Automated Accompaniment

Tyler Seacrest

University of Nebraska, Lincoln

April 20, 2007

Artificial Intelligence

Professor Surkan
The problem as originally stated:
The problem as originally stated:

- Proposed Input

\[\text{\includegraphics[width=\textwidth]{music_notes}}\]
The problem as originally stated:

- Proposed Input

- Proposed Output
The Problem

General aims for the project:

- Develop a representation for musical knowledge that the computer can easily implement
- Have the computer analyze a piece of music, or sections of a piece of music
- Develop some approximation of computerized creativity
- Have the computer get better over time (didn't quite get to this one)
General aims for the project:

- Develop a representation for musical knowledge that the computer can easily implement
General aims for the project:

- Develop a representation for musical knowledge that the computer can easily implement
- Have the computer analyze a piece of music, or sections of a piece of music
General aims for the project:

- Develop a representation for musical knowledge that the computer can easily implement
- Have the computer analyze a piece of music, or sections of a piece of music
- Develop some approximation of computerized creativity
The Problem

General aims for the project:

- Develop a representation for musical knowledge that the computer can easily implement
- Have the computer analyze a piece of music, or sections of a piece of music
- Develop some approximation of computerized creativity
- Have the computer get better over time
General aims for the project:

- Develop a representation for musical knowledge that the computer can easily implement
- Have the computer analyze a piece of music, or sections of a piece of music
- Develop some approximation of computerized creativity
- Have the computer get better over time (didn’t quite get to this one)
Method

Method for tackling the above issues: Fuzzy rules!
Method

Fuzziness Method:

For each time the program wants to determine which chord to play next, an array of doubles called Key[] is created. It has size 12 (each one corresponds to a possible major chord; these are called “fuzzy variables”).

Various (fuzzy) rules are applied. For each rule, values are added or subtracted from some or all of the fuzzy variables associated with each chord.

At the end of the rule application process, the array Key[] might look like:

Key[] = (0, 0, 0.3, 2.4, 0.5, -0.4, 0, 0.7, 1.8, 1.1, 0, 0.5, 0.6)

Notice that the fourth value, 2.4, is the highest. As long as the highest value is above some threshold, then the chord corresponding to that value is played. In this case, a B-chord is played.
Method

Fuzziness Method:

- For each time the program wants to determine which chord to play next, an array of doubles called Key[] is created. It has size 12 (each one corresponds to a possible major chord; these are called "fuzzy variables")
Method

Fuzziness Method:

- For each time the program wants to determine which chord to play next, an array of doubles called `Key[]` is created. It has size 12 (each one corresponds to a possible major chord; these are called "fuzzy variables")
- Various (fuzzy) rules are applied. For each rule, values are added or subtracted from some or all of the fuzzy variables associated with each chord.
Method

Fuzziness Method:

- For each time the program wants to determine which chord to play next, an array of doubles called Key[] is created. It has size 12 (each one corresponds to a possible major chord; these are called "fuzzy variables")
- Various (fuzzy) rules are applied. For each rule, values are added or subtracted from some or all of the fuzzy variables associated with each chord.
- At the end of the rule application process, the array Key[] might look like

\[
\text{Key[]} = (0, \ 0, \ 0.3, \ 2.4, \ 0.5, \ -0.4, \\
0, \ 0.7, \ 1.8, \ 1.1, \ 0, \ 0.5, \ 0.6)
\]
Method

Fuzziness Method:

- For each time the program wants to determine which chord to play next, an array of doubles called Key[] is created. It has size 12 (each one corresponds to a possible major chord; these are called "fuzzy variables")

- Various (fuzzy) rules are applied. For each rule, values are added or subtracted from some or all of the fuzzy variables associated with each chord.

- At the end of the rule application process, the array Key[] might look like

  Key[] = (0, 0, 0.3, 2.4, 0.5, -0.4, 0, 0.7, 1.8, 1.1, 0, 0.5, 0.6)

- Notice that the fourth value, 2.4, is the highest. As long as the highest value is above some threshold, then the chord corresponding to that value is played. In this case, a B-chord is played.
We’ll get back to the main problem in a second, but the fuzzy rules method has an mini application to the problem of determining what key a piece of music is in.
Key Signature

We’ll get back to the main problem in a second, but the fuzzy rules method has an mini application to the problem of determining what key a piece of music is in.

- Most songs are written in a major scale (i.e., do, re, me, . . .). But there are actually 12 different major scales, one for each base note.
Key Signature

We’ll get back to the main problem in a second, but the fuzzy rules method has an mini application to the problem of determining what key a piece of music is in.

- Most songs are written in a major scale (i.e., do, re, me, . . .). But there are actually 12 different major scales, one for each base note. (so again we have 12 choices)
Key Signature

We’ll get back to the main problem in a second, but the fuzzy rules method has an mini application to the problem of determining what key a piece of music is in.

- Most songs are written in a major scale (i.e., do, re, me, ...). But there are actually 12 different major scales, one for each base note. (so again we have 12 choices)
- Given a melody, there is no easy algorithm for deciding what key the melody was written for (often a composer will go off the main scale to create dissonance).
Key Signature

We’ll get back to the main problem in a second, but the fuzzy rules method has an mini application to the problem of determining what key a piece of music is in.

- Most songs are written in a major scale (i.e., do, re, me, . . .). But there are actually 12 different major scales, one for each base note. (so again we have 12 choices)
- Given a melody, there is no easy algorithm for deciding what key the melody was written for (often a composer will go off the main scale to create dissonance).
- Using the fuzzy rules method listed above is a good way to determine the key signature.
What does this have to do with creativity?
What does this have to do with creativity?

- Possible operational definition of **Creativity**: Discovering an unexpected or unorthodox method that is nevertheless effective in some way (in the context of music, “effective” means “sounds decent”)
Creativity

What does this have to do with creativity?

- Possible operational definition of Creativity: Discovering an unexpected or unorthodox method that is nevertheless effective in some way (in the context of music, “effective” means “sounds decent”)

- Therefore, the computer must develop methods for adding accompaniment that cannot be easily anticipated, yet still “sound decent”.

How does this program use fuzzy rules to allow the computer to be “creative”?

The rules are written with two criteria in mind:

- They must take into musical guidelines
- They must be complicated enough that the combine in interesting ways (notice that the fuzziness of the rules allow for them to combine at all – with crisp rules, only one can be applied at a time)

In theory, because of the first criteria, the music generated by the computer will “sound good”. Because of the second criteria, the music will be unexpected and unorthodox.
How does this program use fuzzy rules to allow the computer to be “creative”?

- The rules are written with two criteria in mind:
How does this program use fuzzy rules to allow the computer to be “creative”?

- The rules are written with two criteria in mind:
  - They must take into musical guidelines
How does this program use fuzzy rules to allow the computer to be “creative”?

- The rules are written with two criteria in mind:
  - They must take into musical guidelines
  - They must be complicated enough that the combine in interesting ways (notice that the fuzziness of the rules allow for them to combine at all – with crisp rules, only one can be applied at a time)
Creativity

How does this program use fuzzy rules to allow the computer to be “creative”?

- The rules are written with two criteria in mind:
  - They must take into musical guidelines
  - They must be complicated enough that the combine in interesting ways (notice that the fuzziness of the rules allow for them to combine at all – with crisp rules, only one can be applied at a time)

- In theory, because of the first criteria, the music generated by the computer will “sound good”. Because of the second criteria, the music will be unexpected an unorthodox.
Example Rules

Example fuzzy rules:

Example fuzzy rules:

Rule 1: Use the next four notes to modify the array Key[] as if you were determining the key signature of a song containing those four notes. This tends to create a chord that “sounds good” with the notes around it.

Rule 2: Decrease all the values of array Key[] for every note at regular intervals. This will create rests at regular intervals, which is a common technique in music.

With a large collection of rules like above, you can then randomly select rules from the set, apply them to the melody, and you will get a new accompaniment every time.
Example fuzzy rules:

- **Rule 1**: Use the next four notes to modify the array \( \text{Key}[] \) as if you were determining the key signature of a song containing those four notes.
Example Rules

Example fuzzy rules:

- **Rule 1**: Use the next four notes to modify the array Key[] as if you were determining the key signature of a song containing those four notes.
  - This tends to create a chord that “sounds good” with the notes around it
Example fuzzy rules:

- **Rule 1**: Use the next four notes to modify the array `Key[]` as if you were determining the key signature of a song containing those four notes.
  - This tends to create a chord that “sounds good” with the notes around it

- **Rule 2**: Decrease all the values of array `Key[]` for every note at regular intervals

With a large collection of rules like above, you can then randomly select rules from the set, apply them to the melody, and you will get a new accompaniment every time.
Example fuzzy rules:

- **Rule 1**: Use the next four notes to modify the array `Key[]` as if you were determining the key signature of a song containing those four notes.
  - This tends to create a chord that “sounds good” with the notes around it
- **Rule 2**: Decrease all the values of array `Key[]` for every note at regular intervals
  - With will create rests at regular intervals, which is a common technique in music
Example fuzzy rules:

- **Rule 1**: Use the next four notes to modify the array `Key[]` as if you were determining the key signature of a song containing those four notes.
  - This tends to create a chord that “sounds good” with the notes around it

- **Rule 2**: Decrease all the values of array `Key[]` for every note at regular intervals
  - With will create rests at regular intervals, which is a common technique in music

- With a large collection of rules like above, you can then randomly select rules from the set, apply them to the melody, and you will get a new accompaniment every time.
Learning

“Without learning, you aren’t really doing AI”

-Paraphrased from Professor Surkan
Learning

To incorporate learning:

- Use a genetic algorithm to determine optimal rule selection.
  - Instead of randomly selecting rules to use, you could have a collection of individuals, where each individual is a set of rules.
  - During a supervised training process, the best sets of rules can be selected. The best sets can then be mutated or can be the parents of offspring sets of rules, and the process can repeat.
- Use a genetic algorithm to determine optimal weights for rules.
  - Currently, the weights used in the individual rules are arbitrary.
  - Similarly as above, a genetic algorithm could be used to optimize the weights used in the individual rules.
Learning

To incorporate learning:

- Use a genetic algorithm to determine optimal rule selection
  - Instead of randomly selecting rules to use, you could have a collection of individuals, where each individual is a set of rules.
  - During a supervised training process, the best sets of rules can be selected. The best sets can then be mutated or can be the parents of offspring sets of rules, and the process can repeat.

- Use a genetic algorithm to determine optimal weights for rules
  - Currently, the weights used in the individual rules are arbitrary.
  - Similarly as above, a genetic algorithm could be used to optimize the weights used in the individual rules.
To incorporate learning:

- Use a genetic algorithm to determine optimal rule selection
  - Instead of randomly selecting rules to use, you could have a collection of individuals, where each individual is a set of rules.
  - During a supervised training process, the best sets of rules can be selected. The best sets can then be mutated or can be the parents of offspring sets of rules, and the process can repeat.

- Use a genetic algorithm to determine optimal weights for rules
  - Currently, the weights used in the individual rules are arbitrary.
  - Similarly as above, a genetic algorithm could be used to optimize the weights used in the individual rules.
Questions
Questions

Questions?

This presentation and the Java Applet are available at:

http://www.math.unl.edu/~s-tseacre1/