

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Paulo Mateus Moura da Silva

Sistemas de Recomendação de *Plugins* de Áudio para Produção Musical

Fortaleza – CE

20 de setembro de 2019

Paulo Mateus Moura da Silva

Sistemas de Recomendação de *Plugins* de Áudio para Produção Musical

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Orientador: Prof. Dr. Amauri Holanda de Souza Júnior

Coorientador: Prof. Dr. César Lincoln Cavalcante Mattos

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

20 de setembro de 2019

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

- S586s Silva, Paulo Mateus Moura da.
Sistemas de Recomendação de Plugins de Áudio para Produção Musical / Paulo Mateus Moura da Silva. -
2019.
126 f. : il. color.
- Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus
Fortaleza, 2019.
Orientação: Prof. Dr. Amauri Holanda de Souza Júnior.
Coorientação: Prof. Dr. César Lincoln Cavalcante Mattos.
1. Plugins de áudio. 2. Produção Musical. 3. Sistemas de Recomendação. 4. Classificação. 5. Filtragem
Colaborativa. I. Título.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Paulo Mateus Moura da Silva

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

**Orientador: Prof. Dr. Amauri Holanda de Souza
Júnior**

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

**Coorientador: Prof. Dr. César Lincoln
Cavalcante Mattos**

Universidade Federal do Ceará (UFC)

Prof. Dr. Ajalmar Rêgo da Rocha Neto
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Prof. Dr. José Antônio Fernandes de Macêdo
Universidade Federal do Ceará (UFC)

Fortaleza – CE
20 de setembro de 2019

*Este trabalho é dedicado àqueles que ainda insistem
na ciência como transformadora de realidades.*

Agradecimentos

No desenvolvimento de um trabalho científico, consultamos um conjunto de obras procurando pontos que possam auxiliar no desenvolvimento do trabalho, clarificar ideias, confrontar pensamentos. Parte se transforma em bibliografia do material produzido. A produção de um trabalho de caráter científico, além de explanar um problema e propor alguma solução, é também motivada na esperança de que o conteúdo desenvolvido possa exercer um papel de referência em trabalhos futuros.

No aspecto da contribuição do trabalho, entretanto, o corpo bibliográfico traduz a parte científica, mas não trata sobre as pessoas que auxiliaram no trabalho diretamente e indiretamente. Assim, utilizo este espaço para agradecer os que auxiliaram para a minha formação e para a realização desta dissertação.

Agradeço, primeiramente, à Deus pelas obras que fez em minha vida.

Agradeço aos meus pais, José e Dora, que batalharam durante toda minha vida (e antes dela) para oferecer oportunidades de ensino aos filhos, muitas das quais não dispuseram durante suas vidas de estudantes. Agradeço às minhas irmãs, Ana e Michelle, e seus respectivos esposos, Jonas e Emanuel, pelo apoio dado e por apresentarem-se como exemplo de obtenção do grau de mestre.

Agradeço à Edeline por todo o companheirismo expressado durante a jornada no ensino superior, pelo apoio em situações nas quais o trabalho aparentava cair em uma linha sem sentido e pelos ouvidos na formação de conjecturas sobre resultados esperados e não esperados. Agradecimentos se estendem às revisões de texto, às recomendações e comentários, às diversas formas de incentivo e à sua paciência.

Agradeço ao governo federal nos períodos correspondentes a minha carreira estudantil na rede federal, que englobam desde o governo do Lula, em meu ensino médio/técnico, até o do recém assumido Bolsonaro. Agradecimentos especiais para as equipes de Lula e da Dilma pela expansão da rede federal de ensino, que permitiu a (re)abertura do ensino integrado no antigo CEFET de Fortaleza e a criação do IFCE – Campus Maracanaú, este no qual obtive o título de bacharel em Ciência da Computação. Agradecimentos também aos protagonistas das lutas ocorridas nos períodos de contingenciamento e de cortes nos quatro governos presidenciais que presenciei, que me mostraram o dever de lutar pelo que se acredita.

Agradecimentos aos professores responsáveis pela minha formação, em especial o Ricardo Guedes, por apresentar a linguagem de programação Python **3**, ao professor Alex Dourado pela formação de caráter através do esporte. Aos professores Ajalmar, Amauri,

Daniel, David, Elder, Francelino, Jean, Sandro, Thiago Alves, Wellington e outros pela formação acadêmica na graduação, primordial para o desenvolvimento do trabalho aqui apresentado. Agradecimentos também aos amigos e colegas que conheci nos corredores e no decorrer das disciplinas.

Agradecimentos ao Programa de Pós-Graduação em Ciência da Computação pela oferta do Mestrado Acadêmico em Ciência da Computação com a linha de pesquisa Inteligência Artificial, área na qual este trabalho se encontra.

Agradeço ao orientador pela apresentação do modelo RBM após pacientemente ouvir os devaneios e desejos relacionados ao projeto da graduação – antecessor ao do mestrado. Agradeço também ao seu cuidado na escolha do que e deveria trilhar para a conclusão do trabalho. Agradeço ao coorientador por ter disponibilizado, inúmeras vezes, um tempo em sua semana corrida e concorrida sugerindo experimentos, discutindo resultados, pontuando questionamentos. Agradeço também pela formação complementar ofertada. Agradecimento a ambos pela ajuda ofertada na realização do trabalho.

Agradecimentos à Universidade Federal do Ceará, aqui na figura do então diretor da Secretaria de Tecnologia da Informação, professor Bento, e do então chefe da Divisão de Sistemas de Informação, Luís César, como também os autores

- dos artigos 81, inciso V, 87 e 102, inciso VIII, alínea “e” da lei 8.112/90 com redação dada pela lei nº 9.527, de 10/12/97 (DOU 11/12/97) e Resolução nº 19/2016/CONSUNI, e,
- das resoluções nº 16/2016/CEPE de 17/10/2016 e nº 21/2016/CEPE de 23/12/16

pelos incentivos à pesquisa.

Agradecimentos a todos aqueles que pacientemente ouviram monólogos ou participaram de diálogos sobre o problema tratado nesta dissertação, incluindo familiares, amigos, colegas de trabalho, de laboratório e de estudo e membros das bancas de qualificação e de defesa.

E, por fim, agradecimentos a todos os anônimos que dispuseram um pouco de seu tempo auxiliando na solução de problemas em diversos fóruns, *blogs* e ferramentas de comentários na internet.

Obrigado a todos!

“Atocha!

...

Lai vai.”

(Ricardo B. de M. Guedes)

Resumo

Atores do processo de produção musical operam diversas ferramentas para obter artefatos musicais de alta qualidade. Uma das ferramentas tipicamente utilizadas é o processador digital multiefeitos, que aplica uma cadeia de transformações não lineares (efeitos) aos sinais de áudio. No entanto, o grande número de configurações disponíveis implica que um melhor uso de tais equipamentos é restrito a especialistas. Nesse contexto, sistemas inteligentes para produção musical visam propor ferramentas automáticas para facilitar e apoiar a tomada de decisão de atores do processo de produção musical. Nesta dissertação, o problema específico de selecionar *plugins* de áudio, i.e., implementações de efeitos, é abordado manipulando-o como tarefas de recomendação de sequências de *plugins* em três casos distintos e independentes: *i*) recomendação sobre sequências parcialmente observadas, *ii*) definição automática da ordem dos *plugins* em uma sequência, e, *iii*) recomendação completa de sequência de *plugins*. Para estas tarefas, são aplicados dois paradigmas de aprendizagem de máquina: aprendizagem supervisionada e filtragem colaborativa. No primeiro, classificadores multiclasse amplamente utilizados são aplicados para prever *plugins*/classes em uma posição fixa de uma sequência dado os elementos/*plugins* restantes dessa sequência. Os modelos avaliados foram o Perceptron Multicamadas, a Regressão Logística e a Máquina de Vetores Suporte. No segundo, modelos para filtragem colaborativa procuram executar previsões automáticas baseadas em sequências semelhantes no conjunto de dados disponível. Os modelos *k*-vizinhos mais próximos e a Máquina de Boltzmann Restrita para Filtragem Colaborativa (RBM-CF) foram avaliados. Todos os modelos foram treinados usando dados coletados de repositórios online. Com base nos experimentos, as abordagens de aprendizagem supervisionada superam as de filtragem colaborativa para a tarefa de recomendações com informações de sequência parcial. No entanto, o paradigma de aprendizagem supervisionada não é escalável ao tamanho da sequência, pois requer o treinamento de uma grande quantidade de modelos. Devido à sua capacidade generativa, apenas a RBM-CF pôde ser aplicada para a recomendação de sequências completas e a definição da ordem dos *plugins* de áudio.

Palavras-chave: *plugins* de áudio. produção musical. sistemas de recomendação. classificação. filtragem colaborativa.

Abstract

Music actors operate several tools to achieve high-quality musical artifacts. One of the typically used tools is the digital multi-effects processor, which applies a chain of non-linear transformations (effects) to audio signals. However, the large number of available configurations implies that better use of such equipment is restricted to specialists. In this context, intelligent systems for music production aim to propose automatic tools to ease and support decision making by music actors. In this work, we tackle the problem of sequence-dependent audio plugin (effect implementation) recommendation from data. In particular, we are interested in three settings: *i*) recommendations under partially observed plugin sequences; *ii*) determining the best ordering of plugins in a sequence; *iii*) full recommendations of plugin sequences. For these tasks, we employ two machine learning paradigms: supervised learning and collaborative filtering. In the first, we apply standard classifiers to predict plugins/classes in a fixed position within a sequence given the remaining sequence elements/plugins. We evaluate Multilayer Perceptron, Logistic Regression, and Support Vector Machines. On the other hand, models for collaborative filtering seek to perform automatic predictions based on similar sequences from the available data. We evaluate the k -nearest neighbors method, and the Restricted Boltzmann Machine for Collaborative Filtering (RBM-CF) model. All models are trained using data collected from online repositories. Based on the experiments, supervised learning approaches outperform those of collaborative filtering for the task of recommendations under partial sequence information. However, the supervised learning paradigm does not scale with sequence size as it requires training a large number of models. Given its generative capability, only RBM-CF can be applied to full-sequence recommendations, and to determining plugin orderings.

Keywords: audio plugin. music production. recommendation systems. classification. collaborative filtering.

Lista de ilustrações

Figura 1 – Diagrama de recomendação de <i>plugins</i> de áudio por um sistema de recomendação	31
Figura 2 – Aplicação de efeitos em sinais de áudio por atores do processo de produção musical	36
Figura 3 – Mapa mental das relações entre as definições referentes a <i>unidades de efeito</i> e ao <i>Processamento de sinais de áudio</i>	37
Figura 4 – Unidades de efeitos analógicas e digital	39
Figura 5 – Um <i>patch</i> para a unidade multiefeitos MOD Duo permite <i>plugins</i> conectados em paralelo	40
Figura 6 – Diagrama com casos de recomendação abordados no trabalho	42
Figura 7 – Processo de recomendação pelo modelo kNN de um <i>plugin</i> de áudio dado outros selecionados pelo usuário	48
Figura 8 – Tratamento dos dados para classificação em <i>patches</i> com seis posições e oito possíveis <i>plugins</i>	52
Figura 9 – Representação gráfica de uma MLP com D unidades de entrada, L camadas ocultas e C unidades de saída	54
Figura 10 – Problema de classificação binária em duas dimensões. Padrões de mesma classe com mesma cor. Hiperplano de separação ótimo h^* encontrado por um classificador SVM. Vetores suportes circulados.	55
Figura 11 – Representação gráfica de uma RBM com quatro unidades visíveis e três unidades ocultas	59
Figura 12 – Atualização de Gibbs em bloco: atualização dos elementos da camadas visível e oculta de uma RBM	66
Figura 13 – Representação gráfica de uma RBM-CF com seis unidades visíveis para o problema de recomendação de filmes: as três primeiras unidades correspondem ao <i>rating</i> para o primeiro filme e as três unidades seguintes para o segundo filme	69
Figura 14 – Exemplo de <i>patch</i> para a unidade de efeitos Zoom G3	75
Figura 15 – Distribuição da quantidade de instâncias por <i>plugin</i> contidas na base de dados	77
Figura 16 – Diagrama de Sankey para a base de dados coletada indicando as categorias das instâncias por posição na cadeia de efeitos	77
Figura 17 – Pontuação atribuída pela métrica DCG para itens em uma lista de recomendações por sua posição (<i>rank</i>)	79
Figura 18 – Procedimento para gerar <i>patches</i> incompletos distintos a partir de um único <i>patch</i>	84

Figura 19 – Métrica Hit@5 para os modelos LR, kNN e RBM-CF na tarefa de recomendação de <i>plugins</i> para várias posições em um mesmo <i>patch</i> . . .	87
Figura 20 – Proporção de instâncias posicionadas corretamente pela RBM-CF pelo número de entradas/componentes ausentes	88
Figura 21 – Representação gráfica de uma BM com seis unidades	109

Lista de tabelas

Tabela 1	–	Categorias dos <i>plugins</i> de áudio presentes no Zoom G3 com a quantidade de <i>plugins</i> e a ocorrência por categoria	76
Tabela 2	–	Hiperparâmetros testados na seleção do modelo	84
Tabela 3	–	Resultado dos experimentos para recomendação de <i>plugins</i> para uma única posição de um <i>patch</i>	85
Tabela 4	–	Categorias de 60 <i>plugins</i> de áudio dos 117 presentes no equipamento Zoom G3	123
Tabela 5	–	Categorias dos 57 <i>plugins</i> de áudio restantes dos 117 presentes no equipamento Zoom G3	124

Lista de Algoritmos

1	Gradiente descendente. Adaptado de BOYD; VANDENBERGHE, 2004. . .	61
2	Um algoritmo MCMC ingênuo para minimizar a log-verossimilhança com uma função de partição intratável usando gradiente descendente. Adaptado de GOODFELLOW; BENGIO; COURVILLE, 2016.	64
3	<i>Contrastive Divergence</i> com k passos (CD- k). Adaptado de GOODFELLOW; BENGIO; COURVILLE, 2016.	65
4	<i>Persistent Contrastive Divergence</i> com k passos (PCD- k). Adaptado de GOODFELLOW; BENGIO; COURVILLE, 2016.	65
5	<i>Gibbs update</i> em bloco. Todos os elementos de uma mesma camada são amostrados simultaneamente.	66
6	Predição de uma avaliação para o m -ésimo grupo.	70
7	Amostragem de Gibbs em uma <i>Boltzmann Machine</i>	111

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
BM	<i>Boltzmann Machine</i> – Máquina de Boltzmann
CD	<i>Contrastive Divergence</i>
CF	<i>Collaborative Filtering</i> – Filtragem colaborativa
DAFX	<i>Digital Audio Effects</i>
DSP	<i>Digital Signal Processor</i>
GD	<i>Gradient Descent</i> – Gradiente Descendente
kNN	<i>k-Nearest Neighbors</i> – k-Vizinhos mais próximos
LMS	<i>Least Mean Square</i> – Gradiente Descendente Estocástico
LR	<i>Logistic Regression</i> – Regressão Logística
MAP@ <i>k</i>	<i>Mean Average Precision</i>
MC	<i>Markov Chain</i> – Cadeia de Markov
MCMC	<i>Markov Chain Monte Carlo</i>
MLP	<i>Multi Layer Perceptron</i> – Perceptron Multicamadas
NDCG	<i>Normalized Discounted Cumulative Gain</i>
NLL	Negative log-likelihood – Negativa log-verossimilhança
OLS	<i>Ordinary Least Square</i> – Mínimos Quadrados Ordinário
PCD	<i>Persistent Contrastive Divergence</i>
PMF	<i>Probability mass function</i> – Função de massa de probabilidade
RBF	<i>Radial Basis Function</i>
RBM	<i>Restricted Boltzmann Machine</i> – Máquinas de Boltzmann Restritas
RBM-CF	<i>Restricted Boltzmann Machine for Collaborative Filtering</i> – Máquina de Boltzmann Restrita para Filtragem Colaborativa
SML	<i>Stochastic Maximum Likelihood</i>
SVM	<i>Support Vector Machine</i> – Máquinas de Vetores Suporte

Lista de símbolos

$\langle \cdot \rangle_{P(\cdot)}$	Esperança com respeito a distribuição $P(\cdot)$
$\mathbf{x}^T$	Vetor $\mathbf{x}$ transposto
$\mathbf{0}$	Vetor cujo todos os seus elementos são zeros
$\mathbf{1}$	Vetor cujo todos os seus elementos são uns
$C = A \setminus B$	C e um conjunto formado pelos elementos de A exceto os de B
$\mathbf{x}_{-m}$	Notação alternativa à $\mathbf{x} \setminus \{x_m\}$
$a \perp b$	variável aleatória a é independente da variável aleatória b
$a \perp b c$	variável aleatória a é condicionalmente independente da variável aleatória b dado a variável aleatória c
$\hat{p}$	Aproximação de p
$x \sim \mu(\cdot)$	x é amostrado da distribuição de probabilidade $\mu(\cdot)$
$\ln(\cdot)$	Operador logaritmo natural
$\sigma(\cdot)$	Função sigmoide ou logística
$\text{relu}(\cdot)$	Função <i>Rectified Linear Unit</i>
$\tanh(\cdot)$	Função tangente hiperbólica
$\text{softmax}(\cdot)$	Função <i>softmax</i>
$\text{soft}_+(\cdot)$	Função <i>softplus</i>
Θ	Conjunto dos parâmetros do modelo
θ	Um parâmetro qualquer do modelo ($\theta \in \Theta$)

Sumário

1	INTRODUÇÃO	29
1.1	Objetivos Geral e Específicos	31
1.2	Estrutura da dissertação	32
2	PROCESSAMENTO E EFEITOS DE ÁUDIO	35
2.1	Contextualização	35
2.2	Definições básicas	37
2.3	Trabalhos relacionados	40
2.4	Discussão	42
3	RECOMENDAÇÃO AUTOMÁTICA DE <i>PLUGINS</i> DE ÁUDIO	45
3.1	Definição formal	46
3.2	Recomendação como problema de classificação	46
3.3	Filtragem colaborativa	47
3.3.1	<i>k</i> -vizinhos mais próximos	47
3.3.2	Máquinas de Boltzmann Restritas	48
3.3.3	Comentários sobre filtragem colaborativa de <i>plugins</i> de áudio	49
3.4	Discussão	49
4	CLASSIFICADORES PARA RECOMENDAÇÃO DE <i>PLUGINS</i> DE ÁUDIO	51
4.1	Regressão Logística	52
4.2	Perceptron Multicamadas	53
4.3	Máquina de Vetores Suporte	55
4.4	Discussão	56
5	MÁQUINAS DE BOLTZMANN RESTRITAS	57
5.1	Restricted Boltzmann Machine	57
5.1.1	Representação de uma RBM	58
5.1.2	Marginalização da camada visível	59
5.1.3	Aprendizagem de uma RBM	60
5.1.4	Algoritmos MCMC com amostragem aproximada	63
5.1.4.1	Algoritmo ingênuo para MCMC	63
5.1.4.2	Algoritmo Contrastive Divergence	63
5.1.4.3	Algoritmo Persistent Contrastive Divergence	64
5.1.4.4	Gibbs sampling em bloco para RBM	65

5.1.5	Discussão	66
5.2	Restricted Boltzmann Machine para Filtragem Colaborativa	68
5.2.1	Representação de uma RBM-CF	68
5.2.2	Realização de predições	70
5.2.3	Discussão	71
5.3	Restricted Boltzmann Machine para recomendação de <i>plugins</i> de áudio	72
6	METODOLOGIA PROPOSTA	75
6.1	Base de dados	75
6.1.1	Categorização dos <i>plugins</i> e mineração dos dados	76
6.2	Métricas de avaliação	78
6.3	Discussão	80
7	EXPERIMENTOS COMPUTACIONAIS	83
7.1	Configuração dos experimentos	83
7.1.1	Seleção do modelo	84
7.2	Recomendação de <i>plugins</i> para uma única posição	85
7.3	Recomendação de <i>plugins</i> para múltiplas posições	86
7.4	Posicionamento das instâncias de <i>plugins</i> em um <i>patch</i> com a RBM-CF	87
7.5	Recomendação de um <i>patch</i> completo com a RBM-CF	88
7.6	Discussão	91
8	CONCLUSÃO	93

REFERÊNCIAS	95
------------------------------	-----------

APÊNDICES 105

APÊNDICE A – FÓRMULAS RELACIONADAS A REDES DE MARKOV	107
---	------------

A.1	PMF das variáveis aleatórias de uma Rede de Markov baseada em energia	107
A.2	Log-verossimilhança e seu gradiente	107
A.2.1	Log-verossimilhança $\ln P(\mathbf{v})$	107
A.2.2	Gradiente da log-verossimilhança $\nabla_{\theta} \ln P(\mathbf{v})$	107
A.2.2.1	Termo $\nabla_{\theta} \ln \tilde{P}(\mathbf{v})$	108
A.2.2.2	Gradiente do log da função de partição de uma Rede de Markov: $\nabla_{\theta} \ln Z$	108

	APÊNDICE B – MÁQUINA DE BOLTZMANN	109
B.1	Representação de uma BM	109
B.2	Amostragem de Gibbs	110

	APÊNDICE C – FÓRMULAS RELACIONADAS A MÁQUINA DE BOLTZMANN RESTRITA	113
C.1	Probabilidades Marginais para uma RBM	113
C.1.1	Probabilidade Marginal $P(\mathbf{v})$ de uma RBM	113
C.1.2	Probabilidade Marginal $P(\mathbf{h})$ de uma RBM	114
C.2	Gradiente da função objetivo de uma RBM	114
C.2.1	Termo $\ln P(\mathbf{v})$	114
C.2.2	Termo $\nabla_{\theta} \ln P(\mathbf{v})$	115
C.2.3	Termo $\nabla_{\theta} f(\theta, \mathcal{D})$	115
C.2.4	Gradiente da função de energia de RBM em relação a cada um dos parâmetros do modelo	116
C.2.4.1	Função gradiente de $F(\mathbf{v})$ em relação a matriz de pesos: $\nabla_{\mathbf{W}} F(\mathbf{v})$	116
C.2.4.2	Vetor gradiente de $F(\mathbf{v})$ em relação ao vieses da camada oculta: $\nabla_{\mathbf{c}} F(\mathbf{v})$	117
C.2.4.3	Vetor gradiente de $F(\mathbf{v})$ em relação ao vieses da camada visível: $\nabla_{\mathbf{b}} F(\mathbf{v})$	117
C.2.5	Relação entre derivadas parciais de $F(\mathbf{v})$ e $\mathbb{E}_{P(h_i \mathbf{v})}[h_i]$	118
C.3	Probabilidades Condicionais	119
C.3.1	Probabilidade Condicional dos elementos ocultos dado os visíveis: $P(\mathbf{h} \mathbf{v})$.	119
C.3.1.1	Probabilidade Condicional $P(h_i \mathbf{v})$ em uma RBM	119
C.3.1.2	Independência Condicional	119
C.3.1.3	Probabilidade Condicional $P(h_i = 1 \mathbf{v})$ em uma RBM	120
C.3.2	Probabilidade Condicional dos elementos visíveis dado os ocultos: $P(\mathbf{v} \mathbf{h})$.	120
C.3.2.1	Probabilidade Condicional $P(v_j = 1 \mathbf{h})$ em uma RBM	120

ANEXOS 121

ANEXO A – PLUGINS DE ÁUDIO PARA A UNIDADE DE EFEI- TOS ZOOM G3	123
---	------------

Índice	125
-------------------------	------------

1 Introdução

Da concepção musical de um compositor à música escutada em serviços de *stream* de áudio, o som é trabalhado por uma série de profissionais que buscam trazer uma qualidade e experiência sonora que atenda diversos públicos e contextos sociais. A produção e distribuição musical, a partir do desenvolvimento das técnicas na indústria fonográfica, tornou-se uma tarefa cuja dinâmica envolve muito mais que o tríduo compositor, intérprete e ouvinte (MACEDO, 2007).

Cantores, instrumentistas, *DJs*, técnicos e engenheiros de áudio utilizam ferramentas de áudio sofisticadas para manipular sinais de áudio. Unidades de efeito — sistemas embarcados para este propósito — e programas para computadores são algumas das ferramentas recorridas que facilitam e ampliam as possibilidades de manipulação tanto em *performances ao vivo* (LEVITAN, 1992) quanto em estúdio (IZHAKI, 2013).

Plugins de áudio, algoritmos para o processamento de sinais digitais implementados sobre uma *API* de áudio (SCHIAVONE; GOULART; QUEIROZ, 2012), exercem um papel significativo na produção musical. Executados em DSPs (*Digital Signal Processors*) ou em computadores de propósito geral, realizam a emulação de unidades de efeitos e a sintetização de sons. A variedade existente de *plugins* por um custo inferior a equipamentos equivalentes, aliada a uma interface para controle amigável no ambiente que é executado, traz facilidades ao processo de criação musical (AUGUSTO; CASTRO, 2015).

Entretanto, uma alta possibilidade de configurações torna-se ineficiente quando a capacidade de reconhecer quais efeitos deverão ser aplicados sobre determinado contexto musical, e, conseqüentemente, quais os *plugins* que deverão ser utilizados em uma configuração (*patch*) de uma ferramenta, é restrita a especialistas. Como forma de amenizar esse problema, a comunidade disponibiliza suas configurações compartilhando-as em repositórios mantidos por ela própria ou por empresas que criam os *plugins* de áudio.

A disponibilidade de dados, juntamente com a dificuldade em modelar matematicamente, a partir de princípios físicos, as diferentes configurações de áudio utilizadas por profissionais da indústria fonográfica, abre espaço para aplicação de tecnologias voltadas ao apoio da tomada de decisão por atores do processo de produção musical (REISS, 2011).

A busca de sistemas inteligentes para produção musical envolvendo *plugins* de áudio concentra-se nas tarefas de *i) ajuste de parâmetro*, *ii) identificação de efeitos* e *iii) recomendação de plugins*. O primeiro é relacionado à seleção automática dos valores mais adequados dos parâmetros dos *plugins*, com trabalhos envolvendo a identificação automática de valores para os parâmetros com base em amostras de áudio (PETERS; CHOI; LEI, 2012), controle adaptativo seguindo abordagens lineares (STARK; DAVIES;

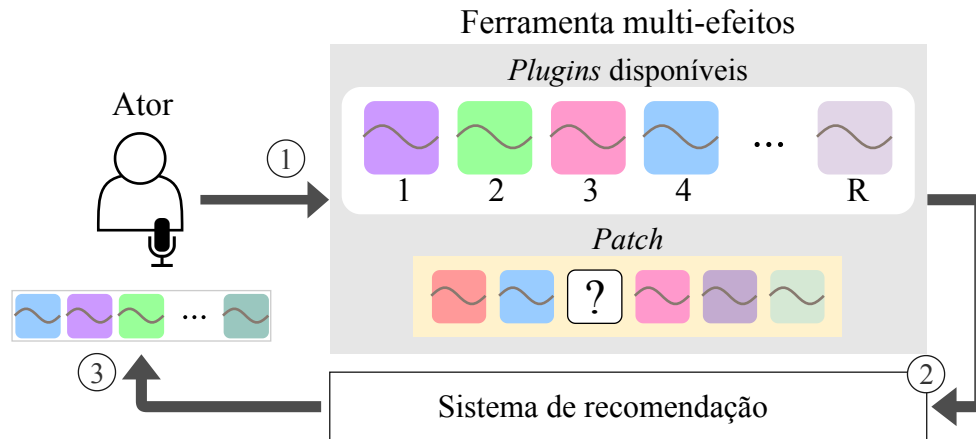
PLUMBLEY, 2008) ou não lineares (CHOURDAKIS; REISS, 2017; RAMÍREZ; REISS, 2018). O segundo trata da identificação automática de efeitos aplicados em sinais de áudio (STEIN et al., 2010; SCHMITT; SCHULLER, 2017). Já o terceiro, relacionado ao tema trabalhado nesta dissertação, trata da tarefa de selecionar *plugins* a partir de um conjunto de opções disponíveis.

Nesta dissertação, é investigado o uso de técnicas de aprendizagem de máquina para recomendação de *plugins* de áudio. São seguidas duas abordagens distintas. Na primeira, o problema de recomendação de *plugins* é formulado como uma tarefa de aprendizagem supervisionada. Nela, são avaliados algoritmos de aprendizagem de máquina amplamente difundidos. Na segunda, o problema é abordado com o arcabouço de filtragem colaborativa. Para este, empregamos uma abordagem generativa na qual são modeladas distribuições de probabilidade discretas sobre *plugins* de áudio para produzir recomendações. Em ambas, as recomendações de *plugins* são baseadas nas experiências de usuários de uma unidade de efeito específica, i.e., o conjunto de dados utilizado para treinamento foi construído sobre dados reais.

Na comunidade de aprendizagem de máquina, a Máquina de Boltzmann Restrita (RBM) tem despertado grande interesse entre os pesquisadores devido a sua capacidade de gerar amostras a partir de uma distribuição probabilística aprendida diretamente dos dados disponíveis (HINTON, 2002; BREULEUX; BENGIO; VINCENT, 2011). Como um bom extrator de características, sua camada latente (oculta) pode ser usada, por exemplo, como uma etapa de pré-treinamento em modelos de aprendizagem profunda (HINTON, 2007). Ainda, as RBMs também podem ser usadas como classificadores não lineares (LAROCHELLE; BENGIO, 2008). No contexto de tarefas de recomendação, RBMs já foram aplicadas com filtragem colaborativa em conjuntos com grandes quantidades de padrões (SALAKHUTDINOV; MNIH; HINTON, 2007; GEORGIEV; NAKOV, 2013). Assim, RBMs denotam a classe de modelos generativos considerados ao longo deste trabalho.

Na metodologia proposta, o sistema gera uma lista de recomendações de *plugins* considerando um conjunto parcial de *plugins* previamente definidos por um usuário. A Figura 1 ilustra o funcionamento do sistema de recomendação proposto. No passo 1, o usuário escolhe um conjunto parcial de *plugins*, que é usado como entrada para o sistema (passo 2). Por fim, o passo 3 consiste na recomendação do sistema para as entradas ausentes.

Figura 1 – Diagrama de recomendação de *plugins* de áudio por um sistema de recomendação. **Passo 1.** Um ator do processo de produção musical define *plugins* de áudio para um *patch*. Blocos coloridos no *patch* são *plugins* previamente definidos, enquanto o bloco de interrogação indica um *plugin* não definido. Todos os *plugins* disponíveis são candidatos para o *plugin* não definido. **Passo 2.** Um sistema de recomendação carrega o *patch* e os *plugins* disponíveis e no **Passo 3** retorna os candidatos classificados pela probabilidade de uso na posição indicada pelo bloco de interrogação.



Fonte: Próprio autor.

Para mitigar o problema de seleção entre uma grande quantidade de combinações de *plugins* de áudio, é proposto o treinamento de sistemas de recomendação pela extração automática do conhecimento de um repositório mantido pela comunidade de usuários de uma ferramenta¹. Em particular, foi considerada uma ferramenta multiefeitos para guitarras elétricas que oferece mais de cem *plugins* de áudio.

1.1 Objetivos Geral e Específicos

O objetivo geral da presente dissertação consiste em desenvolver sistemas de recomendação para suporte na tomada de decisão por atores do processo de produção musical na seleção de *plugins* de áudio, sendo estes seguindo duas metodologias distintas: aprendizagem supervisionada e filtragem colaborativa.

O escopo geral pode ainda ser agrupado em quatro objetivos específicos:

OE1 Introduzir e formular o problema da recomendação de *plugins* de áudio em *patches* incompletos, apresentando ainda uma contextualização e trabalhos até então desenvolvidos;

¹ Repositório Guitar patches pode ser acessado em <<https://guitarpatches.com/units.php>>.

- OE2** Preparar e disponibilizar um novo conjunto de dados preparado a partir de informações reais obtidas de repositório público de configurações de *plugins* de áudio, enumerando os passos de normalização e padronização dos dados;
- OE3** Definir e avaliar a aplicação de métodos de aprendizagem supervisionada para a recomendação de *plugins* de áudio;
- OE4** Definir e avaliar a aplicação de modelos para filtragem colaborativa de *plugins* de áudio, incluindo modelos gerativos baseados em RBM.

Com relação ao objetivo **OE1**, a ênfase está na apresentação do problema. Para o objetivo **OE2**, além do caráter científico ao servir de insumo para trabalhos similares, este possui cunho tecnológico, com oferta de matéria para a construção de sistemas computacionais de recomendação automática eficientes, bem como uma especificação de tratamento de dados para ampliação da base trabalhada. O foco dos objetivos **OE3** e **OE4** é no aspecto algorítmico e metodológico das soluções desenvolvidas. Noções de precisão e eficiência dos modelos devem ser consideradas em tais desenvolvimentos. Mais especificamente, o objetivo **OE4** é motivado pelo estabelecimento de um *baseline* para o problema proposto (objetivo **OE1**), comparando vantagens e limitações contra métodos clássicos e motivado por resultados promissores alcançados através de pré-processamento usando modelos de aprendizagem profunda.

1.2 Estrutura da dissertação

O conteúdo da dissertação está organizado com a seguinte estrutura:

Capítulo 2 expõe informações sobre a utilização de *plugins* de áudio no processo de produção musical (Seção 2.1), definições e conceitos relacionados ao processamento de sinais de áudio e sua aplicação no contexto musical, útil como material de referência (Seção 2.2) e trabalhos correlatos ao tema de recomendação de *plugins* de áudio (Seção 2.3).

Capítulo 3 apresenta detalhes sobre o problema e relaciona os conceitos discutidos no capítulo anterior com a problemática. A Seção 3.1 apresenta uma definição formal matemática para o caso em estudo. A Seção 3.2 inicia a discussão sobre a resolução por uma estratégia envolvendo classificadores (continuada no Capítulo posterior). A Seção 3.3 trata da apresentação da estratégia de filtragem colaborativa, explica a utilização de k -vizinhos mais próximos (Subseção 3.3.1) e introduz o modelo generativo denominado Máquina de Boltzmann Restrita para Filtragem Colaborativa (Subseção 3.3.2), aprofundado no Capítulo 4.

Capítulo 4 dá maiores detalhes sobre como utilizar classificadores para a recomendação de *plugins* de áudio. Ainda, suas seções apresentam os classificadores utilizados: Seção 4.1 – Regressão Logística, Seção 4.2 – Perceptron Multicamadas e Seção 4.3 – Máquinas de Vetores Suporte.

Capítulo 5 aborda os modelos Máquina de Boltzmann Restritiva (Seção 5.1) e Máquina de Boltzmann Restrita para Filtragem Colaborativa (Seção 5.2). Além da apresentação dos modelos, incluindo suas representações, o processo de aprendizagem e de predição, no final de ambas as seções há discussões com informações adicionais e extensões aplicáveis para os modelos. No final deste capítulo, a Seção 5.3 relaciona a teoria apresentada com o problema de recomendação de *plugins* e indica as diferenças no uso dos modelos em comparação com recomendação de filmes, problema de filtragem colaborativa clássico.

Capítulo 6 apresenta a metodologia do trabalho. A Seção 6.1 enumera os passos da construção do conjunto de dados utilizado nos experimentos, discute a categorização de *plugins*, informa o desbalanceamento dentre as categorias e entre a quantidade de uso dos *plugins*, traz informações sobre padrões de uso de *plugins* em conjunto. A Seção 6.2 apresenta as métricas definidas para a avaliação dos modelos nos experimentos.

Capítulo 7 descreve os experimentos computacionais. A Seção 7.1 aborda a configuração dos experimentos, incluindo o processo de validação cruzada, definição de hiperparâmetros e seleção dos modelos. As seções seguintes abordam os casos testados: Recomendação de *plugins* para uma única posição (Seção 7.2), recomendação para múltiplas posições (Seção 7.3), recomendação de um *patch* completo (Seção 7.5) e definição de posicionamento de instâncias de *plugins* em um *patch* (Seção 7.4). Para cada experimento, há comentários que apresentam diferenças entre os testes, além de vantagens e desvantagens das abordagens tratadas nestes cenários.

Capítulo 8 traz a conclusão do trabalho, com considerações finais sobre os experimentos, apresentação de problemas similares em que a metodologia desenvolvida pode tornar-se útil para o desenvolvimento de soluções e enumerações de propostas de trabalhos futuros.

Por fim, como conteúdo adicional, encontram-se formulações referentes à Redes de Markov (Apêndice A), uma apresentação de uma Máquina de Boltzmann (Apêndice B) — modelo no qual deu origem às Máquinas de Boltzmann Restritas — e o desenvolvimento das fórmulas envolvendo Máquina de Boltzmann Restrita (Apêndice C). Ainda, sobre a base de dados utilizada nos experimentos, é apresentado o catálogo de *plugins* com suas respectivas categorias da unidade multiefeitos (Anexo A).

2 Processamento e Efeitos de Áudio

A manipulação de áudio é um recurso importante para atores do processo de produção musical. Tal recurso possibilita a aplicação de efeitos no áudio pelo processamento de seus sinais. Neste sentido, *plugins* de áudio, objetos de recomendação dos sistemas de aprendizagem abordados neste trabalho, apresentam-se como um dos principais meios para a aplicação dos efeitos no sinal.

Este capítulo traz motivações sobre a aplicação de ferramentas para manipulação de áudio, define conceitos para um melhor entendimento sobre elas e discorre sobre os problemas tratados na literatura. Especificamente, a Seção 2.1 apresenta o contexto em que *plugins* de áudio são utilizados. Em seguida, a Seção 2.2 traz definições de conceitos relacionados ao processamento de sinais de áudio e sua aplicação no âmbito musical. Por fim, a Seção 2.3 expõe trabalhos correlatos ao tema de recomendação de *plugins* de áudio.

Caso o leitor se sinta confortável com o tema, sugere-se que prossiga para a Seção 2.3. A Seção 2.2 pode ser utilizada como material de consulta.

2.1 Contextualização

Atores do processo de produção musical trabalham o som para atingir uma experiência sonora desejada. Pertencem a esse grupo cantores, instrumentistas, DJs, técnicos e engenheiros de áudio, dentre outros, representados como *atores* na parte superior da Figura 2.

A manipulação no sinal de áudio ocorre pela aplicação de *efeitos* sobre o áudio, i.e., pela execução de transformações lineares e não lineares parametrizáveis sobre um sinal de áudio. Dentre diversas motivações, encontram-se:

- um cantor pode adicionar uma *reverberação* na voz para parecer que está cantando em um ambiente fechado, como em um banheiro ou em um teatro vazio;
- um instrumentista pode alterar a *equalização* do sinal de áudio de seu instrumento em um solo para destacar determinadas frequências;
- um engenheiro de áudio pode aplicar um *auto tune* para corrigir a afinação de algum vocal ou instrumento;
- um técnico de áudio pode adicionar *ganho* sobre um sinal de áudio para amenizar perdas causadas por cabos longos;
- um DJ pode aplicar um *fader* no final de uma música para que o som diminua gradativamente até chegar em um volume inaudível.

Figura 2 – Uma unidade de efeito aplica um ou mais efeitos de áudio sobre um sinal de entrada. Um efeito aplicado possui parâmetros, os quais são ajustados por *atores* do processo de produção musical. Ícones feitos por Freepik, itim2101, Flat Icons, xnimrodx e smalllikeart de <www.flaticon.com>.



Fonte: Próprio autor.

A aplicação dos efeitos ocorre pela utilização de ferramentas denominadas unidades de efeito. A transformação é ilustrada na parte central na Figura 2: atores selecionam efeitos dentre os disponíveis nas unidades de efeito e os aplicam sobre um sinal de entrada, gerando assim um sinal de saída transformado de acordo com a parametrização definida para cada efeito.

Um efeito pode ser aplicado sobre o sinal por uma unidade de efeito cujo seu circuito é totalmente analógico. É possível, ainda, que as transformações sejam aplicadas sobre o sinal em sua forma digital. Nesta forma, um algoritmo de transformação linear ou não linear é denominado *plugin* de áudio. *Plugins* de áudio são desenvolvidos para serem executados em sistemas embarcados de propósito específico ou em computadores de propósito geral, respeitando a arquitetura do sistema computacional em que o algoritmo é embarcado.

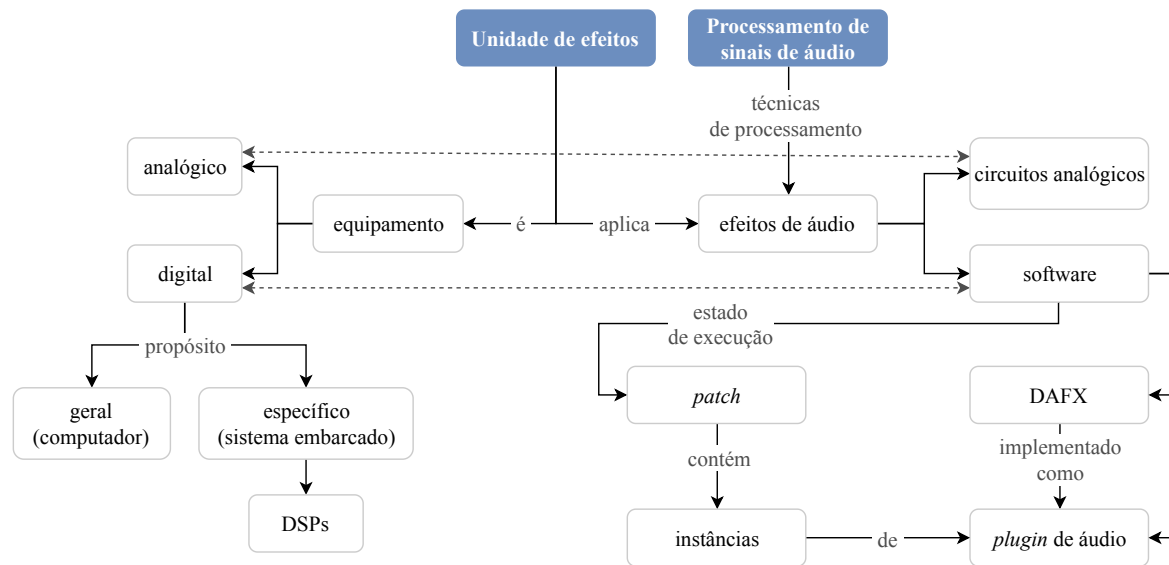
A utilização de *plugins* de áudio ao invés de efeitos analógicos é preferida em diversos cenários. Dentre as motivações, podem ser citadas a ampla disponibilidade de *plugins*, sua qualidade, um custo acessível dos equipamentos que o executam e a facilidade de obtenção. Entretanto, o uso em conjunto de diversas unidades de efeito não é descartado.

Assim, atores musicais têm em mãos ferramentas com uma alta possibilidade de configurações. Neste contexto, ferramentas automáticas para apoio da tomada de decisões nas diversas etapas de produção musical tornam-se úteis, desde uma melhor eficiência de configuração por especialistas quanto um apoio e ponto de aprendizagem para iniciantes na área.

2.2 Definições básicas

Nesta seção são apresentadas definições de termos utilizados nesta dissertação. Alguns dos termos são acompanhados de imagens para facilitar o entendimento. Na Figura 3 a seguir, há um breve resumo do relacionamento entre os conceitos.

Figura 3 – Mapa mental das relações entre as definições referentes a *unidades de efeito* e ao *Processamento de sinais de áudio*.



Fonte: Próprio autor.

Processamento de sinais de áudio: Subárea de processamento de sinais com foco na manipulação eletrônica de sinais de áudio. O processamento pode ser aplicado no áudio nos seus formatos analógico e digital.

Em formato analógico, o sinal de áudio é representado matematicamente por uma função contínua de uma ou mais variáveis contínuas e independentes (BOUZERDOUM; CHAI; BOUSSAID, 2009). O processamento neste campo ocorre diretamente na sua forma elétrica analógica, geralmente usando componentes como resistores, capacitores, indutores e transistores.

Em formato digital, as operações de processamento de sinal são aplicadas na forma de onda de áudio em sua representação discreta, geralmente como uma *stream* binária (ZOELZER, 2011). O áudio manipulado pode ser obtido por um conversor analógico-digital (ADC) em tempo real, em um áudio previamente convertido ou ainda em um sinal de áudio gerado eletronicamente (áudio sintetizado). O sinal é processado em circuitos digitais, como processadores de sinal digital de propósito específico (DSP – *Digital Signal Processor*), microprocessadores e processadores de uso geral.

Efeito: Segundo Zoelzer (2011, Capítulo 1), “denota uma impressão produzida na

mente de uma pessoa, uma mudança na percepção resultante de uma causa”. No corpo da dissertação, “efeito” pode ser usado para referenciar “efeito de áudio”.

Efeitos de áudio: Técnicas de processamento de sinal de áudio aplicadas com o intuito de modificar sinais audíveis para obter uma alteração perceptível dos atributos sonoros, como volume, tom ou timbre (STEIN et al., 2010; VERFAILLE; ZOLZER; ARFIB, 2006). Dentre os efeitos mais comuns aplicáveis em sinais oriundos de instrumentos musicais, encontram-se:

- *Compressor*: controla a dinâmica de um nível de som para diminuir a flutuação, elevando o nível de sinais baixos e diminuindo o nível de sinais altos, para que não haja muita diferença entre elas (GIBSON, 2006, Chapter 7);
- *Chorus*: deixa o som “mais gordo” ao misturar o sinal original com uma cópia levemente desafinada (GIBSON, 2006). Inspirada em corais, onde um grupo de pessoas cantam partes da música juntas;
- *Delay*: repete o sinal introduzido após um intervalo de tempo (KADIS, 2012, Capítulo 10). Uma repetição pode ser o mesmo sinal ou uma versão “colorida”, em que determinadas frequências são acentuadas ou atenuadas. Algumas subcategorias são *eco*, *delay* de fita, *delay* digital, *delay* reverso;
- *Distortion*: gera distorção sobre o som geralmente aumentando seu ganho. A intensidade é usualmente o critério para a classificação do tipo de distorção, indo desde uma distorção “leve” (*overdrive*) a uma distorção extrema (alto ganho ou *high gain*);
- *Equalizer*: incrementa ou decrementa o ganho de frequências de áudio específicas;
- *Reverb*: introduz uma reverberação artificial do som – persistência do som depois que o som é produzido em um lugar (VALENTE; HOSFORD-DUNN; ROESER, 2000). Exemplos de locais são igreja, teatro, banheiro.

Unidade de efeito: Equipamento capaz de aplicar um único efeito ou simultaneamente mais de um (multiefeitos ou *multi-effects*) por um processamento em formato analógico ou digital.

No formato analógico, são utilizados componentes como resistores, transistores, capacitores para a aplicação dos filtros não lineares sobre os sinais de áudio. No formato digital, o equipamento é construído como um sistema embarcado de propósito específico, onde os efeitos de áudio são executados em DSP (*Digital Signal Processor*), processadores especializados em processamento digital de sinal. Ademais, unidades de efeito são comumente utilizadas em conjunto (Figura 4).

Figura 4 – Da esquerda para direita, unidades de efeito analógicas (sob caixas azuis) e unidade multiefeitos digital (sob caixa vermelha). Parâmetros das unidades sob caixas amarelas. A unidade multiefeitos possui diversos potenciômetros e botões para ajustar os parâmetros dos efeitos aplicados sob o sinal.



Fonte: Próprio autor.

Equipamentos de propósito geral também podem ser usados como unidades de efeitos pela instalação de ferramentas de softwares para a execução de *plugins* de áudio.

Processador digital multiefeitos: Sinônimo de unidade multiefeitos digital. O termo é preferencialmente utilizado para definir unidades digitais implementadas como sistemas embarcados de propósito específico.

Digital Audio Effects (DAFX): Ferramentas de software em que os sinais de entrada recebidos são modificados de acordo com os parâmetros de controle da ferramenta e disponibilizados como sinais de saída (ZOELZER, 2011).

Plugin de áudio: Algoritmo para o processamento de sinais digitais implementado sobre uma API (*Application Programming Interface*) de áudio (SCHIAVONE; GOULART; QUEIROZ, 2012). A implementação como *plugins* permite o uso nos equipamentos com suporte a API. DAFX podem ser implementados como *plugins* de áudio.

Instância de plugin de áudio: “Objeto” de determinado *plugin* de áudio cujo comportamento e estado são definidos pelo *plugin* de áudio. As instâncias de um *plugin* compartilham o mesmo conjunto de parâmetros (atributos), embora sejam diferentes quanto ao seus valores (conteúdo desses atributos). Este termo é útil pelo fato de que é possível que um processador digital multiefeitos possa executar mais de uma instância de *plugin* de áudio.

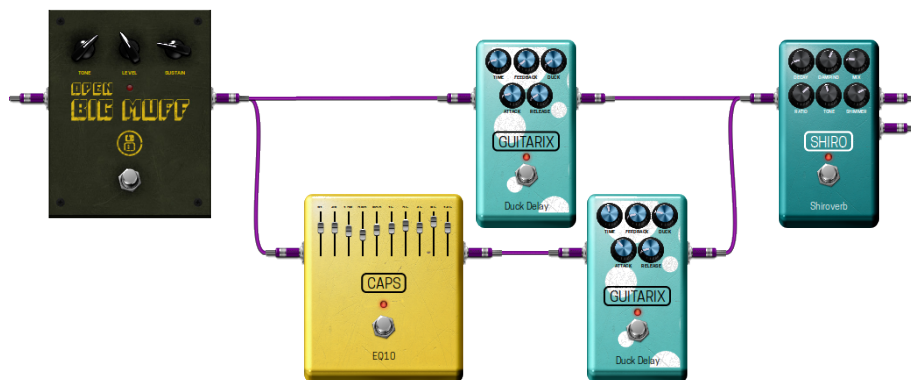
O termo instância associa-se com orientação a objetos: instância de *plugin* de áudio é para um *plugin* de áudio o que um objeto ou instância de uma classe é para uma classe.

Patch ou Pedalboard: Representação serializável (i.e., sujeita a persistência) de um conjunto de instâncias de *plugins* de áudio, os valores dos parâmetros das instâncias e

as conexões entre as *instâncias*. As conexões definem a sequência de transformações que serão aplicadas sobre o sinal de áudio.

Algumas unidades de efeitos digitais permitem o processamento paralelo de efeitos, de modo que não haja perda na “divisão” do sinal. É esperado que não haja qualquer ciclo na sequência de transformações. A Figura 5 apresenta uma representação visual de um *patch* em que *instâncias* são executadas em paralelo. A maioria dos processadores digitais multiefeitos não permitem um processamento paralelo, estando assim as instâncias dispostas em cadeia.

Figura 5 – Representação visual de um *patch* para a unidade multiefeitos MOD Duo. Cada caixa representa uma instância de *plugin* de áudio. A ordem na qual os efeitos são aplicados é dada pelas conexões entre as caixas. O sinal passa da esquerda para direita.



Fonte: Próprio autor.

2.3 Trabalhos relacionados

A busca de sistemas inteligentes para produção musical tem como objetivo propor ferramentas automáticas para facilitar e apoiar a tomada de decisão por atores do processo (REISS, 2011). No contexto de *plugins* de áudio, há três tarefas principais: *ajuste de parâmetros*, *recomendação de plugins* e *identificação automática de efeitos*.

O ajuste de parâmetros tem como intenção encontrar os valores de parâmetros mais adequados (por exemplo, *gain*, *treble level*) de um determinado *plugin*. Notavelmente, a maioria dos trabalhos na literatura aborda o problema de ajuste de parâmetros, incluindo abordagens para a escolha de valores padrão de parâmetros (*presets*) de acordo com o gênero musical (GOUDARD; MULLER, 2003; ROBILLARD, 2014); identificação automática de valores de parâmetros baseado em amostras de áudio (PETERS; CHOI; LEI, 2012); e controle adaptativo de efeitos de áudio digital – a-DAFx (do inglês *adaptive digital audio effects*), onde os ajustes dos parâmetros ocorrem com um controle adaptativo por extração de recurso de sinal de entrada e opcionalmente controle por gestos (VERFAILLE;

ZOLZER; ARFIB, 2006), seguindo estas transformações dinâmicas lineares (STARK; DAVIES; PLUMBLEY, 2008) ou abordagens não lineares (CHOURDAKIS; REISS, 2017; RAMÍREZ; REISS, 2018).

A recomendação de *plugins* lida com a tarefa de selecionar *plugins* a partir de um conjunto de opções disponíveis respeitando as restrições da unidade de efeitos. Apenas alguns autores abordaram a tarefa de recomendar *plugins* de áudio. Stasis et al. (2017) aplicou cadeias de Markov para modelar sequências de *plugins* com base em termos descritivos de timbres, efeitos de áudio desejados e gênero de música. Os autores coletaram um conjunto de dados com 178 formulários respondidos por produtores musicais, onde tiveram que definir uma cadeia de *plugins* para serem aplicados a amostras aleatórias de áudio da biblioteca de *Multitracks Mixing Secrets*¹. Apesar dos resultados interessantes, as recomendações vêm apenas de um conjunto limitado de quatro *plugins* diferentes. Além disso, os experimentos incluem um conjunto de dados anotado por especialistas, que nem sempre está prontamente disponível.

Relacionado ao tema, o compartilhamento de *patches* em repositórios de configuração fornece uma base de conhecimento de referência para os atores do processo de produção musical. *Patches* originais e baseados em artistas ou gêneros modelados pelas fabricantes de unidades de efeito e pela comunidade introduzem formas de uso de *plugins* de áudio para os usuários das unidades de efeito.

Pesquisas sobre identificação automática de efeitos de áudio em gravações de instrumentos, como guitarra e baixo, são pertencentes ao campo de recuperação da informação musical (MIR, do inglês *music information retrieval*) (STEIN et al., 2010). Conforme Schmitt e Schuller (2017) citam, ferramentas deste tipo possuem potencial na construção de conhecimento dos usuários e em possíveis aperfeiçoamentos em sistemas de recomendação como sistemas inteligentes de busca e recomendação de música por possíveis associações a certo humor ou gênero do sistema (SONG; DIXON; PEARCE, 2012).

Técnicas para identificação de efeitos de áudio em músicas são úteis no desenvolvimento do conhecimento do usuário sobre possíveis contextos de aplicação de efeitos. No entanto, a identificação dos efeitos aplicados pode não trazer informação suficiente para uma seleção automática de *plugins*, tendo em vista que, para uma unidade de efeito específica, podem haver diversos *plugins* que estariam na mesma categoria do efeito.

A seleção automática de *plugins* de áudio baseados em termos descritivos apresentada por Stasis et al. (2017) pode gerar saídas altamente correlatas com o conjunto de dados usado para treinar, mas a pesquisa não aborda casos de *plugins* distintos de uma mesma categoria de efeito ou mesmo bancos de dados contendo mais *plugins* de áudio e exemplos de uso. Além disso, o trabalho não explora a possibilidade de recomendar *plugins*

¹ A biblioteca pode ser acessada em <<http://www.cambridge-mt.com/ms-mtk.htm>>

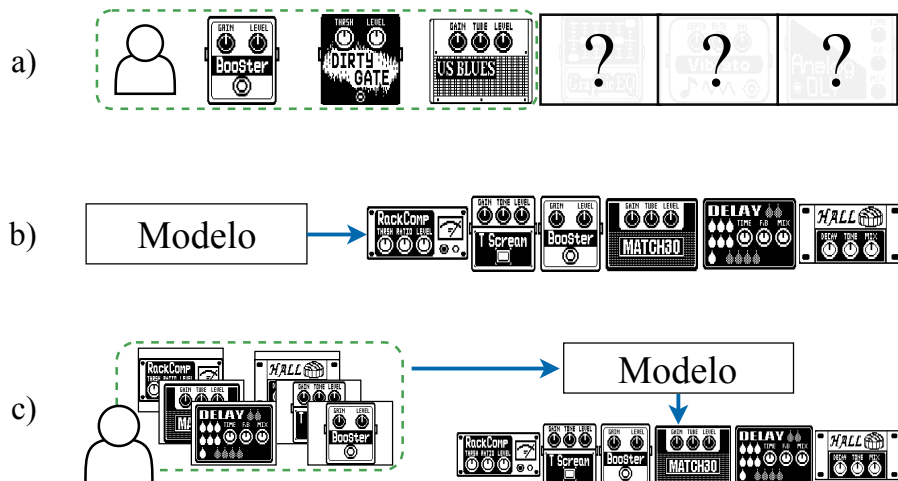
para *patches* incompletos. Sobre o ajuste de parâmetros, é necessário que os *plugins* de áudio já tenham sido selecionados.

2.4 Discussão

Dando continuidade a apresentação do problema (Capítulo 1), este capítulo trouxe maiores detalhes sobre a aplicação de *plugins* de áudio no contexto de produção musical. Inicialmente, foram apresentadas motivações para o emprego de ferramentas de manipulação de áudio (Seção 2.1). Em seguida, foram definidos termos relativos ao processamento de áudio no meio musical utilizados no decorrer do trabalho, enumerados tipos/categorias de efeitos de áudio e ainda formas de como unidades de efeito e *plugins* de áudio são utilizadas (Seção 2.2). Por fim, foram definidas três tarefas principais abordadas na literatura sobre o uso de *plugins*, discorrendo os estudos que tratam destas (Seção 2.3).

Conclui-se da Seção 2.3 que lacunas presentes nos trabalhos apresentados abrem espaço para a exploração de técnicas de sistemas de recomendação para apoio na seleção de *plugins* de áudio. Este trabalho abordará tal tarefa nos seguintes casos, ilustrados na Figura 6.

Figura 6 – Casos de recomendação de *plugins* de áudio: a) usuário pré-seleciona alguns *plugins* de áudio e deseja que um sistema de recomendação auxilie na escolha dos *plugins* para outras posições; b) modelo sugere todos os *plugins* de um *patch*, e; c) usuário seleciona todos os *plugins* de áudio e o modelo define automaticamente a ordem destes no *patch*. Representações gráficas dos *plugins* extraídas de ZOOM (2012).



Fonte: Próprio autor.

- a) recomendação de parte de um *patch*, onde a recomendação de *plugins* para uma ou mais posições deve se basear em outros *plugins* previamente selecionados pelo usuário do sistema de recomendação;

- b) geração de um *patch* completo, onde um modelo recomenda todos os *plugins* de uma configuração, e;
- c) definição automática da ordem dos *plugins* na cadeia de efeitos.

O Capítulo 3 formulará formalmente o problema de recomendação de *plugins* de áudio e apresentará duas abordagens para seu tratamento: classificação e filtragem colaborativa. Enquanto a primeira define um meio para aplicação de classificadores multiclasse de aprendizagem de máquina, a segunda abre espaço para a aplicação de modelos generativos. Dentre motivações para o uso de modelos generativos, destacam-se:

- a geração de amostras a partir da distribuição aprendida, possibilitando, por exemplo, treinamento de modelos distintos para gêneros específicos;
- o cálculo da probabilidade de um padrão pertencer a distribuição aprendida, útil para identificação e categorização de configurações em gêneros musicais, e;
- encontrar uma representação latente para os padrões, i.e., uma projeção para os dados. No novo espaço representado os dados podem, por exemplo, estarem comprimidos ou apresentarem-se mais adequados para tarefas de classificação e de clusterização posteriores.

3 Recomendação automática de *plugins* de áudio

O problema de recomendação de *plugins* de áudio consiste na sugestão de *plugins* de áudio para a utilização em um *contexto* a partir de *informações pré-estabelecidas*. Nós propomos dividir o conceito de contexto em duas partes *i.* finalidade e *ii.* ferramenta.

A finalidade ou o *motivo fim* considera tanto questões objetivas quanto subjetivas, envolvendo sentimentos — buscando suscitar emoções —, a transmissão de ideias musicais e questões técnicas — a harmonia com os equipamentos utilizados em conjunto (instrumentos musicais, amplificadores), o gênero musical abordado.

A ferramenta refere-se ao equipamento no qual os *plugins* de áudio serão utilizados: sistema de propósito geral com programa e tecnologia definidos ou sistema embarcado de propósito específico. A ferramenta definirá o conjunto dos possíveis *plugins* de áudio disponíveis para se trabalhar e a forma como estes podem ser utilizados, incluindo restrições envolvendo a utilização e a capacidade computacional de processamento. A *finalidade* é apresentada para a *ferramenta* como um *patch*, onde a ideia do utilizador é representada pelos *plugins* e seus parâmetros definidos.

O problema de recomendação encontra-se com o universo de produção musical quando a grande quantidade de variáveis envolvendo a escolha de *plugins* de áudio torna-se um fator de ineficiência que atinge tanto usuários amadores quanto especialistas. Nesse contexto, os *patches* compartilhados pelos modeladores nos repositórios fornecem um conhecimento que pode ser absorvido tanto por membros da comunidade quanto por modelos de sistemas de recomendação.

Entretanto, a qualidade da descrição da finalidade para qual os *patches* compartilhados foram concebidos é questionável. Neste aspecto, sistemas de recomendação se apresentam como instrumentos úteis no processo de modelagem de *patches* por possuírem o potencial de capturar relações entre os dados pela análise de uma grande quantidade de configurações.

Desta forma, sendo mais específico, o problema de recomendação de *plugins* de áudio consiste na recomendação de *plugins* adequados para um *patch* pré-configurado dentre um conjunto finito e discreto de possíveis *plugins* a partir do próprio *patch* e do conhecimento adquirido pelo modelo a partir de um processo de aprendizagem de máquina sobre uma base de dados ou de heurísticas previamente definidas.

A seguir, é apresentada a definição formal modelada para o problema de recomendação. Nas seções posteriores, são delineadas as estratégias de recomendação por

classificação e por filtragem colaborativa. Detalhes sobre os modelos utilizados nos experimentos encontram-se no Capítulo 4 e no Capítulo 5, respectivamente, de classificação e de filtragem colaborativa.

3.1 Definição formal

Representemos formalmente o conjunto dos *plugins* de áudios disponíveis para um processador multiefeitos como $\mathcal{S} = \{1, 2, \dots, R\}$, onde R é a quantidade de *plugins* do equipamento. Um *patch* $\mathbf{v}$ consiste em uma coleção de M instâncias de *plugins* de áudio, i.e., $\mathbf{v} = [v_1, v_2, \dots, v_M]^T$ com $v_i \in \mathcal{S}$. Usualmente M , o tamanho do *patch*, é muito menor que o número total de *plugins* disponíveis, i.e., $M \ll R$. Estamos interessados no problema de recomendação de *plugins* de áudio para as entradas ausentes de *patches* incompletos.

Por uma questão de clareza, considere um processador multiefeitos com 5 *plugins* de áudio disponíveis, nomeadamente: $\mathcal{S} = \{\text{Delay}, \text{Reverb}, \text{Overdrive}, \text{Chorus}, \text{Equalizer}\}$. Sem perda de generalidade, assumimos um dicionário que mapeia os nomes dos *plugins* aos seus identificadores únicos (ids) em $\mathcal{S}$. Um *patch* de tamanho $M = 2$ é qualquer combinação (com repetição) de dois *plugins*, como por exemplo $\mathbf{v} = [\text{Chorus}, \text{Overdrive}]^T$ ou $\mathbf{v} = [\text{Delay}, \text{Delay}]^T$. Em cenários reais, em que um usuário possui pouca ou nenhuma indicação sobre qual *plugin* adicionar ao *patch*, um sistema de recomendação pode sugerir as configurações mais prováveis condicionadas aos *plugins* pré-selecionados.

Em uma perspectiva probabilística, cada *patch* $\mathbf{v}$ corresponde a uma variável aleatória discreta M -dimensional. Nosso objetivo é maximizar $P(\mathbf{v}_{\text{unknown}} | \mathbf{v}_{\text{visible}})$, onde $\mathbf{v}_{\text{unknown}} \subseteq \mathbf{v}$ denota os *plugins* desconhecidos (ausentes), e $\mathbf{v}_{\text{visible}}$ representa os *plugins* pré-selecionados no *patch*. Claramente, $\mathbf{v}_{\text{unknown}}$ e $\mathbf{v}_{\text{visible}}$ são disjuntos. Neste trabalho, consideramos inicialmente o caso mais simples, no qual $\mathbf{v}_{\text{unknown}}$ contém uma única variável aleatória v_h , onde h denota um índice no *patch* $\mathbf{v}$.

3.2 Recomendação como problema de classificação

É possível tratar tarefas de recomendação como problemas de classificação (BASU; HIRSH; COHEN, 1998). Seguindo uma estratégia de classificação multiclasse, uma probabilidade pode ser atribuída para cada *plugin*/candidato a determinada posição em um *patch*, de modo que corresponda à expectativa de um usuário selecioná-los (ZHANG; IYENGAR, 2002).

Ao lidar com dados discretos, a representação assume um papel importante. Ids de *plugin* (inteiros $1, \dots, R$) transmitem uma falsa noção de similaridade que fortemente depende da atribuição do índice inicial. Por exemplo, índices sucessivos não devem implicar proximidade: não há razão para acreditar que um i -ésimo *plugin* é mais similar ao $(i + 1)$ -

ésimo *plugin* que qualquer outro *plugin* em $\mathcal{S}$. Nesta dissertação, consideramos duas representações vetoriais discretas para os dados. Ressaltamos que a escolha da representação pode aumentar a dimensão de entrada (vetores *one-hot*) ou perder informações sobre o posicionamento dos *plugins* de áudio no *patch* (*bag-of-words*). Para detalhes sobre as representações, recomendamos a leitura de Johnson e Zhang (2015).

Vale a pena notar que a distribuição dos *plugins* sobre *patches* não é invariante de permutação. Como consequência, sistemas de recomendação baseados em classificação requerem o treinamento de vários modelos, um para cada posição a ser recomendada no *patch*.

3.3 Filtragem colaborativa

O conceito de filtragem colaborativa refere-se aos métodos de realizar predições automáticas (filtragem) sobre os interesses de um usuário com base nas preferências de usuários com perfil similar (colaborativa) (GOLDBERG et al., 1992; HU; KOREN; VOLINSKY, 2008). No cenário padrão, são usadas as avaliações dos usuários sobre os itens (filtragem colaborativa orientada a usuário). Os itens, filmes ou fotos, por exemplo, são geralmente avaliados por *like/dislike*¹ (avaliação binária) ou por um número de estrelas².

Exemplificando em um contexto de avaliações de filmes, se determinado usuário costuma avaliar bem filmes de categorias como terror e romance, mas avalia mal filmes de ação, o modelo deve recomendá-lo obras que ainda não assistiu buscando prever quais notas daria a elas baseando-se nas avaliações de usuários com gosto parecido. Informações sobre o gênero das obras cinematográficas ou sobre o usuário podem também ser utilizadas no processo de predição.

As seguinte associações podem ser feitas para o contexto de recomendação de *plugins* de áudio: *patches* correspondem aos usuários, posições/*slots* no *patch* referem-se a filmes/itens e *plugins* representam as avaliações.

3.3.1 k -vizinhos mais próximos

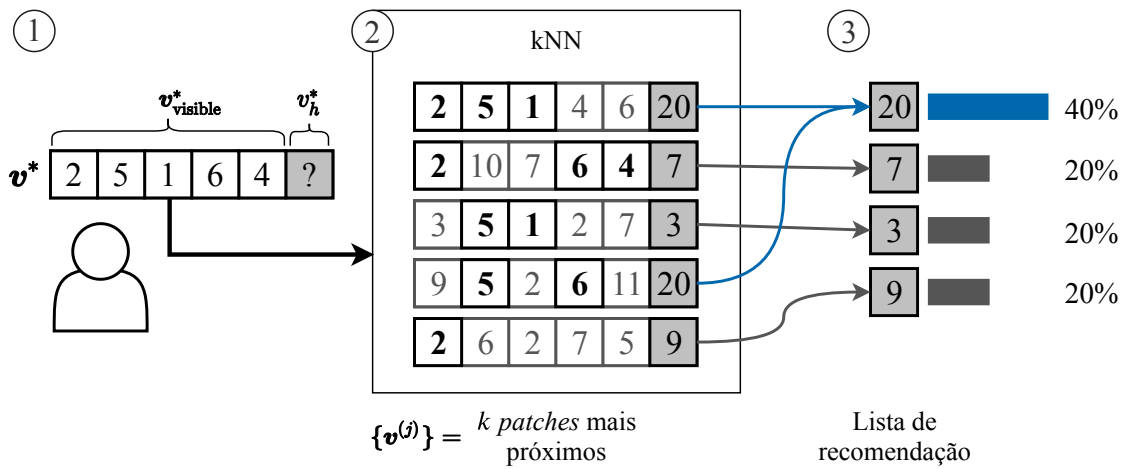
Um modelo k -vizinhos mais próximos (kNN, do inglês *k-nearest neighbors*) recomenda *plugins* seguindo uma abordagem *frequentista* (KOREN; BELL, 2015). Suponhamos que desejamos recomendar o *plugin* associado à variável aleatória desconhecida v_h^* do *patch* de consulta $\mathbf{v}^*$ (Figura 7 – Passo 1). A primeira etapa consiste em procurar os k *patches* mais próximos $\{\mathbf{v}^{(j)}\}$ ao $\mathbf{v}^*$ de acordo com uma métrica de similaridade, como a distância Hamming (ZHANG et al., 2010) (Figura 7 – Passo 2). Durante o cálculo

¹ Um usuário *gosta/não gosta* de um item.

² A quantidade de estrelas define o quanto o usuário gostou do item.

das similaridades, somente os componentes visíveis dos *patches* em sua forma vetorial, $\{v_{\text{visible}}^{(j)}\}$ e v_{visible}^* , devem ser usados. A recomendação final é então amostrada a partir da distribuição empírica calculada sobre o h -ésimo componente dos k vizinhos mais próximos (Figura 7 – Passo 3).

Figura 7 – Processo de recomendação pelo modelo kNN (com $k = 5$) de *plugins* para a posição correspondente à variável aleatória v_h^* de um *patch* de consulta v^* . Números correspondem aos ids dos *plugins* de áudio. **Passo 1.** Um *patch* v^* é modelado por um usuário. **Passo 2.** O modelo kNN recebe-o e lista $\{v_{\text{visible}}^{(j)}\}$, os $k = 5$ *patches* mais próximos de v^* . **Passo 3.** O modelo informa então uma lista de recomendações pela distribuição empírica h -ésimo componente dos k vizinhos mais próximos.



Fonte: Próprio autor.

3.3.2 Máquinas de Boltzmann Restritas

A Máquina de Boltzmann Restrita (RBM, do inglês *Restricted Boltzmann Machine*) e sua especialização para filtragem colaborativa (RBM-CF, do inglês *Restricted Boltzmann Machine for Collaborative Filtering*) são modelos generativos capazes de aprender a distribuição probabilística dos dados utilizados para treino (SMOLENSKY, 1986; SALAKHUTDINOV; MNIH; HINTON, 2007).

Modelos generativos são interessantes no emprego em problemas de recomendação dada a capacidade de codificar distribuições de probabilidade de modo que seja possível reconstruir os dados observados ou gerar novas amostras (SCHEIN et al., 2001). O cálculo da log-probabilidade de padrões pertencerem à distribuição aprendida pode ser útil para identificação de padrões que pertençam ou não a mesma distribuição do conjunto de treino.

Modelos RBM binários são compostos de variáveis aleatórias visíveis e latentes (ocultas), capazes de assumir os valores binários $\{0, 1\}$. O processo de aprendizagem é não supervisionado e busca ajustar os parâmetros do modelo para que sinalize uma maior

probabilidade conjunta para os dados pertencentes à distribuição do conjunto. Este modelo será apresentado com maior profundidade no Capítulo 5.

Na metodologia aplicada, Máquinas de Boltzmann Restrita recomendam *plugins* para *patches* incompletos pela reconstrução do *patch* a partir da distribuição aprendida pelo modelo. É válido pontuar que a reconstrução de um *patch* permite a geração de listas de recomendação para mais de uma posição simultaneamente a partir da distribuição dos dados aprendida por uma única RBM.

3.3.3 Comentários sobre filtragem colaborativa de *plugins* de áudio

A tarefa de recomendação de *plugins* de áudio impõe alguns desafios adicionais em comparação com o problema padrão de filtragem colaborativa. Primeiro, não há análise de sentimento: enquanto a avaliação de um item nos problemas clássicos de recomendação por filtragem colaborativa corresponde a uma quantificação de gosto ou desgosto, um *plugin* de áudio não presente em um *patch* não indica necessariamente que esse não é adequado. De fato, no contexto musical, existem casos de *plugins* similares cujo uso depende do perfil do usuário, enquanto existem *plugins* que realmente não combinam juntos.

Em segundo lugar, enquanto em problemas comumente tratados por filtragem colaborativa é desejável saber apenas se determinado item é adequado, no caso tratado neste trabalho, é desejável saber se determinado item é adequado para uma determinada posição. Além disso, o problema também permite a repetição de itens durante a recomendação em diferentes posições. Por exemplo, para recomendação de filmes, não costuma fazer sentido responder “qual filme assistir após **Os Incríveis** e antes de **Procurando Nemo**?”³ ou sugerir que um usuário assista repetidas vezes um mesmo filme.

Ressaltamos que a abordagem de classificação faz mapeamento de entrada-saída, enquanto um modelo generativo, como a RBM-CF, aproxima a distribuição de probabilidade dos dados e calcula probabilidades condicionais para realizar as recomendações.

3.4 Discussão

Ferramentas para recomendações de *plugins* de áudio para suporte na tomada de decisão devem considerar informações preestabelecidas por seus usuários e respeitar restrições da unidade de efeito para o qual foram modeladas. Como comentado na Seção 3.1, as recomendações para uma configuração $\mathbf{v}$ podem se feitas condicionadas aos *plugins* pré-selecionados $\mathbf{v}_{\text{visible}}$ desta configuração.

Foi comentado que modelos podem ou não considerar o posicionamento dos *plugins* na cadeia de efeitos, resultando no aumento da dimensão de entrada ou no descarte

³ Exceto casos de maratona de filmes, mas a intenção já é explícita.

desta informação. Como discutido na Seção 3.2, ambas as abordagens serão testadas com classificadores multiclasse no decorrer do trabalho. Ainda sobre classificadores, foi destacado que recomendações para posições distintas necessitam de classificadores distintos.

A representação probabilística das M posições de instâncias em um *patch* como variáveis aleatórias trouxe uma identificação do problema com modelos de aprendizagem de máquina com arcabouço probabilístico. Neste sentido, os modelos generativos RBM e RBM-CF apresentam-se como modelos aplicáveis ao problema sobre uma perspectiva de filtragem colaborativa (Subseção 3.3.2). Para filtragem colaborativa, também foi apresentado o modelo k -vizinhos mais próximos, inclusive informando como este pode ser usado para a recomendação de *plugins* para um *patch* incompleto.

A seguir, o Capítulo 4 continuará a explicação sobre como classificadores podem ser usados na tarefa de recomendação de *plugins* de áudio. Ainda, são apresentados os classificadores escolhidos para a realização dos experimentos. Após, no Capítulo 5 os modelos RBM e RBM-CF são aprofundados.

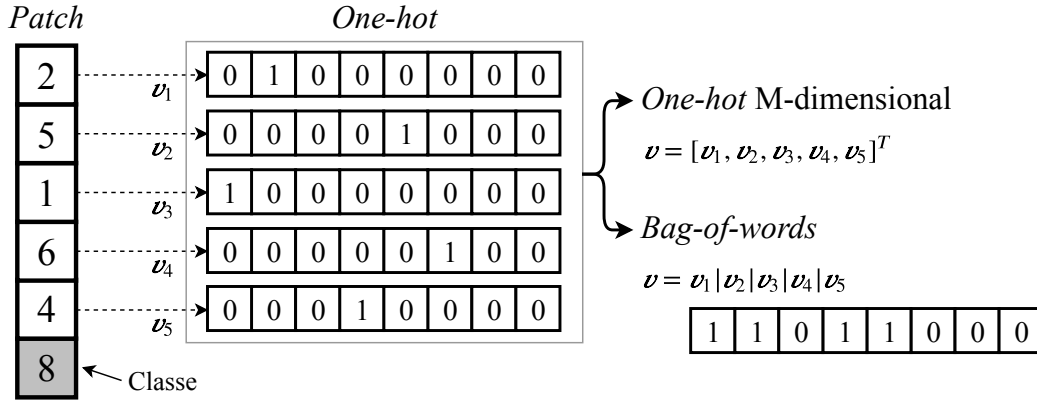
4 Classificadores para recomendação de *plugins* de áudio

Na estratégia de recomendação de *plugins* de áudio usando uma abordagem de classificação, a recomendação de um *plugin* para determinada posição em um *patch* dado os outros *plugins* pré-selecionados pelo usuário se dá pela probabilidade atribuída por um classificador multiclasse para cada *plugin*/candidato para a posição.

Os passos para o emprego desta abordagem estão sintetizados abaixo. Note que a metodologia para a recomendação de *plugins* segue o diagrama apresentado na Figura 1.

1. Delimitar o problema:
 - a) definir uma posição i no *patch* na qual deseja-se recomendar um *plugin*;
 - b) definir $\mathcal{J}$ como o conjunto das posições dos *plugins* pré-definidos pelo usuário;
2. Preparar a base de dados:
 - a) transformar o i -ésimo atributo dos padrões em *classe*. A quantidade de classes corresponde ao total de *plugins* que a unidade de efeito suporta;
 - b) selecionar os atributos correspondentes às posições em $\mathcal{J}$ como entrada do modelo;
 - c) transformar os atributos de entrada para uma codificação adequada ao modelo e ao problema, conforme discutido na Seção 3.2. Codificações testadas nos experimentos foram o *bag-of-words* e o *one-hot* $(M - 1)$ -dimensional;
3. Treinar o modelo, abrangendo também o processo de escolha dos hiperparâmetros, e;
4. Utilizar o modelo:
 - a) obter os atributos de entrada correspondentes às posições em $\mathcal{J}$ do *patch* incompleto;
 - b) transformar estes atributos conforme definido no passo 2.c) e inserindo-os na entrada do modelo treinado;
 - c) realizar a predição das probabilidades para cada classe de um padrão, i.e., a probabilidade de cada possível *plugin* ser aplicado na i -ésima posição do *patch* incompleto;
 - d) criar a lista de recomendações de *plugins* pela ordenação decrescente das probabilidades.

Figura 8 – Tratamento dos dados para classificação em *patches* com seis posições e oito possíveis *plugins*. Da esquerda para direita, *i*) um atributo é definido como classe, *ii*) os outros atributos são codificados individualmente como vetores *one-hot* e *iii*) combinados: concatenados em um vetor $(M - 1)$ -dimensional ou codificados em *bag-of-words* pela operação binária *or*.



Fonte: Próprio autor.

A Figura 8 destaca a parte de preparação dos dados para uma unidade de efeitos com $R = 8$ possíveis *plugins* e $M = 6$ instâncias por *patch*. O elemento em destaque corresponde ao atributo considerado como classe, isto é, a posição que o modelo deverá aprender em prever, no caso, a sexta posição. Note que é necessário treinar um novo modelo para prever *plugins* para outras posições, tendo em vista que a distribuição dos *plugins* sobre *patches* não é invariante a permutação, conforme será mostrado na Subseção 6.1.1, Figura 16.

A obtenção de uma lista de probabilidades depende da estratégia tomada pelo classificador para classificar o padrão. Por exemplo, classificadores que utilizam a função *softmax* na definição da classe de um padrão, a própria saída da função pode ser utilizada. Modelos de aprendizagem com outras estratégias para definição de uma probabilidade para as classes podem ser aplicados desde que o custo assintótico para definir as probabilidades de todas as classes não seja proibitivo.

Nesta dissertação, os seguintes classificadores foram utilizados: Regressão Logística, Perceptron Multicamadas e Máquinas de Vetores Suporte. Esses são apresentados nas seções seguintes.

4.1 Regressão Logística

Logistic Regression (LR) ou Regressão Logística é um classificador binário linear (BERKSON, 1944). A ideia principal deste modelo consiste na associação de um peso para

cada um dos D atributos de um i -ésimo padrão qualquer $\mathbf{x}_i \in \mathbb{R}^D$ do conjunto de dados, de modo que a aplicação da função sigmoide ($\sigma(\cdot)$) sobre o produto escalar do vetor de pesos $\mathbf{w} \in \mathbb{R}^D$ e de um padrão $\mathbf{x}_i$ estime a probabilidade de y_i , a classe correspondente ao padrão, ser 1:

$$\hat{y}_i = P(y_i = 1 | \mathbf{x}_i) = \sigma(\mathbf{w}^T \mathbf{x}_i) = \frac{1}{1 + \exp(\mathbf{w}^T \mathbf{x}_i)}.$$

A definição dos valores dos pesos no processo de treinamento consiste em ajustar as ponderações de modo que $\hat{y}_i$, a probabilidade indicada pelo modelo, aproxime-se da classe binária real y_i do padrão. O treinamento pode ocorrer analiticamente, com o algoritmo de mínimos quadrados ordinário (OLS, do inglês *ordinary least squares*), ou por um processo iterativo, com uso de algoritmos como Gradiente Descendente (GD, do inglês *Gradient Descent*) ou Gradiente Descendente Estocástico (LMS, do inglês *least mean squares*).

Uma classificação multiclasse para K classes pode ser feita com classificadores LR binários por estratégias um-contra-todos com $K - 1$ classificadores binários ou um-contra-um com $K \frac{K-1}{2}$. Já a extensão de LR *regressão logística multinomial* permite uma classificação multiclasse com um único classificador. Neste, no lugar de um vetor, é utilizada uma matriz de pesos $\mathbf{W} \in \mathbb{R}^{D \times K}$ para as K classes e, para a saída do modelo, a função *softmax* é aplicada. Assim, a probabilidade de um i -ésimo padrão possuir sua classe (y_i) como sendo o padrão k é:

$$P(y_i = k | \mathbf{x}_i) = \text{softmax}(\mathbf{x}_i^T \mathbf{W}_{\cdot k}) = \frac{\exp(\mathbf{x}_i^T \mathbf{W}_{\cdot k})}{\sum_{j=1}^K \exp(\mathbf{x}_i^T \mathbf{W}_{\cdot j})}.$$

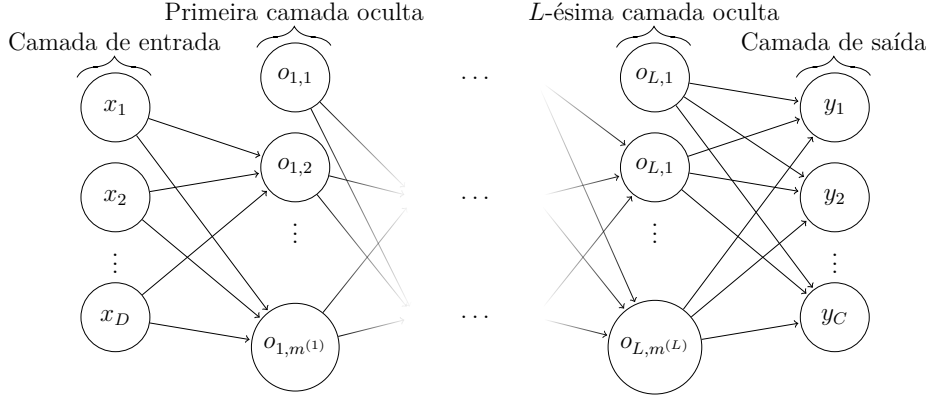
A notação $\mathbf{W}_{\cdot k}$ corresponde a k -ésima coluna da matriz $\mathbf{W}$, ou seja o vetor coluna dos pesos associados aos atributos e à classe k .

4.2 Perceptron Multicamadas

O classificador Perceptron Multicamadas ou, em inglês, *Multi Layer Perceptron* (MLP), é um modelo capaz de resolver problemas não lineares e pertence ao grupo das Redes Neurais Artificiais, algoritmos de aprendizagem de máquina bioinspirados nas sinapses de neurônios (HAYKIN, 2009). Uma rede MLP contém camadas onde cada camada é composta por unidades ou neurônios. A primeira camada é denominada camada de entrada e correspondente a entrada dos dados no algoritmo. Em seguida, uma ou mais camadas ocultas, com unidades que não possuem acesso direto à saída. A última camada é a de saída do modelo.

A Figura 9 ilustra uma MLP com L camadas ocultas. Atente-se que unidades de uma camada possuem acesso a todos as unidades de uma camada anterior e passa informação para todos as unidades de uma camada seguinte, exceto a camada de entrada e a de saída. Observe também que as camadas ocultas podem possuir quantidades diferentes de unidades.

Figura 9 – Representação gráfica de uma MLP com D unidades de entrada, L camadas ocultas e C unidades de saída.



Fonte: Adaptado de STUTZ, 2016.

A ideia do classificador MLP é que uma i -ésima unidade de uma l -ésima camada receba informação proveniente da camada anterior. Esta informação é dada por uma função de ativação $\phi_l(\cdot)$ associada a camada l , na qual considera os valores das unidades da camada anterior e $\mathbf{w}_{l,i}$, o vetor de ponderações entre a unidade i da camada l e cada uma das unidades da camada $l - 1$:

$$\begin{aligned} o_{1,i} &= \phi_1(\mathbf{w}_{1,i}^T \mathbf{x}), \quad \text{para } l = 1, \text{ e} \\ o_{l,i} &= \phi_l(\mathbf{w}_{l,i}^T \mathbf{o}_l) \quad \text{para } l > 1, \end{aligned}$$

onde $\mathbf{x} \in \mathbb{R}^{D+1}$ é um vetor composto dos D atributos e do viés.

Dentre diversas funções de ativação aplicadas, as mais conhecidas são a *Rectified Linear Unit* ($\text{relu}(z) = \max(0, z)$), a sigmoide e a tangente hiperbólica ($\tanh(z) = \frac{\exp(2z)-1}{\exp(2z)+1}$). Para uma tarefa multiclasse, a função *softmax* é aplicada na camada de saída para estimar a probabilidade de cada i -ésima classe:

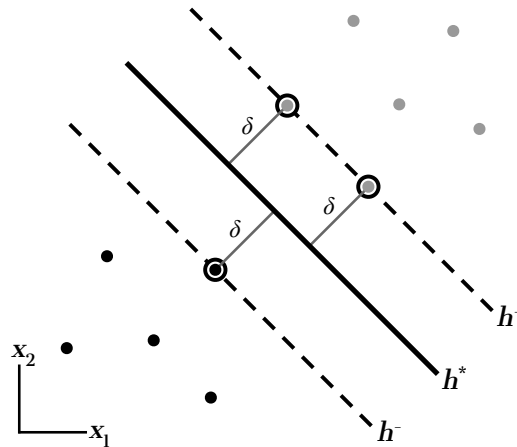
$$y_i = \phi_L(\mathbf{w}_{L,i}^T \mathbf{o}_{L,i}) = \text{softmax}(\mathbf{w}_{L,i}^T \mathbf{w}_{L,i}).$$

O algoritmo de treinamento mais conhecido para uma MLP é o *backpropagation* (retropropagação) (RUMELHART; HINTON; WILLIAMS, 1987). Este é dividido em duas partes: sentido direto (*forward*) da MLP e sentido reverso (*backward*) da MLP. No primeiro, a informação ocorre da entrada para a saída de acordo com as funções de ativação definidas para cada neurônio. No segundo, *i*) gradientes da função custo na camada de saída são calculados, e, recursivamente, *ii*) os erros são retropropagados para a camada anterior (diferenciação por regra da cadeia) e *iii*) todos os pesos são atualizados via gradiente descendente estocástico.

4.3 Máquina de Vetores Suporte

Máquina de Vetores Suporte (SVM, do inglês *Support Vector Machine*) é um modelo de aprendizagem de máquina cujo objetivo é encontrar h^* , uma fronteira (hiperplano de separação) entre as classes que maximize δ , a *margem* de separação entre as classes (BOSER; GUYON; VAPNIK, 1992). Uma fronteira é definida pelos padrões próximos a ela, chamados de *vetores suporte*. A Figura 10 apresenta um problema de classificação de padrões em um espaço bidimensional. A fronteira indicada é ótima, maximizando a margem δ entre os vetores suporte circulados.

Figura 10 – Problema de classificação binária em duas dimensões. Padrões de mesma classe com mesma cor. Hiperplano de separação ótimo h^* encontrado por um classificador SVM. Vetores suportes circutados.



Fonte: DIAS, 2017.

O hiperplano de separação ótimo h^* é obtido a partir da solução de um problema de otimização com restrições, sendo resolvido em sua forma dual, após a inclusão de multiplicadores de Lagrange, por conveniência matemática. Embora o treinamento envolva uma otimização não linear, a função objetivo é convexa e, portanto, uma solução local é um ótimo global (BISHOP, 2006).

Para uma classificação não linear, é possível definir transformações não lineares $\phi(\cdot)$ de modo que os padrões transformados sejam linearmente separados no novo espaço, denominado *espaço de atributos*. Para evitar um maior custo computacional decorrente da geração de espaços com alta dimensionalidade e com maior quantidade de parâmetros, Máquinas de Vetores Suporte utilizam do artifício denominado *truque do kernel* ou *substituição de kernel*, em que produtos internos no espaço de atributos são substituídos por uma função de kernel. Alguns dos mais conhecidos são o linear ($k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$), o polinomial ($k(\mathbf{x}_i, \mathbf{x}_j) = (Q + \mathbf{x}_i^T \mathbf{x}_j)^P$) e o Gaussiano ou RBF, de *Radial Basis Function* ($k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|)$). SVM é um classificador binário dado a separação

por hiperplano. Entretanto é possível utilizar modelos SVM para uma tarefa multiclasse com estratégias um-contrum ou um-contratodos. Para uma maior discussão sobre estratégias para classificação multiclasse, consulte Bishop (2006, Subseção 7.1.3). Para uma interpretação probabilística do modelo, consulte Murphy (2012, Subseção 14.5.5). Para a geração da lista de recomendação, a biblioteca da SVM utilizada neste trabalho aplica um processo interno de validação cruzada para construir um modelo probabilístico (PEDREGOSA et al., 2011; CHANG; LIN, 2011).

4.4 Discussão

Neste capítulo foi definida uma estratégia para que classificadores possam ser usados na tarefa de recomendação de *plugins* de áudio em *patches* incompletos. Nela, um *patch* corresponde a um padrão cujo os atributos de entrada do modelo são os *plugins* de áudio pré-selecionados pelo usuário e a classe é o *plugin* da posição na qual há interesse na recomendação de *plugins*.

Como visto, para que um classificador seja válido para a tarefa, é necessário que seja computacionalmente viável este definir uma probabilidade para cada classe, já que a lista de recomendação é ordenada pela probabilidade de seus elementos. Também foi comentado que a estratégia multiclasse definida implica que é necessário o treinamento de modelos distintos para cada posição no *patch* em que é desejada uma recomendação, pois a forma como os *plugins* são utilizados varia conforme sua posição no *patch*.

Ainda, foram apresentados os classificadores que serão avaliados no decorrer do trabalho: o classificador linear Regressão Logística, e os classificadores não lineares Perceptron Multicamadas e Máquinas de Vetores Suporte.

Enquanto este capítulo teve foco em modelos discriminativos, em que existe a busca de um mapeamento “padrão-classe”, $P(y|\mathbf{x})$, o Capítulo 5 tratará o problema com uma abordagem generativa, onde os modelos aplicados são inicialmente treinados para aprenderem a distribuição conjunta dos dados $P(y, \mathbf{x})$ para depois recomendar os *plugins* para um padrão, $P(y|\mathbf{x})$.

5 Máquinas de Boltzmann Restritas

Neste capítulo, são apresentadas as Máquinas de Boltzmann Restrita e Restrita para Filtragem Colaborativa. Na Seção 5.1 o modelo Máquina de Boltzmann Restrita é apresentado. Em suas subseções, são detalhados a representação do modelo (Subseção 5.1.1) e o processo de aprendizagem (Subseções 5.1.4 e 5.1.3). No fim, há uma discussão sobre pontos relevantes relacionados (Subseção 5.1.5).

A Seção 5.2 apresenta a extensão de Máquina de Boltzmann Restrita para sistemas recomendação por filtragem colaborativa, abordando a representação (Subseção 5.2.1), o processo de predição (Subseção 5.2.2) e uma discussão em forma de revisão bibliográfica de trabalhos que continuaram a pesquisa da aplicação da Máquina de Boltzmann Restrita em sistemas de recomendação (Subseção 5.2.3).

Ainda sobre este capítulo, apêndices foram preparados como material complementar ao leitor. Sendo a Máquina de Boltzmann Restrita uma especialização de uma Máquina de Boltzmann, há uma apresentação deste modelo no Apêndice B. Por sua vez, Máquina de Boltzmann e suas versões Restrita e Restrita para Filtragem Colaborativa são Redes de Markov: o Apêndice A traz formulações referentes à Redes de Markov que são base para algumas das equações desenvolvidas neste capítulo. Por fim, o Apêndice C contém o desenvolvimento das fórmulas deste capítulo.

5.1 Restricted Boltzmann Machine

Harmonium, *Restricted Boltzmann Machine* (RBM) ou Máquina de Boltzmann Restrita é um Modelo Gráfico Probabilístico baseado na função de energia de ‘Boltzmann’ (SMOLENSKY, 1986; FREUND; HAUSSLER, 1992). Uma RBM é uma especialização da Máquina de Boltzmann (Apêndice B) em que existem as seguintes restrições: há uma única camada de elementos visíveis, uma única camada de elementos ocultos e não existem conexões entre os nós de uma mesma camada (HINTON, 2002).

Diferentemente da Máquina de Boltzmann, as restrições impostas ao modelo adicionam tratabilidade no cálculo das probabilidades condicionais entre uma variável aleatória de uma camada e as variáveis aleatórias da outra camada. Será visto adiante que algoritmos de aprendizagem não supervisionada aplicados ao modelo aproveitam dessa característica para contornar a intratabilidade do cálculo da esperança da distribuição aprendida pelo modelo ($\mathbb{E}_{model}[\cdot]$), necessária no processo de otimização (HINTON, 2002; TIELEMAN, 2008). Mesmo com essas restrições, foi provado que esse modelo probabilístico possui ainda a capacidade de aprender quaisquer distribuições binárias (ROUX; BENGIO,

2008).

O modelo tem despertado bastante interesse de cientistas de aprendizagem de máquina devido as suas propriedades. Sua capacidade gerativa permite a amostragem de elementos a partir da distribuição aprendida (HINTON, 2002; BREULEUX; BENGIO; VINCENT, 2011). Por ser um bom extrator de características, é utilizado no processo de pré-treino em algoritmos de aprendizagem (HINTON, 2007). Possui capacidade para classificação de elementos (LAROCHELLE; BENGIO, 2008; NGUYEN et al., 2017), permitindo, por exemplo o uso em sistemas de recomendação (SALAKHUTDINOV; MNIH; HINTON, 2007). Ainda, variações profundas de RBM (*deep learning*) tratam de problemas como redução de dimensionalidade (HINTON; SALAKHUTDINOV, 2006) e formas (*shapes*) de elementos em conjunto de dados com poucas amostras (ESLAMI et al., 2014).

A RBM foi inicialmente definida para trabalhar com dados binários, i.e., os elementos das camadas visíveis e ocultas podem assumir somente os valores 0 ou 1. Na literatura existem variações para tratar com outros tipos de valores, como a *Gaussian-Bernoulli* RBM (GB-RBM), onde os elementos da camada visível podem assumir valores reais (HINTON; SALAKHUTDINOV, 2006). Nessa seção, entretanto, será discutida somente a versão binária, também chamada de *Bernoulli-Bernoulli* RBM (BB-RBM).

5.1.1 Representação de uma RBM

Uma Máquina de Boltzmann Restrita é composta de duas camadas: visível $\mathbf{v}$ e oculta $\mathbf{h}$. Respectivamente, cada camada possui $D, K \in \mathbb{N}^*$ elementos e cada um dos elementos pode assumir valores binários, i.e., $\mathbf{v} \in \mathcal{V}$ e $\mathbf{h} \in \mathcal{H}$, onde $\mathcal{V} = \{0, 1\}^D$ — conjunto de todas as possíveis combinações de valores para $\mathbf{v}$ — e $\mathcal{H} = \{0, 1\}^K$ — conjunto de todas as possíveis combinações de valores para $\mathbf{h}$.

RBM é um grafo bipartido no qual unidades de uma mesma camada não são conectadas entre si, porém uma unidade possui arestas para todas as outras unidades da outra camada (Figura 11).

A função massa de probabilidade conjunta para as variáveis aleatórias do modelo é

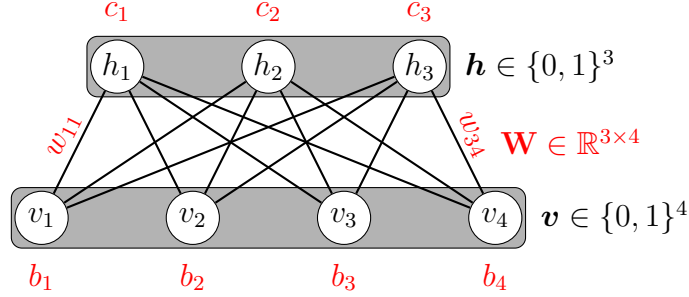
$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})}, \quad (5.1)$$

onde $E(\mathbf{v}, \mathbf{h})$ é a função denominada função de energia do modelo e o Z corresponde a constante de normalização. Também conhecida como função de partição, o termo normalizador da Equação (5.1) corresponde a

$$Z = \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}', \mathbf{h}')}. \quad (5.2)$$

Observe que seu cálculo é intratável já para valores de D e K pequenos, visto que a operação exige a soma de todas as possíveis 2^{D+K} combinações de configuração.

Figura 11 – Representação gráfica de uma RBM com quatro unidades visíveis ($D = 4$) e três unidades ocultas ($K = 3$). Os parâmetros ajustáveis pelos mecanismos de aprendizagem estão dispostos em vermelho.



Fonte: Adaptado de THOMA, 2016.

Em uma Máquina de Boltzmann Restrita, a função de energia é definida como

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}) &= -\mathbf{h}^T \mathbf{W} \mathbf{v} - \mathbf{v}^T \mathbf{b} - \mathbf{h}^T \mathbf{c} \\ &= -\sum_{j=1}^D \sum_{i=1}^K h_i w_{ij} v_j - v_j b_j - h_i c_i. \end{aligned} \quad (5.3)$$

Os parâmetros do modelo são $\Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$, onde $\mathbf{W}$ é a matriz de pesos de dimensão $K \times D$, $\mathbf{b}$ é o vetor de vieses (*bias*) da camada visível de tamanho D , onde cada viés b_j está associado a um elemento visível v_j e $\mathbf{c}$ é o vetor de vieses da camada oculta de tamanho K , onde cada viés c_i está associado a um elemento oculto h_i .

Alguns autores preferem a descrição das funções de probabilidade envolvendo os parâmetros, de forma a destacar a dependência entre uma distribuição e seus parâmetros. Por exemplo, para $P(\mathbf{v}, \mathbf{h})$, denotam com $P(\mathbf{v}, \mathbf{h}; \Theta)$ ou $P(\mathbf{v}, \mathbf{h} | \Theta)$. Para diminuir a sobrecarga da notação, foi escolhido evitar o uso, pressupondo que, ao descrever as funções, fique claro ao leitor do uso implícito dos parâmetros.

5.1.2 Marginalização da camada visível

Uma das formas para verificar se um modelo probabilístico conseguiu aprender a distribuição do conjunto de dados do treino é calculando a probabilidade de alguns dos dados do treino. Espera-se que a probabilidade enunciada pelo modelo seja alta, enquanto para um dado gerado aleatoriamente, por exemplo, seja baixa. Em *Restricted Boltzmann Machine*, a probabilidade de um determinado dado $\mathbf{v}$ é

$$P(\mathbf{v}) = \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}, \mathbf{h}') = \frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')} = \frac{1}{Z} e^{-F(\mathbf{v})}, \quad (5.4)$$

com

$$F(\mathbf{v}) = -\mathbf{v}^T \mathbf{b} - \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i). \quad (5.5)$$

A função *softplus* corresponde a $\text{soft}_+(x) = \ln(1 + e^x)$. A notação $\mathbf{W}_i$ corresponde a i -ésima linha da matriz $\mathbf{W}$ como um vetor transposto. De forma análoga, $\mathbf{W}_j$ indica a j -ésima coluna de $\mathbf{W}$ como um vetor. Uma forma alternativa da *softplus* também aplicável para o cálculo da função de energia é $\mathbf{soft}_+(\cdot)$, sua versão elemento-a-elemento (*element-wise*) $\mathbf{soft}_+(\cdot)$. Com ela, $F(\mathbf{v})$ fica como

$$F(\mathbf{v}) = -\mathbf{v}^T \mathbf{b} - \sum \mathbf{soft}_+(\mathbf{W}\mathbf{v} + \mathbf{c}).$$

Essa variação pode ser melhor aproveitada em sistemas computacionais com hardware otimizados para cálculos matriciais.

A função $F(\mathbf{v})$ apresentada nas Equações (5.4) e (5.5) é denominada função de energia livre (*free energy function*). Essa definição equivalente da probabilidade marginal de $\mathbf{v}$ é inspirada na física. No Apêndice C.1.1 encontra-se a expansão da marginalização de $\mathbf{h}$ para chegar na Função (5.4).

Mesmo a probabilidade $P(\mathbf{v})$ estando parametrizada em termos de $F(\mathbf{v})$ retirar a necessidade de percorrer todos os valores possíveis para $\mathbf{h}'$ no somatório $\sum_{\mathbf{h}' \in \mathcal{H}}$, obter o valor exato de $P(\mathbf{v})$ ainda é intratável devido a presença do termo de normalização Z . Entretanto, a forma não normalizada ou o cálculo da função de energia livre¹ podem ser utilizados para análise, pois Z é constante, invariando independentemente do valor de $\mathbf{v}$ para os mesmo valores de Θ .

5.1.3 Aprendizagem de uma RBM

No processo de aprendizagem, deseja-se descobrir quais os valores de Θ que fazem com que o modelo atribua uma alta probabilidade $P(\mathbf{v})$ para uma amostra de dados observada, uma alta probabilidade para elementos da mesma distribuição da amostra de dados e baixa probabilidade para dados que diferem da distribuição dos dados amostra. Em outras palavras, buscamos valores que façam com que o modelo indique uma baixa energia para os dados utilizados para o treino².

O processo descrito acima é chamado de estimativa da máxima verossimilhança. Queremos então minimizar $f(\Theta, \mathcal{D})$, a média negativa da log-verossimilhança (*negative log-likelihood* – NLL)³,

$$f(\Theta, \mathcal{D}) = \frac{1}{N} \sum_{n=1}^N -\ln P(\mathbf{v}_n) \quad (5.6)$$

para um conjunto de N exemplos $\mathcal{D} = \{\mathbf{v}_n\}_{n=1}^N$. Como em geral não é possível encontrar analiticamente os parâmetros de mínima verossimilhança para uma distribuição de Gibbs

¹ Quanto menor, maior a probabilidade.

² Observe na Equação (5.4) que quanto menor o valor de $F(\mathbf{v})$, maior a probabilidade devido o exponencial e o negativo ($e^{-\cdot}$).

³ Maximizar a verossimilhança é o mesmo que maximizar a log-verossimilhança.

em uma rede de Markov, são utilizados métodos numéricos de aproximação (FISCHER; IGEL, 2012). Será utilizado nesse trabalho um processo de otimização formalmente similar ao do gradiente descendente (*gradient descent*) (YUILLE, 2005).

O algoritmo gradiente descendente consiste em solucionar o problema de minimização $\min f(\mathbf{x})$ para uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ a partir de um ponto inicial $\mathbf{x}^{(0)}$ por meio de um processo iterativo utilizando informação da primeira derivada de f . Considerando que a função objetivo tem solução, existe então um ponto ótimo $\mathbf{x}^*$, tal que $\nabla f(\mathbf{x}^*) = \mathbf{0}$, é também solução do problema de minimização.

Algoritmo 1 Gradiente descendente. Adaptado de BOYD; VANDENBERGHE, 2004.

```

1  #  $\mathbf{x}^{(0)}$ : ponto inicial  $\in \text{dom } f$ 
2  #  $\lambda$ : tolerância. É pequeno e positivo
3  def gradient_descent( $\mathbf{x}^{(0)}$ : vector,  $\lambda$ : real) -> vector:
4       $\mathbf{x} = \mathbf{x}^{(0)}$ 
5      while True:
6           $\Delta \mathbf{x} = -\nabla f(\mathbf{x})$  # Direção da busca
7           $\eta = \text{line\_search}()$  # line_search() busca o tamanho do passo  $\eta$ 
8           $\mathbf{x} = \mathbf{x} + \eta \Delta \mathbf{x}$ 
9
10         if  $\|\nabla f(\mathbf{x})\|_2 \leq \lambda$ : # Critério de parada
11             return  $\mathbf{x}$ 

```

Para uma função convexa em todo o domínio de $\mathbf{x}$, há garantias de que o Algoritmo 1 estará mais próximo do ponto ótimo a cada passo da iteração, i.e., $f(\mathbf{x}^{(n+1)}) \leq f(\mathbf{x}^{(n)})$, exceto no ponto ótimo, onde $f(\mathbf{x}^{(n+1)}) = f(\mathbf{x}^{(n)}) = f(\mathbf{x}^*)$ (BOYD; VANDENBERGHE, 2004, Capítulo 9.3.1).

O gradiente da função objetivo (5.6) pode ser decomposta em duas fases, positiva e negativa

$$\nabla_{\theta} f(\theta, \mathcal{D}) = \underbrace{\frac{1}{N} \sum_{n=1}^N \nabla_{\theta} F(\mathbf{v}_n)}_{\text{Fase positiva}} - \underbrace{\sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') \nabla_{\theta} F(\mathbf{v}')}_{\text{Fase negativa}}. \quad (5.7)$$

Intuitivamente, a fase positiva aumenta a probabilidade de amostras do conjunto de treino ($\mathbf{v}'$) e a fase negativa diminui a probabilidade de exemplos de amostras geradas pelo modelo ($\mathbf{v}' \in \mathcal{V}$) (CÔTÉ; LAROCHELLE, 2016).

Ao tratar os gradientes das funções de energia como transformações, é possível simplificar a notação da Equação (5.7) em termos da esperança. Para um único padrão $\mathbf{v}$, temos

$$\nabla_{\theta} \ln P(\mathbf{v}) = - \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{h}' | \mathbf{v}) \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') + \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}', \mathbf{h}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \quad (5.8)$$

$$= - \mathbb{E}_{P(\mathbf{h}|\mathbf{v})} [\nabla_{\theta} E(\mathbf{v}, \mathbf{h}')] + \mathbb{E}_{P(\mathbf{v}, \mathbf{h})} [\nabla_{\theta} E(\mathbf{v}', \mathbf{h}')] \quad (5.9)$$

$$= - \mathbb{E}_{data} [\nabla_{\theta} E(\mathbf{v}, \mathbf{h}')] + \mathbb{E}_{model} [\nabla_{\theta} E(\mathbf{v}', \mathbf{h}')] \quad (5.10)$$

A $\mathbb{E}_{data}[\cdot]$ representa os valores esperados de cada variável aleatória $h_i \in \mathbf{h}$ dada uma atribuição específica de $\mathbf{v} \in \mathcal{V}$. A $\mathbb{E}_{model}[\cdot]$ representa valores esperados do modelo aprendido no determinado instante de tempo⁴.

Uma forma alternativa de representação da Equação (5.8) é

$$\nabla_{\theta} \ln P(\mathbf{v}) = -\langle \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') \rangle_{data} + \langle \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \rangle_{model}. \quad (5.11)$$

Nela, $\langle \cdot \rangle_{distribution}$ denota a esperança sobre a distribuição (*distribution*) que está subscrita.

As derivadas parciais da função de energia livre dado os parâmetros do modelo são

$$\begin{aligned} \frac{\partial F(\mathbf{v})}{\partial w_{ij}} &= -\sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) v_j = -P(h_i | \mathbf{v}) v_j \\ &= -\mathbb{E}_{P(h_i | \mathbf{v})} [h_i] v_j, \\ \frac{\partial F(\mathbf{v})}{\partial c_i} &= -\sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) = -P(h_i | \mathbf{v}) \\ &= -\mathbb{E}_{P(h_i | \mathbf{v})} [h_i], \\ \frac{\partial F(\mathbf{v})}{\partial b_j} &= -v_j \end{aligned} \quad (5.12)$$

onde $\sigma(x) = \frac{1}{1+e^{-x}}$ é a função sigmoide. A expansão do gradiente da Equação (5.7) está no Apêndice C.2, as derivadas parciais da Equação (5.12) no Apêndice C.2.4 e a relação delas com a esperança condicional ($\mathbb{E}_{P(h_i | \mathbf{v})} [h_i]$) está presente no Apêndice C.2.5. As probabilidades condicionais serão comentadas adiante.

Associando as Equações (5.7), (5.8) e (5.11), a equação $\nabla_{\theta} f(\theta, \mathcal{D})$ pode se descrita como

$$\begin{aligned} \nabla_{\theta} f(\theta, \mathcal{D}) &= \frac{1}{N} \sum_{n=1}^N \nabla_{\theta} F(\mathbf{v}_n) - \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') \nabla_{\theta} F(\mathbf{v}') \\ &= \frac{1}{N} \sum_{n=1}^N \left(-\mathbb{E}_{data} [\nabla_{\theta} E(\mathbf{v}_n, \mathbf{h}')] + \mathbb{E}_{model} [\nabla_{\theta} E(\mathbf{v}', \mathbf{h}')] \right) \\ &= \frac{1}{N} \sum_{n=1}^N \left(-\langle \nabla_{\theta} E(\mathbf{v}_n, \mathbf{h}') \rangle_{data} + \langle \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \rangle_{model} \right). \end{aligned}$$

Observe que a fase negativa da Equação (5.7) não pode ser computada analiticamente em tempo inferior a um tempo exponencial, já que o cálculo da probabilidade $P(\mathbf{v})$ envolve o cálculo de Z . É possível contornar esse problema estimando a esperança $\langle \cdot \rangle_{model}$ por meio de uma média de um conjunto $\mathcal{S} = \{\hat{\mathbf{v}}_s\}_{s=1}^S$ contendo S amostras geradas a partir da distribuição do modelo:

$$\nabla_{\theta} f(\theta, \mathcal{D}) \approx \frac{1}{N} \sum_{n=1}^N \nabla_{\theta} F(\mathbf{v}_n) - \frac{1}{S} \sum_{s=1}^S \nabla_{\theta} F(\hat{\mathbf{v}}_s). \quad (5.13)$$

⁴ Como é parte do algoritmo de otimização, a esperança será “atualizada” a cada atualização dos parâmetros (θ).

Uma das formas de gerar tais amostras é por meio da amostragem de Gibbs elemento por elemento sequencialmente, tal como ocorre com BM (Subseção B.2). No entanto, é necessário executar este MCMC o tempo o suficiente para garantir a convergência à estacionariedade. Desta forma, o custo do algoritmo continua sendo alto, de forma que não permita uma aprendizagem computacionalmente eficiente.

No início deste século, alternativas a esse processo de amostragem na aprendizagem foram desenvolvidas. Dentre elas, algoritmos que geram amostras “aproximadas” com um custo computacional inferior à técnica MCMC anteriormente comentada ganharam destaque nas pesquisas envolvendo Máquinas de Boltzmann Restritas. A subseção seguinte apresentará dois desses algoritmos: o *Contrastive Divergence* (CD) e o *Persistent Contrastive Divergence* (PCD).

5.1.4 Algoritmos MCMC com amostragem aproximada

Será tratada nessa subseção algoritmos MCMC com amostragem aproximada. Inicialmente será apresentado um algoritmo ingênuo (*naive algorithm*) informando a estrutura base dos algoritmos CD e PCD. Posteriormente serão apresentadas as modificações necessárias no *naive algorithm* para chegar nos algoritmos CD e PCD. Por fim, será apresentada a amostragem de Gibbs em bloco aplicada a Máquina de Boltzmann Restrita.

5.1.4.1 Algoritmo ingênuo para MCMC

Uma maneira ingênua de implementar um algoritmo MCMC para a Equação (5.13) é gerando amostras aleatórias e executando a amostragem de Gibbs para obter S amostras. Essas amostras podem ser obtidas a partir de distribuições uniformes ou normais, ou possivelmente uma distribuição cujas marginais correspondem com as marginais do modelo (Algoritmo 2, linha 10) (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para que as amostras aleatórias correspondam à fase negativa da Equação (5.7), é necessário que a cadeia de Markov seja executada tempo o suficiente para que seja *burned*. No algoritmo *naive*, significa que, em cada passo do gradiente, deverão ser executados k passos para que cada $\hat{\mathbf{x}}_s$ corresponda a distribuição do modelo (Algoritmo 2, linha 12), de forma que possivelmente será necessário um k grande, tornando o procedimento computacionalmente inviável.

5.1.4.2 Algoritmo Contrastive Divergence

O algoritmo apresentado a seguir é o *Contrastive Divergence* (em português, Divergência Contrastiva), também conhecido como CD- k . Hinton o desenvolveu para a classe probabilística *Products of Experts*, mas como RBMs são a interseção de BM e

Algoritmo 2 Um algoritmo MCMC ingênuo para minimizar a log-verossimilhança com uma função de partição intratável usando gradiente descendente. Adaptado de GOODFELLOW; BENGIO; COURVILLE, 2016.

```

1  #  $\mathcal{D}$ : Conjunto de dados de treino
2  #  $\eta$ : tamanho do passo ou taxa de aprendizagem
3  #  $k$ : quantidade de passos de Gibbs por iteração
4  #  $M$ : tamanho dos minibatches;  $M \leq |\mathcal{D}|$ 
5  #  $S$ : quantidade de amostras a serem geradas; geralmente  $S = M$ 
6  #  $V$ : tamanho de  $x$ ;  $V = |x_i| \ \forall \ x_i \in \mathcal{D}$ 
7  def naive_mcmc( $\mathcal{D}$ : set,  $\eta$ : float,  $k$ : int,  $M$ : int,  $S$ : int,  $V$ : int) -> vector:
8      while not converged:
9           $\{x_1, \dots, x_M\} = \text{get\_a\_minibatch}(\mathcal{D}, \text{samples}=M)$   #  $\{x_1, \dots, x_M\} \subseteq \mathcal{D}$ 
10          $\{\hat{x}_1, \dots, \hat{x}_S\} = \text{random\_samples}(\text{amout}=S, \text{size}=V)$ 
11
12         for each  $i \in [1, k]$ :
13             for each  $s \in [1, S]$ :
14                  $\hat{x}_s = \text{gibbs\_update}(\hat{x}_s)$ 
15
16          $\Delta x = \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \tilde{P}(x_i) - \frac{1}{S} \sum_{s=1}^S \nabla_{\theta} \log \tilde{P}(\hat{x}_s)$ 
17          $\theta = \theta - \eta \Delta x$ 
18
19     return  $\theta$ 

```

Products of Experts, o algoritmo definido também enquadra-se à utilização (HINTON, 2002).

No CD- k , ao invés de serem geradas S amostras aleatórias, são utilizadas amostras obtidas a partir da base de treino $\mathcal{D}$ (Algoritmo 3, linha 9). A ideia é de que a inicialização da cadeia de Markov a partir de uma distribuição que está próxima o suficiente da distribuição do modelo fará com que não sejam necessários muitos passos para que se chegue na distribuição estacionária. Por exemplo, é comum utilizarem na literatura um único passo (CD-1, ou somente CD) com a base de dados MNIST. Para estruturas mais complexas a serem aprendidas, utilizam talvez 10 ou mais passos, entretanto isso exige um maior custo computacional (GOODFELLOW; BENGIO; COURVILLE, 2016; BENGIO; DELALLEAU, 2009; LAROCHELLE et al., 2007; TIELEMAN, 2008).

5.1.4.3 Algoritmo Persistent Contrastive Divergence

Uma estratégia diferente de CD e do algoritmo ingênuo é, ao invés de inicializar a rede de Markov a cada passo do gradiente, é gerar as amostras uma única vez e a cada passo do gradiente, aproveitar os valores da amostra anterior (Algoritmo 4, linha 8). Este é o princípio do *Persistent Contrastive Divergence* (PCD- k , em português, Divergência Contrastiva Persistente) (TIELEMAN, 2008).

O algoritmo PCD definido por Tieleman (2008) é uma redescoberta independente do *Stochastic Maximum Likelihood* (SML) (YOUNES, 1999) utilizado nas comunidades matemáticas e estatísticas.

Algoritmo 3 *Contrastive Divergence* com k passos (CD- k). Adaptado de GOODFELLOW; BENGIO; COURVILLE, 2016.

```

1 #  $\mathcal{D}$ : Conjunto de dados de treino
2 #  $\eta$ : tamanho do passo ou taxa de aprendizagem
3 #  $k$ : quantidade de passos de Gibbs por iteração
4 #  $M$ : tamanho dos minibatches;  $M \leq |\mathcal{D}|$ 
5 #  $V$ : tamanho de  $x$ ;  $V = |x_i| \ \forall x_i \in \mathcal{D}$ 
6 def CDk( $\mathcal{D}$ : set,  $\eta$ : float,  $k$ : int,  $M$ : int,  $V$ : int) -> vector:
7     while not converged:
8          $\{x_1, \dots, x_M\} = \text{get\_a\_minibatch}(\mathcal{D}, \text{samples}=m)$  #  $\{x_1, \dots, x_M\} \subseteq \mathcal{D}$ 
9          $\{\hat{x}_1, \dots, \hat{x}_M\} = \{x_1, \dots, x_M\}$ 
10
11         for each  $i \in [1, k]$ :
12             for each  $j \in [1, M]$ :
13                  $\hat{x}_j = \text{gibbs\_update}(\hat{x}_j)$ 
14
15              $\Delta x = \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \tilde{P}(x_i) - \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \tilde{P}(\hat{x}_i)$ 
16              $\theta = \theta - \eta \Delta x$ 
17
18     return  $\theta$ 

```

Algoritmo 4 *Persistent Contrastive Divergence* com k passos (PCD- k). Adaptado de GOODFELLOW; BENGIO; COURVILLE, 2016.

```

1 #  $\mathcal{D}$ : Conjunto de dados de treino
2 #  $\eta$ : tamanho do passo ou taxa de aprendizagem
3 #  $k$ : quantidade de passos de Gibbs por iteração
4 #  $M$ : tamanho dos minibatches;  $M \leq |\mathcal{D}|$ 
5 #  $S$ : quantidade de amostras a serem geradas; geralmente  $S = M$ 
6 #  $V$ : tamanho de  $x$ ;  $V = |x_i| \ \forall x_i \in \mathcal{D}$ 
7 def PCDk( $\mathcal{D}$ : set,  $\eta$ : float,  $k$ : int,  $M$ : int,  $S$ : int,  $V$ : int) -> vector:
8      $\{\hat{x}_1, \dots, \hat{x}_M\} = \text{random\_samples}(\text{amout}=M, \text{size}=V)$ 
9     while not converged:
10          $\{x_1, \dots, x_S\} = \text{get\_a\_minibatch}(\mathcal{D}, \text{samples}=S)$  #  $\{x_1, \dots, x_S\} \subseteq \mathcal{D}$ 
11
12         for each  $i \in [1, k]$ :
13             for each  $s \in [1, S]$ :
14                  $\hat{x}_s = \text{gibbs\_update}(\hat{x}_s)$ 
15
16              $\Delta x = \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \tilde{P}(x_i) - \frac{1}{S} \sum_{s=1}^S \nabla_{\theta} \log \tilde{P}(\hat{x}_s)$ 
17              $\theta = \theta - \eta \Delta x$ 
18
19     return  $\theta$ 

```

5.1.4.4 Gibbs sampling em bloco para RBM

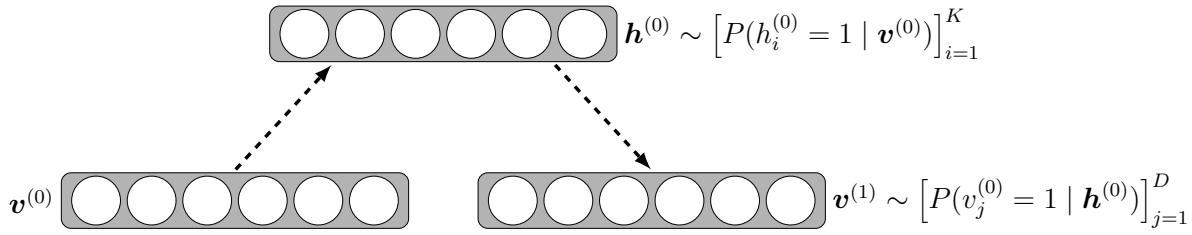
Sabemos que a estrutura bipartida do grafo indica que um elemento de uma camada é independente dos outros de mesma camada dada a camada posterior: $v_i \perp\!\!\!\perp v_l \mid \mathbf{h}$ e $h_j \perp\!\!\!\perp h_k \mid \mathbf{v}$. Uma prova pode ser vista no Apêndice C.3. Tal característica da RBM possibilita realizar uma amostragem de Gibbs **em bloco**, alternando a amostragem de

$\mathbf{h} \sim [P(h_i = 1|\mathbf{v})]_{i=1}^K$ e $\mathbf{v} \sim [P(v_j = 1|\mathbf{h})]_{j=1}^D$, onde

$$\begin{aligned} [P(h_i = 1|\mathbf{v})]_{i=1}^K &= \sigma(\mathbf{W}\mathbf{v} + \mathbf{c}), \\ [P(v_j = 1|\mathbf{h})]_{j=1}^D &= \sigma(\mathbf{W}^T\mathbf{h} + \mathbf{b}). \end{aligned}$$

O cálculo das probabilidades condicionais também pode ser vista no Apêndice C.3.

Figura 12 – Processo de atualização em blocos dos elementos da camadas visível e oculta.



Fonte: Próprio autor.

Assim, podemos definir o algoritmo *Gibbs update* aplicável para os algoritmos apresentados anteriormente nesta mesma seção quando são executadas para uma Máquina de Boltzmann Restrita (Algoritmo 5). Uma representação gráfica do processo de atualização de Gibbs em bloco é apresentada na Figura 12.

Algoritmo 5 *Gibbs update* em bloco. Todos os elementos de uma mesma camada são amostrados simultaneamente.

```

1 def gibbs_update(v(0): vector) -> vector:
2     h(0) ~ [P(hi(0) = 1 | v(0))]i=1K
3     v(1) ~ [P(vj(0) = 1 | h(0))]j=1D
4
5     return v(1)

```

5.1.5 Discussão

Caso o leitor esteja interessado sobre o processo de otimização por gradiente descendente, é recomendado o livro “*Convex Optimization*” de Boyd e Vandenberghe (2004). O livro aborda a base matemática para a realização otimização em funções convexas, quasi-convexas, em problemas com e sem restrição⁵. Para uma visualização iterativa sobre algoritmos de busca e a influência dos parâmetros na busca, o site “*Why Momentum Really Works*” (<<https://distill.pub/2017/momentum/>>) apresenta uma discussão sobre *momentum* (GOH, 2017).

⁵ A notação do livro pode confundir alguns.

Sobre a notação para Máquinas de Boltzmann Restritas, não existe uma convenção de uso. Parte da notação desse manuscrito é inspirada na notação do artigo “*An Infinite Restricted Boltzmann Machine*” (CÔTÉ; LAROCHELLE, 2016).

Hinton e Larochelle, pesquisadores que contribuíram bastante na área de aprendizagem de máquina aqui abordada, apresentam explicações sobre RBMs. Elas podem ser acessadas nos endereços eletrônicos, respectivamente <<https://youtu.be/EZOpZzUKl48>> e <https://www.youtube.com/watch?v=e0Ts_7Y6hZU>.

Informações relativas ao poder de expressividade das Máquinas de Boltzmann Restritas podem ser consultadas na publicação “*Representational power of restricted boltzmann machines and deep belief networks*” (ROUX; BENGIO, 2008).

Existem discussões pertinentes sobre o processo de otimização por utilização de amostras aproximadas. Apesar de *Contrastive Divergence* obter bons resultados, a função de fato não aproxima a função desejada seja com k passos (CD- k) (BENGIO; DELALLEAU, 2009) ou mesmo com um passo (CD-1) (TIELEMAN, 2008). Entretanto, atua como uma aproximação, de forma que, ao minimizar a função objetivo por meio dele, aproxima do resultado desejado, mas um *overfitting* ocasionado por uma execução demasiada não implicará em melhores resultados na base do treino.

Ao comparar o algoritmo gradiente descendente com os algoritmos MCMC com amostragem aproximada, pode se perceber algumas diferenças além das apresentadas anteriormente. Por exemplo, o tamanho do passo (ou também conhecido como taxa de aprendizagem) no gradiente descendente é obtido a partir de uma função de forma que seu valor pode ser adaptado para um valor melhor a cada interação. Um exemplo de função é *busca linear exata*. Já nos algoritmos MCMC, é apresentado como uma constante η . Para que o algoritmo convirja, é necessário um η pequeno o suficiente. Existem outras implementações que também utilizam uma taxa de aprendizagem que varia seu valor no processo de aprendizagem. Um exemplo é a utilização de ADAGRAD (PENG; GAO; LI, 2018; PASSOS; PAPA, 2017).

Outro ponto sobre gradiente descendente é que não há garantias sobre a função a ser otimizada na RBM ser convexa, de forma que pode não haver um único ponto ótimo global. Porém, pela função de log-verossimilhança ser composta por uma subtração de duas funções convexas, o trabalho “*Learning RBM with a DC programming Approach*” de Upadhy e Sastry (2017) utiliza este fato para treinar de uma forma distinta.

Como comentado, é esperado que um modelo treinado tenha conseguido aprender sobre o problema, i.e., ele deve indicar uma alta probabilidade para a base de treino (o modelo aprendeu o que foi ensinado), uma alta probabilidade para a base de teste (o modelo generalizou o problema) e uma baixa probabilidade para dados que não correspondem ao problema. É comum os pesquisadores testarem diferentes valores para os hiperparâmetros

ou ainda definirem variações do modelo buscando um melhor resultado. Entretanto, o cálculo da função de partição torna inviável o cálculo e a comparação da probabilidade $P(\mathbf{v})$ entre os modelos. A comparação da função de energia livre (também presente em $P(\mathbf{v})$) não é recomendável, pois seu valor depende justamente do contexto, que é a função de partição.

Uma alternativa para contornar o cálculo exato de $P(\mathbf{v})$ é estimar a constante Z . *Annealed Importance Sampling* (AIS) (NEAL, 2001) vem sendo usado na literatura como método padrão para estimar a constante de normalização em diversos modelos gráficos não direcionados (SALAKHUTDINOV, 2008; GOODFELLOW; BENGIO; COURVILLE, 2016). Porém, seu custo computacional é suficientemente intensivo, de forma que o cálculo durante o treinamento permanece inviável. Desjardins, Bengio e Courville (2011) fizeram uma combinação entre métodos que estimam Z para acompanhá-lo durante o treinamento com um custo computacional similar ao treinar o modelo sem o algoritmo.

5.2 Restricted Boltzmann Machine para Filtragem Colaborativa

Restricted Boltzmann Machine for Collaborative Filtering (RBM-CF) ou Máquina de Boltzmann Restrita para Filtragem Colaborativa é uma extensão da RBM originalmente concebida para sistemas de recomendação baseados em filtragem colaborativa (collaborative filtering, CF) orientada a usuário (*user-oriented*) (SALAKHUTDINOV; MNIH; HINTON, 2007). A representação desse modelo traz a ideia da modelagem de suas variáveis aleatórias visíveis para dados categóricos, assim, o modelo também é conhecido na literatura como *Categorical RBM* (MURPHY, 2012).

Seu surgimento remete-se ao *Netflix Prize*, uma competição com um prêmio de um milhão de dólares para a equipe que apresentasse o melhor algoritmo para prever avaliações de usuários para filmes sobre um conjunto de dados contendo 100.480.507 milhões de avaliações anônimas de filmes fornecida pela Netflix (BENNETT; LANNING, 2007). A base de dados é composta por avaliações realizadas por 480.189 assinantes anônimos do sistema escolhidos aleatoriamente em 17.770 obras. Uma avaliação é representada por um escalar no intervalo $[1, 5]$ correspondente a quantidade de estrelas dadas ao filme (quanto maior, mais bem avaliado pelo usuário).

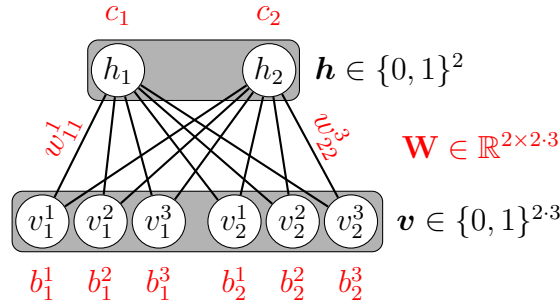
5.2.1 Representação de uma RBM-CF

Em uma RBM-CF, a camada visível é decomposta em M grupos, de forma que $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M]^T$. Cada grupo pode assumir um valor em $[1, R]$, representado em uma codificação *one-hot*, implicando em $D = MR$, onde M corresponde a quantidade de

elementos avaliáveis⁶ e R referencia-se a nota máxima para os elementos avaliáveis⁷. Desse modo, a camada visível é representada por uma codificação *one-hot* M -dimensional.

Uma unidade visível é representada como v_m^r , indicando o r -ésimo elemento do m -ésimo grupo. Um *bias* da camada visível b_m^r está atrelado ao v_m^r elemento também da camada visível. Similar, o sobrescrito m e o subscrito r em w_{im}^r indicam que tal peso está relacionado com v_m^r , e, assim como na RBM, i está relacionada a i -ésima variável aleatória da camada oculta.

Figura 13 – Representação gráfica de uma RBM-CF com seis unidades visíveis, na qual as três primeiras unidades (v_1^1, v_1^2, v_1^3) correspondem ao *rating* para o primeiro filme e as três unidades seguintes (v_2^1, v_2^2, v_2^3) para o segundo filme. Os parâmetros ajustáveis pelos mecanismos de aprendizagem estão dispostos em vermelho.



Fonte: Próprio autor.

Exemplificando no contexto dos filmes, para $M = 2$, $R = 3$ e $|\mathbf{v}| = D = MR$ (Figura 13), temos $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2]^T = [(1, 0, 0), (0, 1, 0)]^T$ um vetor válido correspondente as avaliações ★☆☆ (uma estrela de três) para o filme 1 e ★★☆☆ (duas estrelas de três) para o filme 2. As tuplas presentes na representação do vetor $\mathbf{v}$ possuem caráter representativo, indicando a codificação *one hot* para cada filme. Atente-se que a codificação *one hot* implica que quando uma variável aleatória v_m^r estiver ativa ($v_m^r = 1$), todas as outras do mesmo grupo **não devem** estarem ativas ($\mathbf{v}_m^{i \neq r} = \mathbf{0}$). Dessa forma $[(1, 1, 0), (0, 1, 0)]^T$ é um valor inválido para $\mathbf{v}$.

Uma distribuição multinomial condicional (*softmax*) é usada para modelar cada grupo da camada visível e uma distribuição de Bernoulli condicional para modelar a camada latente do modelo:

$$P(v_m^r = 1 | \mathbf{h}) = \text{softmax}(b_m^r + \sum_{i=1}^K h_i w_{im}^r) = \frac{\exp(b_m^r + \sum_{i=1}^K h_i w_{im}^r)}{\sum_{l=1}^R \exp(b_m^l + \sum_{i=1}^K h_i w_{im}^l)}, \quad (5.14)$$

$$P(h_i = 1 | \mathbf{v}) = \sigma(c_i + \sum_{m=1}^M \sum_{r=1}^R w_{im}^r v_m^r) = \sigma(c_i + \mathbf{W}_i \cdot \mathbf{v}). \quad (5.15)$$

⁶ M de *movies*, filmes.

⁷ R de *ratings*, avaliações.

Atente-se que a esparsidade nas bases de dados aplicáveis para sistemas de recomendação com filtragem colaborativa costuma ser elevada: existe uma grande quantidade de itens avaliáveis e provavelmente os usuários terão conhecimento para avaliar poucos. Em RBM-CF, as avaliações ausentes (*missing ratings*) não realizam quaisquer contribuições para a função de energia. Isto pode ser computacionalmente definindo que todos os elementos dos grupos que não possuem avaliação são zero (por exemplo, para o m -ésimo filme de um vetor $\mathbf{v}$ qualquer, $\mathbf{v}_m = \mathbf{0}$).

5.2.2 Realização de predições

É possível realizar o cálculo exato de uma possível avaliação para um grupo dada as outras avaliações em um tempo linear em relação a quantidade de elementos ocultos. Entretanto, a predição da avaliação esperada para n grupos exige K^n cálculos para cada usuário que se deseja prever as avaliações (SALAKHUTDINOV; MNIH; HINTON, 2007).

Como alternativa, é possível obter uma estimativa por meio de um processo similar a uma atualização de Gibbs em bloco (Algoritmo 6):

1. calcula a probabilidade condicional de ativação para cada elemento h_i da camada oculta

$$\hat{p}_{h_i} = P(h_i = 1 | \mathbf{v});$$

2. calcula a esperança condicional para as R avaliações de um filmes m

$$\mathbb{E}_{P(\mathbf{v}_m | \hat{\mathbf{p}}_h)}[v_m^r] = \sum_{r=1}^R r P(v_m^r = 1 | \hat{\mathbf{p}}_h). \quad (5.16)$$

Além da esperança, o processo diferencia-se do *block Gibbs sampling* pela utilização das probabilidades $\hat{\mathbf{p}}_h$ no lugar de uma amostra realizada por elas ($\mathbf{h}^{(1)} \sim \hat{\mathbf{p}}_h$). Tal processo foi provavelmente definido pelos autores para evitar que ruídos gerados pela amostragem interferissem nas predições. Essa estratégia também costuma ser aplicada no processo de treinamento, na última atualização de um CD- k com $k > 1$ (HINTON, 2012).

Algoritmo 6 Predição de uma avaliação para o m -ésimo grupo.

```

1 # v: Camada visível
2 # m: m-ésimo grupo cuja avaliação será predita
3 def predict(v: vector, m: int) -> vector:
4      $\hat{\mathbf{p}}_h = [P(h_i = 1 | \mathbf{v})]_{i=1}^K$ 
5     return  $\sum_{r=1}^R r P(v_m^r = 1 | \hat{\mathbf{p}}_h)$ 
```

5.2.3 Discussão

A base disponibilizada pela Netflix também contém a informação dos filmes assistidos que os usuários não avaliaram. Salakhutdinov, Mnih e Hinton (2007) utilizam uma variação da RBM denominada Conditional RBM para inserir tal informação no modelo por meio de uma camada conectada a camada oculta. No mesmo artigo outras técnicas ainda foram abordadas, como a utilização de unidades ocultas gaussianas, e a fatoração da matriz $\mathbf{W}$ como o produto de duas matrizes de *rank* menor para contornar o aprendizado de 9 milhões de parâmetros⁸.

Enquanto estratégias para filtragem colaborativa baseadas em usuário modelam a similaridade entre os diferentes usuários para então sugerir itens, abordagens baseadas em itens (*item-based*) modelam a similaridade entre os itens distintos como forma de descobrir potenciais usuários para os itens. Georgiev e Nakov (2013) em “*A non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines*” utilizam uma única RBM — denominada como UI-RBM — para modelar as interações usuário-usuário e item-item por meio de duas camadas ocultas, uma destinada para cada tipo de interação. A representação para as avaliações é por meio de números reais no lugar de categórica. Nos experimentos realizados, foi constatado que a predição dos elementos visíveis obteve melhor resultado utilizando reconstruções livres de ruído (*noise-free reconstructions*), i.e., quando o valor de uma unidade visível é a soma ponderada da camada visível mais o bias.

“*Content-boosted Restricted Boltzmann Machine*” (CB-RBM) (LIU et al., 2014) utiliza uma estratégia similar a de UI-RBM, mas alterando alguns passos:

- é gerado inicialmente uma matriz de avaliações (Usuários x Itens) na qual alguns dos *missing ratings* são previamente populados a partir de um classificador Naïve Bayes sobre uma métrica definida. A matriz gerada é a utilizada no treinamento dessa extensão da RBM;
- no processo de treinamento, a camada visível normalizada é utilizada como a probabilidade de ativação da camada oculta, tal como sugerido por Hinton, Osindero e Teh (2006);
- no cálculo da reconstrução das unidades visíveis, a utilização da função logística foi utilizada ao invés de *noise-free reconstructions* por apresentar melhor resultado.

Diferentemente, “*Item Category Aware Conditional Restricted Boltzmann Machine Based Recommendation*” (IC-CRBMF) de Liu et al. (2015) utiliza-se de duas RBMs distintas para predizer o *rating* R de um item qualquer

$$R = \beta \cdot R_{IC-RBMF_{\text{ItemBased}}} + (1 - \beta) \cdot R_{IC-RBMF_{\text{UserBased}}}, \quad (5.17)$$

⁸ $\mathbf{W} \in \mathbb{R}^{M \times R \times K}$, onde $K = 100$, $R = 5$ e $M = 17770$.

onde $R_{IC-RBMF_{ItemBased}}$ corresponde a recomendação dada pela RBM baseada em itens, $R_{IC-RBMF_{UserBased}}$ corresponde a recomendação dada pela RBM baseada em usuários e β um coeficiente ponderado calculado empiricamente. Similar ao Conditional RBM-CF, IC-CRBMF possui um vetor de atributos (*features*) para informar ao modelo as categorias dos itens (para RBM *item-based*) ou das categorias dos itens relacionados a um usuário (para a contrapartida *user-based*). Porém, a camada se relaciona tanto com os elementos visíveis quanto com os elementos ocultos. As categorias referem-se aos gêneros das obras cinematográficas, informações fornecidas pelas bases de dados MovieLens 100k e MovieLens 1M utilizadas no desenvolvimento do artigo.

O trabalho “*User Occupation Aware Conditional Restricted Boltzmann Machine Based Recommendation*” de Xie et al. (2016) apresenta o modelo UO-CRBMF. Aplica-se a mesma estratégia do IC-CRBMF, mas utiliza dados demográficos dos usuários, mais especificamente a profissão deles. O artigo ainda apresenta uma versão híbrida de UO-CRBMF e IC-CRBMF, combinando linearmente de forma ponderada os *ratings* (tal como na Equação (5.17)), mas utiliza informações sobre entropia para definir o β .

“*Collaborative Filtering Based on Dual Conditional Restricted Boltzmann Machines*” (dCRBM) (LI; LI, 2017) apresenta uma estratégia similar ao IC-CRBMF, porém a base de dados encontra-se em formato *one hot* e as ponderações α e β no cálculo de uma predição qualquer

$$R = \alpha \cdot R_{\text{user}} + \beta \cdot R_{\text{item}}.$$

são parâmetros treináveis a partir de uma função de penalização.

O artigo “*Explainable Restricted Boltzmann Machines for Collaborative Filtering*” (ABDOLLAHI; NASRAOUI, 2016) utiliza a camada de atributos (da Conditional RBM) para extrair informações sobre a explicabilidade dos itens recomendados pelo modelo. As informações são avaliadas pela métrica *Explainability Score* proposta.

5.3 Restricted Boltzmann Machine para recomendação de *plugins* de áudio

O passo inicial para a aplicação dos modelos probabilísticos da família RBM é a definição de uma codificação de entrada que permita que os modelos considerem o posicionamento das instâncias em um *patch* e a possibilidade de existirem mais de uma instância de um mesmo *plugin*.

Na representação aplicada, a camada visível $\mathbf{v}$ é decomposta em M vetores ($\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M]^T$), um para cada posição no *patch*. Formalmente, uma m -ésima posição corresponde a um vetor *one-hot* de variáveis aleatórias $\mathbf{v}_m = [v_m^1, \dots, v_m^R]^T$, onde

$v_m^r = 1$ implica que $v_m^i = 0 \forall i \in \{1, \dots, R\} \setminus \{r\}$ e que o r -ésimo *plugin* é o selecionado (ativo) para a m -ésima posição do *patch*.

A representação de entrada é inspirada em Hinton e Salakhutdinov (2009) e é conhecida como “representação *one-hot* M -dimensional”. Como pode ser observado, a estratégia aplicada é baseada na representação exposta na Subseção 5.2.1, em que a entrada foi codificada para recomendação de filmes, mas aqui aplicável para o caso dos *plugins* de áudio, seguindo as associações previamente comentadas na Seção 3.3.

Desejamos obter uma lista de recomendações de *plugins* para uma instância ausente de determinado *patch*. Probabilisticamente, desejamos que os modelos informem as probabilidades para cada um dos possíveis *plugins* para a m -ésima instância (que se encontra ausente) dadas as instâncias definidas. A probabilidade de um r -ésimo *plugin* dada as instancias definidas é $P(v_m^r = 1 | \mathbf{v}_{-m})$. Ela pode ser aproximada conforme as aproximações condicionais apresentas anteriormente para o cálculo de $P(\mathbf{v}_{\text{unknown}} | \mathbf{v}_{\text{visible}})$: obtemos uma aproximação das probabilidades condicionais ($P(v_m^r = 1 | \mathbf{v}_{-m})$) para cada r -ésimo *plugin* da m -ésima posição (v_m^r) pelas Equações (5.18) e (5.19)

$$\hat{p}_i = P(h_i = 1 | \mathbf{v}) = \sigma(c_i + \sum_{m=1}^M \sum_{r=1}^R w_{im}^r v_m^r) \quad (5.18)$$

$$\hat{q}_m^r = P(v_m^r = 1 | \hat{\mathbf{p}}) = \begin{cases} \sigma(b_m^r + \sum_{i=1}^K w_{im}^r \hat{p}_i), & \text{se é RBM} \\ \frac{\exp(b_m^r + \sum_{i=1}^K \hat{p}_i w_{im}^r)}{\sum_{l=1}^R \exp(b_m^l + \sum_{i=1}^K \hat{p}_i w_{im}^l)}, & \text{se é RBM-CF.} \end{cases} \quad (5.19)$$

Perceba que na RBM-CF de Salakhutdinov, Mnih e Hinton (2007), o processo de obtenção de $\hat{q}_m$, a estimativa de avaliação para um m -ésimo filme, é finalizado pela esperança dada na Equação (5.16):

$$\mathbb{E}_{P(v_m^r | \hat{\mathbf{p}}_h)}[v_m^r] = \hat{q}_m = \sum_{r=1}^R \hat{q}_m^r.$$

Entretanto, para o problema de recomendação de *plugins* de áudio, utilizamos o próprio vetor $\hat{\mathbf{q}}_m$ como a lista de recomendações.

A estratégia de definir uma lista de recomendações baseadas na probabilidade de cada *plugin* de áudio, em vez obter um único a partir da esperança, foi útil porque R é alto no problema tratado e o cálculo da esperança não seria eficaz se mais de um *plugin* apresentar alta probabilidade, já que a ordem destes não corresponde a nenhum critério de similaridade.

Ainda, o uso da esperança em Salakhutdinov, Mnih e Hinton (2007) está relacionado a métrica *erro médio quadrático* utilizada para a avaliação dos modelos, enquanto utilizaremos métricas que consideram a posição do elemento esperado na lista de recomendação, conforme descrito posteriormente na Seção 6.2.

6 Metodologia Proposta

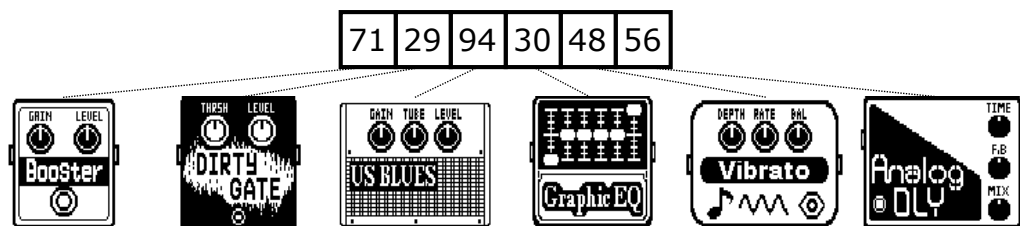
Neste capítulo, são discutidos pontos referentes a metodologia proposta. A Seção 6.1 apresenta o processo de mineração e tratamento dos dados utilizados nos experimentos. Ressalta-se que os dados foram coletados de configurações reais criadas por usuários de uma unidade multiefeitos específica. São discutidos também a categorização de *plugins* e o desbalanceamento da quantidade de *plugins* por categoria e os uso de *plugins* nas diferentes posições da cadeia de efeitos. Em seguida, a Seção 6.2 apresenta as métricas definidas para a avaliação dos modelos nos experimentos, métricas essas comumente aplicadas na avaliação de sistemas de recomendação.

6.1 Base de dados

A base de dados foi criada através da mineração de dados públicos presentes no site *Guitar Patches*¹, um repositório popular de configurações de *plugins* de áudio onde especialistas e amadores compartilham *patches* de algumas das unidades multiefeitos mais conhecidas. Dentre as possíveis unidades de efeitos registradas no site supracitado, *patches* do equipamento Zoom G3 foram coletados pela quantidade de configurações disponível no repositório, pela facilidade de extrair as informações dos arquivos de *patches* e pelo acesso ao equipamento.

Um *patch* para o equipamento Zoom G3 é uma lista ordenada composta por $M = 6$ instâncias de *plugins* de áudio. Existem $R = 117$ possíveis *plugins* de áudio e M^R possíveis *patches*. Quando um *patch* é selecionado no dispositivo de áudio, as transformações de efeito são aplicadas sequencialmente ao sinal de áudio para cada instância presente no *patch*, seguindo a ordem de conexão.

Figura 14 – Exemplo de *patch*. Acima: blocos correspondem às instâncias dos *plugins* de áudio na posição definida no *patch*; números correspondem aos identificadores únicos dos *plugins* de áudio. Abaixo: *Plugins* de áudio correspondentes. Representações gráficas dos *plugins* extraídas de ZOOM (2012).



Fonte: Próprio autor.

¹ Guitar Patches pode ser acessado em <<https://guitarpatches.com/units.php>>.

Na Zoom G3, a conexão entre as instâncias é em forma de cadeia (*chain connection*). Como as transformações são aplicadas em ordem sequencial, i.e., os *plugins* de áudio estão dispostos em série, a ordem dos *plugins* é importante. A Figura 14 apresenta um *patch* presente na base de dados. Os números indicam as instâncias pelos identificadores únicos dos *plugins* (representados graficamente abaixo). As transformações correspondentes a cada instância são aplicadas ao áudio na sequência “da esquerda para a direita”.

6.1.1 Categorização dos *plugins* e mineração dos dados

Os 117 *plugins* foram distribuídos em 13 categorias pelo *software* oficial de edição de *patches* *Zoom Edit & Share*. Nos experimentos realizados, foram consideradas 11 categorias depois de unir categorias com propósitos similares. A Tabela 1 apresenta as categorias e o número de *plugins* por categoria. A relação dos *plugins* de áudio com suas categorias e os identificadores únicos encontra-se no Anexo A. Observamos que um dos *plugins* de áudio disponível é denominado **None**: modeladores de *patches* podem usá-lo se não desejarem uma transformação de sinal em uma posição específica do *patch*.

Tabela 1 – Categorias dos *plugins* de áudio presentes no Zoom G3. São apresentadas ainda a quantidade de *plugins* (coluna 2) e sua ocorrência por categoria (coluna 3).

Categoria	Qtd. <i>plugins</i>	Qtd. instâncias
Amp (Original Amp $\cup$ Amp Modeling)	28	1678
Combination Fx	7	225
Delay	12	1069
Dynamics	7	2451
Filter_EQ	10	1140
Modulation	22	1168
None	1	2131
Pedal Fx	4	248
Reverb	8	1320
SFX	3	40
Stomp (Original Stomp $\cup$ Stomp Modeling)	15	1496

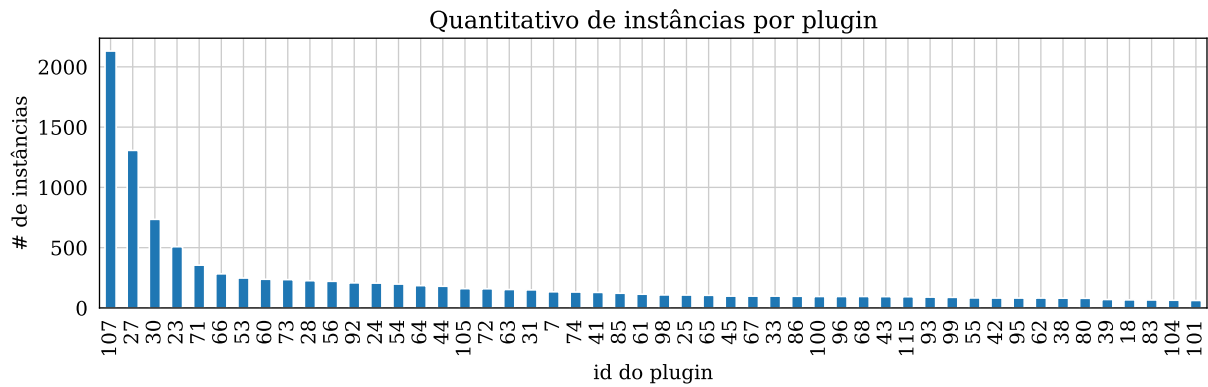
Fonte: Próprio autor.

Foram coletados no sítio 884 registros e, a partir destes, extraídos 5.123 *patches*. O processo de preparação dos dados incluiu: *i*) a remoção de *patches* com mais de três instâncias do *plugin* **None**; *ii*) a padronização do posicionamento dos *plugins* pelo reposicionamento de instâncias **None** para as últimas posições na cadeia de instâncias; *iii*) a remoção de *patches* repetidos; e *iv*) a exclusão de *patches* onde todas instâncias são de um mesmo *plugin*. O conjunto de dados resultante² contém 2.161 *patches* distintos,

² A base de dados e os *scripts* para extração dos *patches* podem ser obtidos em <https://github.com/SrMouraSilva/ZoomG3-Patches>.

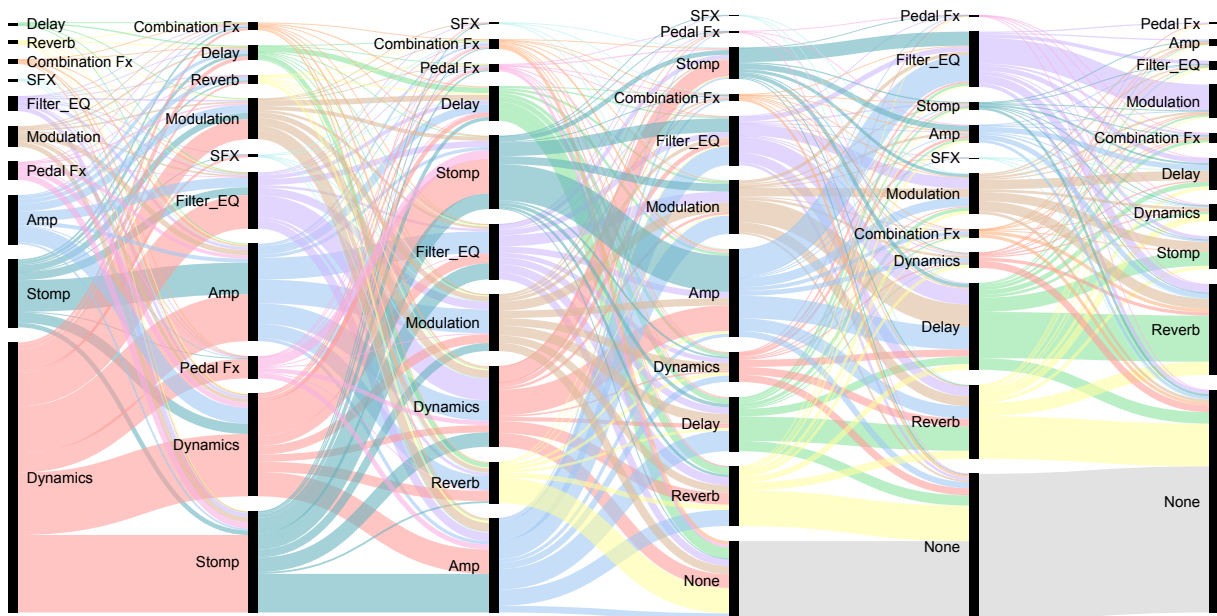
considerando um *patch* totalmente caracterizável por suas instâncias de *plugins* de áudio e a ordem na qual as instâncias se apresentam no *patch*.

Figura 15 – Distribuição da quantidade de instâncias por *plugin* contidas na base de dados. Listado somente *plugins* cuja quantidade de instâncias é superior a 60.



Fonte: Próprio autor.

Figura 16 – Diagrama de Sankey indicando as categorias das instâncias por posição na cadeia de efeitos (*patches*). Cores de arestas iguais para mesmas categorias de origem.



Fonte: Próprio autor.

É digno de nota que o uso do *plugins* de áudio é desbalanceado, os cinco *plugins* de áudio mais utilizados correspondem a 38.84% das instâncias, a saber: None (16.43%), ZNR para redução de ruído (10.08%), Equalizer (5.66%), Compressor (3.91%) e Booster

(2.73%). A Figura 15 apresenta uma melhor noção da quantidade de uso dos *plugins* ao exibir a distribuição dos cinquenta mais utilizados. Perceba que há um decaimento forte, indicando que poucos *plugins* são muito usados e a grande parte é pouco aplicada em *patches*.

Para apresentar a relação das categorias das instâncias com o seu posicionamento no *patch*, organizamos na Figura 16 os dados em um diagrama de Sankey. No diagrama, cada agrupamento de colunas no eixo x representa uma posição no *patch*. O tamanho de uma coluna no eixo y indica a proporção de instâncias de uma mesma categoria em determinada posição. O nome da categoria está do lado da coluna correspondente. Uma aresta liga duas colunas e indica a proporção da coluna da esquerda que é precedida da coluna da direita.

A partir do diagrama de Sankey, nota-se algumas relações entre os posicionamento das categorias e as relações entre categorias de posições vizinhas:

- A categoria **Dynamics** corresponde a cerca de 50% das instâncias na primeira posição;
- O uso de **Dynamics** é amplo, de modo que várias categorias apresentam-se após ela: uma coluna **Dynamics** precede diversas colunas;
- Para as três primeiras posições, há uma tendência de após um **Stomp** haver um **Amp**. O mesmo não ocorre para um **Stomp** na quinta posição;
- Uma proporção significativa das categorias predecessoras de **Amp** são **Dynamics** e **Stomp**;
- Dada a padronização da posição dos **None**, sempre há um **None** após um **None**, exceto no fim da cadeia;
- Um **Reverb**, quando utilizado, tende a ser o último *plugin* da cadeia, ou seja, após um **Reverb** muito provavelmente aparecerá um **None** ou o fim da cadeia.

6.2 Métricas de avaliação

A métrica Acurácia (*accuracy*) contabiliza a média de predições corretas:

$$\text{Acurácia} = \frac{\text{Quantidade de predições corretas}}{\text{Quantidade de predições realizadas}}.$$

Apesar da métrica ser bastante usual em sistemas de classificação, possui deficiências quando há uma grande quantidade de possíveis *plugins* (possíveis classes) e, em casos com desbalanceamento de classes, não fornece informação sobre a ponderação.

Para sistemas de recomendação, é comum usuários buscarem o item desejado dentre uma lista ordenada de recomendações. Espera-se que a lista contenha o item que atenda as necessidades e que esta possua o menor tamanho possível, de forma que não seja necessário buscar bastante o item desejado. Podemos mensurar o problema ao considerar que a

predição é correta se um elemento se encontra entre as k primeiras posições. Essa é a ideia do *Hit Ratio*, representada como $\text{Hit}@k$ ou $\text{recall}@k$ (KARYPIS, 2001):

$$\text{Hit}@k = \frac{\text{Quantidade de itens que são relevantes}@k}{\text{Total de itens relevantes}}.$$

A acurácia é o caso particular em que $k = 1$. Utilizamos na avaliação dos modelos a métrica $\text{Hit}@5$.

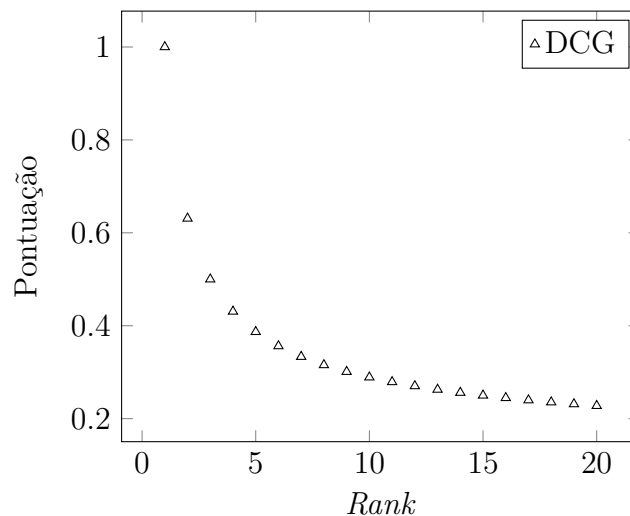
Enquanto a acurácia é uma métrica “rígida” e $\text{Hit}@5$ uma métrica “relaxada”, (*Mean*) *Normalized Discounted Cumulative Gain* (NDCG) considera ainda a lista de recomendações, mas pondera a recomendação pela posição do elemento sugerido, atribuindo uma importância maior aos resultados cujo o item esperado esteja nas primeiras posições (VOORHEES, 2001):

$$\text{NDCG}@k = \frac{1}{Z} \sum_{i=1}^k \frac{2^{\text{rel}(i)} - 1}{\log_2(i + 1)},$$

$$\text{MNDCG}@k = \frac{1}{|\mathbf{U}|} \text{NDCG}@k,$$

onde $\mathbf{U}$ é a matriz de usuários $\times$ itens. A função $\text{rel}(i)$ — de *relevância* — é aplicada neste trabalho como uma função indicadora que retorna 1 caso o i -ésimo elemento seja o item esperado e 0 caso contrário. Z refere-se a constante de normalização, o maior valor possível para NDCG quando o item apresenta um *rank* perfeito. No caso em que no máximo um único item é considerado relevante, a pontuação máxima é 1, de modo que $Z = 1$ e podemos utilizar DCG, a versão não normalizada de NDCG. Veja na Figura 17 que o decaimento torna-se mais suave a medida que o *rank* (posição do item na lista) é incrementado.

Figura 17 – Pontuação atribuída pela métrica DCG (eixo y) para itens em uma lista de recomendações por sua posição (*rank*) (eixo x). Atente-se que para $x = 1$ há sobreposição.



Fonte: Próprio autor.

Buscando amortizar o problema de avaliar negativamente escolhas feitas por gosto, foi aplicada a métrica *Mean Average Precision* (MAP) (ROBERTSON, 2008)

$$\begin{aligned} \text{precision@}k &= \frac{\text{Quantidade de itens relevantes no rank } k}{k}, \\ \text{AP@}k &= \frac{\sum_{i=1}^k \text{precision@}i}{\text{Quantidade de itens relevantes}}, \\ \text{MAP@}k &= \frac{1}{|\mathbf{U}|} \sum_{i=1}^{|\mathbf{U}|} \text{AP@}i. \end{aligned}$$

A métrica MAP foi aplicada para avaliar qual a proporção das $k = 5$ recomendações de *plugins* que estão na mesma categoria do item esperado³. Consideramos itens relevantes em $\text{precision@}k$ os k *plugins* de áudio no topo lista de recomendações com a mesma categoria do item esperado. Um MAP@5 de 50% para um modelo indica que metade das recomendações não são de itens relevantes, i.e, não correspondem ao item esperado e nem a nenhum item de mesma categoria do item esperado.

Resumidamente, as métricas aplicadas para a avaliação do modelo, juntamente com a motivação de uso são

1. **Acurácia:** Item esperado encontra-se na primeira posição;
2. **Hit@5:** Item esperado encontra-se entre as 5 primeiras posições. O número 5 foi arbitrariamente escolhido como o número máximo de testes que um usuário faria com o *plugin* especificado;
3. **MDCG:** Indica o quão próximo o item encontra-se nas primeiras posições;
4. **MAP@5:** Indica a proporção dos 5 itens recomendados que se encontram na mesma categoria do item esperado. Espera dar uma intuição sobre os casos em que o item recomendado não corresponde ao esperado, mas o item era elegível, sendo suficientemente similar, mas não escolhido por questão de gosto.

6.3 Discussão

Um conjunto de dados foi construído pela coleta de configurações desenvolvidas por usuários da unidade multiefeitos Zoom G3. O conjunto construído possui 2.161 cadeias de efeitos distintas para a unidade. Uma cadeia da Zoom G3 possui seis posições nas quais os 117 *plugins* disponíveis no equipamento podem ser aplicados, ocasionando possíveis 117⁶ possíveis combinações, já que o processador multiefeitos permite o uso de um mesmo *plugin* em mais de uma posição no *patch*.

³ É possível que a quantidade de itens da categoria do item esperado seja menor que o k definido (confira a Tabela 1). Para esses casos, k é considerado como o mínimo entre o k definido e a quantidade de itens da categoria.

Conforme indicado na Subseção 6.1.1, há desbalanceamento no uso dos *plugins*, com cerca de 38% das instâncias de *plugins* presentes nos *patches* correspondendo aos cinco *plugins* mais utilizados. No agrupamento dos *plugins* por categoria, percebe-se também um desbalanceamento.

Para a análise dos modelos apresentados nos capítulos anteriores nas tarefas de recomendação, foram descritas quatro métricas de avaliação comumente aplicadas em trabalhos acadêmicos sobre sistemas de recomendação. Nelas, é considerado o posicionamento do item esperado na lista de recomendação para a ponderação da nota de avaliação do sistema. Os detalhes e os resultados desta análise serão apresentados no capítulo seguinte.

7 Experimentos computacionais

Nesta seção, é investigado o desempenho das estratégias de recomendação discutidas no Capítulo 3 para a tarefa de recomendação de *plugins* de áudio. Primeiramente, são apresentados detalhes sobre a configuração dos experimentos na Seção 7.1.

Em seguida, é realizada uma análise dos sistemas de recomendação como ferramentas de suporte na tomada de decisão por atores do processo de produção musical, especificamente nos seguintes cenários:

- Seção 7.2 Recomendação de *plugins* para uma única posição de *patches* considerando os outros *plugins* já selecionados;
- Seção 7.3 Recomendação para mais de uma posição considerando também *plugins* já selecionados;
- Seção 7.4 Definição automática da posição de *plugins* previamente selecionados em um *patch*, e;
- Seção 7.5 Recomendação automática de um *patch* completo.

Ocorrem para cada uma dessas situações discussões sobre vantagens e limitações das estratégias ou de modelos específicos.

Na realização dos experimentos, os seguintes modelos foram considerados para a estratégia de *classificação*: Regressão Logística (LR), Perceptron Multicamadas (MLP) e Máquina de Vetores Suporte para classificação (SVM). Para a abordagem de *filtragem colaborativa*, os modelos *k*-vizinhos mais próximos (kNN) e Máquina de Boltzmann Restrita para Filtragem Colaborativa (RBM-CF) foram avaliados¹. Detalhes sobre os classificadores podem ser obtidos no Capítulo 4. Informações sobre os modelos de filtragem colaborativa podem ser obtidas na Subseção 3.3 e no Capítulo 5.

7.1 Configuração dos experimentos

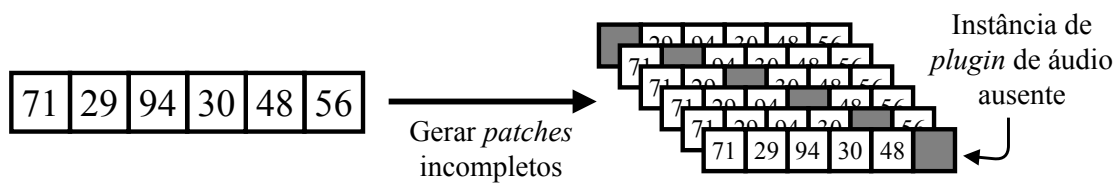
Os sistemas de recomendação de *plugins* de áudio foram avaliados da seguinte forma:

- i*) amostras de treinamento e de teste foram criadas utilizando validação cruzada de 5-*folds* — 4 *folds* para treinamento e a restante para o teste;
- ii*) a seleção do modelo (escolha dos hiperparâmetros do modelo) foi feita por uma validação cruzada interna de 2-*folds* usando as amostras de treinamento;

¹ Código fonte pode ser obtido em <<https://github.com/SrMouraSilva/RBM>>.

- iii) a partir de cada *patch* na base de validação e de teste, foram gerados seis *patches* incompletos, onde cada um corresponde ao *patch* original, mas com uma instância ausente em uma única posição, como ilustrado na Figura 18;
- iv) por fim, foram calculadas as métricas de avaliação para cada método, calculando a média dos cinco conjuntos de dados de teste (gerados pelo processo de validação cruzada). As métricas de avaliação escolhidas estão detalhadas na Seção 6.2.

Figura 18 – Procedimento para gerar 6 *patches* incompletos a partir de um único *patch*. Blocos cinzas são as instâncias de *plugins* de áudio ausentes.



Fonte: Próprio autor.

7.1.1 Seleção do modelo

A Tabela 2 apresenta os hiperparâmetros testados durante a etapa de busca em grade para cada modelo. Note que nenhuma pesquisa foi feita para a Regressão Logística. Atente-se ainda que a busca em grade tem como objetivo selecionar hiperparâmetros com o melhor desempenho médio, i.e., com o melhor resultado para todas as posições ausentes ponderadas igualmente. Naturalmente, em uma aplicação real, a equipe desenvolvedora pode escolher modelos e parâmetros distintos para posições ausentes distintas.

Tabela 2 – Hiperparâmetros testados na seleção do modelo.

Modelo	Hiperparâmetro	Valores
LR	Otimizador	<i>Coordinate Descent</i>
	Seleção da classe	um-contra-todos
MLP	Qtd. camadas ocultas	1
	Qtd. elementos ocultos	20, 40, 80, 100
SVM	C	$2^{-3}, 2^0, \dots, 2^{12}$
	γ	$2^{-13}, 2^{-10}, \dots, 2^2$
	Seleção da classe	um-contra-todos
kNN	Qtd. vizinhos	1, 5, 10, 15, 20, 25, 40, 60, 80, 100
	Métrica de distância	Distância <i>Hamming</i> (ZHANG et al., 2010)
RBM-CF	CD	1
	Qtd. elementos ocultos	10.000
	Tamanho do mini-batch	10
	Taxa de aprendizagem	Adam(1r=0.05)
	Qtd. épocas	Definido por parada prematura

Fonte: Próprio autor.

Os hiperparâmetros utilizados para a RBM-CF foram definidos em experimentos preliminares. Com a base dividida em uma combinação de treino e validação, foram testadas diferentes combinações de tamanho da camada oculta, de passo de aprendizagem (constante, com decaimento e adaptativa: ADAGRAD, AdaMax e Adam), de amostragem (CD- k e PCD- k), de tamanho do mini-batch, de quantidade de épocas no treinamento, de regularização (L1 e L2) e de valores para *momentum*.

Dada uma alta demanda de tempo para a realização da busca em grade com os diversos hiperparâmetros do modelo, foi verificada a possibilidade de utilizar um otimizador Bayesiano para definição dos valores dos hiperparâmetros. Entretanto, o custo apresentou-se ainda proibitivo. Desta forma, no lugar da validação cruzada interna de *2-folds*, foram definidos valores constantes para os hiperparâmetros, exceto a quantidade de épocas de treinamento, definida por parada prematura, separando 20% da base de treino para validação.

7.2 Recomendação de *plugins* para uma única posição

Nosso objetivo é investigar se os modelos avaliados foram capazes de aprender com os *patches* de treinamento disponíveis e se eles foram capazes de generalizar para dados não vistos. Pelo descrito na Tabela 3, todos os modelos conseguiram obter uma acurácia entre 30% e 40%. Como há uma grande quantidade de valores possíveis para v_h e o conjunto de dados utilizado no experimento é escasso (2.161 exemplos $\ll 117^6$), era esperado que os sistemas de recomendação não alcançassem taxas de acurácia muito altas. Outra consequência é o alto desvio padrão. De fato, algumas separações de treinamento e teste levam a situações em que algumas classes encontram-se presentes apenas no subconjunto de teste.

Tabela 3 – Resumo dos experimentos (médias e correspondentes desvios padrão) para recomendação em uma única posição. Melhores resultados em negrito.

Modelo	Codificação	Acurácia (%)	Hit@5 (%)	MDCG	MAP@5 (%)
LR	Vetor <i>one-hot</i>	38,99 $\pm$ 12,81	59,35 $\pm$ 11,38	0,5911 $\pm$ 0,0970	32,54 $\pm$ 15,36
	<i>Bag-of-words</i>	34,27 $\pm$ 13,71	51,52 $\pm$ 14,05	0,5441 $\pm$ 0,1094	31,78 $\pm$ 16,07
MLP	Vetor <i>one-hot</i>	34,74 $\pm$ 12,72	55,24 $\pm$ 11,12	0,5567 $\pm$ 0,0965	30,71 $\pm$ 15,17
	<i>Bag-of-words</i>	35,22 $\pm$ 13,06	56,07 $\pm$ 11,75	0,5640 $\pm$ 0,0996	31,98 $\pm$ 14,99
SVM	Vetor <i>one-hot</i>	39,48 $\pm$ 12,22	58,56 $\pm$ 12,03	0,5849 $\pm$ 0,0997	33,43 $\pm$ 15,65
	<i>Bag-of-words</i>	38,10 $\pm$ 13,08	58,28 $\pm$ 12,07	0,5821 $\pm$ 0,1016	33,37 $\pm$ 15,57
kNN	Plugin ids	35,61 $\pm$ 13,11	52,52 $\pm$ 12,82	0,5448 $\pm$ 0,1037	30,39 $\pm$ 15,58
	<i>Bag-of-words</i>	34,27 $\pm$ 13,58	52,10 $\pm$ 13,15	0,5396 $\pm$ 0,1076	30,03 $\pm$ 15,65
RBM-CF	Vetor <i>one-hot</i>	30,90 $\pm$ 13,80	50,42 $\pm$ 13,48	0,5262 $\pm$ 0,1086	31,01 $\pm$ 15,32

Fonte: Próprio autor.

Em sistemas de recomendação, é comum que o usuário verifique mais de um elemento na lista de recomendações antes de escolher um elemento. Em outras palavras, mesmo que o elemento desejado não seja a primeira recomendação, espera-se que esteja dentre as primeiras posições. Assim, quando a métrica Hit@5 é considerada, os modelos alcançam pontuações próximas a 60%. A métrica MDCG — que indica qual dos modelos tem o item esperado mais próximo da primeira posição — assinala que o modelo Regressão Logística apresenta os melhores resultados.

A métrica MAP@5 indica que um terço dos cinco itens mais prováveis são da mesma categoria que o item esperado. Um valor semelhante também é obtido se recomendarmos os *plugins* mais prováveis por posição. Isso pode indicar que categorização utilizada — com base nas definições extraídas do manual da unidade de efeitos — é insuficiente para definir como os *plugins* são semelhantes. Em futuras investigações, pretendemos avaliar taxonomias alternativas de *plugins* de áudio (VERFAILLE; GUASTAVINO; TRAUBE, 2006).

A abordagem da codificação da entrada dos modelos como *one-hot* permite que considerem a posição de cada *plugin* de áudio no *patch*. Como visto na Tabela 3, a maioria dos modelos obteve melhores resultados ao considerar a codificação *one-hot* em vez da codificação *bag-of-words*, com MLP sendo a única exceção.

Note que os modelos não trabalham com informações sobre a ordem, apesar do *one-hot* trazer informação sobre posicionamento. Por exemplo, modelos não poderão inferir padrões como “**amplificador** sempre vem depois de **compressor**”. Neste exemplo, caso o **amplificador** apareça somente nas três primeiras posições dos *patches* da base de treino, quando for solicitado uma recomendação para a quarta posição, os modelos não sugerirão um **amplificador**, mesmo que **compressor** apareça na terceira posição do *patch*.

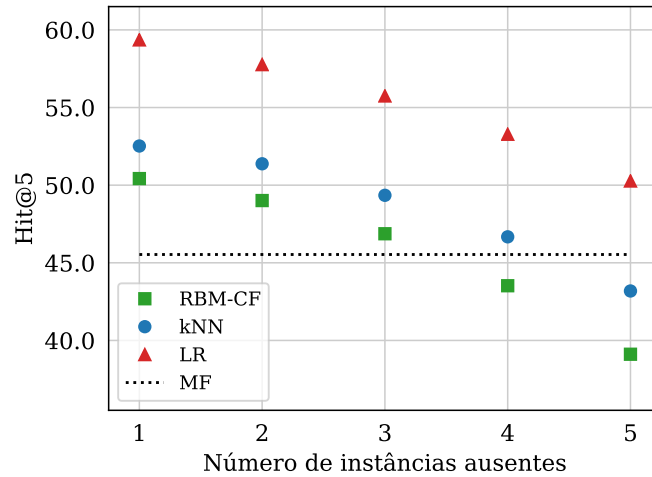
7.3 Recomendação de *plugins* para múltiplas posições

Quando existe uma única posição com um *plugin* desconhecido, os resultados dos classificadores se destacam sobre os de abordagem de filtragem colaborativa. No entanto, em cenários em que é necessário considerar mais de um *plugin* de áudio ausente ou em problemas com M maior, é necessário treinar um modelo distinto para cada combinação de posições com *plugins* ausentes. Assim, o número de modelos a serem treinados aumenta exponencialmente: São necessários $\sum_{i=0}^M i \binom{M}{i}$ modelos distintos para um *patch* com M instâncias.

Tal problema não se aplica aos modelos kNN e RBM-CF, em que é necessário apenas um único modelo ser treinado. A Figura 19 mostra os valores da métrica Hit@5 para os modelos LR, kNN e RBM-CF nos casos com um ou mais *plugins* ausentes. Hit@5 decai para todos os modelos à medida que aumenta o número de posições com *plugins*

desconhecidos. LR apresenta melhores resultados, no entanto, como um exemplo da abordagem de classificação, não é escalável para grandes quantidades de *plugins* e/ou recomendações.

Figura 19 – Métrica Hit@5 para os modelos LR, kNN e RBM-CF na tarefa de recomendação de *plugins* para várias posições em um mesmo *patch*. A linha tracejada indica o valor Hit@5 quando o *plugin* mais frequente é usado.



Fonte: Próprio autor.

A linha pontilhada na Figura 19 indica a recomendação dos cinco *plugins* de áudio mais frequentes (MF) de cada posição. O alto valor apresentado para o patamar MF é decorrente do desbalanceamento da frequência de uso dos *plugins*, conforme discutido na Subseção 6.1.1.

7.4 Posicionamento das instâncias de *plugins* em um *patch* com a RBM-CF

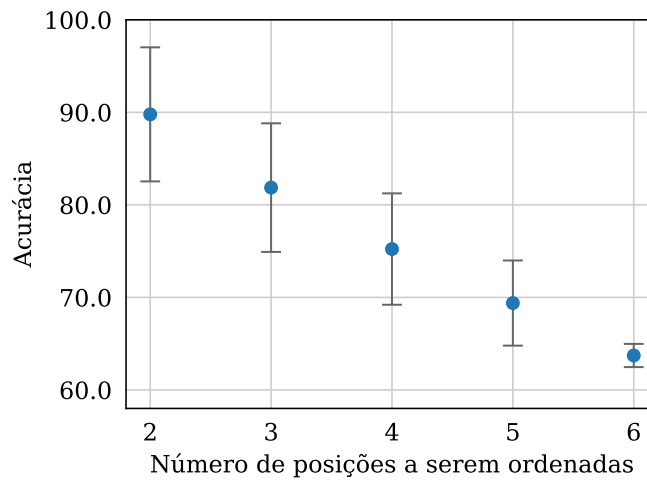
Uma variação da tarefa de recomendação de *plugins* é o problema de posicionamento de instâncias de *plugins*. Ele consiste na busca do melhor posicionamento de instâncias previamente selecionadas pelo usuário em um *patch*. Nesta nova tarefa, o usuário escolhe os *plugins* e o sistema recomenda a ordem em que estes devem se apresentar na cadeia de efeitos. Isto é útil, por exemplo, quando não se sabe a ordem mais adequada para os *plugins*. Ressaltamos que, como *plugins* de áudio aplicam filtros não lineares ao sinal de áudio, a ordem destes no *patch* é relevante.

O modelo RBM-CF pode ser usado para encontrar o melhor posicionamento de instâncias pela seleção da permutação que corresponde a menor energia livre dada

pela Equação (5.5), que é inversamente proporcional à probabilidade marginal $P(\mathbf{v})$ das instâncias de *plugins* de áudio.

A Figura 20 apresenta a capacidade do modelo de recomendar a posição correta para casos de duas a seis instâncias sem uma posição definida. A métrica acurácia, relatada na figura, considera quantas posições foram corretamente indicadas pelo modelo. Observe que uma taxa de acertos em torno de 66% indica que, em média, há apenas uma única permutação errada entre dois *plugins*.

Figura 20 – Proporção de instâncias posicionadas corretamente pela RBM-CF pelo número de entradas/componentes ausentes.



Fonte: Próprio autor.

Esta estratégia não é adequada para problemas com M grande, pois envolve o teste com todas as permutações das instâncias. Uma alternativa seria a adaptação das estratégias utilizadas nas seções anteriores para a recomendação de *plugins*, tendo em vista que a tarefa de posicionamento é equivalente a recomendação de uma parte de um *patch* ($\mathbf{v}_{\text{unknown}} \subseteq \mathbf{v}$) a partir de *plugins* pré-selecionados, adicionada das seguintes restrições: *i*) a quantidade de *plugins* é limitada pelo tamanho de $\mathbf{v}_{\text{unknown}}$ e *ii*) deve-se utilizar somente uma única instância por *plugin*². Esta abordagem não foi tratada nesta dissertação.

7.5 Recomendação de um *patch* completo com a RBM-CF

Podemos facilmente usar o modelo RBM-CF, que segue uma abordagem generativa, para o cenário em que é desejado recomendar todos os seis *plugins* em uma cadeia de

² É fácil estender para os casos com $n + 1$ instâncias de um mesmo *plugin* ao gerar n réplicas do *plugin*.

efeitos. Primeiro, inicializamos a camada visível $\mathbf{v}$ com *plugins* de áudio uniformemente aleatórios. Em seguida, executamos uma cadeia de Gibbs 10.000 vezes³.

Foram gerados *patches* completos, repetindo o processo de amostragem pela cadeia de Gibbs até serem obtidas 14 recomendações válidas. Recomendações inválidas geradas continham para alguma posição uma representação que não respeita a descrita na Subseção 3.3.2: o modelo pode eventualmente recomendar nenhum *plugin* para determinada posição ($\mathbf{v}_m = [0, \dots, 0]^T$), o que geralmente ocorre quando a probabilidade conjunta é bem distribuída entre todos os M elementos. Um *patch* inválido pode tornar-se válido realizando mais algumas poucas cadeias de Gibbs.

Nenhum dos 14 novos *patches* gerados estavam presentes no conjunto de dados. Os quatorze são descritos e comentados a seguir. Para facilitar a leitura, *plugins* None no final da cadeia foram omitidos.

1. RackComp $\rightarrow$ Cho+Dly $\rightarrow$ Arena

Simulação limpa. Combinável com instrumentos acústicos ou com sinais já distorcidos. Destaque para a escolha de um *chorus* e *delay* (Cho+Dly) seguida de um *reverb* (Arena);

2. T Scream $\rightarrow$ GraphicEQ $\rightarrow$ ZNR

Um simples *patch* de *overdrive* (T Scream) com equalizador para filtrar frequências específicas (GraphicEQ) e com um efeito para redução de ruído (ZNR);

3. ZNR $\rightarrow$ Aco.Sim $\rightarrow$ GraphicEQ $\rightarrow$ FilterDly

Um *patch* para simulação de violões acústicos. O efeito para redução de ruído (ZNR) é possivelmente desnecessário caso o sinal não venha com distorção provindo do uso em conjunto com outro equipamento;

4. ZNR $\rightarrow$ VintageCE $\rightarrow$ Comp $\rightarrow$ FilterDly

Patch sem distorções. Uso de ZNR similar ao caso anterior;

5. Comp $\rightarrow$ ReverseDL $\rightarrow$ VintageCE $\rightarrow$ Hall

Patch com diversos tipos de modulação;

6. RackComp $\rightarrow$ PDL MnPit $\rightarrow$ FilterDly $\rightarrow$ Arena

Pitch shifter (PDL MnPit) na segunda posição. No conjunto de dados, PDL MnPit sempre aparece com alguma distorção nessa posição, o que não ocorreu. A adição ou a substituição de PDL MnPit por um pedal de distorção seria interessante;

7. ZNR $\rightarrow$ Comp $\rightarrow$ TapeEcho $\rightarrow$ GraphicEQ $\rightarrow$ DynaDelay $\rightarrow$ Booster

Patch sem distorção e com diversos tipos de modulação;

³ A amostragem de Gibbs por meio da execução de cadeias de Gibbs busca obter observações que correspondam com a distribuição das variáveis aleatórias obtidas pelo modelo. Amostrar todas as variáveis visíveis do modelo significa recomendar *patches* completos.

8. **Comp → ZNR → ZNR → BG CRUNCH**
Compressor (**Comp**) seguido de dois redutores de ruído e um simulador de amplificador (**BG CRUNCH**). Posicionamento adequado dos *plugins*;
9. **ZNR → M Comp → GraphicEQ → ZNR → FD COMBO → Arena**
Duplicação desnecessária de ZNRs. Foi verificado que existem somente 5 *patches* com ZNR na quinta posição no conjunto de dados. Os outros *plugins* recomendados estão em posições consistentes;
10. **Comp → ZNR → ZNR → Air → Booster**
Outro caso com dois ZNRs. A substituição de ZNR por um amplificador ou distorção seria um incremento interessante. O posicionamento dos outros *plugins* está em conformidade com o conjunto de dados;
11. **PedalCry → Comp+AWah → StereoDly → StereoDly → ZNR → Delay**
Combinação de pedal de *wah* (**PedalCry**) com *auto-wah* (**Comp+AWah**), dois *delays* iguais (**StereoDly**) e mais um *delay* no final da cadeia (**Delay**).
É possível que um usuário utilize instâncias de *plugins* específicas em momentos específicos. Por exemplo, cada um dos três *delays* pode possuir uma configuração distinta para uma ou mais músicas.
Foram constatados no conjunto de dados somente dois casos com ZNR na quinta posição;
12. **Comp → Z Clean → NoiseGate → VintageCE → StereoCho → Delay**
Patch com simulador de amplificador limpo e (**Z Clean**) dois *chorus* distintos (**VintageCE**, **StereoCho**). Uso pode depender do momento, tal como descrito no caso anterior;
13. **ZNR → Comp → VinFLNGR → StereoCho → ModDelay → AnalogDly**
Caso similar aos dois itens anteriores com dois *delays*;
14. **ZNR → Comp → StereoCho → ModDelay → StereoCho → TiledRoom**
Dois **StereoCho** aparentemente desnecessários.

Perceba que a organização dos *plugins* de áudio se assemelha com o apresentado na Figura 16, que trata sobre o posicionamento das categorias dos *plugins* nos *patches* presentes na base de dados. Informações sobre a categorização dos *plugins* podem ser obtidas no Anexo A e a descrição dos *plugins* e seus parâmetros no manual da unidade multiefeitos (ZOOM, 2012).

É interessante notar que são gerados *patches* com nenhum e com muitas instâncias de **None**. Em outras palavras, o modelo recomendou⁴ desde cadeias pequenas (somente três efeitos) a cadeias grandes (todos os seis efeitos).

O principal problema detectado em algumas das amostras é a repetição de *plugins* em um mesmo *patch* aparentemente sem justificativas, como são os casos com mais de um ZNR em um *patch*. Entretanto, isso pode ser decorrência do processo de geração de *patches* definido.

Primeiro, note que o número de execuções das cadeias de Gibbs não leva em consideração se as amostras geradas no fim do processo apresentam uma alta probabilidade, levando a suspeita de um número insuficiente de passos de Gibbs. Segundo, a natureza probabilística pode ter adicionado ruídos na amostragem, de forma similar ao que ocorre na geração de recomendações inválidas. Dado a quantidade de *plugins*, um ruído é bem mais perceptível.

Uma sugestão para o processo de amostragem seria interrompê-lo quando as amostras apresentassem uma probabilidade marginal adequada. Uma probabilidade adequada poderia ser definida com uma relação entre as probabilidades de *patches* previamente definidos por usuários do sistema de recomendação e de *patches* com pouca qualidade.

Destaca-se aqui, entretanto, que a metodologia de geração de *patches* aplicada nesta seção não é invalidada pelos pontos supracitados. Foi decidido aplicar tal metodologia para *i*) destacar a simplicidade de utilizar um modelo RBM-CF já treinado, capaz também de realizar outras tarefas relativas à recomendação de *plugins* apresentadas nas seções anteriores e *ii*) criar um espaço de discussão sobre métodos de amostragem de *patches* completos.

7.6 Discussão

Neste capítulo, os modelos apresentados nos Capítulos 3, 4 e 5 foram avaliados por métricas comumente aplicáveis em sistemas de recomendação (Seção 6.2) nem tarefas envolvendo a recomendação de *plugins* de áudio sobre o conjunto de dados preparado (Seção 6.1). Os casos testados seguiram situações aplicáveis nos contextos de produção e execução musical (Capítulo 2 Capítulos 3).

Classificadores apresentaram melhores valores das métricas para a recomendação de um ou mais *plugins* de áudio em *patches* incompletos, entretanto com o custo de não ser escalável para situações em que existem muitas posições a serem recomendadas, tendo em vista a necessidade de treinar um modelo distinto para cada caso.

Na estratégia de recomendação por filtragem colaborativa, os modelos apresentaram

⁴ É possível considerar tamanhos distintos de cadeias de efeito ao ignorar *plugins* **None**.

a vantagem de treinar um único modelo para as diversas situações. Além disso, a capacidade generativa de um dos modelos testados, o RBM-CF, permitiu a resolução das tarefas de posicionamento dos *plugins* de áudio e recomendação de um *patch* completo, ou seja, dado nenhuma informação do usuário no sistema. O primeiro apresenta bons resultados: a taxa de acerto indica que o modelo erra, em média, somente uma permutação de *plugins* nos *patches*. Para o segundo, o posicionamento dos *plugins* nos *patches* gerados pelo modelo segue um posicionamento similar ao discutido na Subseção 6.1.1. Nas subseções também existem discussões para adaptações futuras dos modelos em variações dos experimentos.

8 Conclusão

Atores do processo de produção musical operam diversas ferramentas para obter artefatos musicais de alta qualidade. Dentre elas, processadores digitais multiefeitos são utilizados para a aplicação de transformações não lineares sobre sinais de áudio. As transformações, i.e., os efeitos de áudio, são implementados na forma de *plugins* de áudio.

Dado o suporte a uma grande quantidade de *plugins* de áudio em processadores digitais multiefeitos e, conseqüentemente, um grande número de configurações disponíveis, um uso melhor de tais equipamentos é restrito a especialistas. Desse modo, sistemas de recomendação de *plugins* possuem o potencial de tornar o processo de produção mais eficiente, rápido e interativo, como também servir de instrumento para auxílio na aprendizagem para atores do processo de produção musical.

Neste trabalho, duas metodologias foram aplicadas para o problema de recomendação de *plugins* de áudio: uma abordagem de classificação e uma abordagem de filtragem colaborativa. Métodos de aprendizagem de máquina clássicos foram então aplicados a um novo conjunto de dados construído a partir de um repositório de configurações disponível na internet para uma unidade de efeitos de guitarra específica.

Enquanto outros trabalhos da literatura focam a definição de valores de parâmetros de um determinado conjunto de *plugins*, este trabalho teve como objetivo automatizar o procedimento de recomendação de *plugins* de áudio via técnicas de aprendizagem de máquina. Os modelos foram avaliados usando métricas frequentemente utilizadas para análise de sistemas de recomendação.

Os experimentos realizados indicam que os classificadores utilizados apresentam melhores resultados do que os modelos de filtragem colaborativa aplicados na recomendação de *plugins* para uma única posição. Para várias posições, o número de modelos de classificadores aumenta exponencialmente, de modo que o uso de modelos de filtragem colaborativa se sobressai pela necessidade do treino de somente um modelo. A RBM-CF, a abordagem generativa avaliada, também pode recomendar *pitches* completos e sugerir a melhor posição dos *plugins* pré-selecionados pelo usuário.

O problema de recomendar *plugins* pode ser traduzido em outros problemas em diferentes contextos. Por exemplo, a codificação de vetor *one-hot* M -dimensional pode ser usada para representar dados de contagem esparsa, como palavras em um documento (HINTON; SALAKHUTDINOV, 2009). Assim, a metodologia apresentada nesta dissertação pode se adaptar a problemas onde é desejado selecionar itens que mostram sinergia entre eles. Para casos onde a ordem não contém informação, existe também a possibilidade de usar *bag-of-words*. Exemplos de aplicações para as tais codificações são jogos baseados em cartas

(como *Magic the gathering* (WARD; COWLING, 2009) e *Clash Royale* (LEHTONEN; HARVIAINEN, 2016)), jogos de RPG (como a seleção de times em *Pokemon* (IHARA et al., 2018)) ou relacionamento de documentos com base no seu texto.

Em trabalhos futuros, sugerimos a discussão de tópicos como:

1. a incorporação de informações externas ao problema, como informações demográficas e categorias dos *plugins*. Teste de outras formas de categorização baseadas nas taxonomias apresentadas por Verfaillie, Guastavino e Traube (2006);
2. a investigação de comportamento como sobreajuste e generalização em modelos de aprendizagem profunda treinados em bases de dados pequenas, como a apresentada neste documento;
3. o uso da representação latente da RBM-CF para codificação dos dados, tanto para análise de padrões via projeção dos dados quanto para usar como entrada em modelos supervisionados;
4. a análise de outros modelos de filtragem colaborativa, como o PopItem, o KNNItem e o Neural Collaborative Filtering (HE et al., 2017) na recomendação de *plugins* de áudio;
5. a integração ou desenvolvimento de softwares para a aplicação e aprendizagem online do sistema de recomendação baseados em softwares já estabelecidos como ToneLib (2019) ou bibliotecas que oferecem fácil extensão tal como da SILVA, Costa e Jucá (2017).

Referências

- ABDOLLAHI, B.; NASRAOUI, O. Explainable restricted boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1606.07129*, 2016. Disponível em: <<https://arxiv.org/pdf/1606.07129.pdf>>. Citado na página 72.
- AUGUSTO, J. A.; CASTRO, G. A. S. d. A produção musical e o processo de musicalização por meio da manipulação de softwares de gravação. *Formação@ Docente*, v. 6, n. 2, p. 5–a, 2015. Disponível em: <<http://dx.doi.org/10.15601/2237-0587/fd.v6n2p5-21>>. Citado na página 29.
- BASU, C.; HIRSH, H.; COHEN, W. Recommendation as classification: Using social and content-based information in recommendation. In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. USA: American Association for Artificial Intelligence, 1998. (AAAI '98/IAAI '98), p. 714–720. ISBN 0262510987. Disponível em: <<https://doi.org/10.5555/295240.295795>>. Citado na página 46.
- BENGIO, Y.; DELALLEAU, O. Justifying and generalizing contrastive divergence. *Neural Computation*, v. 21, n. 6, p. 1601–1621, 2009. PMID: 19018704. Disponível em: <<https://doi.org/10.1162/neco.2008.11-07-647>>. Citado 2 vezes nas páginas 64 e 67.
- BENNETT, J.; LANNING, S. The netflix prize. In: *Proceedings of KDD Cup and Workshop 2007*. San Jose, California: ACM, 2007. v. 2007, p. 35. Citado na página 68.
- BERKSON, J. M. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, Taylor & Francis, v. 39, n. 227, p. 357–365, 1944. Disponível em: <<https://doi.org/10.1080/01621459.1944.10500699>>. Citado na página 52.
- BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738. Citado 2 vezes nas páginas 55 e 56.
- BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. New York, NY, USA: ACM, 1992. (COLT '92), p. 144–152. ISBN 0-89791-497-X. Disponível em: <<http://doi.acm.org/10.1145/130385.130401>>. Citado na página 55.
- BOUZERDOUM, A.; CHAI, D.; BOUSSAID, F. Analog signal processing. *Electrical Engineering-Volume I*, EOLSS Publications, p. 298, 2009. Citado na página 37.
- BOYD, S.; VANDENBERGHE, L. *Convex optimization*. Cambridge university press, 2004. ISBN 978-0-521-83378-3. Disponível em: <https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf>. Citado 3 vezes nas páginas 19, 61 e 66.
- BREULEUX, O.; BENGIO, Y.; VINCENT, P. Quickly generating representative samples from an rbm-derived process. *Neural computation*, MIT Press, Cambridge, MA, USA, v. 23, n. 8, p. 2058–2073, aug 2011. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/NECO_a_00158>. Citado 2 vezes nas páginas 30 e 58.

- CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, v. 2, p. 27:1–27:27, 2011. Software available at <<http://www.csie.ntu.edu.tw/~cjlin/libsvm>>. Citado na página 56.
- CHOURDAKIS, E. T.; REISS, J. D. A machine-learning approach to application of intelligent artificial reverberation. *J. Audio Eng. Soc*, v. 65, n. 1/2, p. 56–65, 2017. Disponível em: <<http://www.aes.org/e-lib/browse.cfm?elib=18543>>. Citado 2 vezes nas páginas 30 e 41.
- CÔTÉ, M.-A.; LAROCHELLE, H. An infinite restricted boltzmann machine. *Neural computation*, MIT Press, v. 28, n. 7, p. 1265–1288, 2016. PMID: 27171012. Disponível em: <https://doi.org/10.1162/NECO_a_00848>. Citado 2 vezes nas páginas 61 e 67.
- da SILVA, P. M. M.; COSTA, L. d. S.; JUCÁ, S. C. Pedal Pi — Multi-processador de plugins de áudio DIY. In: *III Escola regional de informática do Piauí*. Picos, Piauí: SBC, 2017. p. 23–28. ISBN 978-85-7669-395-6. Disponível em: <<https://bit.ly/2jXIL6B>>. Citado na página 94.
- DESJARDINS, G.; BENGIO, Y.; COURVILLE, A. C. On tracking the partition function. In: SHAWE-TAYLOR, J. et al. (Ed.). *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011. p. 2501–2509. Disponível em: <<http://papers.nips.cc/paper/4351-on-tracking-the-partition-function.pdf>>. Citado na página 68.
- DIAS, M. L. D. *Novas técnicas baseadas em metaheurísticas para o treinamento de máquinas de vetores suporte*. Dissertação (Mestrado) — Instituto Federal do Ceará, 2017. Citado na página 55.
- ESLAMI, S. M. A. et al. The shape boltzmann machine: A strong model of object shape. *International Journal of Computer Vision*, Springer, v. 107, n. 2, p. 155–176, Apr 2014. ISSN 1573-1405. Disponível em: <<https://doi.org/10.1007/s11263-013-0669-1>>. Citado na página 58.
- FISCHER, A.; IGEL, C. An introduction to restricted boltzmann machines. In: ALVAREZ, L. et al. (Ed.). *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 14–36. ISBN 978-3-642-33275-3. Disponível em: <https://doi.org/10.1007/978-3-642-33275-3_2>. Citado na página 61.
- FREUND, Y.; HAUSSLER, D. Unsupervised learning of distributions on binary vectors using two layer networks. In: MOODY, J. E.; HANSON, S. J.; LIPPMANN, R. P. (Ed.). *Advances in Neural Information Processing Systems 4*. Morgan-Kaufmann, 1992. p. 912–919. Disponível em: <<http://papers.nips.cc/paper/535-unsupervised-learning-of-distributions-on-binary-vectors-using-two-layer-networks.pdf>>. Citado na página 57.
- GEORGIEV, K.; NAKOV, P. A non-iid framework for collaborative filtering with restricted boltzmann machines. In: DASGUPTA, S.; MCALLESTER, D. (Ed.). *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, Georgia, USA: PMLR, 2013. (Proceedings of Machine Learning Research, 3), p. 1148–1156. Disponível em: <<http://proceedings.mlr.press/v28/georgiev13.html>>. Citado 2 vezes nas páginas 30 e 71.

GIBSON, D. *The Mixing Engineer's Handbook, Second Edition*. Thomson Course Technology, 2006. ISBN 9781598637687. Disponível em: <<https://books.google.com.br/books?id=AeoLAAAAQBAJ>>. Citado na página 38.

GOH, G. Why momentum really works. *Distill*, 2017. Disponível em: <<http://distill.pub/2017/momentum>>. Citado na página 66.

GOLDBERG, D. et al. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, Association for Computing Machinery, Inc., v. 35, n. 12, p. 61–71, 1992. Citado na página 47.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Disponível em: <<http://www.deeplearningbook.org>>. Citado 5 vezes nas páginas 19, 63, 64, 65 e 68.

GOUDARD, V.; MULLER, R. Real-time audio plugin architectures. *Comparative study. IRCAM-Centre Pompidou. France*, 2003. Disponível em: <<https://www.researchgate.net/publication/265307102>>. Citado na página 40.

HAYKIN, S. S. *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009. ISBN 9780133002553. Citado na página 53.

HE, X. et al. Neural collaborative filtering. In: *Proceedings of the 26th International Conference on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017. (WWW '17), p. 173–182. ISBN 9781450349130. Disponível em: <<https://doi.org/10.1145/3038912.3052569>>. Citado na página 94.

HINTON, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, MIT Press, v. 14, n. 8, p. 1771–1800, 2002. Disponível em: <<https://doi.org/10.1162/089976602760128018>>. Citado 4 vezes nas páginas 30, 57, 58 e 64.

HINTON, G. E. To recognize shapes, first learn to generate images. In: CISEK, P.; DREW, T.; KALASKA, J. F. (Ed.). *Computational Neuroscience: Theoretical Insights into Brain Function*. Elsevier, 2007, (Progress in Brain Research, v. 165). p. 535–547. Disponível em: <[https://doi.org/10.1016/S0079-6123\(06\)65034-6](https://doi.org/10.1016/S0079-6123(06)65034-6)>. Citado 2 vezes nas páginas 30 e 58.

HINTON, G. E. A practical guide to training restricted boltzmann machines. In: _____. *Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 599–619. ISBN 978-3-642-35289-8. Disponível em: <https://doi.org/10.1007/978-3-642-35289-8_32>. Citado na página 70.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, MIT Press, Cambridge, MA, USA, v. 18, n. 7, p. 1527–1554, jul. 2006. ISSN 0899-7667. Disponível em: <<https://doi.org/10.1162/neco.2006.18.7.1527>>. Citado na página 71.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006. ISSN 0036-8075. Disponível em: <<https://science.sciencemag.org/content/313/5786/504>>. Citado na página 58.

- HINTON, G. E.; SALAKHUTDINOV, R. R. Replicated softmax: an undirected topic model. In: BENGIO, Y. et al. (Ed.). *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., 2009. p. 1607–1614. Disponível em: <<http://papers.nips.cc/paper/3856-replicated-softmax-an-undirected-topic-model.pdf>>. Citado 2 vezes nas páginas 73 e 93.
- HINTON, G. E.; SEJNOWSKI, T. J. Learning and relearning in boltzmann machines. In: _____. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986. p. 282–317. ISBN 026268053X. Citado na página 109.
- HU, Y.; KOREN, Y.; VOLINSKY, C. Collaborative filtering for implicit feedback datasets. In: IEEE. *2008 Eighth IEEE International Conference on Data Mining*. 2008. p. 263–272. ISSN 1550-4786. Disponível em: <<https://doi.org/10.1109/ICDM.2008.22>>. Citado na página 47.
- IHARA, H. et al. Implementation and evaluation of information set monte carlo tree search for pokémon. In: IEEE. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2018. p. 2182–2187. ISSN 2577-1655. Disponível em: <<https://doi.org/10.1109/SMC.2018.00375>>. Citado na página 94.
- IZHAKI, R. *Mixing audio: concepts, practices and tools*. Taylor & Francis, 2013. ISBN 978-0-240-52222-7. Disponível em: <<https://doi.org/10.4324/9780080556154>>. Citado na página 29.
- JOHNSON, R.; ZHANG, T. Effective use of word order for text categorization with convolutional neural networks. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015. p. 103–112. Disponível em: <<https://www.aclweb.org/anthology/N15-1011>>. Citado na página 47.
- KADIS, J. *The science of sound recording*. Boston: Focal Press, 2012. ISBN 978-0-240-82154-2. Citado na página 38.
- KARYPIS, G. Evaluation of item-based top-n recommendation algorithms. In: *Proceedings of the Tenth International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2001. (CIKM '01), p. 247–254. ISBN 1-58113-436-3. Disponível em: <<http://doi.acm.org/10.1145/502585.502627>>. Citado na página 79.
- KOREN, Y.; BELL, R. Advances in collaborative filtering. In: _____. *Recommender Systems Handbook*. Boston, MA: Springer US, 2015. p. 77–118. ISBN 978-1-4899-7637-6. Disponível em: <https://doi.org/10.1007/978-1-4899-7637-6_3>. Citado na página 47.
- LAROCHELLE, H.; BENGIO, Y. Classification using discriminative restricted boltzmann machines. In: ACM. *Proceedings of the 25th International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2008. (ICML '08), p. 536–543. ISBN 9781605582054. Disponível em: <<https://doi.org/10.1145/1390156.1390224>>. Citado 2 vezes nas páginas 30 e 58.
- LAROCHELLE, H. et al. An empirical evaluation of deep architectures on problems with many factors of variation. In: ACM. *Proceedings of the 24th International*

- Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2007. (ICML '07), p. 473–480. ISBN 9781595937933. Disponível em: <<https://doi.org/10.1145/1273496.1273556>>. Citado na página 64.
- LEHTONEN, M. J.; HARVIAINEN, J. T. Mobile games and player communities: Designing for and with clans. *Design Management Review*, Wiley Online Library, v. 27, n. 3, p. 20–26, 2016. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1111/drev.12033>>. Citado na página 94.
- LEVITAN, D. *From Demo Tape to Record Deal*. [S.l.]: Alfred Music Publishing, 1992. ISBN 9781457439469. Citado na página 29.
- LI, C.; LI, J. Collaborative filtering based on dual conditional restricted boltzmann machines. In: IEEE. *2017 36th Chinese Control Conference (CCC)*. 2017. p. 10871–10874. ISSN 1934-1768. Disponível em: <<http://doi.org/10.23919/ChiCC.2017.8029090>>. Citado na página 72.
- LIU, X. et al. Item category aware conditional restricted boltzmann machine based recommendation. In: *Proceeing, Part II, of the 22Nd International Conference on Neural Information Processing - Volume 9490*. New York, NY, USA: Springer-Verlag New York, Inc., 2015. (ICONIP 2015), p. 609–616. ISBN 978-3-319-26534-6. Disponível em: <http://dx.doi.org/10.1007/978-3-319-26535-3_69>. Citado na página 71.
- LIU, Y. et al. Content-boosted restricted boltzmann machine for recommendation. In: WERMTER, S. et al. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2014*. Cham: Springer International Publishing, 2014. p. 773–780. ISBN 978-3-319-11179-7. Disponível em: <https://doi.org/10.1007/978-3-319-11179-7_97>. Citado na página 71.
- MACEDO, F. A. B. O processo de produção musical na indústria fonográfica: questões técnicas e musicais envolvidas no processo de produção musical em estúdio. *Revista eletrônica de musicologia*, XI, set. 2007. Citado na página 29.
- MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. Cambridge, Mass: The MIT Press, 2012. ISBN 9780262018029. Citado 2 vezes nas páginas 56 e 68.
- NEAL, R. M. Annealed importance sampling. *Statistics and Computing*, Springer, v. 11, n. 2, p. 125–139, Apr 2001. ISSN 1573-1375. Disponível em: <<https://doi.org/10.1023/A:1008923215028>>. Citado na página 68.
- NGUYEN, T. D. et al. Supervised restricted boltzmann machines. In: Elidan , G.; Kersting , K. (Ed.). *Uncertainty in Artificial Intelligence-Proceedings of the 33rd Conference, UAI 2017*. Sydney, Australia, 2017. Disponível em: <<http://auai.org/uai2017/proceedings/papers/106.pdf>>. Citado na página 58.
- PASSOS, L. A.; PAPA, J. P. Fine-tuning infinity restricted boltzmann machines. In: IEEE. *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2017. p. 63–70. ISSN 2377-5416. Disponível em: <<https://doi.org/10.1109/SIBGRAPI.2017.15>>. Citado na página 67.
- PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 56.

PENG, X.; GAO, X.; LI, X. On better training the infinite restricted boltzmann machines. *Machine Learning*, Springer, v. 107, n. 6, p. 943–968, Jun 2018. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1007/s10994-018-5696-2>>. Citado na página 67.

PETERS, N.; CHOI, J.; LEI, H. Matching artificial reverb settings to unknown room recordings: A recommendation system for reverb plugins. In: *Audio Engineering Society Convention 133*. [s.n.], 2012. Disponível em: <<http://www.aes.org/e-lib/browse.cfm?elib=16443>>. Citado 2 vezes nas páginas 29 e 40.

RAMÍREZ, M. A. M.; REISS, J. D. End-to-end equalization with convolutional neural networks. In: *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx-18)*. [s.n.], 2018. p. 296–303. Disponível em: <http://dafx2018.web.ua.pt/papers/DAFx2018_paper_27.pdf>. Citado 2 vezes nas páginas 30 e 41.

REISS, J. D. Intelligent systems for mixing multichannel audio. In: IEEE. *2011 17th International Conference on Digital Signal Processing (DSP)*. 2011. p. 1–6. ISSN 2165-3577. Disponível em: <<https://doi.org/10.1109/ICDSP.2011.6004988>>. Citado 2 vezes nas páginas 29 e 40.

ROBERTSON, S. A new interpretation of average precision. In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2008. (SIGIR '08), p. 689–690. ISBN 978-1-60558-164-4. Disponível em: <<http://doi.acm.org/10.1145/1390334.1390453>>. Citado na página 80.

ROBILLARD, D. LV2 atoms: A data model for real-time audio plugins. In: *Proceedings of the Linux Audio Conference (LAC-2014)*. Karlsruhe, Germany: [s.n.], 2014. Disponível em: <<http://lac.linuxaudio.org/2014/papers/31.pdf>>. Citado na página 40.

ROUX, N. L.; BENGIO, Y. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, MIT Press, v. 20, n. 6, p. 1631–1649, June 2008. ISSN 0899-7667. Disponível em: <<https://doi.org/10.1162/neco.2008.04-07-510>>. Citado 2 vezes nas páginas 58 e 67.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: _____. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. MITP, 1987. ISBN 9780262291408. Disponível em: <<https://ieeexplore.ieee.org/document/6302929>>. Citado na página 54.

SALAKHUTDINOV, R. Learning and evaluating boltzmann machines. *Tech. Rep., Technical Report UTML TR 2008-002, Department of Computer Science, University of Toronto*, 2008. Disponível em: <<http://www.cs.toronto.edu/~rsalakhu/papers/bm.pdf>>. Citado na página 68.

SALAKHUTDINOV, R.; MNIH, A.; HINTON, G. Restricted boltzmann machines for collaborative filtering. In: ACM. *Proceedings of the 24th International Conference on Machine Learning*. New York, NY, USA: ACM, 2007. (ICML '07), p. 791–798. ISBN 978-1-59593-793-3. Disponível em: <<http://doi.acm.org/10.1145/1273496.1273596>>. Citado 7 vezes nas páginas 30, 48, 58, 68, 70, 71 e 73.

- SCHEIN, A. I. et al. Generative models for cold-start recommendations. In: *Proceedings of the 2001 SIGIR workshop on recommender systems*. [s.n.], 2001. v. 6. Disponível em: <<https://www.researchgate.net/publication/2605013>>. Citado na página 48.
- SCHIAVONE, F. L.; GOULART, A. J. H.; QUEIROZ, M. APIs para o desenvolvimento de aplicações de áudio. *Seminário Música Ciência Tecnologia*, v. 1, n. 4, 2012. Citado 2 vezes nas páginas 29 e 39.
- SCHMITT, M.; SCHULLER, B. Recognising guitar effects - which acoustic features really matter? In: EIBL, M.; GAEDKE, M. (Ed.). *INFORMATIK 2017*. Gesellschaft für Informatik, Bonn, 2017. p. 177–190. ISBN 978-3-88579-669-5. ISSN 1617-5468. Disponível em: <https://dx.doi.org/10.18420/in2017_12>. Citado 2 vezes nas páginas 30 e 41.
- SMOLENSKY, P. Information processing in dynamical systems: Foundations of harmony theory. In: _____. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986. p. 194–281. ISBN 026268053X. Citado 2 vezes nas páginas 48 e 57.
- SONG, Y.; DIXON, S.; PEARCE, M. A survey of music recommendation systems and future perspectives. In: *9th International Symposium on Computer Music Modeling and Retrieval*. London, UK: [s.n.], 2012. v. 4. Disponível em: <<https://www.researchgate.net/publication/277714802>>. Citado na página 41.
- STARK, A. M.; DAVIES, M. E.; PLUMBLEY, M. D. Rhythmic analysis for real-time audio effects. In: *International Computer Music Conference, ICMC 2008*. [s.n.], 2008. Disponível em: <<https://www.researchgate.net/publication/241813608>>. Citado 2 vezes nas páginas 30 e 41.
- STASIS, S. et al. Audio processing chain recommendation. In: *Proceedings of the 20th International Conference on Digital Audio Effects, (Edinburgh, UK)*. [s.n.], 2017. ISBN 2413-6689. Disponível em: <http://www.dafx17.eca.ed.ac.uk/papers/DAFx17_paper_75.pdf>. Citado na página 41.
- STEIN, M. et al. Automatic detection of audio effects in guitar and bass recordings. In: *Audio Engineering Society Convention 128*. [s.n.], 2010. Disponível em: <<http://www.aes.org/e-lib/browse.cfm?elib=15310>>. Citado 3 vezes nas páginas 30, 38 e 41.
- STUTZ, D. *latex-resources – Collection of LaTeX resources and examples*. 2016. Repositório de código Github. Disponível em: <<https://github.com/davidstutz/latex-resources/blob/master/tikz-multilayer-perceptron/multilayer-perceptron.tex>>. Acesso em: 16 jul 2019. Citado na página 54.
- THOMA, M. *LaTeX-examples – Examples for the usage of LaTeX*. 2016. Repositório de código Github. Disponível em: <<https://github.com/MartinThoma/LaTeX-examples/tree/23e30e1351a631f68045e00369f616640ed22ac6/tikz/restricted-boltzmann-machine>>. Acesso em: 23 set 2018. Citado na página 59.
- TIELEMAN, T. Training restricted boltzmann machines using approximations to the likelihood gradient. In: ACM. *Proceedings of the 25th International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2008. (ICML '08), p. 1064–1071. ISBN 9781605582054. Disponível em: <<https://doi.org/10.1145/1390156.1390290>>. Citado 3 vezes nas páginas 57, 64 e 67.

- TONELIB. *ToneLib — Second life for your Zoom pedal*. 2019. Repositório de código Github. Disponível em: <<https://tonelib.net/>>. Acesso em: 17 fev 2019. Citado na página 94.
- UPADHYA, V.; SASTRY, P. S. Learning RBM with a DC programming approach. In: ZHANG, M.-L.; NOH, Y.-K. (Ed.). *Proceedings of the Ninth Asian Conference on Machine Learning*. PMLR, 2017. (Proceedings of Machine Learning Research, v. 77), p. 498–513. Disponível em: <<http://proceedings.mlr.press/v77/upadhyal7a.html>>. Citado na página 67.
- VALENTE, M.; HOSFORD-DUNN, H.; ROESER, R. J. *Audiology: treatment*. [S.l.]: Thieme New York, NY, 2000. Citado na página 38.
- VERFAILLE, V.; GUASTAVINO, C.; TRAUBE, C. An interdisciplinary approach to audio effect classification. In: *Proc. Of the 9th Int. Conference on Digital Audio Effects*. [s.n.], 2006. Disponível em: <<https://www.researchgate.net/publication/252253776>>. Citado 2 vezes nas páginas 86 e 94.
- VERFAILLE, V.; ZOLZER, U.; ARFIB, D. Adaptive digital audio effects (a-dafx): a new class of sound transformations. *IEEE Transactions on Audio, Speech, and Language Processing*, IEEE, v. 14, n. 5, p. 1817–1831, Sep. 2006. ISSN 1558-7916. Disponível em: <<https://doi.org/10.1109/TSA.2005.858531>>. Citado 2 vezes nas páginas 38 e 41.
- VOORHEES, E. M. The trec question answering track. *Nat. Lang. Eng.*, Cambridge University Press, USA, v. 7, n. 4, p. 361–378, dez. 2001. ISSN 1351-3249. Disponível em: <<https://doi.org/10.1017/S1351324901002789>>. Citado na página 79.
- WARD, C. D.; COWLING, P. I. Monte carlo search applied to card selection in magic: The gathering. In: IEEE. *2009 IEEE Symposium on Computational Intelligence and Games*. 2009. p. 9–16. ISSN 2325-4270. Disponível em: <<https://doi.org/10.1109/CIG.2009.5286501>>. Citado na página 94.
- XIE, W. et al. User occupation aware conditional restricted boltzmann machine based recommendation. In: *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2016. Disponível em: <<https://doi.org/10.1109/ithings-greencom-cpscom-smartdata.2016.109>>. Citado na página 72.
- YOUNES, L. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, Taylor & Francis, v. 65, n. 3-4, p. 177–228, 1999. Citado na página 64.
- YUILLE, A. L. The convergence of contrastive divergences. In: SAUL, L. K.; WEISS, Y.; BOTTOU, L. (Ed.). *Advances in Neural Information Processing Systems 17*. MIT Press, 2005. p. 1593–1600. Disponível em: <<http://papers.nips.cc/paper/2617-the-convergence-of-contrastive-divergences.pdf>>. Citado na página 61.
- ZHANG, D. et al. Self-taught hashing for fast similarity search. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2010. (SIGIR '10), p. 18–25. ISBN

978-1-4503-0153-4. Disponível em: <<http://doi.acm.org/10.1145/1835449.1835455>>. Citado 2 vezes nas páginas 47 e 84.

ZHANG, T.; IYENGAR, V. S. Recommender systems using linear classifiers. *Journal of machine learning research*, JMLR.org, v. 2, p. 313–334, mar. 2002. ISSN 1532-4435. Disponível em: <<https://doi.org/10.1162/153244302760200641>>. Citado na página 46.

ZOELZER, U. (Ed.). *Dafx: Digital Audio Effects*. second. New York, NY, USA: John Wiley & Sons, Inc., 2011. ISBN 0471490784. Citado 2 vezes nas páginas 37 e 39.

ZOOM. *Guitar Effects & Amp Simulator G3 / G3X – Operation manual*. 2012. Disponível em: <https://zoom-na.com/sites/default/files/products/downloads/pdfs/G3_G3X_operationManual_English.pdf>. Acesso em: 29 jun 2019. Citado 4 vezes nas páginas 42, 75, 90 e 123.

Apêndices

APÊNDICE A – Fórmulas relacionadas a Redes de Markov

A.1 PMF das variáveis aleatórias de uma Rede de Markov baseada em energia

A função de massa de probabilidade de uma Rede de Markov baseada em energia com a variável aleatória $\mathbf{x}$ é

$$P(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})}, \quad (\text{A.1})$$

onde Z é a constante de normalização ($Z = \sum_{\mathbf{x}' \in \mathcal{X}} e^{-E(\mathbf{x}')}.$

Ao separar $\mathbf{x}$ em variáveis *visíveis* ($\mathbf{v}$) e em variáveis *ocultas* ($\mathbf{h}$), a Equação (A.1) fica na forma

$$P(\mathbf{x}) = P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{x})} = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})}.$$

A.2 Log-verossimilhança e seu gradiente

A.2.1 Log-verossimilhança $\ln P(\mathbf{v})$

$$\begin{aligned} \ln P(\mathbf{v}) &= \ln \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}, \mathbf{h}') \\ &= \ln \left[\sum_{\mathbf{h}' \in \mathcal{H}} \frac{1}{Z} \tilde{P}(\mathbf{v}, \mathbf{h}') \right] = \ln \left[\frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} \tilde{P}(\mathbf{v}, \mathbf{h}') \right] = \ln \left[\frac{1}{Z} \tilde{P}(\mathbf{v}) \right] \\ &= \ln \tilde{P}(\mathbf{v}) - \ln Z \\ &= \ln \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')} - \ln \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}', \mathbf{h}')} \end{aligned}$$

A.2.2 Gradiente da log-verossimilhança $\nabla_{\theta} \ln P(\mathbf{v})$

O vetor gradiente de $\ln P(\mathbf{v})$ pode ser obtido a partir de

$$\begin{aligned} \nabla_{\theta} \ln P(\mathbf{v}) &= \nabla_{\theta} (\ln \tilde{P}(\mathbf{v}) - \ln Z) \\ &= \underbrace{\nabla_{\theta} \ln \tilde{P}(\mathbf{v})}_{\text{Seção A.2.2.1}} - \underbrace{\nabla_{\theta} \ln Z}_{\text{Seção A.2.2.2}} \\ &= - \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{h}' | \mathbf{v}) \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') + \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}', \mathbf{h}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \end{aligned}$$

Relembre. As derivadas para $\ln f(\theta)$ e $e^{f(\theta)}$ utilizadas nas subseções seguintes são:

$$\begin{aligned}\frac{\partial}{\partial \theta} \ln f(\theta) &= \frac{\frac{\partial}{\partial \theta} f(\theta)}{f(\theta)} \\ \frac{\partial}{\partial \theta} e^{f(\theta)} &= e^{f(\theta)} \frac{\partial}{\partial \theta} f(\theta)\end{aligned}$$

A.2.2.1 Termo $\nabla_{\theta} \ln \tilde{P}(\mathbf{v})$

$$\begin{aligned}\nabla_{\theta} \ln \tilde{P}(\mathbf{v}) &= \nabla_{\theta} \ln \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')} = \frac{1}{\sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')}} \nabla_{\theta} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')} \\ &= \frac{1}{\sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')}} \left(- \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')} \right) \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') \\ &= - \frac{\sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')}}{\sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')}} \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') = - \frac{\frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')}}{\frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')}} \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') \\ &= - \frac{\sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}, \mathbf{h}')}{P(\mathbf{v})} \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') = - \sum_{\mathbf{h}' \in \mathcal{H}} \frac{P(\mathbf{v}, \mathbf{h}')}{P(\mathbf{v})} \nabla_{\theta} E(\mathbf{v}, \mathbf{h}') \\ &= - \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{h}' | \mathbf{v}) \nabla_{\theta} E(\mathbf{v}, \mathbf{h}')\end{aligned}$$

A.2.2.2 Gradiente do log da função de partição de uma Rede de Markov: $\nabla_{\theta} \ln Z$

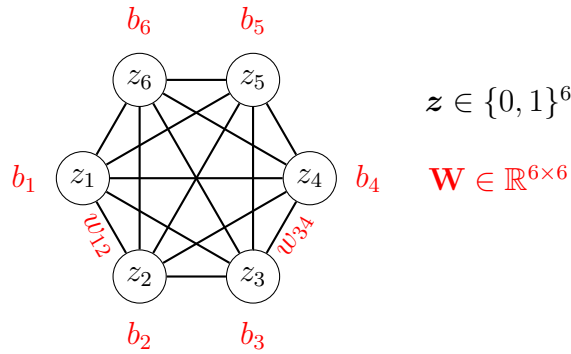
$$\begin{aligned}\nabla_{\theta} \ln Z &= \frac{1}{Z} \nabla_{\theta} Z = \frac{1}{Z} \nabla_{\theta} \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}', \mathbf{h}')} \\ &= \frac{1}{Z} \left(- \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}', \mathbf{h}')} \right) \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \\ &= - \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} \frac{1}{Z} e^{-E(\mathbf{v}', \mathbf{h}')} \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \\ &= - \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}', \mathbf{h}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}')\end{aligned}$$

APÊNDICE B – Máquina de Boltzmann

B.1 Representação de uma BM

Uma *Boltzmann Machine* (BM) ou Máquina de Boltzmann (HINTON; SEJNOWSKI, 1986) é um Modelo Gráfico Probabilístico não direcionado na qual todos os seus $U \in \mathbb{N}^*$ elementos estão conectados entre si, ou seja, sua representação em forma de grafo é fortemente conexo (Imagem 21).

Figura 21 – Representação gráfica de uma BM com seis unidades. Os parâmetros ajustáveis pelos mecanismos de aprendizagem estão dispostos em vermelho.



Fonte: Próprio autor.

Um elemento z_i de uma Máquina de Boltzmann é chamado de uma variável aleatória (ou um estado) e pode assumir o valor 0 ou 1. Ao definir $\mathbf{z}$ o vetor das variáveis aleatórias $\mathbf{z} = \{z_i\}_{i=1}^U$ e $\mathcal{X} = \{0, 1\}^U$ como conjunto de todas as possíveis combinações de valores para $\mathbf{z}$, podemos dizer que $\mathbf{z} \in \mathcal{X}$, ou seja, $\mathbf{z}$ pode assumir um dos possíveis valores de $\mathcal{X}$. Sendo mais específico, num instante de tempo t , o vetor de variáveis aleatórias assume um valor correspondente a um dos vetores da matriz $\mathcal{X}$ ($\mathbf{z}^{(t)} \in \mathcal{X}$); e uma variável aleatória z_i pode assumir um valor 0 ou 1 em um instante de tempo t ($z_i^{(t)} \in \{0, 1\}$).

A função massa de probabilidade conjunta das variáveis aleatórias de uma Máquina de Boltzmann é apresentada como

$$P(\mathbf{z}) = \frac{1}{Z} \exp \left(-\frac{E(\mathbf{z})}{T_e} \right), \quad (\text{B.1})$$

onde $E(\mathbf{z})$ é uma função denominada função de energia, o termo T_e corresponde à temperatura do sistema e Z é uma constante de normalização conhecida como função de partição, definida por

$$Z = \sum_{\mathbf{z}' \in \mathcal{X}} P(\mathbf{z}') = \sum_{\mathbf{z}' \in \mathcal{X}} \exp \left(-\frac{E(\mathbf{z}')}{T_e} \right). \quad (\text{B.2})$$

Note que já para valores pequenos de U , o cálculo da função de partição Z para a normalização da probabilidade conjunta é intratável, pois necessita que seja realizada a soma de todas as possíveis 2^U combinações de configuração para o modelo.

A temperatura do sistema T_e pode ser usada para que BMs não fiquem presas em mínimos locais, num procedimento de otimização chamado *simulated annealing*. Sem perda de generalidade, assume-se aqui que $T_e = 1$.

$E(\mathbf{z})$, a função de energia do sistema no estado $\mathbf{z}$, também presente nas Equações (B.1) e (B.2), é

$$\begin{aligned} E(\mathbf{z}) &= -\sum_{i=1}^U z_i b_i - \sum_{i < j} w_{ij} z_i z_j \\ &= -\mathbf{b}^T \mathbf{z} - \frac{1}{2} \mathbf{z}^T \mathbf{W} \mathbf{z}. \end{aligned} \quad (\text{B.3})$$

Os parâmetros do modelo, introduzidos nesta equação, são $\Theta = \{\mathbf{W}, \mathbf{b}\}$. $\mathbf{W}$ é a matriz de pesos de dimensão $U \times U$, enquanto w_{ij} representa o peso entre z_i e z_j . A matriz $\mathbf{W}$ é simétrica¹, de forma que o peso entre dois nós é o mesmo ($w_{ij} = w_{ji} \ \forall i, j$). Além disso, o peso entre um nó e ele mesmo é nulo ($w_{ii} = 0 \ \forall i$). O vetor $\mathbf{b} \in \mathbb{R}^U$ é denominado de viés (*bias*), onde cada viés b_i está associado a uma variável aleatória z_i .

A probabilidade condicional associada a cada nó de índice i no instante t dado os outros é dada por uma distribuição de Bernoulli

$$P(z_i^{(t)} | z_{j \neq i}^{(t-1)}) = \mu(z_i^{(t)})^{z_i^{(t)}} (1 - \mu(z_i^{(t)}))^{1-z_i^{(t)}},$$

em que $\mu(z_i^{(t)})$ representa $P(z_i^{(t)} = 1 | z_{j \neq i}^{(t-1)})$, a probabilidade da unidade z_i estar **ativa** no tempo t dadas as ativações de todas as outras unidades no tempo anterior ($t - 1$), e é dada pela função logística da soma ponderada das ativações de todas as outras unidades:

$$\mu(z_i^{(t)}) = P(z_i^{(t)} = 1 | z_{j \neq i}^{(t-1)}) = \frac{1}{1 + \exp\left(-\frac{1}{T_e} \sum_{j \neq i}^U w_{ij} z_j^{(t-1)} + b_i\right)}. \quad (\text{B.4})$$

B.2 Amostragem de Gibbs

O procedimento para gerar amostras (*samples*) a partir da distribuição definida por uma BM é apresentado no Algoritmo 7, cuja explicação é a seguinte:

1. Escolha um nó z_i qualquer, e compute $\mu(z_i^{(t)})$;
2. Amostre o valor 1 com probabilidade $\mu(z_i^{(t)})$ (ou o valor zero com probabilidade $1 - \mu(z_i^{(t)})$);

¹ Perceba que na forma matricial da Equação (B.3) a divisão por 2 para eliminar a soma dos pesos “duplicados” só é aplicável devida a simetria de $\mathbf{W}$ e pela diagonal principal de $\mathbf{W}$ ser zero.

3. Atualize o nó de índice i de acordo com o valor gerado no passo anterior, ou seja, faça $z_i^{(t)} = 1$ ou $z_i^{(t)} = 0$;
4. Repita o procedimento até que um critério de parada seja satisfeito.

Algoritmo 7 Amostragem de Gibbs em uma *Boltzmann Machine*

```

1 def gibbs_sampling(z: vector) -> vector:
2     while not stop_criterium(z):
3          $z_i^{(t-1)} = \text{select\_random\_node}(z)$ 
4          $z_i^{(t)} \sim \mu(z_i^{(t)})$ 
5
6     return z

```

Não é claro se o procedimento dinâmico descrito irá convergir para um equilíbrio (*equilibrium distribution*). No entanto, o procedimento acima definido pela dinâmica da BM é um exemplo de *Gibbs sampling* para a qual é possível computar o estado de equilíbrio.

Em *Gibbs sampling* são fixados todos os nós, exceto um nó i , e amostras são extraídas da distribuição a posteriori $P(z_i | \mathbf{z}_{j \neq i})$, e esse procedimento é realizado para todos os nós sequencialmente. A posteriori é dada por

$$\begin{aligned}
 P(z_i = 1 | \mathbf{z}_{j \neq i}) &= \frac{P(z_i = 1, \mathbf{z}_{j \neq i})}{P(z_i = 1, \mathbf{z}_{j \neq i}) + P(z_i = 0, \mathbf{z}_{j \neq i})} \\
 &= \frac{1}{1 + \exp\left(-\frac{1}{T_e}(E(z_i = 0, \mathbf{z}_{j \neq i}) - E(z_i = 1, \mathbf{z}_{j \neq i}))\right)}.
 \end{aligned}$$

A diferença de energia (*energy gap*) $E(z_i = 0, \mathbf{z}_{j \neq i}) - E(z_i = 1, \mathbf{z}_{j \neq i})$ é dada por

$$\begin{aligned}
 \Delta E_i &= E(z_i = 0, \mathbf{z}_{j \neq i}) - E(z_i = 1, \mathbf{z}_{j \neq i}) \\
 &= \sum_{j \neq i}^U w_{ij} z_j + b_i.
 \end{aligned}$$

Logo,

$$P(z_i = 1 | \mathbf{z}_{j \neq i}) = \frac{1}{1 + \exp\left(-\frac{1}{T_e}(\sum_{j \neq i}^U w_{ij} z_j + b_i)\right)}. \quad (\text{B.5})$$

Note que a Equação (B.5) consiste na mesma regra dinâmica da Equação (B.4). Isso significa que a dinâmica da BM é equivalente a *Gibbs sampling* da distribuição de equilíbrio definida na Equação (B.1). Assim, executar atualizações nas variáveis aleatórias irá eventualmente produzir amostras da distribuição de equilíbrio.

Portanto, há duas maneiras equivalentes de gerar amostras a partir de uma BM. A primeira é pelo procedimento descrito no início desta seção, em que a probabilidade de um certo estado $\mathbf{z}$ é dado de acordo com a probabilidade de encontrarmos o referido estado após executarmos várias (muitas!) sucessivas atualizações nas variáveis aleatórias da

BM. A segunda abordagem é escolher uma amostra a partir da distribuição de equilíbrio definida na Equação (B.1). No entanto, essa abordagem é por vezes evitada já que envolve o cálculo da função de partição, que — conforme já comentado — é normalmente intratável para redes com muitos nós.

É importante ressaltar ainda que o custo computacional para treinamento de uma BM é inviável. Uma variante da BM que possui menor custo computacional para treinamento, mas que impõe considerações de independência entre algumas das variáveis, é a Máquina de Boltzmann Restrita, discutida no Capítulo 5.

APÊNDICE C – Fórmulas relacionadas a Máquina de Boltzmann Restrita

C.1 Probabilidades Marginais para uma RBM

C.1.1 Probabilidade Marginal $P(\mathbf{v})$ de uma RBM

$$\begin{aligned}
P(\mathbf{v}) &= \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}, \mathbf{h}') \\
&= \frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-E(\mathbf{v}, \mathbf{h}')} \\
&= \frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{-(-\mathbf{h}'^T \mathbf{W} \mathbf{v} - \mathbf{v}^T \mathbf{b} - \mathbf{h}'^T \mathbf{c})} = \frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{\mathbf{h}'^T \mathbf{W} \mathbf{v} + \mathbf{v}^T \mathbf{b} + \mathbf{h}'^T \mathbf{c}} \\
&= \frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{\mathbf{v}^T \mathbf{b}} e^{\mathbf{h}'^T \mathbf{W} \mathbf{v} + \mathbf{h}'^T \mathbf{c}} = \frac{1}{Z} \sum_{\mathbf{h}' \in \mathcal{H}} e^{\mathbf{v}^T \mathbf{b}} e^{\mathbf{h}'^T (\mathbf{W} \mathbf{v} + \mathbf{c})} = \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \sum_{\mathbf{h}' \in \mathcal{H}} e^{\mathbf{h}'^T (\mathbf{W} \mathbf{v} + \mathbf{c})} \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \sum_{\mathbf{h}' \in \mathcal{H}} e^{\sum_{i=1}^K h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)} \quad (\sum_{i=1}^K \text{ faz com que } i \text{ percorra todos os } h_i \in \mathbf{h}') \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \sum_{\mathbf{h}' \in \mathcal{H}} \prod_{i=1}^K e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)} \quad (\sum_{\mathbf{h}' \in \mathcal{H}} = \sum_{h_1 \in \{0,1\}} \sum_{h_2 \in \{0,1\}} \cdots \sum_{h_K \in \{0,1\}}) \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \sum_{h_1 \in \{0,1\}} \sum_{h_2 \in \{0,1\}} \cdots \sum_{h_K \in \{0,1\}} \prod_{i=1}^K e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)} \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \sum_{h_1 \in \{0,1\}} e^{h_1 (\mathbf{W}_1 \cdot \mathbf{v} + c_1)} \sum_{h_2 \in \{0,1\}} e^{h_2 (\mathbf{W}_2 \cdot \mathbf{v} + c_2)} \cdots \sum_{h_K \in \{0,1\}} e^{h_K (\mathbf{W}_K \cdot \mathbf{v} + c_K)} \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \prod_{i=1}^K \sum_{h_i \in \{0,1\}} e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)} \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \prod_{i=1}^K [e^{0(\mathbf{W}_i \cdot \mathbf{v} + c_i)} + e^{1(\mathbf{W}_i \cdot \mathbf{v} + c_i)}] = \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \prod_{i=1}^K (1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}) \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \prod_{i=1}^K \exp \ln(1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}) \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \exp \sum_{i=1}^K \ln(1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}) = \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b}} \exp \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \\
&= \frac{1}{Z} e^{\mathbf{v}^T \mathbf{b} + \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)} \quad (\text{Função de energia livre (Equação (5.5))}) \\
&= \frac{1}{Z} e^{-F(\mathbf{v})}
\end{aligned}$$

C.1.2 Probabilidade Marginal $P(\mathbf{h})$ de uma RBM

Como o cálculo de $P(\mathbf{h})$ é similar ao de $P(\mathbf{v})$, alguns passos foram omitidos.

$$\begin{aligned}
P(\mathbf{h}) &= \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}', \mathbf{h}) = \frac{1}{Z} \sum_{\mathbf{v}' \in \mathcal{V}} e^{-E(\mathbf{v}', \mathbf{h})} \\
&= \frac{1}{Z} \sum_{\mathbf{v}' \in \mathcal{V}} e^{\mathbf{h}^T \mathbf{c}} e^{\mathbf{h}^T \mathbf{W} \mathbf{v}' + \mathbf{v}'^T \mathbf{b}} = \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \sum_{\mathbf{v}' \in \mathcal{V}} e^{\mathbf{h}^T \mathbf{W} \mathbf{v}' + \mathbf{b}^T \mathbf{v}'} = \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \sum_{\mathbf{v}' \in \mathcal{V}} e^{(\mathbf{h}^T \mathbf{W} + \mathbf{b}^T) \mathbf{v}'} \\
&= \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \sum_{\mathbf{v}' \in \mathcal{V}} \prod_{j=1}^D e^{(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j) v_j} = \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \prod_{j=1}^D \sum_{v_j \in \{0,1\}} e^{(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j) v_j} \\
&= \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \prod_{j=1}^D (e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j} + 1) = \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \prod_{j=1}^D \exp \ln(e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j} + 1) \\
&= \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} \prod_{j=1}^D \exp \text{soft}_+(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j) \\
&= \frac{1}{Z} e^{\mathbf{h}^T \mathbf{c} + \sum_{j=1}^D \text{soft}_+(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j)}.
\end{aligned}$$

C.2 Gradiente da função objetivo de uma RBM

Nessa seção será tratado o gradiente ∇_{θ} da função $f(\theta, \mathcal{D})$ (Equação (5.7)):

$$f(\theta, \mathcal{D}) = \frac{1}{N} \sum_{n=1}^N -\ln P(\mathbf{v}_n).$$

Ao aplicar o gradiente, teremos a expressão matemática abaixo. As seções seguintes tratarão $\nabla_{\theta} \ln P(\mathbf{v}_n)$ e o cálculo para cada um dos parâmetros em θ .

$$\begin{aligned}
\nabla_{\theta} f(\theta, \mathcal{D}) &= \nabla_{\theta} \left[\frac{1}{N} \sum_{n=1}^N -\ln P(\mathbf{v}_n) \right] \\
&= -\frac{1}{N} \sum_{n=1}^N \nabla_{\theta} \ln P(\mathbf{v}_n).
\end{aligned}$$

C.2.1 Termo $\ln P(\mathbf{v})$

A log-verossimilhança para uma configuração visível ($\mathbf{v}$):

$$\begin{aligned}
\ln P(\mathbf{v}) &= \ln \frac{1}{Z} e^{-F(\mathbf{v})} && \text{(Veja o Apêndice C.1.1)} \\
&= \ln e^{-F(\mathbf{v})} - \ln Z \\
&= -F(\mathbf{v}) - \ln Z.
\end{aligned}$$

C.2.2 Termo $\nabla_{\theta} \ln P(\mathbf{v})$

$$\begin{aligned}
\nabla_{\theta} \ln P(\mathbf{v}) &= -\nabla_{\theta} F(\mathbf{v}) - \nabla_{\theta} \ln Z \\
&= -\nabla_{\theta} F(\mathbf{v}) + \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}', \mathbf{h}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \quad (\text{Veja a Seção A.2.2.2}) \\
&= -\nabla_{\theta} F(\mathbf{v}) + \sum_{\mathbf{v}' \in \mathcal{V}} \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{v}') P(\mathbf{h}' | \mathbf{v}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \\
&= -\nabla_{\theta} F(\mathbf{v}) + \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') \sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{h}' | \mathbf{v}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') \\
&= -\nabla_{\theta} F(\mathbf{v}) + \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') (-\nabla_{\theta} \ln \tilde{P}(\mathbf{v}')) \quad (\text{Veja a Seção A.2.2.1}) \\
&= -\nabla_{\theta} F(\mathbf{v}) + \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') (-\nabla_{\theta} \ln e^{-F(\mathbf{v}')}) \\
&= -\nabla_{\theta} F(\mathbf{v}) + \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') \nabla_{\theta} F(\mathbf{v}').
\end{aligned}$$

Perceba que $\sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{h}' | \mathbf{v}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}')$ corresponde à esperança $\mathbb{E}_{P(\mathbf{h}|\mathbf{v})}[\mathbf{h}|\mathbf{v}]$ da Equação (5.8). Por generalização – tendo em vista que BM é uma generalização do RBM –, para Máquinas de Boltzmann Restrita temos

$$\sum_{\mathbf{h}' \in \mathcal{H}} P(\mathbf{h}' | \mathbf{v}') \nabla_{\theta} E(\mathbf{v}', \mathbf{h}') = \mathbb{E}_{P(\mathbf{h}|\mathbf{v})}[\mathbf{h}|\mathbf{v}] = \nabla_{\theta} F(\mathbf{v}'). \quad (\text{C.1})$$

C.2.3 Termo $\nabla_{\theta} f(\theta, \mathcal{D})$

O gradiente da média do negativo log-verossimilhança sobre um conjunto de N exemplos $\mathcal{D} = \{\mathbf{v}_n\}_{n=1}^N$:

$$\begin{aligned}
\nabla_{\theta} f(\theta, \mathcal{D}) &= -\frac{1}{N} \sum_{n=1}^N \nabla_{\theta} \ln P(\mathbf{v}_n) \\
&= -\frac{1}{N} \sum_{n=1}^N \left[-\nabla_{\theta} F(\mathbf{v}_n) + \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') \nabla_{\theta} F(\mathbf{v}') \right] \\
&= \frac{1}{N} \sum_{n=1}^N \nabla_{\theta} F(\mathbf{v}_n) - \sum_{\mathbf{v}' \in \mathcal{V}} P(\mathbf{v}') \nabla_{\theta} F(\mathbf{v}').
\end{aligned}$$

Os vetores gradientes apresentados na Equação (5.12) serão calculados na subseção seguinte.

C.2.4 Gradiente da função de energia de RBM em relação a cada um dos parâmetros do modelo

Utilizaremos nas subseções seguintes o relacionamento entre as funções soft_+ e sigmoide

$$\frac{\partial}{\partial \theta} \text{soft}_+(f(x)) = \sigma(f(x)) \frac{\partial}{\partial \theta} f(x). \quad (\text{C.2})$$

C.2.4.1 Função gradiente de $F(\mathbf{v})$ em relação a matriz de pesos: $\nabla_{\mathbf{W}} F(\mathbf{v})$

Definamos inicialmente a função gradiente em relação a uma matriz $\mathbf{W}$ de cardinalidade $K \times D$ para determinada função f :

$$\nabla_{\mathbf{W}} f = \begin{bmatrix} \frac{\partial f}{\partial w_{11}} & \cdots & \frac{\partial f}{\partial w_{1D}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial w_{K1}} & \cdots & \frac{\partial f}{\partial w_{KD}} \end{bmatrix}.$$

A derivada de $F(\mathbf{v})$ em relação a um peso w_{ij} , um componente da matriz $\mathbf{W}$, é

$$\begin{aligned} \frac{\partial F(\mathbf{v})}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left(-\mathbf{v}^T \mathbf{b} - \sum_{k=1}^K \text{soft}_+(\mathbf{W}_k \cdot \mathbf{v} + c_k) \right) \\ &= -\frac{\partial}{\partial w_{ij}} (\mathbf{v}^T \mathbf{b}) - \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^K \text{soft}_+(\mathbf{W}_k \cdot \mathbf{v} + c_k) \right) \\ &= 0 - \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^K \text{soft}_+(\mathbf{W}_k \cdot \mathbf{v} + c_k) \right) \\ &= -\sum_{k=1}^K \frac{\partial}{\partial w_{ij}} \text{soft}_+(\mathbf{W}_k \cdot \mathbf{v} + c_k) \\ &= -\sum_{k=1}^K \sigma(\mathbf{W}_k \cdot \mathbf{v} + c_k) \frac{\partial}{\partial w_{ij}} (\mathbf{W}_k \cdot \mathbf{v} + c_k) \\ &= -\sum_{k=1}^K \sigma(\mathbf{W}_k \cdot \mathbf{v} + c_k) \frac{\partial}{\partial w_{ij}} (\mathbf{W}_k \cdot \mathbf{v}) \\ &= -\sum_{k=1}^K \sigma(\mathbf{W}_k \cdot \mathbf{v} + c_k) \frac{\partial}{\partial w_{ij}} (w_{k1}v_1 + \dots + w_{kj}v_j + \dots + w_{kD}v_D) \\ &= -\sum_{k=1}^K \sigma(\mathbf{W}_k \cdot \mathbf{v} + c_k) \frac{\partial}{\partial w_{ij}} (w_{kl}v_l) \begin{cases} v_l, & \text{se } k=i, l=j \\ 0, & \text{caso contrário} \end{cases} \\ &= -\sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) v_j \\ &= -P(h_i | \mathbf{v}) v_j. \end{aligned} \quad (\text{Veja a Seção C.3})$$

O gradiente $\nabla_{\mathbf{W}} F(\mathbf{v})$ pode ser representado como

$$\nabla_{\mathbf{W}} F(\mathbf{v}) = -\sigma(\mathbf{W}\mathbf{v} + \mathbf{c})\mathbf{v}^T.$$

C.2.4.2 Vetor gradiente de $F(\mathbf{v})$ em relação ao vieses da camada oculta: $\nabla_{\mathbf{c}}F(\mathbf{v})$

O vetor gradiente da função de energia livre em relação ao vetor $\mathbf{c}$ é:

$$\begin{aligned}
 \nabla_{\mathbf{c}}F(\mathbf{v}) &= \nabla_{\mathbf{c}} \left(-\mathbf{v}^T \mathbf{b} - \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \right) \\
 &= -\nabla_{\mathbf{c}}(\mathbf{v}^T \mathbf{b}) - \nabla_{\mathbf{c}} \left(\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \right) = \mathbf{0} - \nabla_{\mathbf{c}} \left(\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \right) \\
 &= -\nabla_{\mathbf{c}} \left(\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \right) \\
 &= - \begin{bmatrix} \frac{\partial}{\partial c_1} \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \\ \vdots \\ \frac{\partial}{\partial c_K} \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \end{bmatrix} = - \begin{bmatrix} \frac{\partial}{\partial c_1} \text{soft}_+(\mathbf{W}_1 \cdot \mathbf{v} + c_1) \\ \vdots \\ \frac{\partial}{\partial c_K} \text{soft}_+(\mathbf{W}_K \cdot \mathbf{v} + c_K) \end{bmatrix}.
 \end{aligned}$$

A derivada parcial $\frac{\partial F(\mathbf{v})}{\partial c_i}$, i.e., um i -ésimo componente qualquer do vetor $\nabla_{\mathbf{c}}F(\mathbf{v})$ é

$$\begin{aligned}
 \frac{\partial F(\mathbf{v})}{\partial c_i} &= \frac{\partial}{\partial c_i} \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \\
 &= \sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) \frac{\partial}{\partial c_i} (\mathbf{W}_i \cdot \mathbf{v} + c_i) = \sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) \frac{\partial c_i}{\partial c_i} \\
 &= \sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i).
 \end{aligned}$$

Desta forma, o gradiente $\nabla_{\mathbf{c}}F(\mathbf{v})$ pode ser computado como

$$\begin{aligned}
 \nabla_{\mathbf{c}}F(\mathbf{v}) &= - \begin{bmatrix} \frac{\partial}{\partial c_1} \text{soft}_+(\mathbf{W}_1 \cdot \mathbf{v} + c_1) \\ \vdots \\ \frac{\partial}{\partial c_K} \text{soft}_+(\mathbf{W}_K \cdot \mathbf{v} + c_K) \end{bmatrix} = - \begin{bmatrix} \sigma(\mathbf{W}_1 \cdot \mathbf{v} + c_1) \\ \vdots \\ \sigma(\mathbf{W}_K \cdot \mathbf{v} + c_K) \end{bmatrix} \\
 &= -\boldsymbol{\sigma}(\mathbf{W}\mathbf{v} + \mathbf{c}).
 \end{aligned}$$

C.2.4.3 Vetor gradiente de $F(\mathbf{v})$ em relação ao vieses da camada visível: $\nabla_{\mathbf{b}}F(\mathbf{v})$

O vetor gradiente da função de energia livre em relação ao vetor $\mathbf{b}$ é:

$$\begin{aligned}
 \nabla_{\mathbf{b}}F(\mathbf{v}) &= \nabla_{\mathbf{b}} \left(-\mathbf{v}^T \mathbf{b} - \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \right) \\
 &= -\nabla_{\mathbf{b}}(\mathbf{v}^T \mathbf{b}) - \nabla_{\mathbf{b}} \left(\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i) \right) = -\nabla_{\mathbf{b}}(\mathbf{v}^T \mathbf{b}) - \mathbf{0} \\
 &= -\nabla_{\mathbf{b}}(\mathbf{v}^T \mathbf{b}) \\
 &= - \begin{bmatrix} \frac{\partial}{\partial b_1} v_1 b_1 \\ \vdots \\ \frac{\partial}{\partial b_D} v_D b_D \end{bmatrix} = - \begin{bmatrix} v_1 \\ \vdots \\ v_D \end{bmatrix} \\
 &= -\mathbf{v}
 \end{aligned}$$

Como $\nabla_{\mathbf{b}} F(\mathbf{v})$ é o vetor das derivadas parciais de $F(\mathbf{v})$ em relação aos vieses da camada visível, temos que, para uma posição j qualquer:

$$\frac{\partial F(\mathbf{v})}{\partial b_j} = v_j$$

C.2.5 Relação entre derivadas parciais de $F(\mathbf{v})$ e $\mathbb{E}_{P(h_i|\mathbf{v})}[h_i]$

É possível relacionar as derivadas parciais gradientes $\frac{\partial F(\mathbf{v})}{\partial w_{ij}}$ e $\frac{\partial F(\mathbf{v})}{\partial c_i}$ com a esperança condicional $\mathbb{E}_{P(h_i|\mathbf{v})}[h_i]$:

$$\begin{aligned} \sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) &= P(h_i = 1|\mathbf{v}) && \text{(Veja a Seção C.3)} \\ &= 0 \cdot P(h_i = 0|\mathbf{v}) + 1 \cdot P(h_i = 1|\mathbf{v}) \\ &= \sum_{h_i \in \{0,1\}} h_i P(h_i|\mathbf{v}) \\ &= \mathbb{E}_{P(h_i|\mathbf{v})}[h_i]. \end{aligned}$$

Dado que a sigmoide acima é um dos componentes das derivadas parciais, estas ficarão como

- $\frac{\partial F(\mathbf{v})}{\partial w_{ij}} = -\sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) v_j = -\mathbb{E}_{P(h_i|\mathbf{v})}[h_i] v_j$
- $\frac{\partial F(\mathbf{v})}{\partial c_i} = -\sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i) = -\mathbb{E}_{P(h_i|\mathbf{v})}[h_i]$.

C.3 Probabilidades Condicionais

Relembre. As funções *softplus* e *sigmoid* correspondem respectivamente a

$$\text{soft}_+(x) = \ln(1 + e^x)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}.$$

C.3.1 Probabilidade Condicional dos elementos ocultos dado os visíveis:

$$P(\mathbf{h}|\mathbf{v})$$

Calculemos inicialmente $P(\mathbf{h}|\mathbf{v})$, manipulando-o até chegar no seguinte produtório:

$$\begin{aligned} P(\mathbf{h}|\mathbf{v}) &= \frac{P(\mathbf{v}, \mathbf{h})}{P(\mathbf{v})} \\ &= \frac{\prod_{\mathbf{z}} e^{-E(\mathbf{v}, \mathbf{h})}}{\prod_{\mathbf{z}} e^{-F(\mathbf{v})}} = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{e^{-F(\mathbf{v})}} = \frac{e^{\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{v}^T \mathbf{b} + \mathbf{h}^T \mathbf{c}}}{e^{\mathbf{v}^T \mathbf{b} + \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} \\ &= \frac{e^{\mathbf{v}^T \mathbf{b}} e^{\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{h}^T \mathbf{c}}}{e^{\mathbf{v}^T \mathbf{b}} e^{\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} = \frac{e^{\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{h}^T \mathbf{c}}}{e^{\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} \\ &= \frac{e^{\mathbf{h}^T (\mathbf{W} \mathbf{v} + \mathbf{c})}}{e^{\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} = \frac{e^{\sum_{i=1}^K h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{e^{\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} \\ &= \frac{\prod_{i=1}^K e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{e^{\sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} = \frac{\prod_{i=1}^K e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{\prod_{i=1}^K e^{\text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} \\ &= \prod_{i=1}^K \frac{e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{e^{\text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} = \prod_{i=1}^K \frac{e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{e^{\ln(1 + \exp(\mathbf{W}_i \cdot \mathbf{v} + c_i))}} \\ &= \prod_{i=1}^K \frac{e^{h_i (\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}}. \end{aligned} \tag{C.3}$$

C.3.1.1 Probabilidade Condicional $P(h_i|\mathbf{v})$ em uma RBM

$$P(h_i|\mathbf{v}) = \frac{P(h_i, \mathbf{v})}{P(\mathbf{v})} = \frac{\frac{1}{\mathcal{Z}} e^{h_i \mathbf{W}_i \cdot \mathbf{v} + \mathbf{v}^T \mathbf{b} + h_i c_i}}{\frac{1}{\mathcal{Z}} e^{\mathbf{v}^T \mathbf{b} + \sum_{i=1}^K \text{soft}_+(\mathbf{W}_i \cdot \mathbf{v} + c_i)}} = \frac{e^{h_i \mathbf{W}_i \cdot \mathbf{v} + h_i c_i}}{1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}}. \tag{C.4}$$

C.3.1.2 Independência Condicional

Observe que a $P(\mathbf{h}|\mathbf{v})$ (Equação (C.3)) corresponde ao produtório das probabilidades condicionais de cada um das variáveis aleatórias ocultas (Equação (C.4)):

$$P(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^K P(h_i|\mathbf{v}),$$

o que implica que os elementos da camada visível são condicionalmente independentes entre si dado a camada visível, o que já sabíamos, pois a estrutura bipartida do RBM já nos indicava.

C.3.1.3 Probabilidade Condicional $P(h_i = 1|\mathbf{v})$ em uma RBM

A probabilidade condicional de uma variável aleatória da camada oculta ser 1 dada a camada oculta é

$$P(h_i = 1|\mathbf{v}) = \frac{e^{h_i(\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}} = \frac{e^{1(\mathbf{W}_i \cdot \mathbf{v} + c_i)}}{1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}} = \frac{e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}}{1 + e^{\mathbf{W}_i \cdot \mathbf{v} + c_i}} = \sigma(\mathbf{W}_i \cdot \mathbf{v} + c_i).$$

Processadores otimizados para a realização de operações envolvendo matrizes podem realizar rapidamente o cálculo simultâneo de $P(h_i = 1|\mathbf{v}) \forall i \in K$:

$$\left[P(h_i = 1|\mathbf{v}) \right]_{i=1}^K = \begin{bmatrix} P(h_1 = 1|\mathbf{v}) \\ \vdots \\ P(h_K = 1|\mathbf{v}) \end{bmatrix} = \boldsymbol{\sigma}(\mathbf{W}\mathbf{v} + \mathbf{c}).$$

C.3.2 Probabilidade Condicional dos elementos visíveis dado os ocultos: $P(\mathbf{v}|\mathbf{h})$

Semelhante a $P(\mathbf{h}|\mathbf{v})$, a probabilidade condicional $P(\mathbf{v}|\mathbf{h})$:

$$\begin{aligned} P(\mathbf{v}|\mathbf{h}) &= \frac{P(\mathbf{v}, \mathbf{h})}{P(\mathbf{h})} = \frac{\frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} e^{\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{v}^T \mathbf{b}}}{\frac{1}{Z} e^{\mathbf{h}^T \mathbf{c}} e^{\sum_{j=1}^D \text{soft}_+(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j)}} = \frac{e^{\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{v}^T \mathbf{b}}}{\prod_{j=1}^D e^{\text{soft}_+(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j)}} \\ &= \frac{e^{\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{v}^T \mathbf{b}}}{\prod_{j=1}^D (1 + e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j})} = \frac{\prod_{j=1}^D e^{\mathbf{h}^T \mathbf{W}_{\cdot j} v_j + v_j b_j}}{\prod_{j=1}^D (1 + e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j})} = \prod_{j=1}^D P(v_j|\mathbf{h}). \end{aligned}$$

C.3.2.1 Probabilidade Condicional $P(v_j = 1|\mathbf{h})$ em uma RBM

Para um elemento $v_j = 1$, sua probabilidade condicional pode ser apresentada em termos da sigmoide:

$$P(v_j = 1|\mathbf{h}) = \frac{e^{\mathbf{h}^T \mathbf{W}_{\cdot j} v_j + v_j b_j}}{1 + e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j}} = \frac{e^{\mathbf{h}^T \mathbf{W}_{\cdot j} \cdot 1 + 1 \cdot b_j}}{1 + e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j}} = \frac{e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j}}{1 + e^{\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j}} = \sigma(\mathbf{h}^T \mathbf{W}_{\cdot j} + b_j).$$

Processadores otimizados para a realização de operações envolvendo matrizes podem realizar rapidamente o cálculo simultâneo de $P(v_j = 1|\mathbf{h}) \forall j \in D$:

$$\left[P(v_j = 1|\mathbf{h}) \right]_{j=1}^D = \begin{bmatrix} P(v_1 = 1|\mathbf{h}) \\ \vdots \\ P(v_D = 1|\mathbf{h}) \end{bmatrix} = \boldsymbol{\sigma}(\mathbf{W}^T \mathbf{h} + \mathbf{b}).$$

Anexos

ANEXO A – *Plugins* de áudio para a unidade de efeitos Zoom G3

Aviso Legal: Zoom é uma marca comercial da ZOOM Corporation (<www.zoom.co.jp>), registrada nos EUA e/ou em outros países. Todos os nomes de produtos, marcas comerciais e marcas registradas são de propriedade de seus respectivos proprietários. Todas as empresas, produtos e nomes de serviços utilizados neste trabalho são apenas para fins de identificação e não se destinam a infringir os direitos autorais de seus respectivos proprietários. O uso desses nomes, marcas registradas e marcas não implica endosso.

A lista de *plugins* de áudio foi obtida no manual do equipamento Zoom G3 (ZOOM, 2012). O identificador único e as categorias foram obtidas em arquivos de configuração do programa para edição de *patches Zoom Edit & Share*, distribuído pela empresa do equipamento. As categorias *Original Amp* e *Amp Modeling* foram unidas, gerando *Amp*. As categorias *Original Stomp* e *Stomp Modeling* foram unidas, gerando *Stomp*.

Tabela 4 – Categorias de 60 *plugins* de áudio dos 117 presentes no equipamento Zoom G3. São apresentadas ainda a quantidade de *plugins* (coluna 2) e sua ocorrência por categoria (coluna 3).

id	Nome	Categoria	id	Nome	Categoria	id	Nome	Categoria
0	M-Filter	Filter_EQ	20	Comp+AWah	Combination Fx	40	RingMod	Modulation
1	TheVibe	Modulation	21	FLG+VCho	Combination Fx	41	Chorus	Modulation
2	Z-Organ	SFX	22	Comp+OD	Combination Fx	42	Detune	Modulation
3	Slicer	Modulation	23	Comp	Dynamics	43	VintageCE	Modulation
4	PhaseDly	Delay	24	RackComp	Dynamics	44	StereoCho	Modulation
5	FilterDly	Delay	25	M Comp	Dynamics	45	Ensemble	Modulation
6	PitchDly	Delay	26	SlowATTCK	Dynamics	46	VinFLNGR	Modulation
7	StereoDly	Delay	27	ZNR	Dynamics	47	DynaFLNGR	Modulation
8	BitCrush	Modulation	28	NoiseGate	Dynamics	48	Vibrato	Modulation
9	Bomber	SFX	29	DirtyGate	Dynamics	49	PitchSHFT	Modulation
10	DuoPhase	Modulation	30	GraphicEQ	Filter_EQ	50	BendCho	Modulation
11	MonoSynth	SFX	31	ParaEQ	Filter_EQ	51	MonoPitch	Modulation
12	SeqFLTR	Filter_EQ	32	CombFLTR	Filter_EQ	52	HPS	Modulation
13	RndmFLTR	Filter_EQ	33	AutoWah	Filter_EQ	53	Delay	Delay
14	WarpPhase	Modulation	34	Resonance	Filter_EQ	54	TapeEcho	Delay
15	TrgHldDly	Delay	35	Step	Filter_EQ	55	ModDelay	Delay
16	Cho+Dly	Combination Fx	36	Cry	Filter_EQ	56	AnalogDly	Delay
17	Cho+Rev	Combination Fx	37	Octave	Modulation	57	ReverseDL	Delay
18	Dly+Rev	Combination Fx	38	Tremolo	Modulation	58	MultiTapD	Delay
19	Comp+Phsr	Combination Fx	39	Phaser	Modulation	59	DynaDelay	Delay

Fonte: Próprio autor.

Tabela 5 – Categorias dos 57 *plugins* de áudio restantes presentes no Zoom G3. São apresentadas ainda a quantidade de *plugins* (coluna 2) e sua ocorrência por categoria (coluna 3).

id	Nome	Categoria	id	Nome	Categoria	id	Nome	Categoria
60	Hall	Reverb	80	MetalWRLD	Stomp	100.0	BG DRIVE	Amp
61	Room	Reverb	81	HotBox	Stomp	101.0	DZ DRIVE	Amp
62	TiledRoom	Reverb	82	Z Wild	Stomp	102.0	TW ROCK	Amp
63	Spring	Reverb	83	Lead	Stomp	103.0	MATCH 30	Amp
64	Arena	Reverb	84	ExtremeDS	Stomp	104.0	FD VIBRO	Amp
65	EarlyRef	Reverb	85	Aco.Sim	Stomp	105.0	HD Reverb	Reverb
66	Air	Reverb	86	Z Clean	Amp	106.0	Flanger	Modulation
67	PedalVx	Pedal Fx	87	Z MP1	Amp	107.0	None	None
68	PedalCry	Pedal Fx	88	Z Bottom	Amp	108.0	TONE CITY	Amp
69	PDL Pitch	Pedal Fx	89	Z Dream	Amp	109.0	B-BREAKER	Amp
70	PDL MnPit	Pedal Fx	90	Z Scream	Amp	110.0	BGN DRIVE	Amp
71	Booster	Stomp	91	Z Neos	Amp	111.0	DELUXE-R	Amp
72	OverDrive	Stomp	92	FD COMBO	Amp	112.0	ALIEN	Amp
73	T Scream	Stomp	93	VX COMBO	Amp	113.0	REVO-1	Amp
74	Governor	Stomp	94	US BLUES	Amp	114.0	CAR DRIVE	Amp
75	Dist+	Stomp	95	BG CRUNCH	Amp	115.0	MS 1959	Amp
76	Dist 1	Stomp	96	HW STACK	Amp	116.0	VX JMI	Amp
77	Squeak	Stomp	97	TANGERINE	Amp			
78	FuzzSmile	Stomp	98	MS CRUNCH	Amp			
79	GreatMuff	Stomp	99	MS DRIVE	Amp			

Fonte: Próprio autor.

Índice

- ADAGRAD, 67
- Adaptive digital audio effects, 40
- Annealed Importance Sampling, 68
- API, 39
- Audio effect, *veja* Efeito de áudio
- BB-RBM, 58
- Bernoulli-Bernoulli-RBM, *veja* BB-RBM
- BM, 109
 - Amostragem de Gibbs, 110
 - Função de energia, 110
 - Função de partição, 109
 - Representação, 109
- Boltzmann Machine, *veja* BM
- Categorical RBM, *veja* RBM-CF
- CB-RBM, 71
- CD, 63, 67
- Conditional RBM, 71
- Content-boosted Restricted Boltzmann Machine, *veja* CB-RBM
- Contrastive Divergence, *veja* CD
- DAFX, *veja* Digital Audio Effects
- dCRBM, 72
- Digital Audio Effects, 39
- Digital Signal Processor, 38
- Divergência Contrastiva, *veja* CD
- Divergência Contrastiva Persistente, *veja* PCD
- Efeito, 37
 - Identificação automática, 40
- Efeito de áudio, 38
 - Chorus, 38
 - Compressor, 38
 - Delay, 38
 - Distortion, 38
 - Equalizer, 38
 - Reverb, 38
- Effect, *veja* Efeito
- Explainable Restricted Boltzmann Machines for Collaborative Filtering, 72
- Filtragem Colaborativa, 47
- Gaussian-Bernoulli RBM, *veja* GB-RBM
- GB-RBM, 58
- Gradiente descendente, 61
- Harmonium, *veja* RBM
- IC-CRBMF, 71
- Instância
 - de *plugin* de áudio, 39
- Item Category Aware Conditional Restricted Boltzmann Machine Based Recommendation, *veja* IC-RBMF
- k -vizinhos mais próximos, 47
- Máquina de Boltzmann, *veja* BM
- Máquina de Boltzmann Restrita, *veja* RBM
- Máquina de Boltzmann Restrita para Filtragem Colaborativa, *veja* RBM-CF
- Máquinas de Vetores Suporte, 55
 - Truque de kernel, 55
- Métricas, 78
 - Acurácia, 78
 - Hit Ratio, 79
 - Mean Average Precision, 80
 - Normalized Discounted Cumulative Gain, 79
- Multiefeitos, 38
- Parâmetro

- Ajuste automático, 40
- Patch, 39
- PCD, 64
- Pedalboard, *veja* Patch
- Perceptron Multicamadas, 53
 - Backpropagation, 54
- Persistent Contrastive Divergence, *veja* PCD
- Plugin
 - Recomendação, 40
- Plugin de áudio, 39
- Processador digital multiefeitos, 39
- Processamento de sinais de áudio, 37
- Produção musical, 35
 - Atores do processo, 35
- RBM, 57
 - Algoritmo de atualização de Gibbs, 66
 - Aprendizagem, 60
 - ADAGRAD, 67
 - Função de energia, 59
 - Função de energia livre, 60
 - Função de massa de probabilidade conjunta, 58
 - Função de partição, 58
 - Annealed Importance Sampling, 68
 - Gibbs sampling em bloco, 65
 - Marginalização, 59
 - Poder de expressividade, 67
 - Representação, 58
- RBM-CF, 68
 - Predição, 70
 - Exata, 70
 - Por estimativa, 70
 - Representação, 68
- Regressão Logística, 52
- Restricted Boltzmann Machine, *veja* RBM, *veja* RBM-CF
- Restricted Boltzmann Machine for Collaborative Filtering, *veja* RBM-CF
- UI-RBM, 71
- Unidade de efeito, 38
- UO-CRBMF, 72
- User Occupation Aware Conditional Restricted Boltzmann Machine Based Recommendation, *veja* UO-CRBMF