Selection of Attributes for Modeling Bach Chorales by a Genetic Algorithm

Mark A. Hall (mhall@waikato.ac.nz)
Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Abstract

A genetic algorithm selected combinations of attributes for a machine learning system. The algorithm used 90 Bach chorale melodies to train models and randomly selected sets of 10 chorales for evaluation. Compression of pitch was used as the fitness evaluation criterion. The best models were used to compress a different test set of chorales and their performance compared to human generated models. G.A. models outperformed the human models, improving compression by 10 percent.

Introduction

There are many possible ways to describe the surface of a musical piece; any particular note could have literally hundreds of descriptive attributes associated with it. Apart from obvious static properties such as pitch and duration, there are many properties based on interval systems that could be applied. Some examples include the interval between a note and the previous note, the interval between a note and the note that started the measure, the interval between a note and the tonic of the chord, and the difference in start times of two consecutive notes.

If we model attributes individually, it is feasible to test the performance of each attribute. However, it is almost certain that there are relationships to be exploited between attributes: ie. there is predictive power to be gained by explicitly modelling the relationship between two or more attributes. If we have 20 attributes and we are interested in testing the performance of modelling 2 attributes together, then there are 20! / 2! 18! combinations to try. If we are interested in exploring all possible combinations of attributes, then there are 2^{20} combinations!

So, how do we decide which attributes to model? One obvious way would be to consult an expert and exploit his or her a priori musical knowledge. The disadvantage here is that knowledge will vary from expert to expert and is in no way guaranteed to be optimal. Also, with this method, our modelling technique be comes more domain dependent;if we move to a new domain, we must seek new experts. Ideally we want an automated method for selecting attributes to model; one that will work regardless of the domain. It is worth noting however, that although we will hopefully arrive at a method that optimally chooses combinations of attributes, the initial "pool" of properties or attributes that we choose from must still be designed by a human. This paper describes the application of an automated technique for selecting combinations of attributes.

Predictive models

There are many different types of models. With respect to music informal models have developed in our own minds over many years. Other models are constructed from formal theories. Computational models use adaptive learning techniques. All have the ability to predict upcoming events on the basis of what has been heard so far. The system described in [1] is an adaptive learning system that borrows modeling techniques from text compression, specifically P.P.M [2]. Additional power is gained through the use of multiple models ("viewpoints") simultaneously. Each model acts as an independent knowledge source viewing the music from its own perspective. These perspectives are constrained by how the music is represented and what can be derived from the representation. The Bach chorales that the system operates on are represented as discrete sequences of events. Each has a start time, pitch, duration and fermata indicator. More attributes can be derived from these "basic" types. Contour, for example, is derived by examining sequential pitch values. A given pitch is either the same as, higher, or lower than the preceding pitch. Other derived attributes include descriptors such as the interval between a note and the note that began the piece, the interval between a note and the first note in the bar, and the interval between notes that begin phrases.

Viewpoints

The concept behind viewpoints is to use background domain knowledge to arrive at new ways of describing events in a sequence. A viewpoint is defined by the attribute(s) it models. Each viewpoint has an underlying context model, where by context model we are referring to a subclass of the probabilistic finite-state, or Markov, class of grammars. A simple viewpoint models a single attribute. A linked viewpoint is one that models two or more attributes simultaneously. Sequences stored by a linked viewpoint are sets of tuples, eg. $[(a_1, a_2,...,a_n), (a_1, a_2,...,a_n),...]$. Where a_1 - a_n are the attributes modelled by the viewpoint. A tuple will be recorded by a viewpoint at event j, if and only if, all constituent attributes of the viewpoint are defined at that event.

Entropy

Our goal is to automate the selection of attributes for a linked viewpoint such that the predictive power of the viewpoint is optimised. From an information theoretic approach we can measure the predictive power of a viewpoint by the amount of redundancyit removes from a sequence of events. An entropy profile for a sequence of events can be derived by examining the probabilities that the model assigns to events that actually occur in the sequence. Intuitively, the higher the probability assigned to an event, the less surprising it is to the model, and the lower its entropy. Therefore, an entropy profile is an event-by-event plot of entropy against time. The entropy of an event is usually expressed in bits; it can be calculated from the formula $E = -\log_2 p$ [3], where p is the probability assigned to the event by the model. If the model as signs a probability of 50 percent to a event, then the entropy of that event with respect to the model is 1 bit. If the probability is 25 percent, then the entropy is 2 bits.

Entropy provides a measure of the performance of a viewpoint on a event-by-event basis; averaging the individual entropies gives a measure of the performance on the sequence as a whole.

Using a genetic algorithm to select attributes

As mentioned earlier, our goal is to automate the selection of attributes for a viewpoint. The only fail-safe way of locating the optimal combination of attributes is to try all combinations. Unfortunately this is computationally intractable even for a small pool of attributes. The method described here applies a genetic algorithm to select attributes. Genetic algorithms are machine learning and optimisation techniques based on the principles of natural selection in biology [4]. They use a population of competing solutions - evolved over time - to converge to an optimal solution. Effectively, the solution space is searched in parallel. Although they are not guaranteed to find the global optimum [5], the use of a population helps avoid local optima. The algorithm is an iterative process where each successive generation is producedby reproduction among the members of the previous generation. Selection of population members for reproduction is driven by fitness (determinedby some objective measure). There are numerous variations of genetic algorithms; many of them are 'tweaked' for optimal performance on a specific problem. The one presented here is the simple one outlined by Goldberg [5]. Despite its simplicity, it remains a powerful algorithm.

Genetic algorithm

Attribute combinations are coded as a 22 bit string. . If a bit is set (value 1), then the corresponding attribute is modelled in the linked viewpoint; if the bit is not set (value 0), then the attribute is not modelled by the linked viewpoint. Note that the algorithm also evaluates simple viewpoints, ie. those which model only a single attribute. Average entropy is used as the raw fitness of a viewpoint. The objective function builds a model of 90 chorales using the 22 bit string provided by the genetic algorithm. The average entropy of 10 test chorales with respect to the model becomes the fitness of the viewpoint. The lower the average entropy, the fitter the viewpoint. New generations are createdby three genetic operators: selection, crossover, and mutation. Generations are non-overlapping, meaning that in principle each generation is made up of new individuals. In practice however, some members of the new generation will be copies of their parents.

Selection is based on fitness by the simple biased roulette wheel method. Under this scheme, each population member is allocated a slice of the roulette wheel proportional to their fitness. Spinning the wheel provides a candidate for reproduction; those population members with high fitness have greater chance of being selected.

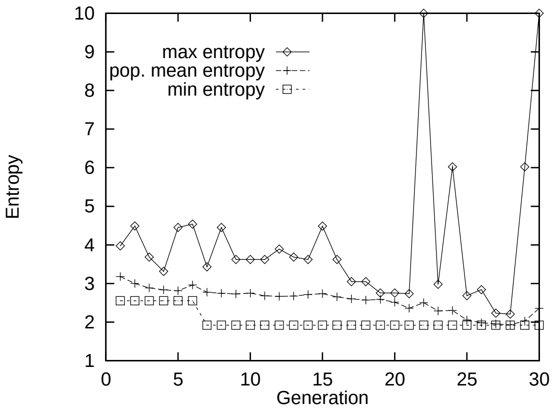
Crossover is the operator for creating new strings from pairs of selected individuals. In the experiments presented here, crossover occurs with a probability of 60 percent. When crossover does not occur, both selected individuals are copied into the new generation without change. When crossover does occur, a crossover point for the two individuals is randomly chosen, and the two offspring are created by concatenating the pieces of the two parents.

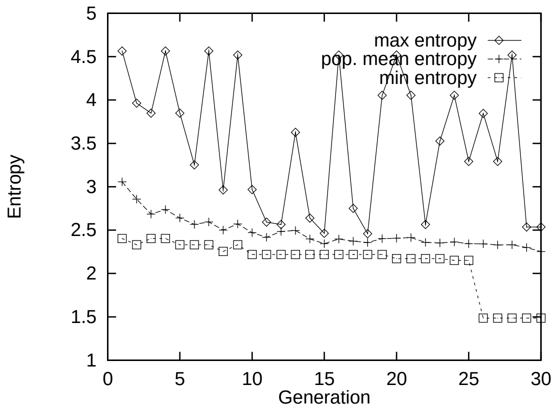
Mutation is a genetic operator designed to introduce a degree of random noise into the procedure by occasionally changing the value of a single bit, chosen at random from an individual. This helps avoid local optima. Mutation is usually applied with some low probability. In these experiments, mutation is used in two slightly different ways. Firstly, mutation is always applied to one offspring when parents selected for crossover are exact copies of oneanother. This insures that we introduce a new string every time crossover occurs. Secondly, a string is mutated if it doesn't contain a pitch related attribute. Since we are interested in the predicting pitch, a viewpoint that does not contain at least one pitch related attribute will never predict anything. Generation 0 is initialised by randomly setting two bits in each population member, with the constraint that at least one correspond to a pitch related attribute. Generation 0 is also constrained to contain only unique population members.

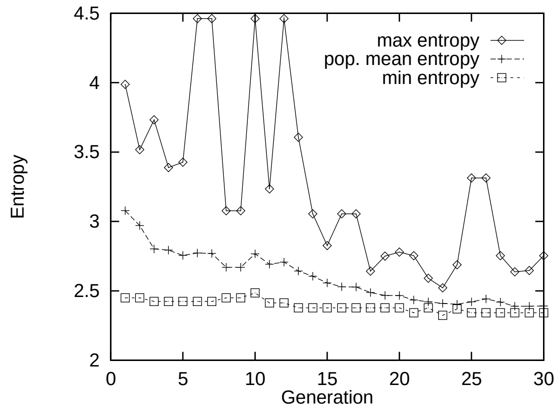
Results

G.A. runs

Figure 1 shows the population maximum, population minimum, and population mean entropy scores for the first experiment over 30 generations. From the 100 Bach chorales in the data base, ten were randomly selected and removed to from a fitness evaluation set. The remaining 90 were used for training. Average entropy starts off at just over 3 bits per pitch; midway through the run the average hovers around 2.7 bits per pitch and by the end the average is at 2.02 bits per pitch. It can be seen that right from the start the population contained a good solution (minimum entropy of 2.55 bits per pitch). This was supersededby a periodic viewpoint that became the population best for the rest of the run. This viewpoint predicts only

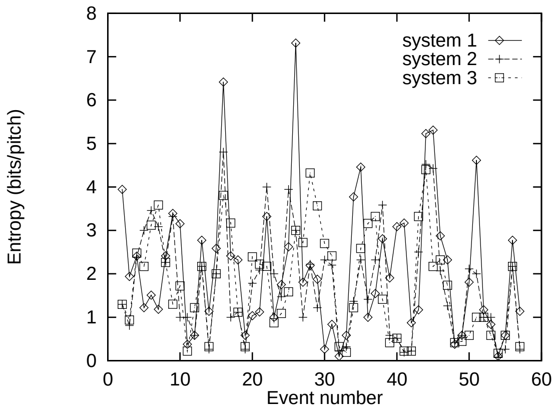






1	best from G.A run 1.	2.1740
2	best from G.A run 2	1.9960
3	best from G.A run 3	2.0288
4	system 1 and Conklin's best	1.9742
5	system 2 and Conklin's best	1.9250
6	system 3 and Conklin's best	1.9432
7	systems 1, 2, and 3	1.8746

As can be seen from the table, two of the three G.A. viewpoints have outperformed the best of Conklin's systems on the test set. One of them (system 10) even has an average entropy of under 2 bits per pitch. All three G.A. viewpoints bring the average entropy under 2 bits when tried in conjunction with system 6. The three G.A. viewpoints tried together have the best result of all - 1.87 bits per pitch.



[5] Goldberg, D. E. *Genetic Algorithms in Search, Optimisation, and Machine Learning.* Addison-Wesley, Reading, MA. (1989).