# 代码深度研究报告



生成时间：2025-08-19 14:58:21｜本报告由 GitHave AI 生成，仅用于研究目的，不构成任何形式的法律建议或保证，不承担因使用本报告而导致的任何损失或损害。

# 一、前言

* **仓库名称**：chordPrediction
* **仓库描述**：和弦预测
* **仓库分支**：master
* **仓库地址**：<https://github.com/kinglegendzzh/chordPrediction>
* **分析路径**：[/Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py) (单项文件: service/numpyMarkov.py)
* **项目总结**：

ChordPrediction 是一个基于马尔科夫链的智能音乐创作工具，通过分析历史和当前的和弦序列，实时预测和生成高质量的和弦序列。它支持多种和弦序列训练，增强不同音乐风格之间的过渡多样性，并提供交互界面进行和弦标注、保存和预览。项目采用模块化设计，便于扩展和定制，适用于音乐创作、音乐教育和音乐游戏等领域。

* **当前分析的service/numpyMarkov.py 文件总结**：

`numpyMarkov.py` 是一个用于音乐和弦序列预测和分析的Python模块，通过马尔可夫链模型实现和弦转换概率的预测和可视化。该模块包含初始化马尔可夫链、构建概率转移矩阵、预测下一个和弦以及可视化转移矩阵等功能，采用面向对象设计，结构清晰，易于维护和扩展。

# 二、项目描述

# ChordPrediction - 智能音乐创作工具

## 项目背景与目标

ChordPrediction 是一个基于马尔科夫链的和弦预测算法的智能音乐创作工具。它旨在帮助用户生成高质量的和弦序列，提高音乐创作的效率和多样性。该工具适用于音乐创作、音乐教育、音乐游戏等多个领域，特别是对于希望快速生成和弦序列的音乐爱好者和专业人士。

## 核心功能与特色亮点

### 主要功能

1. **和弦预测**：通过分析历史和当前的和弦序列，实时预测和生成和弦。
2. **多种和弦序列训练**：支持多种和弦序列训练，以增强不同音乐风格之间的过渡多样性。
3. **交互界面**：提供一个交互界面，允许用户通过简单的演奏生成所需的和弦，并进行标注、保存、预览和模型训练。

### 关键模块

* **sounds**：管理音频相关的资源和功能，包括音频文件的加载、播放和管理。
* **utils**：提供一系列通用的辅助工具类和函数，简化代码的编写和维护。
* **labels**：定义和管理不同音乐风格和模式的标签和模型。
* **models**：存储与处理压弦键映射相关的数据缓存。
* **records**：存储和管理各种音乐相关的数据和模式。
* **service**：处理音乐相关的服务和功能，包括 MIDI 输入管理、和弦序列预测、音阶匹配等。
* **VirtualKeyboard.py**：创建一个智能化音乐创作工具的用户界面，支持 MIDI 输入设备或键盘映射。
* **MatrixView.py**：创建一个图形用户界面，用于显示概率转移矩阵的和弦分布图。

## 项目价值与作用

ChordPrediction 通过智能化的和弦预测算法，为音乐创作提供了强大的支持。它不仅能够帮助用户快速生成高质量的和弦序列，还能够通过多种和弦序列训练，增强不同音乐风格之间的过渡多样性。此外，交互界面的设计使得用户可以方便地进行和弦标注、保存和预览，进一步提高了音乐创作的效率和质量。

## 项目架构设计

ChordPrediction 采用模块化设计，各个模块协同工作，共同实现音乐创作工具的功能。核心模块包括：

* **sounds**：负责音频资源的加载、播放和管理。
* **utils**：提供通用的辅助工具类和函数，提高代码的复用性和可维护性。
* **labels**：定义和管理音乐风格和模式的标签和模型。
* **models**：存储和处理压弦键映射相关的数据缓存。
* **records**：存储和管理各种音乐相关的数据和模式。
* **service**：处理音乐相关的服务和功能，包括 MIDI 输入管理、和弦序列预测、音阶匹配等。
* **VirtualKeyboard.py**：创建用户界面，支持 MIDI 输入设备或键盘映射。
* **MatrixView.py**：创建图形用户界面，用于显示概率转移矩阵的和弦分布图。

## 技术栈与依赖

ChordPrediction 项目使用了以下主要技术栈和依赖：

* **Python**：作为主要编程语言。
* **Pygame**：用于创建用户界面和处理 MIDI 输入设备。
* **PyInstaller**：用于打包 Python 应用程序，生成可执行文件。
* **MIDI 输入设备**：支持 MIDI 输入，用于实时生成和弦序列。

## 使用方式与扩展性

ChordPrediction 提供了一个交互界面，用户可以通过简单的演奏生成所需的和弦，并进行标注、保存、预览和模型训练。此外，项目采用模块化设计，便于扩展和定制。用户可以根据需要添加新的功能模块或修改现有模块，以满足特定的业务需求。

## 安全性与性能

ChordPrediction 项目在设计时注重安全性与性能。通过模块化设计和优化数据访问机制，确保了应用程序的高效运行。同时，项目遵循 GNU Lesser General Public License（LGPL）许可证，保护用户的自由使用和修改软件库的权利。

## 总结

ChordPrediction 是一个基于马尔科夫链的和弦预测算法的智能音乐创作工具。它通过智能化的和弦预测算法，为音乐创作提供了强大的支持。项目采用模块化设计，各个模块协同工作，共同实现音乐创作工具的功能。通过使用 Pygame 和 PyInstaller 等工具，项目能够生成可执行文件，便于在不同环境中部署和运行。

# 三、分析摘要

## 代码结构概览

service/numpyMarkov.py 是 ChordPrediction 项目中的一个重要模块，主要负责音乐和弦序列的预测和分析。该模块通过马尔可夫链模型实现和弦转换概率的预测和可视化。模块中定义了多个函数，包括初始化马尔可夫链、构建概率转移矩阵、预测下一个和弦以及可视化转移矩阵等功能。

### 模块组成及关系

* service/numpyMarkov.py：包含所有与马尔可夫链相关的函数和类。
* 依赖关系：service/numpyMarkov.py 依赖于 numpy 库进行矩阵运算和概率计算。

### 调用依赖关系

* \_\_init\_\_：初始化马尔可夫链模型，调用 index\_chords 和 build\_transition\_matrix。
* index\_chords：为和弦序列中的每个和弦分配一个唯一的索引。
* build\_transition\_matrix：根据输入的和弦序列构建概率转移矩阵。
* predict\_chord：根据当前和弦预测下一个和弦。
* visualize\_transition\_matrix：可视化概率转移矩阵。

## 核心模块和函数

### \_\_init\_\_

* **作用**：初始化马尔可夫链模型，设置马尔可夫链的阶数，构建和弦索引字典，并初始化概率转移矩阵。
* **重要性**：作为模块的入口，初始化了整个马尔可夫链模型，是其他函数的基础。
* **调用关系**：被 service/numpyMarkov.py 中的其他函数调用。

### build\_transition\_matrix

* **作用**：根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。
* **重要性**：构建了马尔可夫链的核心数据结构，是预测和弦转换概率的基础。
* **调用关系**：被 \_\_init\_\_ 调用。

### predict\_chord

* **作用**：根据当前和弦预测下一个和弦，并过滤出概率大于阈值的预测结果。
* **重要性**：实现了和弦预测的核心逻辑，是用户交互的核心功能。
* **调用关系**：被 service/numpyMarkov.py 中的其他函数调用。

### visualize\_transition\_matrix

* **作用**：可视化概率转移矩阵，并将其以热图的形式输出为图片。
* **重要性**：提供了可视化工具，帮助用户理解和弦转换的概率分布。
* **调用关系**：被 service/numpyMarkov.py 中的其他函数调用。

## 代码设计风格分析

### 命名规范

* 变量和函数名使用小写字母和下划线，符合Python的命名规范。
* 类名使用驼峰命名法，符合Python的类命名规范。

### 一致性

* 代码风格一致，使用了 numpy 库进行矩阵运算和概率计算。
* 函数参数和返回值的命名和类型一致，便于理解和维护。

### 封装与抽象程度

* 代码封装良好，每个函数都有明确的职责，例如 \_\_init\_\_ 初始化模型，build\_transition\_matrix 构建转移矩阵等。
* 抽象程度较高，通过函数封装了复杂的逻辑，便于调用和复用。

### 模块职责划分

* 模块职责明确，service/numpyMarkov.py 专注于马尔可夫链模型的构建和预测。
* 模块间依赖关系清晰，每个模块都有明确的职责和调用关系。

## 潜在问题

### 资源释放不当

* 代码中未涉及资源释放操作，如果在实际应用中使用大量内存或文件资源，可能会导致资源泄漏。

### 异常未处理

* 代码中未对异常进行处理，例如在 predict\_chord 中，如果当前和弦数量少于马尔科夫链的阶数，会抛出 ValueError，但未进行捕获和处理。

### 重复或冗余的逻辑

* 代码中未发现明显的重复或冗余逻辑，但可以进一步优化，例如将重复的代码块提取为函数。

### 低效的实现

* 代码中未发现明显的低效实现，但可以进一步优化，例如使用更高效的算法或数据结构。

## 重构建议

### 函数职责过于复杂

* predict\_chord 函数较为复杂，可以将其拆分为多个子函数，例如 filter\_chords 和 normalize\_probabilities，以提高代码的可读性和可维护性。

### 模块边界是否清晰

* 模块边界清晰，每个模块都有明确的职责和调用关系，但可以进一步优化模块间的交互，例如通过接口进行通信。

### 代码是否存在重复可以抽取

* 代码中未发现明显的重复代码，但可以进一步优化，例如将重复的代码块提取为函数。

### 公用模块是否可以拆分

* 代码中未发现明显的公用模块，但可以进一步优化，例如将常用的工具函数提取为公用模块。

## 测试情况

* 代码中未发现测试代码，建议添加单元测试和集成测试，确保主要功能的正确性和健壮性。
* 测试应覆盖主要功能，包括和弦预测、转移矩阵构建和可视化等。
* 测试应考虑边界情况和异常情况，确保代码的健壮性。

## 总结

service/numpyMarkov.py 是 ChordPrediction 项目中的一个重要模块，通过马尔可夫链模型实现和弦转换概率的预测和可视化。模块设计合理，代码风格一致，封装良好，模块职责明确。但存在一些潜在问题，如异常未处理和资源释放不当，建议进行重构和优化。

# 四、模块明细

以下是chordPrediction项目中service/numpyMarkov.py子模块的主要模块信息：

## service/numpyMarkov.py

| 模块名称 | 类型 | 完整路径 | 函数数量 |
| --- | --- | --- | --- |
| numpyMarkov.py | 文件 | /Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py | 7 |

### numpyMarkov.py

**基本信息：**

* **类型：** 文件
* **路径：** service/numpyMarkov.py
* **父路径：** service
* **函数数量：** 7
* **初次分析时间：** 2025-08-19T14:40:17.371523+08:00
* **更新时间：** 2025-08-19T14:40:50.634274+08:00

**模块描述：**

该文件 `numpyMarkov.py` 主要用于实现和弦序列的预测和分析，通过马尔可夫链模型来预测和弦的转换概率。文件中实现的核心功能包括初始化马尔可夫链模型、构建概率转移矩阵、预测下一个和弦以及可视化转移矩阵。关键组件包括和弦索引字典、概率转移矩阵和马尔可夫链的阶数。该文件在项目中起到了关键作用，特别是在音乐分析和和弦预测领域，通过提供准确的和弦转换概率，支持音乐创作和分析工具的开发。设计模式上，该文件采用了面向对象的设计，通过类和方法封装了马尔可夫链模型的各个功能，使得代码结构清晰，易于维护和扩展。

# 五、函数明细

## 重点信息统计

### 前五个重点关注文件（按重要性排序）

1. service/numpyMarkov.py (重要性得分: 12.434)

### 函数统计维度分析(排名前五)

| 统计维度 | 数值 | 函数名称 | 文件路径 |
| --- | --- | --- | --- |
| 被引用次数 | 2次 | state\_to\_index | service/numpyMarkov.py |
| 被引用次数 | 1次 | build\_transition\_matrix | service/numpyMarkov.py |
| 被引用次数 | 1次 | index\_chords | service/numpyMarkov.py |
| 扇出数 | 2次 | **init** | service/numpyMarkov.py |
| 扇出数 | 1次 | build\_transition\_matrix | service/numpyMarkov.py |
| 扇出数 | 1次 | predict\_chord | service/numpyMarkov.py |
| 复杂度 | 27 | build\_transition\_matrix | service/numpyMarkov.py |
| 复杂度 | 26 | predict\_chord | service/numpyMarkov.py |
| 复杂度 | 19 | visualize\_transition\_matrix | service/numpyMarkov.py |
| 复杂度 | 16 | state\_to\_index | service/numpyMarkov.py |
| 复杂度 | 14 | **init** | service/numpyMarkov.py |
| 调用链深度 | 3层 | **init** | service/numpyMarkov.py |
| 调用链深度 | 2层 | build\_transition\_matrix | service/numpyMarkov.py |
| 调用链深度 | 2层 | predict\_chord | service/numpyMarkov.py |
| 调用链深度 | 1层 | visualize\_transition\_matrix | service/numpyMarkov.py |
| 调用链深度 | 1层 | state\_to\_index | service/numpyMarkov.py |

**上述统计中涉及的函数：**

**build\_transition\_matrix** [service/numpyMarkov.py](#servicenumpymarkov.py): 该函数用于根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。

**index\_chords** [service/numpyMarkov.py](#servicenumpymarkov.py): 该函数用于为和弦序列中的每个和弦分配一个唯一的索引，并返回一个和弦到索引的映射字典。同时，它还为未知和弦分配了一个独特的索引。

**init** [service/numpyMarkov.py](#servicenumpymarkov.py): 该类用于初始化一个马尔可夫链模型，用于预测和弦序列中的下一个和弦。它通过为每个唯一和弦分配一个索引，并构建一个概率转移矩阵来实现这一功能。

**predict\_chord** [service/numpyMarkov.py](#servicenumpymarkov.py): 该函数用于根据当前和弦预测下一个和弦，并过滤出概率大于阈值的预测结果。它首先检查当前和弦的数量是否少于马尔科夫链的阶数，然后将当前和弦状态转换为索引，获取该状态的转移概率，并过滤出概率大于阈值的和弦。最后，按概率降序排序并规范化概率，返回预测的和弦及其概率。

**visualize\_transition\_matrix** [service/numpyMarkov.py](#servicenumpymarkov.py): 该函数用于可视化概率转移矩阵，并将其以热图的形式输出为图片。

**state\_to\_index** [service/numpyMarkov.py](#servicenumpymarkov.py): 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。

## 函数信息(按重要性排序)

### 1. predict\_chord ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：73-95
* 重要性得分：2.28559
* 扇出数：1
* 复杂度：26
* 调用链深度：2
* 功能描述： 该函数用于根据当前和弦预测下一个和弦，并过滤出概率大于阈值的预测结果。它首先检查当前和弦的数量是否少于马尔科夫链的阶数，然后将当前和弦状态转换为索引，获取该状态的转移概率，并过滤出概率大于阈值的和弦。最后，按概率降序排序并规范化概率，返回预测的和弦及其概率。
* 实现流程： 检查当前和弦的数量是否少于马尔科夫链的阶数，如果少于则抛出异常。 将当前和弦状态转换为对应的索引。 获取该状态的转移概率。 过滤出概率大于阈值的和弦。 按概率降序排序并规范化概率。 返回预测的和弦及其概率。
* 调用： ValueError,state\_to\_index,keys,
* 内部依赖描述：
  + state\_to\_index: 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。

**代码片段：**

def predict\_chord(self, current\_chords, threshold=0.1):  
 """  
 根据当前和弦预测下一个和弦  
 :param current\_chords: 当前和弦列表  
 :param threshold: 概率阈值，过滤低概率的预测结果  
 :return: 预测的和弦及其概率  
 """  
 if len(current\_chords) < self.order:  
 raise ValueError("当前和弦的数量少于马尔科夫链的阶数。")  
  
 cur\_state = tuple(current\_chords[-self.order:])  
 cur\_state\_index = self.state\_to\_index(cur\_state)  
 probabilities = self.transitions[cur\_state\_index]  
 # 过滤出概率大于阈值的和弦  
 next\_chords = [(list(self.chord\_index.keys())[i], prob) for i, prob in enumerate(probabilities) if  
 prob > threshold]  
  
 # 按概率降序排序并规范化概率  
 next\_chords = sorted(next\_chords, key=lambda x: x[1], reverse=True)  
 total\_prob = sum(prob for \_, prob in next\_chords)  
 next\_chords = [(chord, prob / total\_prob) for chord, prob in next\_chords] if total\_prob > 0 else []  
  
 return next\_chords

### 2. build\_transition\_matrix ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：31-55
* 重要性得分：2.05322
* 被引用次数：1
* 扇出数：1
* 复杂度：27
* 调用链深度：2
* 功能描述： 该函数用于根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。
* 实现流程： 遍历输入的和弦序列，对于每个和弦序列，计算其马尔可夫链的阶数。 对于每个和弦序列中的和弦组合，将其转换为对应的索引，并记录当前状态和下一个和弦的索引。 更新转移矩阵中对应状态和下一个和弦的计数。 对转移矩阵的每一行进行归一化处理，将其转换为概率。 最终返回构建好的概率转移矩阵。
* 调用： state\_to\_index,get,
* 内部依赖描述：
  + state\_to\_index: 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。

**代码片段：**

def build\_transition\_matrix(self, sequences):  
 """  
 根据输入的和弦序列构建概率转移矩阵  
 :param sequences: 和弦序列  
 """  
 for seq in sequences:  
 # 根据马尔科夫链阶数，构建转移概率  
 for i in range(len(seq) - self.order):  
 cur\_state = tuple(seq[i:i + self.order]) # 当前状态（和弦组合）  
 cur\_state\_index = self.state\_to\_index(cur\_state) # 当前状态的索引  
 next\_chord\_index = self.chord\_index.get(seq[i + self.order],  
 self.unknown\_chord\_index) # 获取下一个和弦的索引，处理未知和弦  
 if next\_chord\_index < len(self.transitions[0]): # 确保索引不越界  
 self.transitions[cur\_state\_index, next\_chord\_index] += 1 # 计数转移  
  
 for i in range(len(self.transitions)):  
 row\_sum = np.sum(self.transitions[i])  
 if row\_sum > 0:  
 self.transitions[i] /= row\_sum  
  
 # 将每行的计数归一化为概率  
 for i in range(len(self.transitions)):  
 row\_sum = np.sum(self.transitions[i])  
 if row\_sum > 0:  
 self.transitions[i] /= row\_sum # 归一化处理

### 3. **init** ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：6-16
* 重要性得分：1.95513
* 扇出数：2
* 复杂度：14
* 调用链深度：3
* 功能描述： 该类用于初始化一个马尔可夫链模型，用于预测和弦序列中的下一个和弦。它通过为每个唯一和弦分配一个索引，并构建一个概率转移矩阵来实现这一功能。
* 实现流程： 初始化预测器，设置马尔可夫链的阶数。 构建和弦索引字典，为每个唯一和弦分配一个唯一的整数索引。 计算唯一和弦的数量。 初始化概率转移矩阵，大小为 (n\_chords^order, n\_chords)。 根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。
* 调用： index\_chords,zeros,build\_transition\_matrix,
* 内部依赖描述：
  + index\_chords: 该函数用于为和弦序列中的每个唯一和弦分配一个唯一的索引。
  + build\_transition\_matrix: 该函数用于根据输入的和弦序列构建概率转移矩阵，通过马尔可夫链模型来预测和弦的转换概率。

**代码片段：**

def \_\_init\_\_(self, chord\_sequences, order=1):  
 # 初始化预测器，order表示马尔科夫链的阶数  
 self.order = order  
 # 构建和弦索引字典，每个和弦对应一个唯一的整数索引  
 self.chord\_index = self.index\_chords(chord\_sequences)  
 # 计算唯一和弦的数量  
 self.n\_chords = len(self.chord\_index)  
 # 初始化概率转移矩阵，大小为 (n\_chords^order, n\_chords)  
 self.transitions = np.zeros((self.n\_chords \*\* self.order, self.n\_chords))  
 # 构建转移概率矩阵  
 self.build\_transition\_matrix(chord\_sequences)

### 4. state\_to\_index ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：57-71
* 重要性得分：1.70564
* 被引用次数：2
* 复杂度：16
* 调用链深度：1
* 功能描述： 该函数用于将和弦状态转换为对应的索引，以便在马尔可夫模型中进行状态管理。
* 实现流程： 接收当前和弦状态作为输入。 初始化索引为0。 遍历和弦状态中的每个和弦，计算其对应的索引。 如果和弦在已知和弦索引中，则使用其索引；否则，使用默认的未知和弦索引。 确保索引在有效范围内。 根据和弦的顺序和位置，计算索引的权重并累加到总索引中。 确保最终索引在有效范围内。 返回计算得到的索引。
* 调用： get,
* 内部依赖描述：

**代码片段：**

def state\_to\_index(self, state):  
 """  
 将和弦状态转换为对应的索引  
 :param state: 当前和弦状态  
 :return: 状态的索引  
 """  
 index = 0  
 # 根据和弦组合的每个和弦计算状态索引  
 for j, chord in enumerate(state):  
 # 处理未知和弦，使用默认的未知和弦索引  
 chord\_index = self.chord\_index.get(chord, self.unknown\_chord\_index) # Use the index for unknown chords  
 chord\_index = min(chord\_index, self.n\_chords - 1) # Ensure within bounds  
 index += chord\_index \* (self.n\_chords \*\* (self.order - j - 1))  
 index = min(index, len(self.transitions) - 1) # Ensure final index is within bounds  
 return index

### 5. index\_chords ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：18-29
* 重要性得分：1.69889
* 被引用次数：1
* 复杂度：13
* 调用链深度：1
* 功能描述： 该函数用于为和弦序列中的每个和弦分配一个唯一的索引，并返回一个和弦到索引的映射字典。同时，它还为未知和弦分配了一个独特的索引。
* 实现流程： 初始化一个空集合 unique\_chords 用于存储唯一的和弦。 遍历输入的和弦序列 sequences，将每个和弦添加到 unique\_chords 集合中，确保所有和弦都是唯一的。 计算 unique\_chords 集合的长度，并将其赋值给 unknown\_chord\_index，作为未知和弦的唯一索引。 使用 enumerate 函数遍历 unique\_chords 集合，生成一个和弦到索引的映射字典，并返回该字典。
* 调用： update,
* 内部依赖描述：

**代码片段：**

def index\_chords(self, sequences):  
 """  
 为每个和弦分配一个唯一的索引  
 :param sequences: 和弦序列  
 :return: 和弦索引字典  
 """  
 unique\_chords = set() # 用于存储唯一的和弦  
 for seq in sequences:  
 unique\_chords.update(seq) # 更新唯一和弦集合  
 # 返回和弦到索引的映射字典  
 self.unknown\_chord\_index = len(unique\_chords) # A unique index for unknown chords  
 return {chord: i for i, chord in enumerate(unique\_chords)}

### 6. visualize\_transition\_matrix ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：97-108
* 重要性得分：1.56718
* 复杂度：19
* 调用链深度：1
* 功能描述： 该函数用于可视化概率转移矩阵，并将其以热图的形式输出为图片。
* 实现流程： 创建一个大小为10x8的图形窗口。 使用热图展示转移矩阵，颜色越深表示概率越大。 添加颜色条以显示颜色与概率的对应关系。 设置图形的标题为’Transition Matrix Heatmap’。 设置X轴标签为’Next Chord Index’，Y轴标签为’Current State Index’。 显示图形窗口以输出热图图片。
* 调用： figure,imshow,colorbar,title,xlabel,ylabel,show,
* 内部依赖描述：

**代码片段：**

def visualize\_transition\_matrix(self):  
 """  
 可视化概率转移矩阵并输出为图片  
 """  
 plt.figure(figsize=(10, 8))  
 # 使用热图展示转移矩阵  
 plt.imshow(self.transitions, cmap='hot', interpolation='nearest')  
 plt.colorbar() # 添加颜色条  
 plt.title("Transition Matrix Heatmap")  
 plt.xlabel("Next Chord Index")  
 plt.ylabel("Current State Index")  
 plt.show()

### 7. transition\_matrix\_to\_string ([service/numpyMarkov.py](file:///Users/apple/Public/generates-git/chordPrediction/service/numpyMarkov.py))

* 所属模块/包：ChordPredictor
* 行号位置：110-116
* 重要性得分：1.16793
* 复杂度：8
* 调用链深度：1
* 功能描述： 该函数将概率转移矩阵转换为字符串形式，便于查看和调试。
* 实现流程： 使用np.array2string方法将概率转移矩阵转换为字符串。 设置precision参数为2，确保输出的小数点后保留两位。 使用separator参数为’, ’，设置元素之间的分隔符为逗号和空格。 返回转换后的字符串形式的转移矩阵。
* 调用： array2string,
* 内部依赖描述：

**代码片段：**

def transition\_matrix\_to\_string(self):  
 """  
 将概率转移矩阵转换为字符串形式  
 :return: 转移矩阵的字符串  
 """  
 matrix\_str = np.array2string(self.transitions, precision=2, separator=', ')  
 return matrix\_str