	51	53.8
4	34	58	63	37	41	46.6
5	37	52	33	51	37	42.0
6	48	44	37	37	37	40.6
7	44	41	37	37	37	39.2
8	44	40	52	52	41	45.8
9	48	54	44	44	41	46.2
					Overall	44.2

Gray Code Whole-pitch, P_2R_3

Yankee Doodle

Population Size 400

Set Number	Best Performance of 5 Random Numbers 1	2	3	4	5	Average Result
0	47	50	41	55	62	51.0
1	41	76	41	44	66	53.6
2	41	63	44	53	48	49.8
3	55	37	44	48	37	44.2
4	62	51	44	55	41	50.6
5	41	65	41	70	30	49.4
6	52	70	44	41	52	51.8
7	71	55	44	44	58	54.4
8	43	50	44	65	48	50.0
9	55	62	65	55	58	59.0
					Overall	51.3

Appendix D

Average Information Content Per Note of Various Melodies

Melody	# Notes	Average (H)			
		Equiprobable Pe	One Note Pi	Two Notes P(i/j)	Three Notes P(i/jk)
Auld Lang Syne	28	4.8	2.496	1.469	0.472
Avignon	27	4.7	2.339	1.074	0.640
Farmer in the Dell	24	4.5	2.235	1.286	0.459
Frere Jacques	32	5.0	2.555	1.387	0.358
Happy Birthday	25	4.6	2.709	1.312	0.381
Hickory Dickory	30	4.9	2.642	1.710	0.482
Home on the Range	41	5.3	2.816	1.317	0.552
Jack & Jill	28	4.8	3.244	1.177	0.303
London Bridge	24	4.5	2.288	1.287	0.364
My Bonnie	38	5.2	2.848	1.194	0.716
Nobody Knows	31	4.9	2.233	1.311	0.467
Old MacDonald	26	4.7	2.408	1.226	0.167
Skip to my Lou	34	5.0	2.299	1.279	0.735
Twinkle, Twinkle	28	4.8	2.471	1.231	0.183
Working ..Railroad	29	3.8	2.391	1.314	0.519
Yankee Doodle	30	4.9	2.439	1.406	0.313

* italics are principal procedure melodies

Appendix E: Evaluating the Model Results

Binary Code Pitch-plus

P_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	68	62	69	72	61
2	61	66	63	63	59
5	63	68	65	56	61
6	55	65	55	61	56

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	69	71	70	75	75
2	70	70	56	70	58
5	57	67	56	66	62
6	76	69	70	73	70

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	58	62	61	57	64
2	55	60	58	61	55
5	61	46	53	53	56
6	58	62	58	61	62

Binary Code Pitch-plus

P_2R_3 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	55	79	75	47	34
2	43	34	37	66	61
5	70	75	83	79	69
6	37	38	37	43	37

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	55	35	42	60	51
2	76	73	92	76	42
5	71	60	71	60	75
6	51	56	62	84	75

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	42	52	55	52	41
2	73	76	76	67	66
5	62	58	41	65	54
6	41	62	66	50	48

Binary Code Pitch-plus

P_2R_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	79	79	69	76	77
2	79	70	83	77	86
5	81	79	83	79	79
6	79	79	83	79	79

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	92	87	88	85	92
2	92	88	88	88	87
5	88	84	92	92	88
6	92	88	88	88	87

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	68	71	70	70	72
2	73	67	72	67	66
5	67	75	71	75	75
6	68	73	74	73	64

Gray Code Pitch-plus

P_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	61	56	63	58	54
2	56	70	73	61	63
5	70	63	54	62	55
6	68	68	52	58	61

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	66	70	60	66	66
2	67	67	71	75	73
5	66	82	69	69	74
6	64	67	61	71	70

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	67	57	66	60	54
2	50	60	58	57	53
5	51	55	63	55	57
6	55	51	72	56	60

Gray Code Pitch-plus

P_2R_3 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	65	47	51	56	70
2	76	61	48	79	61
5	37	51	79	38	65
6	62	61	56	54	61

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	47	53	75	75	84
2	76	58	51	73	56
5	67	84	53	56	75
6	41	60	88	42	84

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	47	47	37	44	44
2	58	58	73	41	65
5	52	41	56	70	37
6	47	62	51	51	63

Gray Code Pitch-plus

P_2R_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	86	75	77	62	81
2	83	79	81	77	83
5	83	75	70	75	76
6	73	73	81	77	87

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	88	88	84	92	89
2	88	88	88	88	91
5	71	84	88	88	84
6	88	92	84	87	91

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	68	68	76	72	77
2	68	65	67	66	76
5	71	72	67	58	70
6	73	72	70	65	71

Binary Code Whole-pitch

P_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	66	51	55	58	52
2	51	52	69	54	65
5	69	56	52	58	56
6	58	52	48	65	55

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	80	73	76	52	70
2	65	75	62	61	71
5	60	76	73	75	70
6	62	71	70	66	67

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	56	55	60	57	62
2	63	57	63	53	57
5	54	57	48	58	56
6	52	62	50	54	53

Binary Code Whole-pitch

P_2R_3 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	79	61	56	51	51
2	73	79	56	47	51
5	66	38	43	41	79
6	43	70	43	43	73

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	79	61	56	51	51
2	73	79	56	47	51
5	66	38	43	41	79
6	43	70	43	43	73

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	55	60	44	37	55
2	62	52	44	48	44
5	37	46	54	51	55
6	42	41	41	44	41

Binary Code Whole-pitch

P_2R_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	75	69	73	73	70
2	69	83	75	77	72
5	65	83	79	86	75
6	65	65	65	65	69

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	88	88	80	87	84
2	88	85	84	67	88
5	85	84	87	88	84
6	91	75	84	84	88

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	73	73	66	72	64
2	65	70	66	68	63
5	72	70	68	70	65
6	68	68	68	70	67

Gray Code Whole-pitch

P_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	56	63	54	69	66
2	61	63	68	61	58
5	61	63	66	69	58
6	59	62	72	59	59

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	76	66	69	66	62
2	62	64	67	75	73
5	80	78	65	60	79
6	67	71	79	70	69

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	60	57	62	55	55
2	56	54	56	63	50
5	66	53	67	64	54
6	66	52	58	51	54

Gray Code Whole-pitch

P_2R_3 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	65	36	75	51	51
2	34	61	56	65	65
5	73	79	75	69	77
6	45	83	77	66	52

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	51	51	73	44	51
2	58	73	42	96	58
5	56	75	47	47	67
6	56	64	43	34	60

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	41	76	41	44	66
2	41	63	44	53	48
5	41	65	41	70	30
6	52	70	44	41	52

Gray Code Whole-pitch

P_2R_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	75	65	83	75	80
2	79	86	79	79	69
5	79	61	75	79	77
6	65	75	73	79	79

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	82	92	88	80	88
2	91	84	84	88	84
5	89	92	79	76	84
6	88	88	88	92	92

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	70	72	72	70	71
2	77	65	72	78	72
5	74	61	72	65	76
6	71	76	68	78	75

Circle of Fifths Pitch-plus

P_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	66	55	76	63	66
2	62	55	69	68	69
5	62	61	63	61	68
6	66	58	66	62	54

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	70	74	61	65	67
2	71	71	61	67	64
5	65	80	65	70	70
6	76	80	79	73	67

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	65	54	65	65	50
2	61	58	57	55	63
5	61	52	66	56	53
6	62	54	61	58	56

Circle of Fifths Pitch-plus

P_2R_3 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	37	41	75	61	41
2	37	47	43	43	61
5	43	75	51	56	58
6	55	75	75	37	73

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	60	58	67	60	52
2	51	51	51	96	67
5	65	60	71	92	71
6	92	84	71	92	67

Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	48	54	60	45	48
2	51	73	50	70	42
5	45	78	46	41	68
6	45	58	43	55	68

Circle of Fifths Pitch-plus

P_2R_2 Classifier Format

London Bridge

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	75	79	79	77	83
2	87	69	70	76	80
5	87	73	80	79	70
6	79	69	79	81	73

Old MacDonald

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	88	87	92	88	87
2	88	88	88	84	92
5	88	88	87	88	92
6	89	74	84	84	92

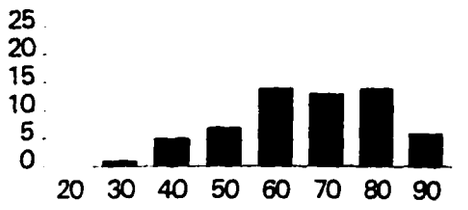
Yankee Doodle

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	70	66	61	73	72
2	72	64	72	75	66
5	71	68	70	68	77
6	68	66	73	75	70

Appendix F: Evaluating the Model Results for All Classifier Formats by Note Representation and Melody

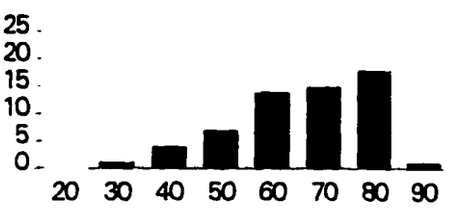
Old MacDonald

Gray Code Whole-pitch



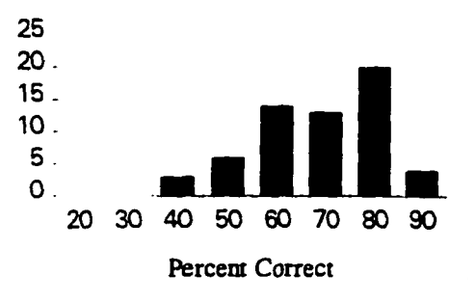
Mean 71.2% Std.Dev. 15.0

Binary Code Whole-pitch



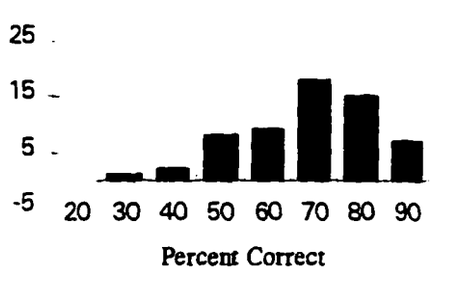
Mean 69.9% Std Dev 13.7

Gray Code Pitch-plus



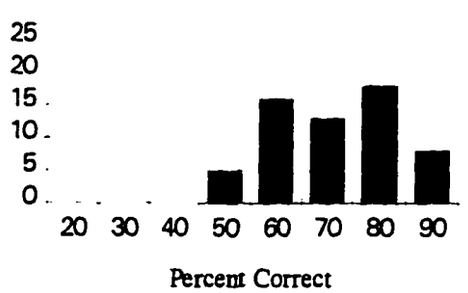
Mean 73.5% Std.Dev. 13.3

Binary Code Pitch-plus



Mean 73.1% Std Dev 14.4

Circle of Fifths Pitch-plus



Mean 75.3% Std Dev 12.5

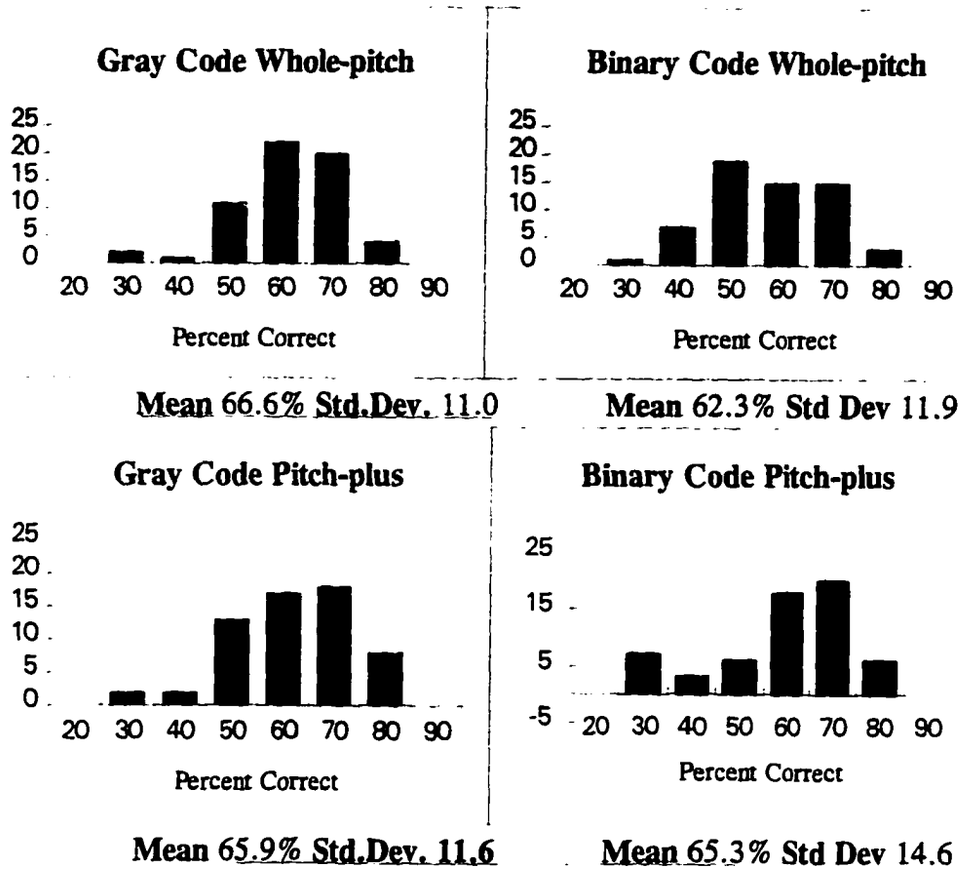
y-axis is the number of tests

Evaluating the Model Results

for All Classifier Formats

by Note Representation and Melody

London Bridge



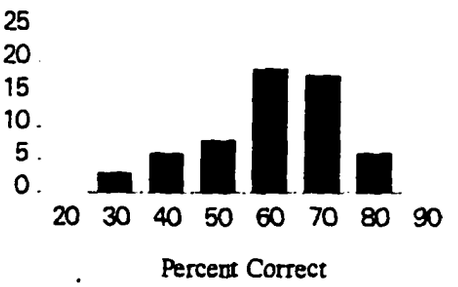
Mean 66.6% Std.Dev. 11.0

Mean 62.3% Std Dev 11.9

Mean 65.9% Std.Dev. 11.6

Mean 65.3% Std Dev 14.6

Circle of Fifths Pitch-plus



Mean 64.9% Std.Dev. 13.1

y-axis is the number of tests

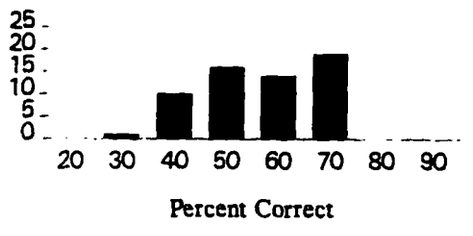
Evaluating the Model Results

for All Classifier Formats

by Note Representation and Melody

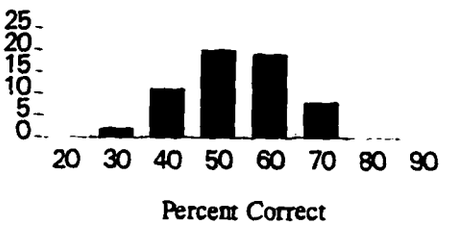
Yankee Doodle

Gray Code Whole-pitch



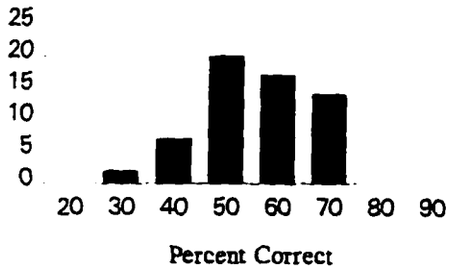
Mean 60.1% **Std.Dev.** 11.8

Binary Code Whole-pitch



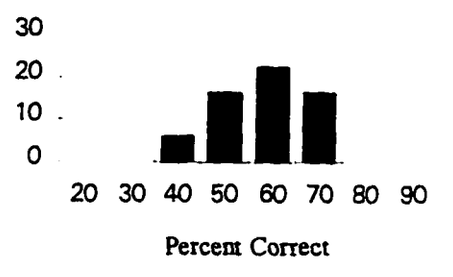
Mean 57.4% **Std Dev** 9.8

Gray Code Pitch-plus



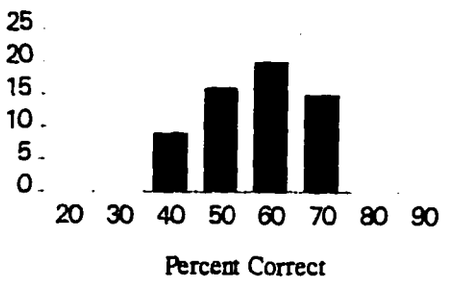
Mean 59.8% **Std.Dev.** 10.2

Binary Code Pitch-plus



Mean 61.9% **Std Dev** 9.3

Circle of Fifths Pitch-plus



Mean 60.9% **Std.Dev.** 9.8

*y-axis is the number of tests

Appendix G:

Applicability of Learning Results

Binary Code Pitch-plus

P_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	73	72	68	69	65
2	76	65	66	65	66
5	73	68	72	66	73
6	75	73	69	65	62

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	76	77	85	85	79
2	76	78	77	83	82
5	80	80	80	77	72
6	72	79	85	76	83

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	50	50	52	51	50
2	57	48	54	50	55
5	53	46	53	48	51
6	45	54	51	51	47

Binary Code Pitch-plus

P_2R_3 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	44	61	66	56	61
2	66	54	44	52	69
5	54	65	48	83	66
6	73	56	69	69	56

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	63	51	85	44	48
2	83	40	47	95	44
5	64	82	63	66	82
6	34	66	40	92	70

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	84	39	55	47	35
2	55	66	64	51	71
5	75	45	63	66	63
6	66	63	66	60	59

Binary Code Pitch-plus

P_2R_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	83	83	79	79	77
2	77	81	65	83	69
5	79	87	75	73	83
6	83	75	79	73	81

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	89	88	82	84	92
2	85	85	89	85	85
5	85	85	89	85	89
6	92	88	89	89	85

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	59	64	63	59	66
2	63	66	66	75	55
5	70	60	64	66	71
6	57	63	65	70	60

Gray Code Pitch-plus

P_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	72	62	62	66	69
2	61	75	62	69	63
5	68	70	65	69	63
6	69	65	63	65	69

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	78	79	82	86	78
2	83	80	72	85	79
5	76	83	76	78	75
6	80	85	82	83	83

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	55	54	55	51	58
2	51	53	52	61	54
5	52	55	55	64	51
6	51	55	57	58	51

Gray Code Pitch-plus

P_2R_3 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	65	50	70	75	69
2	40	75	70	70	48
5	61	73	61	70	61
6	47	65	56	66	61

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	51	67	52	40	47
2	44	47	35	63	41
5	82	60	55	73	57
6	73	40	39	71	71

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	47	58	47	51	40
2	63	78	70	57	55
5	46	64	52	52	73
6	55	63	47	46	75

Gray Code Pitch-plus

P_2R_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	76	73	86	79	77
2	76	73	73	79	79
5	76	83	83	73	77
6	77	75	70	80	81

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	86	92	77	91	88
2	90	89	77	78	90
5	92	92	89	86	78
6	82	85	89	89	89

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	76	67	70	70	70
2	66	63	66	67	66
5	75	70	64	63	78
6	59	55	73	61	70

Binary Code Whole-pitch

P_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	65	63	73	69	59
2	63	63	63	62	69
5	63	69	66	55	65
6	65	69	66	63	63

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	77	77	65	73	79
2	79	77	85	79	76
5	77	79	83	82	82
6	76	78	67	76	76

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	45	50	50	51	58
2	51	44	50	47	53
5	69	57	51	47	46
6	53	51	51	55	51

Binary Code Whole-pitch

P_2R_3 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	70	44	48	43	56
2	75	65	65	61	83
5	75	61	44	70	44
6	56	66	40	44	44

Twinkle, Twinkle Little Star

Set	Best Performance of 5 Random Numbers				
Number	1	2	3	4	5
1	40	75	71	53	78
2	59	78	52	82	64
5	85	55	75	75	70
6	88	71	92	82	64

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	55	48	52	47	40
2	45	52	52	66	64
5	42	55	75	72	44
6	54	40	59	82	64

Binary Code Whole-pitch

P_2R_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	79	70	79	73	79
2	77	81	79	81	76
5	81	79	81	79	79
6	79	73	79	76	79

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	85	89	77	82	89
2	82	82	85	82	79
5	85	85	84	85	89
6	89	89	84	88	89

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	60	63	63	54	76
2	69	59	63	63	72
5	59	63	66	66	59
6	64	69	66	70	75

Gray Code Whole-pitch

P_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	66	59	69	65	65
2	66	69	68	68	66
5	68	65	70	69	66
6	70	75	59	62	62

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	79	75	78	82	77
2	78	80	78	84	83
5	77	77	77	80	78
6	75	72	78	76	82

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	54	54	52	48	45
2	58	54	51	69	44
5	53	57	58	55	55
6	51	46	55	54	47

Gray Code Whole-pitch

P_2R_3 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	77	69	59	41	48
2	65	44	50	51	56
5	65	50	52	59	56
6	44	56	65	48	62

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	65	36	41	40	47
2	82	55	96	59	51
5	47	52	25	44	78
6	82	63	32	78	51

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	36	67	51	51	48
2	47	44	63	59	67
5	51	36	66	51	67
6	58	44	44	53	51

Gray Code Whole-pitch

P_2R_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	77	79	77	77	69
2	65	77	77	79	77
5	79	77	79	72	72
6	73	79	65	79	77

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	89	86	89	84	85
2	85	89	80	77	89
5	82	85	92	89	84
6	85	85	82	85	89

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	67	71	61	64	57
2	55	66	53	67	66
5	66	63	73	67	64
6	69	69	61	67	72

Circle of Fifths Pitch-plus

P_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	63	61	68	70	73
2	66	66	65	69	59
5	80	65	62	72	65
6	69	66	63	68	63

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	77	87	72	78	79
2	76	77	63	71	75
5	76	67	75	78	78
6	79	83	71	70	69

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	52	46	52	51	65
2	59	46	50	57	46
5	65	40	41	51	45
6	66	48	41	46	52

Circle of Fifths Pitch-plus

P_2R_3 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	73	66	65	50	79
2	62	52	52	50	70
5	48	66	56	69	77
6	69	44	38	70	80

Twinkle, Twinkle Little Star

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	55	67	41	46	52
2	40	63	47	47	78
5	55	51	42	66	75
6	51	71	70	76	51

Auld Lang Syne

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	44	47	36	52	45
2	58	75	55	44	70
5	59	46	64	46	48
6	55	70	71	73	55

Circle of Fifths Pitch-plus

P_2R_2 Classifier Format

Farmer in the Dell

Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	75	72	86	77	77
2	79	77	79	81	77
5	75	72	73	79	69
6	72	80	75	76	75

Twinkle, Twinkle Little Star

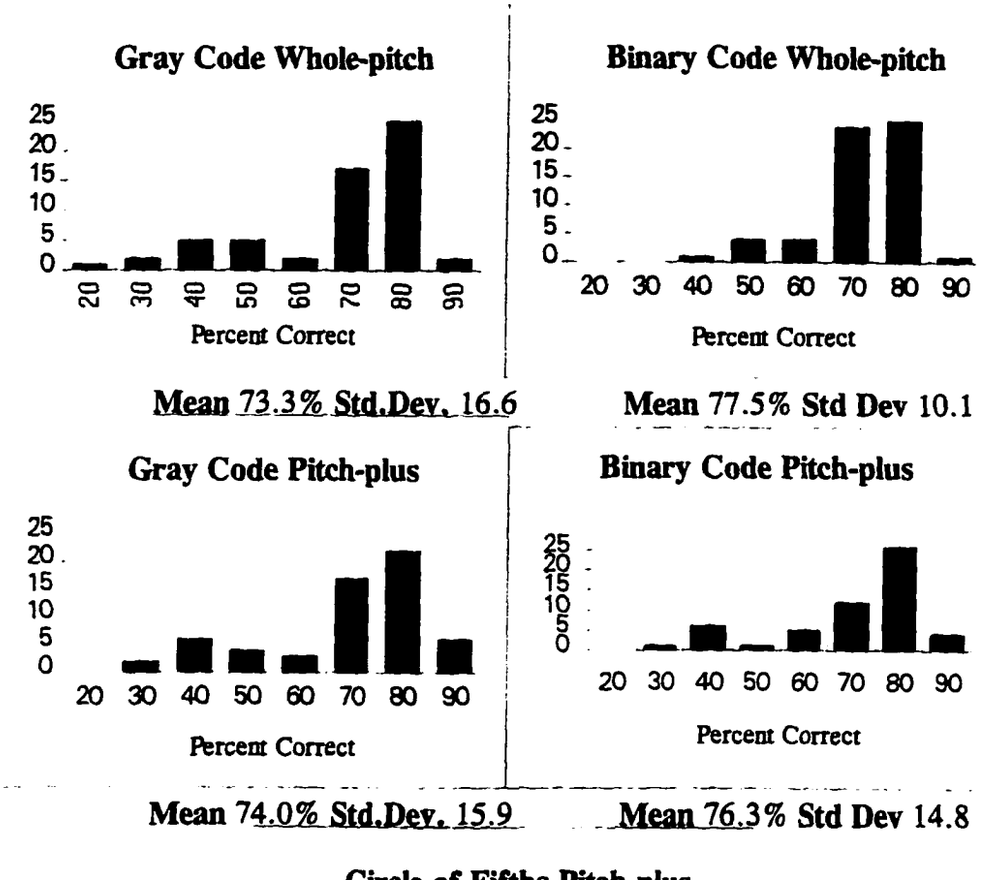
Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	85	85	89	85	88
2	82	89	85	88	89
5	84	85	90	84	84
6	82	82	92	85	85

Auld Lang Syne

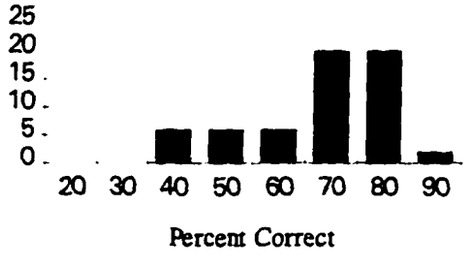
Set Number	Best Performance of 5 Random Numbers				
	1	2	3	4	5
1	58	66	63	65	77
2	66	70	71	64	57
5	58	59	55	59	60
6	66	70	71	69	71

Appendix H: Applicability of Learning Results for All Classifier Formats by Note Representation and Melody

Twinkle, Twinkle Little Star



Circle of Fifths Pitch-plus



Mean 72.7% Std.Dev. 14.1

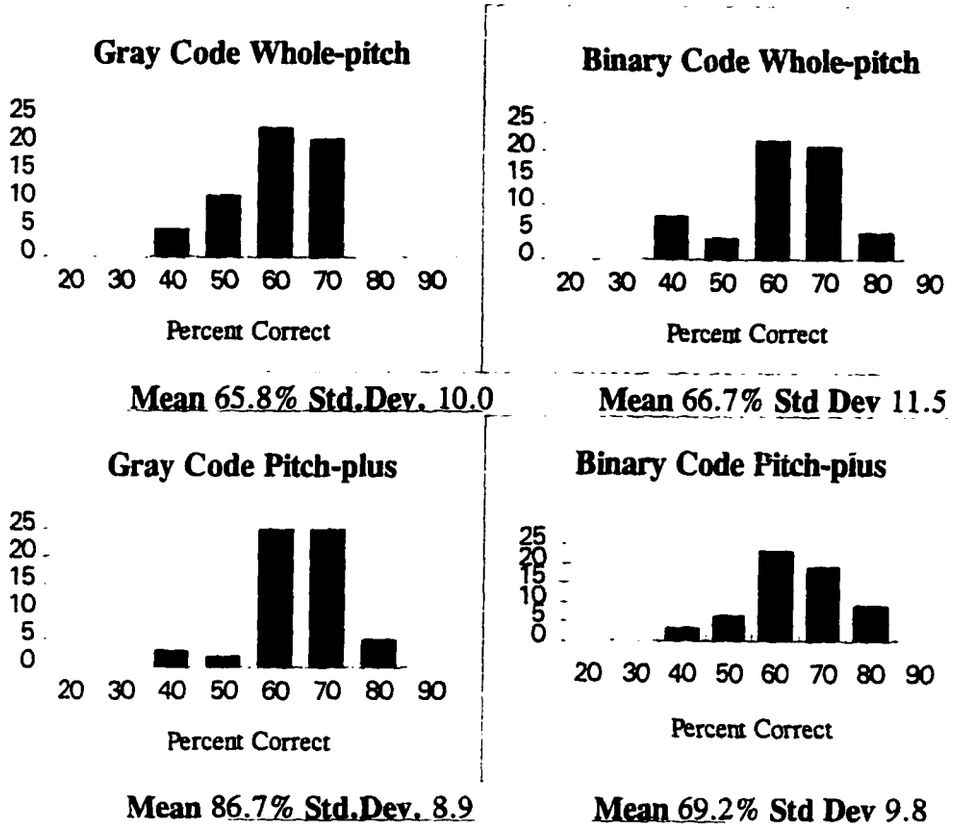
*y-axis is the number of tests

Applicability of Learning Results

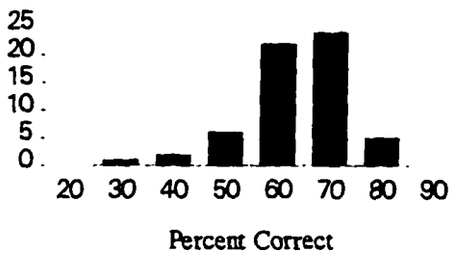
for All Classifier Formats

by Note Representation and Melody

Farmer in the Dell



Circle of Fifths Pitch-plus



Mean 68.2% Std Dev 9.7

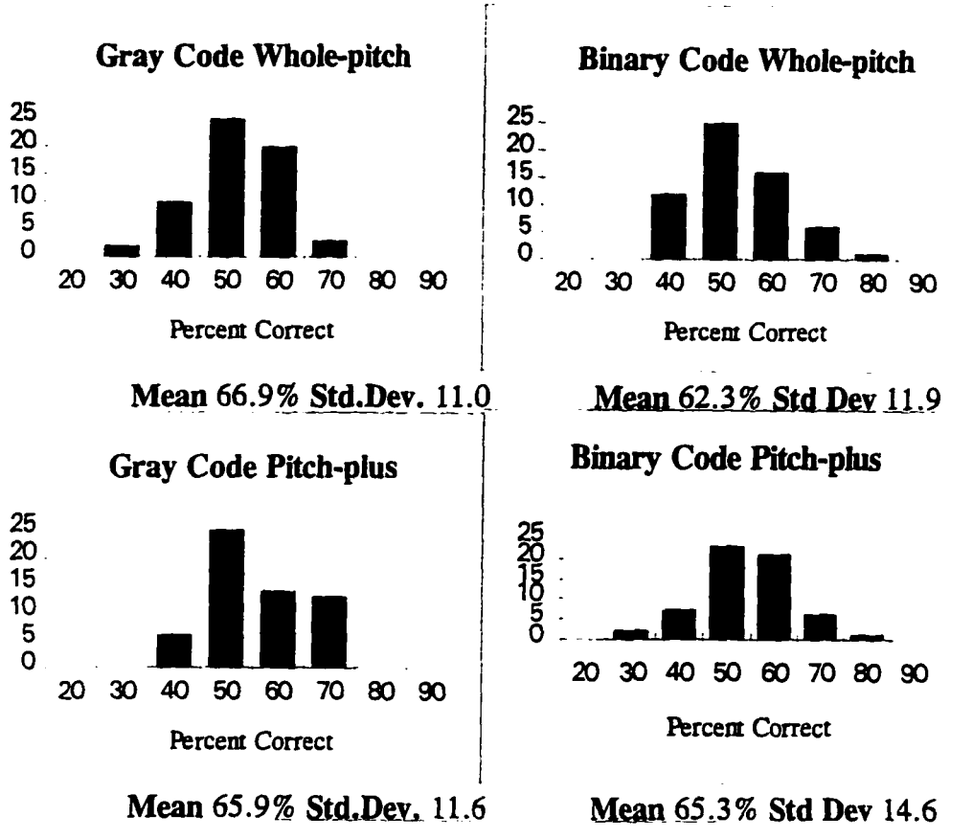
*y-axis is the number of tests

Applicability of Learning Results

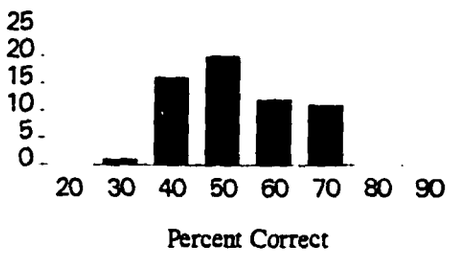
for All Classifier Formats

by Note Representation and Melody

Auld Lang Syne



Circle of Fifths Pitch-plus



Mean 64.9% Std.Dev. 13.1

•y-axis is the number of tests

Appendix I: Summary of Best Performance Results

Evaluating the Model

	Old MacDonald			London Bridge			Yankee Doodle		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
Binary Code Pitch-plus									
P_2	67.5	56	76	62.4	55	72	58.0	46	64
P_2R_3	63.3	35	92	54.9	34	83	57.3	41	76
P_2R_2	88.7	84	92	78.7	69	86	70.5	64	75
Binary Code Whole-pitch									
P_2	68.7	52	80	57.1	48	69	56.3	48	63
P_2R_3	56.5	38	79	57.1	38	79	47.6	37	62
P_2R_2	84.4	67	91	72.6	65	86	68.3	63	73
Circle of Fifths Pitch-plus									
P_2	69.8	61	80	63.5	54	76	58.6	50	66
P_2R_3	68.9	51	96	54.2	37	75	54.4	41	78
P_2R_2	87.0	74	92	77.2	69	87	69.8	61	77
Gray Code Pitch-plus									
P_2	68.7	60	82	61.3	52	73	57.8	50	72
P_2R_3	64.9	41	88	58.9	37	79	52.2	37	73
P_2R_2	87.0	71	92	77.7	62	87	69.6	58	77
Gray Code Whole-pitch									
P_2	69.9	60	80	62.3	54	72	57.6	50	67
P_2R_3	57.3	34	96	62.7	34	83	51.1	30	76
P_2R_2	86.4	76	92	75.6	61	86	71.7	61	78

Applicability of Learning

	Twinkle, Twinkle			Farmer in the Dell			Auld Lang Syne		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
Binary Code Pitch-plus									
P_2	79.1	72	85	69.0	62	76	50.8	45	57
P_2R_3	62.9	34	95	60.6	44	83	59.6	35	84
P_2R_2	87.0	82	92	78.2	65	87	64.1	55	75
Binary Code Whole-pitch									
P_2	77.1	65	85	64.6	55	73	51.5	44	69
P_2R_3	70.4	40	92	57.7	40	83	55.4	40	82
P_2R_2	84.9	77	89	77.9	70	81	64.9	54	76
Circle of Fifths Pitch-plus									
P_2	75.0	63	87	66.6	59	80	50.9	40	66
P_2R_3	57.2	40	78	61.8	38	80	55.6	36	75
P_2R_2	85.9	82	92	76.3	69	86	64.7	55	77
Gray Code Pitch-plus									
P_2	80.1	72	86	66.3	61	75	54.5	51	64
P_2R_3	55.4	35	82	62.6	40	75	56.9	40	78
P_2R_2	86.4	77	92	77.3	70	86	67.4	55	78
Gray Code Whole-pitch									
P_2	78.3	72	84	66.3	59	75	53.0	44	69
P_2R_3	56.2	25	96	55.8	41	77	52.7	36	67
P_2R_2	85.5	71	92	75.3	65	79	64.9	53	73

Appendix J: Trimmed Results

Binary Code Pitch-plus

P_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	68	69	62	63	63	61	65	63	61	61	55	56	62.2
Old MacDonald	71	70	75	70	70	58	66	57	62	73	70	70	67.6
Yankee Doodle	62	61	58	60	58	55	56	53	53	62	58	61	58.0
Farmer in the Dell	72	69	68	66	66	65	73	72	68	73	69	65	68.8
Twinkle, Twinkle	77	85	79	78	77	82	80	80	77	79	83	76	79.4
Auld Lang Syne	50	51	50	55	54	50	53	51	48	51	51	47	50.9

P_2R_3 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	75	55	47	43	37	61	75	79	70	38	37	37	54.5
Old MacDonald	55	42	51	76	76	73	71	71	60	56	62	75	64.0
Yankee Doodle	52	52	42	76	73	67	62	58	54	62	50	48	58.0
Farmer in the Dell	61	61	56	66	54	52	65	54	66	69	69	56	60.7
Twinkle, Twinkle	63	51	48	83	47	44	82	64	66	66	40	70	60.3
Auld Lang Syne	55	47	39	66	64	55	66	63	63	66	63	60	58.9

P_2R_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	79	77	76	79	83	77	81	79	79	79	79	79	78.9
Old MacDonald	92	88	87	88	88	88	88	92	88	88	88	88	88.5
Yankee Doodle	71	70	70	72	67	67	75	71	75	73	73	68	71.0
Farmer in the Dell	83	79	79	81	77	69	83	79	75	81	79	75	78.3
Twinkle, Twinkle	89	88	84	85	85	85	85	89	85	89	89	88	86.7
Auld Lang Syne	64	63	59	66	66	63	70	64	66	63	65	60	64.0

Gray Code Pitch-plus

P_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	61	58	56	70	63	61	63	62	55	68	61	58	61.3
Old MacDonald	66	66	66	67	71	73	74	69	69	67	64	70	68.5
Yankee Doodle	66	60	57	58	57	53	55	57	55	55	60	56	57.4
Farmer in the Dell	69	62	66	69	62	63	69	68	65	69	65	65	66.0
Twinkle, Twinkle	79	82	78	83	80	79	78	76	76	83	82	83	79.9
Auld Lang Syne	55	55	54	53	52	54	55	55	52	55	57	51	54.0

P_2R_3 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	65	51	56	76	61	61	51	65	38	61	61	56	58.5
Old MacDonald	53	75	75	73	58	56	75	67	56	60	84	42	64.5
Yankee Doodle	47	44	44	58	65	58	52	56	41	62	51	51	52.4
Farmer in the Dell	65	70	69	70	70	48	70	61	61	65	56	61	63.8
Twinkle, Twinkle	52	51	47	47	44	41	73	60	57	71	40	71	54.5
Auld Lang Syne	51	47	47	70	63	57	64	52	52	63	55	47	55.6

P_2R_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	81	77	75	83	81	79	76	75	75	73	81	77	77.7
Old MacDonald	88	88	89	88	88	88	84	88	84	91	88	87	87.5
Yankee Doodle	68	76	72	68	67	66	71	70	67	72	71	70	69.8
Farmer in the Dell	76	79	77	76	73	79	83	77	76	77	75	80	77.3
Twinkle, Twinkle	91	86	88	90	89	78	92	89	86	85	89	89	87.6
Auld Lang Syne	70	70	70	66	66	66	75	70	64	59	70	61	67.2

Binary Code Whole-pitch

P_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	58	55	52	52	65	54	58	56	56	58	52	55	55.9
Old MacDonald	76	73	70	71	65	62	75	73	70	70	67	66	69.8
Yankee Doodle	56	60	57	63	57	57	57	54	56	54	52	53	56.3
Farmer in the Dell	65	69	63	63	63	63	66	65	63	66	65	63	64.5
Twinkle, Twinkle	77	77	73	79	79	77	79	82	82	76	76	76	77.7
Auld Lang Syne	50	50	51	51	50	47	57	51	47	53	51	51	50.7

P_2R_3 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	61	56	51	73	56	51	66	43	41	70	43	43	54.5
Old MacDonald	63	60	60	58	52	58	61	48	52	60	51	48	55.9
Yankee Doodle	55	55	44	52	48	44	46	54	51	42	41	41	47.7
Farmer in the Dell	56	48	44	75	65	65	70	61	44	56	44	44	56.0
Twinkle, Twinkle	75	71	53	78	59	64	75	75	70	88	82	71	71.7
Auld Lang Syne	52	48	47	52	52	64	55	72	44	54	59	64	55.2

P_2R_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	73	73	70	77	75	72	83	79	75	65	65	65	72.6
Old MacDonald	88	87	84	88	85	84	85	87	84	88	84	84	85.6
Yankee Doodle	73	72	66	68	66	65	70	70	68	68	68	68	68.5
Farmer in the Dell	79	79	73	81	79	77	81	79	79	79	79	76	78.4
Twinkle, Twinkle	89	85	82	82	82	82	85	85	85	89	89	88	85.2
Auld Lang Syne	63	63	60	69	63	63	63	66	59	69	66	70	64.5

Gray Code Whole-pitch

P_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	63	56	66	63	61	61	63	66	61	62	59	59	61.6
Old MacDonald	69	66	66	64	67	73	79	78	65	71	70	69	69.7
Yankee Doodle	60	57	55	56	56	54	66	64	54	58	54	52	57.1
Farmer in the Dell	66	65	65	68	68	66	68	69	66	70	62	62	66.2
Twinkle, Twinkle	79	78	77	80	78	83	77	77	78	75	78	76	78.0
Auld Lang Syne	54	52	48	58	54	51	57	55	55	51	54	47	53.0

P_2R_3 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	65	51	51	61	56	65	77	75	73	77	66	52	64.0
Old MacDonald	51	51	51	73	58	58	67	56	47	60	56	43	55.9
Yankee Doodle	66	41	44	53	44	48	65	41	41	52	52	44	49.2
Farmer in the Dell	69	59	48	56	50	51	59	52	56	56	62	48	55.5
Twinkle, Twinkle	47	41	40	82	59	55	52	47	44	78	63	51	54.9
Auld Lang Syne	51	51	48	47	63	59	51	66	51	53	44	51	52.9

P_2R_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	75	80	75	79	79	79	79	75	77	75	73	79	77.0
Old MacDonald	88	88	82	88	84	84	89	84	79	88	88	92	86.1
Yankee Doodle	72	71	70	77	72	72	74	72	65	76	71	75	72.2
Farmer in the Dell	77	77	77	77	77	77	79	77	72	79	73	77	76.5
Twinkle, Twinkle	89	86	85	89	85	80	85	89	84	85	85	85	85.5
Auld Lang Syne	67	64	61	66	55	66	66	67	64	69	69	67	65.0

Circle of Fifths Pitch-plus

P_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	66	66	63	62	69	68	62	63	61	66	62	58	63.8
Old MacDonald	70	67	65	71	67	64	70	65	70	79	76	73	69.7
Yankee Doodle	65	65	54	61	58	57	61	56	53	61	58	56	58.7
Farmer in the Dell	63	68	70	66	66	65	72	65	65	68	66	63	66.4
Twinkle, Twinkle	79	77	78	76	75	71	76	75	78	79	71	70	75.4
Auld Lang Syne	52	52	51	57	50	46	51	41	45	52	48	46	49.2

P_2R_3 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	41	61	41	47	43	43	58	51	56	75	73	55	53.6
Old MacDonald	60	60	58	51	51	67	65	71	71	92	84	71	66.7
Yankee Doodle	54	48	48	70	51	50	68	46	45	58	45	55	53.1
Farmer in the Dell	73	66	65	62	52	52	66	56	69	69	44	70	62.0
Twinkle, Twinkle	55	52	46	63	47	47	55	51	66	71	70	51	56.1
Auld Lang Syne	47	44	45	70	58	55	59	48	46	70	71	55	55.6

P_2R_2 Classifier Format

Melody	Best Performance of Each Set												Average Result
	1			2			5			6			
London Bridge	79	79	77	80	70	76	80	79	73	79	79	73	77.0
Old MacDonald	88	88	87	88	88	88	88	88	88	89	84	84	87.3
Yankee Doodle	70	66	72	72	72	66	71	70	68	68	73	70	69.8
Farmer in the Dell	75	77	77	79	79	77	75	73	72	76	75	75	75.8
Twinkle, Twinkle	85	88	85	89	85	88	85	84	84	82	85	85	85.4
Auld Lang Syne	66	63	65	70	66	64	59	58	59	70	71	69	65.0

Appendix K: Trimmed Results Population Set 1

Population Size 100

	Old MacDonald			London Bridge			Yankee Doodle		
Binary Code Pitch-plus									
P_2	84	84	83	73	68	73	67	65	64
P_2R_3	44	26	28	37	37	61	37	35	37
P_2R_2	88	88	64	83	76	79	70	80	76
Binary Code Whole-pitch									
P_2	84	84	80	62	59	63	61	63	66
P_2R_3	26	44	38	36	31	33	37	37	37
P_2R_2	79	79	75	75	73	75	65	66	65
Circle of Fifths Pitch-plus									
P_2	79	79	84	72	69	69	72	67	66
P_2R_3	26	42	35	36	33	33	37	37	33
P_2R_2	75	80	71	83	51	73	65	55	55
Gray Code Pitch-plus									
P_2	80	84	80	63	76	72	66	65	68
P_2R_3	52	35	34	37	33	37	41	37	37
P_2R_2	85	88	88	75	66	83	55	61	70
Gray Code Whole-pitch									
P_2	79	84	85	72	77	73	67	72	74
P_2R_3	34	30	30	44	38	34	37	37	37
P_2R_2	84	69	80	79	73	79	73	76	76

Population Size 200

	Old MacDonald			London Bridge			Yankee Doodle		
Binary Code Pitch-plus									
P_2	79	71	80	69	72	66	64	64	57
P_2R_3	47	35	51	36	37	38	40	44	41
P_2R_2	75	87	79	75	51	65	75	75	72
Binary Code Whole-pitch									
P_2	71	74	76	58	59	61	53	55	57
P_2R_3	42	52	47	47	47	43	41	44	41
P_2R_2	88	84	88	75	73	69	70	70	70
Circle of Fifths Pitch-plus									
P_2	75	75	70	63	66	62	62	67	65
P_2R_3	47	43	48	43	48	48	41	37	40
P_2R_2	84	84	75	81	83	79	73	71	75
Gray Code Pitch-plus									
P_2	75	76	78	73	73	65	64	58	62
P_2R_3	43	35	38	37	61	33	37	37	37
P_2R_2	92	84	88	79	81	79	70	72	76
Gray Code Whole-pitch									
P_2	75	73	76	65	69	65	58	63	66
P_2R_3	47	46	47	55	61	43	44	41	48
P_2R_2	80	84	84	83	81	73	65	70	66

Population Size 400

	Old MacDonald			London Bridge			Yankee Doodle		
Binary Code Pitch-plus									
P_2	71	70	75	68	69	62	62	61	58
P_2R_3	55	42	51	75	55	47	52	52	42
P_2R_2	42	51	87	79	77	76	71	70	70
Binary Code Whole-pitch									
P_2	76	73	70	58	55	52	56	60	57
P_2R_3	63	60	60	61	56	51	55	55	44
P_2R_2	88	87	84	73	73	70	73	72	66
Circle of Fifths Pitch-plus									
P_2	70	67	65	66	66	63	65	65	54
P_2R_3	60	60	58	41	61	41	54	48	48
P_2R_2	88	88	87	79	79	77	70	66	72
Gray Code Pitch-plus									
P_2	66	66	66	61	58	56	66	60	57
P_2R_3	53	75	75	65	51	56	47	44	44
P_2R_2	88	88	89	81	77	75	68	76	72
Gray Code Whole-pitch									
P_2	66	69	66	56	63	66	60	57	55
P_2R_3	51	51	51	65	51	51	66	41	44
P_2R_2	88	88	82	75	80	75	72	71	70

Bibliography

Anderson, J. R. 1983. *The Architecture of Cognition.* Cambridge, Mass.: Harvard University Press.

Andrews, M. W., and Dowling, W. J. 1991. The Development of Perception of Interleaved Melodies and Control of Auditory Attention. *Music Perception.* Vol. 8, No. 4, pp. 349-368.

Belew, R. K., and S. Forrest. 1988. Learning and Programming in Classifier Systems. *Machine Learning*. Netherlands: Kluwer Academic Publishers.

Bharucha, J. J. 1991. Pitch, Harmony, and Neural Nets: A Psychological Perspective. In P. Todd, ed., *Music and Connectionism.* Cambridge, Ma.: The MIT Press.

Booker, L. B. 1985. Improving the Performance of Genetic Algorithms in Classifier Systems. In J. J. Grefenstette, ed., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications.* Hillsdale, New Jersey: Lawrence Erlbaum.

Booker, L. B. 1988. Classifier Systems that Learn Internal World Models. *Machine Learning*. Netherlands: Kluwer Academic Publishers.

Booker, L. B. 1989. Triggered Rule Discovery in Classifier Systems. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Booker, L. B., D. E. Goldberg, and J. H. Holland. 1989. Classifier Systems and Genetic Algorithms. In J. M. Carbonell, ed., *Machine Learning: Paradigms and Methods*. Cambridge, Ma.: The MIT Press.

Carse, B., and T. C. Fogarty. 1994. A Delayed-Action Classifier System for Learning in Temporal Environments. *Proceedings of the First IEEE Conference on Evolutionary Computation.* San Mateo, Ca.: Morgan Kaufman, Inc.

Caruana, R. A., and J. D. Schaffer. 1988. Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*. San Mateo, Ca.: Morgan Kaufman, Inc.

Davidson, L., P. McKernon, and H. Gardner. 1981. The Acquisition of Song: A Developmental Approach. In *Documentary Report of the Ann Arbor*

Symposium: Applications of Psychology to the Teaching and Learning of Music. Reston, Va.: Music Educators National Conference.

Davis, L. 1991. *Handbook of Genetic Algorithms*. New York, N. Y.: Van Nostrand Reinhold.

Davis, L., and M. Steenstrup. 1987. Genetic Algorithms and Simulated Annealing: An Overview. In L. Davis, ed., *Genetic Algorithms and Simulated Annealing*. Altos, Ca.: Morgan Kaufman, Inc.

Davis, L., and D. K. Young. 1988. Classifier Systems with Hamming Weights. In *Proceedings of the Fifth International Conference on Machine Learning*. San Mateo, Ca.: Morgan Kaufman, Inc.

DeJong, K. 1988. Learning with Genetic Algorithms: An Overview. *Machine Learning*. Netherlands: Kluwer Academic Publishers.

DeJong, K. 1975. *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Mi.

Deutsch, D. 1982. The Processing of Pitch Combinations. In D. Deutsch, ed., *The Psychology of Music*. Orlando, Fla.: Academic Press, Inc.

Dietterich, T. G., and R. S. Michalski. 1986. Learning to Predict Sequences. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach, Vol. II*. Altos, Ca.: Morgan Kaufman, Inc.

Dowling, W. J. 1988. Tonal Structure and Children's Early Learning of Music. In J. A. Sloboda, ed., *Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition*. New York, N.Y.: Oxford University Press.

Dowling, W. J. 1993. Procedural and Declarative Knowledge in Music Cognition and Education. In T. J. Tighe, and W. J. Dowling, eds., *Psychology and Music*. Hillsdale, N. J.: Lawrence Erlbaum.

Dowling, W. J., and D. L. Harwood. 1986. *Music Cognition*. Orlando, Fla.: Academic Press, Inc.

Ebcioglu, K. 1992. An Expert System for Harmonizing Chorales in the Style of J. S. Bach. In M. Balaban, K. Ebcioglu, and O. Laske, eds., *Understanding*

Music with AI: Perspectives on Music Cognition. Menlo Park, Ca.: The AAAI Press.

Federman, F. H., and J. A. Jones. 1995. A Computer Model of Early Childhood Development of Familiar Melodies. *Annual Conference of the Society for Music Perception and Cognition*. University of California, Berkeley, Ca.

Field, P. 1995. *A Multary Theory for Genetic Algorithms: Unifying Binary and Nonbinary Problem Representations*. Ph.D. Dissertation, Department of Computer Science, Queen Mary and Westfield College, University of London, United Kingdom.

Goldberg, D. E. 1985. Optimal Initial Population Size for Binary-coded Genetic Algorithms. *TCGA Report Number 850001, The Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics*. University of Alabama University, Alabama.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Ma.: Addison-Wesley.

Goldberg, D. E. 1990. Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding. *Machine Learning*. Vol. 5, No. 4. Netherlands: Kluwer Academic Publishers.

Goldberg, D. E. 1996. E-mail to author, April 22. Department of General Engineering, University of Illinois, Champaign, Ill.

Grefensteete, J. J. 1990. Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms. In J. W. Shavlik and T. G. Dietterich, eds., *Readings in Machine Learning*. San Mateo, Ca.: Morgan Kaufman, Inc.

Hargreaves, D. J. 1986. *The Developmental Psychology of Music*. New York, N.Y.: Cambridge University Press.

Holland, J. H. 1985. Properties of the Bucket Brigade. In J. J. Grefenstette, ed., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Hillsdale, N. J.: Lawrence Erlbaum.

Holland J. H. 1986. Escaping Brittleness: The Possibilities of General-purpose Learning Algorithms Applied to Parallel Rule-based Systems. In R. S Michalski, J. G. Carbonell, and T. M Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach, Vol. II*. Altos, Ca.: Morgan Kaufman, Inc.

Holland, J. H. 1990. Concerning the Emergence of Tag-mediated Lookahead in Classifier Systems. *Physica D*. Vol. 42D. North-Holland, Amsterdam: Elsevier Science Publishers B.V.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems*. Cambridge, Ma.: The MIT Press.

Holland, J. H., K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. 1986. *Induction*. Cambridge, Ma.: The MIT Press.

Horner, A., and D. E. Goldberg. 1991. Genetic Algorithms and Computer-Assisted Music Composition. In R. K. Belew and L. B. Booker, eds., *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Janikow, C. Z. 1993. CSM: A Knowledge-Intensive Genetic Algorithm for Supervised Learning. *Machine Learning*. Vol. 13. Netherlands: Kluwer Academic Publishers.

Jones, J. A., and F. H. Federman. 1993. A Computer Model of Inductive Music Learning. *Annual Conference of the Society for Music Perception and Cognition*. University of Pennsylvania, Philadelphia, Pennsylvania.

Klahr, D., P. Langley, and R. Neches. 1987. Learning, Development, and Production Systems. *Production System Models of Learning and Development*. Cambridge, Ma.: The MIT Press.

Kobayshi, H. 1978. *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*. Reading, Ma.: Addison-Wesley.

Krumhansl, C. L. 1990. *Cognitive Foundations of Musical Pitch*. New York, N.Y.: Oxford University Press, Inc.

Laine, P., and M. Kuuskankare. 1994. Genetic Algorithms in Musical Style Oriented Generation. *Proceedings of the First IEEE Conference on Evolutionary Computation*. San Mateo, Ca.: Morgan Kaufman, Inc.

Lerdahl, F. and R. Jackendoff. 1993. An Overview of Hierarchical Structure in Music. In S. M. Schwanauer and D. A. Levitt, eds., *Machine Models of Music*. Cambridge, Ma.: The MIT Press.

Longuet-Higgins, H. C. 1987a. *Mental Processes Studies in Cognitive Science*. Cambridge, Ma.: The MIT Press.

McAulay, A. D., and C. Oh. 1994. Improving Learning of Genetic Rule-based Classifier Systems. *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 24, No. 1.

McIntyre, R. A. 1994. Bach in a Box: The Evolution of Four Part Baroque Harmony Using the Genetic Algorithm. *Proceedings of the First IEEE Conference on Evolutionary Computation*. San Mateo, Ca.: Morgan Kaufman, Inc.

Michalski, R. S., J. G. Carbonell, and T. M. Mitchell, eds. 1983. *Machine Learning: An Artificial Intelligence Approach*. Vol. I. Palo Alto, Ca.: Tioga Publishing Company.

Miller, B. O., D. L. Scarborough, and J. A. Jones. 1992. On the Perception of Meter. In M. Balaban, K. Ebcioglu, and O. Laske, eds., *Understanding Music with AI: Perspectives on Music Cognition*. Menlo Park, Ca.: The AAAI Press.

Miller, J. H., and S. Forrest. 1989. The Dynamical Behavior of Classifier Systems. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Minsky, M. 1981. Music, Mind, and Meaning. *Computer Music Journal*. Vol. 5, No. 3.

Mitchell, M. 1996. *Introduction to Genetic Algorithms*. Cambridge, Mass.: The MIT Press.

Mitchell, M., and S. Forrest. 1993. Genetic Algorithms and Artificial Life. *Santa Fe Institute Working Paper 93-11-072*.

Mozer, M. C. 1994. Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multiscale Processing. *Connection Science*. Vol. 6, No. 2-3, pp. 247-280.

Newell, A., P. S. Rosenbloom, and J. E. Laird. 1989. Symbolic Architectures for Cognition. In M. I. Posner, ed., *Foundations of Cognitive Science*. Cambridge, Ma.: The MIT Press.

Norusis, M. J. 1990. *The SPSS Guide to Data Analysis*. Chicago, Il.: SPSS Inc.

Riolo, R. L. 1986. CFS-C: A Package of Domain Independent Subroutines for Implementing Classifier Systems in Arbitrary, User-Defined Environments.

Logic of Computers Group, Division of Computer Science and Engineering. University of Michigan, Ann Arbor.

Riolo, R. L. 1988. LETSEQ: An Implementation of the CFS-C Classifier System in a Task-domain that Involves Learning to Predict Letter Sequences. *Logic of Computers Group, Division of Computer Science and Engineering*. University of Michigan, Ann Arbor.

Riolo, R. L. 1989. The Emergence of Coupled Sequences of Classifiers. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Riolo, R. L. 1991. Modeling Simple Human Category Learning with a Classifier System. In R. K. Belew and L. B. Booker, eds., *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Riolo, R. L. 1995. E-mail to author, December 15. University of Michigan, Ann Arbor.

Robertson, G. G. 1988. Population Size in Classifier Systems. In *Proceedings of the Fifth International Conference on Machine Learning*. San Mateo, Ca.: Morgan Kaufman, Inc.

Robertson, G. G., and R. L. Riolo. 1988. A Tale of Two Classifier Systems. *Machine Learning*. Netherlands: Kluwer Academic Publishers.

Rosner, B. S. and L. B. Meyer. 1982. Melodic Processes and the Perception of Music. In D. Deutsch, ed., *The Psychology of Music*. Orlando, Fla.: Academic Press, Inc.

Schaffer, J. D., Caruana, R. A., L. J. Eshelman, and R. Das. 1989. A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Schuurmans, D., and J. Schaeffer. 1989. Representational Difficulties with Classifier Systems. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Schwartz, M. 1970. *Information Transmission, Modulation, and Noise*. New York: McGraw-Hill Book Company.

Seyer, P., A. Novick, and P. Harmon. 1982. *What Makes Music Work.* Burlingame, Ca.: Seyer Associates.

Shavlik, J. W., and T. G. Dietterich. 1990. General Aspects of Machine Learning. *Readings in Machine Learning.* San Mateo, Ca.: Morgan Kaufman, Inc.

Sheppard, R. N. 1982. Structural Representations of Musical Pitch. In D. Deutsch, ed., *The Psychology of Music.* Orlando, Fla.: Academic Press, Inc.

Shu, L. 1992. *The Impact of Data Structures on the Performance of Genetic Algorithm Based Learning.* Ph.D. Dissertation, Department of Computer Science, University of Alberta, Canada.

Shu, L., and J. Schaeffer. 1989. VCS: Variable Classifier Systems. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms.* San Mateo, Ca.: Morgan Kaufman, Inc.

Simon, H. A., and K. Kotovsky. 1963. Human Acquisition of Concepts for Sequential Patterns. *Psychological Review.* Vol. 70, No. 6, pp. 534-546.

Simon, H. A., and R. K. Sumner. 1993. Pattern in Music. In S. M. Schwanauer and D. A. Levitt, eds. *Machine Models of Music.* Cambridge, Ma.: The MIT Press.

Sloboda, J. A. 1985. *The Musical Mind: The Cognitive Psychology of Music.* New York, N.Y.: Oxford University Press.

Smith, R. E. 1992. A Report on the First International Workshop on Learning Classifier Systems. *The First International Conference on Learning Classifier Systems.* Houston, Texas.

Smith, R. E., and H. B. Cribbs III. 1994. Is a Learning Classifier System a Type of Neural Network? *Evolutionary Computation.* Vol. 2, No. 1.

Smith, R. E., and D. E. Goldberg. 1990. Reinforcement Learning With Classifier Systems. In Zeigler and Rozenblit, eds., *Proceedings: AI, Simulation and Planning in High Autonomy Systems.* Alamitos, Ca.: IEEE Computer Society Press.

Smith, R. E., and D. E. Goldberg. 1991. Variable Default Hierarchy Separation in a Classifier System. In G. J. Rawlins, ed., *Foundations of Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Smith, R. E., and M. Valenzuela-Rendon. 1989. A Study of Rule Set Development in a Learning Classifier System. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Spears, W. M. 1993. Crossover or Mutation? In L. D. Whitley, ed., *Foundations of Genetic Algorithms 2*. San Mateo, Ca.: Morgan Kaufman, Inc.

Spears, W. M., and K. A. DeJong. 1991. An Analysis of Multi-point Crossover. In G. J. Rawlins, ed., *Foundations of Genetic Algorithms*. San Mateo, Ca.: Morgan Kaufman, Inc.

Spears, W. M., and K. A. DeJong. 1992. Using Genetic Algorithms For Supervised Concept Learning. In B. P. Buckles and F. E. Petry, eds., *Genetic Algorithms*. Alamitos, Ca.: IEEE Computer Society Press.

Thorpe, L. A., S. E. Trehub, B. A. Morongiello, and D. Bull. 1988. Perceptual Grouping By Infants and Preschool Children. *Developmental Psychology*. Vol. 24, No. 4, pp. 484-491.

Todd, P. M. 1991. A Connectionist Approach To Algorithmic Composition. In P. Todd, ed., *Music and Connectionism*. Cambridge, Ma.: The MIT Press.

Trehub, S. E. 1987. Infants' Perception of Musical Patterns. *Perception & Psychophysics*. Vol. 41, No. 6, pp. 635- 641.

Trehub, S. E., D. Bull, and L. A. Thorpe. 1984. Infants' Perception of Melodies: the Role of Melodic Contour. *Child Development*, pp. 821-830.

Trehub, S. E., B. A. Morongiello, and L. A. Thorpe. 1985. Children's Perception of Familiar Melodies: The Role of Intervals, Contour, and Key. *Psychomusicology*, Vol. 5, pp. 39-48.

Wagman, M. 1991. *Artificial Intelligence and Human Cognition*. New York, N.Y.: Praeger Publishers.

Westerdale, T. H. 1991. Redundant Classifiers and Prokaryote Genomes. In R. K. Belew and L. B. Booker, eds., *Proceedings of the Fourth International*

Conference on Genetic Algorithms. San Mateo, Ca.: Morgan Kaufman, Inc.

Widmer, G. 1992. The Importance of Basic Musical Knowledge for Effective Learning. In M. Balaban, K. Ebcioglu, and O. Laske, eds., *Understanding Music with AI: Perspectives on Music Cognition*. Menlo Park, Ca.: The AAAI Press.

Wilson, S. W. 1987. Quasi-darwinian Learning in a Classifier System. In P. Langley, ed., *Proceedings of the Fourth International Workshop on Machine Learning*. Los Altos, Ca.: Morgan Kaufman, Inc.

Wilson, S. W. 1994. ZCS: A Zeroth Level Classifier System. *Evolutionary Computation*. Vol. 2. No. 1.

Winston, P. H. 1993. *Artificial Intelligence*. Reading, Ma.: Addison-Wesley.

Zhou, H. H. 1987. *CSM: A Genetic Classifier System with Memory for Learning by Analogy*,

Zhou, H. H. 1991. Conceptual Learning and Classifier Systems with Long-term Memory. *Proceedings of the ACM Computer Science Conference*. New York, N.Y.: Association for Computing Machinery.