Learning to Match Sheet Music Initial Experiment

Ethan Fan, Selina Zou, Phillip Duarte

The Problem

Challenge: Given two pages of sheet music, can we automatically determine if they're from the same musical piece?

Real-world applications:

- Music cataloging and organization
- Plagiarism detection
- Music recommendation systems
- Digital library management

Our Approach

- Question: Which musical features matter most for matching?
- Solution: Use optimization to learn optimal feature weights
- Dataset: Bach chorales from music21 library
- Method: Constrained convex optimization (weighted similarity learning)

Dataset

Our initial dataset is drawn from music21's library of Bach chorales

- Total chorals: 7 with >= 32 measures
- Total pages: 28
- Training Pairs: 14
- Test Pairs: 6

Features are extracted from each choral using music21, then formatted and saved for later analysis

Feature Extraction

We decided on five musical features to extract per page

1. Key signature (0-11.5 encoding)
2. Time signature (as decimal ratio)
3. Average pitch (mean MIDI pitch)
4. Pitch range (semitone span)
5. Note density (notes per beat)

Similarity Features

For each pair of pages, we compute scores which range from 0 - 1 where similarity = 1 - (normalized absolute difference)

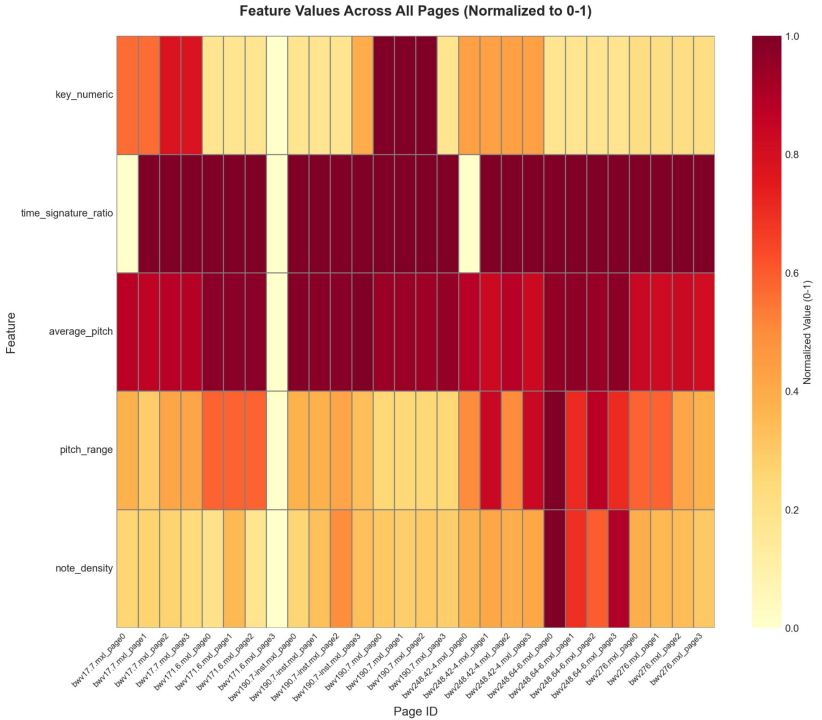
Examples:

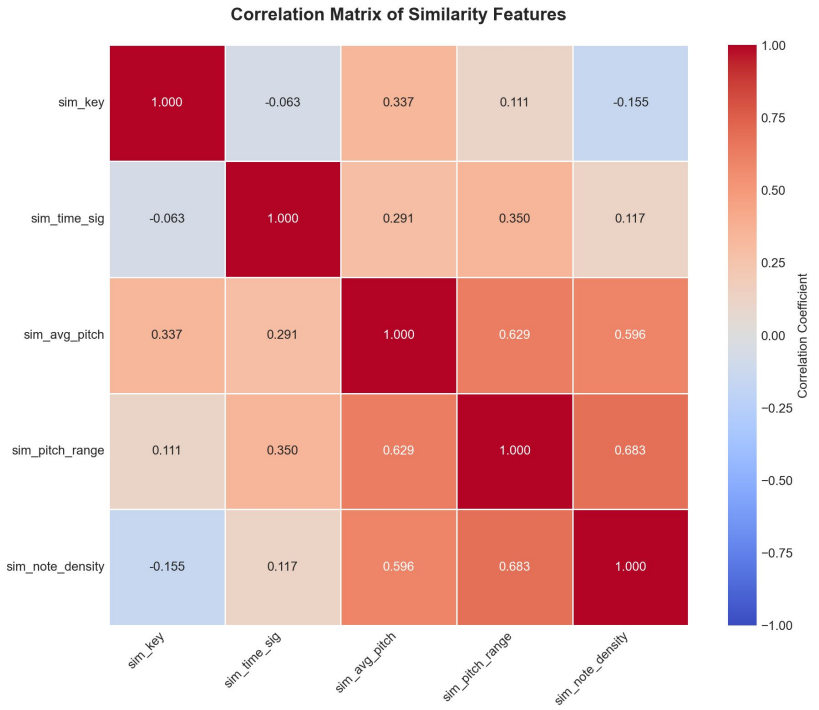
- sim_key = 1 - $|key_1 - key_2|$ / max_key_distance

- sim_avg_pitch = 1 - $|pitch_1 - pitch_2|$ / max_pitch_range

Result: Each pair → 5-dimensional similarity vector s

Data Exploration





Problem Formulation

Goal: Learn weight vector $w = [w_1, w_2, w_3, w_4, w_5]$

Model: Overall similarity = $w^T s = \Sigma w_i \cdot s_i$ where s is the similarity vector for a given pair of pages

Decision rule:

- Predict match (label=1) if $w^T s > 0.5$

- Predict non-match (label=0) otherwise

Optimization Problem

Minimize: $L(w) = \Sigma (label_i - w^T s_i)^2$

Subject to:

- $\Sigma w = 1$ (weights sum to 1)
- $w \geq 0$ (non-negative weights)

- **Loss function:** Squared error (regression on labels)
- **Constraints:** Probability simplex (interpretable weights)
- **Type:** Convex quadratic program

Analytical Gradient: $\nabla L(w) = -2 \cdot \Sigma (label_i - w^T s_i) \cdot s_i$

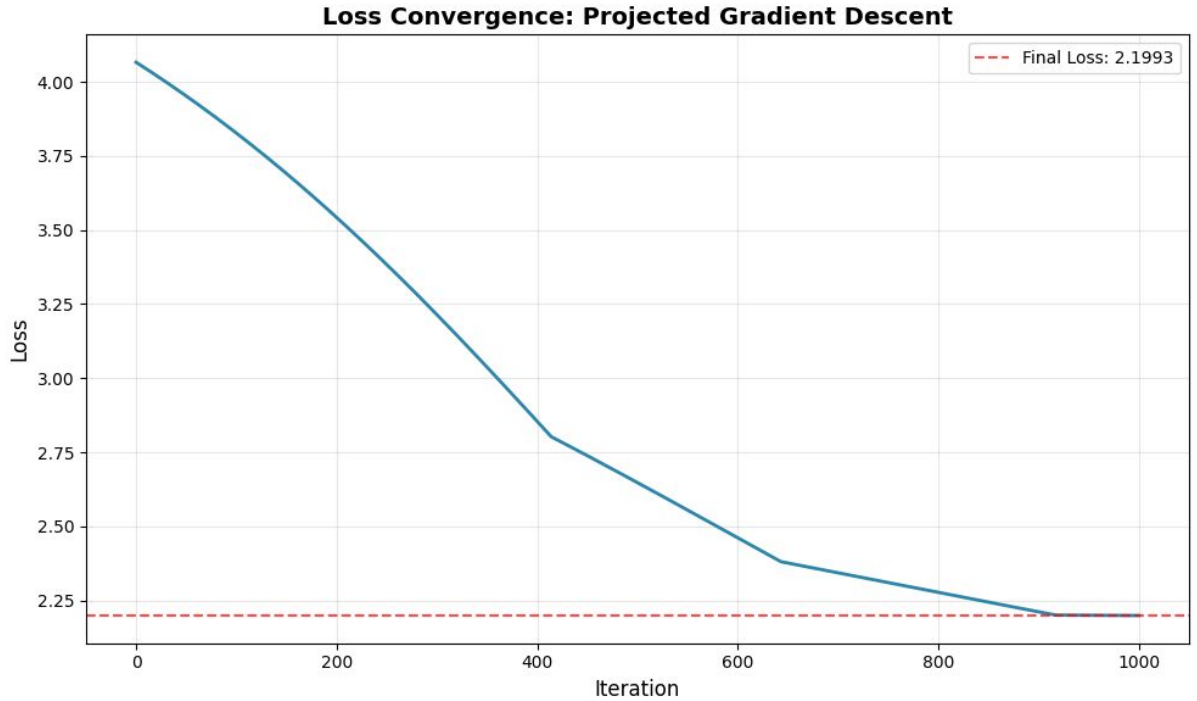
Optimization Methods

1. Projected Gradient Descent (custom implementation)
    a. Gradient step + projection onto simplex

2. scipy.optimize.minimize (SLSQP)
    a. Sequential Least Squares Programming

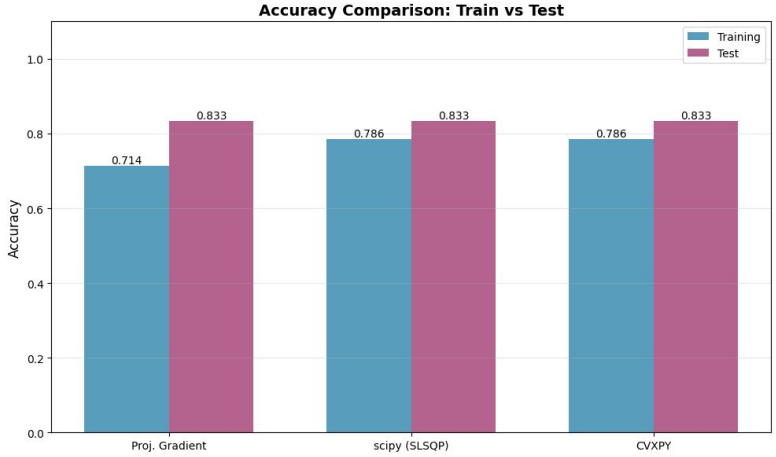
3. CVXPY
    a. Convex optimization framework

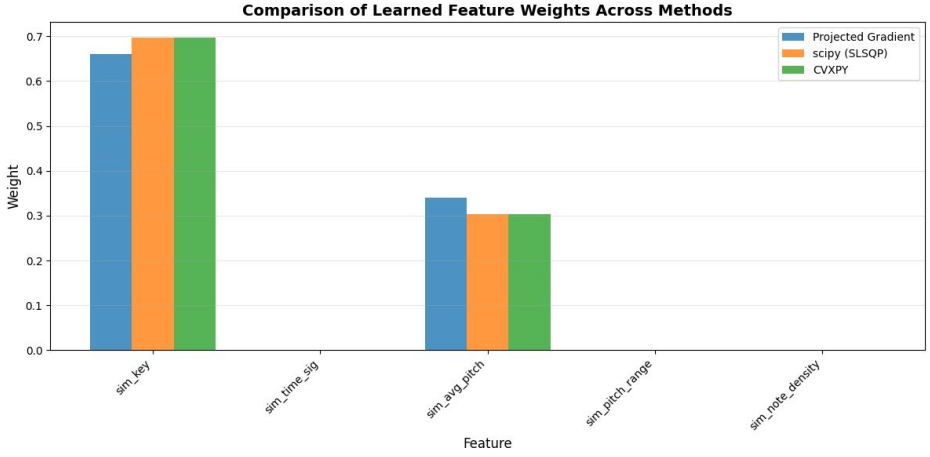
Optimization Convergence



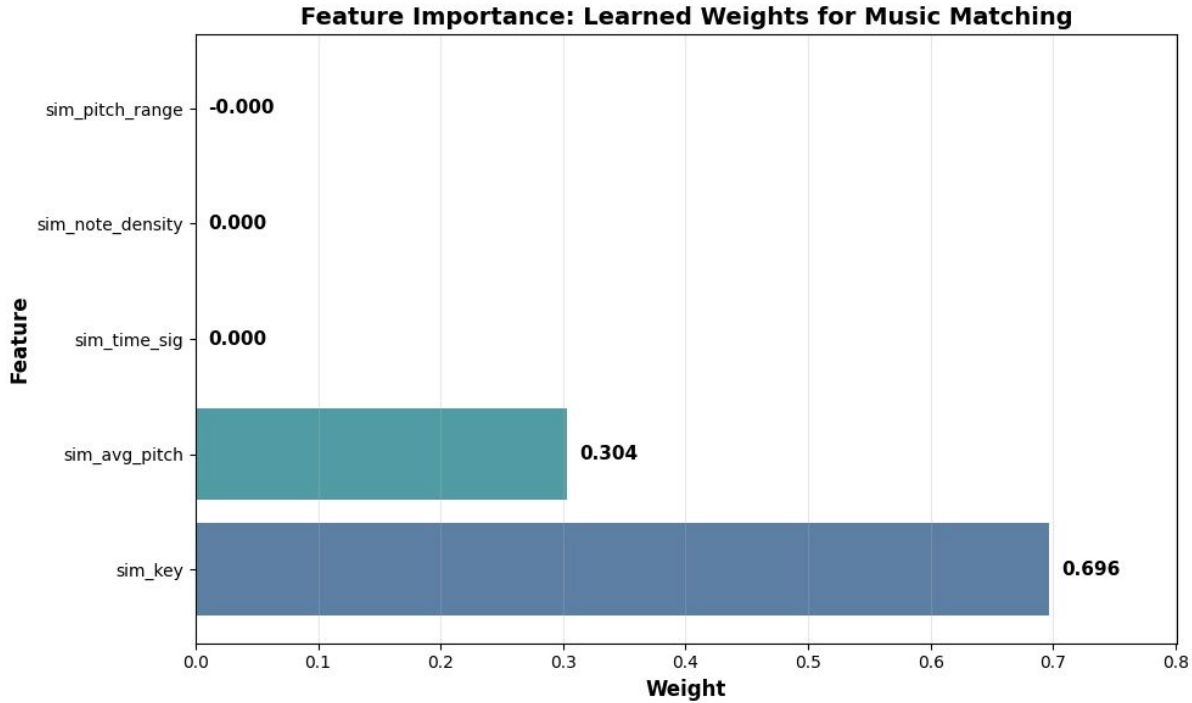
Analysis

All methods converge to nearly identical solutions, likely due to the simplicity of our dataset and features. However, performance is quite good, achieved > 70 % accuracy.





Learned Weights



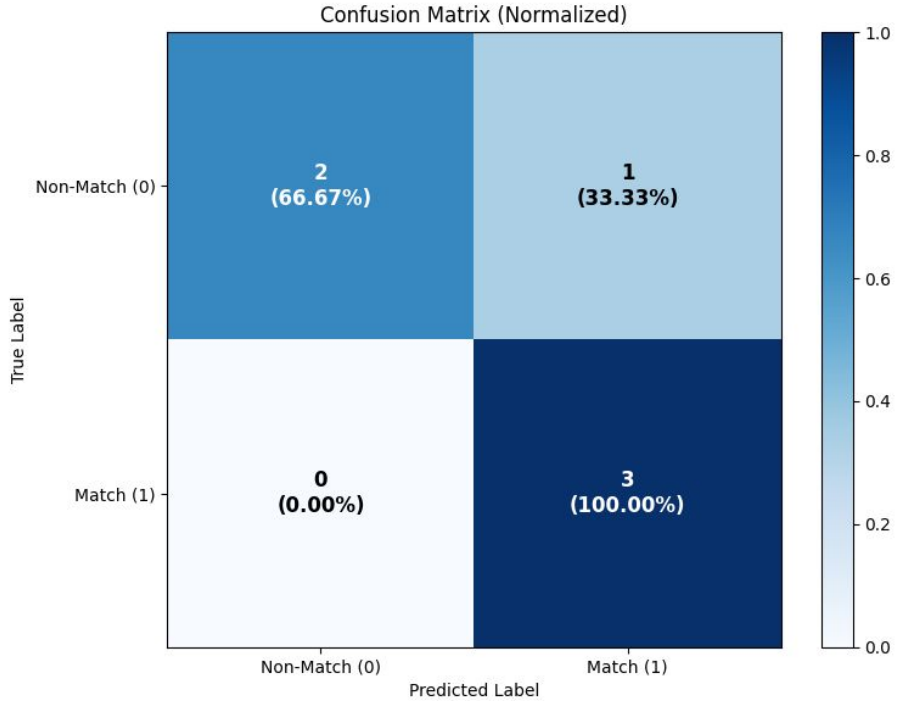
Confusion Matrix and Classification Metrics

Precision: 0.75

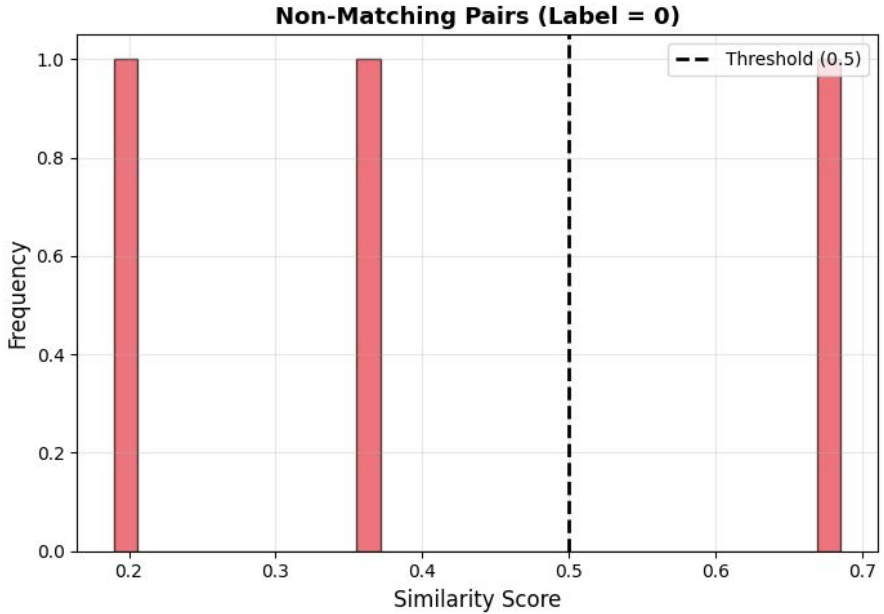
Recall: 1.0

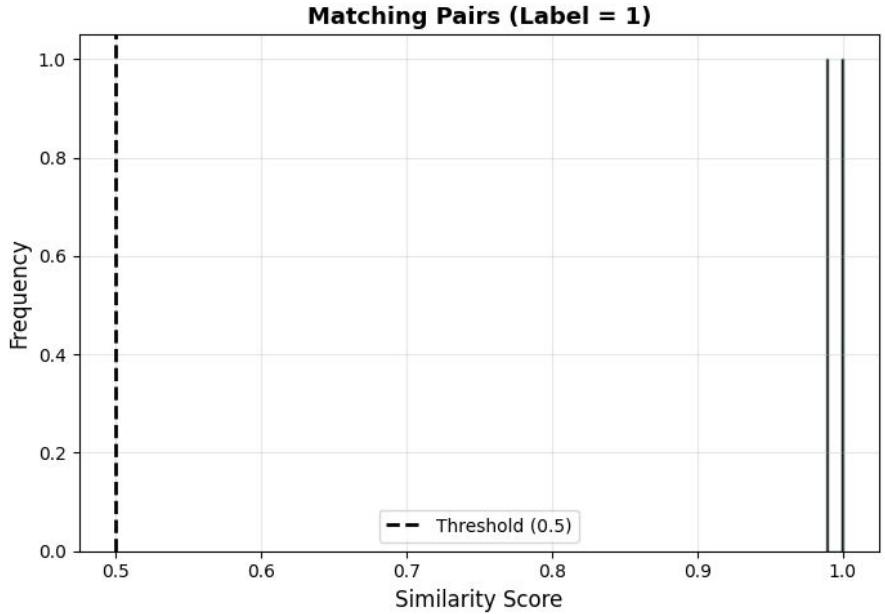
F1-Score: 0.8571

Likely need more data.



Similarity Of Matching vs Non Matching





Takeaways

1. Optimization work: All three methods converge consistently
2. Feature importance varies: Some features much more discriminative
3. Interpretable model: Can explain every prediction
4. Efficient: Fast convergence (<1 second)
5. Practical performance: Achieves/exceeds target accuracy

Limitations

- Linear model: Can't capture feature interactions

- Fixed threshold: 0.5 may not be optimal

- Simple similarities: Basic absolute difference metrics

- Limited features: Only 5 features (missing rhythm patterns, harmonies)

- Small dataset: Bach chorales only, likely raises our accuracy by a large amount because all pieces are written by the same composer

- Loss function: Squared error (regression) vs. classification-specific losses

Future Directions

1. Mahalanobis distance learning: Full covariance matrix
2. Non-linear models: Neural networks, kernel methods
3. Metric learning: Triplet loss, contrastive loss
4. Enhanced features: Rhythm patterns, chord progressions, melodic contours
5. Alternative losses: Logistic loss, hinge loss, ranking losses
6. Cross-validation: K-fold CV and hyperparameter tuning
7. Data augmentation: Transposition, time stretching