Charles Hendrix: A Tool for Enhancing Genetic Algorithms for Music Composition with User Preference Modeling

Carmine Iemmino

ACM Reference Format:
Carmine Iemmino. 2024. Charles Hendrix: A Tool for Enhancing Genetic Algorithms for Music Composition with User Preference Modeling. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Artificial intelligence has revolutionized the way we create music by becoming a collaborative partner to artists. AI-powered tools now compose original pieces ranging from simple melodies to complex orchestrations, but more importantly, they inspire musicians by offering novel ideas and breaking through creative blocks. By analyzing vast amounts of musical data to learn underlying patterns and structures across genres, these tools suggest innovative melodies, harmonies, and rhythms that artists might not conceive on their own.
For example, OpenAI's Jukebox generates music based on inputs like genre, artist, and lyrics, allowing artists to co-create and explore new musical concepts with the AI. Similarly, EvoBackMusic combines a neural network with a genetic algorithm to compose background music tailored to user preferences and emotional states, fostering a dynamic partnership between the AI and the user. These tools exemplify how AI serves as a source of inspiration, enhancing the creative process by partnering with artists rather than replacing them. By acting as a creative catalyst, AI is opening up new possibilities in music composition, enabling artists to push the boundaries of their art. Musicians are increasingly embracing AI as a collaborator, integrating AI-generated elements into their work and exploring innovative musical expressions. This collaboration between human creativity and machine intelligence is transforming the musical landscape, fostering a symbiotic relationship that enriches the artistic process.

2 METHODOLOGY

The development of the project was based on extending an existing project that composed simple melodies using a genetic algorithm with a fitness function based on objective musical criteria derived from music theory. However, these criteria were not sufficient to make the music composer satisfactory, resulting in generic compositions that might or might not appeal to the user. Music is highly subjective, and reducing it to adherence to objective musical criteria is limiting. This led to the idea of improving the fitness function—the main driver of the genetic algorithm—so that it would evaluate the generated compositions in a more precise, refined, and especially personal and subjective manner. Here, machine learning comes into play, where the output of a model trained on the user's preferences becomes the output of the fitness function, ensuring that at the end of the algorithm's execution, a composition more aligned with the user's tastes is produced. Below the steps followed to reach the solution

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Enhancing the Complexity of Compositions

To get closer to realistic compositions, the genetic algorithm was made more complex. A harmonic track based on different pre-set progressions was added to the already existing melodic track, and the use of different instruments was provided for both the melodic and harmonic tracks.
To avoid excessive complexity, the compositions consist of only 4 measures, and depending on the BPM, they can vary in duration from 4 to 16 seconds.
The detailed gene space specification, which outlines the structure and parameters of each gene in the genetic algorithm, is provided in Appendix A.

2.2 Searching for Data

To train a machine learning model, it is essential to have data on which it can learn. In this case, since the data concerns the user's musical preferences, an attempt was made to collect them through the Spotify APIs. However, these did not prove effective in providing the necessary data: there is too much difference between a complete song, which includes various instruments and a vocal melody, and the simple compositions generated by the genetic algorithm. Moreover, Spotify presents attributes like "Danceability," and the proprietary algorithm used to calculate it is unknown, making it impossible to evaluate the generated compositions accurately.

2.3 Constructing the Dataset

It was decided to build the dataset from scratch by setting the genetic algorithm in an interactive manner. The output of the fitness function is determined directly by the user, who can rate the composition from 1 to 5. To avoid inconsistent evaluations caused by fatigue and loss of concentration, a maximum of 30-40 compositions were evaluated at a time, reaching a total of approximately 250 compositions.

2.4 Data Feature Engineering

A data feature engineering pipeline was defined with the following steps:

- **Data Cleaning**: Since the dataset was built manually, no cleaning or imputation was necessary.
- **Feature Construction**: Features such as the number of notes, average note duration, average pitch, pitch range, and the proportion of rests to notes were constructed. The BPM

value was divided into categories: slow, medium, fast, very fast, and extremely fast.
- **Feature Scaling**: One-hot encoding was performed on categorical features, and standard scaling was applied to numerical features. The scaler was then saved to correctly transform new instances that will be generated by the genetic algorithm for inference.
- **Feature Selection**: Columns related to individual notes were removed.
- **Data Balancing**: SMOTE was used to balance the dataset, as it lacked instances at the extremes of the user rating scale, specifically ratings of 1 and 5.

2.5 Models Testing and Hyperparameter Optimization

Various models were tested, particularly *RandomForestClassifier*, *RandomForestRegressor*, and *GradientBoostingRegressor*. Each model was evaluated using a 10-fold cross-validation repeated 10 times, and hyperparameter tuning was performed.

2.6 Genetic Algorithm with Inference

The best model identified during the testing phase is used in the new fitness function to predict the fitness value of each new composition generated by the genetic algorithm.

3 IMPLEMENTATION

The implementation required the use of several libraries, the most important ones are:

- **PyGad**, used to define genetic algorithms for various purposes, the main one being the generation of musical compositions.
- **Music21**, used to convert the chromosomes of the genetic algorithm into reproducible MIDI musical compositions.
- **Pandas**, used for dataset manipulation, particularly in the data feature engineering phase.
- **Scikit-learn**, used for machine learning models, model evaluation metrics, and K-fold splitting.
- **Joblib**, used for saving and loading trained models.
- **SHAP**, used to create a global surrogate model for explainability.
- **LIME**, used to create a local surrogate model for explainability.
- **CodeCarbon**, used to track the system's energy consumption.

The system features two main scripts: one that tests which model is the best along with hyperparameter optimization, and another that executes the genetic algorithm for music composition.
The first script tests the models *RandomForestClassifier*, *RandomForestRegressor*, and *GradientBoostingRegressor*. Each model is evaluated using 10 times repeated 10-fold cross-validation. Hyperparameter optimization for each model is performed using a genetic algorithm, which is more effective than grid search and random search. Depending on a flag, the optimization can occur before cross-validation on a common 80/20 train-test split or during each internal iteration of the cross-validation, after which the continuous hyperparameters are averaged, and the mode is taken for categorical ones. Finally, the best model among these three is trained with the entire dataset and saved locally to be used for future inference.
The second script is the actual generation of musical compositions through a genetic algorithm. Depending on a flag, this script can be used for data collection or to generate compositions independently and autonomously using the trained model. In the latter case, there is an active learning component: when the best final composition generated by the genetic algorithm with inference is played and its fitness value is displayed, the user is asked to rate the composition from 1 to 5. The new composition is then added to the dataset with its evaluation, and upon reaching 50 new compositions, a new testing of the different models on the updated dataset is automatically performed, updating the saved model.

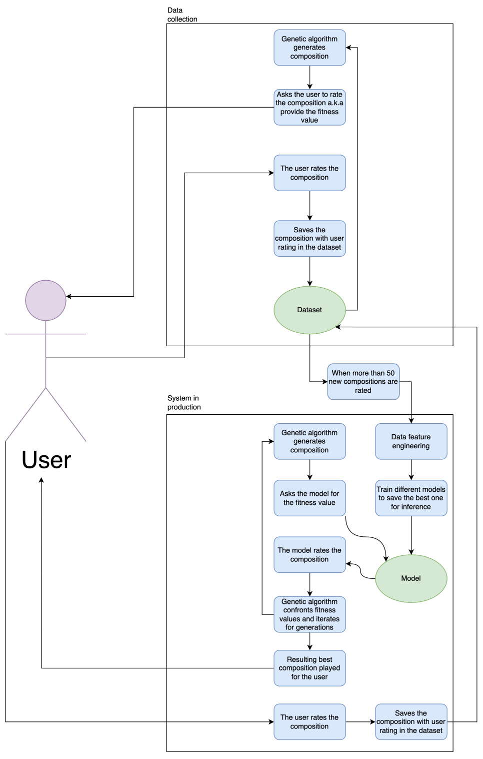
Data collection
Genetic algorithm generates composition
Asks the user to rate the composition a.k.a provide the fitness value
The user rates the composition
Saves the composition with user rating in the dataset
Dataset
When more than 50 new compositions are rated
System in production
Genetic algorithm generates composition
Data feature engineering
Asks the model for the fitness value
Train different models to save the best one for inference
The model rates the composition
Model
Genetic algorithm confronts fitness values and iterates for generations
Resulting best composition played for the user
The user rates the composition
Saves the composition with user rating in the dataset
User

Figure 1: Implementation

4 QUALITY ASSURANCE

The main threat to the model's quality is the small size of the dataset, around 250 compositions rated. For this reason, as already mentioned in the methodology and implementation, several tests were conducted for each model using 10-times repeated 10-fold cross-validation with parameter optimization for each internal iteration of the cross-validation, in order to achieve the best possible configuration.
The metrics used to compare the models are:

- **MAE (Mean Absolute Error)**: This measures the average magnitude of errors between predicted and actual values. It is calculated as the average of the absolute differences between predicted and actual values. Lower MAE values indicate better model performance, as predictions are closer to actual values on average. MAE treats all errors equally, regardless of their direction or magnitude. The MAE formula is

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

where y_i represents the actual value, $\hat{y}_i$ is the predicted value, and n is the total number of observations.

- **QWK (Quadratic Weighted Kappa)**: This is a statistic that measures the agreement between two raters (or a model and true labels), adjusted for agreement occurring by chance, considering the ordinal nature of the ratings. It takes into account the differences between predicted and actual ratings and assigns higher penalties for larger discrepancies. QWK ranges from -1 (complete disagreement) to 1 (perfect agreement). QWK is more sensitive to the magnitude of disagreements, especially larger discrepancies due to the quadratic weighting. The QWK formula is

$$\text{QWK} = 1 - \frac{\sum_{i,j} \omega_{i,j} O_{i,j}}{\sum_{i,j} \omega_{i,j} E_{i,j}}$$

where:
- $O_{i,j}$ is the observed confusion matrix between predicted and actual classes. - $E_{i,j}$ is the expected confusion matrix, calculated under the assumption of random predictions. - $\omega_{i,j}$ is a penalty factor that increases with the distance between i and j

Using both metrics allows for a more complementary approach. MAE provides a direct interpretation of how far the predictions are from the actual values on average, while QWK penalizes predictions that deviate significantly from the actual values due to its quadratic weighting. Moreover, unlike specific metrics like R^2, precision, and recall, they can be used interchangeably in both classification and regression. Only in the case of regression, for the calculation of QWK, it is necessary to round the predicted values to integers.

For each internal iteration of the cross-validation, MAE and QWK were calculated; for each repetition of the cross-validation, the mean was calculated, and at the end of the 10 repetitions, a final mean of the means was calculated. A similar approach was applied for hyperparameter optimization, where means were calculated for continuous parameters and modes for categorical ones. The genetic algorithm for hyperparameters was defined as a multi-objective optimization problem of MAE and QWK.

An important consideration in the modeling process was the choice between regression and classification models for predicting user preferences. Despite classifiers exhibiting slightly better performance metrics, regression models were preferred for integration into the genetic algorithm's fitness function. Regression models provide continuous output values, offering nuanced fitness scores that enable the genetic algorithm to better distinguish between compositions. This continuous feedback enhances the algorithm's ability to evolve solutions that more closely align with user preferences.

In contrast, classification models yield discrete class labels, which may not capture subtle differences between compositions. This lack of granularity can limit the genetic algorithm's effectiveness by reducing its capacity to fine-tune solutions based on slight variations in user preferences.

An informal evaluation of the model was conducted in the field by running the genetic algorithm in inference mode and comparing the predicted score of the final solution with the score that the user would have actually assigned to that composition. The results were overall positive, with the model capable of satisfactorily approximating the user's preferences. This evaluation, however, presents a possible threat to validity, as the user responsible for the evaluation is the same person who developed the system, and it is understandably difficult to be completely impartial in judging the system's performance.

Model	MAE	QWK	Best Parameters
RandomForestClassifier	0.90	0.49	{'n_estimators': 117, 'criterion': 'entropy', 'max_depth': 18, 'min_samples_split': 6, 'min_samples_leaf': 2, 'max_features': 'log2', 'bootstrap': False}
RandomForestRegressor	0.90	0.35	{'n_estimators': 121, 'criterion': 'squared_error', 'max_depth': 18, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'bootstrap': False}
GradientBoostingRegressor	0.95	0.38	{'loss': 'squared_error', 'learning_rate': 0.17618, 'n_estimators': 270, 'subsample': 0.7651, 'criterion': 'friedman_mse', 'min_samples_split': 6, 'min_samples_leaf': 5, 'max_depth': 6, 'max_features': 'sqrt', 'alpha': 0.529}

Table 1: Comparison of MAE, QWK, and Best Parameters for Different Models

5 EXPLAINABILITY

5.1 Global surrogate

A global surrogate provides explainability for complex "black-box" models by approximating them with a simpler, interpretable model. The simpler model in this case is a *DecisionTreeRegressor* rather than the more complex *RandomForestRegressor* actually used. To avoid training and testing on the same data, cross-validation is used to produce predictions for each sample in the dataset. In this process, the black-box model is trained on k-1 folds and used to predict on the k -th fold, iterating this for all folds. This generates "out-of-fold" predictions, providing a comprehensive set of predictions for the entire dataset without data leakage from using the same samples for training and testing. The interpretable model is then trained on the entire dataset with these out-of-fold predictions, aiming to approximate the black-box model's behavior. The surrogate model's R^2 score is checked against the black-box model's out-of-fold predictions. A high R^2 indicates that the surrogate model closely replicates the black-box model's predictions. Using SHAP values, feature importance in the surrogate model can be interpreted, reflecting feature importance in the black-box model. This helps to understand which features drive predictions in the black-box model. The R^2 obtained is 0.44, so not very high, but still useful to get a rough understanding of the importance of each feature. This SHAP summary plot shows that `avg_note_duration`, `std_note_duration`, and num_notes_played are the most influential features in predicting user preferences for music compositions (rated from 1 to 5). Higher avg_note_duration tends to push predictions toward higher ratings, indicating users favor compositions with longer average note durations. Lower std_note_duration (more consistent note lengths) also positively impacts user ratings, suggesting a preference for rhythmic consistency. The num_notes_played feature affects predictions, though without a clear directional trend. Other features, such as Chord_Instrument_Piano, show minor positive influences, while various chords and scales have minimal impact, with SHAP values clustering around zero. Overall, the model prioritizes note duration and consistency over specific chords or scales in predicting user preferences.

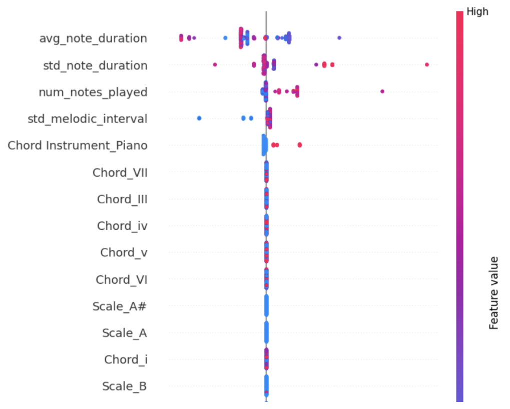

Figure 2: SHAP summary plot

5.2 Local surrogate

A local surrogate provides single-instance explanations, offering insights into why the model assigned a particular rating to an individual music composition. For this purpose, LIME is used, which fits a simpler, interpretable model (such as a linear model) around the prediction of a specific instance, making it possible to understand the local behavior of the black-box model.
In this case, for each final output from the genetic algorithm, which generates music compositions, LIME helps identify the top contributing features that led to the model's predicted user preference score. Focusing on individual predictions allows for a detailed view of how specific musical attributes influence the rating for each composition. This provides not only an interpretability layer over the algorithm's final output but also actionable insights for further improving the composition process based on feature importance.
This approach ensures that each composition generated by the genetic algorithm comes with an interpretable, feature-level breakdown of the predicted user preference score, aligning music generation with user feedback and making it easier to iterate on and improve the composition strategy.

6 SUSTAINABILITY

The tracking of the system's energy consumption is present in both main scripts. Regarding model testing, due to the high number of cross-validations performed (10 times repeated * 10-fold cross-validation * 3 models = 300), with hyperparameter optimization for each fold, there is a significant energy consumption.
These results were obtained using *CodeCarbon*.

Script	Duration	CO_2 (g)	Energy consumption (kWh)
Test models optimization first	1 min 26 sec	0.3	0.0012
Test models optimization in cross-validation	18 min 58 sec	5.0	0.0153
Genetic algorithm with inference	4 min 14 sec	1.1	0.0034

Table 2: Comparison of testing models and genetic algorithm with inference

7 CONCLUSIONS

This project has explored the potential of combining machine learning and genetic algorithms to create a music composition tool aimed at aligning output with individual user preferences. By replacing a fixed fitness function with a model trained on user-specific feedback, this project aimed to shift from purely objective musical criteria to a more subjective, user-oriented composition process. However, the resulting compositions remain simple and limited in their scope due to the MIDI structure and basic melodic-harmonic complexity. While the model shows some promise in approximating a specific user's musical tastes, these compositions are not robust or diverse enough to be compelling for broader audiences or varied musical applications. Several significant limitations constrain the project's practical impact and generalizability. The MIDI-based structure of the compositions restricts musical richness, resulting in pieces that lack the layered complexity typical of real-world music. Moreover, the algorithm's reliance on a single user's feedback for training creates a system that is tightly tailored to one set of preferences, making it largely unusable for others without extensive retraining. Scalability is a critical challenge, as collecting and curating a personalized dataset for each user would require considerable time and effort, rendering the system impractical for wider adoption. Given these limitations, future work would likely need to pivot towards addressing these fundamental issues. One avenue could involve diversifying the musical structure by incorporating more complex instruments and arrangements, moving beyond MIDI limitations to achieve compositions that are closer to real-world music. To address scalability, collaborative filtering or preference clustering could be explored to develop generalized models that cater to groups of users with similar musical tastes, thus reducing the need for personalized training from scratch. However, the limitations in real-world usability and personalization remain, suggesting that while this approach is technically interesting, its practical applications may be limited without substantial rethinking of the framework.

A GENE SPACE SPECIFICATION

This appendix provides a detailed specification of the gene space used in the genetic algorithm to enhance the complexity and realism of musical compositions.

- **Gene 1**: Scale
  An integer between 0 and 23, representing one of the 24 possible scales (12 major and 12 minor).
- **Gene 2**: Tempo (BPM)
  An integer specifying the tempo of the composition, ranging from 75 to 215 beats per minute.
- **Gene 3**: Melody Instrument
  An integer between 0 and 7, representing one of 8 possible instruments for the melody (e.g., *Piano, Violin, Flute, Saxophone*, etc.).
- **Gene 4**: Chord Instrument
  An integer between 0 and 4, representing one of 5 options for the chord instrument (e.g., *Piano, Acoustic Guitar, Electric Guitar, Organ, Sitar*).
- **Gene 5**: Chord Progression
  An integer between 0 and 7, specifying one of 8 predefined chord progressions suited to both major and minor scales.
- **Genes 6 to 101**: Melody Notes
  A sequence of 32 notes, where each note is represented by a set of three genes:
  - **Scale Degree**: An integer between 0 and 8, defining the position of the note within the scale.
  - **Octave**: An integer, either 4 or 5, specifying the octave of the note.
  - **Duration**: An integer representing one of 10 possible durations, mapped to different note lengths (e.g., 0.25 for sixteenth notes, 0.5 for eighth notes, 1.0 for quarter notes).

The genespace structure, totaling 101 genes, enables the genetic algorithm to control various aspects of the composition, including tonality, rhythm, instrument choice, and melody-harmony interactions. This design enhances the adaptability of compositions to match user preferences while keeping the complexity manageable.