Universidade Federal da Bahia
Instituto de Computação
Bacharelado em CIÊNCIA DA COMPUTAÇÃO

Trabalho de Conclusão de Curso

Composição de Melodias com Aprendizado por Reforço: Investigando a Teoria da Música como Ferramenta Para Inferência de Sentimento

Edio Marcos de Souza

Salvador - Bahia
4 de março de 2023

Composição de Melodias com Aprendizado por Reforço: Investigando a Teoria da Música como Ferramenta Para Inferência de Sentimento

Edio Marcos de Souza

Trabalho de Conclusão de curso apresentado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Prof(a). Dr. Marlo Vieira dos Santos e Souza.

Salvador - Bahia
4 de março de 2023

Composição de Melodias com Aprendizado por Reforço: Investigando a Teoria da Música como Ferramenta Para Inferência de Sentimento

Edio Marcos de Souza

Trabalho de Conclusão de curso apresentado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Banca Examinadora:

Prof. Dr. Marlo Vieira dos Santos e Souza
UFBA

Prof. Dr. Flávio Morais de Assis Silva
UFBA

Profa. Dra. Tatiane Nogueira Rios
UFBA

À minha família

Agradecimentos

Agradeço a meus pais Aparecido de Souza e Ignes Conceição dos Santos Souza, por todos os sacrifícios que fizeram para construir minha educação e me despertar a paixão pelo conhecimento.

Ao meu orientador Prof(a). Dr. Marlo Vieira dos Santos e Souza, por me guiar pelos caminhos da pesquisa com paciência e sabedoria.

A todos os meus professores, desde a alfabetização até a finalização desse curso, pela dedicação, paciência e sobretudo, pelo empréstimo de seus conhecimentos.

A todos os meus colegas de curso, pela troca de informações e descobertas conjuntas, em especial a meu amigo e companheiro João Pedro Brito, pelos recorrentes momentos de aprendizado e construção de conhecimento, que compartilhamos em enumeráveis momentos durante todo o nosso curso, contribuindo significativamente em meu desenvolvimento acadêmico.

A meus filhos Kauê Magalhães de Souza, Cainã Magalhães de Souza e Caíque Magalhães de Souza, pela resignação em me ter distante de suas vidas, nesses anos em que estive ausente nessa missão, e a minha eterna companheira Mari Angela Pinto de Magalhães pela ajuda e suporte, sem os quais estaria fadado ao fracasso.

Finalmente agradeço a todos os meus verdadeiros amigos, que me incentivaram e ajudaram a construir e fortalecer minha determinação durante esse processo, e agradeço também aquelas pessoas que não acreditaram em minha capacidade e determinação, pois nos momentos mais difíceis, me ajudaram a transformar minha persistência em teimosia, me impulsionando um pouco mais frente.

Resumo

A composição musical feita com o aprendizado sobre um corpo de músicas, com uso de RNN, é capaz de gerar linhas melódicas muito convincentes em alguns casos. Contudo, por estar desprovida de uma estrutura definida, pode acabar gerando melodias sem muito sentido musical.

Natasha Jaques, Shixiang Gu, Richard E. Turner, Douglas Eck em seu artigo, propõem uma nova arquitetura, que se utiliza do aprendizado por reforço, para impor uma estrutura global coerente, otimizando algumas funções de recompensas, enquanto mantém as propriedades aprendidas com os dados.

Um treinamento RNN feito sobre um grande corpus de música, para prever a próxima nota em uma melodia, vai ser então refinada em um aprendizado por reforço, que tem como função de recompensa, a combinação entre e as regras da teoria da música propostas, e uma cópia da saída do treinamento RNN.

Os autores demonstram, que essa arquitetura é capazes de construir melodias mais consistentes, reduzindo sensivelmente, construções melódicas com aparência aleatória e resultados sem sentido musical, nessas composições.

As regras da música são implementadas por funções, que estruturam e limitam os comportamentos das melodias geradas, fornecendo saídas de recompensas, para o aprendizado por reforço. Nesse trabalho vamos interferir nessas funções da teoria da música, alterar seus parâmetros e hiper parâmetros, seus pesos de saída e combinar funções.

Executar testes com o propósito de demonstrar a possibilidade de se incorporar minimamente, de forma intencional, aspectos de sentido emocional nessas melodias feitas por máquinas.

Fazer uso desse mecanismo como ferramenta de inserção de sentimentos específicos nas melodias geradas.

Verificar possíveis influências de sentimentos nos resultados melódicos, para demonstrar a possibilidade, do uso dessas modificações, como ferramentas para esse propósito.

Abstract

The musical composition made by learning about a body of music, using RNN, is capable of generating very convincing melodic lines in some cases. However, as it lacks a defined structure, it can end up generating melodies without much musical sense.

Natasha Jaques, Shixiang Gu, Richard E. Turner, Douglas Eck in their article, propose a new architecture, which uses reinforcement learning, to impose a coherent global structure, optimizing some reward functions, while maintaining the properties learned from the Dice.

An RNN training done on a large corpus of music, to predict the next note in a melody, will then be refined in a reinforcement learning, which has as a reward function, the combination between and proposed music theory rules, and a copy of the RNN training output.

The authors demonstrate that this architecture is capable of building more consistent melodies, significantly reducing melodic constructions with a random appearance and meaningless musical results in these compositions.

The music rules are implemented by functions, which structure and limit the behaviors of the generated melodies, providing reward outputs for reinforcement learning.

In this work we are going to interfere with these music theory functions, change their parameters and hyper parameters, their output weights and combine functions.

Run tests with the purpose of demonstrating the possibility of minimally, intentionally incorporating aspects of emotional meaning in these melodies made by machines.

Make use of this mechanism as a tool for inserting specific feelings into the generated melodies.

Check possible influences of feelings on melodic results, to demonstrate the possibility of using these modifications as tools for this purpose.

Sumário

Capítulo 1

Introdução a música Computadorizada

Na segunda metade do século XVII, o compositor Wolfgang Amadeus Mozart cria, como diversão, um jogo semelhante ao jogo de dados, que viria a ser a primeira tentativa de composição aleatória que se tem registro [89], publicado *post mortem* em seu livro *Musikalisches Würfelspiel* (jogo de dados).

Este jogo era constituído de trechos de melodias preestabelecidas, com tamanho padrão de compassos. Sorteadas por dois dados e colocadas em sequência, esses trechos de melodias geravam composições musicais aleatórias, a partir dessas combinações. Somente em 1840, em uma abordagem sob o ponto de vista da matemática, Ada Lovelace propõe a Charles Babbage, a utilização de sua máquina analítica para criação de músicas e imagens [54], e desenvolve uma abordagem teórica para esse tema.

Com a criação do computador eletrônico, no início da década de 1950, Geoff Hill implementa em um computador CSIRAC na Austrália, a primeira composição computadorizada [21]. Neste mesmo ano, Lejaren Hiller e Leonard Isaacson, estrearam a *Illiac Suite*, para quarteto de cordas, o que seria a primeira composição algorítmica.[78].mal,pq

Em 1981 Dave Smith e Chet Wood, propuseram uma interface universal para sintetizadores, na amostra da Áudio *Engineering Society* [60]. Essa interface se tornaria fundamental para a comunicação entre sistemas computacionais, e todo tipo de desenvolvimento de software musical.

Essa interface, passou a ser usada como estrutura básica de comunicação entre equipamentos, sintetizadores, interfaces, *softwares*, e no desenvolvimento de processos de inteligência artificial em música. Hoje, ele é mundialmente conhecido como o protocolo MIDI, *Musical Instrument Digital Interface*, e adotado por toda a indústria musical.

Já na década de 1990, com o desenvolvimento acentuado dos computadores pessoais e de grande porte, a Inteligência Artificial sofre forte impulso, com importantes trabalhos

de pesquisadores nas mais variadas áreas([73].

Nos últimos 20 anos contudo, a composição musical algorítmica, teve um desenvolvimento muito acentuado. Com a utilização de técnicas de aprendizado de maquina.

Na classificação de áudio, reconhecimento de estilo e de motivos musicais [8], com *Provide Motif Translation Invariance* [42]. Na utilização de ferramentas como LSTM (*Long Short-Term Memory*) [38], que ajudou em muito a resolver os problemas da geração de musicas aleatórias, muitas vezes sem sentido musical, das saídas estocásticas dos RNN , que não levam em consideração, as notas anteriores da melodia composta. [12].

Juntamente com técnicas de "*Artificial Neural Network Tunning with Re-inforcement Learning*"[39],descrito por Natasha Jaques e colegas, e regressões logísticas e lineares, esses processos contribuem em muito para um resultado mais natural dessas composições.

A utilização de Redes Neurais, Algoritmos Genéticos e outras tecnologias, vem contribuindo, para que se obtenham resultados com qualidade cada vez mais importantes. Contudo os desafios são ainda muito relevantes se pensarmos em criatividade, expressão de sentimentos e texturas, elementos com propriedades de difícil modelagem.

Novas arquiteturas vêm se mostrando cada vez mais eficientes, como podemos acompanhar no projeto Magenta do Google [1]. Esse projeto, de código aberto, está voltado ao desenvolvimento de música computacional em vários aspectos, com participação de toda comunidade científica.

Mesmo com essas evoluções estruturais, notamos uma dificuldade latente nessas composições, na obtenção de direcionamento estético, no que tange a expressão de sentimentos, e controle específicos dessas características, no sentido de imprimir tais considerações objetivamente.

1.1 Contexto

Os músicos e mesmo as pessoas comuns, costumam fazer uso de metáforas para definir sentimentos ao se referirem a melodias e contextos musicais [64], devido a dificuldade de classificação direta sobre esses parâmetros. Ao se introduzir composições desenvolvidas por aprendizado de máquina, essas identificações surgem de forma aleatória, ou imposta estruturalmente por um corpus preestabelecido de forma homogênea, identificado por um especialista, para direcionar essas composições com esses tipos de sentido criativo estabelecidos.

Dentre vários trabalhos observados, nos chama a tenção o artigo de Natasha Jaques *et al.*, *Tunning with Re-inforcement Learning* [40]. Observamos nessa arquitetura, a possibilidade de exercer algum controle sobre as interferências de caráter emocional de

[1] https://magenta.tensorflow.org/

elementos subjetivos. Alguns desses elementos como tensão, tristeza, impacto de perda, desenvoltura, alegria, êxtase, vitória, derrota, estado contemplativo, muita tensão, ira, entre outros poderiam ser manipulados de alguma forma.

Ao fazer uso dessa arquitetura, interferindo em suas regras musicais, que são usadas como recompensas em um aprendizagem por reforço, ou mesmo criando outras regras, acreditamos poder estabelecer esses objetivos.

Os métodos para inserir recompensas nessa arquitetura, obtendo resultados melódicos que representem uma atmosfera compatível com algumas dessas situações abordadas, torna-se um grande desafio, tanto pela natureza subjetiva do processo, quanto pelos métodos necessários para este tipo de intervenção.

Poderíamos fazer uma análise de como o uso de regras musicais ou estruturais, a reestruturação de alguns metadados ou mesmo a criação de novos elementos de restrição, possam enfatizar nossas necessidades composicionais. A mudança de pesos de recompensas, de parâmetros e hiper parâmetros, que podem interferir nessas características, permitindo assim algum controle, sobre como conduzir composições a experiências emocionais especificas, dentro de um contexto de composição algorítmica.

1.2 Justificativa

A capacidade de traduzir subjetivamente, algum tipo de estado de espírito em um ambiente gerado por máquinas, se apresenta com uma certa dificuldade de construção, o que certamente seria a mola propulsora da criatividade de artistas compositores.

Esse trabalho busca encontrar na arquitetura proposta por Natasha Jaques e colegas, formas de se interferir nesses elementos, no processo composicional, para que possamos ter algum controle no manejo desses elementos subjetivos.

Ao direcionarmos esta proposta a ambientes práticos, não poderíamos deixar de observar o desenvolvimento dos *games* da atualidade, em ações competitivas de grande audiência, em reproduções de performances gravadas e exibidas no *Youtube* com milhões de seguidores, e nos jogos que se desenvolvem em cenários e contextos variados, de acordo com a performance dos personagens.

De forma particular, notamos uma crescente customização dos episódios, onde personagens passam a vivenciar cada momento nesse cenário especifico, como se fosse construído para esse fim, naquele contexto, com variações dentro do mesmo episódio dependendo da performance obtida pelo personagem naquele momento.

Diferentemente de como ocorre nos filmes, com suas trilhas musicais construídas especificamente para cada sequência, tornando-as muito mais vivas, movimentadas e expressivas, com texturas e coloração impregnadas de emoções.

A construção de trilhas para *games* em tempo real, não passa por esse processo, tendo em vista a impossibilidade de se prever, o desenrolar e desenvolvimento das ações antecipadamente.

Com os personagens e os contextos já definidos de forma matemática a cada passo, visualizamos um cenário mais propício para interpretação desses parâmetros e construção de melodias em tempo real, por meio de composições algorítmicas. Intimamente ligadas aos acontecimentos, essas composições com recompensas estabelecidas pelo desenrolar das ações, poderiam dar a essas cenas, sentido estético condizente com os cenários emocionais propostos de forma mais expressiva.

O uso de formas de composição algorítmica, com recompensas estabelecidas pelo desenrolar das ações, nos direciona a pesquisa dessas tecnologias.

Seria também um fator importante na iteração com músicos de forma motivacional, como fator indutivo, na criação de novas composições, em tempo real, em duetos, como desenvolvidos no projeto magenta [2] ou na utilização dessas sugestões melódicas, para criação de peças completas, por compositores.

No campo da iteração humano computador, podemos também pensar em parcerias, onde sementes compostas por máquinas, direcionadas as emoções que o usuário necessita, sirvam perfeitamente a desenvolvimentos conjuntos de composição.

1.3 Objetivo

O objetivo primeiro dessa pesquisa, é reproduzir o experimento de geração de melodias com essa arquitetura. Buscar compreender as tecnologias envolvidas e os processos de desenvolvimento. Testar as funções que restringem e recompensam a escolha de cada nota das composições geradas.

Desenvolver testes isolando alumas dessas funções para que se possa usá-las como ferramentas de controle, na inserção de aspectos subjetivos emocionais nas intenções de composição.

Podemos entender, que notas geradas que estejam dentro de uma determinada escala, serão recompensadas, evidentemente. Se usarmos uma graduação variável, poderíamos verificar como esse aspecto pode influenciar nas intenções das melodias . Controlando a rigidez das regras e determinando uma nota qualquer como sendo uma nota tônica, ou seja a referência de onde inicia e termina essa escala, podemos instaurar várias referências de texturas e sentimentos como por exemplo as existentes nos modos gregos [62]. A implementação de tipos diferentes de escalas como maiores, menores, pentatônicas, menores harmônicas e outras, certamente servem também de base pra criação de ambientes diversos,com diferentes texturas, como cores claras e escuras, quentes ou frias comumente

observadas em composições musicais [13]. Buscar instituir modelos e funções de controle desses elementos de restrição com recompensas são fundamentais.

Propomos:

- verificar, organizando alguns testes, isolando funções e tratando suas influências no contexto total das gerações composicionais, se essas intervenções podem atuar no resultado expressivo, com relação aos sentimentos musicais associados as composições.
- Investigar como esses processos afetam o resultado musical, e se poderiam de fato estabelecer alguma atuação modificadoras.
- Verificar a possibilidade dessas interferências poderem ser usadas como ferramentas, capazes de estabelecer contextos básicos de expressão de sentimentos específicos.
- Estudar e compreender o artigo "*Conservative Fine-Tuning of Sequence Generation Models with KL-control* (Natasha Jaques , Shixiang Gu , Richard E. Turner , Douglas Eck 2016) .Este artigo estrutura uma nova arquitetura, que foi desenvolvida para compor sequências de melodias, com uso de aprendizado por reforço das regras da teoria da música em conjunto com um aprendizado RNN sobre um corpus de músicas.
- Reproduzir o experimento original implementado.
- Desenvolver e modificar recompensas nas regras musicais, com a utilização das tecnologias disponíveis nessa arquitetura, para desenvolver subsídios para obtenção de um conjunto de regras, que possam direcionar a caracterização desses estados de contexto emocional.
- Interferir nas regras da música implementando variações de parâmetros funções e hiperâmetros, e observar os resultados, para tentar estabelecer uma correlação entre a combinações dessas modificações com os estados de sentimento.

1.4 Estrutura

Este documento será dividido em capítulos, organizado da seguinte forma:

- Capítulo 2: Fundamentos
  - Fundamentos sobre regras musicais, um relato sobre o ponto de vista da música como estrutura e arte, e os métodos de aprendizado computacional.
  - Compreensão e classificação dos sentimentos associados a musica, histórico e definições.

  - O protocolo MIDI, suas aplicações e a importância dessa ferramenta na formalização e comunicação dentro da música. Aplicação na industria e na interação com os humanos. A geração de uma nova perspectiva para a computação e notadamente para a inteligência artificial.
  - O aprendizado por reforço. Sua história, fases de desenvolvimento e simulação. A função Valor e a tabela Q, com seus processos evolutivos.
    * A função Valor, funcionamento e simulação.
      Q-learning o aprendizado com uso de um *grid*, um mundo simplificado.
    * *Deep Q-Learning*, as redes neurais, conceitos de neurônio humano e Perceptron.
    * Redes neurais, o modelo humano de neurônio, o *Perceptron* como modelo matemático primário e seus agrupamentos em redes.

- Capítulo 3: Introdução a computação musical.
  - Histórico, desenvolvimento e abrangência de seus processos.
  - Redes neurais.
  - Breve descrição de algumas linguagens particulares para música.

- Capitulo 4: Trabalhos Relacionados.
  - Relação de trabalhos pesquisados para escolha de um modelo para estudo. Breve descrição.

- Capitulo 5: *Tuning Recurrent Neural Networks with Reinforcement Learning*
  - Geração de música com utilização das tecnologias RNN e LSTM, dados da implementação
  - Meios de atuação do aprendizado por reforço, em especial nessa arquitetura descrição da composição dos componentes do aprendizado, por reforço das regras da música e pela saída RNN.
  - A teoria da música aplicada a esse modelo, histórico e aplicação como recompensa no aprendizado.
  - Descrição do artigo de Jaques e colegas, e funcionamento do modelo.

- Capitulo 6: Inserção de sentimentos, estruturação de ferramentas e testes.
  - Reprodução do experimento original, utilizando todas as regras de música implementadas. Implementação e análise de resultados.

  - Experimento usando apenas as regras musicais como aprendizado por reforço, implementação e análise de funcionamento do RL_Tuner.
    * Experimento para resolver problemas com redundância e geração de *looping*
  - Experimento implementado com uso exclusivo do treinamento RNN sobre o banco de dados, resultados.
  - Experimento com a função de tônica da escala, implementação para verificação de seu comportamento.
  - Experimento com a função de tonalidade da escala, implementação para verificação de seu comportamento
    * Apresentação de um resultado específico, que gerou melodias no modo Lídio.
  - Experimento para construção e modificação de escalas de referência.
    * As escalas Menores implementação e analise de resultados.
    * Escalas Pentatônicas implementação e analise de resultados.

- Capitulo 7: Conclusão

Capítulo 2

Fundamentos

Neste capitulo vamos fazer uma abordagem sobre conceitos básicos da teoria da musica, para que se possa ter uma compreensão mínima sobre elementos que serão abordados, durante o desenvolvimento desse trabalho.

Outro aspecto que também precisamos abordar é a questão do sentimento, notadamente relacionado a música , e sua relação com o ouvinte.

Técnicas de aprendizado de máquina, de aprendizado por reforço. Interface de comunicação, e breve relato sobre sentimentos em música

2.1 Estrutura Musical

A música é a arte de combinar os sons, simultânea e sucessivamente, com ordem, equilíbrio e proporção dentro do tempo [9]. A teoria da música é qualquer sistema, ou conjunto de sistemas, destinado a analisar, classificar e compor estruturas para a sua prática e compreensão. Essas regras são desenvolvidas a partir da análise das composições de artistas relevantes de cada época, uma estruturação desses trabalhos de forma analítica, para que se possa, tendo a sua compreensão como uma base de instrução, desenvolver trabalhos de composição, usando-se essas estruturas, já notadamente bem-sucedidas dentro daquele contexto.

Estas regras estabelecem por exemplo, elementos de harmonia, que é conjunto dos sons dispostos em forma simultânea (concepção vertical da música) estruturados segundo regras que se modificam através do tempo. A execução dessas notas que soam simultaneamente, no desenvolvimento de uma composição a sucessão de acordes com funções especificas de repouso relaxamento e outras texturas (cadências).

Elementos de melodias é o conjunto de sons dispostos em ordem sucessiva(estruturação horizontal da música) [9]

Possuem padrões rítmicos e de intervalo conjuntos ou disjuntos, entre suas notas,

que caracterizam e diferenciam as melodias umas das outras. A repetição, variação e desenvolvimento desses padrões com criatividade e originalidade, estabelecem um cenário de composição musical causando maior ou menor interesse de acordo com essas combinações, levando-se em conta itens como beleza e agradabilidade de se ouvir dentro do curso temporal e geográfico.

Uma música é dividida em compassos, que é o menor espaço de repetição da pulsação rítmica na qual ela se desenvolve, e o tempo, a menor unidade de pulsação dentro desse espaço. Por exemplo, em uma música como a valsa, temos uma pulsação ternária, em que, a cada três pulsações rítmicas temos um tempo forte. No samba e na marcha essas pulsações ocorrem a cada dois tempos (compassos binários), e no jazz e no rock, a cada quatro tempos(compassos quaternários).

Uma composição musical possui uma forma, que é a macro estrutura, medida e agrupada em grupos de compassos, utilizado em cada trecho da composição. Um blues por exemplo, tem uma estrutura quaternária que se repete a cada 16 compassos. Estes trechos musicais, ou períodos, são normalmente nomeados com letras maiúsculas, e é comum encontramos essas formas, descrita por exemplo como AABA, onde temos duas vezes a parte A, uma vez a parte B ,também chamada de segunda parte e novamente a repetição da primeira parte (canção).

São regras desse tipo, em um universo evidentemente mais amplo, que são usadas nas composições algorítmicas, para restringir os conjuntos possíveis de saídas. Conferem sentido estético, refinam a qualidade artística, a agradabilidade resultante.

Dentro de um determinado cenário de possíveis saídas melódicas, podemos restringir (ou penalizar), por exemplo, tonalidade, forma, pulsação, notas fora de escalas, só para citar alguns elementos.

Suponhamos estarmos em um determinado trecho musical em dó maior, seguindo uma determinada cadência de acordes, onde as melodias devem conter notas pertinentes a esses acordes, ou transitarem por escalas pertencentes a esse campo harmônico, sobre o ponto de vista tonal. Criamos assim uma série de restrições que nos tiram do aleatório, para uma possibilidade mais bem estruturada.

Este cenário foi criado para exemplificar como podemos usar restrições de estruturas musicais introduzidas como penalidades em composições usando aprendizado por reforço.

2.2 Sentimento em música

Um dos primeiros trabalhos que se tem notícia, que se propôs a codificar as principais atitudes emotivas dos homens e animais foi feita por Charles Darwin. [52]. Nessa

proposta, o renomado cientista, afirma que as principais atitudes emotivas expressas por homens e animais são herdadas, inatas, ou seja , não são aprendidas.

A percepção mais refinada da interpretação das emoções nos indivíduos, é uma tarefa de difícil compreensão, devido as variantes de suas manifestações em cada um, e ainda dependentes do momento em que estejam atuando. O humor serve como pano de fundo para a emoção, ou seja, se a pessoa está melancólica, uma música com humor semelhante pode desencadear uma emoção relacionada à tristeza, podendo levar a uma reação física como o choro.

> A mesma música em uma pessoa com um humor de alegria pode inspirar a tristeza, mas provavelmente não terá o mesmo efeito causado no individuo anteriormente descrito. O humor de certa forma amplifica a emoção a que ele está relacionado. [17, p. 10].

As emoções mais básicas podem ser verificadas de forma comum entre humanos e animais. Darwin identificou similaridades em organismos diferentes, inclusive com seres humanos, como por exemplo urinar e defecar em situações de perigo extremo, a piloereção em situações de perigo, supostamente para que possam parecer mais violentos do que em outras situações. Segundo Darwin, provavelmente a ereção dos pelos é uma das expressões emocionais mais genéricas, presente em cães, leões, hienas, vacas, porcos, antílopes, cavalos, gatos, roedores e morcegos. Os pelos também se eriçam levemente no corpo humano em situações de fúria ou terror.Como esses estados são responsáveis pela ereção dos pelos em animais peludos, tendo então a piloereção uma finalidade de cunho emocional[50].

Para uma universalização dos sentimentos, em uma primeira abordagem, vamos considerar apenas alguns dos sentimentos primários como classificados por Ortony et al. [71], sentimentos estes como: raiva, desânimo, tendência de ação como desespero, medo, ódio , esperança, amor, tristeza, felicidade, surpresa, mágoa, ira e pavor, ansiedade, alegria, aflição.

Atualmente muitos estudos, têm demonstrado também, que elementos relacionados à expressividade (tempo, andamento, articulação, etc.) podem ser facilmente obtidos e manipulados, permitindo que se faça uma relação confiável e sistemática desses elementos com o julgamento de expressividade dos ouvintes como mostrado em artigo de Juslin et al. [43].

Em seu artigo *What music makes us feel* [15], Alan S. Cowen e colegas examinaram em 2 países com culturas diferentes, Estados Unidos e China, os sentimentos evocados por 2168 trechos de musica, descobrindo 13 tipos distintos de experiências relatadas, observando que categorias como assombro tem uma conotação mais marcante do que características afetivas amplas. Notaram também que emoções tratadas até aquele momento como discretas, podem ser misturadas como gradientes contínuos, fornecendo respostas

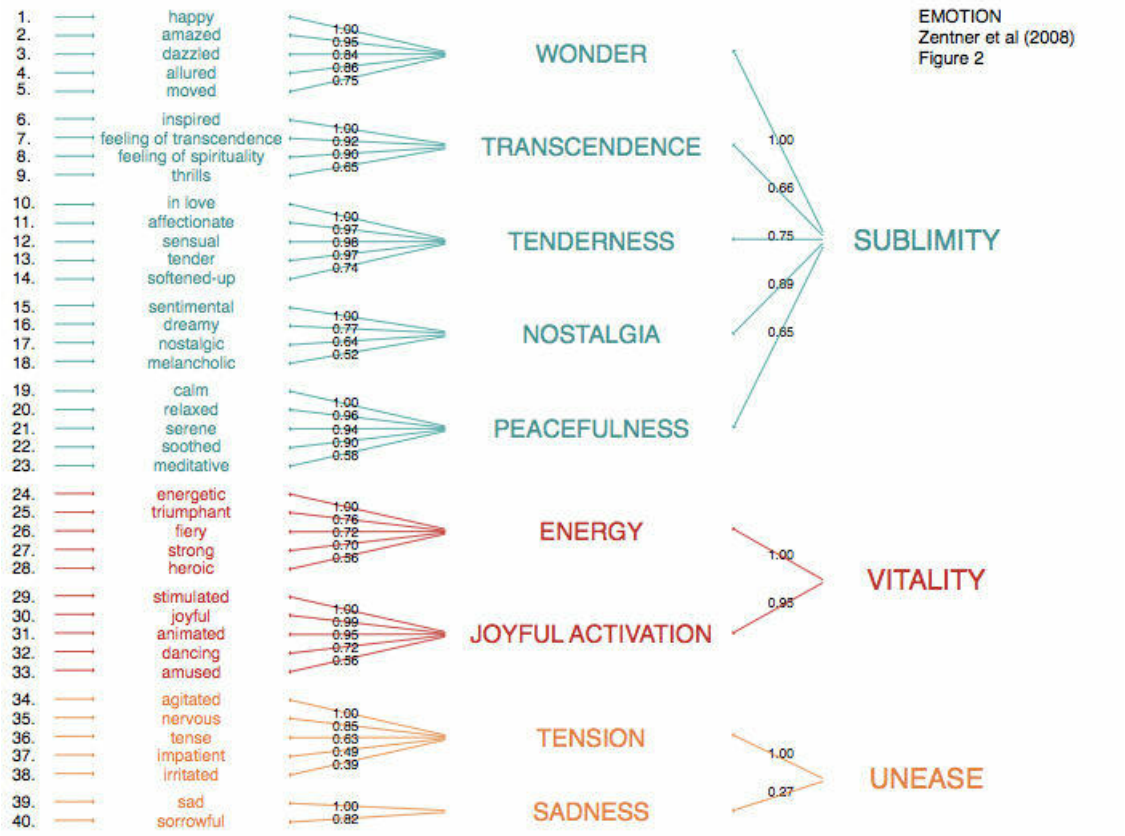
sobre a natureza de experiências subjetivas associadas a musica, em um cenário, onde 2759 ouvintes relatavam sentimentos específicos como: raiva, triunfo, paz, assombro, medo para citar alguns sentimentos.

Essas descobertas, revelam um espaço de experiência subjetiva associada à música, podendo informar investigações que vão desde a etiologia dos distúrbios afetivos, até a base neurológica da emoção. A música também modula a atividade em regiões do cérebro implicadas no processamento relacionado à emoção[48]

Um dispositivo de domínio específico para medir emoções induzidas musicalmente é introduzido : a Escala de Música Emocional de Genebra.

O GEMS é o primeiro modelo concebido para capturar as emoções evocadas musicalmente. Um estudo baseado em uma ampla variedade de amostras de musicas e ouvintes, constituído de 9 escalas e 45 rótulos de emoções, frequentemente utilizado em estudos sobre música e emoção, como mostra a Figura 2.1.

Figura 2.1: Gems



Fonte: Elaborado por [90]

Conceituar uma experiência subjetiva está relacionado com a natureza dos conceitos que caracterizam os nossos sentimentos, e é visto como a questão central da ciência afetiva[14]. Ao ouvirem musica comovente ou efervescente, sentimentos particulares a ela

como: tristeza, medo, alegria, seriam a base de sua experiência, ou mesmo características afetivas mais gerais, como valência, desagradável vs. agradável e excitação, por exemplo.

Alguns estudos descobriram que dentro de um grupo cultural, as pessoas podem com uma certa precisão, associar sentimentos relacionados a pequenas variedades de seleções musicais.

Rotulando esses sentimentos em uma quantidade pequena de categorias de emoções, podem ser mais facilmente entendidas. Raiva, felicidade, medo, tristeza, surpresa e ternura , como mostra Justin e colegas em seu artigo *Expression, perception, and induction of musical emotions* [43] ou mesmo usando escalas de valência e excitação.

A música instrumental mesmo sendo abstrata, nos leva a experiências subjetivas, provocando nas pessoas sentimentos como calafrios, arrepios na espinha, risos, lágrimas, arrepios, nó na garganta e descargas de adrenalina[33].

2.3 O Protocolo MIDI

Quando pensamos em fazer uma atividade de música de forma computadorizada, nos deparamos com um problema, o acesso da máquina ao áudio que a música está gerando. Compreender as notas que estão sendo tocadas, os ritmos dos vários instrumentos envolvidos, os andamentos, os compassos, os blocos harmônicos, os timbres envolvidos e outros parâmetros, tem um custo computacional extremamente elevado para ser compreendido , codificado e processado por máquinas.

Em 1982 surge o protocolo MIDI , uma solução matemática muito simples para representação da música, codificando com grande precisão todo o comportamento em uma musica, podendo ser reproduzido em um contexto de audio discreto mas mutio eficiente.

Publicado formalmente em sua versão 1.0 em agosto de 1983 [57], o *Musical Instrument Digital Interface*, foi publicado por um consórcio de fabricantes de sintetizadores japoneses e americanos, é hoje utilizado por toda indústria da música.

Este protocolo pode ser abstraído como uma representação matemática, das ocorrências mecânicas efetuadas em um teclado musical, ao gerar um som. Imagine que ao se tocar uma nota em um teclado, ao invés dele reproduzir um som, ele iria , em função do tempo, dizer qual nota está sendo tocada, registrando um parâmetro *"pitch"* que estabelece a altura da nota tocada, com um número de 0 a 127 e apenas um bit, que está *"on"*, *"off"*. A nota Lá por exemplo, quando acionada registra *"on"*, e quando termina a execução um sinal *"off"* desliga a ocorrência, simples assim, um bit *on-off*.

Ele também registra vários outros parâmetros, como a altura da nota tocada, com apenas 1 numero de 8 *bits*, o volume com que esta nota está sendo tocada *(velocity)*, o ataque na nota executada ou seja a pressão que está sendo colocada na tecla*("aftertouch")*,

o *decay* , *pitch bend* e outros mais, capazes nesse conjunto, com apenas alguns números, registrar perfeitamente o comportamento daquela som emitido, sem nenhuma emissão de som.

Em uma atitude reversa, essa notação quando der entrada em um sintetizador, por exemplo, especifica a ele, que deve emitir o som daquela determinada nota da forma como foi registrada, e ela será executada de forma idêntica, como se fosse um gravador de áudio, gravada sem áudio só com alguns números.

Esses parâmetros podem também indicar, que você pode executar polifonicamente um instrumento. Pode distinguir qual instrumento está sendo tocado, (timbre), e separado por canais MIDI, pode agregar vários instrumentos, como piano, bateria, contrabaixo, flauta e outros [69]. Controla também o tempo de música e todos os parâmetros para a sua exata reprodução.

Este sistema viabilizou a comunicação entre a música como som, e os processos de tecnologia de forma eficiente e com baixíssimo custo operacional. Todos os trabalhos de composição algorítmicas aqui pesquisados, são baseados nessa tecnologia.

Para uma performance mais eficiente de treinamento, nesse trabalho foi usado mais uma simplificação sobre os arquivos MIDI, o *note-sequence* [1].

Essa abordagem visa simplificar ainda mais o processo de treinamento. Representado com uma subdivisão mínima de tempo como uma posição em um vetor, onde cada posição desse vetor registra uma nota. Essa nota seria a de menor valor da composição, um quarto de tempo da pulsação em nosso caso, o que em um compasso de 4/4 seria uma semi colcheia. Cada posição desse vetor será preenchida com um número, que representa a nota tocada entre 0 e 127, -2 para ausência de som e -1 para deixar soando a nota anteriormente executada. Sem levar em conta os outros parâmetros MIDI como intensidade, volume , timbre etc... simplifica significativamente o processamento de aprendizagem dos modelos.

Os pesados processamentos de aprendizado de máquina, redes neurais redes recorrentes e outras tecnologias, processam apenas um pequeno bloco de números que vão se modificando na linha do tempo, verificando ou introduzindo padrões de forma simples e rápida.

2.4 Aprendizado por reforço

Em 2016, Lee Sedol, o então campeão mundial por 18 vezes, do jogo de tabuleiro oriental Go, foi derroto 4 vezes em uma série de 5 partidas, em confronto com a maquina de IA da Google , o AlphaGo. [85]. Este evento foi um marco no confronto entre homem

[1] https://github.com/magenta/note-seq

e a a a máquina, onde a máquina vence.

O Alpha Go usou um conjunto de dados de aproximadamente 100.000 partidas para aprender as melhores táticas.

Em uma versão seguinte o AlphaGoZero foi treinado, jogando contra si mesmo, iniciando com movimentos aleatórios e aprendendo apenas com as regras do jogo, em milhões de repetições desse processo de aprendizado, executado durante 3 dias. Foi então capaz de derrotar completamente o AlphaGo, utilizado no confronto com Lee, apenas com aprendizado por reforço.

As primeiras teorias sobre aprendizado por reforço foram representadas em outros contextos. Edward L. Thorndike, em seu trabalho sobre a "Lei do Efeito" [82], cuja proposição era que a ação de movimentar uma alavanca por um gato confinado, seria reforçada por uma recompensa de forma imediata, o fornecimento de um peixe para alimentação do felino. Nesses experimentos, onde alguns tipos de resposta que produziam consequências mais recompensadoras ao felino , passavam a ter sua frequência aumentada com o decorrer do tempo, enquanto os comportamentos que produziam frustração em seus objetivos tinham sua frequência diminuída sistematicamente [29].

Esse aprendizado por reforço serviu de inspiração para o desenvolvimento de um algoritmo que aprende uma politica ótima por meio de iteração com o meio ambiente [47]. Uma recompensa é uma quantificação de uma determinada ação, executada por um agente em um determinado estado, para atingir de uma forma ótima o seu objetivo. A estratégia que o agente se utiliza para a escolha da próxima ação, é chamada de política (π).

A função Valor, Vπ(s) é o retorno esperado a longo prazo da ação atual, que diminui seu valor estimado, a cada passo em que o objetivo a ser atingido se torna mais distante no futuro. Um agente aprende a função valor para uma dada política, prevendo uma solução ótima com base nessa função. Usa um controle fora da política, e atualiza a seleção da ação, usando a equação ótima de Bellman e uma política [86].

Um agnte usa um controle fora da política, que separa a política de diferimento da política e-greede de aprendizagem e atualiza a seleção da aação o usando a equaação ótima de Bellman e uma política [86].

O aprendizado por reforço é definido pela *Markov decision process* (MDP) [88] onde um agente introduz o conceito da função valor para o aprendizado, ligada a equação de Bellman. Constrói a equação de Bellman usando o MDP e a função valor. O *q-learning* é então aplicado para a resolução, em um ambiente probabilístico, onde estados de transição e recompensas são aleatórios.

Estado é um conjunto "S"onde o agente pode observar, e ações é um conjunto "A"de ações possíveis em um estado. A probabilidade de transição de um estado "s"para

outro estado "s' "ao executar uma ação "a", é uma representação numérica que para o MDP dependem apenas do estado atual e da ação, construindo uma matriz de decisão. A probabilidade do próximo estado ser compensado pela próxima compensação e magnitude [81] pode ser representado da seguinte forma :

$$P^a_{ss'} = P[S_{t+1} = s' | S_t = s, A_t = a] \tag{2.1}$$

onde $P^a_{ss'}$ é a probabilidade contida na matriz "P"ao mover para o estado "s'"quando a ação é realizada no estado "s"e "t"denota um instante de tempo. A recompensa é a informação dada ao agente para que possa aprender no estado "s",na ação "a"e no tempo "t". A recompensa recebida será:

$R^a_{ss'} = E[R_{t+1} | S_t = s, A_t = a]$

onde $R^a_{ss'}$ é a definição da função de recompensa, "t"é tempo e 'E', é o valor esperado para a recompensa,dada a ação "a"e ocorre quando o agente se move de um estado "s"para "s'". Pode expressar o valor da compensação como um valor esperado e quando realiza uma ação "A"no estado "S", o ambiente informa ao agente no instante "t + 1", o próximo estado "S'"em que pretende entrar e a recompensa que receberá. Um fator de desconto será introduzido, a cada estado que o agente atua.O estado recebe uma compensação, que vai diminuindo com o passar do tempo, introduzindo assim o conceito de depreciação, com valores entre 0 e 1 [20]. Ao chegar a um determinado estado sua ação é determinada por uma política:

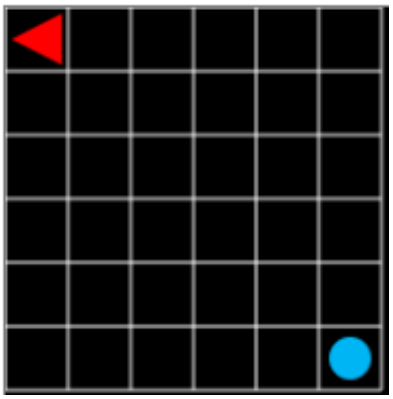
$$\pi(a/s) = p[A_t = a | S_t = s]$$

Para uma visão exemplificada, vamos então criar um ambiente simples de *grid* de 8 x 8 quadros,[45] onde temos representada uma matriz de 8 x 8 , uma seta vermelha representando o agente, e um círculo azul representando o objetivo a ser atingido, como mostra a Figura 2.2.

Caminhando quadro a quadro, quando o agente atinge o mesmo quadro do *grid* em que se encontra o objetivo, dentro de um período de tempo determinado, ele é bem sucedido, terminando assim um episódio, e outro se inicia. Para atingir esse objetivo , ou alvo, o agente pode se movimentar em uma das quatro direções vizinhas a sua posição atual (para cima, abaixo, direita e esquerda), a cada etapa, com movimento restrito, demarcado pela moldura da grade.

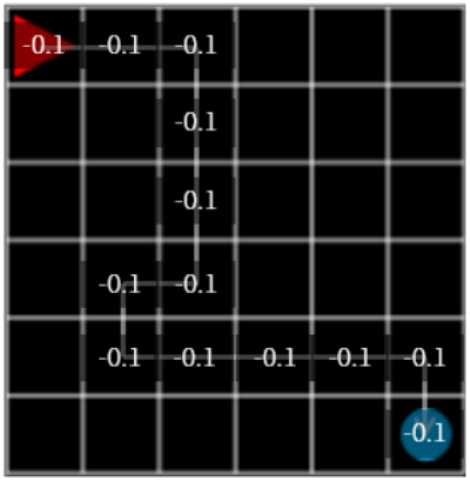
Quando o agente atinge o alvo recebe uma recompensa de +1.0 usada como exemplo para esse cenário. A cada etapa que o agente percorre , recebe uma recompensa negativa de -0,1, com o intuito de incentivar a ação de buscar a recompensa máxima, no menor numero de passos possíveis como mostra o exemplo da Figura2.3

Figura 2.2: Grid de 8X8



Fonte: Elaborado por [45]

Figura 2.3: *grid* com as recompensas negativas por mudança de estado. Ao atingir o alvo tem uma recompensa 1.0 como teve uma somatória no percurso de -1.1, fica com uma recompensa de -0.1 no objetivo



Fonte: Elaborado por [45]

Essas configurações são importantes para que os agentes ajam no sentido de maximizar a compensação do melhor caminho, e penalizar os desvios de percurso ótimo, no aprendizado por reforço.

Vamos imaginar para uma simulação, que o agente faz um movimento aleatório a cada passo, num total máximo de 100 episódios, onde com sorte, poderá tingir o objetivo. Contudo, na maioria das vezes essa busca será frustrada, pois o agente não sabe aonde se encontra o objetivo.

Nessa simulação vamos criar um percurso aleatório até o objetivo, adicionando a cada etapa uma recompensa (negativa) de -0,1 a cada passo, e uma recompensa positiva de +1 quando o agente atingi o objetivo. Em um percurso simulado de 11 passos com

valores de -0,1 a cada passo, numa somatória total de -1.1, que somada a recompensa final de +1.0, chegara um resultado de final de :

$$V_{grid} = max(\gamma * V_{neighborgrid}) + \alpha$$

$-0.1 ==> (-0.1 * 11 + 1.0 = -0.1)$ recompensa essa que será repassada a todas as posições que seus movimentos percorreram, como mostra o exemplo a da Figura2.3

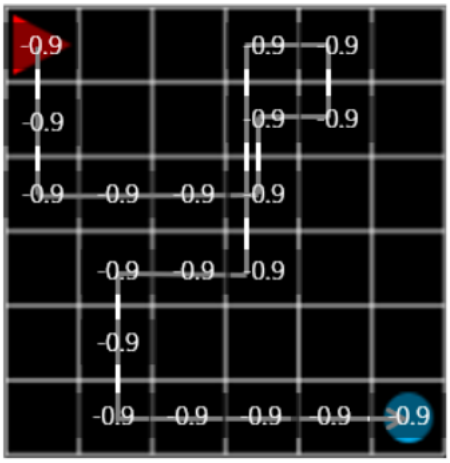
Em um eventual segundo percurso aleatório usado no exemplo, o modelo propagaria uma recompensa de $-0,9$ para cada movimento,

$$V_{grid} = max(\gamma * V_{neighborgrid}) + \alpha$$

$$(-0.1 * 19) + 1.0 = -0.9$$

,Figura2.4

Figura 2.4: Grid com as recompensas repassadas da recompensa final que é -0.9



Fonte: Elaborado por [45]

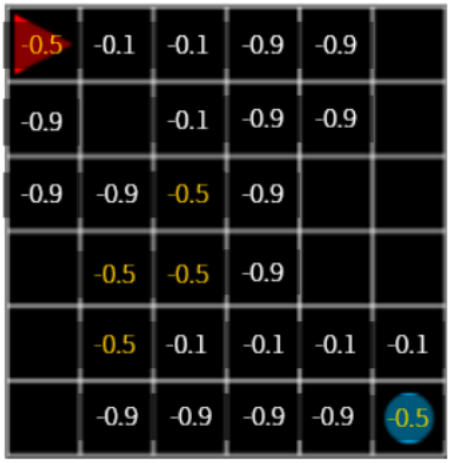
sendo que nos pontos sobrepostos dos dois percursos, vamos estabelecer uma média entre os valores de recompensa desses pontos, ao invés do máximo valor, apenas para efeito de uma simulação como mostrado na Figura 2.5

$$(-0.1 + -0.9)/2 = -0.5$$

Numa simulação de 50 episódios, com no máximo 200 passos a cada episódio , usando os critérios acima chegaremos a configuração mostrada na figura 2.6, com o gráfico mostrando o valor das recompensas a cada episódio e as grades com as recompensas já sobrepostas.

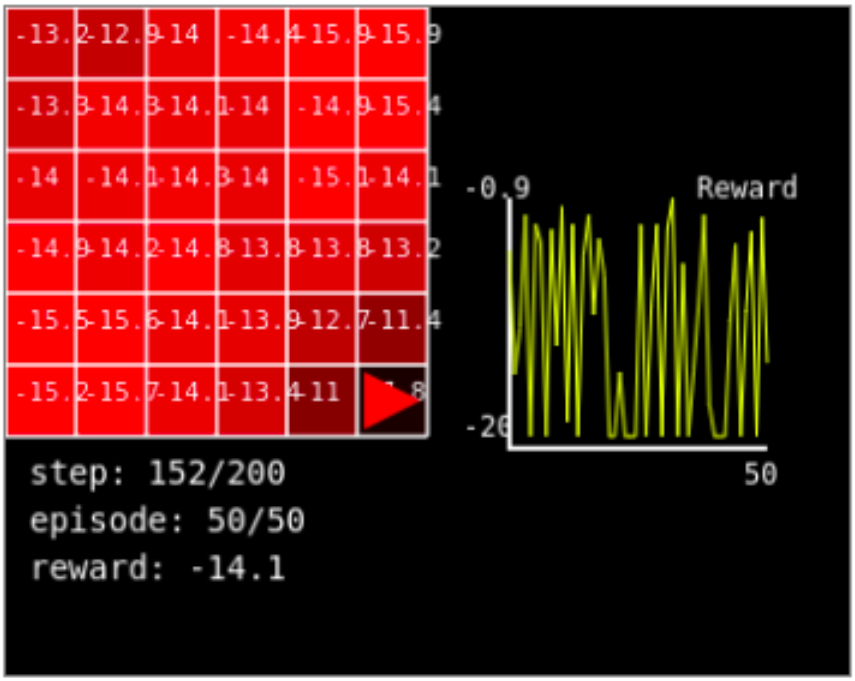
Nesta outra tabela criada da mesma forma, o agente em sua execução de percurso, busca um caminho com as melhores recompensas, (as menos negativas) mas nem sempre

Figura 2.5: Grid com as medias nas sobreposições



Fonte: Elaborado por [45]

Figura 2.6: Simulação de 200 passos

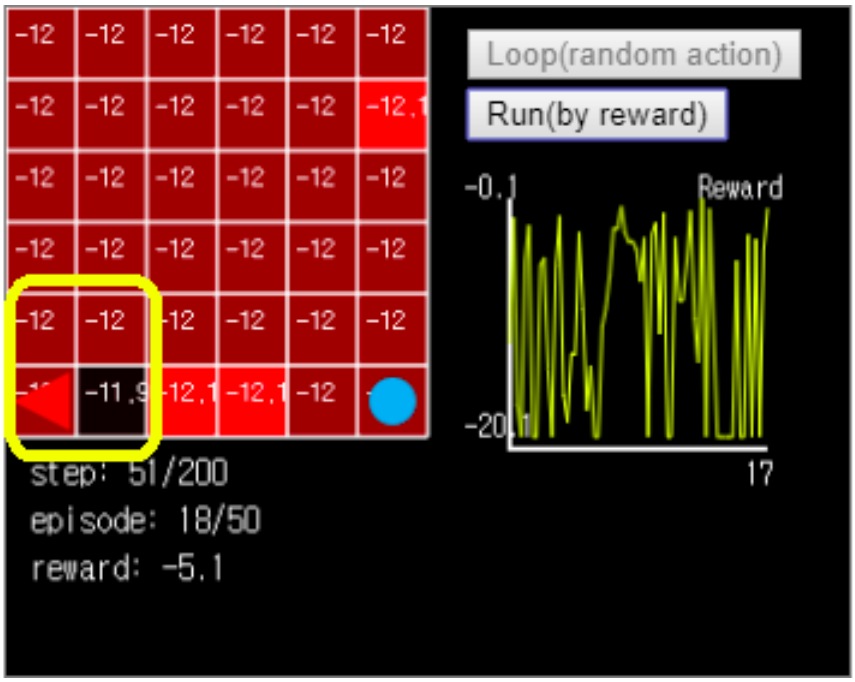


Fonte: Elaborado por [45]

ele se sairá muito bem nessa tarefa , pois podem ocorrer situações em que esse movimento entra em um *looping* infinito, como mostrado na figura 2.7 onde faria o percurso $(x = 0, y = 8) ==> (x = 1, y = 8)$ e vice versa. Verificamos também que os valores de recompensa das grades, são quase todos constantes o que tornaria a escolha do maior muito aleatória, pois seria esse o procedimento em caso de vizinhos com mesmo valor.

Os agentes com movimentos aleatórios, acabam passando muitas vezes por uma mesma grade, caracterizando movimentos desnecessários, o que nos leva a buscar um

Figura 2.7: Simulação de 200 passos e looping



Fonte: Elaborado por [45]

critério tal que pudesse estabelecer quais os movimentos que são realmente úteis nesse processo, em outras palavras, que contribuem efetivamente para diminuir percurso que o agente efetua para atingir o objetivo final.

2.4.1 Função Valor

Quase todos os algoritmos de aprendizado por reforço envolvem estimativas de funções de valor, funções de estados (ou de pares estado-ação) que estimam o quanto é bom para o agente estar em um determinado estado (ou o quanto é bom realizar uma determinada ação em um determinado estado) [80].

A noção de "o quanto é bom" aqui é definida em termos de que podemos esperar uma recompensa futura, ou para ser mais preciso, uma política específica.

Uma política, π, é um mapeamento entre cada estado, $s \in S$, e uma ação, $a \in A(s)$, com probabilidade $\pi(a|s)$ de realizar a ação "a"no estado "s". Informalmente, o valor de um estado "s"sob uma política π, denotado V π (s), é o retorno esperado ao iniciar em "s"e depois seguir a politica π .

G t é definido como alguma função específica da sequência de recompensa.

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

onde γ é um parmetro entre 0 e 1, denominado taxa de desconto, que determina o

valor presente das recompensas futuras: uma recompensa recebido k passos de tempo no futuro vale apenas $^(k1)$ vezes o que valeria, se fosse recebido imediatamente.

Para *Markov decision process* (MDP), podemos definir V π (s) formalmente como:

$$V_\pi(s) = E_\pi[G_t|S_t = s] = E\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{2.3}$$

onde E_π $[\cdot]$ denota o valor esperado de uma variável aleatória dado que o agente segue a política π, e t é qualquer passo de tempo. Observe que o valor do estado terminal, se houver, é sempre zero.

Informalmente, o valor de um estado s sob uma política π, denotado por V_π (s), é o retorno esperado ao começar em s e seguir a política π daí em diante [80]. Claro que as recompensas que o agente pode esperar receber no futuro, vão depender de quais ações ele vai tomar. Assim, as funções de valor são definidas no que diz respeito a políticas específicas. Da mesma forma, definimos o valor da ação "a"no estado "s"sob uma política π, denotado q π (s, a), como o retorno esperado a partir de "s", realizando a ação "a", e depois seguindo a política π:

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] = E\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \tag{2.4}$$

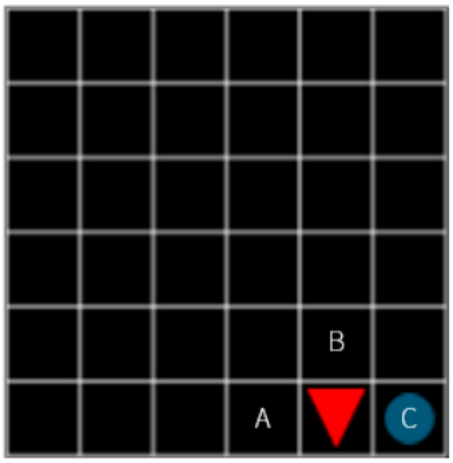
Se um agente segue a política π e mantém uma média, para cada estado encontrado, dos retornos reais que seguiram aquele estado, então a média irá convergir para o valor do estado, V π (s), na medida em que, o número de vezes que esse estado é encontrado e caminha para o infinito. Se médias separadas forem mantidas para cada ação tomada em um estado, então essas médias irão convergir da mesma forma para a ação, q π (s, a)[80].

Para essa análise em nosso cenário de *grid*, vamos considerar os movimentos de forma reversa, ou seja partindo do destino final para o início. Numa posição onde o agente está prestes a atingir o objetivo, numa casa do *grid* vizinha ao objetivo, como mostrado na Figura 2.8 . Mover o agente para a casa "C"seria evidentemente a melhor escolha , pois acarretaria numa recompensa de +1,0 ao passo que se esse movimento for para "A"ou "B", a recompensa seria de -0,1.

Se nos afastarmos pouco a pouco do objetivo, como mostra a Figura 2.9 , as recompensas que os movimentos possíveis receberiam seriam iguais, tanto "A", "B"ou "C"pois receberiam -0,1, mas se pudéssemos olhar a grade vendo o objetivo certamente escolheríamos a posição "C", pois estaria mais próxima desse objetivo.

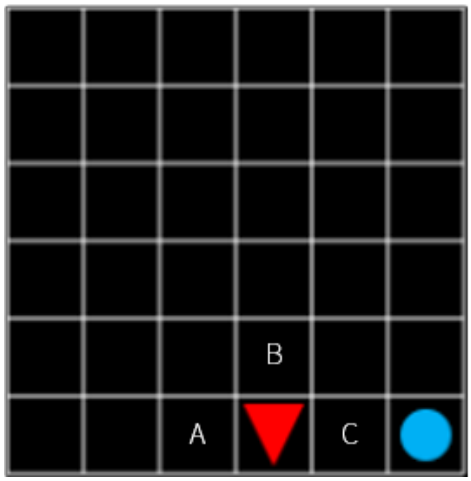
Logicamente as recompensas que você pode obter no próximo movimento "C"serão maiores do que as recompensas obtidas no movimento seguinte caso escolha "A"ou "B"como mostra a Figura 2.9 e assim progressivamente a cada passo teremos uma posição que deixaria o agente mais próximo do objetivo.

Figura 2.8: Agente a 1 passo do objetivo



Fonte: Elaborado por [45]

Figura 2.9: Agente vai se distanciando do objetivo



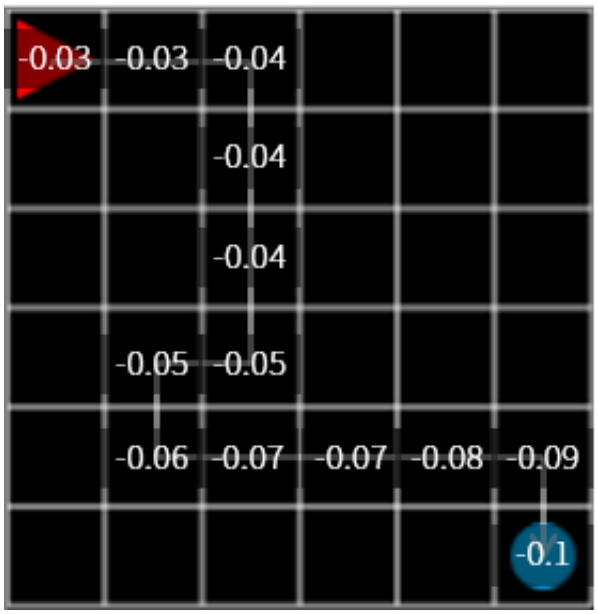
Fonte: Elaborado por [45]

Propagando essa ideia de forma reversa até o início dos movimentos e introduzindo uma taxa de desconto nessa ação reversa, estaríamos estabelecendo um conceito, que indicaria que a cada passo teríamos um desconto, por nos distanciarmos do objetivo. A recompensa futura vale menos que a recompensa atual, que será expressa em um valor entre 0.0 e 1.0 e chamaremos de γ na expressão matemática.

Calculando agora essas recompensas multiplicando seus valores atualizados por um γ no valor de $0,9$ a partir do objetivo em direção ao início (de forma reversa), atualizaremos os valores das posições no *grid* nessa simulação, de tal forma que o percurso passaria a ser traçado por valores progressivamente maiores em direção ao objetivo. Podendo ser percorrido a partir de qualquer ponto diretamente ao objetivo e escolhendo a cada etapa o vizinho com maior maior valor, o sistema estaria traçando um caminho direto ao objetivo

(na simulação por uma questão de espaço estamos usando 2 casas decimais).

Figura 2.10: valores com $\gamma = 0,9$



Fonte: Elaborado por [45]

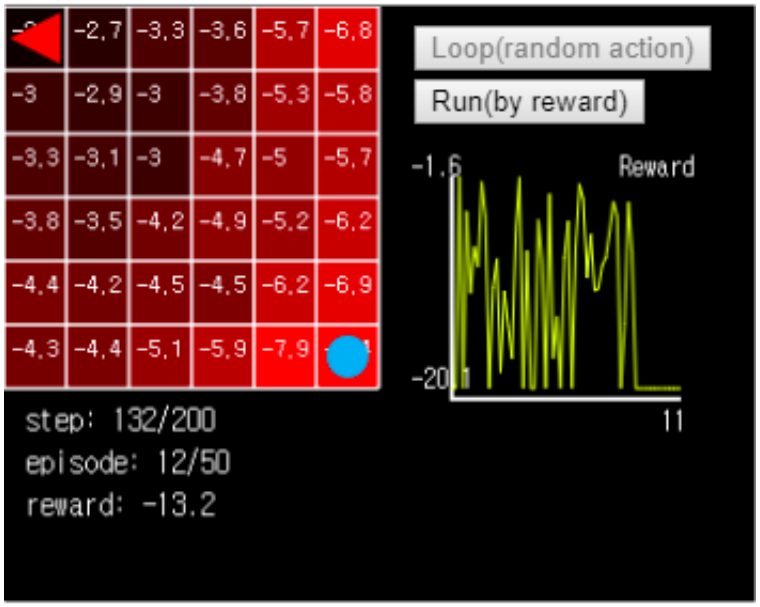
Se atualizarmos todo o *grid* com $\gamma = 0,9$, podemos facilmente observar que a diferença entre as posições ficam, mais pronunciadas do que na tabela anterior, contudo, ainda vamos ter um grande numero de agentes se afastando do objetivo como mostra a simulação da figura 2.10.

$V_{t+1} = V_t * \gamma$
$-0,1 * 0,9 = -0,09$
$-0,09 * 0,9 = -0,081 => -0,08$
$-0,081 * 0,9 = -0,0729 => -0,07$
$-0,0729 * 0,9 = -0,0651 => -0,07$
$-0,0651 * 0,9 = 0,059 => -0,06$

Note que na posição $(x = 0, y = 0)$ o agente numa eventual execução entrará em *looping* com a posição $(x = 1, y = 0)$ Figura 2.11

Para resolvermos essa incerteza, vamos criar uma função de valor, que seria uma representação numérica do valor de um estado. Se obtivermos uma representação para cada estado e efetuarmos um movimento para um estado de maior valor poderíamos eliminar esse problema.

Figura 2.11: *looping* na posição $(x = 1, y = 0)$



Fonte: Elaborado por [45]

É necessário que exista um modelo de mundo, e um valor atribuído a uma ação "a"realizada em estado um "s"em um determinado tempo "t". Mesmo que ainda não se saiba para onde essa ação leva, para podermos trabalhar com uma função de avaliação $Q(s_t, a_t)$, e estados com ações associadas, a escolha da ação ótima é feita pela regra [25]:

$$\pi^* = argmax_a Q(s, a)$$

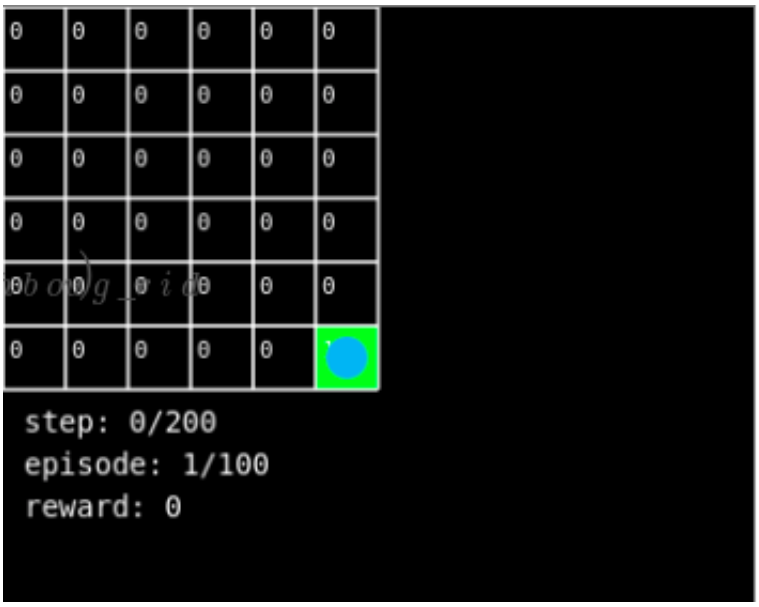
A função valor será inicializada com "0"para todos os estados Figura 2.12, no entanto como o maior valor está aonde o episódio termina, daremos a este estado o valor da compensação, +1 no nosso cenário.

Não sendo a grade de destino, adicione -0,1 porque você recebe essa recompensa (negativa) não importa para onde se mova. O valor da posição do *grid* a cada passo deverá ser o maior valor de *grid* dos seus vizinhos, (sendo i,j uma posição , seu vizinhos seriam as posições (i, j+1),(i, j-1),(i+1, j),(i-1, j)). Multiplicado por gama e somado a recompensa (negativa) recebida a cada passo, ou seja, na posição do *grid* em que estamos, escolhemos o melhor vizinho que seria o de maior valor e atualizamos o valor desse *grid* com o resultado da função valor calculada. Nesse momento em nossa simulação quando a grade de destino já estiver valorada, podemos aplicar a função:

$$V_{grid} = \alpha + max(\gamma * V_{neighborgrid})$$

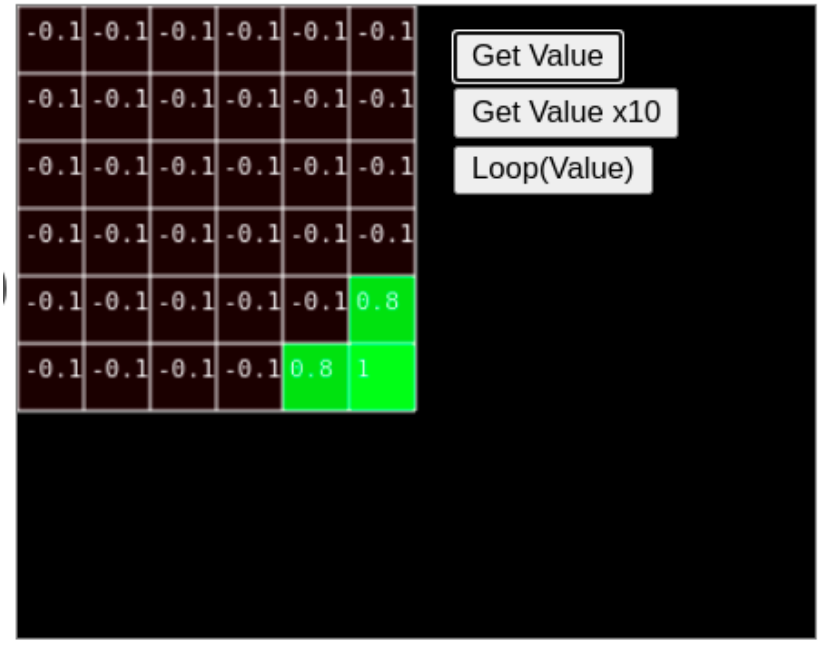
a seus vizinhos, pois ele é o vizinho de maior valor, e prosseguir recursivamente nesse processo até atingir o objetivo final ou parar a simulação, como mostram as sequências de Figuras 2.13, 2.14,2.15, 2.16

Figura 2.12: Função valor preenchida com zeros em todas as posições



Fonte: Elaborado por [45]

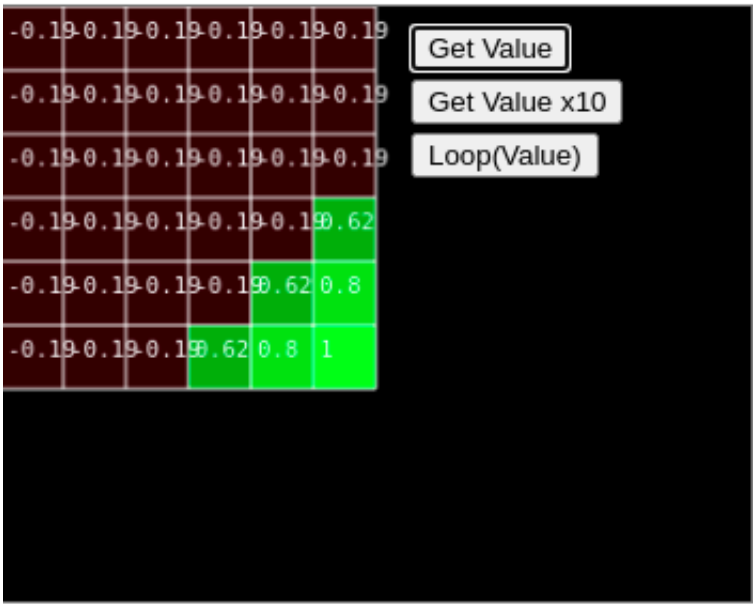
Figura 2.13: primeiro passo $V_{grid} = -0,1 + (0,9 * 1) = 0,8$



Fonte: Elaborado por [45]

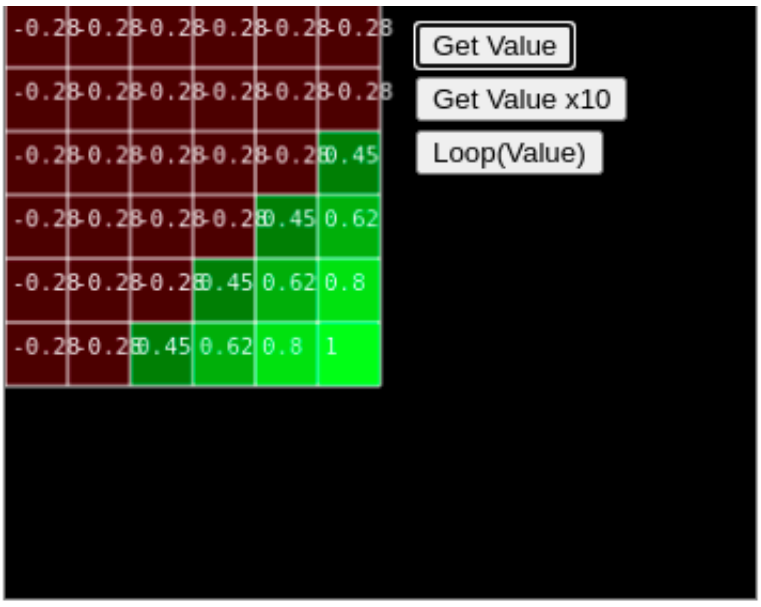
Quando o agente compara o valor das grades vizinhas com o valor da grade em que está atualmente localizado ele se move para a grade com o valor mais alto, se duas ou mais grades vizinhas tiverem o mesmo valor, ele se moverá para uma dessas grades aleatoriamente. Se toda a função de valor do *grid* for calculada, o agente encontrará

Figura 2.14: segundo passo $V_{grid} = -0,1 + (0,9 * 0,8) = 0,62$



Fonte: Elaborado por [45]

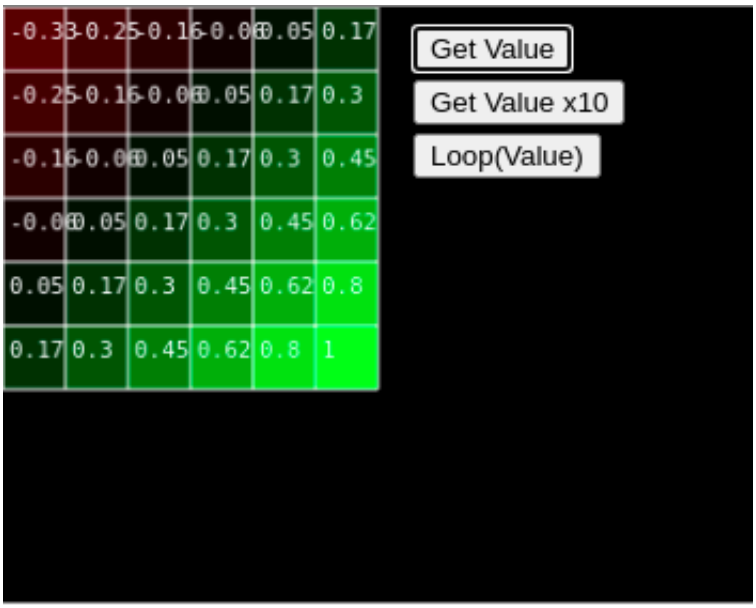
Figura 2.15: terceiro passo $V_{grid} = -0,1 + (0,9 * 0,62) = 0,458$



Fonte: Elaborado por [45]

imediatamente um caminho ótimo.

Figura 2.16: Até completar toda a tabela



Fonte: Elaborado por [45]

O *Grid World* é um dos ambientes mais básicos de aprendizado por reforço, e o cálculo e uso da função valor um de seus algorítimos básicos.

Se estivermos em um ambiente extremamente grande, mesmo sendo fixo, o tamanho da memória para armazenar a função valor também deve aumentar. Em ambientes dinâmicos, teremos uma grande dificuldade para responder, precisando de várias etapas para para calcular a função valor. Em nosso exemplo reduzido necessitamos de 11 etapas para calcular a função valor, no entanto se o alvo se movimenta ou o agente se movimenta, temos de calcular a função valor a cada passo. Para resolver esse problema seria melhor utilizarmos uma entrada de tamanho fixo, que independa do tamanho do ambiente, como por exemplo o campo de visão de um robô.

2.4.2 Q-learning

No aprendizado por reforço o desenvolvimento de um algorítimo de controle TD (*Temporal Difference*) fora da política, ou seja a função *Q-learning* aprende com ações que estão fora da política atual, como executar ações aleatórias, buscando aprender uma política que maximize a recompensa total [86]. Esse foi um dos mais importantes avanços no aprendizado por reforço, que pode ser definido em sua forma mais simples como[80]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t + \alpha)\left[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{2.5}$$

A função valor aprendida aproxima diretamente Q* de uma função valor-ação ótima seja qual for a política que se siga, tendo ainda a politica a função de determinar

quais pares de estado-ação serão visitados e atualizados, e por ser um requisito mínimo para qualquer método encontrar um comportamento ótimo, a tabela Q mostra convergir com probabilidade 1 para Q*[87].

Inicialize $Q(s,a), \forall s \in S, a \in A(s)$, arbitrariamente, e Q(estado-terminal, $\cdot$) = 0
repita (para cada episódio):

Inicialize S

Repita (para cada passo do episódio):

procure A em S usando a política derivada de Q(e.g., /ϵ-greedy)

tome a ação A, observe R, S']

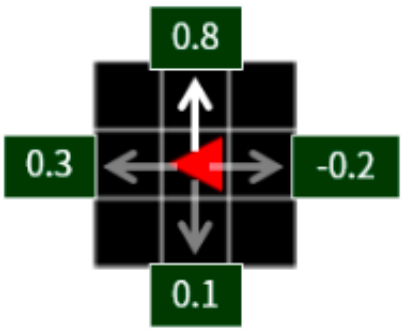
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t + \alpha)\left[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{2.6}$$

S $\leftarrow$ 0 ;

Até S ser terminal[80]

Como vimos, no cenário de *grid*, a função de valor é capaz de substituir todo o espaço de um estado por um valor. Ao compararmos a função de valor com o estado, ao escolhermos o movimento que nos leva a ação de valor mais alto, caminhamos para uma solução do problema. O agente pode se mover nas 4 direções(para cima, para baixo, para a direita e para a esquerda) e função Q vai substituir um estado por uma ação.,verificando o valor desses movimentos para cada estado e escolhe a ação que possa gerar o maior valor de Q para esse estado como mostra a figura 2.17. O Q tem como significado a qualidade de um determinado comportamento , e essa função é também chamado de Política.

Figura 2.17: Q escolhe a ação de maior valor



Fonte: Elaborado por [45]

A função Q vai ser expressa como (Q(s,a)), ou seja tem um valor determinado por seu estado e por seu comportamento nesse estado, como representado na figura 2.17. O

agente seleciona a ação com maior valor de Q, ou alta probabilidade de ser, no caso, o movimento para cima tem o maior valor da função Q. Para atualizar a função Q será levado em conta o valor presente da função Q que é o valor quando a ação(a), no presente estado (s) é executada. Q(s,a) é o valor futuro , quando a ação (a')vir a ser executada no estado (s'), Q(s',a').Temos de considerar também a recompensa R para cada movimento que poderia ser a soma das recompensas com o valor futuro ou seja Q(s,a) vai ser proporcional a função Q futura somada a recompensa Q(s',a')[**?**].

$Q(s,a) \simeq \alpha + Q(s',a')$

Não podemos nos esquecer da taxa de desconto (gama) que será aplicada sempre a uma ação futura, portanto aplicar uma taxa de desconto a essa equação, seria multiplicarmos gama pela ação futura $Q(s,a) \simeq \alpha + \gamma Q(s',a')$. Como a ação (a') futura pode ser executada em qualquer uma das 4 direções , vamos escolher a de maior valor $Q(s,a) \simeq \alpha + \gamma maxQ(s',a)$. Nosso objetivo é fazer com que Q(s',a) fique o mais próximo possível de $\alpha + \gamma maxQ(s',a)$, e a melhor maneira de fazermos isso seria fazer vários episódios, uma média de todos eles, que de forma simples poderia ser somando todas as entradas e dividindo pela quantidade de entradas. A cada vez que for adicionado um novo valor, fazer novamente essa média, com todos os valores existentes, transformando na seguinte expressão alterada, para poder adicionar a diferença entre as médias novas e anteriores, dividida por n.

$average = 1/(n-1)\sum_{i=1}^{n-1} A_i$

$$\begin{aligned} average' &= 1/n(\sum_{i=1}^{n-1} A_i + A_n) \\ &= 1/n((n-1)xi/(n-1)(\sum_{i=1}^{n-1} A_i + A_n)) \\ &= 1/n((n-1)xaverege + An) \\ &= (n-1)/nxaverage + A_n \\ &= average + i/n(A_n - average) \end{aligned} \tag{2.7}$$

Notamos que a fração 1/n vai diminuindo a medida em vamos fazendo as *runs*, então vamos substituí-la por uma constante alfa (0,1 no caso), chamada de taxa de aprendizagem , ajustada para aumentar ou diminuir conforme o numero de tentativas, você pode levar nosso sistema a um melhor aprendizado.

$$average' = average + \alpha(A_n - average) \tag{2.8}$$

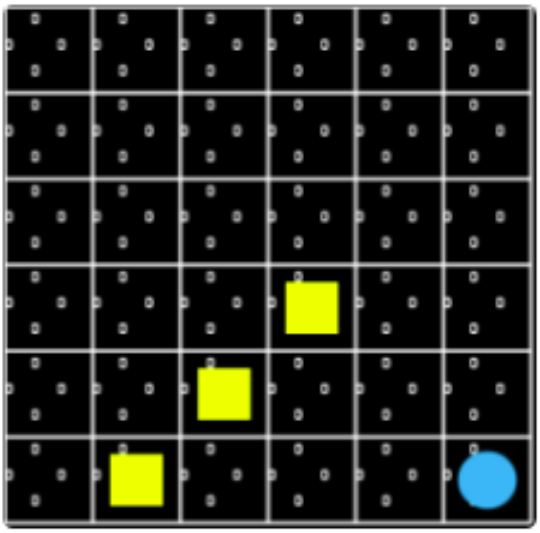
vamos usar esse raciocínio para introduzir a função Q

Equação de Bellman

$$Q(S_t, A_t) = (1 - \alpha)Q(S_t, A_t) + \alpha(R_t + \gamma max_a Q(S_(t + 1), a)) \tag{2.9}$$

Vamos encontrar o valor da função Q em um ambiente fixo, usando um ambiente de *grid* com os 3 pontos de penalidade, $(x = 5, y = 1), (x = 4, y = 2), (x = 3, y = 3)$, quadros amarelos e um objetivo fixo (x = 5, y = 5) bolinha azul. Vamos calcular então a função Q para cada estado sobre todas as ações possíveis, inicializando com o valor zero em todos as ações possíveis em cada estado, pois não exite ainda nenhum valor 2.18.

Figura 2.18: Inicializando com zeros



Fonte: Elaborado por [45]

Para a seleção de uma próxima ação, podemos escolher uma ação aleatoriamente, o que a longo prazo, resultaria em uma exploração uniforme das ações ou políticas, mas com convergência muito lenta ou escolher valores. O agente sempre pode escolher uma explotação de Q previamente aprendido, que resultaria em uma convergência relativamente rápida, porém, alguns caminhos podem permanecer inexplorados até o final. No pior caso poderia fornecer políticas não ótimas, mas uma combinação entre exploração e explotação com uma alta parcela de exploração no início e reduzi-la cada vez mais poderia otimizar todo o processo[25].

Para solucionar esse problema vamos adotar um método de pesquisa chamado Epsilon-greedy [83], para escolhas entre explotação que são as escolhas dos caminhos ótimos pelo agente, ou exploração onde o agente faz escolhas aleatórias, que permitam passar por caminhos que talvez nunca tivesse a possibilidade de passar. ϵ é um número muito pequeno, que vamos comparar com um número randômico entre 0 e 1. Se ϵ for

maior que esse numero, retorna a função Q. Isso seria o equivalente a dizer que se $\epsilon = 0{,}4$ teremos 40% de chance de escolher um comportamento aleatório.//

if *randomNumber* $< \epsilon$*:* **then**
  | getRandomAction()
else
  | getGreedyAction()
end

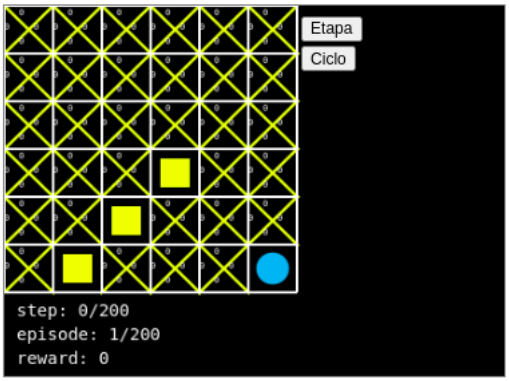
Algorithm 1: Epsilon-greedy

Nesse simulação o agente se movimenta partindo do ponto (x = 0, y = 0) escolhendo sempre o melhor valor de Q futuro para a sua ação, faz o movimento e atualiza a função Q a cada passo , até que atinja o objetivo ou se der 200 passos, o que caracteriza um episódio, com um num total de 200 passos

$$\begin{aligned} Q(s,a) &= Q(s,a) + \alpha(R + \gamma(s',a') - Q(s,a)) \\ &= 0 + -0,1(R + 0,9X0) - 0 \\ &= -0,02 \end{aligned} \tag{2.10}$$

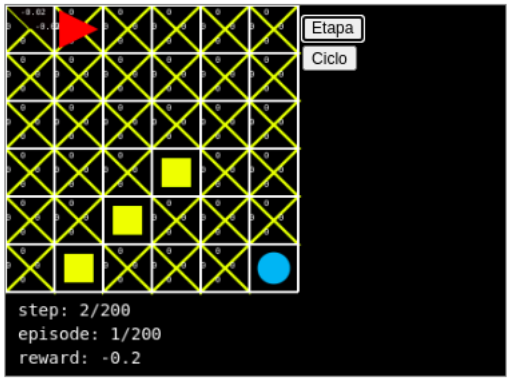
Após essa primeira mudança de estado,é feita a atualização do valor de Q, fazendo o mesmo processo recursivamente a cada mudança de estado, sempre pelo maior valor de Q, considerando ϵ.Figuras 2.19, 2.20 , 2.21, 2.22.

Figura 2.19: Inicializando com zeros



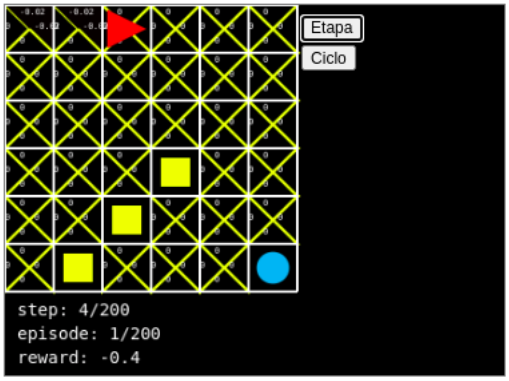
Fonte: Elaborado por [45]

Figura 2.20: inicia a mudança de estado pelo melhor valor considerando ϵ



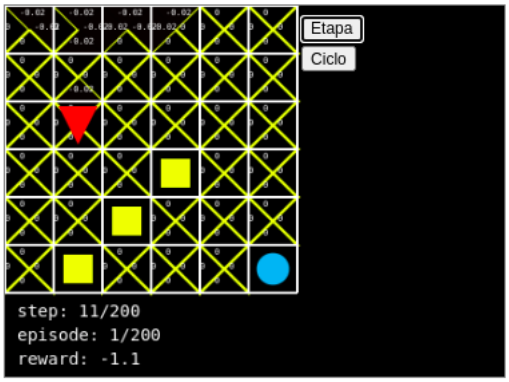
Fonte: Elaborado por [45]

Figura 2.21: Prossegue recursivamente



Fonte: Elaborado por [45]

Figura 2.22: Continua atualizando a tabela

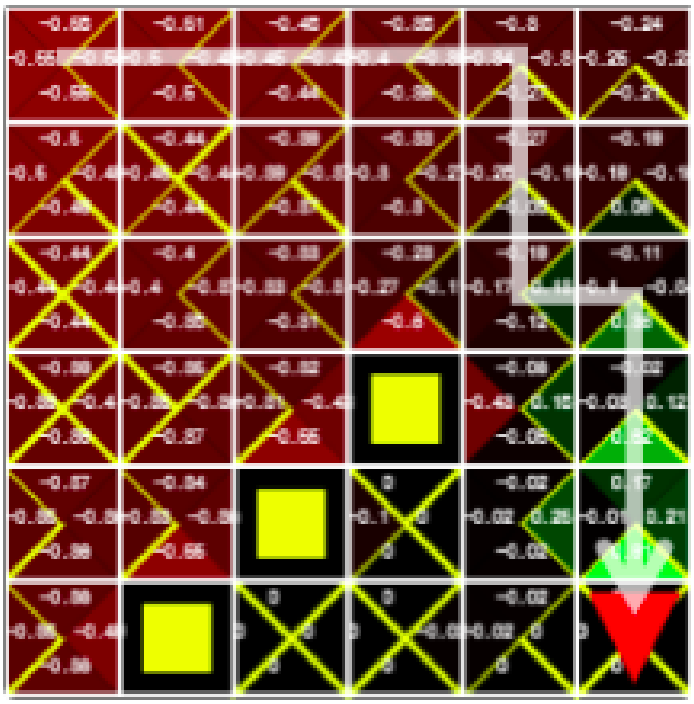


Fonte: Elaborado por [45]

Os triângulos amarelos representam o valor da função e tem uma coloração de fundo degrade do vermelho para o verde, sendo que o vermelho seria o menor valor e o

verde o maio,r como ilustra a figura 2.23. Ao final do treinamento temos uma tabela otimizada em que o agente pode atingir seus objetivos apenas seguindo os triângulos com maior valor de cada estado. Após aproximadamente 100 episódios nesse cenário de simulação o valor de ϵ vai se tornando muito pequeno , e a solução ótima para os valores de Q serão atingido.

Figura 2.23: Até o término do processo



Fonte: Elaborado por [45]

Se pensarmos em um processo estático, preenchendo a tabela Q poderemos obter dessa forma resultados ótimos se esse ambiente não for muito grande. Ao expandirmos esta para um ambiente de grades proporções a quantidade de memória exigida cresceria muito rapidamente. Para solucionar essa questão podemos estabelecer uma premissa básica de manter constante o tamanho do ambiente , independentemente do tamanho do ambiente proposto real, como por exemplo no caso de um robô , limitaríamos o nosso ambiente ao campo de visão desse ator, independente do tamanho do mundo possível de ser utilizado. Este processo nos levaria a um comportamento dinâmico, pois a cada ação, modificaríamos potencialmente todo o ambiente (a paisagem), sendo necessária uma atualização a cada passo, o que nos levaria a um aumento exponencial de memória, desproporcional ao ambiente utilizado naquele passo, tornando inviável o armazenamento em uma função Q como uma tabela.

2.4.3 Deep Q-learning

Volodymyr Mnih demonstrou que uma rede neural convolucional pode superar esses desafios para aprender políticas de controle bem-sucedidas em ambientes complexos de aprendizado por reforço, observados em um jogo de Atari [68], onde a rede foi treinada com uma variante do algoritmo Q-learning, com gradiente descendente estocástico para atualização dos pesos, abordagem aplicada ao Atari 2600 implementados no "The Arcade Learning Environment"[7].

O aprendizado em um ambiente *arcade*, uma plataforma de avaliação para agentes gerais, tem o objetivo de criar um agente de rede neural capaz de aprender a jogar esse jogo com sucesso . A rede não recebeu nenhuma informação específica do jogo, aprendendo apenas com a entrada de vídeo , a recompensa, os sinais terminais e o conjunto de ações possíveis, da mesma forma que um humano aprenderia.

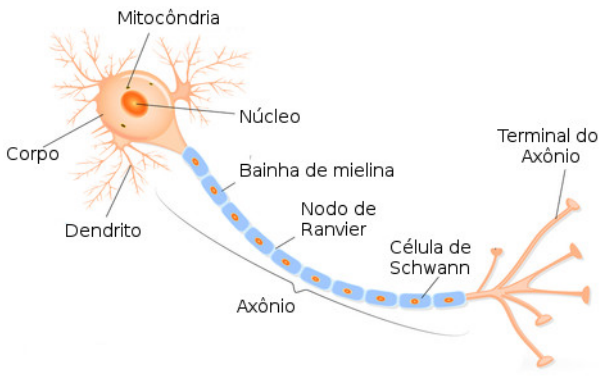
Se até agora num ambiente estático substituíamos o estado por um comportamento através da tabela Q , o DQN faz essa mesma substituição através de uma rede Q profunda.

2.4.4 Redes Neurais

As redes Neurais artificiais são modelos matemáticos baseados na estrutura das redes neurais biológicas. O neurônio é uma unidade básica celular do cérebro humano com a função de transmitir informações.

Constituído de três partes principais, um corpo celular com algumas ramificações denominadas de dendritos, e por uma ramificação mais longa chamada axônio, em cuja extremidade encontram se os nervos terminais, por onde é feita a transmissão de informação para outros neurônios Figura 2.24. Essa transmissão é chamada de sinapse, que é o processo de comunicação entre os neurônios.

Figura 2.24: CEstrutura de um neurônio



Fonte: Elaborado por [3]

Nos neurônios bipolares, os dendritos recebem os sinais elétricos enviados por ou-

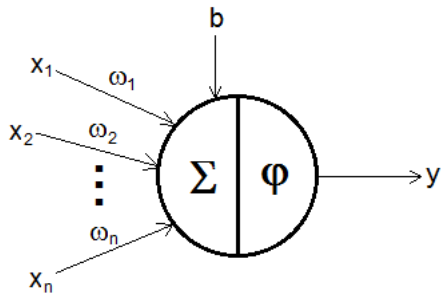
tros neurônios através da sinapse, e enviam sinais através do axônio. Um impulso elétrico é a mensagem que os neurônios emitem.

O corpo do neurônio é responsável por coletar e combinar as informações de outros neurônios. Sinais elétricos enviados pelos sensores do organismo, tais como as papilas gustativas, a retina ocular e o tímpano, entre outros sensores, são enviados aos neurônios. Se esses sensores emitirem sinais superiores a um limiar de disparo, seguem seu caminho pelo axônio, caso contrário são bloqueados. Cada neurônio recebe sinais pelos inúmeros dendritos, e esses podem ser atenuados, amplificados ou bloqueados em cada neurônio.

Esse modelo biológico serviu de base para que pesquisadores desenvolvessem um modelo matemático. Warren McCulloch e Walter Pitts em 1943 [61] em seu artigo *A logical calculus of the ideas immanent in nervous activity*, descreveram um modelo que implementa de maneira simplificada os componentes e o funcionamento de um neurônio biológico. Basicamente calcula uma soma ponderada de várias entradas, aplica uma função e tem um resultado de saída.

Em 1958 Frank Rosemblat, em seu artigo *The perceptron: a probabilistic model for information storage and organization in the brain* [75], propôs um modelo matemático baseado no artigo de McCulloch et tal., conhecido como *Perceptron* de Rosemblat ou simplesmente Perceptron, Figura 2.25.

Figura 2.25: Figura ilustrativa do modelo Perceptron de Rosemblat



Fonte: Elaborado por [18]

O *Perceptron* é constituído como um neurônio não-linear, que possui n entradas x_i . Rosemblat propôs uma regra simples para o calculo das saídas, acrescentar pesos que expressam a importância de cada uma das entradas. A saída do neurônio é 0 ou 1, e determinada pela soma ponderada dos pesos sinápticos do perceptron, representados por $w_1, w_2 \ldots w_n$, e das entradas representados por x_1, e ainda um baias aplicado externamente, representado por b.

$$u = \sum_{n=1}^{m} w_i x_i + b \tag{2.11}$$

Se essa saída u for menor do que algum valor limiar (*threshold*) a saída é 0. Se a somatória for maior ou igual ao *threshold* a saída será 1, desta forma ativando ou desativando o neurônio.

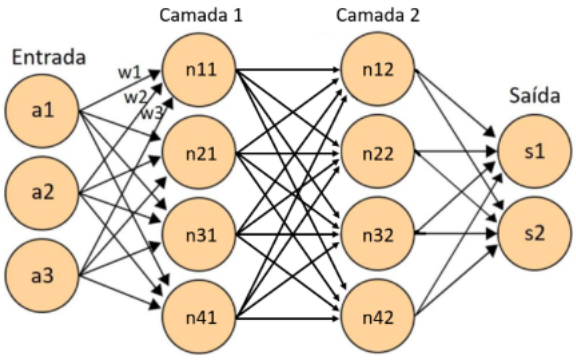
A função f() é responsável por definir a ativação de saída do neurônio, em termos do seu nível de ativação interna. Normalmente o sinal de ativação do neurônio pertence ao intervalo (0, 1) ou (-1, 1). A função degrau é uma das funções mais utilizadas em Perceptron, e determinada por:

$$f(u) = \begin{cases} 0 & \text{se } u < \tau \\ 1 & \text{se } u \geq \tau \end{cases} \tag{2.12}$$

O perceptron é um algoritmo de aprendizado de classificadores binários. Ele é capaz de resolver apenas problemas linearmente separáveis. Por meio do processo de treinamento, o algoritmo aprende a classificar as entradas em dois grupos diferentes.

Em 1986 Rumelhart, Hilton e Williams desenvolveram o algoritmo de treinamento backpropagation [76], mostrando ser possível treinar eficientemente redes de conjuntos de perceptrons organizados em várias camadas, com camadas intermediárias, chamadas de camadas ocultas. Figura 2.26

Figura 2.26: rede de Perceptron com as camadas ocultas



Fonte: Elaborado por [4]

Essa topologia funciona como se fosse uma rede progressiva, onde cada saída de um neurônio vai se comunicar com um ou mais neurônios da próxima camada. Se a conexão for de cada um para todos seria uma rede fortemente conectada.

Cada neurônio recebe todos os valores da camada anterior, no caso de ser uma rede fortemente conectada. Esses valores vão ser multiplicados pelos pesos sinápticos e

somados ao bias como mostrada na equação (2.11).

Logo após esse resultado, o neurônio passa por mais um cálculo com a função de ativação, que pode ser uma função linear , como a função identidade ou degrau mostrada na equação, ou funções não lineares como gaussiana, tangente hiperbólica ou sigmoide equação(2.13) , por exemplo.

$$f(u) = \frac{1}{1 + e^{-u}} \tag{2.13}$$

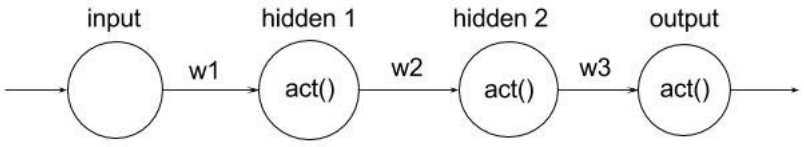
Esse modelo constituiu as redes neurais mais utilizados , capazes de lidar com problemas não linearmente separáveis, adicionando camadas ocultas de neurônios no modelo de Rosenblatt.

Uma rede neural quando inicializada, os neurônios vão receber valores aleatórios. Quando os pesos sinápticos são multiplicados pelos valores recebidos, muitas vezes não atingem valores desejáveis no treinamento. Para corrigir esses pesos sinápticos, usa se a retro propagação, para corrigir os valores dos pesos pela diferença entre os valores recebidos e esperados.

Nesse processo de correção, é introduzido um valor na camada de entrada e outra na camada de saída, o processo caminha entre as camadas fazendo as devidas operações fluindo até a camada de saída, Figura 2.27, processo denominado *feedforward* 2.14

.

Figura 2.27: Diagrama de dupla camada para rede neural



Fonte: [5]

$$\begin{aligned} output &= act(w3 * hidden2) \\ hidden2 &= act(w2 * hidden1) \\ hidden1 &= act(w1 * input) \\ output &= act(w3 * act(w2 * act(w1 * input))) \end{aligned} \tag{2.14}$$

O resultado obtido é comparado com o valor esperado, e caso ele não ative o neurônio de saída, ocorre então o processo de aprendizagem. A rede vai derivando o erro para as camadas internas e corrigindo os pesos sinápticos até a entrada, processo conhecido como *backpropagation* .

Para que se possa realizar os cálculos de *backpropagation*, é necessário que a nossa função de ativação seja diferenciável, para que possa ser calculada a derivada parcial do erro em relação a um dado peso w_i, j, *loss*(E), saída de nó oj e saída de rede j.

Se derivarmos a função aplicando a regra da cadeia com relação ao peso w1,teríamos 2.15:

$$\frac{\partial}{\partial w1} output = \frac{\partial}{\partial hidden2} output * \frac{\partial}{\partial hidden1} hidden2 * \frac{\partial}{\partial w1} hiden1 f(u) = \frac{1}{1+e^{-u}} \quad (2.15)$$

Como já escolhemos uma função de ativação e de erro, temos agora os resultados como um valor numérico , a derivada do erro pode ser usada na descida do gradiente para otimizar os pesos de forma iterativa. Repetimos o processo para todos os pesos da rede,e chamaremos de retro propagação, porque, usamos os erros dos pesos de saída para a atualização dos pesos, antes da próxima etapa. Temos assim que o peso corrigido vai ser, o peso antigo menos a derivada, vezes a taxa de aprendizado, que normalmente é uma constante de pequeno valor.

Se a derivado for < 0, implica em que um aumento de peso aumentará o erro, se a derivada for > 0 implica que um aumento de peso vai diminuir o erro, e se a derivada for igual a 0 não é necessário aumentar os pesos. Fazendo esse processo de treinamento, muitas vezes teremos um modelo que poderá prever novas entradas, ou estabelecer singularidades e agrupamentos.

Capítulo 3

Introdução a computação musical

Fundamentos teórico-práticos, do uso de sistemas e tecnologias computacionais para composição, síntese, performance de sons e música, assim como para educação musical com auxílio do computador, são exemplos de utilização da computação na música.

3.0.1 Histórico

Influenciados pela musica de Webern, Messiaen [26] e pelos ensinamentos de Hebert Heimert, Karlheinz Stockhausen [63] e Pierre Boulez entre outros, desenvolvem um método computacional, onde [32] os parâmetros musicais tem seus valores eletronicamente determinados, gerados por aparelhos eletrônicos, segundo um método serial e registrado em fita magnética. Esse sistema passou a ser conhecida como *Elektronische Music*, ou musica eletrônica pura [65].

Em 1952, foi criado em Köln, na Alemanha, o segundo estúdio de música, que em oposição aos princípios da música concreta, trabalhava exclusivamente por meios eletrônicos, sem a interferência de sons naturais. Karlheinz Stockhausen ,importante compositor contemporâneo[66], foi um dos principais influenciadores desse tipo de música.

Max Mathews, [70] considerado o pai da Computação Musical, desenvolveu no *Bell Laboratories*, em Nova Jersey, o primeiro programa de computador para a música em 1957, em um computador de grande porte. O programa chamado Music I era monofônico e tinha uma forma de onda triangular, não possuía ADSR e só controlava a afinação, intensidade e duração dos sons. Robert Moog [78] criou na década de 60 o primeiro sintetizador de sons*(sintesis)*. A música popular começou a assimilar a música eletrônica de vanguarda, e grupos ingleses de rock progressivo da década de 70 como ELP, Yes e Genesis [41], começaram a utilizar sintetizadores em suas performances. A utilização da orquestra sinfônica, coral e sintetizadores na música popular, foi difundida principalmente por Rick Wakeman, com a obra Viagem ao Centro da Terra, obtendo uma sonoridade ímpar até então na música popular.

Os formatos de áudio de alta qualidade mais usados, Wave e AIFF são do tipo áudio digitalizado sem compressão. Baseiam-se na codificação PCM (Pulse Code Modulation). Outro problema dos arquivos do tipo áudio digitalizado, além do grande volume de dados gerado, é que eles representam a informação sonora e nenhuma informação musical.

3.0.2 Linguagens

Algumas das linguagens utilizadas específicamente para música:

- JMSL (BURK & DIDKOVSKY,2001) Java Music Specification Language (JMSL) [19] ou Linguagem de Especificação Musical Java, é uma ferramenta de desenvolvimento baseada em Java para experimentos em composição algorítmica, performances ao vivo e projeto de instrumentos inteligentes. JMSL é a evolução de HMSL (linguagem de especificação musical hierárquica), baseada em FORTH. JMSL inclui suporte a orientação a objetos, API para rede, gráficos, etc. e a propriedade de rodar em múltiplas plataformas via navegadores da Internet ("Web browsers"). É uma extensão de Java com classes para organização hierárquica de objetos composicionais, geradores de sequência, função de distribuição e outras ferramentas relacionadas à música.

- 4ML é uma linguagem de marcação para música e letra escrita em XML [66]. A ideia é fornecer uma forma independente de plataforma, flexível e simples de se escrever música e letra podendo ser usada por músicos e programadores.

- OpenMusic [66]É um ambiente de programação visual para criação de aplicações, para composição musical, auxiliada por computador, desenvolvido para a plataforma Macintosh, no IRCAM [6] por Gerard Assayag e Carlos Agon

  OpenMusic é essencialmente uma interface visual para a linguagem Lisp, que traz uma variedade de sub-rotinas embutidas, e estruturas abstratas, representadas por ícones.

- JavaSound [28] É uma API do Java que especifica mecanismos para capturar, processar e reproduzir dados de áudio e MIDI, promovendo flexibilidade e extensibilidade. É um mecanismo de 64 canais para renderização de áudio e controlador MIDI de síntese, que oferece confiança e alta qualidade sonora em todas as plataformas Java 2. Também suporta um conjunto de bancos sonoros General MIDI de alta qualidade

- jMAX [66] É um ambiente para programação visual altamente modular, para performances multi-modo e música em tempo-real, desenvolvido pelo IRCAM.

Capítulo 4

Trabalhos Relacionados

A música computacional tem despertado o interesse de muitos pesquisadores nos dias de hoje, e muitos trabalhos relacionados a essas atividades, podem ser encontrados nas publicações cientificas dos últimos anos.

Dentro de um contexto desenvolvido sobre música, harmonia e probabilidade, Briot et al. [11], aborda em seu artigo *Deep learning for music generation: challenges and directions*, um novo modelo probabilístico de progressões harmônicas, com a geração automática dessas progressões, baseado neste modelo. Aplicações diretas de aprendizado profundo para a geração desse tipo de conteúdo, logo atinge limites, e esse conteúdo gerado tende a imitar o conjunto de treinamento, sem exibir criatividade.

Além disso estas arquiteturas não possuem formas de controlar a geração de melodias, como impor tonalidades ou outros tipos de restrições. Tem um comportamento muito mais como autômatos autistas, sem usuários humanos. Relegam aspectos como estrutura, criatividade e interatividade , bem como a visão sobre algumas limitações dessa arquitetura, são o foco da análise desse artigo de Briot et al.

Um sistema de geração baseado em aprendizado de máquina, pode aprender automaticamente em um modelo, um estilo, de um corpus de música. A geração pode então ocorrer usando previsão (por exemplo, para prever o tom da próxima nota de uma melodia) ou classificação (por exemplo, para reconhecer o acorde correspondente a uma melodia), com base na distribuição e correlações aprendidas pelo modelo profundo, que representam o estilo do corpus.

Gaëtan Hadjeres, Frank Nielsen [35], observam que as Redes Neurais Recorrentes (RNNS), tem sido usadas em tarefas de prever sequencias, pela sua capacidade de aprender dependências de longo alcance, e gerar sequências de comprimento arbitrário, mas com um controle limitado, em seu procedimento de geração da esquerda para a direita, permitindo apenas um controle restrito para uma eventual interação de um usuário.

Faz uso de uma arquitetura chamada Antecipação-RNN, que possui os recursos

de modelos generativos baseados em RNN, permitindo impor restrições posicionais de sequências melódicas. Na mesma ordem de complexidade de modelos RNN tradicionais, rápida e interativamente, parecendo apropriadas para futuros desenvolvimentos em tempo real, para fins criativos. Demonstram sua eficiência, na tarefa de gerar melodias, que satisfaçam restrições posicionais, no estilo das partes soprano, das harmonizações dos corais de JS Bach.

A ideia é introduzir uma rede neural para resumir o conjunto de restrições C. Fazendo isso, poderia reescrever o conjunto C como uma sequência $c = (c_1, \cdots, c_n)$ onde $c_i \in A \cup \{NC\}$. Em seguida, introduzir uma RNN chamada *Constraint-RNN* [44] para resumir a sequência de todas as restrições. Este RNN vai para trás (de $c_n a c_1$), e todas as suas saídas, são usadas para condicionar uma segunda RNN, chamada *Token-RNN*. Essa arquitetura é chamada de Antecipação-RNN, já que o Token-RNN está condicionado ao que pode vir a seguir. A Antecipação-RNN, portanto, recebe como entrada uma sequência de *tokens* $(s_0, \cdots, s_{n-1})$ e uma sequência de restrições $(c_1, \cdots, c_n)$ e tem que prever a sequência deslocada $(s_1, \cdots, s_n)$. Esta abordagem seria um primeiro passo para a geração de sequências musicais, submetidas a restrições.

Estes modelos tendem a enfrentar limitações semelhantes, ou seja, eles não fornecem maneiras musicalmente interessantes, para um usuário interagir com eles. Na maioria dos casos esses modelos só podem especificar uma semente de entrada, para condicionar o modelo. Uma vez finalizada,a geração, o usuário só pode aceitar o resultado, não participando ativamente do processo de criação da musica, gerando dificuldade com relação a criatividade desses processos.

A geração nesses modelos Generativos, são realizados geralmente da esquerda para a direita, e essas redes (RNNs) são usadas para estimar o próximo evento musical [37]. Essa modelagem caminhando da esquerda para a direita, parece natural, observando-se que a música transcorre através do tempo, mas não correspondem aos princípios de composição no mundo real, pois geralmente, composições são processos criativos interativos e não sequencias.

Como modelo simplificado, gerando uma melodia que terminaria em uma nota específica, mas gerando uma melodia baseada no estilo aprendido, como propõe Hadjeres et al., em geral não é um problema trivial, mas podem ser resolvidos usando um modelo de Markov[72], mas ainda assim, pode se apresentar como uma solução difícil, quando consideramos um RNN arbitrário.

A fim de resolver questões, levantadas pelo esquema de amostragem da esquerda para a direita, abordagens baseadas na cadeia de Markov Monte Carlo (MCMC)[55], podem gerar melodias convincentes, enquanto acata restrições impostas pelo usuário. Por ter um processo de geração, com a ordem de magnitude mais longa em relação ao modelo

anterior, seria um obstáculo para o uso desses modelos em tempo real. Gerar sequencias enquanto as restrições de usuários são aplicadas, tem uma importância crucial, para a elaboração de sistemas generativos interativos.

A Apresentação de uma arquitetura de rede neural chamada de Antecipação-RNN, proposta por Hadjeres et al., capaz de gerar, no estilo aprendido de um banco de dados, enquanto impõe restrições posicionais definidas pelo usuário, é muito geral e trabalha com qualquer implementação de RNN. Além disso, o processo de geração é rápido, pois requer apenas duas chamadas de função por música, e seria capaz de gerar um estilo aprendido em um banco de dados, enquanto impõe restrições posicionais sugeridas pelo usuário, muito conveniente para aplicações em tempo real.

Usando a música, aprendizado de máquina e interação humano-computador como ferramentas criativas para música, Fiebrink e Caramiaux [27] apresentam uma nova abordagem dos algoritmos de aprendizagem como interfaces homem-computador, mostrando o cruzamento de suas *affordances*, com os objetivos dos usuários humanos, argumentando que a natureza dessas interações, configuram um impacto profundo na usabilidade e utilidade desses algoritmos. Uma visão centrada no ser humano, motiva a discussão desse artigo, tendo em vista o emprego do aprendizado de máquina de maneira criativa.

Um modelo probabilístico, para caracterização da música aprendida a partir de amostras de música, é projetado por Roig et al. [74], extraindo parâmetros musicais automaticamente, como tempo, fórmula de compasso, padrões rítmicos e contornos de altura de notas e padrões rítmicos, os quais são utilizados para caracterizar estilos musicais.

Uma abordagem genética para a geração de composições musicais é apresentada por Matić et al.[59], com o objetivo de compor melodias curtas, representadas por uma matriz de números, com informações sobre das notas e suas durações. Os parâmetros de entrada, determinam o comprimento da composição, tonalidade, número e intervalo de notas permitidas, o número de iterações, critérios para finalizações e assim por diante.[1]

Em experimentos onde as notas estão em um conjunto de duas oitavas, numa escala de sol maior que já define as notas da escala, como um conjunto de notas "boas", e as outras fora de escala como "ruins", mas que não são excluídas. Consoantes perfeitas recebem valor menor que ouros intervalos. O numero de compassos, as médias aritméticas de referência e os desvios de cada compasso individual são definidos e dependendo dessas médias e desvios , obtêm-se diferentes distribuições de intervalos consonantes e dissonantes, obtendo por exemplo soluções bastante agradáveis quando exigem consoantes mais perfeitas no primeiro e quarto compasso, permitindo ainda a liberdade para o aparecimento de outros intervalos nos compassos do meio. As soluções são séries de notas com

[1]Algumas composições obtidas pelo GA podem ser baixadas em http://www.pmfbl.org/matematika/zaposleni/dmatic/files/music.html

durações diferentes e os indivíduos obtidos soam mais como boas improvisações do que composição melódica.

Estes algoritmos numa visão prática, são uma ferramenta com a possibilidade de controlar vários parâmetros, que afetam a qualidade e a forma da composição. O autor ainda sugere, que seria interessante implementar algumas outras meta-heurísticas , para comparações ou hibridizações com Algoritmos Genéticos, pois com ajuste de parâmetros de forma adequada, pode-se investigar como o Algoritmo Genético poderia gerar composições, que pertencem a um gênero musical específico.

Tendo como meta a geração de ritmos, propondo uma arquitetura baseada em RNN, combinação de camadas LSTM e *feed forward* (condicionais), capazes de aprender longas sequências de bateria, sob restrições impostas por informações de ritmo, e uma determinada sequência de contra baixo, Makris et al.[56], propõem uma combinação de diferentes camadas de redes neurais, *feedforward* e recorrente, para compor sequências de bateria com base em métricas, e da execução do contra baixo, para composição de ritmo condicional.

As redes neurais recorrentes (RNNs), têm uma arquitetura semelhante, mas as camadas ocultas são mais sofisticadas, incluindo as conexões recorrentes, usadas para lembrar eventos passados na linha do tempo. *Long Short-Term Memory*(LSTM) é uma forma de RNN inicialmente proposta por Hochreiter e Schmidhuber [77].

A aplicação de RNAs para *Algorithmic (or Automatic) Music Composition* (AMC), traz a vantagem de não ter exigência de conhecimento a priori, como regras e restrição de domínio, pois os RNAs aprendem com o exame de exemplos da entrada, contudo na aplicação de AMC, a capacidade de mapear sequências de entrada, para produzir recursos de música de alto nível, é diminuída sensivelmente.

A arquitetura proposta, possui dois módulos separados para prever o próximo evento da bateria. Um LSTM aprende uma sequências de eventos de bateria consecutivos, e uma camada de *feedforward*, que leva informações sobre a estrutura, e a métrica do movimento de notas do contra baixo, prevendo o próximo evento da bateria, como saída da rede.

Nesse processo, o LSTM tenta prever o próximo passo de forma estocástica. Em cada previsão para o índice de tempo n, a rede gera as probabilidades de cada estado. O baixo e o espaço de entrada métrico é então alimentado no módulo Hidden, em uma camada densa totalmente conectada. Uma camada de mesclagem é usada para obter a saída de cada módulo, que é então passado pela não linearidade *softmax*, para gerar a distribuição de probabilidade da previsão. Durante a fase de otimização, a função de perda *AdaGrad* [22] é calculada, como a média da (categórico) entropia cruzada entre a previsão e o alvo. A camada condicional, permite que as redes LSTM simule humanos em

ambas as tarefas: responder a mudanças em outros instrumentos (por exemplo, no contra baixo), enquanto está em sintonia com certas estruturas métricas.

Esses experimentos segundo os autores, mostraram resultados promissores, ressaltando que sistemas LSTM sem uma camada condicional, aprende e compõe música, independentemente das condições dadas, daí a importância da camada *Feed Forward*. Portanto, um sistema de percussão, baseado em uma arquitetura LSTM simples, não pode ser afetado por mudanças potenciais no contra baixo, o que não é o caso com bateristas humanos. Além disso, a preservação de uma estrutura métrica em sistemas LSTM simples, depende apenas, de sua capacidade de aprender a estrutura métrica dos dados em que são treinados.

Devido aos recentes sucessos, na transferência de estilo usando redes neurais nas artes visuais, pesquisadores tem se voltado, em estabelecer um processo de transferência de estilo musical consistente, apesar de ainda não ser um conceito bem definido, do ponto de vista científico.

Por possuir dificuldades estabelecidas, pelo caráter intrínseco multinível e multimodal da representação da música, muito diferente das imagens, acabam resolvendo problemas que são pertencentes, a uma grande variedade de sub áreas muito diferentes na música computacional. Lidando com vários níveis de representações ao mesmo tempo, acaba por levar a resultados não muito satisfatórios.

Shuqi et al.[16] propõem em seu artigo *Music Style Transfer: A Position Paper*, uma definição mais cientificamente viável, de transferência de estilo musical: transferência de estilo de timbre para som, estilo de performance, e transferência para controle de desempenho e estilo de composição.

Existe ai um problema não trivial no que se refere a estilo musical, pois este termo pode se referir, a quaisquer aspectos da música, como sequência de acordes, orquestração, repetição de motes, tipos de escalas, textura de sons e timbre entre outros.

Essa ambiguidade existe devido ao carácter intrínseco multinível e multimodal da musica pois a mesma pode ser lida, escrita ou executada, e ainda temos de contar com a escrita das partituras, uma interpretação abstrata de nível superior, o som que é uma representação concreta, nível inferior, e controle, que seria uma representação intermediária , muito diferente da representação de imagens.

Como consequência, a maior parte dos estudos, se concentra em apenas um nível ou modalidade de representação musical, tendo assim diferentes interpretações do estilo musical,e sua essência também varia muito, podendo até fazer referência, a problemas que evoluíram de diferentes subcampos da musica computadorizada, como composição algorítmica, ou seja, estamos diante do problema de muitos para um.

Neste documento de posicionamento, os autores fazem uma importante contri-

buição, no que diz respeito a estabelecer uma definição precisa de estilo musical, tendo em vista a singularidade dessa representação. Ter como foco a separação das representações de conteúdo de estilo, seria a chave, para que se consiga desenvolver, algoritmos de transferência de estilo de alta qualidade.

A equipe da Magenta foi premiada, como "Melhor Demo" na conferência *Neural Information Processing Systems*(2016) em Barcelona,[2] feita no *Google Brain*, com a biblioteca Magenta de código aberto, construída com o *TensorFlow*, gerando dados MIDI com modelos de rede neural. Com uma interface fornecendo uma camada de comunicação interativa, estabelecida entre o *TensorFlow* e os dispositivos MIDIs, estruturados para interação entre as composições geradas pela máquina e as executadas por um músico, em tempo real.

Essa *jamsession* entre homem e máquina, foi estruturada em um conjunto de 6 redes neurais: uma melodia básica RNN , uma melodia RNN com *lookback* , uma melodia RNN com treinamento *attention* , uma RNN que foi ajustada com aprendizado por reforço , um RNN de bateria e um RNN polifônico.

Na *2017 International Joint Conference on Neural Networks (IJCNN 2017))* em Ancoragem, foi realizada a *International Workshop on Deep Learning for Music*, onde foram apresentados significativos avanços, no estado da arte em inteligência de máquina, para música e aprendizado profundo, com a discussão de pesquisas de ponta nessa área, identificando metodologias eficientes, potenciais aplicações futuras e grandes desafios. [3]

[2]https://magenta.tensorflow.org/2016/12/16/nips-demo

[3]http://dorienherremans.com/dlm2017/

Capítulo 5

O RL Tuner

Uma das formas de se gerar música, no caso melodias , com uso de inteligência artificial, dá se com o uso de modelagens gerativas de musica com redes neurais profundas, como uma rede *Long Short-Term Memory* (LSTM) que vai fazer uma previsão de uma próxima nota nessa sequência melódica que está sendo construída, com uso de máxima verossimilhança [24] por exemplo.

Ao se inicializar esse sistema com uma pequena sequência de notas, essas Note RNN podem ser usadas para gerar novas melodias, ou seja, repetidas amostras oriundas da saída dessa distribuição geradas pelo modelo, vão gerar a próxima nota.

Embora em uso atualmente em muitos trabalhos, esse modelo acaba gerando em muitos casos melodias sem sentido, pois não possuem estruturas bem definidas que possam criar situações musicais interessantes, como sugere Natasha et al. [39].

Elementos estruturais da composição como repetição de motes, conservação de tonalidade, uso de notas de passagem, coerência com o processo harmônico, andamento adequado a cada tipo de melodia, repetição de notas entre outras estruturas, não são incorporadas ao sistema.

No artigo *Tuning Recurrent Neural Networks With Reinforcement Learning*, Natasha et al. [39], propõem uma abordagem de aprendizado usando como base um corpus treinado com um LSTM, com uma rede Q profunda (DQN), que usa uma função de recompensa modificada, composta por dois elementos, uma recompensa baseada nas regras musicais e outra baseada na probabilidade de saída dessa Note RNN treinada.

Essa função objetivo pode ser relacionada, com um controle ótimo estocástico(SOC) e derivar mais dois métodos fora da política, para o refinamento do RNN, através da penalização da divergência KL [49] com a política original. Uma comparação entre a saída do modelo RNN e a saída do DQN. Nesse processo o modelo aprende a se utilizar de um conjunto de regras da música, mantendo a probabilidade do aprendizado original dos dados de treinamento, ou seja, sem que se perca o conhecimento adquirido no aprendizado

com os dados.

Os autores demonstram a capacidade desse modelo produzir, com exito, composições mais variadas, interessantes e melodicamente mais consistentes do que as produzidas pela saída da *Note RNN*.

5.1 Geração de musica com RNN/LSTM

Na geração de musica usando *deep learning*, o treinamento de um RNN aprende a prever uma sequência de notas. Modelos que são treinado para prever a próxima nota de uma melodia monofônica, serão chamados de *Note RNN*. Em muitas situações o *Note RNN* vai ser implementado usando uma rede LSTM [31], que são redes [77] onde cada célula aprende a controlar o armazenamento de informações, através do uso de uma porta de entrada, uma porta de saída e uma porta de esquecimento. As duas primeiras controlam se a informação é capaz de fluir para dentro e para fora da célula, e a última controla se o conteúdo da célula deve ou não ser redefinido. Devido a essas propriedades, os LSTMs são melhores em aprender dependências de longo prazo nos dados, e podem se adaptar mais rapidamente a novos dados [1].

Trabalhos anteriores com geração de música usando aprendizado profundo, como shaleen et al. *Efficient music auto-tagging with convolutional neural networks*[8] e Heaton et al. em *Deep learning* [36], envolveram o treinamento de um RNN para aprender a prever sequências de notas. Eck et al. em seu trabalho *Finding Temporal Structure in Music:Blues Improvisation with LSTM Recurrent Networks* [23], já se utiliza de aprendizado profundo para a geração de musica mostrando que o LSTM é um bom mecanismo para esse aprendizado. Mesmo carecendo de coerência global, apresentam resultados experimentais que mostram com sucesso o aprendizado, com um tipo de blues capaz de compor melodias nesse estilo, demonstrando que o LSTM é capaz de compor blues com uma estrutura adequada.

Sturm et al. em seu artigo *Music transcription modelling and composition using deep learning* [79], treinam um RNN para aprender a prever a próxima nota em uma melodia monofônica, aplicando métodos de aprendizado profundo especificamente LSTM, para modelagem e composição de transcrições de musicas e usá-las para gerar novas transcrições.

O treinamento desse LSTM foi feito usando um *softmax cross-entropy loss* e um *back propagation* [34].

Para gerar melodias a partir deste modelo, primeiramente, ele vai ser inicializado com uma sequência curta de notas, e a cada passo de tempo, a próxima nota será escolhida por uma amostragem da distribuição de saída da camada *softmax* do modelo, e essa nota

será realimentada na rede como entrada do próximo passo de tempo. As melodias geradas por esse modelo, no entanto, muitas vezes flutuam sem sentido e carecem de estrutura musical. Nessa etapa, o objetivo dessa arquitetura, seria o de impor regras da estrutura da teoria da musica, usando aprendizado por reforço.

5.2 Como Atua o Aprendizado por Reforço Nessa Arquitetura

Em um aprendizado por reforço, baseado em uma tabela Q , $Q(S_t, a_t)$ onde 'S' é o estado e 'a' seria a próxima ação a ser executada, o agente vai reagir com o ambiente. O estado seria a composição existente até aquele instante, no tempo t, s_t. O agente vai executar uma ação a_t de acordo com uma determinada politica $\pi(a_t/s_t)$, que determina a próxima nota seguindo uma tabela Q que mostra a melhor escolha para aquele momento, e receber uma certa recompensa $r(s_t, a_t)$, fornecido pelas regras musicais combinadas com o aprendizado RNN. Esse meio ambiente vai se deslocar para um novo estado s_{t+1}, compondo a próxima nota da melodia.

As técnicas de *Q-Learning* [86] [81] aprendem essa função Q ótima de forma iterativa, minimizando o resíduo de Bellman.

A política ótima é dada por $\pi^*(a|s) = argmax_a Q(s, a)$. Esse *Deep Q-learning* [67] usa uma rede neural chamada *Deep Q-network* (DQN) para aproximar a função $Q(s, a; \theta)$. Os parâmetros θ da rede são aprendidos aplicando se um *Stochastic Gradient Descent* (SGD) [10] atualizados segundo a seguinte função de perda (Loss)5.1

$$L(\theta) = E_\beta[(r(s, a) + \gamma max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \tag{5.1}$$

onde β é a politica de exploração e θ^- são os parâmetros da *Target Q-network* que serão mantidos fixos durante o calculo de gradiente [67]. A media móvel θ é usada como θ^-como proposto por Lillicrap et al. [51]. A exploração pode ser feita tanto com *e-greedy method* ou *Boltzmann sampling*. Técnicas adicionais como *Replay Memory* [67] e *Deep Double Q-learning* ([84], também foram utilizadas para estabilizar e melhorar a aprendizagem.

5.3 A Teoria da Música Usada Como Regras de Recompensa

A questão da capacidade de uma RL, poder ser usada para receber uma *Note Sequence* e fazer com que essa sequência, possa aderir a uma estrutura desejada, fez com

que os autores desenvolvessem várias regras de estrutura musical, baseadas em um texto sobre composições melódicas de Gauldin[30].

Tendo como único propósito, estabelecer algumas regras estruturais, orientando o modelo para uma estrutura de composição melódica, baseadas na teoria do contraponto do século XVIII, com o objetivo de investigar, como restrições baseadas em teoria da musica, podem ser usadas para melhorar o desempenho da geração de música, construída por um RNN.

Definiram a função de recompensa $r_M(a, s)$, para encorajar as composições a terem algumas características específicas: as notas adicionadas devem pertencer a mesma tonalidade, a melodia deve ser iniciada pela nota tônica da tonalidade escolhida, se dó maior, começar e terminar pela nota dó. Uma única nota não deve ser repetida mais do que 3 vezes, a não ser, no caso em que ela seja ligada a outra nota igual, o que aumentaria o seu tempo de duração, ou se for colocado pausas entre elas. Se a composição tiver uma alta correlação consigo mesma, mesmo com atraso de 1, 2 ou 3 tempos, o modelo deve ser penalizado, para o incentivo da variedade, e aplicada uma penalização, toda a vez que o coeficiente de correlação for maior que 0,15. A composição deve evitar saltos de sétima maior ou maiores que uma oitava, sendo que Gauldin enfatiza, que boas composições dentro dessa estrutura, devem se mover por graus conjuntos e pequenos intervalos, com interferência de intervalos harmônicos de maior amplitude. Contudo ao se registrar intervalos maiores do que uma quinta justa, deve ser resolvido com um salto no sentido oposto ou um movimento gradual, também no sentido inverso ao salto, sendo que dois saltos na mesma direção será recompensado negativamente. O ponto culminante dessa melodia ou nota mais alta deve ser única, assim como a mais baixa, exceto se forem as inicias e finais. O modelo será também recompensado por compor motivos, ou seja, uma sucessão de notas que representem uma ideia musical curta, visto que a repetição de motivos, é um forte elemento para uma correlação emocional, dentro de uma melodia com demonstrado por Livingstone et al. [53].

Essa regras aplicadas pelos autores,,entre outras, não são necessariamente uma escolha ótima ou única, entre possíveis regras musicais, mas servem ao propósito de guiar esse experimento para uma estrutura bem definida, de forma satisfatória.

5.4 O Refinamento do RNN com Aprendizado de Reforço

A partir de uma *Note RNN*, treinada, a próxima etapa seria ensinar as regras e conceitos da teoria da música, de forma tal, que apesar de impor recompensa específicas de domínio com base nas sequências geradas, mantenha as informações originais aprendidas

no corpus.

Para a realização dessa tarefa , os autores propõe um novo método de treinamento, baseado em aprendizado por reforço, o *RL Tuner*. Usando um *Note Sequence* treinado em um grande corpus de 30.000 músicas, para fornecer os pesos iniciais para três redes em nosso modelo: a *Q-network* e a *Target Q-network* no algoritmo DQN como descrito anteriormente. Uma *Reward RNN* de recompensa, será mantida fixa durante o treinamento, e usada para fornecer parte do valor da recompensa, que vai ser somada ao aprendizado por reforço, sobre o domínio das regras musicais estabelecidas,. Vai ser usada para treinar o modelo, e tratada como uma política prévia, que pode fornecer a probabilidade de um dado *token*, em uma sequência originalmente aprendida, como mostra a Figura 5.1

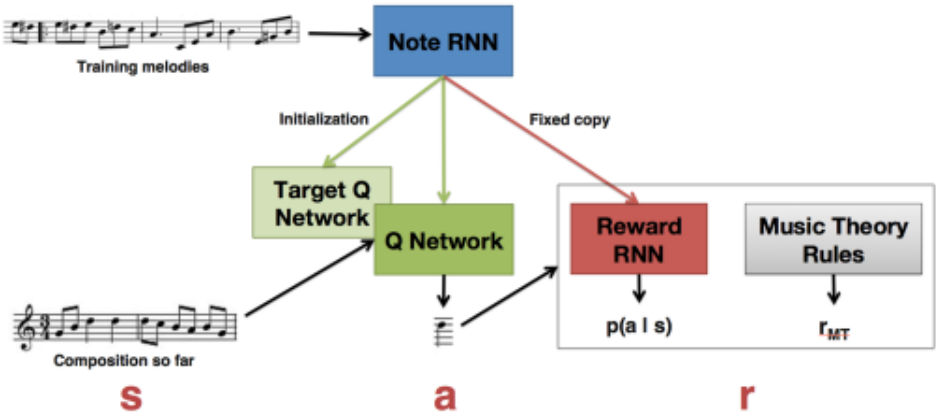


Figura 5.1: Um Note RNN é treinado e fornece os pesos iniciais para a rede *Q-network*, *Target-Q-network* e pesos finais para o *Reward RNN*

Fonte: Elaborado por [39]

O treinamento do corpus foi previamente treinado em três versões,*Basic, Lookback e Attention* de livre escolha na execução, e serão tratados para efeito de experimentos, como uma caixa preta.

Para treinar o modelo, Natasha e colegas extraíram melodias de um corpus de 30.000 músicas MIDI, e codificando-as como sequências de notas textitone-hot. Essas melodias são então usados para treinar um LSTM com uma camada de 100 células.

A otimização foi realizada com Adam [46], um tamanho de lote de 128, taxa de aprendizado inicial de 0,5 e um decaimento gradual da taxa de aprendizado de 0,85 a cada 1.000 passos. Os gradientes foram cortados para garantir que a norma L2 fosse menor que 5, e a regularização de peso foi aplicada com $\beta = 2,510^{-5}$.

Finalmente, as perdas para as primeiras 8 notas de cada sequência não foram usadas. O RNN treinado eventualmente obteve uma precisão de validação de 92 % e uma pontuação de perplexidade log de 0,2536.

Este modelo foi usado como descrito acima, para inicializar as três sub-redes no modelo *Sequence Tutor* . Esse modelo com tamanho de lote de 32, e um fator de desconto de recompensa de $\gamma = 0.5$. Os pesos da rede *Target-Q* θ^- foram gradualmente atualizados para aqueles da rede Q (θ) de acordo com a fórmula: $(1 - \eta)\theta^- + \eta\theta$, onde $\eta = .01$ é a taxa de atualização da *Target-Q-network*.

Um ponto forte desse modelo é a influência dos dados e recompensas específicas de tarefas, que podem ser explicitamente controladas ajustando o parâmetro de temperatura c. Eles replicaram os resultados para uma série de configurações para c; utilizaram os resultados para c=.5 abaixo porque acreditamos que eles são os mais musicalmente agradáveis [1].

Para que a composição musical possa ser formulada como um problema de aprendizado por reforço, vamos colocar a próxima nota da composição como uma ação, e o estado do ambiente s vai ser composto tanto pelas notas colocadas na composição até aquele instante t, quanto ao estado interno das células LSTM. O *Q-network* quando comparado ao *Target Q-network* por um controle estocástico (*KL_Control*) para decidir se coloca arbitrariedade no processo para inserir na rede. Juntamente com a recompensa RNN, combinado com as probabilidades aprendidas, com os dados de treinamento obtidos no aprendizado da teoria musical;

Foi definido um conjunto de regras baseadas na teoria da música, para impor restrições à melodia que o modelo está compondo. Se uma nota estiver fora da escala proposta, ou uma sequencia de saltos melódicos seguidas ou se uma mesma nota tiver muitas repetições seguidas, por exemplo, iriam impor restrições a melodia que o modelo está compondo, receberiam uma recompensa negativa, ou seja, penalizada através de um sinal de recompensa $r_{MT}(a, s)$.

Ainda assim, permanece a necessidade do modelo ser "criativo", e não simplesmente aprender uma composição que poderia explorar essas regras, portanto ainda vai ser usada a recompensa RNN ou a nota treinada RNN para uma composição, s para calcular $log\ p(a|s)$, ou seja, a probabilidade logarítmica de uma nota a dado a composição s, e incorporá-las a função de recompensa. Em alguns momentos as regras da teoria da música poderiam ser simplesmente desprezadas, gerando desvios inesperados que seriam considerados como atitudes fora do contexto pre-fixado, estabelecendo um carácter original, ou criativo.

A combinação das probabilidades aprendidas com os dados de treinamento, e o conhecimento de teoria musical, serão então usados para o calculo da recompensa.

[1] resultados adicionais estão disponíveis em https://goo.gl/cTZy8r

A recompensa total dada no tempo t é, portanto dada pela equação 5.2

$$r(s,a) = logp(a|s) + \frac{1}{C} r_{MT}(a,s) \tag{5.2}$$

onde C é uma constante que controla a enfase dada a teoria musical

Dada a equação da *loss function* do DQN 5.1 e a equação de recompensa modificada 5.2 , a nova função de loss 5.3 e a política de aprendizado vão ser dada pelas equações a seguir:

$$L(\theta) = E_{\beta}[logp(a|s) + \frac{1}{C} r_{MT}(a,s) + \gamma max_{a'} Q(s', a'; \theta^{-}) - Q(s,a;\theta))^2] \tag{5.3}$$

$$\pi_{\theta}(a|s) = \delta(a = max_a Q(s,a;\theta)) \tag{5.4}$$

Com essa modificaçãoo na função de perda, o modelo vai ser forçado a aprender que as ações mais valiosas, são aquelas que estão de acordo com as regras musicais mas ainda com altas probabilidades sobre os dados originais treinados no RNN.

Capítulo 6

Inserção de Sentimentos Emotivos em Composições Musicais com *Rl_tuner*

Para que possamos estabelecer um procedimento de manipulação de parâmetros e hiper parâmetros , bem como a manipulação de funções , precisamos inicialmente estabelecer uma série de desenvolvimento de testes sobre o modelo proposto por Natasha Jaques et al.

Nossos experimentos foram desenvolvidos em um ambiente Colab na linguagem Python versão 3.7.15, com a biblioteca Magenta versão 2.1.4 , Matplotlib versão 3.1.1, o Tensorflow 2.9.1 .Como o experimento original foi efetuado com com o Tensorflow versão 1.5 já descontinuado, usamos o adaptador Tensorflow.compat.v1 e o Pretty-midi versão 0.2.9. Todas as partituras foram construídas e editadas pelo *softer Music Score 3*.

Buscamos primeiramente reproduzir o experimento da forma como proposto por Natasha Jaques et al.[40], reproduzindo um contexto de equilíbrio de recompensas entre os dois componentes do aprendizado por reforço, a recompensa RNN e a recompensa da somatória de todas as regras da teoria musical propostas.

A primeira interferência consiste em excluir a recompensa obtida pelos RNN, para entendimento da atuação das regras musicais como aprendizado por reforço, isoladamente.

A segunda constituiria no isolamento do reforço RNN, ou seja desenvolver um treinamento onde o aprendizado por reforço seria exclusivamente as próprias recompensas do treinamento RNN.

Após esse entendimento, precisamos fazer modificações nos parâmetros de duas funções básicas, no estabelecimento de tonalidades e notas de maior importância na construção de melodias. Essas caracterizações estão nas funções de recompensa da tônica da escala proposta(função *reward_tonic*), e a própria tonalidade (função *reward_key*).

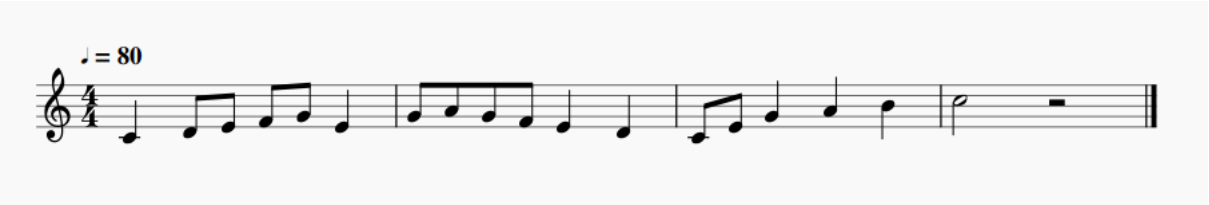
6.1 Reprodução do experimento original

Para a reprodução do experimento o riginal, usamos uma função, que coleta a somatória das recompensas da teoria musical, que escolhermos para participar do treinamento. Neste caso, todas as funções da teoria da música do código original, pois estamos interessados em usar neste momento, todas as regras para entendermos a amplitude de seus efeitos na composição.

Nessa arquitetura, a recompensa dada a cada ação, é composta pela soma da recompensa RNN, com a somatória de todas as recompensas da teoria da música. Um valor de recompensa se a nota em questão estiver dentro da escala, somada a uma recompensa se não for uma nota repetida continuamente, somada a uma recompensa pelo tipo de intervalo que ela tem com a nota anterior, e assim por diante. Essa somatória será multiplicada por um fator chamado *reward_scaler*, que será responsável pela enfase dada a recompensa do conjunto das regras musicais. Note que por se tratar de um somatório, se usarmos uma só regra teremos um somatório muito menor do que quando usarmos todas as regras, por exemplo. Podemos fazer uma compensação com esse fator, *reward_scaler* se necessário, visto que as recompensas RNN seguem sempre um mesmo padrão. Esse fator pode também ser usado, para ampliar certas configurações, ao fazermos testes de interferência dessas regras. Usamos esse artifício, para não ser necessário fazer treinamentos muito longos e demorados, toda vez que formos verificar como alguma mudança de parâmetro, poderia atuar no contexto, visto termos muitas variáveis nas regras da teoria da música.

Neste experimento em particular, vamos usar um fator *reward_scaler* com valor 1.0 e um arquivo MIDI chamado *'exp_origen.mid'* que escreví, de forma a estabelecer claramente uma melodia simples em Dó maior e com 4 compassos, como mostra a Figura 6.1 para preparar o modelo construído.

Figura 6.1: melodia do arquivo MIDI *'exp_origen.mid'*



Fonte: [58]

Esse experimento foi treinado, com apenas um milhão de passos de treinamento, devido a insuficiência de recursos computacionais, e o *exploration_steps* estipulado em quinhentos mil. Apesar do experimento original ter sido treinado com três milhões de passos de treinamentos, obtivemos resultados consistentes, operando com esse valor.

Estatisticamente em um universo de 100 composições, podemos destacar alguns dos resultados, com relevância para validar a experiência:

- 7 notas fora da tonalidade, um ótimo índice que não deveria ser absoluto, pra não ser extremamente rígido.
- apenas 4 notas continham repetições de motivos, que seriam cópias de trechos de 3 ou 4 notas anteriormente composta na melodia, caracterizando diversidade.
- 94 melodias iniciadas com a tônica, a primeira nota da escala, confirmando a definição da tonalidade em 93 % dos casos.
- Apenas 2 saltos melódicos superiores a uma oitava, com boas resoluções de saltos melódicos, uma das regras de penalização muito bem atendida.
- Uma quantidade significativa de intervalos conjuntos, que são intervalos de segundas (2531 intervalos) ou terças, em um total de 6400 notas compostas, denota fluidez nas melodias.

Os resultados demonstram no conjunto, excelente utilização das regras da música, com algumas atividades fora do contexto o suficiente para ser criativo, fora das estruturas, sem perda dos conhecimentos adquiridos no treinamento RNN, como mostra a Tabela 6.1.Todos essas referências são proposições das regras originais.

Tivemos um equilíbrio entre as recompensas da teoria da música e as recompensas do RNN como mostra a Figura 6.2, onde a linha amarela é a recompensa atribuída a teoria da musica, a linha verde corresponde recompensa RNN e a linha azul representa o valor final das recompensas atribuídas a cada passo de treinamento.

Esse equilíbrio entre as duas recompensas, e o fato das recompensas da teoria terem uma amplitude muito maior, acaba por refletir esse comportamento no valor resultante, parecendo uma cópia deslocada no eixo y. Esse gráfico mostra também que, se fizermos 3 milhões de passos de treinamento provavelmente estas curvas tenderiam a se equilibrar ainda mais, pelo que podemos observar na convergência dos valores na medida em que o treinamento vai se desenvolvendo.

Devemos levar em conta, que em se tratando de composição musical, onde, se tivermos apenas uma ou duas notas fora do contexto já poderia caracterizar um melodia mal desenvolvida. Por esse motivo a convergência deve ser vista com certo cuidado, pois se em mais 2 milhões de passos de treinamento fizermos essa correção, já seria plenamente válido.

Para termos uma visão geral das probabilidades de cada nota, temos plotado um gráfico com um eixo horizontal que representa, o posicionamento dos 64 *beats*, ou tempos

.

Tabela 6.1: Tabela de estatística do resultado do treinamento, 1 milhão de passos de treinamento

Tipo de ocorrência	**Valor**
notas fora da tonalidade	7
notas nos motivos	5294
notas nos motivos repetidos	4
composições iniciadas com a tônica	94
quantidade de notas repetidas	4
saltos de oitava	2
intervalos de quintas	96
intervalos de sexta	41
intervalos de segunda	2531
intervalos de quartas	259
intervalos de sétima	3
notas ligadas	0
notas ligadas em intervalos especiais	0
intervalos especiais na tonalidade	83
saltos resolvidos	83
saltos duas vezes seguidas	10
notas mais agudas únicas	48
notas mais graves únicas	43
composições	100
total de notas em todas as composições	6400

Fonte: [58]

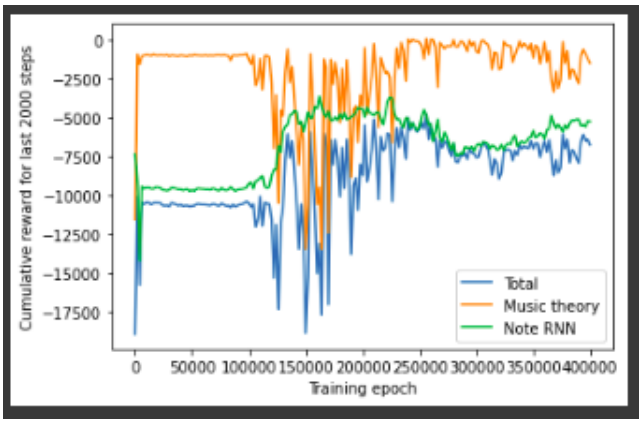
mínimos de execução de cada nota na composição, e no eixo vertical, o valor dos elementos pertencentes ao conjunto das notas, entre 0 e 36, com a intensidade de vermelho definindo a probabilidade de cada nota em cada tempo, como mostra a Figura 6.3.

Das 9 sequências geradas, que são as melodias compostas pelo modelo, foram escolhidas duas para servirem de exemplo demonstrativo dos resultados obtidos. Essas duas melodias estão representadas abaixo, em um pentagrama Figuras 6.4 e 6.5

Todas as melodias observadas estão bem consistentes, dentro da tonalidade, iniciando a execução pela tônica e observando demais propriedades como anteriormente mencionadas, com composições subjetivamente bem desenvolvidas.

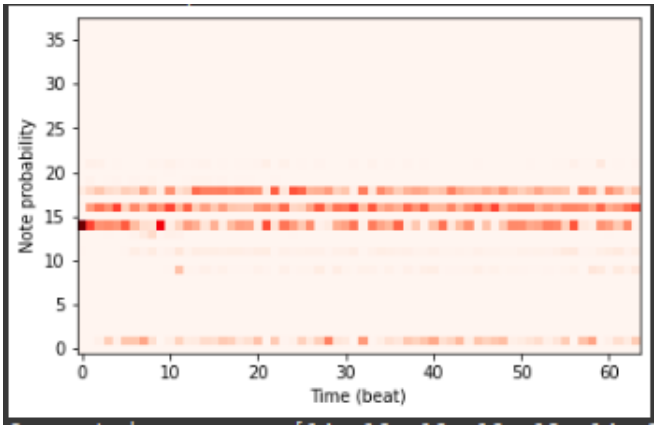
Devemos ainda observar que toda a construção do modelo, no que tange a tonalidade, é referente a tonalidade de Dó maior, contudo, verificamos que existe uma grande tendência a desenvolver melodias na tonalidade de Ré maior.

Figura 6.2: Gráfico das recompensas do RNN, da teoria da música e resultado final



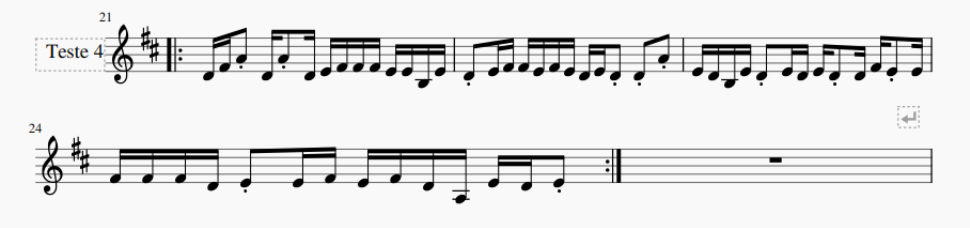
Fonte: [58]

Figura 6.3: Probabilidade de cada nota



Fonte: [58]

Figura 6.4: Melodia teste 4

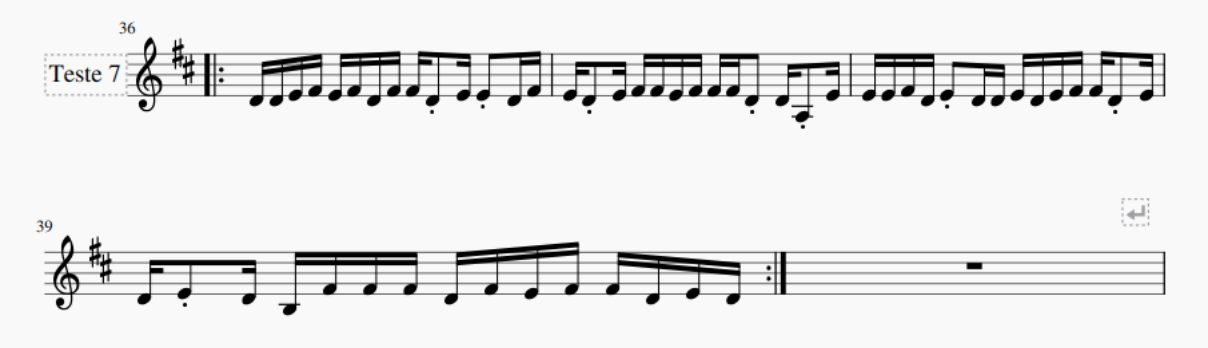


Fonte: [58]

Pela observação do código, verificamos que nos hiper parâmetros C_MAJOR_SCALE = [2, 4, 6, 7, 9, 11, 13, 14, 16, 18, 19, 21, 23, 25, 26] localizado em rl_tuner_ops.py, seria a construção da escala de Dó Maior.

No vetor C_MAJOR_KEY = [0, 1, 2, 4, 6, 7, 9, 11, 13, 14, 16, 18, 19, 21, 23, 25, 26, 28, 30, 31, 33, 35, 37], temos na primeira posição do vetor, o número 0 e na segunda posição o número 1. Esses números são atribuídos a pausas e continuidade da nota anterior,

Figura 6.5: Melodia teste 7



Fonte: [58]

respectivamente, e como a nota inicial é dada como MIDI 48 (INITIAL_MIDI_VALUE = 48), e essa nota é um Dó, quando definimos o hiper parâmetro C_MAJOR_SCALE, o numero 2 da primeira posição vai ser a posição 50 no MIDI, que é um Ré. Isso nos daria um deslocamento de duas posições para todas as notas da escala. Esse fato nos leva ao entendimento de ser esse o motivo pelo qual, todas as composições entendem o Dó maior como Ré maior, como se fosse um instrumento de transposição de 2 semitons. Trata-se apenas de uma observação constatada, pois como foge a nossos objetivos, sugerimos uma investigação futura sobre este tópico.

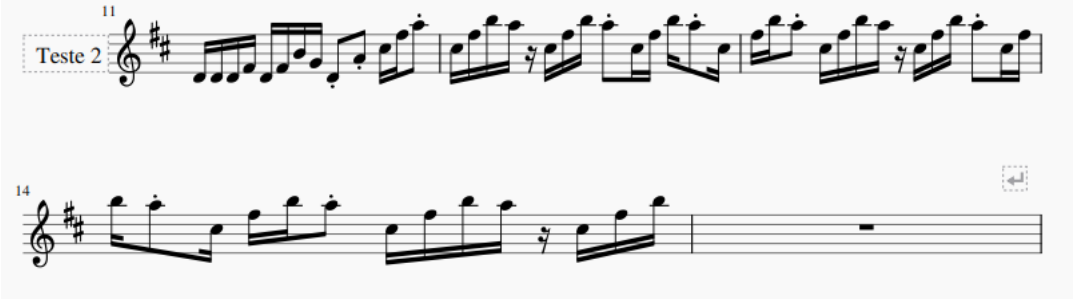
6.2 Experimento Usando Apenas as Regras Musicais Como Aprendizado de Reforço

Para termos uma avaliação de quanto a teoria da música, como regra para o aprendizado por reforço, poderia interferir na construção de melodias pelo modelo que estamos estudando, vamos propor o uso de uma função denominada *music_theory_only*, que aglutina todas as regras implementadas no modelo proposto pelo autor, sem se utilizar do reforço referente a saída RNN, para resolvermos este problema de entendimento.

Em um primeiro teste com um milhão de passos de treinamento, notamos que rapidamente o treinamento encontra uma otimização das recompensas, e interrompe o aprendizado. Um motivo musical ótimo, ou seja, um fragmento de melodia que atinge uma recompensa ótima, entra em *looping*, e passa a repetir sempre o mesmo trecho melódico, pois está em um estado ótimo, como mostrado nos pentagramas das Figuras 6.6 e 6.7.

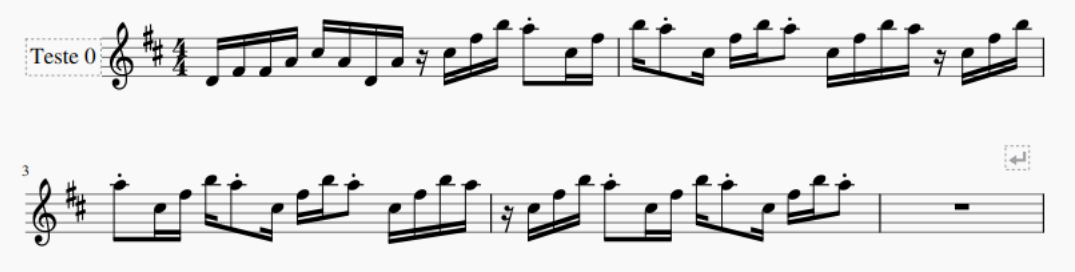
Podemos verificar que a atuação da recompensa RNN, está sem efeito praticamente, por se encontrar perto de zero, enquanto as recompensas da teoria da música encontram-se positivas e elevadas, devido a otimização dos valores de recompensa mostrado na Figura 6.8.

Figura 6.6: Melodia teste 2, para observação das repetições melódicas



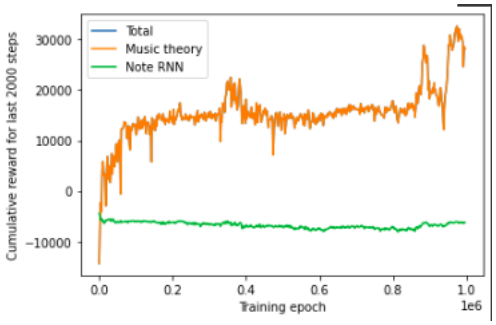
Fonte: [58]

Figura 6.7: Melodia teste 0, para observação das repetições melódicas



Fonte: [58]

Figura 6.8: Treinamento só com a recompensa da teoria da música, com1 milhão de passos de treinamento.

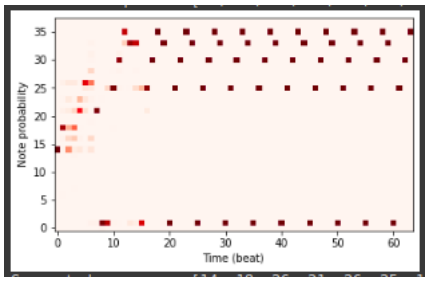


Fonte: [58]

No gráfico referente as probabilidades de cada nota ser executada, Figura 6.9, fica visível que logo após as primeiras notas emitidas, as probabilidades caminham para um vermelho intenso, denotando altíssimas probabilidades de cada uma das notas colocadas, o que também evidencia a repetição das melodias resultantes.

Lembramos ainda, que a totalização das recompensas, estão coincidentes com a recompensa da teoria musical, pois apesar de termos as recompensas RNN presente no gráfico de treinamento, ela não é somada a recompensa da teoria da música. O total das recompensas tem o mesmo valor da recompensa da teoria da música, validando assim o

Figura 6.9: Probabilidade de ocorrência de cada nota



Fonte: [58]

Figura 6.10: Valores das recompensas a cada período de execução durante a implementação

```
Training iteration 864000
        Reward for last 2000 steps: 16299.908610974093
                Music theory reward: 16299.908610974093
                Note RNN reward: -7476.3632200956345
Training iteration 866000
        Reward for last 2000 steps: 19394.698960547463
                Music theory reward: 19394.698960547463
                Note RNN reward: -7138.344660639763
Training iteration 868000
        Reward for last 2000 steps: 18111.1859613438
                Music theory reward: 18111.1859613438
                Note RNN reward: -7288.778016328812
Training iteration 870000
        Reward for last 2000 steps: 17343.851451750616
                Music theory reward: 17343.851451750616
                Note RNN reward: -7485.303875923157
Training iteration 872000
        Reward for last 2000 steps: 19562.060856574462
                Music theory reward: 19562.060856574462
                Note RNN reward: -7095.517081022263
Training iteration 874000
        Reward for last 2000 steps: 20772.02519031733
                Music theory reward: 20772.02519031733
                Note RNN reward: -6939.3953058719635
Training iteration 876000
        Reward for last 2000 steps: 19614.9787416461
                Music theory reward: 19614.9787416461
                Note RNN reward: -7227.245076417923
```

Fonte:[58]

isolamento da referida recompensa, como mostrado na Figura 6.10

6.2.1 Experimento Para Solucionar o Problema da Repetição Melódica

Como verificado no experimento anterior, não pudemos ter uma avaliação da construção das melodias, pelo fato da melodia entrar em repetição por atingir uma condição de recompensa ótima, e não produzir mais variação melódica. Para que se possa ter uma visão mais ampla, do quanto a recompensa pela teoria da música se apresenta nessa con-

figuração, executamos este modelo repetidas vezes, com quantidades de passos variáveis, para se obter uma configuração, que melhor identifique pelo treinamento, melodias com aplicação das regras, sem que haja repetições melódicas constantes.

Em uma abordagem com 100 mil passos de treinamento, com um fator 3 de multiplicação sem uso das regras RNN, obtivemos um resultado satisfatório na maioria dos requisitos e as melodias não entraram em *looping*, o que seria um dos nossos objetivos.

Os resultados estatísticos relacionados a 100 composições são os mostrados na tabela 6.2.

.

Tabela 6.2: Tabela de estatística do resultado do treinamento, 100 mil passos de treinamento

Tipo de ocorrência	**Valor**
notas fora da tonalidade	41
notas nos motivos	5030
notas nos motivos repetidos	10
composições iniciadas com a tônica	0
quantidade de notas repetidas	1
saltos de oitava	47
intervalos de quintas	263
intervalos de terça	72
intervalos de sexta	58
intervalos de segunda	317
intervalos de quartas	117
intervalos de sétima	211
notas ligadas	14
notas ligadas em intervalos especiais	3
intervalos especiais na tonalidade	14
saltos resolvidos	126
saltos duas vezes seguidas	92
notas mais agudas únicas	47
notas mais graves únicas	43
composições	100
total de notas em todas as composições	6400

Fonte: [58]

Podemos notar que as melodias geradas possuem uma boa estruturação, de uma forma geral. Vamos ressaltar alguns aspectos relevantes nos resultados obtidos:

- Itens como iniciar pela nota da tonalidade, desaparecem quase que completamente.

Para solucionar esse problema, verificamos que os pesos para esta função, não são muito altos e requerem um treinamento com mais passos para surtir efeito, o que nos levaria novamente aos *looping*.

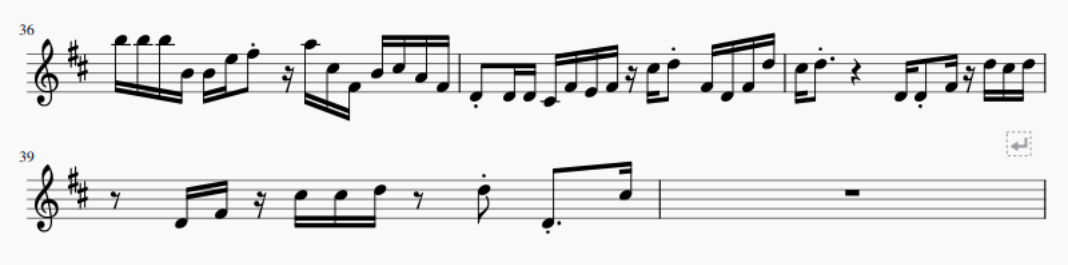
- As notas fora da tonalidade saltaram para 2694, devido ao pouco treinamento.
- Os saltos de oitava subiram para 2468, também devido ao pouco treinamento.
- Os intervalos segundas até sexta ficaram uniformes entre 250 e 350, não fazendo praticamente nenhuma diferença os pesos diferentes dados a essas variáveis.
- Os saltos duplos também não foram devidamente penalizados, aparecendo em 322 ocorrências.
- O numero de sétimas é muito expressivo.

Mais a frente devemos abordar a função que reforça as notas iniciais como solução a esse problema.

O modelo mantém uma estrutura razoavelmente reconhecível, mesmo assim, como podemos constatar nas melodias abaixo 6.11 6.12, mais uma vez selecionadas dentre 9 melodias produzidas.

Adiante em outros experimentos, vamos aumentar a quantidade de passos de aprendizado, e reduzir o fator de multiplicação da recompensa, que conjuntamente coma inserção da recompensa do RNN certamente vão atenuar esses fatores.

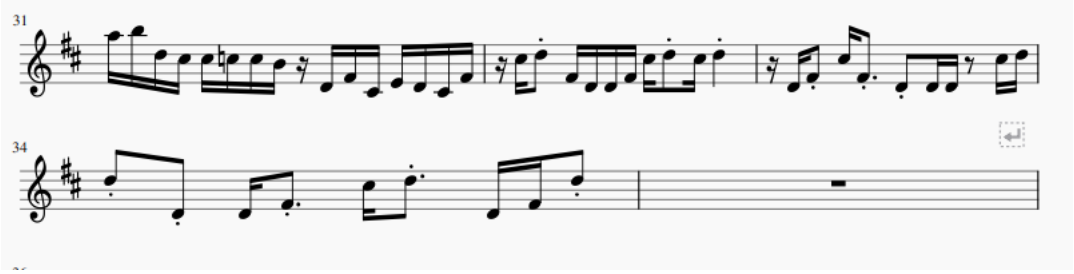
Figura 6.11: Melodia só com uso de reforço das regras da teoria da musica



Fonte: [58]

Analisando essas melodias, podemos verificar uma forte presença da tonalidade. Verificamos a existência de poucas notas fora da escala, mas notamos claramente uma certa falta de criatividade. Podemos observar uma atividade muito mecânica, sem motivos melódicos desenvolvidos, poucos graus conjuntos, sem pontos de repouso nos inícios e finais de frases. Certamente essa falta de coesão melódica, deriva da ausência do componente de reforço, oriundo do aprendizado RNN.

Figura 6.12: Melodia só com uso de reforço das regras da teoria da musica



Fonte: [58]

Quase todas as melodias começam pela nota Sí, apenas uma delas pela nota Lá, mas logo seguida da nota Si, retornando ao padrão, com algumas repetições, seguidas por um processo de progressão descendente em semicolcheias, todas muito parecidas. Nenhuma das composições tem inicio com a nota Ré tônica da tonalidade, evidenciando que esse atributo necessite de um treinamento mais intenso para de fazer valer.

Sobre um corpus de 30.000 musicas usados no experimento original, que certamente enriquecem em muito a diversidade e a sensibilidade, por se tratar de um conhecimento adquirido sobre composições reais, a ausência do aprendizado por reforço RNN, mostra a necessidade de seu papel nessa arquitetura, se o objetivo for refinar a qualidade das composições obtidas.

Apesar de não termos obtidos valores que demonstrassem a efetividade das funções de restrição, da teoria da música, podemos entender que essas restrições necessitam de muito treinamento para serem efetivas, e que não são capazes de gerar qualidade criativa nas melodias resultantes.

6.3 Experimento Usando Apenas o Reforço RNN

Da mesma forma que fizemos o experimento isolando o aprendizado pelas regras da música, faremos agora um experimento isolando o aprendizado por reforço do RNN. Note que não será sobre a saída do RNN, pois será aplicado o aprendizado por reforço da arquitetura, então teremos como saída de recompensa, a recompensa RNN somado a ela mesma, devido a implementação do código original como mostra a Figura 6.13, onde verificamos que o total das recompensas é igual ao dobro da recompensa RNN.

Após várias tentativas, com passos de treinamento variando de 1 milhão até os valores que aqui serão demonstrados, tivemos uma tendência muito grande de tender a zero, em todas as posições do vetores da composição. As melodias com poucas notas no início começam a surgir a partir de 50 mil passos de treinamento,de forma descendente.

Em uma execução com 10 mil passos de treinamento, obtivemos um resultado

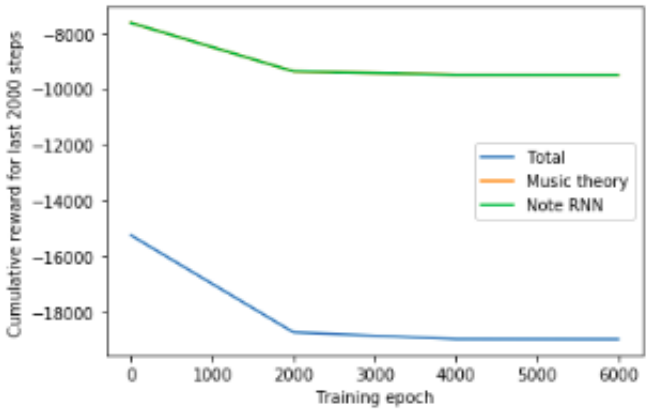
Figura 6.13: Valores das recompensas na execução, a cada período, durante a implementação

```
Training iteration 2000
        Reward for last 2000 steps: -15187.643408060074
                Music theory reward: -7593.821704030037
                Note RNN reward: -7593.821704030037
Training iteration 4000
        Reward for last 2000 steps: -18781.380974769592
                Music theory reward: -9390.690487384796
                Note RNN reward: -9390.690487384796
Training iteration 6000
        Reward for last 2000 steps: -18926.608385562897
                Music theory reward: -9463.304192781448
                Note RNN reward: -9463.304192781448
Training iteration 8000
        Reward for last 2000 steps: -19185.028302192688
                Music theory reward: -9592.514151096344
                Note RNN reward: -9592.514151096344
```

Fonte: [58]

satisfatório, onde podemos ver a construção das melodias até o final dos compassos estipulados, e verificar também a existência apenas da recompensa RNN mostrado na Figura 6.14, bem como as probabilidades de cada nota no decorrer do tempo de execução Figura 6.15.

Figura 6.14: Recompensa só RNN com 10 mil passos de aprendizado

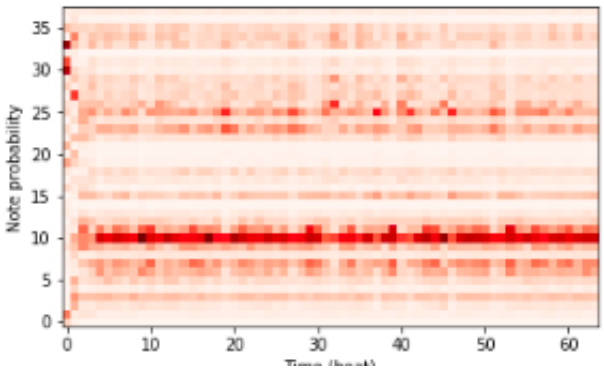


Fonte: [58]

Podemos visualizar aqui três exemplos das melodias geradas nesse experimento, nos pentagramas abaixo.Figura 6.16.

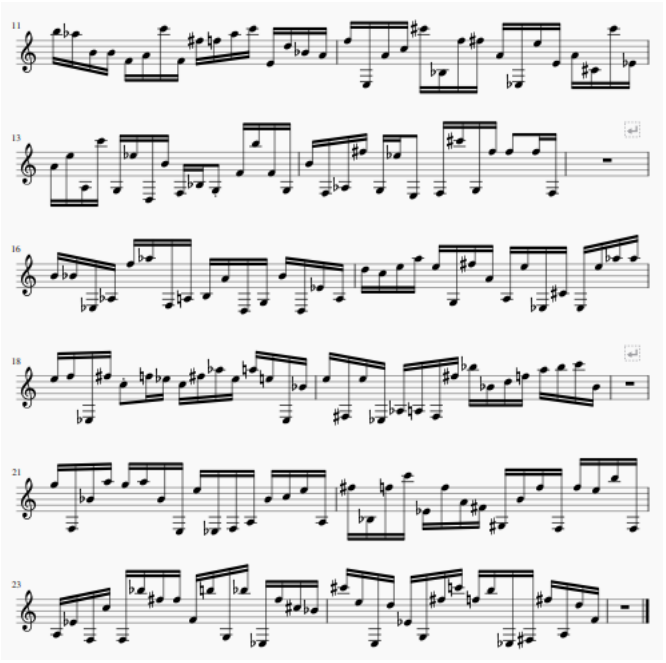
Elas se apresentam totalmente destituídas de estrutura. Cada melodia em uma tonalidade, os intervalos são aleatórios com muitos saltos maiores do que uma oitava. Existe uma prevalência de semicolcheias sem pausas, e raramente encontramos uma colcheia, demonstrando nenhuma valorização a variedade rítmica. Os acidentes ocorrentes proliferam, não existe coesão de motivos melódicos, como também uma baixa adesão de

Figura 6.15: probabilidade de ocorrência de cada nota



Fonte: [58]

Figura 6.16: Exemplos de melodias só com recompensa RNN



Fonte: [58]

graus conjuntos, como mostram os exemplos da Tabela 6.4 a seguir.

Chama a atenção o tratamento dado aos intervalos, que tem quantias dentro de uma mesma faixa de valores. A quantidade de dois saltos em consecutivos, ocorrem 322 vezes, o que é um valor extremamente elevado para cem composições. Ainda notamos a quantia de 2694 notas fora da tonalidade, quase a metade das notas das composições. Por esses motivos podemos concluir, que não existe uma estrutura desse modelo, tornando-se muito oportuna a utilização das recompensas baseadas nas restrições impostas por regras musicais.

Em várias tentativas de execução com uma quantidade maior de passos de treinamento, os resultados tendem todos a zero, não sendo possível fazer qualquer tipo de

wraptableo8cm

Tabela 6.3: Tabela de estatística do resultado treinamento

Tipo de ocorrência	**Valor**
notas fora da tonalidade	2694,
notas nos motivos	5696,
notas nos motivos repetidos	0
composições iniciadas com a tônica	4
notas repetidas	4
saltos de oitava	2468,
intervalos de quintas	249,
intervalos de sexta	252,
intervalos de segunda	378,
intervalos de quartas	344,
intervalos de sétima	173,
notas ligadas	1
notas ligadas em intervalos especiais	0
intervalos especiais na tonalidade	25
saltos resolvidos	9
saltos duas vezes seguidas	322
agudas únicas	60
notas mais graves únicas	42
composições	100
total de notas em todas as composições	6400

Fonte: [58]

avaliação.

6.4 Experimento com a Função *Tonic*

Nesse teste vamos buscar um entendimento sobre a nota tônica da tonalidade, seu comportamento e como se desenvolve ao longo das composições.

Primeiramente fizemos um desenvolvimento, utilizando 500 mil passos de treinamentos, um *reward_scaler=2.0* que é o fator de importância dado as recompensas da teoria, para compensar um pouco a pequena quantidade de passos. A variável *amount* passada como parâmetro pela função colocamos em 9.0, sendo que no original o valor é de 3.0, triplicando a taxa de recompensa para essa função.

Esse certo exagero se faz necessário, devido a inconsistências de testes anteriores, feitos com valores menores onde não era possível observar o comportamento da função. Para podermos verificar claramente sua atuação chegamos a essa combinação de valores dos parâmetros instanciados empiricamente.

Essa função alterada, permanece dentro da escolha do modelo,*music_theory_all* que trata de todas as funções da teoria da música, em conjunto, onde apenas alteramos os valores da função estudada, isoladamente.

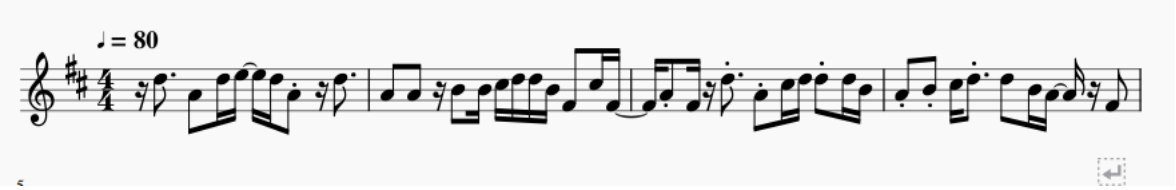
Em testes onde implementamos dentro de uma classe filha apenas o uso dessa função, os resultados não foram satisfatórios, gerando saídas, sem qualquer possibilidade de entendimento das ações dessa função, e foram descartados.

Verificamos que o experimento gera uma constante repetição da tônica da escala, Ré maior no caso, durante o desenvolvimento do processo composicional, apesar de não ser reproduzido sempre na primeira nota da melodia, mas costuma se repetir nos inícios e finais de períodos melódicos. Esse fato é muito interessante, pois conseguimos uma conotação modal, apesar de exagerada, como mostram os pentagramas da Figura 6.17 e da Figura 6.18. Mesmo as frases que não iniciam com a nota tônica, conservam um carácter modal mostrado na 6.19.

Esse resultado é importante, pois a conotação modal, em cada um de seus graus, tem características particulares de contexto emocional, que pode ser usado, de forma abrangente, como ferramenta para ajudar a compor, um direcionamento a sentimentos específicos.

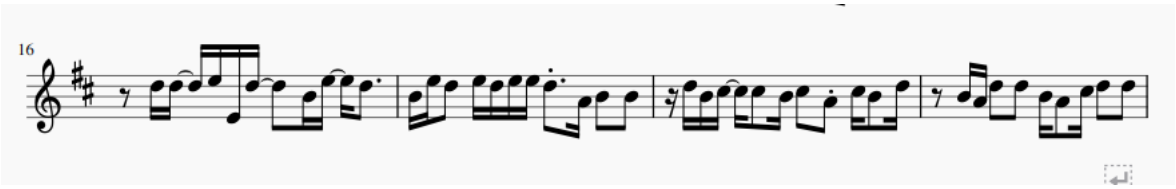
Em um segundo desenvolvimento, vamos reduzir o *amount* para 4.0, o que seria um aumento de um terço nas recompensas originais, amenizando os efeitos da função. O objetivo desse outro teste seria buscarmos um equilíbrio para esse parâmetro, visto já termos um certo entendimento de sua atuação, no processo de construção das melodias, verificados no experimento anterior.

Figura 6.17: Melodia Modal *Tonic*



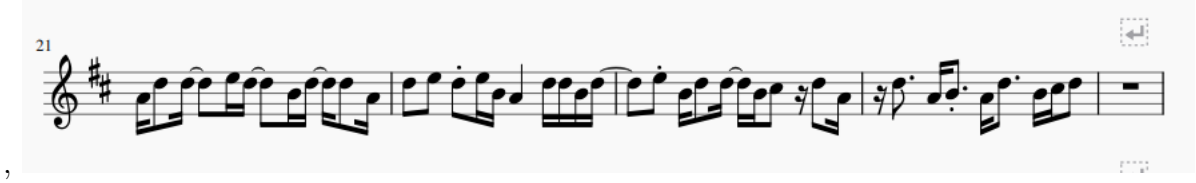
Fonte: [58]

Figura 6.18: Melodia Modal *Tonic*



Fonte: [58]

Figura 6.19: Melodia Modal *Tonic*, não iniciada pela nota fundamental



Fonte: [58]

Os resultados foram bem satisfatórios, produzindo melodias consistentes e com uma interferência mais suave, permitindo assim que outros elementos aparecessem um pouco mais, imprimindo mais naturalidade as composições.

O modelo acaba produzindo algumas melodias, que não retratam esses parâmetros com tanta intensidade quanto no experimento anterior, mas ficam muito mais interessantes sobre o ponto de vista melódico.

Pelo fato de termos uma somatória de valores de recompensa da teoria da música, sempre que aumentamos os valores de uma das recompensas, ela pode se sobrepor a pequenas variações em outros parâmetros, inibindo assim alguns efeitos de regras necessárias.

Na maioria das composições resultantes, o comportamento de gerar mais notas fora das estruturas estabelecidas, são plenamente compensadas por uma musicalidade muito mais presente, como podemos ver na Figura 6.21, considerando-se sempre o objetivo, de se aproximar de composições mais realistas.

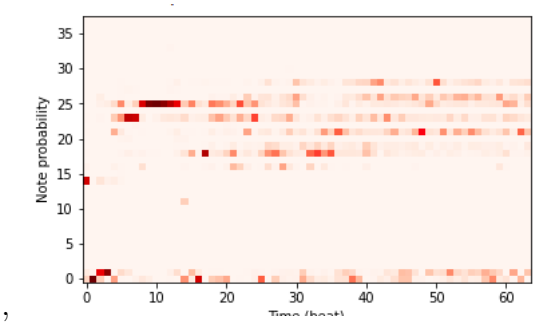
Como nesse experimento estamos desenvolvendo e observando a atuação dessa função, entendemos como plenamente válidos e conclusivos, estes testes.

As composições resultantes, mantém suas notas geradas, em sua maioria, dentro

de uma tessitura central, como podemos verificar tanto nos pentagramas 6.21, onde a maioria das notas ficam dentro do pentagrama, quanto no gráfico de probabilidades das notas compostas, que compõe sua maioria entre os valores de 15 e 30 (eixo y),Figura 6.20.

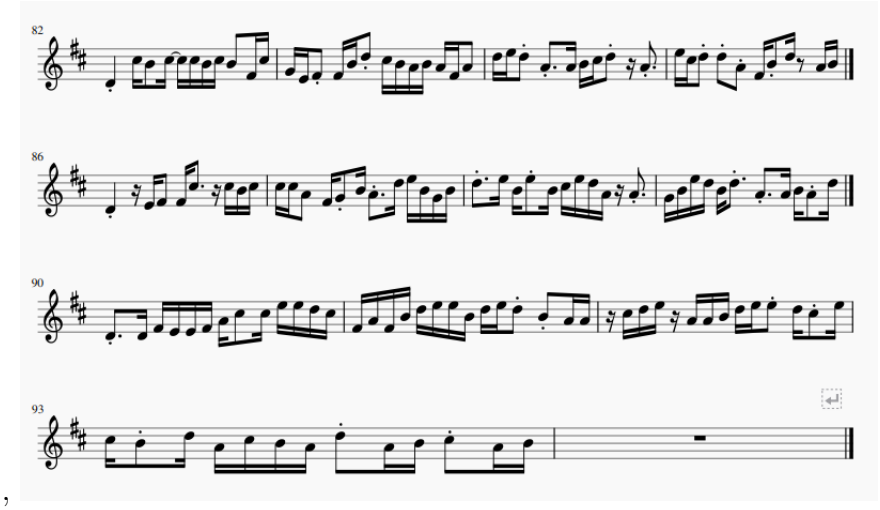
Essa região mediana do pentagrama, corresponde ao que seria uma região no entorno do ponto médio da magnitude do vetor de notas possíveis.

Figura 6.20: Melodia Modal *Tonic*, não iniciada pela nota fundamental



Fonte: [58]

Figura 6.21: Melodia Modal *Tonic*, melhores resultados



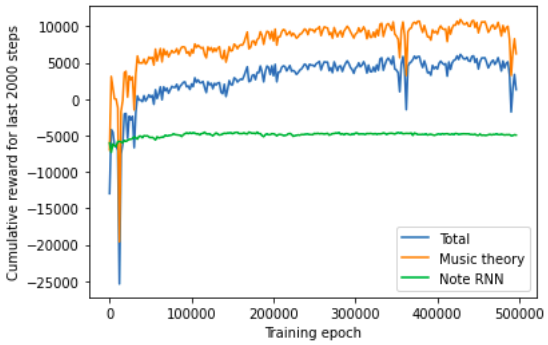
Fonte: [58]

Observando o gráfico da Figura 6.22, que representa a distribuição das recompensas em função das épocas. Notamos que a distribuição das recompensas RNN em verde, e das recompensas da teoria da música, em amarelo, estão muito bem distribuídas e equidistantes do resultado final, linha em azul.

Se comparamos a curva da recompensa da teoria da música com a curva do resultado final, observamos que elas estão muito simétricas. Isso se deve ao fato, de darmos um fator de multiplicação alto para a teoria da música, forçando uma predominância de seus valores sobre o resultado

Podemos então concluir que os parâmetros ajustados para as variáveis nesse teste, estão satisfatoriamente equilibradas para o nosso propósito.

Figura 6.22: Melodia Modal *Tonic*, não iniciada pela nota fundamental



,

Fonte: [58]

6.5 Experimento com a Função *Key*

A função *reward_key*, trata da penalização que se dá as notas compostas, que não pertençam a uma determinada tonalidade, notas fora da escala. Essas notas acabam por ficar com recompensas menores, aumentando a probabilidade das notas pertencentes as escalas estabelecidas, ou seja, as notas que pertencerem a escala implementada, tem mais chances de serem as escolhidas como a melhor opção, em um determinado tempo t .

Este experimento foi primeiramente implementado com a função *reward_key*, passando como parâmetro a variável *penalty_amount*, com o valor de -4.0, o que seria da ordem de 4 vezes o valor original, para que pudéssemos observar claramente seu comportamento.

O treinamento foi feito com 500 mil passos de treinamento, e o fator *scale* passado como 2.0, para compensarmos a pouca quantidade de passos de treinamento.

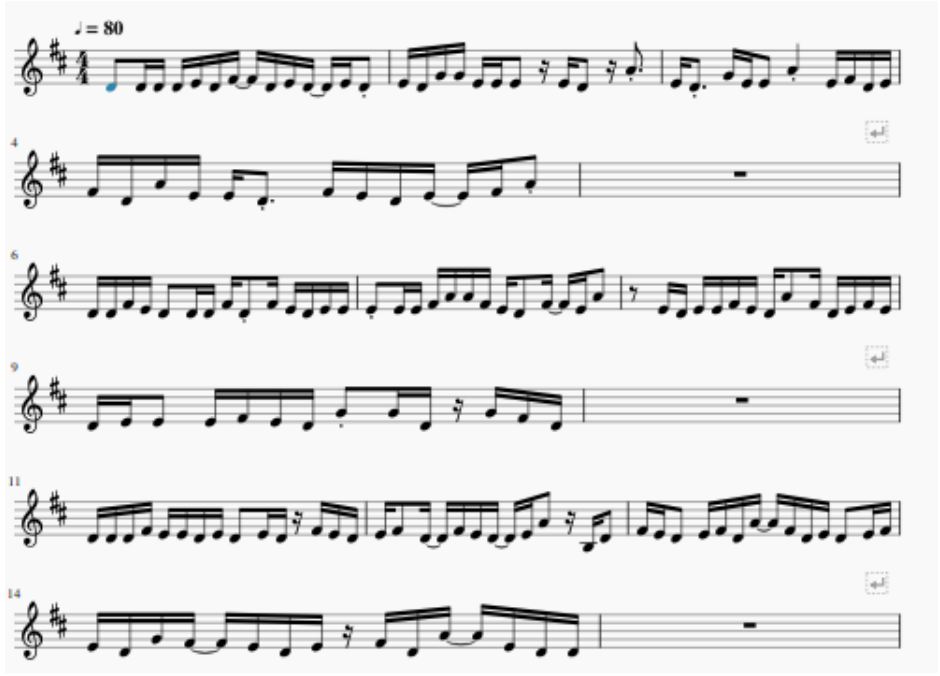
Novamente implementamos valores de *penalty_amount*, que penaliza excessivamente as notas compostas, cuja primeira nota não seja a tônica da escala, para entender o comportamento da função.

Observamos que 100% das composições iniciam com a nota fundamental, Figura 6.23, o que seria uma importante descoberta, pois nas composições em que quisermos imprimir algum sentimento direcionado, podemos usar essa ferramenta para ajudar a garantir explicitamente as tonalidades, notadamente no manuseio de perspectivas modais de composição.

Temos uma grande concentração de intervalos de segunda, e um numero médio entre os intervalos de quartas e quintas, 271 notas ligadas que muito ajudam na diversidade rítmica e com 586 notas ligadas em intervalos preponderantes a tonalidade, que reforçam

os repousos melódicos em notas importantes como tônicas, terças e quintas da tonalidade.

Figura 6.23: todas as composições iniciam com a fundamental



,

Fonte: [58]

Essas composições modais são muito importantes na implementação de sentimentos, pois com uma única escala, maior por exemplo, apenas manuseando as notas iniciais e finais podemos impor 7 condições modais. Os modos gregos são escalas subtraídas de uma escala maior [62]. Seja por exemplo uma escala de Dó maior: Dó, Ré, Mi, Fá, Sol, Lá e Si. Se partirmos da nota Dó, primeira da escala teremos o modo Jônico, se partirmos da segunda , Ré, teremos o modo Dórico e assim por diante como mostrado na Tabela 6.5

Estes modos gregos tem características próprias de impressão de sentimento, e é amplamente usado pelos compositores para este fim, pois cada um deles impactam texturas próprias. Essa ferramenta pode ser utilizada como um elemento de influência no direcionamento de sentimentos, dentro de uma estrutura composicional.

Apesar de podermos extrair essas importantes conclusões, temos que observar que as melodias geradas, transitam basicamente dentro de um intervalo de quinta, como podemos observar no pentagrama da Figura 6.23, ou seja, a amplitude de notas utilizada está praticamente, entre 9 notas, o que restringe em muito o desenvolvimento das composições.

Para inserir essas características aqui abordadas, em implementações futuras, devemos cuidar da grandeza dos valores de *penalty_amount*, em detrimento da valorização de outros aspectos necessários a composição. Aqui estamos apenas tentando entender, o quanto essa função impacta no direcionamento dessas intenções.

Devemos nos lembrar, que ao inserirmos valores maiores de recompensas a funções

!htb

Tabela 6.4: Tabela de estatística do resultado treinamento

Tipo de ocorrência	**Valor**
notas fora da tonalidade	4,
notas nos motivos	5303
notas nos motivos repetidos	2
composições iniciadas com a tônica	100
notas repetidas	2
saltos de oitava	1
intervalos de quintas	10
intervalos de sexta	20
intervalos de segunda	2124
intervalos de terça	10
intervalos de quartas	395
intervalos de sétima	0
notas ligadas	271
notas ligadas em intervalos especiais	586
intervalos especiais na tonalidade	275
saltos resolvidos	167
saltos duas vezes seguidas	2
agudas únicas	21
notas mais graves	20
composições	100
total de notas em todas as composições	6400

Fonte: [58]

Tabela 6.5: Modos Gregos

Nota inicial da escala maior	**nome do modo grego**
iniciando pela primeira nota	escala Jônica
iniciando pela segunda nota	escala Dórica
iniciando pela terceira nota	escala Frígia
iniciando pela quarta nota	escala Lídia
iniciando pela quinta nota	escala Mixolídia
iniciando pela sexta nota	escala Eólica
iniciando pela sétima nota	escala Lócria

Fonte: [58]

isoladamente, precisamos nos atentar ao fato de que a recompensa da teoria da musica, é uma somatória de recompensas. Certamente ao aumentar, em escala, essas funções alteradas, em algum momento poderia implicar em um aumento excessivo da recompensa da teoria da música. Esse problema pode ser corrigido com a variação do fator *scaler*, responsável pelo quanto vamos valorar estas regras, dentro da soma das recompensas, com a recompensa RNN, como já havíamos observado anteriormente.

6.5.1 Um Resultado Lídio

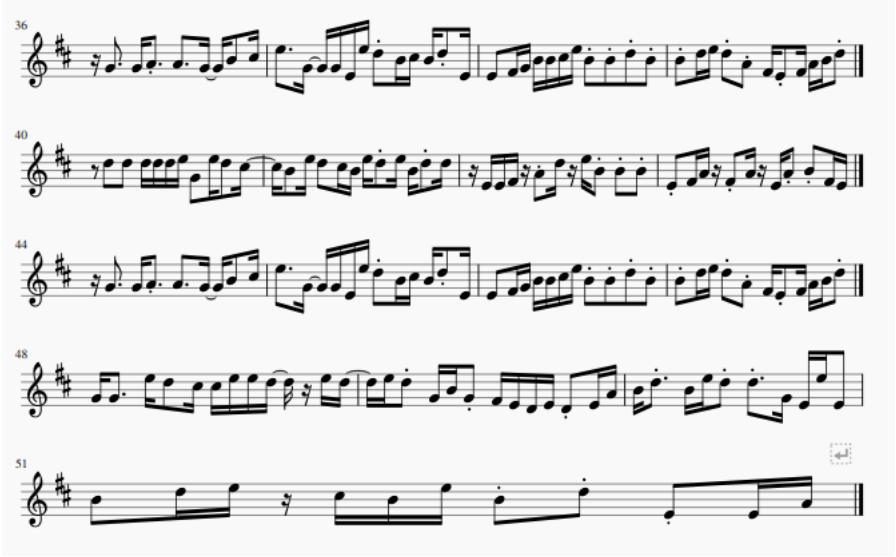
Curiosamente em um novo experimento com *penalty_amount* em -0.2 com 100mil passos de treinamento, obtivemos um resultado muito interessante, como mostrado nos pentagramas da figura 6.24.

Na maioria das composições geradas neste experimento, obtivemos resultados no modo Lídio, que possui uma sonoridade bem característica. Apesar de não nos aprofundarmos mais nesse experimento, entendemos existir nele uma ferramenta importante para a construção de sentimentos específicos.

Ela reflete uma sonoridade que expressa uma certa tensão, com suavidade e com um sentido interrogativo.

Sugerimos em uma futura investigação, onde certamente, num refinamento mais apurado, resulte em um entendimento dos processos que compõe este resultado, seu desenvolvimento e variações, de acordo com as modificações ocorridas, para um melhor entendimento e capacitação do uso desses parâmetros, como ferramenta de controle deses sentimentos observados.

Figura 6.24: Experimento gerando melodias em modo Lídio



,

Fonte: [58]

6.6 Experimento para construção de escalas alternativas

Um dos mais importantes fatores para implementação de sentimentos, são as escalas utilizadas, pois elas são capazes, por si só, de produzir contextos de sentimentos bem específicos.

Em uma composições na qual temos a intensão de implementar sentimentos relacionados a tristeza ou depressão por exemplo, fica claro que o uso de escalas menor natural e menor harmônica, são naturalmente uteis para a implementação essas texturas musicais.

Escalas diminutas, menor melódicas e de tons inteiros aplicadas a contextos de função de cadência harmônica dominante, são também muito uteis para criar sentimentos de tensão, nervosismo e irritabilidade, entre outros.

Apenas estes fatores já seriam um argumento suficiente para percebermos a necessidade de abordarmos o controle do uso dessas escalas. Seria certamente uma ferramenta importante, para direcionar o aprendizado por reforço das regras musicais, para contextos de sentimentos específicos.

6.6.1 As Escalas Menores

Apesar de não ser uma condição obrigatória, normalmente as melodias com carácter de sentimento melancólicos, são compostas em andamentos mais lentos, bem como as melodias alegres ficam bem descritas em um andamento mais rápido, em outras palavras nos parece essencial o controle do andamento, como ferramenta auxiliar na construção

de sentimentos nas canções. Fizemos variações do hiper parâmetro *DEFAULT_QPM* = 80.0 para 55 com pleno exito no controle dos andamentos propostos nas composições. Lembrando que são hiper parâmetros, estabelecidos antes dos treinamentos, e imutáveis durante o processo, portante não pertencem ao processo de aprendizagem propriamente dito.

Foram feitos experimentos modificando os hiper parâmetros *C_MAJOR_SCALE*, *C_MAJOR_KEY*, *C_MAJOR_TONIC* e *A_MINOR_TONIC* sem modificar seus nomes, apenas modificando os valores das posições desses vetores, para recompensar notas da escala menor natural.

Refizemos as estruturas de valores dos vetores, para compor escalas específicas, usando as mesmas denominações:
rl_tuner_ops.C_MAJOR_SCALE = [2, 4, 5, 7, 9, 10, 12, 14, 16, 17, 19, 21, 22, 24, 26]
rl_tuner_ops.C_MAJOR_KEY = [0, 1, 2, 4, 5, 7, 9, 10, 12, 14, 16, 17, 19, 21, 22, 24, 26, 28, 29, 31, 33, 34, 36].

Na terceira posição do vetor *C_MAJOR_SCALE* temos o valor 5 que seria um Mí bemol, correspondente a terça menor da escala, no lugar do valor 6 anteriormente usado, que sería um Mí natural, correspondente a terça maior da escala. Na posição 6 do vetor trocamos o valor 11 por 10 e na posição 7 trocamos o valor 13 por 12, e assim por diante para estruturar a escala de Dó menor. Usamos o mesmo procedimento para definir o vetor de tonalidade.

Fizemos também experimento implementando escalas menores como eólica, sem deixar o sétimo grau como sensível, com a colocação do Hiper parâmetro *A_MINOR_TONIC* = 23, como nota fundamental da escala implementada.

A implementação de escalas diminutas e pentatônicas como novos hiper parâmetros, também se mostram inconsistentes. Alteramos também em uma nova classe filha todas as funções que tinham relação com esses vetores, obtendo os mesmos resultados.

Entenda-se que não alcançamos nossos objetivos nesses testes, porque não conseguimos obter os resultados esperados, mas tivemos interessantes construções melódicas na maioria das tentativas, mas não conseguimos implementar o controle dos sentimentos propostos nesses testes especificamente.

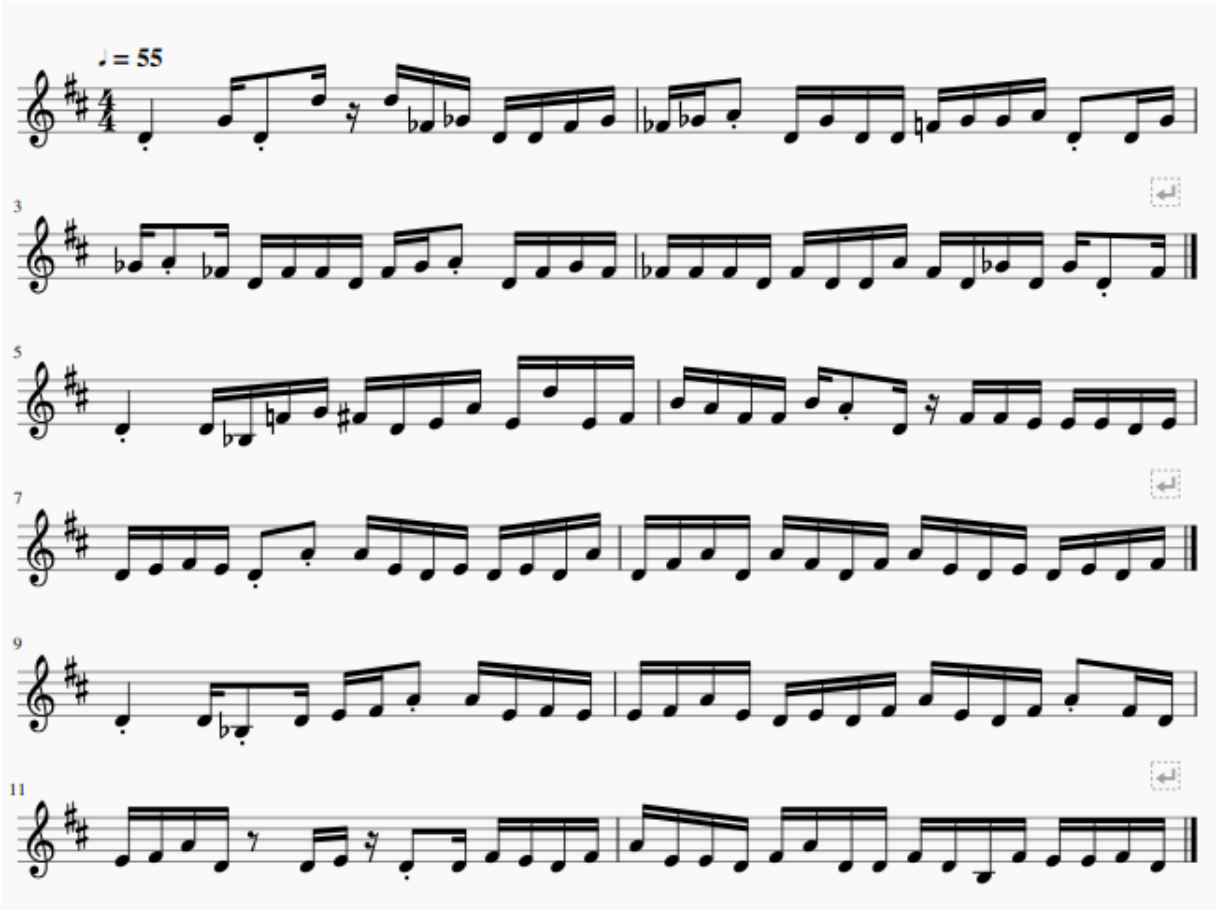
Podemos ver no exemplo da Figura 6.25, com o objstivo de criar um contexto melancólico, na tentativa de imprimir melodias sobre tonalidades menores. Não conseguimos determinar fatores importantes que pudessem ser usados como ferramentas, apesar dos inúmeros testes gerados.

Sugerimos em futuros trabalhos um a profundamente neste tema, que parece ser um elemento básico na construção de sentimentos em composições.

Estes testes foram inconclusivos, não obtivemos o resultado esperado em nenhuma

Figura 6.25: com amount -4 todas as composições iniciam com a fundamental

,



Fonte: Elaborado por

composição. Fizemos ajustes para escalas diminutas e pentatônicas e também não obtivemos resultados conclusivos.

6.6.2 Escala Pentatônica

Em um dos experimentos, casualmente nos deparamos com um resultado que não poderia deixar de ser apresentado, por ter atingido um de nossos objetivos. Esse resultado está fora de nossa metodologia, mas devido a seu resultado, deve ser relatado para que possa ser tema de investigação em estudos futuros.

Em uma implementação onde buscávamos manipular elementos que pudessem impor sentimentos melancólicos, com uso de hiper parâmetros que supostamente nos levariam a escalas menores, onde implementamos os hiper parâmetros a seguir:

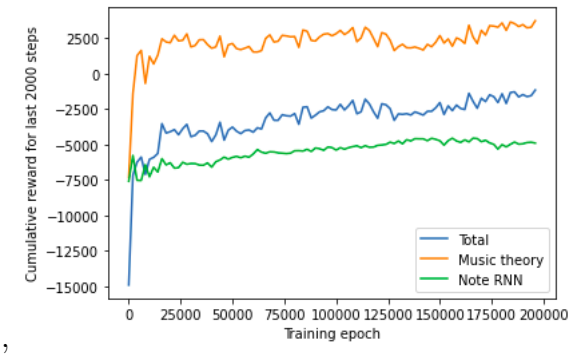
- $rl\_tuner\_ops.THIRTH = 3$
- $rl\_tuner\_ops.SEVENTH = 10$

- *rl_tuner_ops.SIXTH* = 8

A classe *reward_tonic* com *reward_amount* = 4.0 foi valorada dentro do escopo da função como também os parâmetros *NOTE_OFF* = 0 e *NO_EVENT* = 1, com um treinamento com 200 mil passos de treinamento.

No gráfico de treinamento, como mostra a Figura 6.26, podemos verificar que o resultado do treinamento, linha azul, está mais próximo da recompensa RNN, apesar de possuir um perfil semelhante ao da recompensa da teoria da música.

Figura 6.26: Treinamento do experimento que gerou pentatônicas

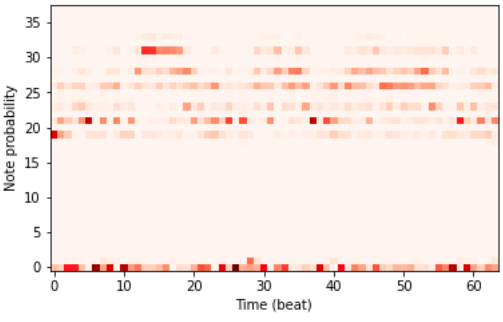


,

Fonte: [58]

No gráfico das probabilidades também não observamos nenhuma anomalia, com uma concentração de probabilidades na região mediana, e uma boa probabilidade de pausas e notas contínuas, como mostra a Figura 6.27.

Figura 6.27: Probabilidades do experimento que gerou pentatônicas

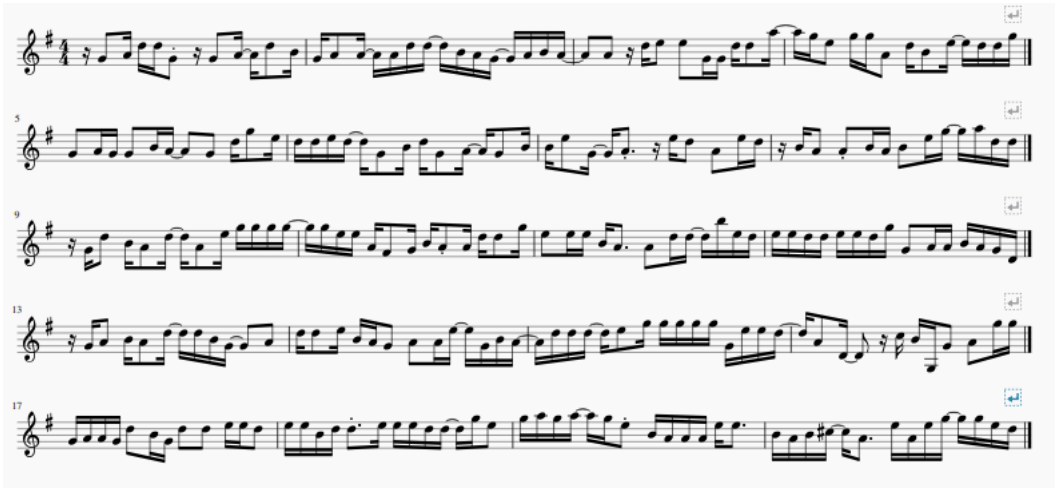


,

Fonte: [58]

Notamos que as melodias geradas em sua maioria são pentatônicas de Sol , com poucas notas fora da escala apenas em algumas composições, como pode ser verificado no pentagrama da Figura 6.28. Não pudemos encontrar uma justificativa, que explicasse o surgimento dessas melodias dentro desse contexto. Seria necessário uma investigação mais aprofundada, para que uma compreensão mais apurada se tornasse uma ferramenta, capaz de reproduzir esse contexto, sempre que desejado.

Figura 6.28: Partitura do experimento que gerou melodias pertencentes a escala pentatônica



,

Fonte: [58]

Capítulo 7

Conclusão

O artigo de Jaques e colegas, que me introduziu o esse fascinante mundo, onde máquinas se tornam cada vez mais capazes e competentes ao tentar criar, por assim dizer, um estado da arte. Capacidade esta, que elevou o ser humano a um outro patamar civilizatório e de expressividade única. Compreender como essas tecnologias atuam, não para substituir o homem em um de seus quesitos mais significativos como espécie, mas para repetir o processo criativo de forma matemática, nos aprofundado nos efeitos colaterais sensitivos, que essas tecnologias poderiam ser capazes de reproduzir no futuro.

O processo de composição musical proposto por Jaques e colegas, com uma arquitetura que se utiliza, do conhecimento adquirido de compositores na área musical, o aprendizado RNN/LSTM, podendo ser visto metaforicamente, como a tecnologia que capacita um modelo, ao que seria escutar muita música, para entender sensitivamente o conjunto dessas manifestações artísticas, e reconhecer e desenvolver, também sensitivamente, um sentido de qualidade.

Caminhando na direção do conhecimento, orientado por seus mestres, que insistentemente, durante todo seu aprendizado, procuram incutir em seus discípulos, a necessidade conectiva com a arte já produzida, no caso, a música. Manter sempre como parte de sua meta, esse aspecto quase que subliminar, tão importante na construção de um artista, a intuição.

Combinar esse conhecimento com um treinamento dentro de conceitos e regras estruturais, como faz um indivíduo que sente o ímpeto de desvendar, os fundamentos e os processos de se fazer música, se enveredando pelos caminhos do aprendizado teórico dessa matéria.

Nessa arquitetura, esse papel cabe claramente ao aprendizado por reforço, que atua na imposição das estruturas definidas pela musica, aqui estabelecido em um conjunto, digamos que apenas basicamente mínimo, para validação dos processos. Todo o conjunto de conhecimento adquirido empiricamente, apenas ao escutar e reproduzir música, nunca

poderia ser esquecido, como muito bem observado pelos autores, representado pelo reforço do aprendizado sobre o corpus de canções, uma ótima modelagem do comportamento da vida real.

Pudemos comprovar com nossa reprodução do modelo original, a eficiência e precisão desses processos em nossos testes, que foram muito efetivos, criando de fato melodias agradáveis. Essas composições com uma certa beleza estética, poderiam ser cada vez mais refinadas, a medida em que nos aprofundássemos no desenvolvimento de estruturas, que pudessem regrar cada vez mais e com mais eficiência. Criando, testando e aperfeiçoando funções que poderiam atuar para cada necessidade composicional, em um conjunto de recompensas e penalizações mais abrangentes em termos composicionais, certamente refinaríamos os resultados melódicos.

Foi também muito importante em nossos estudos verificarmos o desempenho dos reforços separadamente, o que nos levou a um entendimento, da vulnerabilidade dos resultados de forma disjunta, onde não conseguimos entender, nenhum tipo de resultado convincente esteticamente, sob o ponto de vista da composição, o que em ultima análise, valida a arquitetura desenvolvida por Natasha et al.

Num processo de composição, o artista tem a necessidade de transformar seus sentimentos em musica, correlacionando desde sempre em suas tentativas de compor, durante sua vida musical. A arte como expressão do sentimento.

Nas máquinas esse processo é impossível simplesmente porque máquinas não são pessoas, não sentem, ao menos, não por enquanto. Para que se possa introduzir sentimentos, em uma composição feita por algorítimos, teremos obrigatoriamente que simular esse processo subjetivo.

Pensando no comportamento de um músico, iniciado nas atividades composicionais, nos momentos de construção de melodias, em busca de inspiração, ele manuseia seu instrumento, sua memória ou faz uso de outros artifícios que o conduzam nessa busca, quase que desesperada de um fator sensitivo conhecido como inspiração.

De imediato traz para as ferramentas que conhece, seu instrumento talvez, essa construção. Busca sequencia de notas, seguindo sensorialmente ou com total lucidez ritmos, andamentos, escalas, tonalidades, fragmentos aleatórios, enfim , toda a sorte de ferramentas que possam ajudar na inicialização desse processo.

Partimos da hipótese que poderíamos, manipulando as regras da teoria musical e ir de encontro a esses preceitos. Como máquina não tem intuição, nem percepção de seus sentimentos, teríamos que qualificar esse caminho por escolha. O algoritmo não vai estar alegre, o especialista vai dizer a ele que alegria é o estado inicial, para que ele possa com seu conhecimento caminhar nessa direção. Para isso precisamos controlar, os pesos dessas regras que serão aplicados a cada tarefa específica.

Nossa pesquisa seria, como dito anteriormente por hipótese, verificar a possibilidade de atuar com essas funções e verificar se de fato elas podem imprimir alguma interferência na busca dessas texturas almejadas.

Vimos em nossos testes com tonalidades, notas de maior importância dentro de melodias , modificação de escalas, não categoricamente, mas apenas sugestivamente, para que os algorítimos façam suas escolhas e façam suas construções melódicas. Demonstramos que essas interferências são possíveis e significativas, na orientação e expressões de algum sentimento. Notamos também que o controle desse itens são muito mais complexos e difíceis de serem utilizados como ferramenta de precisão, pois possuem muitas variáveis, e são inseridas em um processo sugestivo, não determinante, mas esse assunto já foge de nosso escopo. Concluímos ser plenamente válido, o uso dessas ferramentas para se desenvolver futuramente, tecnologias que permitam manipular essas variáveis e introduzir sentimentos ou regiões de sentimento, satisfatoriamente, com o uso dessas tecnologias.

Referências Bibliográficas

[1] Gerando sequências com redes neurais recorrentes. *arXiv pré-impressão arXiv:1308.0850*.

[2] Magenta all jam session-best demo "nips 2016".

[3] Info escola navegando e aprendendo, 2006-2021. disponível em ¡https://www.infoescola.com/educacao/gestao-escolar/¿, 2020. Acesso em: 16 de setembro de 2020.

[4] introducao-a-redes-neurais-e-deep-learning. https://didatica.tech/introducao-a-redes-neurais-e-deep-learning/, 2022. Acesso em: 1 de dezembro de 2022.

[5] treinamento-de-redes-neurais. https://www.deeplearningbook.com.br/algoritmo-backpropagation-parte-2-treinamento-de-redes-neurais/, 2022. Acesso em: 1 de dezembro de 2022.

[6] Gérard Assayag, Camilo Rueda, Mikael Laurson, Carlos Agon, and Olivier Delerue. Computer-assisted composition at ircam: From patchwork to openmusic. *Computer music journal*, 23(3):59–72, 1999.

[7] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[8] Shaleen Bengani, S Vadivel, and J Angel Arul Jothi. Efficient music auto-tagging with convolutional neural networks. 2019.

[9] MED Bohumil. Teoria da música. *Brasília: MusiMed*, 1996.

[10] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[11] François Briot, Jean-Pierre e Pachet. Aprendizagem profunda para geração de música: desafios e direções.

[12] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[13] Yara Borges Caznok. *Música: entre o audível e o visível*. Unesp, 2004.

[14] Alan Cowen, Disa Sauter, Jessica L Tracy, and Dacher Keltner. Mapping the passions: Toward a high-dimensional taxonomy of emotional experience and expression. *Psychological Science in the Public Interest*, 20(1):69–90, 2019.

[15] Alan S Cowen, Xia Fang, Disa Sauter, and Dacher Keltner. What music makes us feel: At least 13 dimensions organize subjective experiences associated with music across different cultures. *Proceedings of the National Academy of Sciences*, 117(4):1924–1934, 2020.

[16] Shuqi Dai, Zheng Zhang, and Gus G Xia. Music style transfer: A position paper. *arXiv preprint arXiv:1803.06841*, 2018.

[17] Charles Darwin and Phillip Prodger. *The expression of the emotions in man and animals*. Oxford University Press, USA, 1998.

[18] José Tarcísio Franco de Camargo and Estéfano Vizconde Veraszto. Redes neurais e o estudo de séries temporais: Uma aplicação no ensino de engenharia.

[19] Nick Didkovsky and Philip L Burk. Java music specification language, an introduction and overview. In *ICMC*, 2001.

[20] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

[21] Paul Doornbusch. Computer sound synthesis in 1951: The music of csirac. *Computer Music Journal*, 28(1):10–25, 2004.

[22] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[23] Juergen Eck, Douglas e Schmidhuber. Encontrando estrutura temporal na música: improvisação de blues com redes recorrentes lstm.

[24] Scott R Eliason. *Maximum likelihood estimation: Logic and practice*. Number 96. Sage, 1993.

[25] Wolfgang Ertel. *Introduction to artificial intelligence*. Springer, 2018.

[26] Silvio Ferraz. Pequena trajetória da ideia de tempo na música do século xx. *A música dos séculos XX e XXI. Edited by Nascimento et al*, pages 86–104, 2014.

[27] Baptiste Fiebrink, Rebecca e Caramiaux. O algoritmo de aprendizado de máquina como ferramenta musical criativa.

[28] ELIAS DO NASCIMENTO MELO FILHO-CONSERVATÓRIO. Ensino de música a distância: Análise de softwares de edição e criação musical.

[29] Michael Gabriel and John Moore. *Learning and computational neuroscience: Foundations of adaptive networks*. MIT Press, 1990.

[30] Robert Gauldin. *A practical approach to eighteenth-century counterpoint*. Waveland Press, 1988.

[31] Jürgen e Cummins Fred Gers, Felix A e Schmidhuber. Aprendendo a esquecer: previsão contínua com lstm. *computação neural*, 12.

[32] Jonathan Goldman. *The musical language of Pierre Boulez: Writings and compositions*. Cambridge University Press, 2011.

[33] Avram Goldstein. Thrills in response to music and other stimuli. *Physiological psychology*, 8(1):126–129, 1980.

[34] Jürgen Graves, Alex e Schmidhuber. Classificação framewise de fonemas com lstm bidirecional e outras arquiteturas de rede neural. *Redes neurais*, 18.

[35] Gaetan Hadjeres and Frank Nielsen. Interactive music generation with positional constraints using anticipation-rnns. *arXiv preprint arXiv:1709.06404*, 2017.

[36] James B Heaton, Nick G Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

[37] Jeff Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Natasha Jaques, Shixiang Gu, Richard E Turner, and Douglas Eck. Generating music by fine-tuning recurrent neural networks with reinforcement learning. 2016.

[40] Natasha Jaques, Shixiang Gu, Richard E Turner, and Douglas Eck. Tuning recurrent neural networks with re-inforcement learning. *arXiv preprint arXiv:1611.02796*, 2016.

[41] Mark Jenkins. *Analog Synthesizers: Understanding, Performing, Buying–From the Legacy of Moog to Software Synthesis*. Routledge, 2009.

[42] Philippe Jost, Pierre Vandergheynst, Sylvain Lesage, and Rémi Gribonval. Motif: An efficient algorithm for learning translation invariant dictionaries. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V. IEEE, 2006.

[43] Patrik N Juslin and Petri Laukka. Expression, perception, and induction of musical emotions: A review and a questionnaire study of everyday listening. *Journal of new music research*, 33(3):217–238, 2004.

[44] Hamid Khodabandehlou and M Sami Fadali. Training recurrent neural networks as a constraint satisfaction problem. *arXiv preprint arXiv:1803.07200*, 2018.

[45] Hwanhee Kim. Learn reinforcement learning (1) - value function. Disponível em: `https://greentec.github.io/reinforcement-learning-first-en/`, 2022. Acesso em: 01 de fevereiro 2022.

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[48] Stefan Koelsch. Towards a neural basis of music-evoked emotions. *Trends in cognitive sciences*, 14(3):131–137, 2010.

[49] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[50] Joseph LeDoux. *The emotional brain: The mysterious underpinnings of emotional life*. Simon and Schuster, 1998.

[51] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[52] Christian Alessandro Lisboa. A intenção do intérprete e a percepção do ouvinte: um estudo das emoções em música a partir da obra piano piece de jamary oliveira. 2008.

[53] Steven R Livingstone, Caroline Palmer, and Emery Schubert. Emotional response to musical repetition. *Emotion*, 12(3):552, 2012.

[54] Ada Lovelace. Ada lovelace. *Birth*, 1815.

[55] Adrian Hinojosa Luna. Introdução aos métodos de monte carlo avançados.

[56] Dimos Makris, Maximos Kaliakatsos-Papakostas, Ioannis Karydis, and Katia Lida Kermanidis. Combining lstm and feed forward neural networks for conditional rhythm composition. In *International conference on engineering applications of neural networks*, pages 570–582. Springer, 2017.

[57] Orlando Marcos Martins Mancini. Uma breve hitÓriari do midi.

[58] Hsouza Edio Marcos. pentagramas gerados. Disponível em: `https://www.https://`, 2022. Acesso em: 01 de dezembro 2022.

[59] Dragan Matić. A genetic algorithm for composing music. *Yugoslav Journal of Operations Research*, 20(1), 2016.

[60] Guerino Mazzola, Yan Pang, William Heinze, Kyriaki Gkoudina, Gian Afrisando Pujakusuma, Jacob Grunklee, Zilu Chen, Tianxue Hu, and Yiqing Ma. *A Short History of MIDI*, pages 115–116. Springer International Publishing, Cham, 2018.

[61] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[62] Bohumil MED. *Teoria da música*, volume 996. Brasília: Musimed, 1996.

[63] Daniel de Souza Mendes et al. O cálculo e a invençãoo na poética de stockhausen. 2009.

[64] FORMA E METÁFORA and CAIO NELSON DE SENNA NETO. Doutorado em musica.

[65] Werner Meyer-Eppler. *Elektrische Klangerzeugung: Elektronische Musik und synthetische Sprache: mit 122 Abbildungen*. Ferd. Dümmler, 1949.

[66] Evandro Manara Miletto, Leandro Lesqueves Costalonga, Luciano Vargas Flores, Eloi Fernando Fritsch, Marcelo Soares Pimenta, and Rosa Maria Vicari. Introdução à computação musical. In *IV Congresso Brasileiro de Computação*. sn, 2004.

[67] Koray e Silver David e Graves Alex e Antonoglou Ioannis e Wierstra Daan e Riedmiller Martin Mnih, Volodymyr e Kavukcuoglu. Jogando atari com aprendizado de reforço profundo. *arXiv pré-impressão arXiv:1312.5602*.

[68] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[69] Robert A Moog. Midi: musical instrument digital interface. *Journal of the Audio Engineering Society*, 34(5):394–404, 1986.

[70] Andrew J Nelson. *The sound of innovation: Stanford and the computer music revolution*. MIT Press, 2015.

[71] Andrew Ortony and Terence J Turner. What's basic about basic emotions? *Psychological review*, 97(3):315, 1990.

[72] Alexandre Papadopoulos, François Pachet, Pierre Roy, and Jason Sakellariou. Exact sampling for regular and markov constraints with belief propagation. In *International Conference on Principles and Practice of Constraint Programming*, pages 341–350. Springer, 2015.

[73] Theophilo Augusto Pinto. *Música e eletrônica no Brasil-Vôos abortados de uma pesquisa frutífera*. PhD thesis, Universidade de São Paulo, 2017.

[74] Carles Roig, Lorenzo J Tardón, Isabel Barbancho, and Ana M Barbancho. Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems*, 71:419–434, 2014.

[75] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[77] Sepp e outros Schmidhuber, Jürgen e Hochreiter. Memória de curto prazo longa. *Neural Comput*, 9.

[78] Vladimir Alexandro Pereira Silva. A música eletroacústica numa perspectiva histórica. *ICTUS-Periódico do PPGMUS-UFBA— ICTUS Music Journal*, 1, 1999.

[79] João Felipe e Ben-Tal Oded e Korshunova Iryna Sturm, Bob L e Santos. Modelagem e composição de transcrição de música usando aprendizado profundo.

[80] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[81] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[82] Edward L Thorndike. The law of effect. *The American journal of psychology*, 39(1/4):212–222, 1927.

[83] Michel Tokic and Günther Palm. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In *Annual conference on artificial intelligence*, pages 335–346. Springer, 2011.

[84] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[85] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

[86] Christopher JCH Watkins and Peter Dayan. \cal q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[87] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[88] Chelsea C White III and Douglas J White. Markov decision processes. *European Journal of Operational Research*, 39(1):1–16, 1989.

[89] Neal Zaslaw. *Mozart's Modular Minuet Machine*. na, 2005.

[90] Zurück zum Institut für Psychologie. Gems - geneva emotional music scales. Disponível em:`https://www.uibk.ac.at/psychologie/fachbereiche/pdd/personality_assessment/gems/about-the-gems/index.html.en`, 2022. Acesso em: 30 de Agosto de 2022.