
# CONSIDERANDO O RUÍDO NO APRENDIZADO DE MODELOS PREDITIVOS ROBUSTOS PARA A FILTRAGEM COLABORATIVA

Filipe Braida do Carmo

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Rio de Janeiro
Setembro de 2018

CONSIDERANDO O RUÍDO NO APRENDIZADO DE MODELOS PREDITIVOS ROBUSTOS PARA A FILTRAGEM COLABORATIVA

Filipe Braida do Carmo

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Zimbrão da Silva, D.Sc.

Prof. Leandro Guimarães Marques Alvim, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof.ª Jonice de Oliveira Sampaio, D.Sc.

Prof. Carlos Eduardo Ribeiro de Mello, Ph.D.

Prof. Ruy Luiz Milidiú, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2018

Carmo, Filipe Braida do

Considerando o ruído no aprendizado de modelos preditivos robustos para a filtragem colaborativa/Filipe Braida do Carmo. – Rio de Janeiro: UFRJ/COPPE, 2018.

XIV, 100 p.: il.; 29, 7cm.

Orientadores: Geraldo Zimbrão da Silva

Leandro Guimarães Marques Alvim

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 88 – 100.

1. Sistemas de Recomendação. 2. Filtragem Colaborativa. 3. Ruído de Classe. 4. Imperfectly Supervised Learning. I. Silva, Geraldo Zimbrão da *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Para Therezinha Moraes do Carmo [1932-2018]

*In memoriam*

# Agradecimentos

Qualquer tese é a extensão da vida do autor. Ela demonstra a consequência de uma árdua jornada de desafios, construção, amadurecimento, tornando-se assim a materialização deste processo de escolhas.

O suporte de algumas pessoas foi fundamental para percorrer esse longo caminho, e seria muito penoso sem elas. Portanto, considero justo meu sincero e profundo agradecimento àqueles que me ajudaram a trilhar nesse caminho.

Primeiramente, agradeço em especial àqueles que me apoiaram ao longo da vida, ensinando os valores da educação e da ética: meus pais;

Ao Geraldo Zimbrão pela valiosa orientação, amizade e principalmente pela confiança, além dos conselhos e dos almoços;

Ao Leandro Alvim pela orientação e parceria de pesquisa. Sua ajuda e motivação foram essenciais para o sucesso dessa tese.

À Astrid Lacerda, meu amor, por estar ao meu lado ao longo desta jornada, sendo paciente, companheira e me dando apoio necessário. Suas contribuições foram fundamentais e muito além da revisão atenciosa do texto a que se dispôs.

À todos os colegas de pesquisa do PESC pelos papos, dicas, almoços e colaborações, em especial para o Pedro Rougemont, Marden Pasianato e Gustavo Guedes;

Aos colegas do Departamento de Ciência da Computação da UFRRJ pelo apoio e confiança;

Aos meus amigos do curso de graduação. Vocês tornaram mais agradável esse longo caminho;

Ao Fellipe Duarte e ao Bruno Dembogurski pelo suporte emocional e incentivo na conclusão da tese.

Aproveito o ensejo, para manisfestar também meu agradecimento a todos os funcionários do Programa de Engenharia de Sistema e Computação da UFRJ;

Meu também agradecimento aos membros da banca examinadora da defesa pelas contribuições e participação;

Por fim, aos meus professores da vida, aos da faculdade de ciência da computação, do mestrado e do doutorado;

Meus sinceros agradecimentos.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

# CONSIDERANDO O RUÍDO NO APRENDIZADO DE MODELOS PREDITIVOS ROBUSTOS PARA A FILTRAGEM COLABORATIVA

Filipe Braida do Carmo

Setembro/2018

Orientadores: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Programa: Engenharia de Sistemas e Computação

Em sistemas de recomendação, denomina-se ruído natural as inconsistências que são introduzidas por um usuário. Inconsistências estas que são responsáveis por afetar o desempenho geral do recomendador. Até então, surgiram propostas de *data cleansing* que se baseiam em identificar essas avaliações inconsistentes e corrigi-las. Contudo, abordagens que consideram o ruído no processo de aprendizado apresentam qualidade superior. Neste cenário, surgiram procedimentos de alteração da função de custo, cuja solução para a minimização desta com dados ruidosos, corresponde à mesma solução utilizando a função original com dados sem ruído. Entretanto, estes são dependentes de um conhecimento *a priori* da distribuição do ruído e, para poder estimá-la, são necessárias certas suposições acerca dos dados. No caso da filtragem colaborativa, estas condições não são satisfeitas. Neste trabalho é proposta a utilização destas funções de custo para construir um modelo preditivo que considere o ruído no seu aprendizado. Adicionalmente, apresentamos: (a) uma heurística de geração de ruído de classe para problemas de filtragem colaborativa; (b) uma análise do quantitativo de ruído em bases; (c) análise da robustez de modelos preditivos. De forma a validar a proposta, foram selecionadas três bases mais representativas ao problema. Para tais bases, foram realizados comparativos com métodos do estado-da-arte. Nossos resultados indicam que a proposta obtém qualidade superior aos demais métodos em todas as bases e mantém uma robustez competitiva até mesmo quando se comparado com o modelo que conhece *a priori* o gerador do ruído. Por fim, abre-se um novo caminho para métodos que consideram ruído ao processo de aprendizado de modelos preditivos para filtragem colaborativa, e que, pesquisas nesta direção devem ser consideradas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

# CONSIDERING THE NOISE IN LEARNING OF ROBUST PREDICTIVE MODELS FOR COLLABORATIVE FILTERING

Filipe Braida do Carmo

September/2018

Advisors: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Department: Systems Engineering and Computer Science

In Recommendation Systems, it is named natural noise the inconsistencies that are introduced by a user. These inconsistencies affect the overall performance. Until then, data cleansing proposals have emerged with the objective to identify and correct these inconsistencies. However. approaches that consider noise in the learning process present a superior quality. Meanwhile, procedures for changing the cost function have arisen whose solution for the minimization of this with noisy data corresponds to the same solution using the original function with noiseless data. However, these procedures are dependent on previews knowledge of the noise distribution and in order to estimate it, certain assumptions regarding data are required. These conditions are not satisfied in collaborative filtering. In this work it is proposed to use these cost functions to construct a predictive model that considers noise in its learning. In addition, we present: (a) a class noise generation heuristic for collaborative filtering problems; (b) a baseline noise quantitative analysis; (c) robustness analysis of predictive models. In order to validate the proposal, three most representative datasets were selected for the problem. For such datasets, comparisons were made with state-of-the-art. Our results indicate that the proposal obtains superior prediction quality to the other methods in all the datasets and maintains a competitive robustness even when compared with the model that knows *a priori* the generator of the noise. Finally, a new direction is opened for methods that consider noise to the learning process of predictive models for collaborative filtering.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Símbolos

| | |
|---|---|
| $P$ | conjunto de preferência |
| $r_{ui}$ | avaliação de um usuário $u$ para um item $i$ |
| $\tilde{r}_{ui}$ | previsão por um modelo da avaliação de um usuário $u$ para um item $i$ |
| $\breve{r}_{ui}$ | preferência real de um usuário $u$ para um item $i$ |
| $R$ | conjunto de notas |
| $T$ | matriz de transição de ruído |
| $\mathcal{A}$ | conjunto de avaliações âncoras |
| $\tau$ | limiar |
| $U$ | conjunto dos usuários |
| $I$ | conjunto dos itens |
| $w_{uv}$ | similaridade entre os usuários $u$ e $v$ |
| $w_{ij}$ | similaridade entre os itens $i$ e $j$ |
| $N_i(u)$ | $k$ vizinhos mais próximos do usuário $u$ que avaliaram o item $i$ |
| $N_u(i)$ | $k$ vizinhos mais próximos do item $i$ que foram avaliaram o usuário $u$ |
| $\bar{r}_u$ | média das notas do usuário u |
| $\bar{r}_i$ | média das notas do item i |
| $p_u$ | $k$ variáveis latentes do usuário $u$ |
| $q_i$ | $k$ variáveis latentes do item $i$ |
| $\lambda$ | regularização |
| $\gamma$ | taxa de aprendizado |
| $\mu$ | média global |
| $b_u$ | *bias* associado ao usuário |
| $b_i$ | *bias* associado ao item |
| $\varphi$ | modelo de previsão |

# Lista de Acrônimos

| | |
|---|---|
| $NCAR$ | ruído completamente aleatório |
| $NAR$ | ruído aleatório |
| $NNAR$ | ruído não aleatório |
| $IRSVD$ | *Improved Regularized SVD* |
| $KNN$ | algoritmo do vizinho mais próximo |
| $COFILS$ | *Collaboraive Filtering to Supervised Learning* |
| $A-COFILS$ | Autoencoder COFILS |
| $IBk$ | classificador baseado em instâncias com parâmetro $k$ |
| $SVM$ | máquina de vetores de suporte |
| $ROC$ | *receiver operating characteristic curve* |
| $SVD$ | decomposição em valores singulares |
| $MAE$ | *mean absolute error* |
| $RMSE$ | *root mean absolute error* |

# Capítulo 1

# Introdução

*"I would rather discover a single fact, than to debate the great issues at length, without discovering anything."*

— Galileo Galilei

## 1.1 Contextualização

O desempenho de um classificador é mensurado através da sua capacidade de previsão de seu modelo induzido para novas amostras, sendo que a qualidade dos dados utilizados para induzi-lo é um importante fator para o desempenho. Contudo, em problemas reais, os dados utilizados para treinar o classificador possuem inconsistências e estas são causadas por diversas razões, dentre elas pode-se exemplificar a subjetividade da tarefa de rotular classes (FRENAY e VERLEYSEN, 2013). Essas inconsistências são denominadas ruído.

Em aprendizado supervisionado, o ruído pode ser descrito como flutuações que obscurecem a relação entre os atributos e a classe. Essas flutuações podem aparecer tanto nos atributos quanto na classe. Desta forma, o ruído é dividido em dois tipos: ruído de atributos e ruído de classe. Sendo que o ruído de classe é potencialmente mais prejudicial do que o ruído de atributos (SÁEZ *et al.*, 2014; ZHU e WU, 2004).

A filtragem colaborativa é uma das abordagens mais bem sucedidas de Sistemas de Recomendação (ADOMAVICIUS e TUZHILIN, 2005). Ela se baseia nas avaliações dos usuários sobre os itens, com o objetivo de sugerir ou delinear algum item que o usuário não possua preferência explícita que venha a gostar. Todos os dados utilizados para esta tarefa são oriundos de um processo de elicitação e essa aquisição de dados não difere de qualquer outra, estando sujeito a erros nos dados.

De modo geral, as pesquisas realizadas em Sistemas de Recomendação consideram que os dados não possuem inconsistências. Neste contexto, surgiram alguns

trabalhos apontam para problemas na qualidade das notas (preferências) atribuídas pelos usuários (AMATRIAIN *et al.*, 2009a; PHAM e JUNG, 2013). AMATRIAIN *et al.* (2009a) mostraram que o processo de elicitação de notas não é isento de erros, portanto pode possuir ruído. Ele pode ser introduzido de forma intencional (ruído malicioso) ou natural (ruído natural). O ruído malicioso é uma área de pesquisa já bem estabelecida na academia. Já o ruído natural é um tópico ainda pouco explorado.

O ruído natural possui uma maior importância na filtragem colaborativa, pois todo o processo de recomendação nesta abordagem gira em torno das avaliações dos usuários. Surgiram propostas de *data cleansing* com o propósito de corrigir os possíveis ruídos buscando a melhoria da qualidade do classificador (TOLEDO *et al.*, 2015). Essas heurísticas estimam o processo latente do gerador de ruído e verificam se os dados são verossímeis com essa estimativa, caso contrário são corrigidos para construir um modelo preditivo. Este tipo de abordagem apresenta dois principais problemas: correção excessiva (*overcleasing*); e a informação do ruído não é utilizada durante o processo de aprendizado. Ambos os problemas podem diminuir a qualidade do classificador.

Devido à dificuldade do processo de correção, alguns trabalhos buscam modelar conjuntamente com o classificador o ruído de classe. Desta forma, é possível aprender o modelo de geração de ruído simultaneamente com o classificador, desacoplando esses dois componentes, aumentando a qualidade do classificador. Nesse contexto, surgiram trabalhos que alteram a função de custo e tornam o modelo mais robusto.

PATRINI *et al.* (2016a) propuseram dois procedimentos de correção de função de custo de forma a tornar o modelo mais resistente ao ruído. Eles criaram um arcabouço teórico demonstrando que as soluções derivadas do aprendizado utilizando a partir destas funções com dados ruidosos são as mesmas que com os dados limpos. Contudo, faz-se necessário o conhecimento da distribuição do ruído sobre os dados.

PATRINI *et al.* (2016a); YU *et al.* (2017) propuseram um método para estimar a distribuição do ruído, porém faz-se necessário assumir certas características sobre os dados. Uma delas é a necessidade de se encontrar o que denomina-se *exemplo perfeito*, instâncias com 100% probabilidade de pertencerem à classe correta. Adicionalmente, devem existir dados suficientes para que esta condição se mantenha. Em certos cenários, *e.g.* classificação de imagem, é factível possuir essas propriedades e, caso não seja possível, é possível rotular manualmente como sugerido por YU *et al.* (2017).

As suposições feitas por PATRINI *et al.* (2016a) não são válidas no caso da filtragem colaborativa. A tarefa do usuário de expressar a sua preferência sobre um item é ambígua e abstrata, tornando a filtragem colaborativa um problema inerentemente ruidoso. Ademais, existe uma inconsistência natural do próprio usuário cujo

valor escolhido pode variar em diversas situações como: condições pessoais, escala das avaliações, o contexto da avaliação e outras (TOLEDO *et al.*, 2015). Diferentemente da proposta feita por YU *et al.* (2017), não é possível, de forma manual, corrigir ou alterar as avaliações dos usuários que são desconhecidos no conjunto de dados.

Tendo em vista os problemas descritos, o ruído é um problema ainda desafiador para a filtragem colaborativa o que leva a necessidade de um modelo robusto. Através do arcabouço teórico proposto por PATRINI *et al.* (2016a), é possível utilizar procedimentos para corrigir a função de custo e ter garantias de que a minimização destas sob dados ruidosos é a mesma se utilizássemos dados limpos. Contudo, faz-se necessário a criação de métodos para estimar a distribuição do ruído na filtragem colaborativa, assim como o estudo da robustez dos modelos criados a partir desses procedimentos. Estes desafios que permeiam os objetivos do presente trabalho e são apresentados nas próximas seções.

## 1.2 Definição do Problema

Com base nos trabalhos supracitados, no contexto de aprendizado de máquina e filtragem colaborativa, é apresentado a seguir a hipótese abordada nesta tese:

**Hipótese.** *É possível construir um modelo de previsão robusto para a filtragem colaborativa que considere o ruído ao longo do processo de aprendizagem utilizando apenas os dados das avaliações dos usuários.*

Adicionalmente à hipótese, assume-se também as seguintes premissas:

(i) existe um padrão subjacente no processo de geração de ruído nos dados, que por sua vez, pode ser aproximado por uma matriz de transição de probabilidades entre classes;

(ii) o conjunto de dados apresenta uma quantidade imprevisível de amostras de ruído e que está sujeita a alguma relação à premissa (i).

## 1.3 Objetivo da Tese

Desta forma, com base na definição do problema, o objetivo geral deste trabalho é propor um modelo de previsão para a filtragem colaborativa que considere o ruído durante o aprendizado para a construção do classificador, e que além disso seja robusto, ou seja, resistente as sucessivas adições de ruído em uma determinada base.

Para tanto, foram utilizados procedimentos de correção de função de custo tal que minimizar a sua esperança com dados ruidosos equivale a minimizar a função original com os dados sem ruído. Contudo, é necessário o conhecimento prévio da distribuição do ruído e, por esta razão, será proposto um método para tal.

De maneira mais específica, com intuito de indicar que a hipótese da tese foi alcançada, este trabalho se dividiu em:

(i) **Proposta de um modelo robusto para filtragem colaborativa**: Será proposto um modelo robusto para a filtragem colaborativa, ou seja, um modelo que não seja sujeito ao ruído natural e que considere o ruído durante sua construção.

(ii) **Estimativa do padrão subjacente de ruído**: Será proposta uma heurística para encontrar um subconjunto de elementos similares (*âncoras*) de um *exemplo perfeito*, que serão utilizados para estimar uma matriz de ruído de classes. Esta será incorporada a uma função de custo durante a estimação do ruído.

(iii) **Análise empírica**: Avaliar a robustez do modelo proposto e comparar com os demais da literatura. Para tal, será gerado, em várias taxas, ruído artificial, e desta forma, será possível avaliar a robustez das métricas de acurácia e tomada de decisão para cada modelo em bases de dados mais representativas.

## 1.4 Resumo dos Resultados

Para a resolução do problema, adotamos as premissas descritas na seção 1.2. Selecionamos três bases bem estabelecidas no domínio de sistemas de recomendação e que, de acordo com nossa análise prévia, representam as demais em função de suas características. Para essas bases, simulamos cenários em diferentes condições de ruído, tanto em relação à sua quantidade, quanto à sua natureza. Sob o ponto de vista da análise comparativa da proposta com os métodos do estado-da-arte, selecionamos os seguintes métodos: data *cleansing*; aprendizado que considera ruído em funções de custo; e um modelo neural.

A partir dos experimentos realizados, os resultados indicam que um dos modelos propostos obteve resultados superiores aos demais nas três bases e para os dois tipos de geração de ruído. Ademais, este obteve um resultado superior comparado ao mesmo procedimento, mas com conhecimento prévio da geração de ruído. Ao final, foi feita uma análise da robustez de cada método. Um dos modelos propostos com a matriz aplicada aos dados obteve o melhor resultado de robustez em comparação com os métodos do estado-da-arte. Em valores baixos de ruído, a sua robustez era semelhante ao modelo teórico. Contudo, para altas taxas de ruído houve uma grande queda do desempenho e assim diminuindo a sua robustez.

## 1.5 Contribuições

Esta tese está contextualizada na área de Sistemas de Recomendação, mais especificamente na filtragem colaborativa, contribuindo nessa área por apresentar: (i) Um modelo robusto para filtragem colaborativa; (ii) Um método para estimar a distribuição do ruído na filtragem colaborativa; (iii) Métodos para a geração de ruído artificial; (iv) Avaliação da robustez do modelo proposto; (v) Avaliação da robustez dos algoritmos de *data cleansing* para filtragem colaborativa; (vi) Avaliação de algoritmos da literatura do ruído de classe na filtragem colaborativa; (vii) Avaliação do impacto do ruído nos modelos clássicos da filtragem colaborativa.

## 1.6 Organização do Trabalho

Esta tese está organizada da seguinte forma: Nos capítulos 2 e 3 serão apresentados conceitos referentes à fundamentação teórica do trabalho; e os conceitos de ruído de classe e sistemas de recomendação respectivamente; No capítulo 4 será apresentada a proposta de um modelo robusto para tratar o ruído natural na filtragem colaborativa; No capítulo 5 será apresentada a metodologia empírica para a condução dos experimentos e análises dos resultados; Por fim, no capítulo 6, serão apresentadas as considerações finais acerca do trabalho e as principais direções de pesquisa referentes a este.

# Capítulo 2

# Ruído de Classe

*Noise is irrelevant or meaningless data or output occurring along with desired information.*

— merriam-webster dictionary

Este capítulo descreve o problema do ruído no aprendizado supervisionado. Ele é responsável por discutir os conceitos básicos sobre o ruído, conceituando o ruído de classe assim como sua taxonomia e suas consequências no aprendizado. Por fim, é apresentado uma das formas de solucionar o ruído: modelar o ruído em conjunto do classificador. Assim, o modelo será capaz de generalizar o ruído durante o processo de aprendizagem.

## 2.1 Aprendizado Imperfeito

Aprendizado supervisionado é uma classe de algoritmos dentro da área de aprendizado de máquina que possui como característica inferir uma função dado um conjunto de dados. Esse conjunto é dado pelo par entrada, também chamado de atributos, e saída, também chamada de classe ou rótulo, em que a entrada possui as características do exemplo em si e a saída o valor desejado. Através deste conjunto de dados, o algoritmo aprende a função que mapeia o valor da entrada com o valor da saída. (BISHOP, 2006)

Contudo, em um processo comum de mineração de dados, é comum considerar que os dados possuam um certo grau de inconsistência e, por tanto, o pré-processamento é uma das etapas mais importantes desse processo. Essa inconsistência é chamada na literatura de ruído. O ruído é definido como qualquer coisa que obscureça a relação entre os atributos e seu respectivo rótulo, sendo comum em conjuntos de dados reais (ANGLUIN e LAIRD, 1988).

Existem dois tipos de ruído: de atributo ou de classe (ZHU e WU, 2004). O ruído

de classe é consequência de uma rotulação manual incorreta, da falta de informação ou de falhas no processo de medição, *e.g.* definir incorretamente um rótulo como negativo em uma instância que deveria ser positiva em uma classificação binária. Já o ruído de atributo, geralmente é ocasionado em consequência de uma falha no processo de coleta dos dados, *e.g.* adicionar em um atributo um valor aleatório definido por uma distribuição normal.

SÁEZ *et al.* (2014); ZHU e WU (2004) afirmam que o ruído de classe é mais prejudicial do que o ruído de atributos e destacam a importância de lidar com esse tipo de ruído. Isso ocorre porque, nos conjuntos de dados, existem geralmente diversos atributos e somente um rótulo. Além disso, cada atributo apresenta relevância distinta para o aprendizado e o rótulo sempre terá uma grande importância nesse processo. QUINLAN (1986) obteve resultados similares, exceto para uma grande quantidade de atributos com ruído.

FRENAY e VERLEYSEN (2013) consideram que o ruído de classe é um processo estocástico e os casos aos quais o erro do processo de rotulação podem ocorrer intencionalmente ou induzidos maliciosamente por um agente não são considerados. SALMON (1973), por sua vez, mostra que o ruído é, em geral, um processo complexo. Em alguns contextos, o ruído pode ser utilizado como um processo estocástico e pode ser gerado intencionalmente, *e.g.* em uma aplicação que possa proteger a privacidade dos usuários (VAN DEN HOUT *et al.*, 2002).

Situações de aprendizado nos quais o ruído pode ocorrer são denominadas aprendizado supervisionado imperfeito (*imperfectly supervised*), *i.e.* reconhecimento de padrão em que a corretude do rótulo não é válida para todos os elementos do conjunto de treinamento (BARANDELA e GASCA, 2000).

De modo geral, o ruído é especialmente relevante para problemas supervisionados, pois altera as relações entre os atributos e a saída. Esta é a razão pela qual o ruído é estudado especialmente nos problemas de classificação e a regressão. Nestes, o ruído impede a extração do conhecimento a partir dos dados e prejudica o aprendizado de modelos, se for comparado àqueles sem a presença de ruído, o que nesse caso representaria o verdadeiro conhecimento do problema (GARCÍA *et al.*, 2015).

Além de alterar as relações entre os atributos e a saída, o ruído pode deixar o problema de aprendizado mais complexo. Quando existem fronteiras complexas ou não lineares, o ruído pode dificultar o aprendizado do modelo e assim prejudicar o desempenho do classificador. Por exemplo, quando a classe minoritária é decomposta em diversos subgrupos com poucos exemplos em cada um e rodeada pela classe majoritária (Figura 2.1a) (JO e JAPKOWICZ, 2004). Outro a citar seria se existissem alguns exemplos de diferentes classes com características semelhantes e em particular estes estivessem localizados em uma fronteira de decisão (Figura 2.1b) (GARCÍA *et al.*, 2008, 2007).

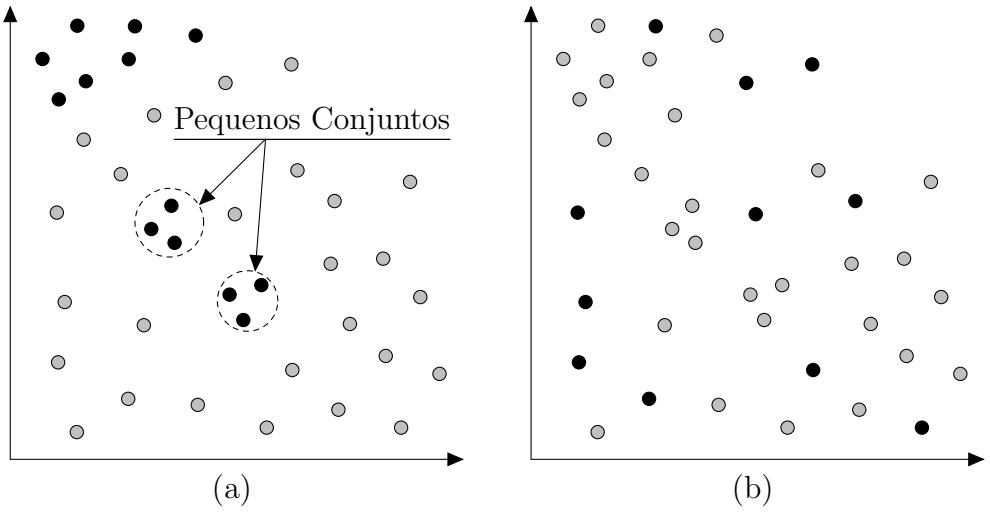



Figura 2.1: Exemplos de interação entre classes: pequenos conjuntos (a) e sobreposição de classes (b). (GARCÍA *et al.*, 2015)

Nesse interim, robustez (HUBER, 2004) é a capacidade de um algoritmo construir um modelo que seja insensível à corrupção de dados e que seja capaz de sofrer menos com o impacto do ruído. Assim pode-se afirmar que, quanto mais robusto o algoritmo for, mais o modelo gerado será similar ao modelo construído livre de ruído. Robustez pode ser considerada a característica mais importante do modelo quando o assunto é ruído.

Detecção de *outliers* e detecção de anomalias estão relacionadas intimamente com ruído de classe. Se houver uma baixa probabilidade de existir ruído, as instâncias que foram rotuladas de forma errada poderão ser consideradas como *outliers*. Similarmente, o contrário também é verdade, pois podem existir conjuntos de instâncias anômalas que podem ser tratadas como ruído. Assim, diversas técnicas que lidam com *outliers* e anomalias podem ser utilizadas para tratarem ruído (FRENAY e VERLEYSEN, 2013).

Contudo, o ruído não necessariamente é um *outlier* ou uma anomalia. O grande problema é que as suas definições são subjetivas (COLLETT e LEWIS, 1976). Por exemplo, se existir um erro no rótulo de alguma instância perto da região de fronteira de um classificador, onde todas as classes são equiprováveis, essas instâncias não serão eventos raros e nem serão anômalas. Em virtude disso, *outliers* não necessariamente são gerados por ruído e, a depender do contexto, precisarão ser considerados (LIU *et al.*, 2002).

O ruído de classe está diretamente relacionado com a sobreposição de classes. Exemplos de fronteiras são: instâncias que estiverem localizadas na área das fronteiras entre as classes. Na Figura 2.2 mostra a exemplificação do ruído e das instâncias de fronteiras. Em GARCÍA *et al.* (2006), mostrou que as classificações erradas geralmente ocorrem na fronteira entre as classes. Além disso, há uma degradação maior

do desempenho do classificador quando existem elementos ruidosos nas fronteiras entre classes do que quando estes estão localizados mais distantes dessas regiões (NAPIERAŁA *et al.*, 2010).

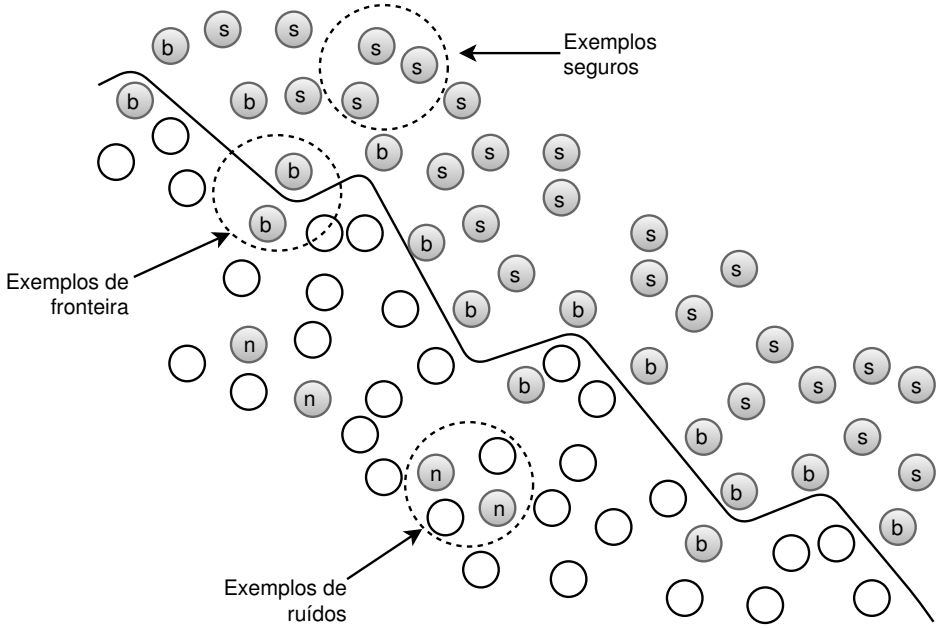



Figura 2.2: Três tipos de instâncias: exemplos confiáveis (rotuladas como s), fronteiras (rotuladas como b) e ruídos (rotulados como n). A linha contínua mostra a fronteira de decisão entre as duas classes. (GARCÍA *et al.*, 2015)

De modo geral, a origem da geração do ruído não é necessariamente importante, quando o objetivo for reduzir o impacto causado por ele (HICKEY, 1996). Entretanto, no caso em que o modelo de ruído é incorporado ao modelo de aprendizado, é importante escolher o melhor modelo que explica o ruído. O ruído de classe ocorre geralmente quando pessoas estão envolvidas diretamente no problema. FRENAY e VERLEYSEN (2013) dividiram as possíveis causas em quatro classes.

No primeiro caso, a informação fornecida ao especialista é insuficiente para a rotulação das classes (HICKEY, 1996; BRODLEY e FRIEDL, 1999). Além disso, a linguagem para descrição pode ser limitada (HARTONO e HASHIMOTO, 2007), diminuindo assim a quantidade de informação para a conclusão por parte do especialista. Em alguns casos, a informação pode ser de baixa qualidade ou essa qualidade variar por algum motivo externo. Por exemplo, um paciente pode variar suas respostas enquanto está preenchendo a ficha do seu histórico médico por causa de questões repetitivas (DAWID e SKENE, 1979).

Outro possível caso de fonte de ruído é o erro ocorrer pelo próprio especialista (HICKEY, 1996). Surgiram nos últimos anos diversas ferramentas que auxiliam em tarefas que precisam ser executadas em larga de escala, *e.g.* obter rótulos de não especialistas com intuito de reduzir custos de tempo e de coleta. Um exemplo de

*framework* é o *Amazon Mechanical Turk*[1]. Rótulos originados de não especialistas são menos confiáveis, mas dependendo do problema, este pode aliviar a falta de rótulos disponíveis (SNOW *et al.*, 2008).

O terceira classe de erro, ocorre quando a tarefa de rotulação é subjetiva e existe uma discordância entre os especialistas sobre a classe correta. Exemplos comuns dessa classe são as aplicações médicas (FRÉNAY e KABÁN, 2014). No caso da quarta classe, o ruído pode ocorrer devido a problemas de codificação ou comunicação. Um exemplo deste caso, seria o usuário marcar acidentalmente um e-mail como spam (FRÉNAY e KABÁN, 2014). Estima-se que, nos bancos de dados reais existam em torno de 5% de erro por causa de codificação ou comunicação quando não é adotada nenhuma medida específica (ORR, 1998).

## 2.2 Taxonomia do Ruído de Classe

NETTLETON *et al.* (2010) caracterizam a geração do ruído através da sua distribuição, o alvo do ruído, ou seja, ruído de atributo ou de classe, e se existe alguma dependência com alguma outra variável. FRENAY e VERLEYSEN (2013) propuseram uma taxonomia para o problema de ruído utilizando como base as características descritas anteriormente e a taxonomia do problema de valores ausentes proposta por (SCHAFER e GRAHAM, 2002).

A Figura 2.3 ilustra três modelos estatísticos que representam tipos de ruídos. O modelo considera quatro variáveis aleatórias: $X$ como vetor dos atributos, $Y$ como a classe correta, $\tilde{Y}$ como a classe observada e $E$ uma variável binária que indica a presena de ruído, ou seja, $Y \neq \tilde{Y}$. O conjunto dos possíveis atributos é $X$, enquanto as possíveis classes é $Y$. As setas são dependências estatísticas. Os modelos são ruído completamente aleatório (NCAR), ruído aleatório (NAR) e ruído não aleatório (NNAR).

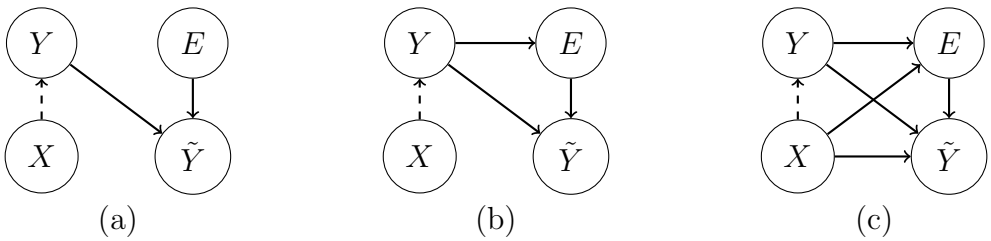



Figura 2.3: Taxonomia sob uma óptica estatística proposta por (FRENAY e VERLEYSEN, 2013). (a) ruído completamente aleatório (NCAR), (b) ruído aleatório (NAR) e (c) ruído não aleatório (NNAR). As setas correspondem às dependências estatísticas. Para melhor clareza, a dependência entre $X$ e $Y$ foi colocada como uma seta tracejada.

[1]https://www.mturk.com

O modelo mais simples é o de ruído completamente aleatório (*noisy completely at random* - NCAR). Neste, a ocorrência do erro $E$ independe das outras variáveis, incluindo o próprio rótulo. Pode-se definir uma probabilidade $p_e$ que informa se a classe observada é diferente da classe real, ou seja, $p_e = P(E = 1) = (Y \neq \tilde{Y})$. Essa probabilidade é também chamada de taxa de erro ou taxa do ruído. (KALAI e SERVEDIO, 2005)

No caso do problema de classificação binária, o ruído NCAR é necessariamente simétrico. O evento do erro ocorrer entre as classes é equiprovável. Já para o problema multiclasse, a classe incorreta é escolhida aleatoriamente em $\gamma/\{y\}$ quando o erro $E = 1$ (ASLAM e DECATUR, 1996). Desta forma, existe uma probabilidade de um rótulo observado estar correto ou não, e a seguir um dado com $|\gamma| - 1$ lados é jogado, onde $|\gamma|$ é o número de classes do problema, para definir o rótulo incorreto. Esse modelo é chamado de ruído de classe uniforme (FRENAY e VERLEYSEN, 2013).

Quando o ruído depende da classe correta $Y$, é denominado de ruído aleatório (*Noisy at Random* - NAR). O erro $E$ ainda é independente de $X$, mas esse modelo assume que possa existir uma assimetria no rótulo quando houver um ruído, *i.e.* podendo haver classes que tenham uma maior inclinação ao ruído. O NCAR é um caso especial do NAR e é definido assim as probabilidades:

$$P(\tilde{Y} = \tilde{y}|Y = y) = \sum_{e \in \{0,1\}} P(\tilde{Y}|E = e, Y = y)P(E = e|Y = y) \tag{2.1}$$

e o NAR pode ser caracterizado como uma matriz de transição,

$$\lambda = \begin{pmatrix} \lambda_{11} & \dots & \lambda_{1n_\gamma} \\ \vdots & \ddots & \vdots \\ \lambda_{n_\gamma 1} & \dots & \lambda_{n_\gamma n_\gamma} \end{pmatrix} = \begin{pmatrix} P(\tilde{Y} = 1|Y = 1) & \dots & P(\tilde{Y} = n_\gamma|Y = 1) \\ \vdots & \ddots & \vdots \\ P(\tilde{Y} = 1|Y = n_\gamma) & \dots & P(\tilde{Y} = n_\gamma|Y = n_\gamma) \end{pmatrix} \tag{2.2}$$

em que $n_\gamma = |\gamma|$ corresponde ao número de classes. Sendo que cada linha soma um e no caso do ruído uniforme a matriz fica

$$\lambda = \begin{pmatrix} 1 - p_e & \dots & \frac{p_e}{n_\gamma - 1} \\ \vdots & \ddots & \vdots \\ \frac{p_e}{n_\gamma - 1} & \dots & 1 - p_e \end{pmatrix}. \tag{2.3}$$

De modo geral, os trabalhos assumem que o ruído afeta todas as instâncias sem distinção. Entretanto, no mundo real nem sempre se pode assumir um dos dois modelos anteriores. Podemos considerar um modelo que abranja tanto o NCAR quanto o NAR e que leve em conta a existência de uma dependência do erro $E$ com os atributos $X$. Esse modelo é chamado de ruído não aleatório (*Noisy Not at*

*Random Model* - NNAR).

O modelo do NNAR é mais complexo de estimar do que os demais modelos, principalmente por depender de cada um dos atributos de $X$. Como os demais, pode-se definir uma probabilidade do erro como

$$p_e = P(E = 1) = \sum_{y \in \gamma} P(Y = y) \times \int_{x \in \chi} P(X = x|Y = y)P(E = 1|X = x, Y = y)dx \tag{2.4}$$

sendo $X$ contínuo. No entanto, em diversos casos, $p_e$ é perto de zero em quase todos espaços e a considerar a existência de alguns picos em certas regiões. Assim,

$$P_e(x, y) = P(E = 1|X = x, Y = y) \tag{2.5}$$

pode ser uma forma mais apropriada para representar a confiabilidade dos rótulos.

## 2.3 Estimando a Taxa do Ruído

O desafio do ruído de classe é que a sua distribuição é desconhecida no problema. Diversas soluções para amenizar o problema do ruído, de forma direta ou indireta, tentam estimar a taxa de ruído, e como consequência, estudam este fenômeno em função dos dados. Desta forma, surgiram alguns trabalhos cujo objetivo é estimar a matriz de transição de ruído.

Em SANDERSON e SCOTT (2014), propõe-se uma estratégia para aprendizado em um contexto em que as distribuições das instâncias do treino e do teste são diferentes. Eles propõem reduzir a tarefa de de estimação de proporção de misturas (*mixture proportion estimation*) e utilizam a curva ROC para estimar tais esses parâmetros. Já em RAMASWAMY *et al.* (2016), os autores seguem no mesmo contexto do trabalho anterior, mas propõem um algoritmo baseado em *kernel mean embedding*.

MENON *et al.* (2015) propõem estimar a taxa do ruído binário utilizando um estimador de probabilidade que foi treinado com os dados corrompidos e assim utilizar esses parâmetros em classificadores não tradicionais. A ideia do trabalho é estimar as proporções do ruído a partir do mínimo e do máximo da função de probabilidade de classe corrompida. LIU e TAO (2016) propõem também estimar a taxa do ruído binário com um estimador de probabilidade. Diferentemente, eles tentam encontrar, ao invés de uma faixa de mínimo e máximo, o limite máximo de ruído daqueles dados.

## 2.4 Consequências do Ruído no Aprendizado

Nesta seção, são descritas as consequências do ruído de classe e suas possíveis consequências negativas. No entanto, o ruído de classe pode também ter possíveis vantagens. Por exemplo, este pode ser utilizado para proteger a privacidade, *e.g.* proteger a privacidade das respostas de um questionário, tornando impossível dada sua estatística, obter as respostas individuais (VAN DEN HOUT *et al.*, 2002). Em BREIMAN (2000); MARTÍNEZ-MUÑOZ e SUÁREZ (2005); MARTÍNEZ-MUÑOZ *et al.* (2008); MARTÍNEZ-MUÑOZ *et al.* (2006), o ruído de classe foi utilizado para aprimorar os resultados da classificação.

A maioria dos trabalhos acerca das consequências do ruído de classe discute a respeito do deterioramento da performance dos classificadores (FRENAY e VERLEYSEN, 2013). No caso de problemas simples, a acurácia dos classificadores não é afetada. Em WILSON e MARTINEZ (2000); SÁNCHEZ *et al.* (1997); OKAMOTO e NOBUHIRO (1997), os autores avaliaram que a performance do classificador baseado no algoritmo dos k vizinhos mais próximos ($k$NN) é afetada pelo ruído de classe, principalmente no caso onde $k = 1$. OKAMOTO e NOBUHIRO (1997) realizaram um estudo da consequência do ruído no classificador $k$NN e mostraram que o número ótimo de $k$ aumenta linearmente com o aumento do número de instâncias ruidosas.

Em NETTLETON *et al.* (2010), foi realizado um estudo para analisar como o ruído de classe afeta a qualidade dos modelos criados por diversas técnicas de aprendizado supervisionado. No presente trabalho, as técnicas avaliadas foram o classificador Naïve Bayes, a árvore de decisão induzida pelo método C4.5, classificadores IBk (classificador baseado em instâncias com parâmetro $k$) e a máquina de vetores de suporte (SVM). Tal análise mostrou que o comportamento de cada técnica depende do tipo e da quantidade do ruído, o desequilíbrio das classes e as características dos conjuntos de dados, e assim tornando a análise complexa. Dos classificadores testados, o Naïve Bayes foi que se mostrou mais robusto ao ruído devido a suposição de independência e o SVM o menos robusto.

O ruído de classe pode afetar a complexidade do aprendizado, *e.g.* número de instâncias necessárias, e tendo como uma possível consequência o aumento do tempo de aprendizado (FRÉNAY e KABÁN, 2014). QUINLAN (1986); BRODLEY e FRIEDL (1999) mostraram que o tamanho das árvores de decisão geradas após o aprendizado é maior quando existe o ruído de classe. Nessa mesma linha de pensamento, ABELLÁN e MASEGOSA (2010) mostraram que o número de nós aumentou e que a acurácia foi reduzida.

Por conseguinte, BRODLEY e FRIEDL (1999); LIBRALON *et al.* mostraram que a filtragem dos possíveis ruídos reduziram a complexidade do SVM (número

de vetores suportes), árvores de decisão geradas pelo algoritmo C4.5 (número de árvores) e o classificador baseado em regras RIPPER (número de regras). Além desses casos, a redução do ruído ajuda a produzir modelos que ficam mais fáceis de entender, o que é desejável em alguns casos (LORENA e DE CARVALHO, 2004; SEGATA *et al.*, 2009)

Além de afetar diretamente a complexidade do problema e a acurácia, o ruído pode afetar outras tarefas correlacionadas. Em certos problemas em que existe um grande desequilíbrio entre as classes, o ruído pode interferir nas frequências das classes observadas e assim afetar diretamente a distribuição dos dados. Estudos médicos possuem uma grande preocupação em medir a incidência de uma determinada doença e sua estimativa pode ser tendenciosa em função do ruído. Outro problema semelhante com relação à distribuição é que a validação de um modelo pode ser mal avaliada na presença do ruído de classe (FRÉNAY e KABÁN, 2014), como no casos dos filtros de spam (CORMACK e KOLCZ, 2009).

## 2.5 Aprendizado Guiado ao Ruído

Existem três abordagens para amenizar o impacto do ruído no processo de aprendizagem. A primeira forma é através de heurísticas para limpeza ou filtragem dos dados. Existe uma dificuldade inerente nessa classe de métodos, pois a quantidade de elementos selecionados pode afetar diretamente o desempenho do classificador. (FRENAY e VERLEYSEN, 2013)

A segunda abordagem é através da robustez natural de certos classificadores, visto que alguns classificadores são mais afetados pelo ruído do que outros. Já a última abordagem é considerar o aprendizado do ruído dentro da criação do modelo, aprendendo simultaneamente o ruído de classe em conjunto com o classificador.

A terceira abordagem possui uma vantagem sobre as demais, pois se comparado às outras, ela desacopla os dois componentes, classificador e detecção do ruído, no processo de aprendizado, aumentando a acurácia do classificador. Por esta razão, surgiram diversas propostas com o objetivo de aprender o ruído durante o processo de criação do classificador.

A seguir, são relatados trabalhos dos quais considera-se o ruído durante a construção do modelo. Em um primeiro momento, serão relacionadas as técnicas que alteram a arquitetura da rede neural e assuntos correlatos à proposta. Em seguida, serão analisadas soluções para o ruído de classe que utilizam funções de custos mais robustas, que constituem a base da proposta deste trabalho. Estas, que dão sustentação teórica para solução feita por PATRINI *et al.* (2016a), discutida posteriormente no próximo capítulo.

### 2.5.1 *Deep Learning* com Ruído de Classe

Diversos trabalhos surgiram para tentar amenizar as consequências do ruído de classe nos modelos de *Deep Learning*, principalmente em Visão Computacional. Os trabalhos, na sua maioria, procuram incorporar o conhecimento do ruído na arquitetura da rede neural a fim de deixá-lo mais robusto.

No trabalho de SUKHBAATAR e FERGUS (2014), os autores propõem uma camada adicional na saída da rede neural com objetivo que ela capture a distribuição do ruído que está incorporado aos dados, ou seja, a matriz de pesos irá aprender esse comportamento. Complementarmente, eles propõem modificações no processo de treinamento a fim de forçar o aprendizado do ruído somente nesta camada. O treinamento proposto pode ser divido em duas etapas e, após o treinamento, a última camada será removida para realização das previsões no conjunto de testes.

A primeira etapa definirá os pesos dessa camada extra como a identidade. Durante esta parte do treinamento não haverá atualizações sobre os pesos nesta camada, e assim a rede começa o processo geral de aprendizado. Depois de algumas épocas, a última camada começará a ser atualizada. Para evitar que o conhecimento do aprendizado passe para essa última camada, eles propõem em utilizar uma regularização nesta camada a fim de deixar as modificações entre épocas pequenas. As figuras 2.4 e 2.5 ilustram o processo proposto.

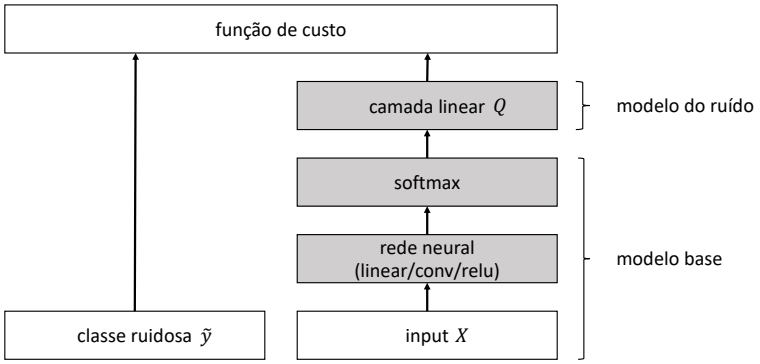



Figura 2.4: O ruído de classe é modelado como uma camada extra após a saída do classificador e a ideia que a distribuição do ruído se torne a matriz de pesos desta camada e assim mudando as probabilidades da saída do classificador. (SUKHBAATAR e FERGUS, 2014)

XIAO *et al.* (2015) propõem uma arquitetura de uma rede neural que tentará aprender sobre o tipo do ruído que pode estar associado aos dados em conjunto com o rótulo. Essa arquitetura é dividida em três etapas. A primeira e a segunda são independentes. São dois modelos onde um aprenderá o rótulo e o outro o tipo do ruído, sendo que o conjunto de treinamento para o segundo modelo foi construído manualmente. A terceira parte é responsável por unir essas duas informações e utilizá-las em conjunto para estimar o verdadeiro rótulo dos dados. A arquitetura

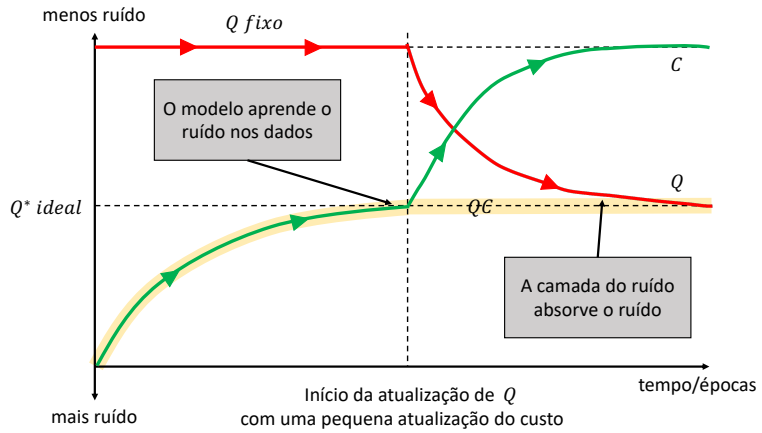



Figura 2.5: O gráfico mostra o processo de treinamento com os dados ruidosos. Nas primeiras épocas, a camada utiliza a matriz identidade e a rede começa aprender a tarefa em si. No segundo momento os pesos desta camada começam lentamente a serem atualizados e isso capturando as propriedades do ruído incorporado aos dados. (SUKHBAATAR e FERGUS, 2014)

pode ser vista na imagem 2.6.

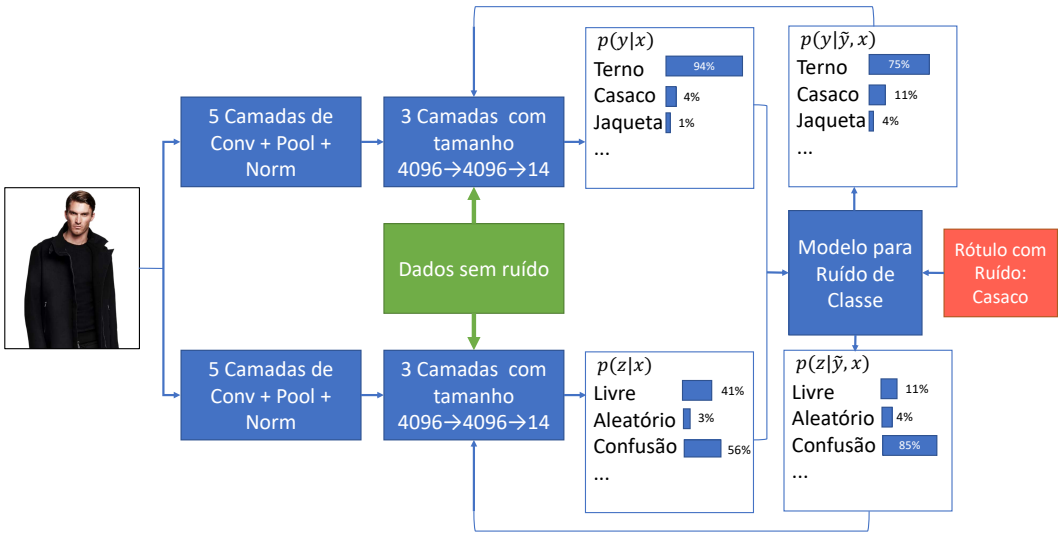



Figura 2.6: Arquitetura proposta por XIAO *et al.* (2015). Essa arquitetura é dividia em três partes. Uma rede para aprender sobre as probabilidades dos rótulos, uma outra para aprender o tipo do ruído e por fim uma para juntar as informações das duas anteriores.

GOLDBERGER e BEN-REUVEN (2017) investigam também a utilização de uma camada, denominada *noise channel*, ao final da rede neural com objetivo de absorver a distribuição do ruído e assim aumentar a robustez do modelo. Esse trabalho é similar ao trabalho de SUKHBAATAR e FERGUS (2014), sendo que a única diferença neste é que o processo de aprendizado não muda com relação a uma rede neural clássica, enquanto no outro o processo de aprendizado é dividido em duas etapas.

### 2.5.2 Abordagens através da Função de Custo

O aprendizado supervisionado pode ser descrito com uma maior formalização, da seguinte forma: dado um conjunto de treinamento com $n$ exemplos na forma $\{(x_1, y_1), ..., (x_n, y_n)\}$, o algoritmo de aprendizado procura no espaço de hipóteses $G$ a função que faça o mapeamento do espaço de entrada $X$ em relação à saída $Y$, ou seja, $g : X \to Y$. A função $g$ é um elemento do espaço $G$ que é chamado de espaço de hipóteses.

Com objetivo de determinar o quanto a função $g$ se encaixa melhor com os dados, a função de custo (*loss function*) $\ell : Y \times Y \to \Re^+$ é definida. Ela mede numericamente o quão longe o valor estimado está do real. Dada uma instância $(x_i, y_i)$, o custo do valor estimado $\hat{y} = g(x_i)$ será dado por $\ell(y_i, \hat{y})$.

Definindo o risco como a esperança da função de custo

$$R(g) = \frac{1}{n} \sum_i \ell(y_i, g(x_i)) \ ,$$

o objetivo de um algoritmo de aprendizado é minimizar o risco para um conjunto de dados. São listados na tabela 2.1 alguns exemplos de função de custo.

Tabela 2.1: Alguns exemplos de função de custo.

| **Nome** | **Equação** |
|---|---|
| Erro Quadrático | $\ell(y_i, \hat{y}) = \lvert y_i - \hat{y} \rvert$ |
| Erro Absoluto | $\ell(y_i, \hat{y}) = (y_i - \hat{y})^2$ |
| Entropia Cruzada | $\ell(y_i, \hat{y}) = -y_i \cdot log(\hat{y}_i)$ |

Surgiram na literatura estudos sobre combater o ruído de classe através de funções de custo mais robustas. De modo geral, os trabalhos apresentam correções na função de custo para combater o ruído de classe e assim aumentar sua robustez. Contudo, em todos os trabalhos, é necessário um conhecimento *a priori* sobre os dados ou assumir certas características.

STEMPFEL e RALAIVOLA (2009) propõem uma modificação da função de custo do algoritmo SVM para lidar com ruído de classe em um problema de classificação binária. Para tal, eles sugerem uma heurística para estimar as taxas de ruído e a utilizam na função de custo para minimizar o impacto do ruído. Eles realizam uma análise teórica garantindo a proximidade do objetivo funcional proposto ao sem ruído, porém devem existir certas condições como a simetria do ruído entre as duas classes. Desta forma, a matriz de transição seguiria a seguinte forma com uma taxa

$\sigma$ para o problema de classificação binária:

$$T = \begin{bmatrix} 1-\sigma & \sigma \\ \sigma & 1-\sigma \end{bmatrix} \ . \tag{2.6}$$

NATARAJAN *et al.* (2013) desenvolvem uma extensão à função de custo (imagem 2.7) para problemas de classificação binária com dados corrompidos de forma simétrica. Ele chamou o método de "estimador não-viesado". No seu trabalho, ele desenvolve uma teoria que demonstra a eficiência da função proposta. Contudo, faz-se necessário ao funcionamento do método conhecimento sobre o ruído sobre os dados. Eles propõem utilizar algum método de validação cruzada para descobrir o valor de $\sigma$, ou seja, é necessário conhecer a matriz de transição de ruído.

REED *et al.* (2014) sugeriram lidar com a falta de confiança dos rótulos utilizando uma modificação da função de custo em que é adicionada uma combinação convexa entre os rótulos e a previsão do modelo. A ideia é que a medida que o modelo melhora ao longo do tempo, suas previsões se tornam mais confiáveis e é possível diminuir o impacto do rótulos ruidosos dando um peso maior para as previsões vindas do modelo. Foram propostas duas modificações à função de custo entropia cruzada que eles chamaram de *bootstrapping soft* e *bootstrapping hard*.

O *bootstrapping soft* utiliza todas as probabilidades das classes previstas, ou seja,

$$\ell_{soft}(y_i, \tilde{y}) = -[\beta y + (1-\beta)\tilde{y}_i] \log(\tilde{y}_i) \ . \tag{2.7}$$

Já o *bootstrapping hard* utiliza somente a classe que o modelo classificou

$$\ell_{soft}(y_i, \tilde{y}) = -[\beta y + (1-\beta)\tilde{z}_i] \log(\tilde{y}_i) \ , \tag{2.8}$$

onde $z_i = \mathbf{1}$, sendo que $i = \text{argmax } \tilde{y}_i, i = 1...C$ e $C$ o número de classes.

ROOYEN (2015) estende o trabalho de NATARAJAN *et al.* (2013) e desenvolve métodos gerais para construir estimadores não-viesados para ruído simétricos para classificação multiclasse. Além disso, ele desenvolve os limites inferiores e superiores para a utilização dessa função de custo nos algoritmos de aprendizagem. Em particular, ele demostrou que a utilização da função não afeta a taxa em que o aprendizado o ocorre. Ela só afeta as contantes no limite superior e inferior. A figura 2.7 mostra a função de custo proposta.

$$\ell_T(y, a) = \frac{(1 - \sigma)\ell(y, a) - \sigma\ell(-y, a)}{1 - 2\sigma}$$

Figura 2.7: Função de custo proposta nos trabalhos NATARAJAN *et al.* (2013) e depois estendida por ROOYEN (2015) para o problema de classificação binária com ruído simétrico.

Utilizando a base teórica desenvolvida nos trabalhos anteriores e em PATRINI *et al.* (2016a), PATRINI *et al.* (2016b) desenvolveram dois procedimentos genéricos para classificação multiclasse para ruído assimétrico do tipo NAR, denominado processo de correção *backward* e *forward*. Neste trabalho, é desenvolvida a teoria dos dois processos de correção demonstrando que são não-enviesados. Os dois processos precisam do conhecimento da matriz de transição de ruído e para tal, é proposto um método para estimá-lo.

# Capítulo 3

# Ruído na Filtragem Colaborativa

*"Data science becomes the art of extracting label out of thin air"*
— MALACH e SHALEV-SHWARTZ (2017)

Esse capítulo descreve o problema do ruído natural no panorama da filtragem colaborativa. Ele é dividido, para um melhor entendimento, em duas seções. Na primeira seção são apresentados os conceitos e técnicas sobre a área de Sistemas de Recomendação e se aprofundando no problema da filtragem colaborativa. Após a fundamentação da área, é discutido o problema do ruído na filtragem colaborativa e detalhando cada técnica disponível na literatura para amenizar esse problema.

## 3.1 Sistemas de Recomendação

Durante toda a história, as pessoas sempre recorreram a recomendações com o objetivo de facilitar ou minimizar o risco de uma tomada de decisão. A importância da recomendação cresceu com o surgimento da sociedade de informação, na qual a informação ganhou grande importância tornando-se o fator de poder e de mudança social. Essa importância se deveu ao desenvolvimento e o barateamento das tecnologias de informação e de comunicação (TIC).

Sistemas de Recomendação (SR) são uma realidade. Em decorrência da grande quantidade e variedade de itens torna-se inviável um usuário avaliar cada um a fim de decidir o que irá consumir. Além disso, essa sobrecarga de informação, paradoxalmente, torna-se um problema e não uma solução, pois quanto mais opções são dadas ao usuário, maior é sua expectativa sobre a sua escolha, dificultando assim sua tomada de decisão quanto a escolha. Esse problema é conhecido como paradoxo da escolha (SCHWARTZ, 2005).

Nesse cenário, surgem os sistemas de recomendação, que são um conjunto de ferramentas e técnicas computacionais, criadas com o fim de selecionar itens per-

sonalizados para um usuário (SARWAR *et al.*, 2002), que podem ser referentes a: música, filmes, notícias, anúncios, produtos em uma loja virtual, serviços e outros. Diversas empresas como Google[1], Netflix[2] e Amazon[3] vêm utilizando intensivamente essas técnicas com o objetivo de obter vantagens comerciais.

Devido a simplicidade dos algoritmos, estes acabaram sendo amplamente utilizados para fins comerciais e atualmente possuem uma grande importância nesse contexto. Estudos têm demostrado que SR trazem três principais benefícios para o comércio eletrônico: o aumento das vendas, vendas cruzadas e uma maior lealdade dos seus usuários. (LINDEN *et al.*, 2003)

O primeiro sistema de recomendação surgiu no início da década de 90, denominado *Tapestry* (ADOMAVICIUS e TUZHILIN, 2005). Durante a elaboração desse sistema, foi mencionada pela primeira vez a expressão Filtragem Colaborativa, cujo o objetivo é designar um tipo de sistema que utiliza a colaboração entre pessoas para realizar a filtragem de informação. Tal termo foi adotado para denominar uma categoria de sistemas de recomendação posteriormente.

A origem dos sistemas de recomendação pode ser traçada em trabalhos em ciência cognitiva, busca e recuperação da informação, teorias de previsão, ciência da gestão e marketing (ADOMAVICIUS e TUZHILIN, 2005). Apesar disso, as primeiras pesquisas surgiram de forma independente, com o foco nos problemas de recomendação, que dependem explicitamente da estrutura de uma avaliação. Na formulação mais comum, o problema de recomendação pode ser reduzido ao problema de estimação das notas de todos os itens que um usuário não viu. Intuitivamente, a estimação pode ser feita utilizando as notas do usuário sobre os outros itens e outras informações como sua idade.

ADOMAVICIUS e TUZHILIN (2005) formalizaram o problema de recomendação como: Seja $U$ o conjunto de todos os usuários e $I$ o conjunto de todos os itens. Seja $\mathbf{r}$ a função utilidade que informa a importância do item $i$ ao usuário $u$, *i.e.* $\mathbf{r} : U \times I \rightarrow P$ onde $P \in \mathbb{R}$, é o conjunto de preferência do usuário pelo item. Então, para cada usuário $u \in U$ é escolhido o item $i' \in I$ onde maximize a função utilidade $\mathbf{r}$, ou seja:

$$\forall u \in U,\ i'_u = \arg\max_{i \in I} \mathbf{r}(u, i). \tag{3.1}$$

Cada elemento do espaço $U$, pode ser definido por um perfil representado pelas informações dos usuários e inclui, como exemplo: idade, gênero, altura, entre outras. Do mesmo modo, cada elemento de $I$ pode ser definido por um conjunto de características. Por exemplo, dentro de um contexto de um sistema de recomendação de

[1]http://www.google.com
[2]http://www.netflix.com
[3]http://www.amazon.com

filme, as características podem ser o título, resumo, diretor, ano do lançamento e demais.

Em sistemas de recomendação, a função utilidade é representa normalmente por um valor numérico que pode corresponder à nota (*ranking*) do usuário sobre o item, *e.g.* Astrid deu uma nota 6 (de um total de 10) para o filme "Poderoso Chefão". Contudo, a função utilidade pode ser uma função arbitrária, definida pelo usuário ou calculada pelo sistema. A tabela 3.1 é um fragmento de uma matriz de notas de um sistema de recomendação de filmes, em que o usuário define explicitamente o seu gosto sobre os filmes.

Tabela 3.1: Fragmento de uma matriz de notas de um sistema de recomendação de filmes.

| | Titanic | Poderoso Chefão | Matrix |
|---|---|---|---|
| Filipe | 4 | $\emptyset$ | 3 |
| Astrid | 4 | 5 | 5 |
| Bruno | 4 | 5 | 5 |
| Fellipe | $\emptyset$ | 5 | $\emptyset$ |

O problema central é que, a função utilitária **r** não é definida para todo o espaço $U \times I$, mas somente para um subconjunto dele. Assim, torna-se necessário extrapolá-la para todo o espaço. Todavia, o objetivo de um sistema de recomendação consiste em prever as notas dos usuários para todos os itens que ainda não tenham sido vistos, a fim de utilizar essas previsões para recomendação. A extrapolação da função utilidade é passível de ser feita através de heurísticas, que podem receber informações do perfil do usuário e das características do item, validando empiricamente a seu desempenho, ou estimando tal que a função é otimizada por um critério de desempenho, como o erro quadrático.

Após estimar todas as notas dos itens que não foram avaliados pelos usuários, utilizando alguma das duas abordagens citadas acima, o sistema de recomendação seleciona o item desse conjunto cujo valor da nota é o maior e o recomenda ao usuário. Uma outra abordagem é recomendar os $N$ melhores itens para o usuário.

A estimativa da nota de um item que não foi avaliado por um usuário pode ser feita de diversas formas, seja através da utilização de técnicas de aprendizado de máquina, da teoria da aproximação ou através de outras heurísticas. A abordagem que é utilizada para estimar a nota a partir dos dados, é a mesma utilizada para classificar os algoritmos. BURKE (2007) propôs uma taxonomia constituída de cinco classes e o trabalho foi estendido por RICCI *et al.* (2011) adicionando uma nova classe. A seguir, a descrição das classes:

- Baseada em conteúdo: a recomendação é feita utilizando as informações dos itens e dos usuários;

- Filtragem Colaborativa: a recomendação é feita através da descobertas de padrões observando as preferências de comportamento sobre a comunidade de usuários;
- Demográfico: o sistema utiliza a informação demográfica do usuário para realizar a recomendação;
- Baseada no conhecimento: a partir de um conhecimento prévio sobre o domínio, são feitas inferências para adequar às necessidades dos usuários.
- Baseada na comunidade: a recomendação é feita utilizando as preferências dos usuários relacionados;
- Abordagens híbridas: método que combina quaisquer das abordagens citadas acima.

Outro problema que é discutido dentro de Sistemas de Recomendação, além de prever a preferência de um usuário sobre um item, é prever a ordem relativa do consumo dele, ou seja, produzir uma lista dos melhores $N$ itens recomendados (*Top-N*), tal que o item classificado melhor é aquele de maior preferência (SARWAR *et al.*, 2001a). É esperado que o usuário avalie essa lista por inteiro. Um exemplo dessa categoria seria a de lojas virtuais.

Adiante, é apresentado um maior detalhamento a respeito da abordagem filtragem colaborativa e seus principais algoritmos.

### 3.1.1 Filtragem Colaborativa

Desde os primórdios as pessoas recorrem às opiniões de outras pessoas para obterem auxílio em alguma tomada de decisão sobre alguns itens. Através dessas opiniões, uma pessoa poderia ponderar com relação à confiança dela pela fonte da recomendação, e sua expectativa sobre o produto ou serviço. Ela é também conhecida como recomendação boca a boca. A filtragem colaborativa utiliza como base esse conceito e pode ser interpretada como uma automatização desse tipo de recomendação.

A ideia central dessa abordagem é identificar o conjunto de usuários $U' \subset U$ que explicitaram uma nota para o item $i$ e que possuem o mesmo comportamento do que o usuário $u$. Pressupondo que o usuário $u$ tenda para o mesmo comportamento do conjunto $U'$, podemos afirmar que a nota dele para o item $i$ terá a mesma característica que o conjunto $U'$. Devido às suas características, a filtragem colaborativa também é chamada na literatura de abordagem social.

Uma possível forma de representar o problema da filtragem colaborativa é visualizá-la como um grafo bipartido (MELLO *et al.*, 2010), em que os vértices de uma

partição são representados pelo conjunto dos usuários $U$, o da segunda partição pelos itens $I$ e as arestas correspondem as notas atribuídas ao par usuário-nota. Desta forma, o objetivo do sistema consiste em prever novas arestas e seus pesos, com base nas arestas e pesos informados previamente. Essa representação é exemplificada na figura 3.1.

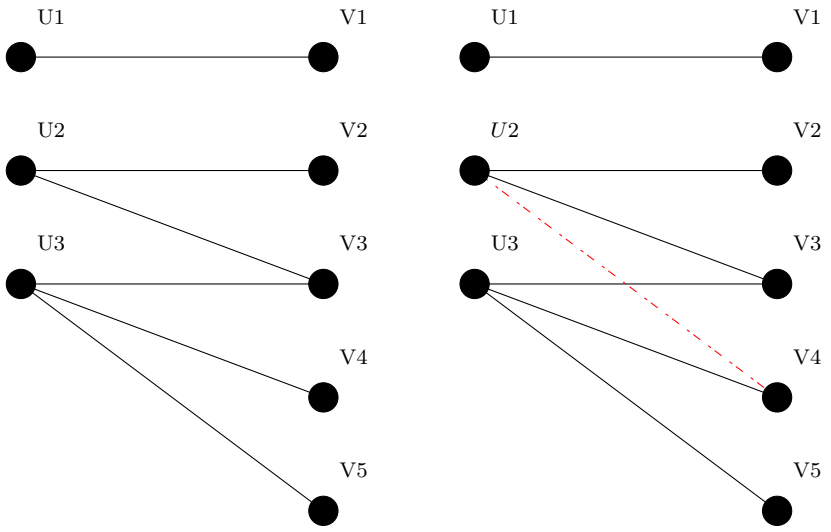



Figura 3.1: Representação do problema de filtragem colaborativa como um grafo bipartido não direcionado.

A filtragem colaborativa possui diversos benefícios sobre a abordagem baseada em conteúdo. Um deles é não depender de informação externa de usuários e itens, isso porque a qualidade da recomendação está diretamente relacionada à qualidade das informações disponíveis. Outro benefício é que a filtragem colaborativa apresenta mais dinamicidade e diversificação quanto as recomendações para um usuário, visto que sua recomendação depende de um comportamento macro dos usuários e itens. Fato que não ocorre na recomendação por conteúdo, cuja dependência forte dos dados externos do usuário acarretam super especialização.

Entretanto, existem diversos desafios nesta abordagem. Usualmente, as bases dos sistemas de recomendação possuem uma grande quantidade de usuários e de itens. Consequentemente, isso torna a matriz usuário-item extremamente esparsa e o desempenho dos algoritmos está diretamente relacionado com relação a esparsidade da base (LINDEN *et al.*, 2003). A esparsidade dos sistemas de recomendação normalmente fica em torno de 1%, ou seja, a quantidade de notas com relação ao total possível. (SARWAR *et al.*, 2001b)

Existem diversas abordagens que tentam minimizar o efeito do problema des-

crito anteriormente. As soluções, por sua maioria, utilizam técnicas de redução de dimensionalidade como a decomposição de valores singulares (SVD) ou análise dos principais componentes (PCA) para lidar com a esparsidade e com isso prover boas recomendações. (RICCI *et al.*, 2011; GOLDBERG *et al.*, 2001; BILLSUS, 1998)

Outro problema decorrente da estrutura da filtragem colaborativa, é o surgimento de um usuário ou item novo. Existe uma necessidade intrínseca nesse tipo de abordagem, pois é necessário um conjunto de avaliações para realizar uma predição. Desta forma, As avaliações dos usuários são somente utilizadas para realizar a predição, dessa forma um usuário ou item novo falhará, já que não será possível compará-lo com ninguém dentro do sistema, impossibilitando assim, a recomendação desse item até o momento que ele obtiver alguma avaliação de algum usuário ou no caso do usuário quando ele avaliar algum item. Esse problema é chamado *cold starter*. (ADOMAVICIUS e TUZHILIN, 2005)

Em alguns cenários, podem existir usuários que não concordem ou discordem com nenhum grupo de pessoas dentro do sistema, e por isso não se beneficiam da filtragem colaborativa. Esses grupos de usuários que possuem preferências opostas dos demais, são chamados de *gray sheep*. (ADOMAVICIUS e TUZHILIN, 2005)

Outro desafio da filtragem colaborativa, são os *shilling atacks*. Em alguns cenários, pessoas podem se utilizar das avaliações para fim próprio. Essas podem gerar avaliações positivas para os seus produtos e negativas para os materiais dos seus competidores. É desejável que os sistemas de recomendação baseados em filtragem colaborativa, introduzam precauções que desencorajem esses usuários de realizarem tal comportamento. (GUNES *et al.*, 2012)

Em um processo clássico de mineração de dados, é considerado que os dados possuem um grau de inconsistência e por isso, torna-se o pré-processamento um dos passos mais importantes dentro de um processo de mineração (HAN, 2005). Em sistemas de recomendação é pressuposto que as avaliações feitas pelos usuários sobre os itens sejam livre de irregularidades. Apesar disso, AMATRIAIN *et al.* (2009a) demonstraram que as avaliações dos usuários podem ter inconsistências mesmo quando eles explicitam diretamente a sua nota para um item. Essa inconsistência é chamada de ruído natural (O'MAHONY *et al.*, 2006) e pode existir mesmo sem nenhuma intenção maliciosa.

No mundo real, conceitos geralmente não são estáveis e mudam com o tempo. Como exemplo, o gosto de um usuário sobre filmes pode mudar da adolescência para sua fase adulta. Essas mudanças, fazem com que o modelo gerado para os dados antigos diminua sua acurácia com o tempo. Para tal, é necessário realizar procedimentos para atualizar o modelo com a entrada de novos dados. Contudo, os conceitos dependem de certos contextos escondidos, ou seja, eles não são descritos por um conjunto de entradas explicitadas. Mudanças nesse contexto podem induzir

pequenas ou grandes mudanças em um conceito e isso é denominado de *concept drift*. Em muitos domínios essas mudanças de conceitos são recorrentes gerando uma grande dificuldade em detectar o que é mudança e o que é ruído. (GAMA *et al.*, 2014)

Mesmo com esses desafios, a abordagem por filtragem colaborativa obteve grandes sucessos na teoria e na prática (SARWAR *et al.*, 2002; HUANG e GONG, 2008; SCHAFER *et al.*, 1999). No entanto, ainda existem diversas questões para serem pesquisadas com o fim de superar os desafios intrínsecos à filtragem colaborativa como: a esparsidade, a escalabilidade, ruído e entre outros. Para tal, a literatura divide essa abordagem em duas grandes classes: algoritmos baseados em memória e algoritmos baseados em modelo (SU e KHOSHGOFTAAR, 2009).

### Algoritmos baseados em memória

Os algoritmos baseados em memória, também são conhecidos como vizinhos mais próximos ou algoritmos baseados em heurísticas por causa das suas características (ADOMAVICIUS e TUZHILIN, 2005). Uma forma de interpreta-lo é a automatização da recomendação boca-a-boca. As pessoas buscam outros indivíduos que tenham consumido os itens como livros, filmes, artigos, restaurantes e outros; a fim de formar uma opinião contanto que a pessoa seja considerada uma fonte confiável. Através dessas, ela pode ponderar com relação à confiança pela fonte da recomendação e expectativa sobre o produto ou serviço.

O algoritmo de vizinhos mais próximos baseado no usuário pressupõe que os usuários podem ser agrupados pelas suas similaridades. Definindo $N_i(u)$ como $k$ vizinhos mais próximos do usuário $u$ que avaliaram o item $i$, $w_uv$ como a similaridade entre os usuários $u$ e $v$ e $\tilde{r}_{ui}$ a previsão da nota do usuário $u$ para $i$. Desta forma, a previsão será dada pela média ponderada das notas dos usuários que estão contidos em $N_i(u)$ ponderadas pelas similaridades, como mostra a Equação 3.2.

$$\tilde{r}_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{3.2}$$

Paralelamente, o algoritmo de vizinhos mais próximos baseado no item pressupõe que os itens podem ser agrupados pelas suas similaridades. Definindo como $N_u(i)$ como $k$ vizinhos mais próximos do item $i$ que foram avaliaram o usuário $u$ e $w_ij$ como a similaridade entre os itens $i$ e $j$ Desta forma, a previsão será dada pela média ponderada das notas destes itens que estão contidos em $N_u(i)$ como acentua a Equação 3.3, abaixo discriminada.

$$\tilde{r}_{ui} = \frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{j \in N_u(i)} |w_{uj}|} \tag{3.3}$$

Um problema desta abordagem é que os usuários possuem formas diferentes de avaliar os itens. Cada usuário ou item possui uma tendência com relação as suas avaliações. Na literatura, a normalização pela média central é a solução mais utilizada. O objetivo da média central é utilizar a variação com relação à média das notas do usuário ($\bar{r}_u$) ou do item ($\bar{r}_i$) como o valor na qual se deseja prever. Assim, podemos reescrever as equações utilizando a média central, no caso do baseado no usuário (3.4) e do item (3.5).

$$\tilde{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \eta_i(u)} w_{uv}(r_{vi} - \bar{r}_u)}{\sum_{v \in \eta_i(u)} |w_{uv}|} \quad (3.4) \qquad \tilde{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in \eta_u(i)} w_{ij}(r_{uj} - \bar{r}_j)}{\sum_{j \in \eta_u(i)} |w_{uj}|} \quad (3.5)$$

O fator chave na recomendação por vizinhos mais próximos é a similaridade. A seleção e o grau da importância dos vizinhos se dá através dela e esses serão utilizados para a previsão. Devido a esses fatores, a escolha do método de similaridade impacta na precisão e no desempenho do algoritmo de recomendação (RICCI *et al.*, 2011). Existem diversas abordagens para o cálculo da similaridade entre dois usuários ou dois itens. Na maioria dos casos, a similaridade é baseada nas notas que foram co-avaliadas (ADOMAVICIUS e TUZHILIN, 2005). Existem dois métodos comuns na literatura: a correlação baseada em Pearson (SHARDANAND e MAES, 1995)

$$w_{xy} = sim(x,\, y) \;=\; \frac{\sum_{s \,\in\, S_{xy}} (r_{xs} \;-\; \bar{r}_x)\,(r_{ys} \;-\; \bar{r}_y)}{\sqrt{\sum_{s \,\in\, S_{xy}} (r_{xs} \;-\; \bar{r}_x)^2 \sum_{s \,\in\, S_{xy}} (r_{ys} \;-\; \bar{r}_y)^2}} \qquad (3.6)$$

e pelo cosseno (LINDEN *et al.*, 2003)

$$sim(x,\, y) \;=\; cos(\vec{x},\, \vec{y}) \;=\; \frac{\vec{x} \,\bullet\, \vec{y}}{\|\vec{x}\| \;\times\; \|\vec{y}\|} \;=\; \frac{\sum_{s \,\in\, r_{xy}} r_{xs}\; r_{ys}}{\sqrt{\sum_{s \,\in\, S_{xy}} r_{xs}^2}\; \sqrt{\sum_{s \,\in\, S_{xy}} r_{ys}^2}}, \qquad (3.7)$$

onde $S_{xy}$ é o conjunto de itens que foram co-avaliados pelos usuários $x$ e $y$. No caso da similaridade de dois itens $S_{xy}$ é o conjunto dos usuários que deram avaliações para os itens $x$ e $y$.

### Algoritmos baseados em modelo

Em contraste aos algoritmos baseados em memória, os algoritmos baseados em modelo utilizam as avaliações com intuito de aprender um modelo matemático para identificar os padrões complexos das notas. Em 2006, essa classe de algoritmos ganhou relevância após a competição feita pela Netflix[4]. Nesta competição, o obje-

[4]http://www.netflix.com

tivo era melhorar o algoritmo que era utilizado pela empresa e assim avançando o estado-da-arte dos algoritmos de recomendação. Nela os melhores resultados foram alcançados pelos algoritmos que utilizam fatoração (KOREN *et al.*, 2009).

FUNK (2006) propôs um método que foi inspirado nas técnicas de processamento de linguagem natural, que se utiliza de um modelo de regressão para realizar predições. Para tal, ele emprega a técnica decomposição em valores singulares (SVD) para a construção. Por conseguinte, surgiu uma nova classe de algoritmos baseada em modelos que usam o conceito de variáveis latentes para a previsão de notas.

Variáveis latentes são variáveis que não são observadas diretamente e são inferidas através de um modelo matemático (DUMAIS, 2005). Algoritmos que utilizam variáveis latentes tentam explicar a preferência de um item para um usuário caracterizando-se por variáveis que, a princípio, não são observadas. Desta forma, será definido um modelo que irá estimá-las. Exemplificando, no caso de um sistema de recomendação de filmes podemos interpretar os valores latentes de um filme como o quanto este se caracteriza com relação a uma categoria. Já no caso do usuário seria a preferência deste sobre as categorias. Como é exemplificado na Figura 3.2.

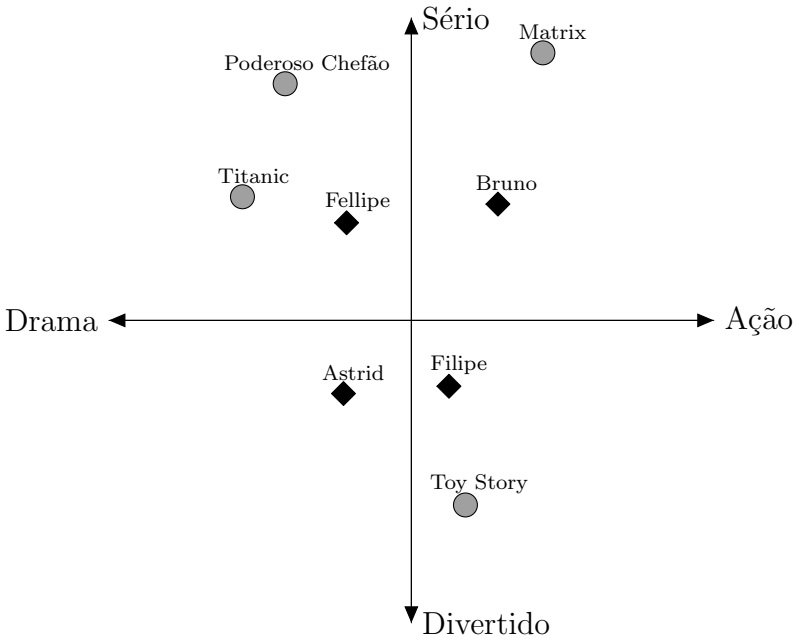



Figura 3.2: Ilustração das variáveis latentes no contexto de filmes. (KOREN *et al.*, 2009)

O método proposto por FUNK (2006) foi batizado por PATEREK (2007) como *Regulared SVD*. Ele utiliza a saída do algoritmo de decomposição em valores singulares (SVD) para a inicialização do modelo em que a matriz de notas $m \times n$ é decomposta em $PSQ^t$, sendo que $P$ é $m \times m$, $Q$ é $n \times n$ e $S$ são os valores singulares $m \times n$. Podemos manter as $k$ primeiras colunas e obter uma matriz original aproxi-

mada. Elas são interpretadas como as $k$ variáveis latentes do usuário ($P_k$) e do item ($Q_k$), sendo que $p_u$ as $k$ variáveis latentes do usuário $u$ e $q_i$ as $k$ variáveis latentes do item $i$. Assim, FUNK (2006) assume-se que a nota de um usuário $u$ para um item $i$ pode ser aproximada pela multiplicação dos fatores latentes de cada um.

$$\tilde{r}_{ui} = q_i^t p_u \tag{3.8}$$

Com o modelo das notas, é necessário encontrar $p$ e $q$ que minimizem o erro entre a nota dada pelo modelo e a nota que o usuário deu ao item. Neste trabalho, o autor utilizou o algoritmo SVD como inicialização e realizou um procedimento de aprendizado utilizando o algoritmo do gradiente descendente com o objetivo de minimizar a soma do erro quadrático. No trabalho foi proposta a utilização da constante $\lambda$ com o valor de 0.02 para a regularização da função e a função de minimização pode ser vista na Equação 3.9, sendo que $R$ o conjunto de todas as avaliações.

$$\arg\min_{q_*,p_*} \sum_{(u,i,r)\in R} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{3.9}$$

Com o modelo definido, o objetivo é encontrar $P$ e $Q$ que minimizem o erro quadrático das previsões de todas as avaliações. Desta forma, foi proposta a utilização do algoritmo do gradiente descendente para realização dessa busca. Durante cada etapa da busca, para cada avaliação do conjunto de treinamento, será realizada a correção dos parâmetros com relação ao erro total calculado. No trabalho em tese, foi proposta a utilização da constante com o objetivo de controlar a taxa de aprendizado ($\gamma$) e o valor utilizado foi de 0.001. Na Equação 3.10 está o cálculo do erro e nas Equação 3.11 e Equação 3.12 a atualização do $p$ e $q$ respectivamente.

$$e_{ui} = r_{ui} - q_i^T p_u \tag{3.10}$$

$$q_i \leftarrow q_i + \gamma \times (e_{ui} p_u - \lambda q_i) \tag{3.11}$$

$$p_u \leftarrow p_u + \gamma \times (e_{ui} q_i - \lambda p_u) \tag{3.12}$$

PATEREK (2007) estendeu o trabalho de FUNK (2006) alterando a função objetivo e adicionando a tendência do usuário e do item ao modelo e o denominou de *Improved Regularized SVD*. Em muitos sistemas de recomendação nota-se que os itens e os usuários possuem uma tendência (*bias*) independentes de qualquer interação, ou seja, um item que possui uma nota baixa tende a receber notas baixas dos demais usuários.

Assim, a ideia do modelo é expressar a nota além da interação das variáveis latentes do usuário e do item. Existe uma porção da nota que é explicada pela

tendência do usuário e do item com relação a média global inerente aquele sistema de recomendação complementando a porção explicada pelas variáveis latentes. A Equação 3.13 mostra o *bias* do usuário por item $b_{ui}$ tal que $\mu$ é a média global e $b_u$ e $b_i$ é o *bias* associado ao usuário e o item respectivamente.

$$b_{ui} = \mu + b_u + b_i \tag{3.13}$$

Podemos ilustrar esse modelo da seguinte forma: no contexto de filmes, queremos prever a nota da usuária Astrid sobre o filme *Matrix*. A média global do sistema é de 3.6, e esse filme, devido ao seu grande sucesso, possui uma tendência de receber notas 0.5 acima da média. Já a usuária Astrid, normalmente muito crítica, possui uma tendência negativa de 0.3. Nesse exemplo, a tendência relativa a esse usuário sobre esse filme é de 3.8. A Equação 3.14 mostra a nota com a tendência.

$$\tilde{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{3.14}$$

Estendendo o modelo de aprendizado aos novos parâmetros, temos a nova função objetivo:

$$\underset{b_*,q_*,p_*}{\arg\min} \sum_{(u,i,r)\in R} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\mu + b_u + b_i + ||q_i||^2 + ||p_u||^2). \tag{3.15}$$

Valendo-se da mesma técnica proposta por FUNK (2006) que utiliza o gradiente descendente para a resolução da função objetivo, tem-se:

$$b_i \leftarrow b_i + \gamma \times (e_{ui} - \lambda b_i) \tag{3.16}$$

$$b_u \leftarrow b_u + \gamma \times (e_{ui} - \lambda b_u) \tag{3.17}$$

$$q_i \leftarrow q_i + \gamma \times (e_{ui} p_u - \lambda q_i) \tag{3.18}$$

$$p_u \leftarrow p_u + \gamma \times (e_{ui} q_i - \lambda p_u) \tag{3.19}$$

Devido à configuração do problema da filtragem colaborativa, a aplicação de técnicas de aprendizado de máquina torna-se não trivial. Nesse tipo de técnica, espera-se ter como entrada uma amostra representada por um conjunto de atributos. A problemática esbarrada aqui, está na definição de quais atributos que irão representar um usuário e um item, sendo que nesse contexto, o único conhecimento é a nota.

BRAIDA *et al.* (2015) propuseram uma metodologia para transformar o problema da filtragem colaborativa em um problema de Aprendizado Supervisionado denominado *Collaboraive Filtering to Supervised Learning* (COFILS). Essa transformação consiste em um conjunto de operações que visam construir um novo

espaço de características onde estariam os usuários e os itens. Assim, a previsão de uma nota utilizando esse novo espaço, se torna um problema de reconhecimento de padrão e consequentemente sendo possível a aplicação das técnicas de aprendizado de máquina.

Aprendizado de máquina profundo (*deep learning*) vem sendo utilizado com grande sucesso em reconhecimento de fala, visão computacional e processamento de linguagem natural. As técnicas de aprendizado profundo cada vez mais estão ganhando atenção pela área de sistemas de recomendação devido o seu desempenho e gerando recomendações de alta qualidade. Esse tipo de aprendizado é capaz de capturar efetivamente as relações não-lineares dos itens com os usuários. (ZHANG *et al.*, 2017)

Em SEDHAIN *et al.* (2015a), propuseram um modelo baseado em Autoencoders e o chamaram de AutoRec. Esse modelo utiliza um vetor parcial de usuários ou um de itens como entrada, afim de reconstruí-lo na saída minimizando diretamente o RMSE. Já BARBIERI *et al.* (2017), utilizaram o Autoencoder como técnica para extração das variáveis latentes e utilizaram a abordagem COFILS para a criação de um modelo supervisionado para recomendação superando os algoritmos clássicos da filtragem colaborativa.

A recomendação possui como característica a interação de duas entidades: usuários e itens. Independentemente da categoria da recomendação, sempre existirá essa interação de duas vias entre esses dois elementos. Assim, HE *et al.* (2017) propuseram uma rede neural dupla para modelar essa interação e o denominaram *Neural Collaborative Filtering* (NCF). O NCF é um *framework* que tenta capturar a relação não-linear entre essas duas entidades e no seu trabalho os autores propuseram dois modelos.

Primeiramente eles propuseram uma forma mais geral do NCF utilizando uma *multi-layer perceptron* (MLP) para aprender a relação não-linear. A camada da entrada consiste em dois vetores onde cada um descreve um usuário $u$ e um item $i$ respectivamente. É utilizado apenas o identificador único do usuário e do item como recurso da entrada, transformando-o em um vetor binário esparso e assim o codificando. Após essa camada, existe uma camada para incorporação (*embedding layer*) para cada um dos vetores e ela é conectada totalmente com a camada esparsa. Ela representa as variáveis latentes do usuário e do item.

As camadas de incorporação do usuário e do item serão utilizadas como entrada para uma arquitetura de rede neurais com múltiplas camadas de neurônios, mapeando as variáveis latentes em previsões de notas. Dependendo da complexidade do problema, a arquitetura pode ser definida para melhor se adequar ao problema. Diferentemente da metodologia COFILS, esse modelo aprende as variáveis latentes em conjunto com a rede neural em si.

Além do MLP, HE *et al.* (2017) propõem outra instanciação do *framework* onde tenta imitar a ideia dos modelos de fatoração de matriz e eles o denomina *Generalized Matrix Factorization* (GMF). Nesse modelo as duas camadas de incorporação são multiplicadas, elemento por elemento. Esse novo vetor que será utilizado como entrada a uma arquitetura de rede neurais. Esses modelos podem ser utilizados em conjunto, dando mais informação à rede neural sobre a relação entre um usuário e um item. Os dois modelos podem ser vistos nas figuras 3.3 e 3.4.

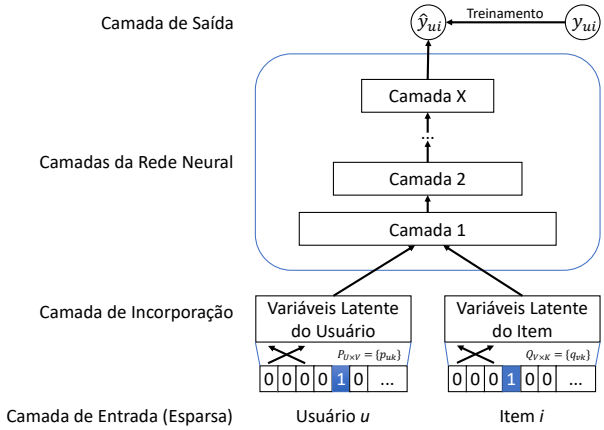



Figura 3.3: *Multi-Layer Perceptron* do *framework* proposto por HE *et al.* (2017) chamado *Neural Collaborative Filtering*.

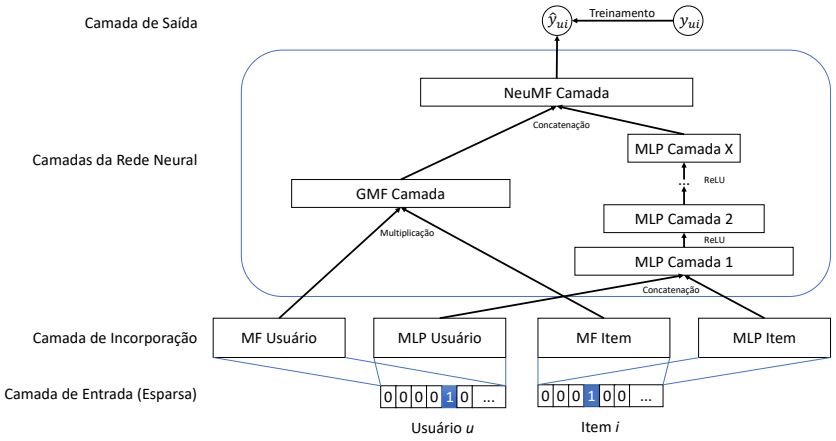



Figura 3.4: *Generalized Matrix Factorization* do *framework* proposto por HE *et al.* (2017) chamado *Neural Collaborative Filtering*.

## 3.2 Ruído Natural

Conforme mostrado nos capítulos anteriores, atualmente está havendo um crescimento exponencial de usuários que se utilizam de sistemas de comércio eletrônico. Esses sites oferecem um grande número de produtos e serviços a seus usuários. Essa enorme quantidade de opções sobrecarregam os consumidores nas suas decisões de

compra ou consumo. Os Sistemas de Recomendação têm desempenhado um papel importante neste contexto (ADOMAVICIUS e TUZHILIN, 2005).

A maioria das pesquisas realizadas em Sistemas de Recomendação possuem o foco na melhoria dos algoritmos de previsão, com o objetivo de aprimorar a acurácia (RICCI *et al.*, 2011). Houve uma grande evolução dos algoritmos após a competição *Netflix Prize*, demonstrando que havia muitas oportunidades para melhorar significativamente a acurácia dos algoritmos (KOREN *et al.*, 2009). Recentemente, surgiram algumas pesquisas com temas correlatos ao problema de previsão que influenciam diretamente o desempenho dos recomendadores, *e.g. cold start* e esparsidade.

No entanto, todas as pesquisas realizadas consideram que os dados possuem nenhuma inconsistência. Como todo processo de aquisição de dados, todas as informações possuem um grau de inconsistência e o pré-processamento é um dos passos fundamentais no processo de mineração de dados. Recentemente surgiram algumas pesquisas que apontam problemas na qualidade das notas feitas pelos usuários (AMATRIAIN *et al.*, 2009a; PHAM e JUNG, 2013). AMATRIAIN *et al.* (2009a) mostraram que o processo de elicitação de notas não é isento de erros, portanto pode possuir ruído.

### 3.2.1 Tipos de Ruídos

Em O'MAHONY *et al.* (2006), categorizou o ruído em Sistemas de Recomendação em:

- **Ruído Malicioso**: associado ao ruído que foi introduzido intencionalmente por um agente externo gerando um resultado enviesado para o recomendador;
- **Ruído Natural**: o ruído foi introduzido pelo usuário naturalmente no processo de elicitação, e ele pode afetar o resultado da recomendação.

O ruído malicioso é uma área de pesquisa já consolidada (GUNES *et al.*, 2012). Já o ruído natural, é um tópico recente de pesquisa que está ganhando cada vez mais importância na área, e saliente-se, foi introduzido por O'MAHONY *et al.* (2006). De modo geral, os algoritmos propostos para amenizar o primeiro caso de ruído estão associados à busca de alguns padrões nos perfis dos usuários para detectar um possível ataque (GUNES *et al.*, 2012). Já no ruído natural, a identificação é mais difícil, já que ele tende aparecer em diversas formas (LI *et al.*, 2013). Por esta razão, as técnicas de processamento dos dois tipos de ruídos são diferentes.

No caso do ruído natural, diversos autores sugerem que a nota não deve ser considerada como o valor que expressa o real gosto do usuário pelo item, pois o processo de elicitação das preferências é intrinsecamente ruidoso (AMATRIAIN *et al.*, 2009a). Assim, AMATRIAIN *et al.* (2009a) e PHAM e JUNG (2013) esboçaram

duas possíveis razões para que o ruído natural ocorra nos conjuntos de dados de Sistemas de Recomendação: 1. o fato de que as preferências dos usuários podem mudar com o tempo, e 2. o fato de que existe uma imprecisão inerente ao processo de elicitação das notas.

No primeiro caso, a filtragem colaborativa é caracterizada como um problema de *concept drift* no mundo real (DING *et al.*, 2006). Tanto o usuário quanto o item, mudam com o passar do tempo. Os algoritmos tradicionais não conseguem capturar essa mudança através do tempo e refletir nas recomendações. Em virtude disso, surgiu uma nova geração de algoritmos de filtragem colaborativa que consideram o efeito temporal. Essa classe de algoritmos é chamada *time-awared collaborative filtering* (RICCI *et al.*, 2010).

A causa do segundo caso é provocada por diversos fatores, tais como: condições pessoais, influências sociais, estado emocional, contexto ou até a escala da avaliação (KLUVER *et al.*, 2012; SAID *et al.*, 2012). AMATRIAIN *et al.* (2009a) realizaram alguns experimentos com os usuários para verificar a inconsistência deles analisando suas avaliações em período de tempos variados. Concluíram que em todos os casos, os usuários tendem a ser inconsistentes e justificaram os resultados com duas possibilidades: a preferência do usuário muda com o tempo e a imprecisão do usuário. No primeiro caso a inconsistência se desenvolve em um longo período de tempo se comparado com a segunda. Por fim, os resultados deste trabalho indicaram que essa inconsistência pode afetar consideravelmente a acurácia do recomendador e, por essa razão, deve ser tratada como ruído.

### 3.2.2 Algoritmos de *Data Cleansing*

Diversas propostas surgiram com o objetivo de lidar com o ruído natural. O'MAHONY *et al.* (2006) propuseram um método para filtrar as possíveis notas que são ruídos. Para tal, sugeriram um método que compara a nota que o usuário deu para um item com a nota que o modelo gerou como previsão. Se essa diferença for maior que um limite pré definido, essa avaliação sairá do conjunto de treinamento. Ao final, será realizado um novo treinamento utilizando as notas que não foram descartadas. O procedimento pode ser visto no Algoritmo 1.

**Algoritmo 1** Filtragem das avaliações com ruído proposto por O'MAHONY *et al.* (2006).

**Entrada** conjunto de notas $R$, limiar $\tau$, modelo de previsão $\varphi$.

**Saída** conjunto de notas sem ruídos $R^*$.

$R^* \leftarrow \{\}$

**for** $(u, i, r) \in R$ **do**

  $\tilde{r} \leftarrow \varphi(u, i)$

  **if** $|\tilde{r} - r| < \tau$ **then**

    $R^* \leftarrow R^* \cup \{(u, i, r)\}$

  **end**

**end**

TOLEDO *et al.* (2015) propuseram um processo para lidar com o ruído em filtragem colaborativa. Assim como o algoritmo anterior, o método proposto não utiliza nenhuma informação extra para melhorar a recomendação e pode ser definido em duas fases. A primeira irá caracterizar os usuários e os itens de acordo com as suas notas e inferir um modelo de classificação que irá avaliar se as notas são possíveis ruídos. A segunda fase é responsável pela correção da avaliação. O método pode ser visto na Figura 3.5.

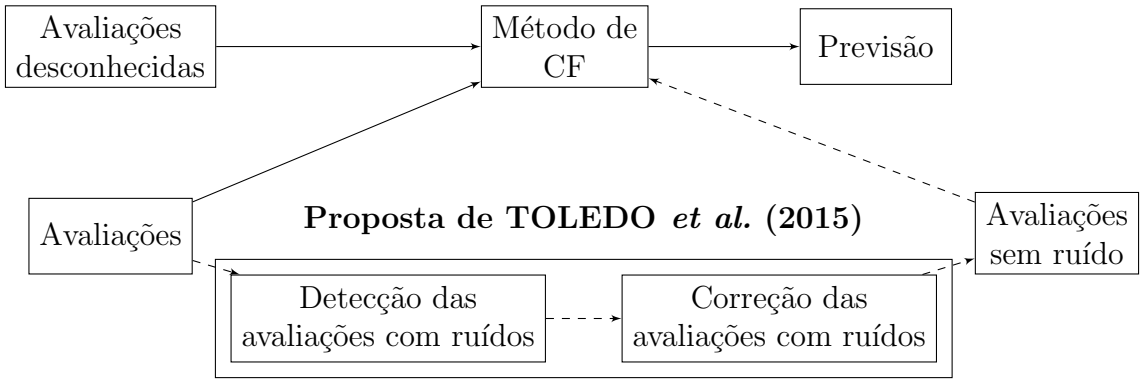



Figura 3.5: Método de TOLEDO *et al.* (2015) para detecção e correção de ruído.

Em maiores detalhes, a primeira fase é responsável pela classificação dos usuários e dos itens. A proposta considera o fato da personalidade do usuário estar relacionada diretamente com o perfil das suas notas e sob consequência direta um item tem uma preferência global de acordo com as preferências dos usuários (HU e PU, 2013; CANTADOR *et al.*, 2013). Assim, uma nota que contradiz o comportamento correspondente do usuário e do item representa um possível ruído. Os Algoritmos 2 e o 3 mostram o processo de classificação do usuário e do item respectivamente. No Algoritmo 4, por sua vez, mostra o processo de detecção dos possíveis ruídos.

**Algoritmo 2** Classificação dos usuários proposto por TOLEDO *et al.* (2015)

**Entrada** conjunto de notas $R$, limiares de classificação $\kappa_u$, $\nu_u$, conjunto dos usuários U.

**Saída** usuários classificados: usuários que dão notas baixas $W_u$, usuários que dão notas médias $A_u$ e usuários que dão notas altas $S_u$.

$W = \{\}, A = \{\}, S = \{\}$
**for** $(u, i, r) \in R$ **do**
  **if** $r < \kappa_u$ **then**
    $W \leftarrow W \cup \{(u, i, r)\}$
  **else if** $r \geq \kappa_u \wedge r < \nu_u$ **then**
    $A \leftarrow A \cup \{(u, i, r)\}$
  **else**
    $S \leftarrow S \cup \{(u, i, r)\}$
  **end**
**end**
$W_u = \{\}, A_u = \{\}, S_u = \{\}$
**for** $u \in U$ **do**
  **if** $|W| \geq |A| + |S|$ **then**
    $W_u \leftarrow W_u \cup \{u\}$
  **end**
  **if** $|A| \geq |W| + |S|$ **then**
    $A_u \leftarrow A_u \cup \{u\}$
  **end**
  **if** $|S| \geq |W| + |A|$ **then**
    $S_u \leftarrow S_u \cup \{u\}$
  **end**
**end**

**Algoritmo 3** Classificação dos itens proposto por TOLEDO *et al.* (2015).

**Entrada** conjunto de notas $R$, limiares de classificação $\kappa_i$ e $\nu_i$, conjunto dos itens $I$.

**Saída** itens classificados: itens que recebem notas baixas $W_i$, itens que recebem notas médias $A_i$ e itens que recebem notas altas $S_i$.

$W \leftarrow \{\}, A \leftarrow \{\}, S \leftarrow \{\}$
**for** $(u, i, r) \in R$ **do**
  **if** $r < \kappa_i$ **then**
    $W \leftarrow W \cup \{(u, i, r)\}$
  **else if** $r \geq \kappa_u \wedge r(u, i) < \nu_i$ **then**
    $A \leftarrow A \cup \{(u, i, r)\}$
  **else**
    $S \leftarrow S \cup \{(u, i, r)\}$
  **end**
**end**
$W_i = \{\}, A_i = \{\}, S_i = \{\}$
**for** $i \in I$ **do**
  **if** $|W| \geq |A| + |S|$ **then**
    $W_i \leftarrow W_i \cup \{i\}$
  **end**
  **if** $|A| \geq |W| + |S|$ **then**
    $A_i \leftarrow A_i \cup \{i\}$
  **end**
  **if** $|S| \geq |W| + |A|$ **then**
    $S_i \leftarrow S_i \cup \{i\}$
  **end**
**end**

**Algoritmo 4** Detecção dos possíveis ruídos proposto por TOLEDO *et al.* (2015).

**Entrada** conjunto de notas $R$, limiar de classificação $\kappa$, $\nu$, usuários classificados ($W_u$, $A_u$ e $S_u$) e itens classificados ($W_i$, $A_i$ e $S_i$).

**Saída** conjunto de notas com possibilidades de serem ruídos $R^{'} \subseteq R$.

$R^{'} \leftarrow \{\}$

**for** $(u, i, r) \in R$ **do**

  **if** $(u \in A_u) \wedge (i \in W_i) \wedge (r \geq \kappa)$ **then**

    $R^{'} \leftarrow R^{'} \cup \{(u, i, r)\}$

  **end**

  **if** $(u \in A_u) \wedge (i \in A_i) \wedge ((r < \kappa) \vee (r \geq \nu))$ **then**

    $R^{'} \leftarrow R^{'} \cup \{(u, i, r)\}$

  **end**

  **if** $(u \in S_u) \wedge (i \in S_i) \wedge (r < \nu)$ **then**

    $R^{'} \leftarrow R^{'} \cup \{(u, i, r)\}$

  **end**

**end**

Uma vez que os possíveis ruídos foram detectados na primeira etapa, a próxima fase é responsável por corrigir essas avaliações. Esse trabalho, ao invés de descartar as avaliações consideradas como ruído como no trabalho de O'MAHONY *et al.* (2006), prefere corrigi-las, como sugerido por diversos autores com o fim de evitar a perda da informação (ZHU e WU, 2004). O algoritmo é semelhante ao proposto em O'MAHONY *et al.* (2006). Nele é definido um limiar $\tau$ e se a previsão do modelo for superior a esse valor, a avaliação será substituída pela que for gerada pelo modelo, conforme visto no Algoritmo 5 abaixo relacionado.

**Algoritmo 5** Filtragem das Avaliações com Ruído proposto por TOLEDO *et al.* (2015)

**Entrada** conjunto das possíveis notas com ruído $R^{'} \subseteq R$, limiar $\tau$, modelo de previsão $\varphi$.

**Saída** conjunto de notas sem ruídos $R^{*}$.

$R^{*} \leftarrow \{\}$

**for** $(u, i, r) \in R^{'}$ **do**

  $\tilde{r} \leftarrow \varphi(u, i)$

  **if** $|\tilde{r} - r| < \tau$ **then**

    $R^{*} \leftarrow R^{*} \cup \{(u, i, r)\}$

  **else**

    $R^{*} \leftarrow R^{*} \cup \{(u, i, \tilde{r})\}$

  **end**

**end**

A maior dificuldade desse método é definir os parâmetros de classificação $\kappa_u$, $\nu_u$, $\kappa_i$ e $\nu_i$. TOLEDO *et al.* (2015) propuseram três formas. A primeira é denominada *global-pv*, em que foram definidos valores globais tanto para o usuário quanto o item. Eles propuseram um cálculo em que definem todos os parâmetros de $\kappa$ e $\nu$ utilizando o valor mínimo e máximo do conjunto de preferência $P$. Já o limiar $\tau$ é definido como a diferença de duas avaliações, ou seja, é o valor entre duas avaliações. Segue abaixo o cálculo do $\kappa$ e do $\nu$.

$$\kappa = \kappa_u = \kappa_i = min(P) + \lfloor \frac{1}{3} \cdot (max(P) - min(P)) \rceil \tag{3.20}$$

$$\nu = \nu_u = \nu_i = max(P) - \lfloor \frac{1}{3} \cdot (max(P) - min(P)) \rceil \tag{3.21}$$

As outras propostas utilizam a ideia que o usuário e o item podem ter perfis diferentes. Os parâmetros são definidos para cada usuário e para cada item, e eles são calculados utilizando a média ($\bar{x}$) e o desvio padrão ($\sigma$). Segue abaixo o cálculo do $\kappa_u$, $\nu_u$, $\kappa_i$ e $\nu_i$.

$$\kappa_u = \bar{x}_u - \sigma_u \tag{3.22}$$

$$\nu_u = \bar{x}_u + \sigma_u \tag{3.23}$$

$$\kappa_i = \bar{x}_i - \sigma_i \tag{3.24}$$

$$\nu_i = \bar{x}_i + \sigma_i \tag{3.25}$$

Para calcular o valor de $\kappa$ e $\nu$, serão adotados duas abordagens: uma baseada no usuário - *user-based-pv* ($\kappa = \kappa_u$ e $\nu = \nu_u$) e outra baseada no item - *item-based-pv* ($\kappa = \kappa_i$ e $\nu = \nu_i$). Já o limiar é definido como $\tau = \sigma_u$ no caso do baseado no usuário e $\tau = \sigma_i$ no caso do baseado no item.

No trabalho de YERA *et al.* (2016), foi introduzida uma abordagem para correção dos ruídos utilizando um modelo *fuzzy*, diferentemente das outras abordagens rígidas, tal modelo é flexível com relação à incerteza inerente ao problema. Esse trabalho divide o processo em quatro partes: a) Perfil *Fuzzy* b) Detecção do Ruído c) Correção do Ruído d) Saída. O processo pode ser visto na Figura 3.6.

Na primeira fase, as avaliações, os usuários e os itens são transformados a fim de utilizarem a lógica *fuzzy* e assim removendo as incertezas das avaliações.

A segunda fase, por sua vez, é responsável pela detecção dos possíveis ruídos. Em um primeiro momento as avaliações são analisadas para determinar se a nota é elegível para etapa seguinte, que é a classificação do ruído mediante a comparação dos perfis do usuário e do item com uma função de distância para assim determinar se a avaliação faz sentido, dados os perfis.

Após os ruídos serem detectados, é passado para $3^a$ etapa, a de correção. Dife-

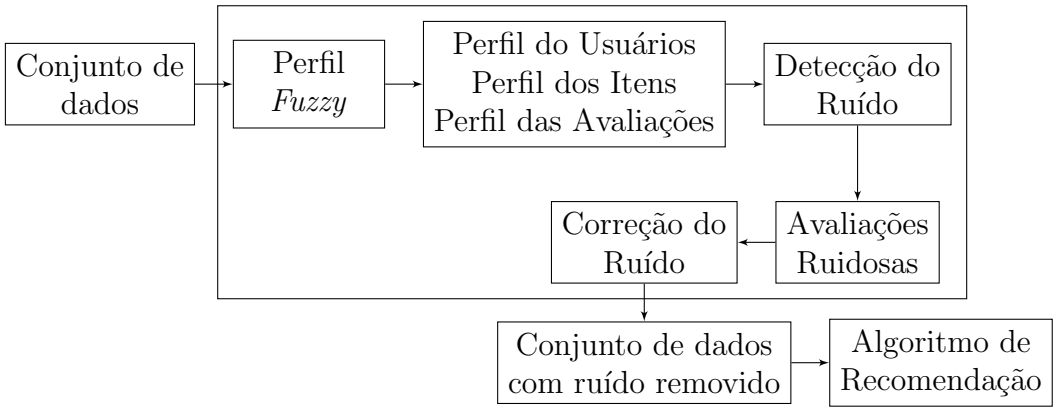



Figura 3.6: *Framework* proposto por (YERA *et al.*, 2016) para detecção e correção de ruído utilizando modelo *fuzzy*.

rentemente das propostas anteriores, essa proposta não só identifica o ruído como também, informa o grau dele. Assim, a correção se torna mais flexível e adaptável. Para tal, esse trabalho utiliza a dissimilaridade normalizada de Manhattan entre os perfis para informar o seu grau.

Uma vez que é conhecido o grau do ruído da avaliação $\mathfrak{n}_{r_{ui}}$, a proposta irá prever a nota $\tilde{r}_{ui}$ utilizando um modelo tradicional de previsão de filtragem colaborativa. Finalmente, a última etapa, a avaliação ruidosa será corrigido por:

$$r_{ui} = r_{ui} \cdot (1 - \mathfrak{n}_{r_{ui}}) + \tilde{r}_{ui} \cdot \mathfrak{n}_{r_{ui}} \tag{3.26}$$

Após esses procedimentos o conjunto de dados estaria sem ruído e pronto para ser utilizado por um recomendador.

# Capítulo 4

# Proposta: Modelo Robusto ao Ruído Natural para Filtragem Colaborativa

*In the moment when I truly understand my enemy, understand him well enough to defeat him, then in that very moment I also love him.*
— Orson Scott Card, Ender's Game

Este capítulo descreve uma proposta de um modelo robusto para a filtragem colaborativa. Inicialmente, são discutidas as motivações para este trabalho detalhando a adversidade do ruído natural na filtragem colaborativa em conjunto com a discussão das soluções de *data cleansing*. Logo depois, são discutidas quais características assumidas por essas soluções não são verdadeiras no contexto da filtragem colaborativa.

Neste mesmo capítulo é proposto um modelo robusto para a filtragem colaborativa utilizando funções de custos modificadas proposta por PATRINI *et al.* (2016a). Os dois procedimentos de correção são apresentados em conjunto com sua sustentação teórica que demonstra sua robustez. Contudo, essas correções precisam da matriz de transição de ruído que não é conhecida no problema. Portanto, é proposto também um método para estimá-la para a filtragem colaborativa.

## 4.1 Motivação

Nos trabalhos TOLEDO *et al.* (2015); YERA *et al.* (2016); O'MAHONY *et al.* (2006), os autores propuseram algoritmos para lidar com o ruído sem nenhum conhecimento adicional. Para validar as propostas, aplicaram algoritmos clássicos de filtragem colaborativa em um conjunto de dados objetivando aumentar a acurácia


Figura 4.1: Ilustração da filtragem colaborativa e possíveis cenários utilizando modelos gráficos probabilísticos; (a) filtragem colaborativa, (b) modelos de filtragem e correção de ruído natural no problema da filtragem colaborativa.

após a aplicação dos algoritmos propostos de *data cleansing*.

Existe uma dificuldade inerente à construção de uma heurística para a realização de *data cleansing* na filtragem colaborativa. De um modo geral, os conjuntos de dados não possuem mais de uma avaliação do mesmo usuário para um item, impossibilitando a construção de um modelo que verifique a verossimilhança da nota. O objetivo das heurísticas é o de construir um modelo que se aproxime de um modelo hipotético que seja capaz de verificar se a avaliação em questão faz sentido. Caso não seja, a avaliação será marcada como um possível ruído.

A abordagem por filtragem colaborativa pode ser ilustrada na figura 4.1a. Note que existe uma dependência entre o usuário e o item, ou seja, a avaliação de um usuário para um item é única e depende das duas entidades. Como essa dependência é cíclica, não é possível modelar diretamente. As heurísticas propostas na literatura modelam esse cenário conforme ilustrado na figura 4.1b, em que não existe dependência entre o usuário e o item.

Tal modelagem pode ser ilustrada no cenário de filmes, no caso em que um usuário não goste de filmes de terror e os avalie negativamente. Em contrapartida, para as demais categorias, ele emita notas altas, avaliando positivamente. Nas heurísticas citadas, se o usuário avaliar um filme de terror famoso, *i.e.* que costuma ter boas avaliações com nota muito baixa, as propostas irão marcar essa avaliação como um possível ruído. Nesse caso, as heurísticas não analisam o usuário e o item em conjunto para verificar se nesse caso é ruído. Ou seja, elas estão analisando individualmente os padrões de cada entidade e não estão utilizando os demais dados para enriquecer essa busca do padrão, inclusive abandonando o princípio da filtragem colaborativa. Essa análise pode ser feita também para o item com relação ao usuário.

Outra questão é que esses trabalhos não avaliam se os algoritmos propostos realmente estavam eliminando o ruído ou simplesmente eliminando os elementos de difícil previsão, diminuindo a cobertura e aumentando a acurácia. Não obstante, não existe nenhuma evidência que pequenos melhoramentos no RMSE podem impactar na qualidade das recomendações (KOREN, 2008; EKSTRAND *et al.*, 2014).

Além disso, existe um impedimento na realização desse tipo de avaliação no que tange ao desempenho dos algoritmos de detecção de ruído em bases de filtragem colaborativa. De modo geral, as bases disponíveis não possuem a informação se a avaliação é ou não ruído. Esse tipo de impasse é comum na literatura de ruído e nesse ínterim, existem diversas propostas para gerar artificialmente os três tipos de ruídos (NCAR, NAR e NNAR).

No caso do ruído natural na filtragem colaborativa, AMATRIAIN *et al.* (2009a) e PHAM e JUNG (2013) listaram duas opções possíveis para que o ruído natural aconteça. Na primeira, tanto o usuário quanto o item podem mudar o seu conceito com o passar do tempo, influenciando e alterando a sua visão sobre os itens. Cita-se como exemplo, um usuário que não gostava na adolescência de rock, mas passa a gostar na sua fase adulta. No que concerne ao item, cite-se também, pessoas que perdem o interesse em comprar determinado objeto tecnológico, pois surgiu algo mais moderno que trouxe novas funcionalidades.

O segundo motivo decorre da inconsistência natural do usuário. Nesse caso, podem existir inúmeras razões para a ocorrência dessa inconsistência, pode-se elencar como exemplos: condições pessoais, influências sociais, escala e ordem de avaliação, condições externas ambientais, dentre outros. Os sistemas de recomendação tradicionais concentram as suas recomendações nos itens mais relevantes ao usuário e não levam em conta as informações do contexto. Dependendo da aplicação, a informação do contexto é essencial. Destaca-se como exemplo também o domínio de viagem, que fez com que surgisse a sub-área chamada *Context-Aware Recommender System* (CARS) (BALTRUNAS, 2011).

Um fator que não foi levado em consideração pelos autores é a possibilidade de mais de uma pessoa utilizar o mesmo perfil, já que, geralmente os algoritmos assumem que um usuário é um único indivíduo. Esse tipo de questão é bem comum, por exemplo, no sistema de recomendação de *streaming* de filmes e seriados (*Netflix*), em que as pessoas compartilham as suas contas. Dessa situação, cabe mencionar que há duas possibilidades de ocorrência: intencional ou não. Será intencional quando, por exemplo, as pessoas compartilharem o serviço, devido ao valor. Em contrapartida, será não intencional quando, por exemplo, os pais exibirem filmes em determinadas horas para entreter crianças e, em outras horas, façam o uso para fim próprio.

Dependendo do domínio da aplicação e regras de negócio, tais ruídos podem ocorrer com mais frequência. Esse impasse é tão grave que a *Netflix* desenvolveu uma funcionalidade no seu sistema em que o usuário pode criar perfis na sua conta, possibilitando também que o mesmo troque o perfil quando for outra pessoa que estiver utilizando. A Figura 4.2 ilustra a ontologia das possíveis causas do ruído natural.

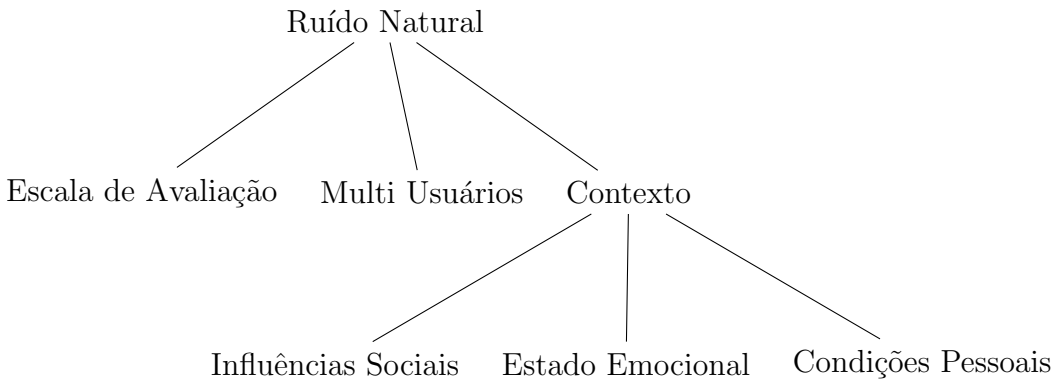



Figura 4.2: Ontologia das possíveis causas do ruído natural.

Em todos os casos, fica evidente que o ruído natural depende do usuário ou item e, no contexto real de uma aplicação, ele pode ser classificado como ruído não aleatório (NNAR). Além dos fatores citados acima, por causa da estrutura do problema da filtragem colaborativa, o impacto do ruído é maior do que em problemas de aprendizado supervisionados clássicos.

A ideia central da filtragem colaborativa é simular a recomendação boca a boca, ou seja, encontrar padrões nas avaliações dos usuários sobre os itens e utilizá-los para prever outras interações desconhecidas. Essa estrutura é diferente da estrutura de um problema de aprendizado supervisionado em que a classe não é utilizada para previsão de outras instâncias e os atributos são os mesmos e limitados. No primeiro caso, uma avaliação ruidosa pode perturbar um grande número de relações de forma indireta e assim afetando usuários e itens que não estão diretamente relacionados ao ruído em si.

Tendo em vista os problemas discutidos anteriormente, é possível concluir que a filtragem colaborativa é inerentemente ruidosa. Qualquer perturbação pode influenciar diretamente na qualidade das recomendações ao usuário, fazendo-se necessário que algoritmos de previsão sejam robustos, ou seja, menos sensíveis a um possível ao ruído.

Na literatura de ruído de classe existem três metodologias para lidar com esse problema (FRENAY e VERLEYSEN, 2013). A primeira abordagem infere que certos algoritmos de aprendizado são naturalmente menos sensíveis do que outros em relação ao ruído de classe. Existem diversos estudos que demonstram que alguns algoritmos são menos influenciados pelo ruído de classe do que outros. Entretanto, o ruído não é considerado em nenhum momento nesse tipo de abordagem e sim sua capacidade de evitar *overfitting*.

Outra forma de lidar com ruído é utilizar métodos que melhorem a qualidade dos dados. Nesse caso, os ruídos são identificados e tratados antes do treinamento do modelo, podendo ser filtrados ou corrigidos de acordo com algum outro modelo.

Essa abordagem é fácil e barata de ser implementada, mas em compensação, pode remover desnecessariamente uma grande quantidade de dados.

Por último, quando o ruído é incorporado diretamente ao modelo ou o algoritmo de aprendizado é modificado para considerá-lo, separando assim o modelo de ruído do modelo de classificação. A vantagem desse tipo de abordagem é que, diferentemente da anterior, não há necessidade de eliminar nenhum dado ou separar o modelo de classificação com uma heurística de detecção de ruído. Isto permite a utilização de informações adicionais para aprimorar o resultado do classificador.

Em regra, os sistemas de recomendação possuem uma grande base de usuários e de itens, o que acarreta em uma matriz usuário-item esparsa. Essa esparsidade, em geral, é em torno de 1% (ADOMAVICIUS e TUZHILIN, 2005). Alguns trabalhos relacionam diretamente a esparsidade dos dados com o desempenho dos modelos (TOLEDO *et al.*, 2015). Por essa razão, diversos autores sugerem corrigir as informações anômalas ao invés de removê-las para não perder informação. No entanto, não existe nenhum trabalho ao qual o ruído seja incorporado ao modelo de previsão. Para tal, neste trabalho um algoritmo de previsão é proposto tal que seja robusto comparado aos algoritmos clássicos.

## 4.2 Modelo Robusto

Em aprendizado de máquina, o aprendizado de modelos consiste de um problema de otimização. Busca-se nessa otimização encontrar um conjunto de parâmetros associado a um conjunto de variáveis independentes (atributos de instâncias) que minimizem uma função de custo relacionada a um conjunto de variáveis dependentes a serem aprendidas.

Em geral, o processo de otimização assume que o conjunto de instâncias não apresenta erros em sua rotulação, ou seja, os dados não apresentam erros. Entretanto, é comum que dados apresentem um percentual de erro em suas rotulações. Uma questão relevante é que, se tivéssemos uma base de dados limpa e, uma segunda com os mesmos dados, porém algumas rotulações incorretas. Se fossem aprendidos os dois modelos, um em cada base, e os minimizadores forem os mesmos, podemos nomear a função de custo como robusta, o processo de aprendizado robusto e os modelos derivados como robustos.

PATRINI *et al.* (2016a) propuseram uma correção para a função de custo para problemas de classificação multi-classe em que os minimizadores são os mesmos para a função original com os dados limpos. Contudo, é necessário o conhecimento de uma taxa do ruído acerca dos dados e, no primeiro momento, será considerado que essa informação é conhecida. O objetivo é utilizá-las em conjunto com um modelo, *e.g.* rede neural, e assim gerarmos um classificador robusto para a filtragem colaborativa.

Nas próximas seções serão apresentados esses procedimentos de correção da função de custo.

### 4.2.1 Formalização

Com o objetivo de maior clareza será fixada a notação para as próximas seções que será a mesma utilizada por PATRINI *et al.* (2016b).

Será definido como $[c] = \{1, \ldots, c\}$ para qualquer $c$ inteiro positivo. Vetores serão definidos como negrito (*e.g.* $\boldsymbol{v}$) e matrizes em letra maiúscula (*e.g.* $V$). Para as coordenadas de uma matriz será utilizado o subscrito (*e.g.* $\boldsymbol{v}_j$). A linha ou uma coluna de uma matriz será denotada por um ponto no subscrito (*e.g.* $V_{j\cdot}$ e $V_{\cdot j}$ respectivamente). Para um caso especial de vetor em que todos os elementos sejam iguais a um, será utilizada a simbologia $\mathbf{1}$ e $\Delta^{c-1} \subset [0,1]^c$ o $c$-simplex

Considerando uma classificação com espaço de $c$ classes, o espaço de atributos será dado por $\mathcal{X} \subseteq \mathbb{R}^d$ e o espaço das classes como $\mathcal{Y} = \{\boldsymbol{e}^i : i \in [c]\}$. Já $\boldsymbol{e}^i$ significa o $i^{\text{o}}$ vetor da base canônica $\mathbb{R}^c$, *i.e.* $\boldsymbol{e}^i \in \{0,1\}^c, \mathbf{1}^\top \boldsymbol{e}^i = 1$. Um exemplo observado $(\boldsymbol{x}, \boldsymbol{y})$ é definido por uma distribuição desconhecida $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$ em $\mathcal{X} \times \mathcal{Y}$. A esperança de $p(\boldsymbol{x}, \boldsymbol{y})$ será dada por $\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}$.

Uma rede neural com $n$ camadas, sendo essas densas, será representada pelas transformações $\boldsymbol{h} \colon \mathcal{X} \to \mathbb{R}^c$, tal que $\boldsymbol{h} = (\boldsymbol{h}^{(n)} \circ \boldsymbol{h}^{(n-1)} \circ \cdots \circ \boldsymbol{h}^{(1)})$ que corresponde a composição das transformações das camadas intermediárias e é definida por:

$$(\forall i \in [n-1])\, \boldsymbol{h}^{(i)}(\boldsymbol{z}) = \boldsymbol{\sigma}(W^{(i)}\boldsymbol{z} + \boldsymbol{b}^{(i)}) \ .$$

em que $W^{(i)} \in \mathbb{R}^{d^{(i)} \times d^{(i-1)}}$ e $\boldsymbol{b}^{(i)} \in \mathbb{R}^{d^{(i)}}$ são parâmetros que serão estimados e $\boldsymbol{\sigma}$ é a função de ativação e $\boldsymbol{h}(\boldsymbol{x})$ representa a saída da rede.

A função de custo contabiliza a diferença entre a classe real $\boldsymbol{y} = \boldsymbol{e}^i$ e a saída da rede, sendo definida por $\ell \colon \mathcal{Y} \times \Delta^{c-1} \to \mathbb{R}$. No caso, a função de custo entropia cruzada é definida por:

$$\ell(\boldsymbol{e}^i, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = -(\boldsymbol{e}^i)^\top \log\ \hat{p}(\boldsymbol{y}|\boldsymbol{x}) = -\log\ \hat{p}(\boldsymbol{y} = \boldsymbol{e}^i|\boldsymbol{x}) \ . \tag{4.1}$$

Com objetivo de facilitar a notação, a função de custo na sua forma vetorial $\boldsymbol{\ell} :\ \Delta^{c-1} \ \to\ \mathbb{R}^c$, calculada para cada classe:

$$\boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \big(\ell(\boldsymbol{e}^1, \hat{p}(\boldsymbol{y}|\boldsymbol{x})), \ldots, \ell(\boldsymbol{e}^c, \hat{p}(\boldsymbol{y}|\boldsymbol{x}))\big)^\top \in \mathbb{R}^c \ . \tag{4.2}$$

### 4.2.2 Processo de correção *backward*

Considerando o cenário ao qual o ruído é assimétrico, ou seja, cada instância dentro de um conjunto de treinamento com sua classe associada $\boldsymbol{y}$ pode ser alterada

para $\tilde{\boldsymbol{y}} \in \mathcal{Y}$ com probabilidade $p(\tilde{\boldsymbol{y}}|\boldsymbol{y})$ e os atributos se mantenham inalterados, a probabilidade das amostras poderá ser calculada como:

$$p(\boldsymbol{x}, \tilde{\boldsymbol{y}}) = p(\tilde{\boldsymbol{y}}|\boldsymbol{x})p(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\mathcal{Y}} p(\tilde{\boldsymbol{y}}|\boldsymbol{y})p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}).$$

Assim, pode-se definir a matriz de transição de ruído $T \in [0,1]^{c\times c}$, tal que a probabilidade de classe $i$ pode ser alterada para outra $j$ de modo que a matriz $T$ é definida como

$$T_{ij} = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | y = \boldsymbol{e}^i).$$

Como propriedade, essa matriz apresenta cada linha derivada de um processo estocástico. Esta matriz não necessariamente é simétrica.

Com base nas teorias desenvolvidas por NATARAJAN *et al.* (2013) e estendidas por ROOYEN (2015), PATRINI *et al.* (2016b) foram desenvolvidos dois métodos de correção de função de custo robustas ao ruído. O objetivo da correção é adicionar a informação da matriz de transição $T$, assumindo que ela é conhecida no problema, junto a função de custo $\ell$. No teorema 1 é o primeiro método de correção chamado *backward*. O teorema foi mantido igual ao original a fim de clareza com a fonte.

**Teorema 1.** *Assuma que a matriz de transição de ruído $T$ é não-singular. Dado uma função de custo $\ell$, o procedimento de correção backward $\ell^{\leftarrow}$ é dado como:*

$$\ell^{\leftarrow}(p(\boldsymbol{y}|\boldsymbol{x})) = T^{-1}\ell(p(\boldsymbol{y}|\boldsymbol{x})) \tag{4.3}$$

*A função de custo é não-viesada caso:*

$$\forall \boldsymbol{x}, \quad \mathbb{E}_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}\ \ell^{\leftarrow}(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\ \ell(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x}))\ ,$$

*e os minimizadores são os mesmos:*

$$\arg\min_{\hat{p}(\boldsymbol{y}|\boldsymbol{x})} \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}}\ \ell^{\leftarrow}(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \arg\min_{\hat{p}(\boldsymbol{y}|\boldsymbol{x})} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\ \ell(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x}))\ .$$

*Demonstração.*

$$\mathbb{E}_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}\ \ell^{\leftarrow}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\ T\ell^{\leftarrow}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\ T\ T^{-1}\ell(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\ \ell(\hat{p}(\boldsymbol{y}|\boldsymbol{x}))\ .$$

□

A função de custo será corrigida através da combinação linear de cada linha da matriz inversa de transição de ruído ($T^{-1}$) pelo resultado da função de custo de cada probabilidade de cada classe de um valor observado. A grande dificuldade desse método é que a matriz $T$ não pode ser singular, ou seja, ela precisa ter uma

inversa, podendo ser problemático na prática. Uma solução para esse dilema seria gerar pequenas perturbações na matriz para que ela se torne não-singular.

O teorema 1 afirma que o aprendizado de um modelo utilizando a função de custo *backward* com os dados ruidosos é o mesmo do que utilizando sem ruído, ou seja, significa que minimizar o risco da função corrigida com dados ruidosos é equivalente a minimizar o risco da função original de custo com dados limpos.

### 4.2.3 Processo de correção *forward*

Outro método proposto por PATRINI *et al.* (2016a) é o processo de correção *forward* e sua proposta é corrigir a previsão diretamente na saída do preditor. O procedimento é dado como:

$$\ell^{\rightarrow}(p(\boldsymbol{y}|\boldsymbol{x})) = \ell(T^T p(\boldsymbol{y}|\boldsymbol{x})) \tag{4.4}$$

O procedimento acima terá o comportamento analisado como foi realizado no anterior, avaliando o impacto da mudança na função de custo original. Para tal, faz-se necessário as propriedades da família de funções de custo chamadas *proper composite* (REID e WILLIAMSON, 2010). Primeiramente, será necessário definir as funções *link*. Funções essas que são utilizadas para mapear um valor real para um intervalo entre $[0, 1]$. Geralmente, os algoritmos de aprendizado fazem previsões de um valor real que não necessariamente são interpretados diretamente como probabilidades. Assim, necessitando de uma função *link* para mapear esses valores em um intervalo adequado.

Supondo que exista uma função *link* inversível $\boldsymbol{\psi} : \Delta^{c-1} \rightarrow \mathbb{R}^c$, a função de custo é dita como composta (*composite loss*), e denotada por $\ell_{\boldsymbol{\psi}} : \mathcal{Y} \times \mathbb{R}^c \rightarrow \mathbb{R}$, caso ela possa ser escrita através de uma decomposição de $\boldsymbol{\psi}^{-1}$, ou seja,

$$\ell_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\ell}(\boldsymbol{y}, \boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))) \ . \tag{4.5}$$

No caso da entropia cruzada, a função *softmax* é a função *link* inversa.

As funções de custo apropriadas (*proper losses*) são funções de custo que naturalmente são utilizadas para estimação de probabilidade de classe. Quando uma função de custo é associada a essas duas características, ou seja, apropriadas e compostas, seu minimizador assume a forma particular da função de ligação aplicada às probabilidades condicionais de classe $p(\boldsymbol{y}|\boldsymbol{x})$:

$$\operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}} \ \ell_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\psi}(p(\boldsymbol{y}|\boldsymbol{x})) \ . \tag{4.6}$$

Entropia cruzada e a função quadrática são exemplos de funções de custo. O proce-

dimento *forward* irá manter os mesmos minimizadores utilizando funções de custo compostas e apropriadas. Ressalto aqui que, o teorema foi mantido igual ao original a fim de clareza com a fonte.

**Teorema 2.** *Assuma que a matriz de transição de ruído $T$ é não-singular. Dado uma função de custo composta e apropriada $\boldsymbol{\ell}_{\boldsymbol{\psi}}$, o procedimento de correção forward $\ell^{\rightarrow}$ é dado como:*

$$\boldsymbol{\ell}^{\rightarrow}{}_{\boldsymbol{\psi}}(\boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\ell}(T^{\top}\boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))) \ .$$

*Assim, os minimizadores da função de custo corrigida com dados ruidosos serão os mesmos minimizadores da função original com os dados limpos, ou seja,*

$$\operatorname*{argmin}_{\boldsymbol{h}} \ \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}} \ \ell^{\rightarrow}{}_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \operatorname*{argmin}_{\boldsymbol{h}} \ \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}} \ \ell_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) \ .$$

*Demonstração.* Primeiramente temos:

$$\ell^{\rightarrow}_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \ell(\boldsymbol{y}, T^{\top}\boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))) = \ell_{\boldsymbol{\phi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))$$

onde é definido que $\boldsymbol{\phi}^{-1} = \boldsymbol{\psi}^{-1} \circ T^{\top}$. Equivalentemente, $\boldsymbol{\phi} = (T^{-1})^{\top} \circ \boldsymbol{\psi}$ é inversível pela composição de funções inversíveis, em que o domínio é $\Delta^{c-1}$ a partir de $\boldsymbol{\psi}$ e seu contradomínio é $\mathbb{R}^c$. A última função de custo da equação 4.2.3 é, portanto, apropriada e composta com o *link* $\boldsymbol{\phi}$. Finalmente, através da equação 4.6, os minimizadores da função de custo com dados ruidosos serão

$$\operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}} \ \ell_{\boldsymbol{\phi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\phi}(p(\tilde{\boldsymbol{y}}|\boldsymbol{x})) = \boldsymbol{\psi}((T^{-1})^{\top} p(\tilde{\boldsymbol{y}}|\boldsymbol{x})) = \boldsymbol{\psi}(p(\boldsymbol{y}|\boldsymbol{x})) \ ,$$

que demonstra o Teorema dado pela Equação 4.6. □

Na prática, é utilizado $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ que aproxima $p(\boldsymbol{y}|\boldsymbol{x})$, tornando necessário construir um estimador. Por exemplo, uma rede neural que seja precisa o suficiente para tornar essa aproximação a melhor possível. Diferentemente do procedimento anterior, a robustez aplica-se somente aos minimizadores. Contudo, neste método não se faz necessário realizar a inversão da matriz de transição de ruído, tornando um fator importante na prática.

PATRINI *et al.* (2016a) relataram que mesmo com garantias não tão fortes quanto ao procedimento *backward*, o *forward* obteve resultados melhores. Eles levantaram duas possíveis causas: a primeira seria devido aos problemas numéricos gerados a partir da inversão da matriz; a segunda mudança da faixa de valores que ocorre no método *backward*. Já no *forward*, após a multiplicação da matriz de transição, os valores se mantém na mesma faixa de valores, ou seja, entre $[0, 1]$.

## 4.3 Estimando a Matriz de Transição de Ruído

Os procedimentos de correção *backward* e *forward* supõem que no problema é conhecida a matriz de transição de ruído. Em problemas reais, de modo geral, a matriz é desconhecida, o que impossibilita o uso dessas correções. Desta maneira, a matriz precisa ser estimada. Na literatura de ruído, surgiram propostas para estimar essa informação.

PATRINI *et al.* (2016a) apresentaram um método para estimar a matriz em um cenário de classificação multi-classe. Para tal, era necessário assumir certas características sobre o problema, como exemplo a existência de instâncias ao qual as chances das suas respectivas classes estarem corretas deveriam ser iguais a 100%, e existirem dados suficientes para construir um modelo em que a acurácia do estimador para uma instância ruidosa seja alta. Assim, PATRINI *et al.* (2016a) propuseram o teorema 3 para estimar a matriz de transição de ruído $T$. Ressaltamos aqui que o teorema foi mantido igual ao original a fim de obter maior clareza com a fonte.

**Teorema 3.** *Assuma que* $p(\boldsymbol{x}, \boldsymbol{y})$ *é tal que:*

1. *Se existirem exemplos perfeitos de cada classe* $j \in [c]$*, então*

$$(\exists \bar{\boldsymbol{x}}^j \in \mathcal{X}) : p(\bar{\boldsymbol{x}}^j) > 0 \wedge p(\boldsymbol{y} = \boldsymbol{e}^j | \bar{\boldsymbol{x}}^j) = 1$$

2. *dados exemplos corrompidos suficientes,* $h$ *é rica o suficiente para o modelar* $p(\tilde{\boldsymbol{y}}|\boldsymbol{x})$ *com previsão.*

*Assim segue que* $\forall i, j \in [c], T_{ij} = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \bar{\boldsymbol{x}}^i)$

*Demonstração.* Dado que (2) seja verdade, pode-se considerar que $p(\tilde{\boldsymbol{y}}|\boldsymbol{x})$ é $\hat{p}(\tilde{\boldsymbol{y}}|\boldsymbol{x})$. Para todos $j \in [c]$ e qualquer $x \in \mathcal{X}$, temos:

$$\begin{aligned} p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \boldsymbol{x}) &= \sum_{k=1}^{c} p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \boldsymbol{y} = \boldsymbol{e}^k)\ p(\boldsymbol{y} = \boldsymbol{e}^k | \boldsymbol{x}) \\ &= \sum_{k=1}^{c} T_{kj}\ p(\boldsymbol{y} = \boldsymbol{e}^k | \boldsymbol{x})\ . \end{aligned}$$

Dado (1), quando $\boldsymbol{x} = \bar{\boldsymbol{x}}^i$, $p(\boldsymbol{y} = \boldsymbol{e}^k | \bar{\boldsymbol{x}}^i) = 0$ para $k \neq i$. □

Considerando as características assumidas, o Teorema 3 afirma que cada linha da matriz $T$ pode ser estimada utilizando as probabilidades do *exemplo perfeito* de cada classe dada por um estimador probabilístico $\tilde{p}$ treinado com dados ruidosos. Esse estimador pode ser uma rede neural com uma saída para cada classe utilizando uma função *softmax*.

Considerando que devam existir *exemplos perfeitos* presentes no conjunto de dados, só será necessário utilizar o elemento que maximize o valor da probabilidade da classe correspondente. Assim, não será necessário que os dados estejam rotulados, eles precisam apenas pertencer à distribuição dos dados. Esse conjunto de exemplos será denotado por $X'$. A matriz $T$ pode ser aproximada em dois passos:

$$\bar{\boldsymbol{x}}^i = \underset{\boldsymbol{x} \in X'}{\operatorname{argmax}} \ \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^i | \boldsymbol{x}) \tag{4.7}$$

$$\hat{T}_{ij} = \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \bar{\boldsymbol{x}}^i) \ . \tag{4.8}$$

Em muitos cenários, como o de classificação de imagens, pode-se assumir como verdade a primeira característica. Nesses casos, a segunda nem sempre é verdadeira, pois depende do modelo em si e da complexidade da tarefa. PATRINI *et al.* (2016a) realizaram experimentos mostrando que em algumas bases de dados o algoritmo conseguiu estimar a matriz com um pequeno grau de erro, mas não interferindo na qualidade das previsões do classificador treinado utilizando a função de custo proposto em conjunto da matriz estimada. Por fim, os métodos de correção obtiveram um alto grau de robustez utilizando a matriz estimada quando o ruído artificial era incorporado na base. O método de estimação utilizando o argmax é descrito no algoritmo 6.

**Algoritmo 6** Estimando a matriz T através do argmax

**Entrada** conjunto de dados $X^{'}$, conjunto de classes $C$,
estimador de probabilidade $\hat{p}$.
**Saída** matriz de ruído estimada $T$.

```
for i ← 1...|C| do
    x' ← argmax_{x∈X'} p̂(ỹ = e^i|x)
    for j ← 1...|C| do
        T̂_ij ← p̂(ỹ = e^j|x')
    end
end
```

Em um problema correlato ao do ruído, YU *et al.* (2017) utilizaram o trabalho de PATRINI *et al.* (2016a) em um outro problema de aprendizado imperfeito que é o de rótulos complementares. Neste trabalho foram utilizados os métodos *backward* e *forward* para treinar uma rede neural com objetivo de prever qual a classe que uma determinada imagem ela não pertence. Contudo, para estimar a matriz, utilizaram um conjunto de cinco imagens para cada classe. Estas imagens que foram rotuladas manualmente. Diferentemente do algoritmo 6, foi utilizado não o valor máximo para cada linha da matriz e sim a média das probabilidades geradas pelo estimador para

todas as imagens da classe.

### 4.3.1 Estimando o ruído na Filtragem Colaborativa

A matriz de transição de ruído estimada através do algoritmo 6 obteve ótimos resultados em conjunto com os procedimentos *backward* e *forward* para problemas de classificação de imagens (PATRINI *et al.*, 2016a). Contudo, no problema de recomendação, particularmente na filtragem colaborativa, o algoritmo não funciona de forma adequada, pois não é possível assumir nos dados as características listadas no teorema 3.

Métodos de filtragem colaborativa têm como essência a utilização das avaliações dos usuários sobre os itens para o processo de recomendação. No geral, estes métodos operam em bases de avaliações de usuários sobre itens, tal que para a construção das avaliações, cada usuário necessita explicitar a sua preferência em um grau pertencente a subconjunto dos reais ou dos naturais. Explicitar uma preferência é uma tarefa complicada e abstrata, pois o usuário precisa quantificar um sentimento. Adicionalmente, essa escolha pode ter várias interferências e possui diversos significados o que torna o valor definido naquele momento como um grau aproximado naquele contexto. Esse valor é diferente do real e não expressa puramente a sua preferência.

Por expressar um sentimento, dependendo do conjunto de preferência, as notas podem possuir uma sobreposição entre elas. Considerando que o conjunto de preferência possui valores inteiros entre 0 e 10, um usuário pode dar uma nota baixa 2, mas o valor 1 talvez tivesse o mesmo significado para ele para aquele item, ou mesmo um valor intermediário que ele não possa usar para se expressar. Assim, a diferença entre esses dois valores se torna subjetiva, informando que ele não gostou daquele item.

Além desse problema de traduzir o sentimento em um grau, como já discutido anteriormente, existe uma inconsistência natural do próprio usuário. O valor escolhido pode variar por diversas razões, como por exemplo: condições pessoais, influências sociais, contexto e outras razões. Pode-se observar algumas razões na figura 4.2. Por fim, a essência do problema é de regressão em que as notas possuem uma ordem natural, ao contrário dos problemas abordados pelos trabalhos PATRINI *et al.* (2016b) e YU *et al.* (2017).

Desta maneira, considerando a discussão acima, a primeira condição do teorema 3 não é válida para a filtragem colaborativa. Não existe um *exemplo perfeito* por classe. A solução dada por YU *et al.* (2017) de realizar a média para minimizar o erro entre os possíveis exemplos perfeitos também não funciona. Uma solução seria calcular uma média utilizando a base completa ou re-rotular algumas avaliações e nesse subconjunto computar uma média.

No primeiro caso, calcular a média com todas as avaliações por classe não funcionaria, pois o significado de cada avaliação é inconsistente entre os usuários, sendo que é comum que seja influenciada pelo o tempo e contexto. Já no segundo caso, não seria possível rotular sem o especialista, no caso o próprio usuário. Surgiram trabalhos de re-rotulação (AMATRIAIN *et al.*, 2009a,b), entretanto, ainda sim, inviáveis em um sistema de recomendação real. Isto porque geram uma sensação de desconfiança do sistema pelo usuário. De qualquer modo, a tarefa ainda continuaria sujeita à inconsistência do usuário.

Utilizando como base o trabalho de YU *et al.* (2017), é necessário um método que seja capaz de selecionar um conjunto de avaliações para cada classe que minimize a inconsistência e assim diminua a variância do cálculo da média. Esse conjunto será denominado *conjunto dos âncoras*. Na próxima subseção será discutida uma proposta de escolher essas avaliações âncoras e assim será possível estimar a matriz de ruído para a filtragem colaborativa.

### 4.3.2 Encontrando os âncoras

Considerando que existem exemplos o suficiente para construir um estimador, o Teorema 3 afirma que se existir um *elemento perfeito* para cada classe, ou seja, um elemento que apresente 100% de chance de estar rotulado em sua classe, é possível estimar a matriz de transição de ruído. Contudo, é pouco provável que exista um elemento com essa propriedade em um conjunto de dados na filtragem colaborativa. Isso acontece por causa da natureza do problema, como discutido anteriormente.

O grau de inconsistência de uma avaliação pode variar e essa variação pode ser baixa em alguns casos. Por esta razão, acredita-se que algumas avaliações são próximas aos seus valores reais. Avaliações com essa característica serão denominadas *avaliações âncoras*. Formalmente, temos:

**Definição 1.** Seja uma avaliação $r_{uv}$ realizada por um usuário $u$ para um item $v$ e $\breve{r}_{uv}$ a real preferência dele para este item, considera-se essa avaliação como *âncora*, se somente se,

$$|\breve{r}_{uv} - r_{uv}| < \tau \ , \tag{4.9}$$

em que $\tau$ corresponde a um limiar próximo a zero.

Através dessas *âncoras*, calcula-se uma aproximação para a matriz de transição de ruído. Com o objetivo de minimizar o erro, será adotada a mesma estratégia realizada por YU *et al.* (2017) em que será utilizado um conjunto de elementos por classe ao invés de um único elemento. Será utilizado um estimador probabilístico para calcular as probabilidades de cada âncora e também utilizando a média desses valores. Será adotada essa estratégia, pois não existe uma confiança em um único

elemento. Assim, o valor flutuará menos se comparado com a estratégia original (algoritmo 6). Essa modificação pode ser vista no algoritmo 7.

**Algoritmo 7** Estimando a matriz T

**Entrada** conjunto de notas dos âncoras $\mathcal{A} \mid \mathcal{A} \subseteq R$, conjunto de preferência $P$, estimador de probabilidade $\hat{p}$.

**Saída** matriz de transição de ruído estimada $T$.

**for** $c \in P$ **do**
  $\mathcal{A}_c \leftarrow \{\forall_{(u,i,r)} \in \mathcal{A} \mid r = c\}$ **for** $w \in P$ **do**
    **for** $(u, i, r) \in \mathcal{A}_c$ **do**
      $T_{cw} \leftarrow \hat{p}(\tilde{y} = c|u, i, w) \cdot \|\mathcal{A}_c\|^{-1}$
    **end**
  **end**
**end**

Desta forma, será possível calcular a matriz de transição de ruído, entretanto existe uma dificuldade inerente na definição 1 que é o conhecimento do valor real das preferências dos usuários sobre os itens, ou seja, $\breve{r}_{uv}$ é desconhecido. Esse valor precisará ser estimado, sendo possível determinar qual é a avaliação âncora. Neste trabalho será utilizado um modelo de previsão para tal. Ele será treinado utilizando todos os dados, mesmo eles possuindo alguma inconsistência. Como no algoritmo 7, o erro será minimizado com a utilização da média. O algoritmo 8 mostra o procedimento para encontrar os âncoras.

**Algoritmo 8** Algoritmo para encontrar os âncoras para a Filtragem Colaborativa.

**Entrada** conjunto de notas $R$, limiar $\tau$, modelo de previsão $\varphi$.

**Saída** conjunto dos âncoras $\mathcal{A}$.

$\mathcal{A} = \{\}$
**for** $(u, i, r) \in R$ **do**
  $\tilde{r} \leftarrow \varphi(u, i)$
  **if** $|\tilde{r} - r| < \tau$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{(u, i, r)\}$
  **end**
**end**

## 4.4 Geração Artificial de Ruído

Não existe disponível uma base de dados para filtragem colaborativa no qual os rótulos incorretos sejam identificados. Esse problema não é exclusivo no contexto de sistemas de recomendação, existindo poucas bases com essa identificação (FRENAY

e VERLEYSEN, 2013). Por essa razão, diversas metodologias foram propostas para gerar artificialmente cada tipo de ruído. Para tal, é necessário realizar uma suposição: a base escolhida é livre de ruído (GARCÍA *et al.*, 2015).

Existem diversas soluções para a geração de bases de dados artificiais com presença de ruído (FRENAY e VERLEYSEN, 2013). No caso do modelo NCAR, a maioria dos trabalhos introduz o ruído artificial em uma instância de forma aleatória com uma probabilidade $\alpha$, e alterando sua classe por uma outra qualquer com uma probabilidade uniforme. O procedimento pode ser visto no Algoritmo 9.

**Algoritmo 9** Geração de ruído uniforme utilizando o modelo NCAR.

**Entrada** conjunto de notas $R$, conjunto de preferência $P$, taxa de ruído $\alpha$
**Saída** conjunto de notas com ruído $\tilde{R}$
$\tilde{R} = \{\}$
**for** $(u, i, r) \in R$ **do**
  Seja $x$ um elemento seguindo a distribuição $\mathcal{X} \sim U([0, 1])$
  **if** $x < \alpha$ **then**
    Seja $\tilde{r}$ um elemento seguindo a distribuição $\mathcal{X} \sim U(P \setminus r)$
    $\tilde{R} \leftarrow \tilde{R} \cup \{(u, i, \tilde{r})\}$
  **else**
    $\tilde{R} \leftarrow \tilde{R} \cup \{(u, i, r)\}$
  **end**
**end**

Surgiram diversas propostas para o ruído do tipo NAR, principalmente na área de classificação de imagens e é também chamado nesse contexto de *flip label noise* (SUKHBAATAR e FERGUS, 2014; PATRINI *et al.*, 2016b). Esse tipo de ruído pode ser representado como uma matriz de transição (equação 2.2) e a geração é dada em como montar essa matriz. Devido ao conhecimento do problema, PATRINI *et al.* (2016b) utilizaram uma matriz de transição fixa para a incorporação do ruído no conjunto de dados.

A geração do tipo *pairwise* foi introduzido por ZHU *et al.* (2003) e foi utilizada em diversos trabalhos (GARCÍA *et al.*, 2015; SÁEZ *et al.*, 2014; ZHU e WU, 2004; JINDAL *et al.*, 2017; BRUZZONE e PERSELLO, 2009; GARCIA *et al.*, 2016). Duas classes $c1$ e $c2$ são selecionadas e cada instância da classe $c1$ possui uma probabilidade $P_e$ para estar incorretamente rotulada como $c2$ e vice versa. Essa geração será adaptada para a filtragem colaborativa, em que $c1$ e $c2$ serão as classes *recomenda* e *não recomenda* respectivamente.

Será utilizada a equação 5.3 para determinar quais as notas pertencem a super classe *recomenda* e *não recomenda*. Depois de determinadas as notas, serão distribuídas entre esses valores de probabilidade $P_e$. Desta forma, a matriz de transição

de ruído utilizando a geração *pairwise* com taxa de 0.1 para um conjunto de preferência $[1, 5]$ e o valor de 4 para itens que serão recomendados ficaria da seguinte forma:

$$T = \begin{bmatrix} 0.9 & 0.0 & 0.0 & 0.05 & 0.05 \\ 0.0 & 0.9 & 0.0 & 0.05 & 0.05 \\ 0.0 & 0.0 & 0.9 & 0.05 & 0.05 \\ 0.0\overline{3} & 0.0\overline{3} & 0.0\overline{3} & 0.9 & 0.0 \\ 0.0\overline{3} & 0.0\overline{3} & 0.0\overline{3} & 0.0 & 0.9 \end{bmatrix}.$$

Com a matriz de transição é possível generalizar o algoritmo de geração de ruído, que pode ser visto no algoritmo 10. Ele pode ser utilizado tanto para o ruído uniforme quanto para o *pairwise*, só dependendo da matriz de transição.

**Algoritmo 10** Geração de ruído utilizando o modelo NAR ou NCAR.

**Entrada** conjunto de notas $R$, matriz de transição $T$

**Saída** conjunto de notas com ruído $\tilde{R}$

$\tilde{R} = \{\}$

**for** $(u, i, r) \in R$ **do**

  Seja $\tilde{r}$ um elemento seguindo a distribuição $\mathcal{X} \sim Multi(1, T_{r\cdot})$

  $\tilde{R} \leftarrow \tilde{R} \cup \{(u, i, \tilde{r})\}$

**end**

# Capítulo 5

# Avaliação Experimental

*If I have seen further it is by standing on the sholders of Giants.*
— Isaac Newton, Quoting Bernard of Chartres

Neste capítulo apresentamos, primeiramente, a metodologia de validação; as bases de dados utilizadas; métricas de avaliação; e por fim; uma análise de três experimentos: análise da qualidade em relação as demais, análise da robustez de todos os métodos, uma estimativa do ruído nas bases e uma análise qualitativa das matrizes de transição de ruído de classes estimadas.

## 5.1 Metodologia e Organização dos Experimentos

A metodologia da avaliação experimental da proposta será composta por diversos experimentos que têm por objetivo avaliar o desempenho dos modelos para cenários distintos. Como discutido anteriormente, é comum na literatura de ruído supor que o conjunto de dados é livre de ruído e assim, partindo desta conjectura, poluir a base através de alguma técnica de geração de ruído artificial. Através dos experimentos, serão avaliados os algoritmos utilizando duas técnicas de geração de ruído: uniforme e *pairwise*. A primeira irá simular o ruído do tipo NCAR e a segunda do tipo NAR. Para essas duas gerações, serão aplicadas diversas taxas de ruído variando de 0% até 30% em intervalos de 5%.

Serão realizados quatro experimentos para cada tipo de geração de ruído. Cada experimento será responsável por avaliar a proposta em um conjunto de algoritmos do estado-da-arte. O primeiro experimento irá avaliar a função de custo *forward* e *backward* utilizando a matriz de transição estimada através da nossa proposta e será comparada com os mesmos procedimentos, porém utilizando a matriz original que foi utilizada para perturbar a base. Assim, será possível avaliar o quão distante a proposta estará do ideal teórico.

No segundo experimento, serão alisadas as duas funções de custo utilizando a matriz de transição estimada contra outros modelos resistentes ao ruído. Os modelos propostos serão comparados com as funções de custo empregando a proposta original, ou seja, utilizando a estimação da matriz pelo algoritmo 6. Também serão comparados com duas outras funções de custo que são resistentes ao ruído: *boostrapping soft* e *boostrapping hard* (REED *et al.*, 2014). Por fim, serão avaliados ainda em conjunto o modelo proposto por XIAO *et al.* (2015), que é uma arquitetura de rede neural resistente ao ruído.

Na literatura da filtragem colaborativa foram propostos algoritmos de limpeza de dados para resolver o problema do ruído natural. O terceiro experimento será conduzido para avaliar a robustez dos modelos propostos contra os algoritmos de limpeza de dados. Serão utilizados os algoritmos propostos por O'MAHONY *et al.* (2006) e TOLEDO *et al.* (2015) na avaliação. Será incluído o mesmo modelo, porém sem o pré-processamento, já que o intuito é valiar o ganho do pré-processamento.

Por fim, o último experimento tem por objetivo avaliar a robustez dos algoritmos clássicos da filtragem colaborativa. Foram escolhidos um algoritmo mais representativo de cada abordagem e um modelo baseado em redes neurais, ou seja, um algoritmo baseado em memória e dois algoritmos baseados em modelo. Para o primeiro caso, foi utilizado o algoritmo dos vizinhos mais próximos. E para o segundos caso, foram escolhidos o *Improved Regularized SVD* (IRSVD) e o MLP do *framework Neural Collaborative Filtering* (HE *et al.*, 2017).

Definidos os experimentos, é necessário também definir o modelo que será utilizado em conjunto com as funções de custo resistentes ao ruído e dos algoritmos de *data cleansing*. Será utilizada uma rede neural para o modelo e a arquitetura utilizada será a MLP do *framework Neural Collaborative Filtering*. Tal arquitetura foi escolhida já que as variáveis latentes dela são aprendidas em conjunto com o classificador, diferentemente dos *frameworks* COFILS e A-COFILS (BRAIDA *et al.*, 2015; BARBIERI *et al.*, 2017), simplificando assim o processo de aprendizado. Os algoritmos de *data cleansing* serão aplicados no mesmo modelo utilizado pela proposta, tornando justa a comparação.

O protocolo utilizado para validação dos modelos será a validação cruzada. Através dela será possível avaliar a variabilidade de cada algoritmo. Neste trabalho, foi utilizado o 10-*fold validation* que consiste em dividir a amostra original aleatoriamente em 10 sub-amostras. Em cada uma dessas divisões será calculado o erro dos algoritmos utilizando essa sub-amostra e as demais serão utilizadas como treinamento.

Uma dificuldade inerente à construção de modelos é a estimização de seus parâmetros. Cada modelo possui diversos parâmetros que podem variar para cada base. Além disso, as características da base podem mudar toda vez que for apli-

cada uma técnica de geração de ruído. Esse problema é conhecido como hiper-parametrização (BERGSTRA *et al.*, 2011). Devido à quantidade de modelos e parâmetros, a força bruta, ou também chamado de *grid search*, se torna inviável. Assim sendo, foi escolhido um algoritmo de busca heurística chamado algoritmo genético para o processo de busca de parâmetros (RECHENBER, 1970). Sua escolha se deu em função da sua facilidade em realizar buscas com parâmetros discretos diferentemente do *simulated annealing*;

Para cada etapa do 10-*fold* para cada cenário de ruído, será realizada uma busca pelo melhores parâmetros utilizando o algoritmo genético. Será feita uma busca para cada modelo base, ou seja, uma para MLP, outra para o IRSVD e por fim uma para o algoritmo de vizinhos mais próximos. Após determinar os parâmetros, estes serão empregados na construção dos demais modelos. Será utilizado o valor de 1% para a mutação e o cruzamento será dado pela técnica de classificação.

Como o objetivo da busca é encontrar os parâmetros para criar o modelo mais acurado, a função de otimização, ou função *fitness*, utilizada será o RMSE. Foi escolhida essa função ao invés do MAE, pois ela penaliza os grandes erros, sendo mais adequada para este cenário de busca.

Uma dificuldade do experimento é o seu tempo computacional. Considerando que para treinar um modelo está na ordem de minutos e existe um enorme número de testes, uma grande quantidade de indivíduos por geração tornaria inviável o experimento. Desta forma, serão utilizados seis indivíduos por geração, sendo que o ponto de parada foi definido através de dois critérios: a primeira por um tempo máximo limite (duas horas) e a segunda através da paciência, que é uma técnica que define um ponto de parada após uma quantidade fixa de gerações não ter obtido uma melhora na função de otimização. Neste trabalho foi utilizado o valor de quatro para esse parâmetro, e seguindo essa configuração, caso o experimento utilizasse uma única máquina, este demoraria aproximadamente 10 meses para ser finalizado. Contudo, o procedimento é totalmente paralelizável entre máquinas e portanto é possível reduzir seu o custo de tempo.

Com objetivo de tornar a busca mais eficaz foi utilizada a técnica de elitização. Nesta, para cada geração, foi mantido o indivíduo com a melhor função de *fitness*. Além disso, entre os *folds*, os melhores parâmetros foram passados para inicializar a busca do *fold* seguinte, garantindo que a busca já inicie em um bom ponto de partida.

O algoritmo genético possui facilidade em trabalhar com a codificação discreta, sendo que nem todos os parâmetros dos modelos são valores discretos. Todavia alguns parâmetros foram discretizados para facilitar o uso do algoritmo genético. No caso do IRSVD, os parâmetros que serão otimizados são: número de variáveis latentes, taxa de aprendizado e a regularização. Já no caso dos vizinhos mais próximos,

será otimizado o número máximo e mínimo de vizinhos, utilizando a abordagem baseado no item e será utilizado a similaridade por cosseno.

A rede neural possui diversos parâmetros que correspondem tanto à sua arquitetura quanto ao processo de aprendizado. Neste caso será feita uma busca para encontrar os melhores valores para seis parâmetros: número de atributos por usuário e item, número de neurônios na camada escondida, o valor do *dropout*, valor do *batch*, número de épocas e o valor da paciência do treinamento. No caso da paciência, serão usados 10% do conjunto de treinamento para ser utilizado como conjunto de validação.

A rede possuirá duas camadas escondidas. A última camada utilizará a função de *Softmax* e terá a quantidade de neurônios igual a quantidade de elementos do conjunto de preferências, ou seja, cada neurônio representará uma nota. Foi escolhido essa função pois ela é uma função *link* inversível. No caso da função de custo, será escolhida a função Entropia Cruzada pois ela é uma função de custo composta e apropriada. Essas duas escolhas são necessárias para satisfazer o Teorema 2.

A função de ativação utilizada nas camadas intermediárias será a *ReLu*. Foi escolhida essa função de ativação pois ela possui uma característica particular fazendo o Hessian da função de custo Entropia Cruzada não depender do ruído, e portanto, a curvatura local é mantida inalterada (PATRINI *et al.*, 2016a). Desta forma, existe uma garantia que no procedimento *backward* utilizando a matriz de transição de ruído estimada, mesmo sendo estimada por um péssimo estimador, não tem impacto sobre a curvatura. Contudo, essa propriedade não é válida para o procedimento *forward*. No caso de outras funções de custos, *e.g.* sigmoíde, não possuem essa garantia. Por fim, o otimizador será o *Adam* (KINGMA e BA, 2014). O resumo de cada parâmetro e os seus respectivos valores é possível der ser vista na tabela 5.1.

Tabela 5.1: Lista dos parâmetros e seus respectivos valores para cada modelo que será otimizado através do algoritmo genético.

| Modelo | Parâmetro | Valores |
|---|---|---|
| Rede Neural | Variáveis Latentes | $\{x \mid x = 10 + 10 \cdot i, 0 \leq i \leq 9\} \cup \{150\}$ |
| | Neurônios | $\{x \mid x = 100 + 50 \cdot i, 0 \leq i \leq 8\}$ |
| | Dropout | $\{x \mid x = 0.1 \cdot i, 0 \leq i \leq 10\}$ |
| | Batch | $\{64, 128, 256\}$ |
| | Épocas | $\{10, 20, 30, 50\}$ |
| | Paciência | $\{1, 2, 3, 4, 5\}$ |
| IRSVD | Variáveis Latentes | $\{1, 2\} \cup \{x \mid x = 10 \cdot i, 1 \leq i \leq 9\}$ |
| | Taxa de Aprendizagem | $\{x \mid x = 0.01 \cdot i, 0 \leq i \leq 10\}$ |
| | Regularização | $\{x \mid x = 0.01 \cdot i, 0 \leq i \leq 10\}$ |
| KNN | Número Mínimo de Vizinhos | $\{5, 10\}$ |
| | Número de Vizinhos | $\{15, 30, 50\}$ |

O algoritmo de busca das âncoras possui um parâmetro que é limiar $\tau$. Através de vários testes, foi determinado que esse valor será de 0.005 para o procedimento *forward* e 0.0001 para o *backward*. Desta forma, serão estimadas duas matrizes para cada um dos procedimentos. Outro ponto crucial é a quantidade mínima de elementos âncoras que será considerada na média para estimar a matriz. Com uma quantidade baixa de âncoras, a estimativa pode se tornar muito sensível. Assim, foi escolhido o mínimo de cinco elementos. Caso não encontre esse número mínimo, será considerado que a classe não possui ruído.

## 5.2 Bases de Dados

Em todas as bases de dados de filtragem colaborativa há avaliações dos usuário sobre os itens. Esses dados podem acompanhar informações adicionais, já que é comum incluírem informações sobre os usuários e/ou sobre o conteúdo dos itens. Algumas também possuem a informação do tempo quando a nota foi capturada pelo sistema, sendo que esse atributo pode não significar que o usuário consumiu naquele momento e sim que foi informado.

Com propósito de avaliar a proposta em diversos cenários, foram escolhidas três bases de dados com características distintas: MovieLens 100k, CiaoDVD e Yahoo! Music. No caso do Movielens 100k, essa base é oriunda de um sistema de recomendação de filmes. Já o CiaoDVD é de compra de DVDs de filmes e o Yahoo! Music é um serviço de *streaming* de música. Cada uma das bases possui diferentes domínios, estes bem particulares, deixando-as com propriedades dissemelhantes. A citar como exemplo o Yahoo! Music, onde os usuários tendem dar notas extremas, enquanto que o MovieLens 100k as avaliações seguem uma distribuição normal.

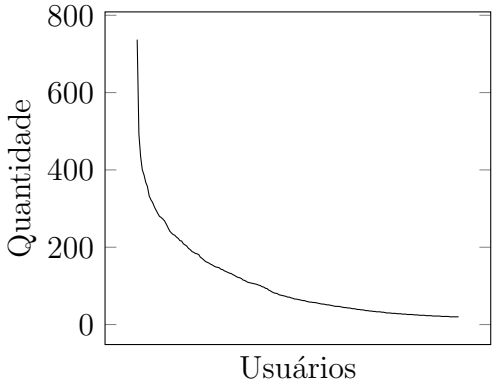
### 5.2.1 MovieLens 100k

O conjunto de dados MovieLens (MILLER *et al.*, 2003) é oriundo de um sistema de recomendação de filmes desenvolvido pelo *GroupLens*[1] e possui 100.000 avaliações. Nela existem 943 usuários e 1682 filmes e o seu conjunto de preferência é um número inteiro que varia entre um e cinco. Na figura 5.1 pode ser vista algumas estatísticas dessa base.

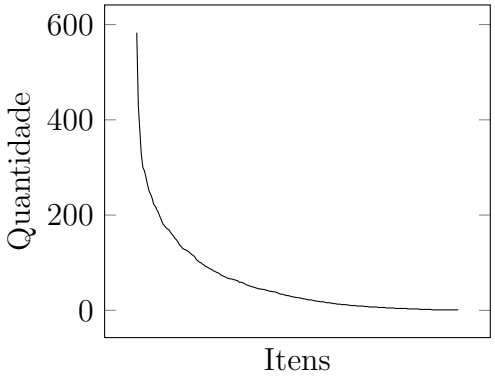
Neste sistema, os usuários tinham que avaliar no mínimo quinze filmes ao entrar, para depois avaliar a quantidade de filmes que quisessem. Assim, uma característica deste sistema é que a avaliação é feita posteriormente e não no ato do consumo do item.

[1] http://www.grouplens.org/

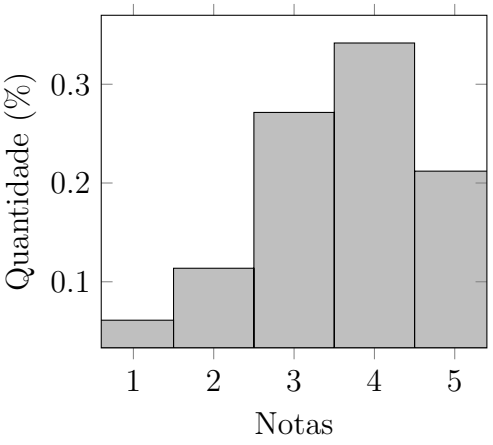
Além das avaliações, existem informações sobre usuários como a idade e o sexo, e sobre os filmes, como título e a data de lançamento. Essas informações não foram utilizadas em nenhum momento pois se fossem utilizadas descaracterizariam a proposta que está dentro do contexto da filtragem colaborativa.



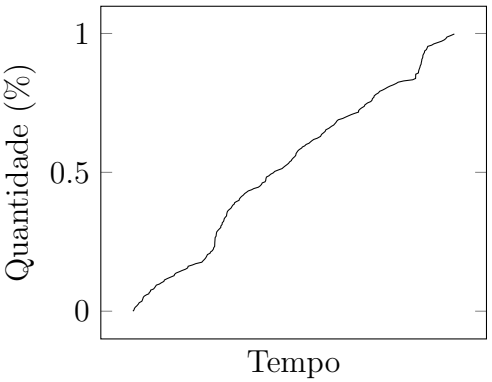

(a) Quantidade de Avaliações por usuário da base MovieLens 100k.




(b) Quantidade de Avaliações por item da base MovieLens 100k.




(c) Histograma das avaliações da base MovieLens 100k.




(d) Quantidade de Avaliações por item da base MovieLens 100k.

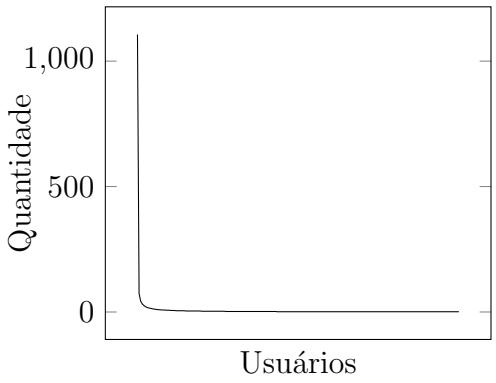
Figura 5.1: Análise estatística da base da dados MovieLens 100k.

### 5.2.2 CiaoDVD

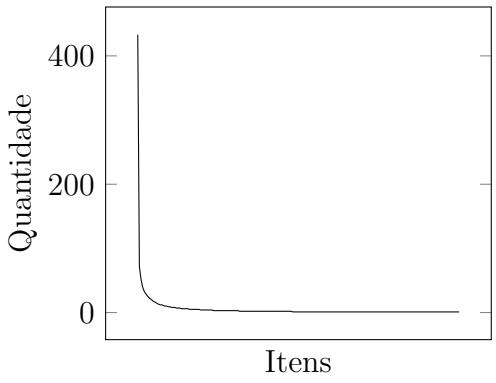
CiaoDVD é um conjunto de dados que foi gerado a partir da extração do conteúdo da categoria de DVD do site de compras online Ciao[2] em dezembro de 2013 (GUO *et al.*, 2014). Nesse *dataset*, os usuários podem criar uma análise sobre cada produto que eles compraram ou que foi utilizado no passado. Além disso, outros usuários podem avaliar essas análises em relação a utilidades delas. Caso um usuário considere essa análise útil, ele pode marcar o escritor dela como confiável e adicioná-lo na sua lista de confiança.

[2]http://dvd.ciao.co.uk/

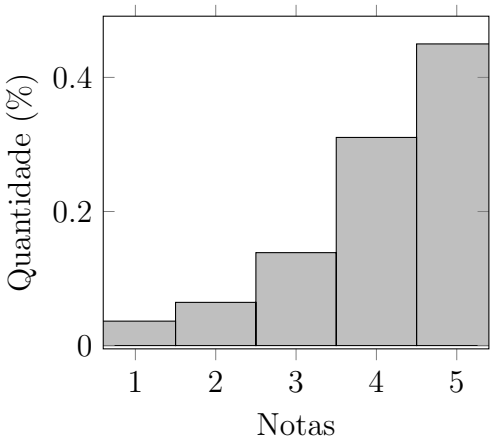
O CiaoDVD foi publicado com as avaliações sobre os produtos, sendo que o usuário pode avaliá-los com um número inteiro variando entre um e cinco. Ele possui 17615 usuários e 16121 itens e contendo 72665 avaliações. Todas as avaliações foram disponibilizadas em conjunto com a data em que foram realizadas. Nele está contido a rede de confiança e as avaliações dos usuários sobre as análise. Neste trabalho esses dados não foram utilizados.



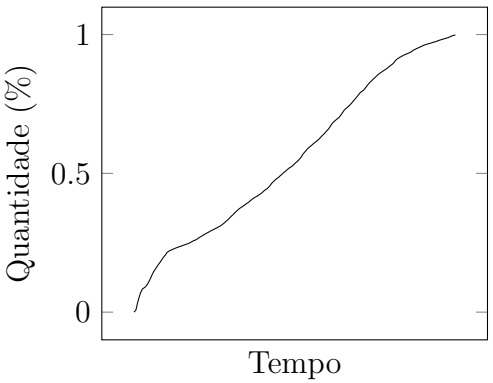

(a) Quantidade de Avaliações por usuário da base CiaoDVD.




(b) Quantidade de Avaliações por item da base CiaoDVD.




(c) Histograma das avaliações da base CiaoDVD.




(d) Quantidade de Avaliações por item da base CiaoDVD.

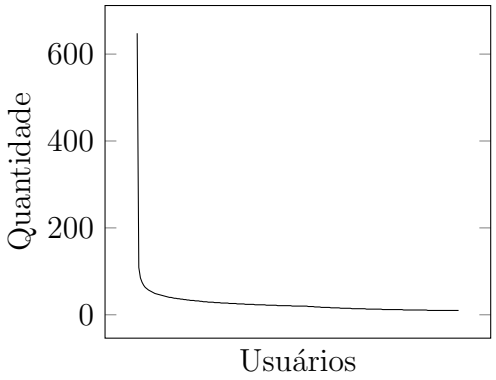
Figura 5.2: Análise estatística da base da dados CiaoDVD.

### 5.2.3 Yahoo! Music

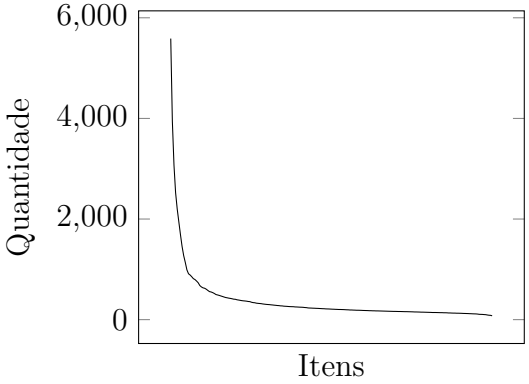
R3 Yahoo! Music[3] é um conjunto de dados onde as avaliações de músicas foram coletadas de duas fontes. A primeira é através da interação dos usuários com o serviço de *streaming* de música chamado Yahoo! Music Services e a segunda consiste na pesquisa realizada pela Yahoo! Research.

[3]http://research.yahoo.com/Academic_Relations

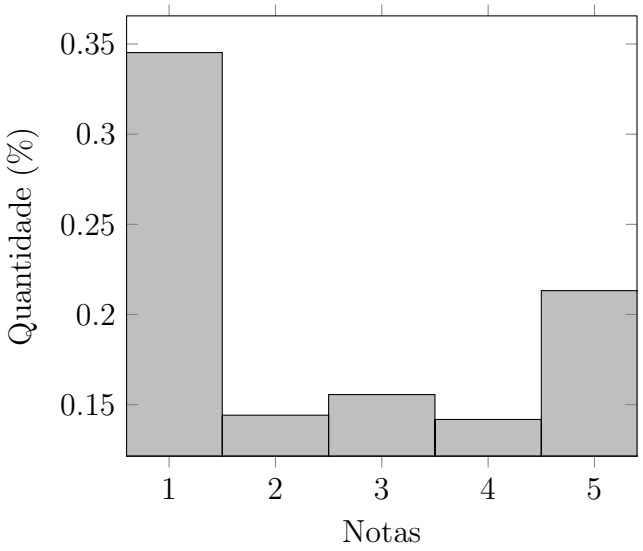
O conjunto de dados possui 365.704 avaliações, sendo que estas para 1000 músicas e realizadas por 15.400 usuários. A avaliação é um número inteiro entre 1 e 5. Na figura 5.3 pode ser visto algumas estatísticas dessa base.




(a) Quantidade de Avaliações por usuário da base Yahoo! Music.




(b) Quantidade de Avaliações por item da base Yahoo! Music.




(c) Histograma das avaliações da base Yahoo! Music.

Figura 5.3: Análise estatística da base de dados Yahoo! Music.

## 5.3 Métricas de Avaliação

Visando comparar os modelos, serão utilizadas quatro métricas, sendo duas destas normalmente utilizadas para comparar os algoritmos de recomendação. Serão utilizadas duas métricas para avaliar a acurácia dos modelos e outras duas para avaliar suporte a decisão dos modelos. A primeira e mais utilizada, *Mean Absolute Error* (MAE), é a métrica que calcula a média da diferença absoluta entre as

previsões e as notas reais (GOLDBERG *et al.*, 2001; HERLOCKER *et al.*, 2004).

$$MAE = \|R\|^{-1} \cdot \sum_{(u,i) \in R} |r_{ui} - \tilde{r}_{ui}| \tag{5.1}$$

A *Root-Mean-Square error* (RMSE) é uma outra métrica de erro e se tornou popular devido ao *Netflix prize* para avaliação de desempenho de algoritmos de recomendação. A diferença entre ela e o MAE é que ela penaliza os erros grandes em comparação aos pequenos.

$$RMSE = \sqrt{\|R\|^{-1} \cdot \sum_{(u,i) \in R} (r_{ui} - \tilde{r}_{ui})^2} \tag{5.2}$$

Outra classe de métricas em recomendação é a métrica para suporte de decisão. que tem por objetivo avaliar se o recomendador ajuda o usuário a fazer boas decisões, *e.g.* ajudar o usuário escolher bons itens ao invés de itens ruins. No caso de um sistema onde o conjunto de preferência são valores entre $[1, 5]$, pode-se interpretar que o item será recomendado caso a nota prevista pelo modelo seja superior a 4 ($\hat{r}_{ui} \geq \delta | \delta = 4$) e o contrário não. Existe uma dificuldade inerente a essas métricas na filtragem colaborativa, pois é difícil determinar o que é uma boa decisão, já que esta pode pode variar entre base de dados, usuário, contexto ou tempo.

Esse trabalho unificará entre as bases o limiar do que será considerado um bom item ou não. Esse limiar será único para todos usuários e para os itens, e vai ser dado por

$$\delta = max(P) - \lfloor \frac{1}{3} \cdot (max(P) - min(P)) \rceil \ , \tag{5.3}$$

onde $P$ é o conjunto de preferências.

Estabelecido uma medida para definir o que é uma boa recomendação será possível definir as métricas para o suporte à decisão. Muitas das métricas utilizadas dentro dessa classe são da área da Busca e Recuperação de Informação, como exemplo *precision* e *recall*. O *precision* ($P$) é a fração dos itens bons que foram considerados como bons (equação 5.4), enquanto o *recall* ($R$) é a fração dos itens bons recuperados (equação 5.5).

$$P = \frac{|\{\text{itens bons}\} \cap \{\text{itens que foram considerados bons}\}|}{\{\text{itens que foram considerados bons}\}} \tag{5.4}$$

$$R = \frac{|\{\text{itens bons}\} \cap \{\text{itens que foram considerados bons}\}|}{\{\text{itens bons}\}} \tag{5.5}$$

Geralmente, as duas métricas não são discutidas de forma isolada e um bom método precisa encontrar um balanço entre as duas métricas. Quando o algoritmo for conservativo, irá selecionar poucas instâncias e por isso será mais preciso, mas

em compensação, deixará na base instâncias que não deveriam ser selecionadas. Por outro lado, quando ele é mais agressivo, tende a selecionar mais instâncias erroneamente. Essa escolha dependerá do contexto do algoritmo. Com a intenção de unificar métricas, surgiram propostas que combinam o *precision* e *recall*. A mais utilizada é o *F1 score* (ou *F-measure*) que é a média harmônica dessas duas métricas (equação 5.6).

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{5.6}$$

Por fim, será utilizada a métrica *Macro-average F1 score* ou *Macro F1*. Considerando cada nota como uma classe, podemos definir o *precision* e o *recall* para cada classe. Assim, o *Macro F1* é o *F1 score* utilizando a média aritmética de cada *precision* (*Macro-average precision* para cada classe e o mesmo para o *recall* (*Macro-average recall*).

## 5.4 Experimentos

Nesta seção serão apresentados os resultados obtidos em cada experimento. O objetivo dos experimentos é o de analisar a robustez dos modelos propostos em relação aos outros métodos da literatura. Essas análises serão feitas a partir de métricas preditivas e de tomada de decisão. Para isto, serão utilizadas três bases dados com características distintas. Além disso, foram aplicadas duas formas de geração de ruído artificial para diversas taxas com a finalidade de avaliar a robustez dos modelos.

O primeiro experimento foi conduzido para verificar a robustez dos dois modelos propostos, *backward* ($\ell^{\leftarrow}$ learn T) e *forward* ($\ell^{\rightarrow}$ learn T), utilizando a estimação da matriz de transição de ruído proposta. Considerando que as bases não possuem ruído, foi aplicado ruído artificial entre 0% a 30% utilizando os métodos de geração uniforme e *pairwise*. Com o objetivo de verificar o limite da robustez tendo conhecimento desse ruído aplicado, foram avaliados os mesmos dois métodos ($\ell^{\leftarrow}$ e $\ell^{\rightarrow}$), porém utilizando a matriz de transição de ruído que foi utilizada para perturbar os dados. Desta forma, será possível comparar o desempenho utilizando a matriz estimada e a matriz real aplicada. O resultado pode ser visto na figura 5.4 tal que cada coluna representa os resultados para cada uma das bases.

O modelo *forward* utilizando a matriz de ruído estimada obteve melhores resultados com relação as métricas do que os demais nas bases MovieLens 100k e Yahoo! Music. Além disso, obteve o resultado muito próximo ao *forward* utilizando a matriz original aplicada na base CiaoDVD. Já o comportamento do *backward* variou com relação as métricas e bases. Esse procedimento utilizando a matriz aplicada

obteve resultados melhores de modo geral do que utilizando a matriz estimada. Em alguns casos, como no MovieLens e CiaoDVD, o procedimento utilizando a matriz estimada obteve um resultado inicial próximo à matriz aplicada, porém tendo uma queda de desempenho depois dos 10% de ruído. Com relação aos dois procedimentos, o *forward* obteve um resultado superior. Eles obtiveram resultados próximos somente na base Yahoo! Music, porém em valores baixos de ruído.

A utilização da matriz que foi aplicada para perturbar as bases, à principio, deveria ser o teto do desempenho dos classificadores. Contudo, a utilização da matriz estimada, no procedimento *forward*, obteve um resultado superior. Isso pode ocorrer, pois para a realização dos experimentos, foi considerado que as bases de dados utilizadas não possuíam ruído. Contudo, ela não é verdadeira, mas essa suposição precisa ser feita porque não existe uma base de dados com os ruídos indicados para a filtragem colaborativa.

No caso do MovieLens 100k e do Yahoo! Music, a estimação da matriz de transição conseguiu capturar, além do ruído aplicado, o ruído natural das bases. Quando o ruído aplicado era acima de 25%, o desempenho do modelo teórico ficou superior ou igual. Essa grande quantidade de ruído incorporado deve ter sido o suficiente para mascarar os padrões dentro da base e tornando difícil a construção de um classificador acurado, prejudicando também a estimação da matriz de transição de ruído.

Já no caso do CiaoDVD, os resultados utilizando a matriz aplicada e a estimada foram próximos. Uma possível razão para essa proximidade é que nesta base o ruído natural pode ser menor que as demais. Desta forma, a estimação estaria capturando somente o ruído aplicado. Essa discussão será aprofundada na seção 5.4.1, onde serão feitas uma análise qualitativa e quantitativa de quão ruidosa cada base é.

O segundo experimento foi conduzido para avaliar a robustez de outros modelos que incorporam no seu aprendizado o ruído. Foram avaliadas quatro propostas que realizam modificações na função de custo e uma que propõe uma arquitetura da rede neural diferenciada para o ruído. Ademais, esses modelos serão comparados com os dois modelos propostos. No caso das funções de custo, serão avaliados os procedimentos *backward* e *forward* que utiliza a estimação da matriz proposta com o algoritmo 6 ($\ell^{\leftarrow}$ argmax T e $\ell^{\rightarrow}$ argmax T respectivamente). Além disso, serão utilizados os métodos *bootstrapping hard* (BHard) e *bootstrapping soft* (BSoft) e a proposta por SUKHBAATAR e FERGUS (2014) (Channel), que é uma modificação da arquitetura da rede neural adicionando uma camada com objetivo de filtrar o ruído. O resultado pode ser visto na figura 5.5 e seguirá o mesmo padrão de visualização do primeiro.

O procedimento *forward* que utiliza a proposta para estimar a matriz de ruído obteve os melhores resultados nas bases MovieLens 100k e Yahoo! Music tanto para

o ruído uniforme quanto para o *pairwise*. No caso do *backward*, obteve o segundo melhor resultado na base Yahoo! Music, ficando atrás somente do outro método proposto. No caso do CiaoDVD, o método *forward* com a matriz estimada obteve um resultado um pouco superior aos demais, porém ficou próximo dos métodos *bootstrapping soft*, *bootstrapping hard* e do *Channel*.

Os procedimentos que utilizaram a estimação dada pelo algoritmo 6 obtiveram os piores resultados em todas as bases. No MovieLens 100k e no Yahoo! Music, esses modelos foram piores mas não houve uma diferença significava, acompanhando os erros dos demais. No caso do CiaoDVD, eles obtiveram uma diferença significativa nos dois tipos de ruídos. Esses resultados demonstram que as características necessárias para o Teorema 3 não são válidas para a filtragem colaborativa, necessitando um outro método para estimar a matriz de transição de ruído.

Os métodos *bootstrapping soft*, *bootstrapping hard* e *Channel* obtiveram um resultados próximos, seguindo a mesma tendência de perda de desempenho. Dentre eles, o *bootstrapping soft* obteve o melhor resultado, sendo que na base CiaoDVD, obteve um resultado próximo aos modelos propostos. Principalmente na geração de ruído uniforme e para pequenas taxas. A vantagem desse método em relação aos demais é sua complexidade de implementação, pois só adiciona um peso entre os rótulos e a previsão do modelo, sendo de fácil aplicação para qualquer cenário. Já o modelo *Channel* obteve um resultado próximo aos outros em valores pequenos de ruído, contudo obteve um detrimento acentuado em valores grandes se comparado aos demais.

O terceiro experimento foi conduzido para avaliar a robustez dos algoritmos de *data cleansing* para a filtragem colaborativa, onde serão avaliados dois métodos: algoritmo 1 (Mahony) e algoritmo 5 (Toledo). Serão utilizados os modelos propostos e também o modelo sem nenhum pré-processamento (MLP) para fins de comparação. O resultado é ilustrado na figura 5.6 e segue o mesmo padrão dos anteriores.

O método *forward* com a matriz estimada obteve resultados melhores do que os algoritmos de *data cleansing*. O algoritmo Mahony obteve melhores resultados que o Toledo, sendo que o Toledo foi proposto com um avanço para o Mahony. Em TOLEDO *et al.* (2015), utilizou outros modelos de previsão, KNN e o *Regularized SVD*. Além disso, o trabalho fixou todos os parâmetros, tanto dos modelos quanto dos algoritmos de *data cleasing*, podendo assim ter influenciado o seu desempenho.

Nas bases MoviLens 100k e Yahoo! Music, o algoritmo Mahony obteve um resultado próximo ao *forward* com a matriz estimada. Já o Toledo alcançou um resultado próximo no MovieLens 100k, porém obteve um resultado péssimo no Yahoo! Music. O seu desempenho foi muito semelhante ao modelo sem pré-processamento, evidenciando que nesta base o algoritmo não conseguiu selecionar adequadamente o conjunto de possíveis ruídos. Esse algoritmo divide as notas em faixas para o

usuário e item. Desta forma, classifica cada usuário e item nessas faixas. Os autores utilizaram a base MovieLens 100k para proporem essa divisão. Contudo, o Yahoo! Music não segue o comportamento típico, possuindo uma grande parte das notas nas notas extremas dificultando o algoritmo.

No caso do CiaoDVD, essa base possui uma distribuição diferente do MovieLens, porém mais próxima se comparada a do Yahoo! Music. Nesta base, os algoritmos obtiveram resultados semelhantes ao *forward*. Sendo que o *forward* obteve um MAE um pouco superior principalmente nas faixas entre 10% a 25% de ruído. Contudo, para $F1_{rec}$ obteve um resultado próximo e para F1 os algoritmos do Mahony e Toledo obtiveram um resultado melhor para o ruído do tipo *pairwise*.

O último experimento foi realizado para avaliar a robustez dos algoritmos de previsão da filtragem colaborativa. Até então, não houve um estudo para avaliar a robustez dos métodos, já que os trabalhos possuíam o enfoque na redução da acurácia (TOLEDO *et al.*, 2015; YERA *et al.*, 2016; O'MAHONY *et al.*, 2006). Assim, foram escolhidos alguns métodos da literatura para serem avaliados e serão utilizados os modelos propostos como base de comparação. Serão avaliados dois métodos, o primeiro o algoritmo de vizinhos mais próximos (KNN), o *Improved Regularized SVD* (IRSVD) e o MLP do *framework Neural Collaborative Filtering*. O resultado é ilustrado na figura 5.7.

Nesse experimento é possível avaliar o impacto do ruído nos modelos, mostrando o detrimento das métricas. Nos três modelos clássicos, a queda do desempenho seguiu na mesma ordem de grandeza, seguindo aproximadamente uma relação linear com a taxa de ruído. Na base CiaoDVD, a diferença entre o desempenho dos modelos, nas primeiras taxas de ruído, obtiveram um valor inferior nas outras bases. Essa diferença corrobora com que foi analisado no primeiro experimento, em que o CiaoDVD possui um ruído incorporado nos dados inferior as demais bases. Essa discussão será melhor abordada na próxima seção.

Em todos os experimentos foram utilizados dois tipos de geração de ruído e com complexidades diferentes, neste caso o ruído *pairwise* possui um impacto maior do que o uniforme. Essa característica pode ser vista em todos os experimentos. De modo geral, todos os modelos obtiveram um desempenho inferior quando era aplicado o ruído *pairwise*. Em alguns casos, a curva se tornava mais acentuada mostrando a dificuldade de aprender neste cenário.

### 5.4.1 Análise do Ruído nas Bases

Com o objetivo em avaliar a robustez, foi necessário fazer uma suposição ao qual as bases de dados utilizadas são livres de ruído. Desta forma, através dos geradores de ruído artificial foi possível avaliar a robustez dos métodos propostos e comparar

com os de mais da literatura. Contudo, essa suposição é fraca, pois é natural que as bases possuam inconsistências naturais.

O primeiro experimento foi realizado para comparar os procedimentos de correção *forward* e *backward* que utiliza a matriz de transição de ruído que foi aplicada nos dados e utilizando o algoritmo proposto para estimá-la. Verificou-se que nas taxas iniciais de ruído, a utilização da estimação obteve um resultado superior à matriz utilizada para perturbar as bases. Essa diferença vai sendo reduzida com o decorrer do aumento da taxa. Para valores superiores, os desempenhos se tornam mais próximos e, em alguns casos, a matriz estimada obtém um resultado inferior.

Não é possível verificar o quão ruidosa é cada base. Isto porque nenhuma delas possuem alguma marcação que informe se alguma avaliação é ruidosa. Outro ponto que poderia ajudar nessa avaliação, seria se elas possuíssem mais de uma avaliação para o par usuário-item, semelhante ao que foi proposto em AMATRIAIN *et al.* (2009a). Esse problema não é característico somente da filtragem colaborativa, sendo comum em outros domínios (FRENAY e VERLEYSEN, 2013). Contudo, é possível analisá-las para verificar indícios do quão ruidosas essas bases são.

O MovieLens 100k é um sistema de teste do grupo GroupLens para avaliar os algoritmos de recomendação. As avaliações feitas pelos os usuários para os filmes não era realizada exatamente no momento do consumo do item e não restringe a ordem das avaliações. Essa ordem interfere na tomada de decisão da definição de uma nota para o filme, pois é comum utilizar um filme base como escala de gosto. No caso do Yahoo! Music, um sistema de *streaming* de música, esse tipo de item é muito suscetível ao contexto e ao tempo, ou seja, a ordem em que as músicas estão sendo apresentadas ao usuário.

Os dados do CiaoDVD são oriundos de uma única categoria. Neste caso, de DVDs de uma loja virtual. Assim, quando um usuário deseja comprar um DVD, ele já possui uma certa certeza e opinião sobre esse item. Ele provavelmente já consumiu o item em outra fonte, *e.g.* no cinema, e quer guardá-la de recordação. Além disso, o usuário precisa pagar para consumir, sendo difícil comprar algo que não deseje. Desta forma, possuirá uma maior certeza do consumo e do gosto sobre o item se comparado às outras bases.

Essas características podem ser vistas no histograma das avaliações (figuras 5.1, 5.3 e 5.2), sendo que o CiaoDVD segue uma exponencial e nos outros dois, MovieLens 100k e Yahoo! Music, uma normal e uma parábola, respectivamente.

Outra análise que pode ser feita, além do entendimento do domínio de recomendação, é através dos algoritmos de *data cleansing*. Esses algoritmos definem uma heurística com objetivo de encontrar as possíveis avaliações inconsistentes e assim corrigi-las. Se um algoritmo marcar percentualmente mais elementos em uma base do que a outra, existe uma tendência que essa base seja mais ruidosa. A grande

dificuldade é que cada heurística é uma aproximação do modelo de geração de ruído, podendo assim mascarar os resultados e podendo levar à conclusões falsas.

O algoritmo proposto por O'MAHONY *et al.* (2006), tem como característica construir um modelo e verificar o quão distantes os dados que foram utilizados para treiná-lo estão da sua previsão. Dependendo da distância, essa avaliação é marcada como ruído e será substituída pelo valor estimado. Já o método proposto por TOLEDO *et al.* (2015), define uma heurística que seleciona uma quantidade de elementos do conjunto de treinamento para ser avaliado pelo algoritmo anterior. Para tal, ele irá categorizar os usuários e os itens em faixas de comportamento de notas e irá verificar se as avaliações seguem essas características. Caso contrário, ele será marcado como um possível ruído.

Essa divisão de faixa do algoritmo proposto por TOLEDO *et al.* (2015) usou a distribuição de notas da base MovieLens 100k, ou seja, uma normal. Esse comportamento também foi visto no terceiro experimento, fazendo crer que o emprego desse método não é adequado em outras bases com outras características. Por essa razão, foi utilizado somente o algoritmo do O'MAHONY *et al.* (2006) para verificar a quantidade de elementos marcados como ruído. Foi adotado o algoritmo dos vizinhos mais próximos com a similaridade por cosseno e o valor de 60 para o número de vizinhos. Esses valores foram recomendados por TOLEDO *et al.* (2015) e O'MAHONY *et al.* (2006). O percentual de elementos selecionados é ilustrado na figura 5.8.

A quantidade de elementos selecionados seguiu o comportamento até então descrito, em que o CiaoDVD possui a menor inconsistência e o Yahoo! Music a maior. Esse comportamento pode ser visto no primeiro experimento em que a diferença do desempenho do modelo que utiliza a matriz estimada com relação a matriz teórica cresce nesta mesma ordem. Isso indica que o modelo que utiliza a matriz estimada consegue aprender o ruído inerente a base e o aplicado.

### 5.4.2 Análise da Robustez

Até então, foram realizadas análises das principais métricas em que se comparou o desempenho dos modelos propostos com os demais. Eles obtiveram um resultado igual ou superior ao modelo teórico e superior aos demais nas três bases e nos dois cenários de ruído. Sendo que a complexidade do ruído diminuía a diferença dos desempenhos. Contudo, não foi avaliado a robustez dos modelos propostos e dos demais modelos.

HUBER (2004) definiu robustez como a capacidade de um algoritmo construir um modelo que seja insensível à corrupção de dados. Como consequência, ele será capaz de resistir melhor ao impacto do ruído e terá o comportamento mais próximo

do modelo construído com os dados limpos. Essa métrica é considerada mais importante do que a acurácia quando estamos no cenário de ruído.

Quanto menor a diferença do desempenho em relação à base não corrompida, mais robusto é o modelo. Para tal, será utilizada como forma de avaliação da área da curva, e como base utiliza-se o valor do desempenho da métrica em 0% de ruído para cada modelo. Sendo que no MAE será a área abaixo da curva e nas demais métricas será a área acima. Na tabela 5.2 é possível avaliar, para cada base e para cada forma de geração de ruído, a área da curva para a métrica MAE e as tabelas 5.3 e 5.4 para as demais métricas. Foram utilizadas duas faixas de curva: entre 0% até 20% e 0% até 30%.

Tabela 5.2: Análise da área embaixo da curva para a métrica MAE dos resultados dos quatro experimentos. O cálculo da área de baixo da curva foi utilizado o valor de 0% de ruído como base de cálculo. Quanto mais escura for a cor verde, mais próximo do maior valor.

| Base de Dados | Geração | Faixa da Curva | $l^{\leftarrow}$ | $l^{\leftarrow}$argmax T | $l^{\leftarrow}$learn T | BHard | BSoft | Channel | $l^{\rightarrow}$ | $l^{\rightarrow}$argmax T | $l^{\rightarrow}$learn T | IRSVD | KNN | Mahony | MLP | Toledo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MovieLens 100k | Uniforme | 0-30 | 8,46E-03 | 1,06E-02 | 8,50E-03 | 1,43E-02 | 9,24E-03 | 1,20E-02 | 5,32E-03 | 9,24E-03 | 6,03E-03 | 1,71E-02 | 1,26E-02 | 7,93E-03 | 1,44E-02 | 7,99E-03 |
| | | 0-20 | 3,97E-03 | 4,69E-03 | 2,93E-03 | 5,72E-03 | 3,82E-03 | 4,89E-03 | 2,50E-03 | 4,00E-03 | 2,67E-03 | 7,34E-03 | 5,14E-03 | 3,39E-03 | 6,04E-03 | 3,34E-03 |
| | *Pairwise* | 0-30 | 1,22E-02 | 1,65E-02 | 1,55E-02 | 1,96E-02 | 1,38E-02 | 1,46E-02 | 7,79E-03 | 1,43E-02 | 9,54E-03 | 2,00E-02 | 1,66E-02 | 1,14E-02 | 1,95E-02 | 1,09E-02 |
| | | 0-20 | 5,40E-03 | 7,06E-03 | 5,39E-03 | 7,95E-03 | 5,30E-03 | 5,36E-03 | 3,33E-03 | 5,82E-03 | 3,42E-03 | 8,62E-03 | 6,82E-03 | 4,43E-03 | 8,17E-03 | 4,59E-03 |
| Yahoo! Music | Uniforme | 0-30 | 1,03E-02 | 2,53E-02 | 1,44E-02 | 2,54E-02 | 2,11E-02 | 2,58E-02 | 4,18E-03 | 2,29E-02 | 7,09E-03 | 3,06E-02 | 2,35E-02 | 1,86E-02 | 2,75E-02 | 2,51E-02 |
| | | 0-20 | 4,61E-03 | 1,12E-02 | 5,35E-03 | 1,11E-02 | 8,59E-03 | 1,13E-02 | 1,15E-03 | 1,03E-02 | 3,15E-03 | 1,42E-02 | 9,48E-03 | 7,12E-03 | 1,22E-02 | 1,03E-02 |
| | *Pairwise* | 0-30 | 1,54E-02 | 4,07E-02 | 1,79E-02 | 3,57E-02 | 3,22E-02 | 3,69E-02 | 6,13E-03 | 3,17E-02 | 9,22E-03 | 4,06E-02 | 3,69E-02 | 3,12E-02 | 3,98E-02 | 3,80E-02 |
| | | 0-20 | 5,64E-03 | 1,61E-02 | 7,37E-03 | 1,48E-02 | 1,21E-02 | 1,60E-02 | 1,76E-03 | 1,45E-02 | 3,56E-03 | 1,86E-02 | 1,46E-02 | 1,13E-02 | 1,65E-02 | 1,69E-02 |
| CiaoDVD | Uniforme | 0-30 | 1,27E-02 | 1,21E-01 | 2,21E-02 | 2,37E-02 | 1,92E-02 | 3,13E-02 | 7,96E-03 | 5,12E-02 | 1,07E-02 | 2,70E-02 | 2,70E-02 | 2,07E-02 | 2,67E-02 | 1,92E-02 |
| | | 0-20 | 5,42E-03 | 7,18E-02 | 9,78E-03 | 9,47E-03 | 7,57E-03 | 1,36E-02 | 3,22E-03 | 2,19E-02 | 4,67E-03 | 1,12E-02 | 1,13E-02 | 8,49E-03 | 1,11E-02 | 7,54E-03 |
| | *Pairwise* | 0-30 | 2,22E-02 | 4,83E-02 | 2,52E-02 | 3,11E-02 | 2,45E-02 | 3,73E-02 | 1,24E-02 | 8,33E-02 | 1,79E-02 | 3,59E-02 | 3,28E-02 | 2,42E-02 | 3,49E-02 | 2,78E-02 |
| | | 0-20 | 8,17E-03 | 9,44E-03 | 9,46E-03 | 1,25E-02 | 9,23E-03 | 1,60E-02 | 4,85E-03 | 2,99E-02 | 5,85E-03 | 1,38E-02 | 1,35E-02 | 9,05E-03 | 1,44E-02 | 1,15E-02 |

Tabela 5.3: Análise da área acima da curva para a métrica $F1_{rec}$ dos resultados dos quatro experimentos. O cálculo da área de baixo da curva foi utilizado o valor de 0% de ruído como base de cálculo.Quanto mais escura for a cor verde, mais próximo do maior valor.

| Base de Dados | Geração | Faixa da Curva | $l^{\leftarrow}$ | $l^{\leftarrow}$argmax T | $l^{\leftarrow}$learn T | BHard | BSoft | Channel | $l^{\rightarrow}$ | $l^{\rightarrow}$argmax T | $l^{\rightarrow}$learn T | IRSVD | KNN | Mahony | MLP | Toledo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MovieLens 100k | Uniforme | 0-30 | 8,09E-03 | 1,68E-02 | 1,05E-02 | 2,40E-02 | 1,58E-02 | 2,20E-02 | 1,91E-03 | 1,31E-02 | 2,77E-03 | 2,11E-02 | 2,04E-02 | 1,22E-02 | 2,30E-02 | 1,81E-02 |
| | | 0-20 | 4,09E-03 | 8,14E-03 | 2,58E-03 | 9,72E-03 | 6,81E-03 | 1,09E-02 | 7,83E-04 | 5,95E-03 | 1,41E-03 | 9,42E-03 | 8,90E-03 | 4,72E-03 | 1,05E-02 | 7,49E-03 |
| | *Pairwise* | 0-30 | 8,59E-03 | 1,87E-02 | 1,40E-02 | 2,34E-02 | 1,46E-02 | 2,11E-02 | 4,46E-03 | 1,69E-02 | 2,54E-03 | 2,00E-02 | 1,98E-02 | 1,22E-02 | 2,28E-02 | 1,69E-02 |
| | | 0-20 | 4,24E-03 | 8,14E-03 | 3,16E-03 | 9,59E-03 | 6,31E-03 | 1,01E-02 | 2,01E-03 | 6,78E-03 | 6,10E-04 | 8,82E-03 | 8,98E-03 | 5,02E-03 | 1,03E-02 | 7,35E-03 |
| Yahoo! Music | Uniforme | 0-30 | 7,12E-03 | 2,76E-02 | 1,08E-02 | 3,05E-02 | 2,72E-02 | 3,31E-02 | 1,51E-03 | 2,28E-02 | -9,88E-04 | 2,44E-02 | 2,36E-02 | 1,79E-02 | 3,33E-02 | 3,14E-02 |
| | | 0-20 | 3,88E-03 | 1,29E-02 | 3,62E-03 | 1,30E-02 | 1,13E-02 | 1,43E-02 | 1,38E-04 | 9,93E-03 | -9,80E-05 | 1,20E-02 | 9,87E-03 | 6,85E-03 | 1,55E-02 | 1,42E-02 |
| | *Pairwise* | 0-30 | 7,26E-03 | 3,58E-02 | 1,30E-02 | 3,80E-02 | 3,09E-02 | 3,29E-02 | 2,91E-03 | 3,12E-02 | -4,86E-04 | 2,15E-02 | 2,67E-02 | 2,32E-02 | 3,83E-02 | 3,18E-02 |
| | | 0-20 | 2,45E-03 | 1,54E-02 | 7,24E-03 | 1,61E-02 | 1,25E-02 | 1,38E-02 | 6,12E-04 | 1,45E-02 | 2,20E-04 | 1,03E-02 | 1,13E-02 | 8,78E-03 | 1,74E-02 | 1,55E-02 |
| CiaoDVD | Uniforme | 0-30 | 5,00E-03 | 5,20E-02 | 2,53E-02 | 3,10E-02 | 2,59E-02 | 1,07E-02 | 3,87E-03 | 5,30E-02 | 1,06E-02 | 3,15E-02 | 2,79E-02 | 1,78E-02 | 3,52E-02 | 1,93E-02 |
| | | 0-20 | 1,95E-03 | 2,81E-02 | 1,13E-02 | 1,30E-02 | 1,05E-02 | 7,29E-04 | 1,25E-03 | 2,47E-02 | 5,38E-03 | 1,45E-02 | 1,27E-02 | 7,17E-03 | 1,61E-02 | 7,85E-03 |
| | *Pairwise* | 0-30 | 8,88E-03 | 3,25E-02 | 2,84E-02 | 3,76E-02 | 3,11E-02 | 2,77E-02 | 6,84E-03 | 7,17E-02 | 1,66E-02 | 3,35E-02 | 3,09E-02 | 2,03E-02 | 4,16E-02 | 2,55E-02 |
| | | 0-20 | 3,45E-03 | 1,33E-02 | 1,15E-02 | 1,60E-02 | 1,24E-02 | 1,21E-02 | 2,61E-03 | 3,12E-02 | 5,87E-03 | 1,67E-02 | 1,49E-02 | 6,75E-03 | 1,88E-02 | 1,03E-02 |

Nas tabelas 5.2, 5.3 e 5.4 é possível ver que todas as soluções para amenizar o problema do ruído fazem aumentar a robustez dos modelos, tanto incorporando no aprendizado quanto realizando um pré-processamento. Ademais, o procedimento *forward* que utiliza a matriz aplicada nos dados obteve a melhor robustez se comparada aos demais modelos para todas as métricas, sendo que este mesmo procedimento, obteve um resultado próximo ao teórico com relação à robustez e, como já visto em alguns casos, obteve um desempenho superior.

Tabela 5.4: Análise da área acima da curva para a métrica F1 dos resultados dos quatro experimentos. O cálculo da área de baixo da curva foi utilizado o valor de 0% de ruído como base de cálculo. Quanto mais escura for a cor verde, mais próximo do maior valor.

| Base de Dados | Geração | Faixa da Curva | $l^{\leftarrow}$ | $l^{\leftarrow}$argmax T | $l^{\leftarrow}$learn T | BHard | BSoft | Channel | $l^{\rightarrow}$ | $l^{\rightarrow}$argmax T | $l^{\rightarrow}$learn T | IRSVD | KNN | Mahony | MLP | Toledo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MovieLens 100k | Uniforme | 0-30 | 6,77E-03 | 8,86E-03 | 3,26E-03 | 1,01E-02 | 7,86E-03 | 3,79E-02 | 3,26E-03 | 6,95E-03 | 4,91E-03 | 1,16E-02 | 9,22E-03 | 7,71E-03 | 1,18E-02 | 9,48E-03 |
| | | 0-20 | 3,31E-03 | 4,07E-03 | 1,96E-05 | 4,47E-03 | 3,45E-03 | 2,08E-02 | 1,38E-03 | 3,06E-03 | 2,09E-03 | 5,42E-03 | 4,13E-03 | 3,60E-03 | 4,81E-03 | 4,18E-03 |
| | *Pairwise* | 0-30 | 8,93E-03 | 1,29E-02 | 1,20E-02 | 1,46E-02 | 1,21E-02 | 3,97E-02 | 5,46E-03 | 1,18E-02 | 7,93E-03 | 1,44E-02 | 1,46E-02 | 9,49E-03 | 1,56E-02 | 1,49E-02 |
| | | 0-20 | 4,03E-03 | 5,64E-03 | 4,17E-03 | 6,64E-03 | 5,35E-03 | 2,15E-02 | 2,23E-03 | 5,30E-03 | 2,20E-03 | 6,57E-03 | 5,95E-03 | 4,02E-03 | 6,48E-03 | 6,47E-03 |
| Yahoo! Music | Uniforme | 0-30 | 4,88E-03 | 1,20E-02 | 5,62E-03 | 1,26E-02 | 9,64E-03 | 2,69E-02 | 1,59E-03 | 1,06E-02 | 2,65E-03 | 1,21E-02 | 8,81E-03 | 6,10E-03 | 1,38E-02 | 1,22E-02 |
| | | 0-20 | 2,24E-03 | 5,36E-03 | 2,09E-03 | 5,19E-03 | 3,57E-03 | 6,61E-03 | 4,69E-04 | 4,70E-03 | 1,12E-03 | 5,76E-03 | 3,53E-03 | 2,09E-03 | 6,11E-03 | 5,14E-03 |
| | *Pairwise* | 0-30 | 6,82E-03 | 1,96E-02 | 7,53E-03 | 1,77E-02 | 1,42E-02 | 3,57E-02 | 2,24E-03 | 1,44E-02 | 3,41E-03 | 1,60E-02 | 1,30E-02 | 1,06E-02 | 1,92E-02 | 1,70E-02 |
| | | 0-20 | 2,68E-03 | 7,26E-03 | 3,23E-03 | 7,10E-03 | 5,09E-03 | 1,20E-02 | 6,18E-04 | 6,59E-03 | 1,28E-03 | 7,22E-03 | 5,23E-03 | 3,67E-03 | 8,01E-03 | 7,81E-03 |
| CiaoDVD | Uniforme | 0-30 | 9,86E-03 | 1,39E-02 | 3,01E-03 | 7,34E-03 | 5,93E-03 | -2,29E-03 | 5,73E-03 | 8,65E-03 | 6,60E-03 | 8,47E-03 | 1,01E-02 | 7,03E-03 | 7,32E-03 | 1,44E-03 |
| | | 0-20 | 4,71E-03 | 7,72E-03 | 4,85E-04 | 3,42E-03 | 2,80E-03 | -3,80E-03 | 2,45E-03 | 4,16E-03 | 3,19E-03 | 3,59E-03 | 4,47E-03 | 3,46E-03 | 3,31E-03 | 1,09E-05 |
| | *Pairwise* | 0-30 | 1,01E-02 | 5,93E-03 | 8,73E-03 | 8,10E-03 | 5,37E-03 | 8,29E-03 | 7,35E-03 | 1,30E-02 | 7,85E-03 | 2,32E-02 | 1,29E-02 | 6,68E-03 | 8,75E-03 | 8,03E-03 |
| | | 0-20 | 5,24E-03 | 2,53E-03 | 4,05E-03 | 3,82E-03 | 1,93E-03 | 3,41E-03 | 3,68E-03 | 5,07E-03 | 3,62E-03 | 8,80E-03 | 5,92E-03 | 2,73E-03 | 3,75E-03 | 3,41E-03 |

Em taxas pequenas de ruído, a robustez do método que utiliza a matriz aplicada e a matriz estimada são próximas. Contudo, para valores maiores houve um detrimento da robustez do modelo com a matriz estimada e, em decorrência disso, a diferença entre os dois métodos aumentou desproporcionalmente. Esse fenômeno acontece, pois existe uma necessidade de construir um modelo para os âncoras. Para as taxas maiores de ruído, a quantidade de avaliações alteradas se torna maior, tornando assim mais difícil de construir um classificador acurado. Pela mesma razão, a diferença é superior no caso da geração do ruído *pairwise* em relação ao ruído uniforme. O primeiro ruído um impacto na quebra dos padrões do que o segundo.

PATRINI *et al.* (2016a) mostraram através de experimentos em classificação de imagens que o procedimento *backward* possuía um desempenho inferior ao *forward*. Esse comportamento se manteve nos experimentos realizados para a filtragem colaborativa. Mesmo tendo garantias de otimalidade do procedimento, os autores dessa característica. Eles acreditam que essa diferença é causada por problemas numéricos. Estes que podem ser devido à inversão de matrizes ou na dramática mudança dos valores da função de custo.

Uma solução que obteve um resultado considerável em função da sua simplicidade, é o *bootstrapping soft*. Ela pode se tornar uma alternativa com um bom custo benefício, pois não é necessário realizar nenhum cálculo prévio para a construção do modelo, apenas uma ponderação na função de custo. Outro método que precisa ser evidenciado é o Mahony que obteve uma robustez considerável. Contudo, os dois possuem um desempenho e robustez inferiores ao modelo proposto.

### 5.4.3 Análise da Matriz de Transição de Ruído Estimada

Verificou-se na seção 5.4 que a utilização da matriz estimada através do algoritmo das âncoras com os métodos de correção de função de custo *backward* e *forward* obteve um resultado superior tanto na acurácia quanto na robustez do que utilizando a matriz estimada pelo algoritmo proposto por PATRINI *et al.* (2016a). Contudo,

não foi realizada uma análise do comportamento desses dois métodos de estimação separadamente das funções de custo durante os experimentos. Desta forma, foi estimada a matriz de transição de ruído utilizando esses dois métodos para as bases MovieLens 100k, CiaoDVD e Yahoo! Music. Foi utilizado o algoritmo genético para encontrar os melhores parâmetros dos modelos utilizados, porém, diferentemente do experimento anterior, foi utilizada a base completa para estimação.

Na figura 5.9 é possível observar as matrizes de transição de ruído estimadas que utilizam o método proposto ($T_{\mathcal{A}}$) e proposto pelo trabalho de PATRINI *et al.* (2016a) ($T_{max}$) nas bases sem incorporação de ruído artificial. Pode-se verificar que a matriz $T_{max}$ fica próxima da identidade nas bases MovieLens 100k e Yahoo! Music, e dessa forma, a função de custo com essa matriz é a mesma do que a função original. Já no CiaoDVD, o método estimou que existe um ruído com relação as notas mais baixas e corrigindo principalmente para o valor três, ou seja, um valor intermediário do conjunto de preferência.

Quanto a utilização do método proposto, a matriz estimada possui uma incerteza na vizinhança de cada classe, ou seja, se uma pessoa explicitou a nota dois para um item, existe uma probabilidade que essa avaliação seja um ou três. Desta forma, como foi discutido no capítulo 4, a avaliação possui uma incerteza intrínseca. O usuário estima o seu sentimento em um valor, contudo esse sentimento é um conceito vago e abstrato que faz com que a nota varie em função de diversos fatores externos.

Nas três bases, a probabilidade das avaliações extremas estarem corretas é muito superior em relação as avaliações intermediárias. Desta forma, quando o usuário possui uma repulsa ou um grande apreço sobre um item, a escolha da nota é uma tarefa menos indecisa do que a escolha de notas intermediárias, isso porque ele irá escolher entre o menor e maior valor do conjunto de preferência respectivamente.

Uma deficiência do método de estimação proposto é quando não é possível encontrar âncoras para a realização do cálculo, e que o valor mínimo utilizado nesse trabalho foi de cinco âncoras. Na base CiaoDVD, as avaliações um e dois não obtiveram esse valor mínimo e por isso foi considerado que aquela classe não possuía ruído. Uma possível causa se dá pela proporção de instâncias dessas classes na base de dados. Essas duas classes somadas possuem menos do que 10% do total de avaliações, resultado em um desbalanceamento de classes, o que dificulta a criação de um modelo para previsão.

As figuras 5.10 e 5.11 ilustram as matrizes estimadas para as três bases, porém com ruído artificial aplicado do tipo *pairwise* de 5% e 10%, respectivamente. Com a aplicação do ruído, a matriz estimada $T_{max}$ continuou sendo próxima da matriz identidade para as bases MovieLens 100k e Yahoo! Music. No caso da matriz $T_{\mathcal{A}}$, a incerteza de cada classe aumentou com a elevação da taxa de ruído, conseguindo assim capturar o ruído que foi inserido.

No caso do CiaoDVD, o método proposto teve dificuldade em estimar as avaliações um e dois nesses dois novos cenários. Como a taxa de ruído é proporcional, essas classes continuaram com uma proporção baixa na base, dificultando a criação do estimador. Essa pode ser uma razão ao qual os métodos *forward* e *backward* não obtiveram um grande ganho nesta base contra os demais métodos do estado-da-arte se comparado às bases MovieLens 100k e Yahoo! Music.

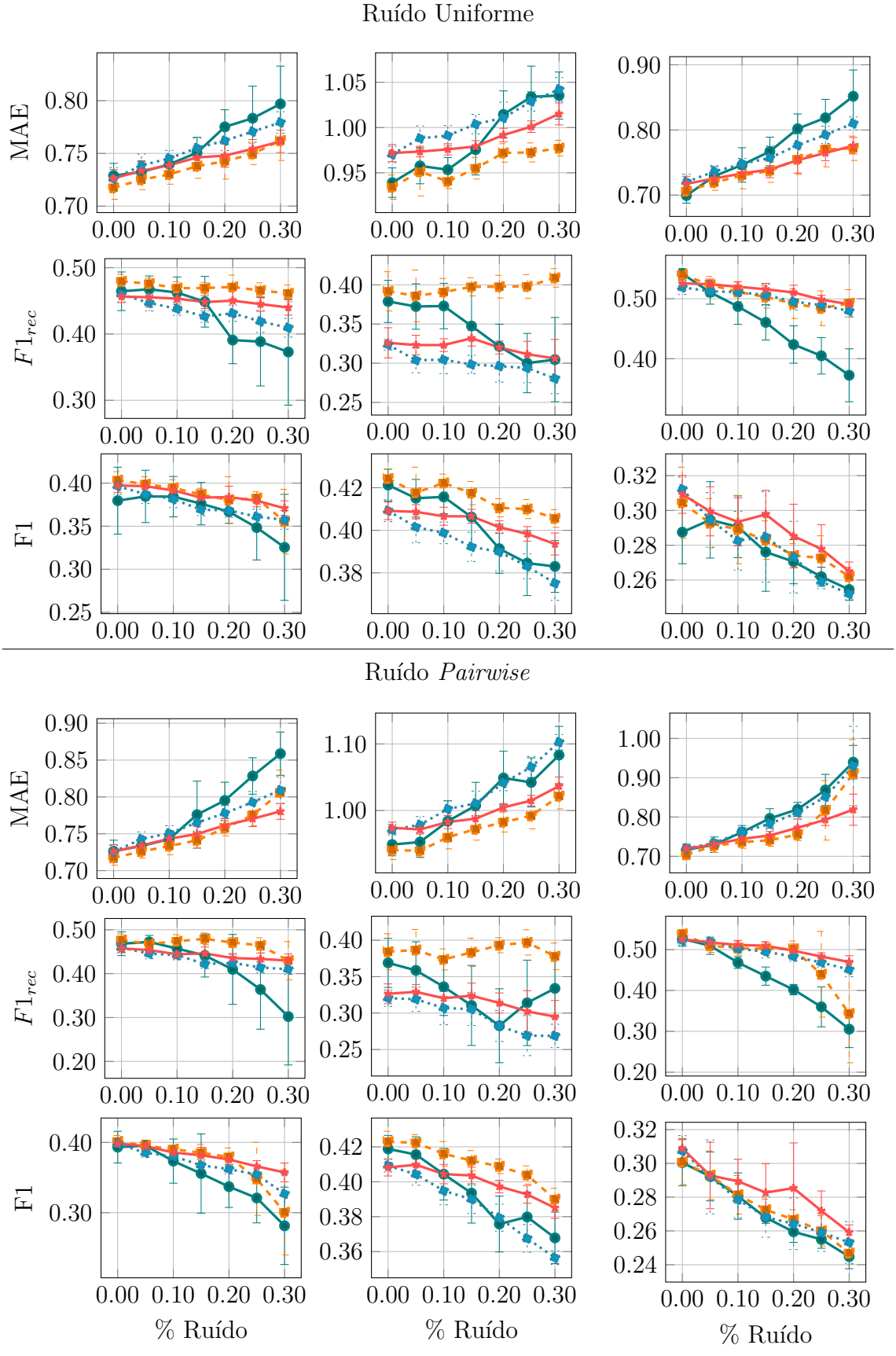



Figura 5.4: Avaliação do primeiro experimento.

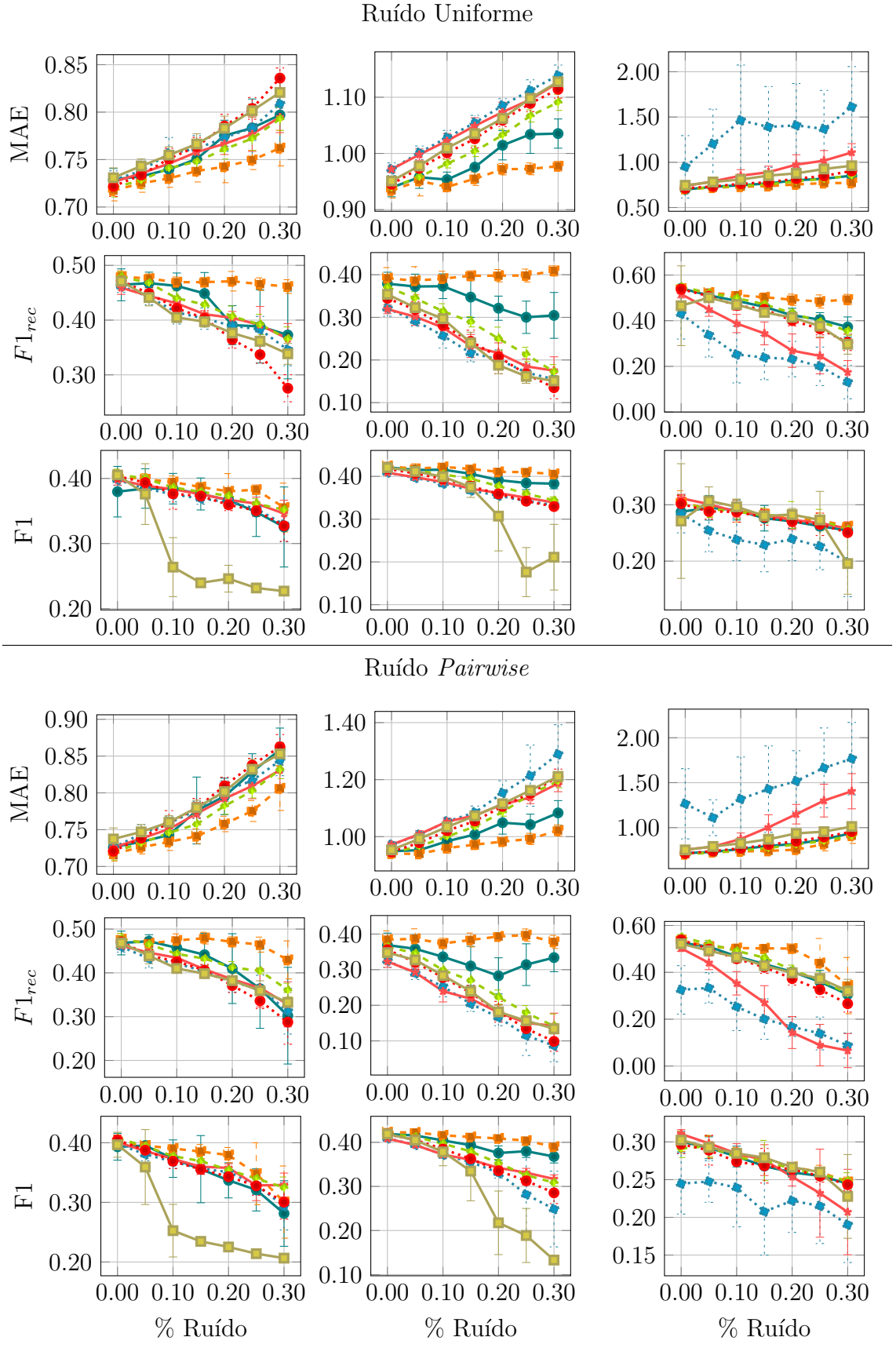



Figura 5.5: Avaliação do segundo experimento.

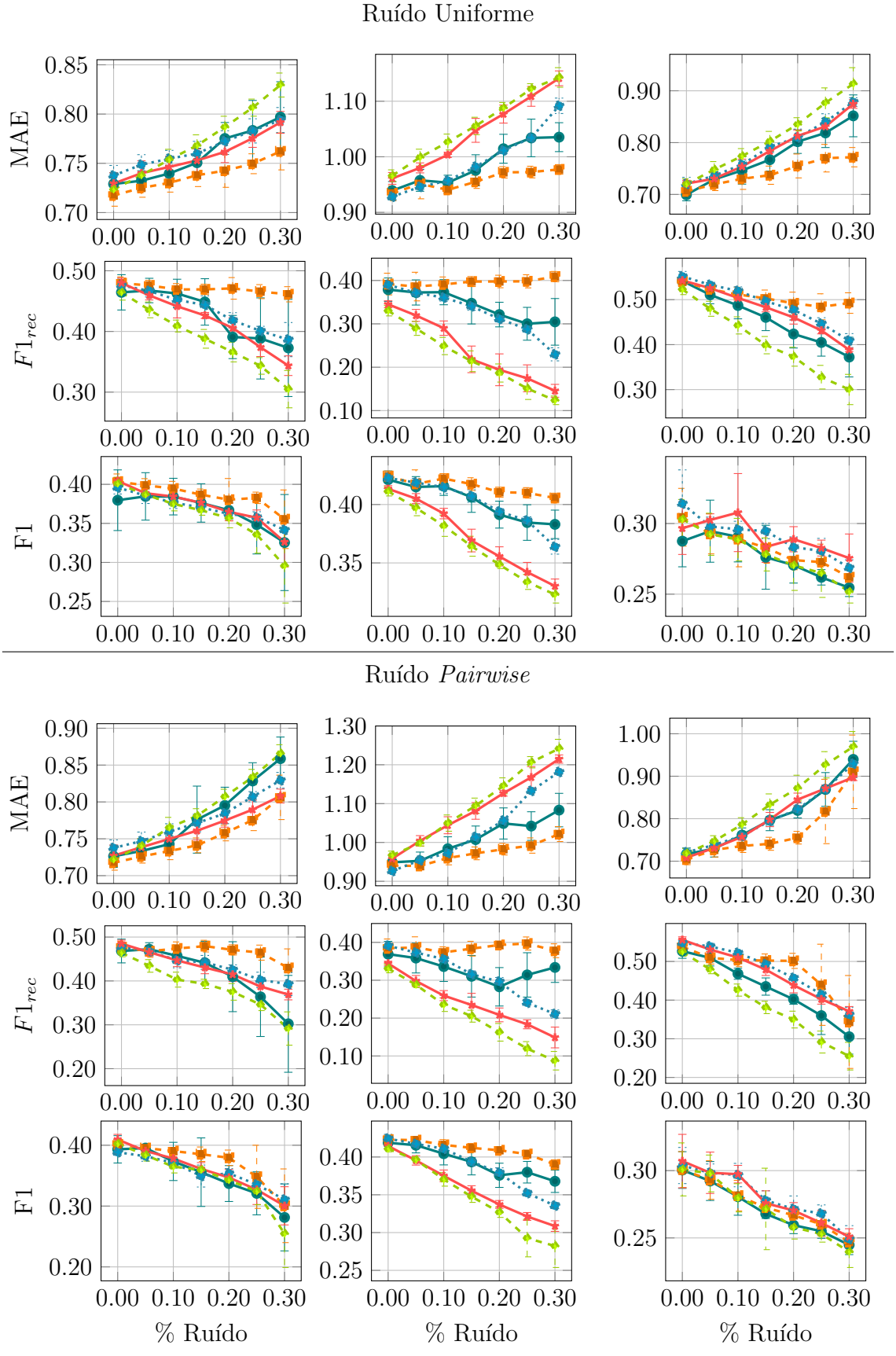



Figura 5.6: Avaliação do terceiro experimento.

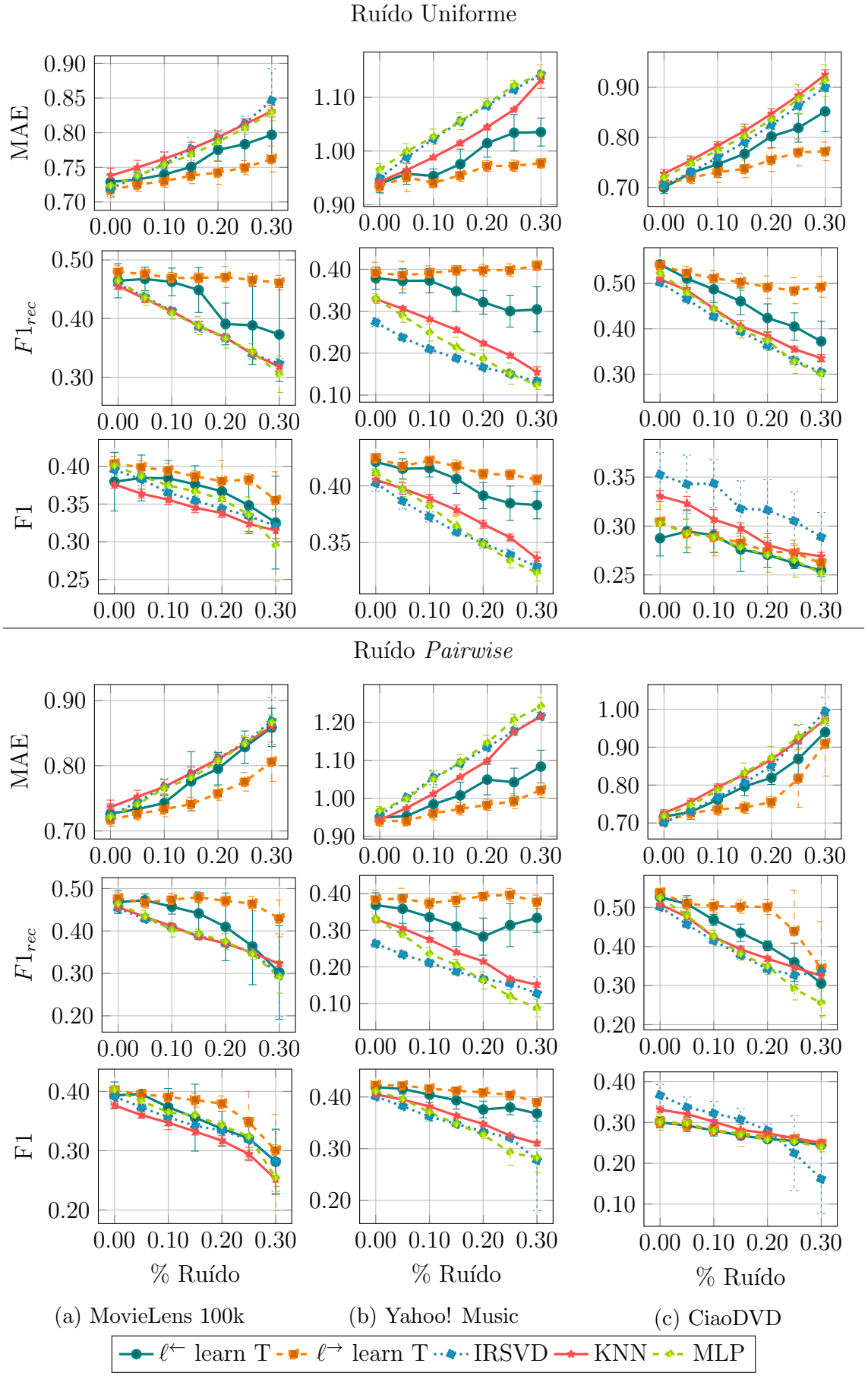



Figura 5.7: Avaliação do quarto experimento.

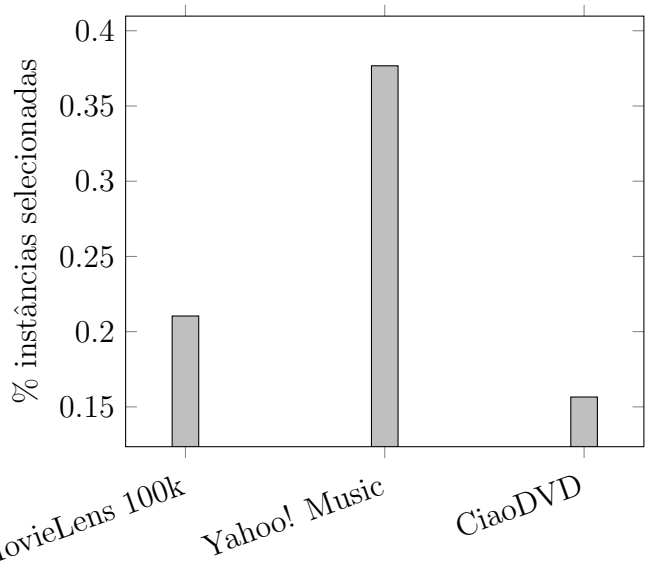



Figura 5.8: Quantidade de avaliações selecionadas como ruído pelo algoritmo de *data cleansing* proposto por O'MAHONY *et al.* (2006) em cada conjunto de dados.

MovieLens 100k

$$T_{\mathcal{A}} \cong \begin{bmatrix} 0.89 & 0.07 & 0.03 & 0.01 & 0.0 \\ 0.24 & 0.5 & 0.2 & 0.06 & 0.0 \\ 0.04 & 0.19 & 0.55 & 0.2 & 0.02 \\ 0.01 & 0.04 & 0.19 & 0.51 & 0.25 \\ 0.0 & 0.01 & 0.05 & 0.19 & 0.74 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.98 & 0.02 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.01 & 0.99 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

CiaoDVD

$$T_{\mathcal{A}} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.13 & 0.22 & 0.34 & 0.22 & 0.1 \\ 0.01 & 0.03 & 0.11 & 0.65 & 0.19 \\ 0.0 & 0.0 & 0.0 & 0.05 & 0.94 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 0.17 & 0.24 & 0.3 & 0.19 & 0.1 \\ 0.15 & 0.24 & 0.34 & 0.18 & 0.08 \\ 0.01 & 0.07 & 0.57 & 0.31 & 0.03 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Yahoo! Music

$$T_{\mathcal{A}} \cong \begin{bmatrix} 0.86 & 0.04 & 0.04 & 0.02 & 0.03 \\ 0.16 & 0.48 & 0.21 & 0.09 & 0.06 \\ 0.08 & 0.2 & 0.43 & 0.22 & 0.07 \\ 0.06 & 0.08 & 0.19 & 0.45 & 0.21 \\ 0.05 & 0.02 & 0.03 & 0.07 & 0.83 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.99 & 0.01 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.99 & 0.01 & 0.0 \\ 0.0 & 0.0 & 0.01 & 0.99 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Figura 5.9: Estimação do ruído nas bases originais MovieLens 100k, CiaoDVD e Yahoo! Music utilizando o método proposto ($T_{\mathcal{A}}$) e o método de PATRINI *et al.* (2016a) ($T_{max}$).

Ruído Aplicado

$$T = \begin{bmatrix} 0.95 & 0.0 & 0.0 & 0.025 & 0.025 \\ 0.0 & 0.95 & 0.0 & 0.025 & 0.025 \\ 0.0 & 0.0 & 0.95 & 0.025 & 0.025 \\ 0.01\overline{6} & 0.01\overline{6} & 0.01\overline{6} & 0.95 & 0.0 \\ 0.01\overline{6} & 0.01\overline{6} & 0.01\overline{6} & 0.0 & 0.95 \end{bmatrix}$$

MovieLens 100k

$$T_{\mathcal{A}} \cong \begin{bmatrix} 0.84 & 0.07 & 0.04 & 0.03 & 0.02 \\ 0.17 & 0.38 & 0.26 & 0.14 & 0.06 \\ 0.06 & 0.2 & 0.48 & 0.23 & 0.04 \\ 0.03 & 0.05 & 0.18 & 0.5 & 0.23 \\ 0.03 & 0.03 & 0.08 & 0.25 & 0.62 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.03 & 0.88 & 0.09 & 0.0 & 0.0 \\ 0.0 & 0.03 & 0.9 & 0.07 & 0.0 \\ 0.0 & 0.0 & 0.05 & 0.92 & 0.03 \\ 0.0 & 0.01 & 0.01 & 0.03 & 0.94 \end{bmatrix}$$

CiaoDVD

$$T_{\mathcal{A}} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.12 & 0.19 & 0.33 & 0.27 & 0.09 \\ 0.03 & 0.06 & 0.17 & 0.51 & 0.22 \\ 0.01 & 0.01 & 0.01 & 0.07 & 0.91 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 0.17 & 0.19 & 0.23 & 0.22 & 0.19 \\ 0.16 & 0.24 & 0.37 & 0.19 & 0.04 \\ 0.03 & 0.11 & 0.47 & 0.36 & 0.03 \\ 0.0 & 0.01 & 0.08 & 0.83 & 0.08 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Yahoo! Music

$$T_{\mathcal{A}} \cong \begin{bmatrix} 0.78 & 0.05 & 0.05 & 0.05 & 0.08 \\ 0.17 & 0.41 & 0.22 & 0.12 & 0.08 \\ 0.1 & 0.19 & 0.4 & 0.21 & 0.09 \\ 0.08 & 0.1 & 0.22 & 0.41 & 0.2 \\ 0.1 & 0.04 & 0.06 & 0.09 & 0.71 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.02 & 0.92 & 0.05 & 0.0 & 0.0 \\ 0.02 & 0.03 & 0.91 & 0.04 & 0.01 \\ 0.0 & 0.0 & 0.05 & 0.93 & 0.01 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Figura 5.10: Estimação do ruído nas bases MovieLens 100k, CiaoDVD e Yahoo! Music utilizando o método proposto ($T_{\mathcal{A}}$) e o método de PATRINI *et al.* (2016a) ($T_{max}$) sendo que foi aplicado 5% de ruído do tipo *pairwise*.

Ruído Aplicado

$$T = \begin{bmatrix} 0.9 & 0.0 & 0.0 & 0.05 & 0.05 \\ 0.0 & 0.9 & 0.0 & 0.05 & 0.05 \\ 0.0 & 0.0 & 0.9 & 0.05 & 0.05 \\ 0.0\overline{3} & 0.0\overline{3} & 0.0\overline{3} & 0.9 & 0.0 \\ 0.0\overline{3} & 0.0\overline{3} & 0.0\overline{3} & 0.0 & 0.9 \end{bmatrix}$$

MovieLens 100k

$$T_{\mathcal{A}} \cong \begin{bmatrix} 0.66 & 0.12 & 0.09 & 0.08 & 0.05 \\ 0.15 & 0.35 & 0.29 & 0.15 & 0.07 \\ 0.06 & 0.21 & 0.44 & 0.22 & 0.07 \\ 0.04 & 0.06 & 0.2 & 0.45 & 0.25 \\ 0.04 & 0.06 & 0.12 & 0.28 & 0.5 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.09 & 0.76 & 0.15 & 0.0 & 0.0 \\ 0.0 & 0.02 & 0.95 & 0.03 & 0.0 \\ 0.0 & 0.0 & 0.05 & 0.93 & 0.01 \\ 0.0 & 0.0 & 0.0 & 0.01 & 0.99 \end{bmatrix}$$

CiaoDVD

$$T_{\mathcal{A}} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.12 & 0.17 & 0.33 & 0.3 & 0.09 \\ 0.04 & 0.07 & 0.19 & 0.46 & 0.24 \\ 0.01 & 0.02 & 0.03 & 0.08 & 0.86 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 0.18 & 0.2 & 0.28 & 0.22 & 0.12 \\ 0.17 & 0.21 & 0.34 & 0.21 & 0.07 \\ 0.03 & 0.09 & 0.51 & 0.36 & 0.01 \\ 0.0 & 0.01 & 0.13 & 0.8 & 0.05 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Yahoo! Music

$$T_{\mathcal{A}} \cong \begin{bmatrix} 0.67 & 0.07 & 0.06 & 0.09 & 0.11 \\ 0.18 & 0.37 & 0.21 & 0.14 & 0.11 \\ 0.1 & 0.22 & 0.34 & 0.23 & 0.11 \\ 0.1 & 0.1 & 0.2 & 0.37 & 0.23 \\ 0.11 & 0.07 & 0.08 & 0.11 & 0.63 \end{bmatrix} \quad T_{max} \cong \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.01 & 0.95 & 0.03 & 0.01 & 0.0 \\ 0.0 & 0.11 & 0.76 & 0.12 & 0.01 \\ 0.01 & 0.0 & 0.07 & 0.86 & 0.07 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Figura 5.11: Estimação do ruído nas bases MovieLens 100k, CiaoDVD e Yahoo! Music utilizando o método proposto ($T_{\mathcal{A}}$) e o método de PATRINI *et al.* (2016a) ($T_{max}$) sendo que foi aplicado 10% de ruído do tipo *pairwise*.

# Capítulo 6

# Conclusões

Neste capítulo, o último do presente trabalho, apresentamos um resumo da proposta e dos resultados; elencamos nossas contribuições e, ao final, apresentamos os principais trabalhos futuros que foram vislumbrados durante o desenvolvimento da tese.

## 6.1 Sumário da Proposta

O ruído está presente em diversos domínios, principalmente naqueles aos quais existam alguma interferência humana para a coleta ou geração das informações. Como visto, os procedimentos de correção *backward* e *forward* resolvem o problema de ruído a partir de modificações na função de custo, tornando o modelo mais robusto. Entretanto, estes apresentam certas limitações que, quando aplicadas no domínio de Sistemas de Recomendação não são possíveis de utilizá-las e precisam do conhecimento da matriz de transição de ruído.

Até então, abordagens que propuseram resolver ruído natural na filtragem colaborativa, eram baseadas em data *cleansing*, e estimam o gerador do ruído a fim de identificar e corrigir os elementos ruidosos antes de utilizá-los no aprendizado de um modelo. Contudo, esse conhecimento não é incorporado na construção do modelo, acarretando em modelos menos robustos e acurados.

Desta forma, no contexto da filtragem colaborativa, propusemos um modelo em que, durante o seu aprendizado, o ruído seja considerado e modelado em conjunto do classificador. Para a transformação de domínio, o problema foi relaxado e foi considerado que o ruído neste problema seja do tipo NAR e propusemos a utilização dos procedimentos de correção *backward* e *forward* para esta tarefa. Propusemos, também, uma heurística para estimar uma matriz de probabilidade entre classes, responsável por aproximar o padrão latente de geração de ruído na base. Como principal objetivo, buscamos melhorar a qualidade preditiva de modelos sujeitos às condições de ruído. Adicionalmente, como objetivo secundário, buscamos aumentar

a robustez destes modelos com relação à adição de ruído em bases.

## 6.2 Sumário dos Resultados

Para a resolução do problema, assumimos duas condições especiais: (i) a existência de um padrão subjacente no processo de geração de ruído que pode ser aproximado por uma matriz de probabilidades entre classes; (ii) o conjunto de dados apresenta quantidade imprevisível de amostras de ruído, que possui alguma relação com a premissa (i). De acordo com este cenário, selecionamos três bases bem estabelecidas no domínio de sistemas de recomendação e que, de acordo com nossa análise prévia, representam as demais em função de suas distribuições. Realizamos um comparativo com os métodos do estado-da-arte baseados em data *cleansing*, aprendizado que considera ruído em funções de custo e um modelo neural.

Contudo, as bases selecionadas não possuem indicações sobre os elementos ruidosos ou a quantidade de ruído presente na base, dificultando a avaliação da robustez. Desta forma, foram propostos dois métodos de geração de ruído artificial: uniforme e *pairwise*. Sendo que o primeiro simulando ruído do tipo NCAR e o segundo do tipo NAR. Os experimentos foram conduzidos variando a incidência do ruído em cada base para cada tipo de geração, verificando assim o comportamento de cada método com relação a essa variação.

Os experimentos realizados mostraram que em um dos modelos propostos, a função de custo *forward* utilizando a heurística para estimar a matriz de transição de ruído, obteve resultados superiores aos demais nas três bases e nos dois tipos de geração de ruído. Ademais, ele obteve um resultado superior à utilização do mesmo com a matriz que foi originalmente aplicada na base para perturbá-la. Uma das possíveis razões para essa superioridade é o fato da base possuir inconsistências naturais, e o algoritmo para estimar a matriz conseguiu detectar tanto o ruído aplicado quanto o ruído latente.

Além da avaliação do desempenho dos modelos em diversos cenários, foi feita uma análise da robustez de cada método. O procedimento *forward* com a matriz aplicada aos dados obteve o melhor resultado de robustez. A proposta obteve o segundo melhor resultado, sendo superior aos demais métodos. Em valores baixos de ruído, a sua robustez era semelhante ao modelo teórico. Contudo, em taxas mais superiores houve um detrimento da sua robustez. Isso aconteceu pelo fato da necessidade em construir um modelo para definir os âncoras. A quantidade de ruído era tanta que criar um classificador acurado come esses dados torna-se uma tarefa árdua.

Ao final, foi realizada uma avaliação dos métodos de estimação da matriz de transição de ruído e verificou-se que as notas extremas possuem uma menor incon-

sistência em relação as demais. Essas notas são mais fáceis de serem decididas pelo usuário, pois estão associadas aos sentimentos de total repulsa ou um grande apreço sobre o item. Sendo que a escolha de um valor numérico estaria representado pelos valores extremos do conjunto de preferência.

Através da análise da matriz, foi observado também que o método proposto para estimar a matriz conseguiu capturar o ruído artificial inserido nas bases. Houve um aumento proporcional da incerteza dos rótulos com o aumento do ruído inserido, ou seja, um decréscimo dos valores da diagonal principal da matriz. O valor não foi o mesmo do ruído inserido, mas se manteve proporcional entre as taxas.

Devido ao problema de desbalanceamento das classes, o método proposto não conseguiu encontrar o número mínimo de âncoras para realizar a estimação das classes um e dois para a base CiaoDVD. Essa pode ser uma das razões pela qual, nesta base, o modelo proposto não obteve a mesma ordem de grandeza de ganho de acurácia se comparada aos demais métodos do estado-da-arte que foi observado nas outras bases.

## 6.3 Sumário das Contribuições

As principais contribuições desta tese podem ser sumarizadas da seguinte forma:

(i) **Proposta de um modelo robusto para filtragem colaborativa:** Foi proposto um modelo robusto para a filtragem colaborativa que considere o ruído durante o processo de aprendizado. Para tal, foram utilizados os procedimentos *backward* e *forward* para tornar a função de custo robusta.

(ii) **Estimativa de uma matriz de transição de ruído:** Para utilizar os métodos de correção da função de custo, faz-se necessário o conhecimento prévio de uma matriz de transição de ruído, que é desconhecida no problema. Desta maneira, foi proposto um método que, através da seleção de *âncoras*, foi possível estimar esta matriz de transição.

(iii) **Geração de ruído artificial:** Foram propostas duas formas de geração de ruído artificial para a filtragem colaborativa, sendo que uma para o ruído NCAR e outra para NAR. A partir desssas duas propostas, foi possível avaliar a robustez dos métodos e assim considerar o ruído natural como um problema de ruído de classe.

(iv) **Avaliação da robustez de modelos:** Os experimentos permitiram avaliar a robustez do método proposto em três bases de dados com caraterísticas distintas e utilizando duas formas de geração de ruído. Os experimentos avaliaram a

robustez dos métodos com relação a métricas de acurácia e tomada de decisão, e compará-los com outros métodos da literatura.

## 6.4 Trabalhos Futuros

Durante a elaboração da presente tese, alguns estudos foram detectados e relacionados para um futuro estudo. Eles estão divididos em três grupos de trabalhos futuros: ($a$) considerar o problema como NNAR, ($b$) ruído malicioso, ($c$) base de dados e outros tipos de ruídos, ($d$) outras arquiteturas neurais e ($e$) âncoras.

(a) **Considerar o Problema como NNAR**

Os procedimentos de correção *backward* e *forward* são adequados a problemas do tipo NAR, ou seja, o ruído de classe só depende da classe. No caso da filtragem colaborativa, o ruído depende não somente da classe mas também do usuário e do item. Neste cenário foi identificado uma extensão do presente trabalho:

($a$.1) Estudar outras formas para a correção da função de custo que utilizará uma matriz de transição de ruído por usuário e/ou por item, de forma a atingir robustez para ruídos do tipo NNAR.

(b) **Ruído Malicioso**

Além do ruído natural, existe outro tipo de ruído, denominado ruído malicioso (*shilling attacks*). Diferentemente do natural, esse tipo de ruído está associado a alguma inconsistência acerca dos dados que foram introduzidos intencionalmente por um agente externo. Usuários maliciosos ou empresas rivais podem inserir usuários falsos dentro do sistema, inserindo avaliações com o objetivo de afetar o algoritmo de recomendação por fins próprios. Alguns ataques podem intencionalmente aumentar a popularidade de alguns itens (*push attacks*) e outros podem prejudicar (*nuke attacks*). (GUNES *et al.*, 2012)

No caso de Sistemas de Recomendação que utilizam a filtragem colaborativa, é essencial lidar com esse tipo de problema. Surgiu uma linha de pesquisa que tenta identificar quando acontece um ataque, para assim tomar uma ação após a detecção. Uma outra solução é a criação de algoritmos que sejam menos suscetíveis ao impacto de um ataque. Neste cenário foi identificado um possível trabalho:

($b$.1) Avaliar o modelo proposto no cenário do ruído malicioso e compará-lo a diversos tipos de ataques com os outros métodos da literatura.

(c) **Base de Dados e Outros Tipos de Ruídos**

No presente trabalho foram escolhidas três bases com características distintas para serem usadas na avaliação do método proposto. Como não há base de dados, até então, com marcações indicativas de ruído, supôs-se ausência de ruído nas bases e foram necessárias inserções de ruído artificial nestas. Com propósito de avaliar a robustez do modelo, foram propostos dois métodos de geração de ruído artificial. O primeiro do tipo NNAR e o segundo do tipo NAR. E, neste contexto, foram enumerados alguns trabalhos futuros da presente tese na perspectiva da avaliação do modelo:

($c$.1) Avaliar a robustez do modelo em outras bases de dados que possuam características distintas das que foram utilizadas no presente trabalho.

($c$.2) Propor outros métodos para a geração de ruído, principalmente para o ruído do tipo NNAR. Avaliar a robustez do método proposto através dessa nova geração.

(d) **Outras Arquiteturas Neurais**

A tese utilizou redes neurais como o modelo de previsão e a arquitetura da rede foi a MLP do *framework Neural Collaborative Filtering* (HE *et al.*, 2017). A arquitetura foi escolhida devido a sua simplicidade e para validar a robustez do método. Contudo, é possível aumentar a complexidade do modelo fazendo surgir alguns trabalhos futuros:

($d$.1) Utilizar a função de custo corrigida em conjunto da matriz de transição de ruído estimada no treinamento com outras arquiteturas neurais como a *NeuMF* (HE *et al.*, 2017).

($d$.2) Utilizar a função de custo corrigida utilizando a matriz de transição de ruído estimada em um *autoencoder* para a extração das variáveis latentes no *framework* proposto por BRAIDA *et al.* (2015) e estendido por BARBIERI *et al.* (2017).

($d$.3) Utilizar a função de custo corrigida utilizando a matriz de transição de ruído estimada no algoritmo *Autorec* (SEDHAIN *et al.*, 2015b).

(e) **Âncoras**

A proposta para estimar a matriz de transição de ruído para a filtragem colaborativa se deu através da escolha de avaliações âncoras, ou seja, avaliações que possuem uma grande chance de estarem corretas na base. Para tal, utilizou-se um modelo de previsão e verificou quais as avaliações ele conseguiria prever com exatidão. Através desse subconjunto de avaliações, foi utilizado o cálculo

da média das probabilidades estimadas deste subconjunto ao invés de um único elemento para estimar a matriz. Assim, temos:

($e$.1) Utilizar a incerteza do estimador para determinar a confiabilidade de uma nota e assim através de um limiar ele ser considerado um âncora.

($e$.2) Verificar e validar outros valores para o parâmetro para detecção do âncora.

($e$.3) Utilizar o valor padrão de 5% de ruído quando o número de âncoras não for encontrado.

($e$.4) Utilizar esse método de encontrar os âncoras para estimar a matriz de transição em outros problemas que possuem a mesma característica da filtragem colaborativa na qual não é possível possuir as características prévias necessárias para o Teorema 3.

# Referências Bibliográficas


GARCÍA, S., LUENGO, J., HERRERA, F., et al. *Data Preprocessing in Data Mining*, v. 72, *Intelligent Systems Reference Library*. Springer, 2015. ISBN: 978-3-319-10246-7. doi: 10.1007/978-3-319-10247-4. Disponível em: <http://dx.doi.org/10.1007/978-3-319-10247-4http://www.scopus.com/inward/record.url?eid=2-s2.0-84906871736{&}partnerID=tZOtx3y1>.

FRENAY, B., VERLEYSEN, M. "Classification in the Presence of Label Noise: a Survey", *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 5, pp. 845–869, 2013. ISSN: 2162237X. doi: 10.1109/TNNLS.2013.2292894.

SUKHBAATAR, S., FERGUS, R. "Learning from Noisy Labels with Deep Neural Networks", *arXiv preprint arXiv:1406.2080*, v. 2, n. 3, pp. 4, 2014.

XIAO, T., XIA, T., YANG, Y., et al. "Learning from massive noisy labeled data for image classification", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, v. 07-12-June, pp. 2691–2699, 2015. ISSN: 10636919. doi: 10.1109/CVPR.2015.7298885.

NATARAJAN, N., DHILLON, I. S., RAVIKUMAR, P., et al. "Learning with Noisy Labels", *Advances in neural information processing systems*, pp. 1196–1204, 2013. ISSN: 10495258.

ROOYEN, B. V. *Machine Learning via Transitions*. Tese de Doutorado, The Australian National University, 2015.

KOREN, Y., BELL, R., VOLINSKY, C. "Matrix factorization techniques for recommender systems", *Computer*, v. 42, n. 8, pp. 30–37, 2009. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5197422>.

HE, X., LIAO, L., ZHANG, H., et al. "Neural Collaborative Filtering", pp. 173–182, 2017. ISSN: 10535888. doi: 10.1145/3038912.3052569. Disponível em: <http://arxiv.org/abs/1708.05031>.

TOLEDO, R. Y., MOTA, Y. C., MARTÍNEZ, L. "Correcting noisy ratings in collaborative recommender systems", *Knowledge-Based Systems*, v. 76, pp. 96–108, 2015. ISSN: 09507051. doi: 10.1016/j.knosys.2014.12.011. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2014.12.011>.

YERA, R., CASTRO, J., MARTÍNEZ, L. "A fuzzy model for managing natural noise in recommender systems", *Applied Soft Computing*, v. 40, pp. 187–198, 2016. ISSN: 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2015.10.060. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494615007048>.

O'MAHONY, M. P., HURLEY, N. J., SILVESTRE, G. C. "Detecting noise in recommender system databases", *Proceedings of the 11th international conference on Intelligent user interfaces - IUI '06*, p. 109, 2006. doi: 10.1145/1111449.1111477. Disponível em: <http://portal.acm.org/citation.cfm?doid=1111449.1111477>.

PATRINI, G., NIELSEN, F., NOCK, R., et al. "Loss factorization, weakly supervised learning and label noise robustness", 2016a. Disponível em: <http://arxiv.org/abs/1602.02450>.

SÁEZ, J. A., GALAR, M., LUENGO, J., et al. "Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition", *Knowledge and Information Systems*, v. 38, n. 1, pp. 179–206, 2014.

ZHU, X., WU, X. "Class Noise vs. Attribute Noise: A Quantitative Study", *Artificial Intelligence Review*, v. 22, n. 3, pp. 177–210, 2004. ISSN: 0269-2821. doi: 10.1007/s10462-004-0751-8.

ADOMAVICIUS, G., TUZHILIN, A. "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions", *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 6, pp. 734–749, jun 2005. ISSN: 1041-4347. doi: 10.1109/TKDE.2005.99.

AMATRIAIN, X., PUJOL, J. M., OLIVER, N. "I like it... i like it not: Evaluating user ratings noise in recommender systems", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 5535 LNCS, pp. 247–258, 2009a. ISSN: 03029743. doi: 10.1007/978-3-642-02247-0_24.

PHAM, H. X., JUNG, J. J. "Preference-based User Rating Correction Process for Interactive Recommendation Systems", *Multimedia Tools Appl.*,

v. 65, n. 1, pp. 119–132, jul. 2013. ISSN: 1380-7501. doi: 10.1007/s11042-012-1119-8. Disponível em: <http://dx.doi.org/10.1007/s11042-012-1119-8>.

YU, X., LIU, T., GONG, M., et al. "Learning with Biased Complementary Labels", pp. 1–25, 2017. Disponível em: <https://arxiv.org/pdf/1711.09535.pdfhttp://arxiv.org/abs/1711.09535>.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA, Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

ANGLUIN, D., LAIRD, P. "Learning from noisy examples", *Machine Learning*, v. 2, n. 1984, pp. 343–370, 1988. ISSN: 08856125. doi: 10.1007/BF00116829.

QUINLAN, J. R. "Induction of Decision Trees", *Machine Learning*, v. 1, n. 1, pp. 81–106, 1986. ISSN: 15730565. doi: 10.1023/A:1022643204877.

SALMON, V. "The nature of noise", *Water, Air, and Soil Pollution*, v. 2, n. 3, pp. 257–265, 1973. ISSN: 0049-6979. doi: 10.1007/BF00159658. Disponível em: <http://dx.doi.org/10.1007/BF00159658>.

VAN DEN HOUT, A., VAN DER HEIJDEN, P. G. M., VAN DEN HOUT, A., et al. "Randomized Response, Statistical Disclosure Control and Misclassification: A Review", *International Statistical Review / Revue Internationale de Statistique*, v. 70, n. 2, pp. pp. 269—-288, 2002. ISSN: 03067734. doi: 10.2307/1403910. Disponível em: <http://www.jstor.org/stable/1403910>.

BARANDELA, R., GASCA, E. "Decontamination of Training Samples for Supervised Pattern Recognition Methods", pp. 621–630, 2000. ISSN: 03029743.

JO, T., JAPKOWICZ, N. "Class imbalances versus small disjuncts", *ACM SIGKDD Explorations Newsletter*, v. 6, n. 1, pp. 40, 2004. ISSN: 19310145. doi: 10.1145/1007730.1007737.

GARCÍA, V., MOLLINEDA, R. A., SÁNCHEZ, J. S. "On the k-NN performance in a challenging scenario of imbalance and overlapping", *Pattern Analysis and Applications*, v. 11, n. 3-4, pp. 269–280, 2008. ISSN: 14337541. doi: 10.1007/s10044-007-0087-5.

GARCÍA, V., SÁNCHEZ, J., MOLLINEDA, R. "An empirical study of the behavior of classifiers on imbalanced and overlapped data sets", *Progress in Pattern Recognition, Image Analysis and Applications, Proceedings*, v. 4756, pp. 397–406, 2007. ISSN: 0302-9743. doi: 10.1007/978-3-540-76725-1_42.

HUBER, P. *Robust Statistics*. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 2004. ISBN: 9780471650720.

COLLETT, D., LEWIS, T. "The Subjective Nature of Outlier Rejection Procedures", *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, v. 25, n. 3, pp. 228–237, 1976. ISSN: 00359254.

LIU, X., CHENG, G., WU, J. X. "Analyzing Outliers Cautiously", *IEEE Trans. on Knowl. and Data Eng.*, v. 14, n. 2, pp. 432–437, 2002. ISSN: 1041-4347. doi: 10.1109/69.991726. Disponível em: <http://dx.doi.org/10.1109/69.991726>.

GARCÍA, V., ALEJO, R., SÁNCHEZ, J. S., et al. "Combined Effects of Class Imbalance and Class Overlap on Instance-Based Classification". In: Corchado, E., Yin, H., Botti, V., et al. (Eds.), *Intelligent Data Engineering and Automated Learning – IDEAL 2006: 7th International Conference, Burgos, Spain, September 20-23, 2006. Proceedings*, pp. 371–378, Berlin, Heidelberg, Springer Berlin Heidelberg, 2006. ISBN: 978-3-540-45487-8. doi: 10.1007/11875581_45. Disponível em: <http://dx.doi.org/10.1007/11875581_45>.

NAPIERAŁA, K., STEFANOWSKI, J., WILK, S. "Learning from Imbalanced Data in Presence of Noisy and Borderline Examples". In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., et al. (Eds.), *Rough Sets and Current Trends in Computing: 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30,2010. Proceedings*, pp. 158–167, Berlin, Heidelberg, Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-13529-3. doi: 10.1007/978-3-642-13529-3_18. Disponível em: <http://dx.doi.org/10.1007/978-3-642-13529-3_18>.

HICKEY, R. J. "Noise modelling and evaluating learning from examples", *Artificial Intelligence*, v. 82, n. 1–2, pp. 157–179, apr 1996. ISSN: 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(94)00094-8. Disponível em: <http://www.sciencedirect.com/science/article/pii/0004370294000948>.

BRODLEY, C. E., FRIEDL, M. A. "Identifying Mislabeled Training Data", *Journal of Artificial Intelligence Research*, v. 11, pp. 131–167, 1999. ISSN: 10769757. doi: 10.1613/jair.606.

HARTONO, P., HASHIMOTO, S. "Learning from imperfect data", *Applied Soft Computing Journal*, v. 7, n. 1, pp. 353–363, 2007. ISSN: 15684946. doi: 10.1016/j.asoc.2005.07.005.

DAWID, A. P., SKENE, A. M. "Maximum likelihood estimation of observer error-rates using the EM algorithm", *Journal of the Royal Statistical Society Series C Applied Statistics*, v. 28, n. 1, pp. 20–28, 1979. ISSN: 00359254. doi: 10.2307/2346806. Disponível em: <http://www.jstor.org/stable/2346806>.

SNOW, R., O'CONNOR, B., JURAFSKY, D., et al. "Cheap and Fast—but is It Good?: Evaluating Non-expert Annotations for Natural Language Tasks". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pp. 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. Disponível em: <http://dl.acm.org/citation.cfm?id=1613715.1613751>.

FRÉNAY, B., KABÁN, A. "A Comprehensive Introduction to Label Noise", *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, , n. April, pp. 23–25, 2014. Disponível em: <https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2014-10.pdf>.

ORR, K. "Data Quality and Systems Theory", *Commun. ACM*, v. 41, n. 2, pp. 66–71, fev. 1998. ISSN: 0001-0782. doi: 10.1145/269012.269023. Disponível em: <http://doi.acm.org/10.1145/269012.269023>.

NETTLETON, D. F., ORRIOLS-PUIG, A., FORNELLS, A. "A study of the effect of different types of noise on the precision of supervised learning techniques", *Artificial Intelligence Review*, v. 33, n. 3-4, pp. 275–306, 2010. ISSN: 02692821. doi: 10.1007/s10462-010-9156-z.

SCHAFER, J. L., GRAHAM, J. W. "Missing data: Our view of the state of the art." *Psychological Methods*, v. 7, n. 2, pp. 147–177, 2002. ISSN: 1082-989X. doi: 10.1037//1082-989X.7.2.147. Disponível em: <http://doi.apa.org/getdoi.cfm?doi=10.1037/1082-989X.7.2.147>.

KALAI, A. T., SERVEDIO, R. A. "Boosting in the presence of noise", *Journal of Computer and System Sciences*, v. 71, n. 3, pp. 266–290, 2005. ISSN: 00220000. doi: 10.1016/j.jcss.2004.10.015.

ASLAM, J. A., DECATUR, S. E. "On the sample complexity of noise-tolerant learning", *Information Processing Letters*, v. 57, n. 4, pp. 189–195, 1996. ISSN: 00200190. doi: 10.1016/0020-0190(96)00006-3.

SANDERSON, T., SCOTT, C. "Class Proportion Estimation with Application to Multiclass Anomaly Rejection." *Aistats*, v. 33, n. 1, pp. 850–858, 2014. ISSN: 15337928.

RAMASWAMY, H. G., SCOTT, C., TEWARI, A. "Mixture Proportion Estimation via Kernel Embedding of Distributions", *ICML*, 2016. Disponível em: <http://arxiv.org/abs/1603.02501>.

MENON, A. K., ROOYEN, B. V., ONG, C. S., et al. "Learning from Corrupted Binary Labels via Class-Probability Estimation", v. 37, 2015.

LIU, T., TAO, D. "Classification with Noisy Labels by Importance Reweighting", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 38, n. 3, pp. 447–461, 2016. ISSN: 01628828. doi: 10.1109/TPAMI.2015.2456899.

BREIMAN, L. E. O. "Randomizing Outputs to Increase Prediction Accuracy", *Mach. Learn.*, v. 40, n. 3, pp. 229–242, 2000. ISSN: 0885-6125. doi: 10.1023/A:1007682208299. Disponível em: <http://dx.doi.org/10.1023/A:1007682208299>.

MARTÍNEZ-MUÑOZ, G., SUÁREZ, A. "Switching class labels to generate classification ensembles", *Pattern Recognition*, v. 38, n. 10, pp. 1483–1494, 2005. ISSN: 00313203. doi: 10.1016/j.patcog.2005.02.020.

MARTÍNEZ-MUÑOZ, G., SÁNCHEZ-MARTÍNEZ, A., HERNÁNDEZ-LOBATO, D., et al. "Class-switching neural network ensembles", *Neurocomputing*, v. 71, n. 13-15, pp. 2521–2528, 2008. ISSN: 09252312. doi: 10.1016/j.neucom.2007.11.041.

MARTÍNEZ-MUÑOZ, G., SÁNCHEZ-MARTÍNEZ, A., HERNÁNDEZ-LOBATO, D., et al. "Building Ensembles of Neural Networks with Class-Switching." *Icann (1)*, pp. 178–187, 2006. ISSN: 03029743.

WILSON, D. R., MARTINEZ, T. R. "Reduction Techniques for Instance-Based Learning Algorithms", *Machine Learning*, v. 38, n. 3, pp. 257–286, 2000.

ISSN: 1573-0565. doi: 10.1023/A:1007626913721. Disponível em: <http://dx.doi.org/10.1023/A:1007626913721>.

SÁNCHEZ, J., PLA, F., FERRI, F. "Prototype selection for the nearest neighbour rule through proximity graphs", *Pattern Recognition Letters*, v. 18, n. 6, pp. 507 – 513, 1997. ISSN: 0167-8655. doi: http://dx.doi.org/10.1016/S0167-8655(97)00035-4. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167865597000354>.

OKAMOTO, S., NOBUHIRO, Y. "An Average-case Analysis of the K-nearest Neighbor Classifier for Noisy Domains". In: *Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'97, pp. 238–243, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN: 1-555860-480-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1624162.1624198>.

ABELLÁN, J., MASEGOSA, A. R. "Foundations of Information and Knowledge Systems: 6th International Symposium, FoIKS 2010, Sofia, Bulgaria, February 15-19, 2010. Proceedings". cap. Bagging Decision Trees on Data Sets with Classification Noise, pp. 248–265, Berlin, Heidelberg, Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-11829-6. doi: 10.1007/978-3-642-11829-6_17. Disponível em: <http://dx.doi.org/10.1007/978-3-642-11829-6_17>.

LIBRALON, G. L., CARVALHO, A. C. P. D. L. F., LORENA, A. C. "Pre-processing for noise detection in gene expression classification data", *Journal of the Brazilian Computer Society*, v. 15, n. 1, pp. 3–11. ISSN: 1678-4804. doi: 10.1007/BF03192573. Disponível em: <http://dx.doi.org/10.1007/BF03192573>.

LORENA, A. C., DE CARVALHO, A. C. P. L. F. "Evaluation of noise reduction techniques in the splice junction recognition problem", *Genetics and Molecular Biology*, v. 27, n. 4, pp. 665–672, 2004. ISSN: 14154757. doi: 10.1590/S1415-47572004000400031.

SEGATA, N., BLANZIERI, E., CUNNINGHAM, P. "Case-Based Reasoning Research and Development: 8th International Conference on Case-Based Reasoning, ICCBR 2009 Seattle, WA, USA, July 20-23, 2009 Proceedings". cap. A Scalable Noise Reduction Technique for Large Case-Based Systems, pp. 328–342, Berlin, Heidelberg, Springer Berlin Heidelberg, 2009. ISBN: 978-3-642-02998-1. doi: 10.1007/978-3-642-02998-1_24. Disponível em: <http://dx.doi.org/10.1007/978-3-642-02998-1_24>.

CORMACK, G. V., KOLCZ, A. "Spam Filter Evaluation with Imprecise Ground Truth". In: *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pp. 604–611, New York, NY, USA, 2009. ACM. ISBN: 978-1-60558-483-6. doi: 10.1145/1571941.1572045. Disponível em: <http://doi.acm.org/10.1145/1571941.1572045>.

GOLDBERGER, J., BEN-REUVEN, E. "Training Deep Neural Networks using a Noise Adaptation Layer", , n. 2014, pp. 1–9, 2017. Disponível em: <https://openreview.net/forum?id=H12GRgcxg>.

STEMPFEL, G., RALAIVOLA, L. "Learning SVMs from sloppily labeled data", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 5768 LNCS, n. PART 1, pp. 884–893, 2009. ISSN: 03029743. doi: 10.1007/978-3-642-04274-4_91.

REED, S., LEE, H., ANGUELOV, D., et al. "Training Deep Neural Networks on Noisy Labels with Bootstrapping", pp. 1–11, 2014. ISSN: 1939-4608. doi: 10.2200/S00196ED1V01Y200906AIM006. Disponível em: <http://arxiv.org/abs/1412.6596>.

PATRINI, G., ROZZA, A., MENON, A., et al. "Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach", 2016b. doi: 10.1109/CVPR.2017.240. Disponível em: <http://arxiv.org/abs/1609.03683>.

MALACH, E., SHALEV-SHWARTZ, S. "Decoupling "when to update"from "how to update"", 2017. Disponível em: <http://arxiv.org/abs/1706.02613>.

SCHWARTZ, B. *The Paradox of Choice: Why More Is Less*. Harper Perennial, January 2005. ISBN: 0060005696.

SARWAR, B., KARYPIS, G., KONSTAN, J., et al. "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering". In: *Proceedings of the Fifth International Conference on Computer and Information Technology*, pp. 158–167, 2002. Disponível em: <http://grouplens.org/papers/pdf/sarwar_cluster.pdf>.

LINDEN, G., SMITH, B., YORK, J. "Amazon. com recommendations: Item-to-item collaborative filtering", *Internet Computing, IEEE*, v. 7, n. 1,

pp. 76–80, 2003. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1167344>.

BURKE, R. "Hybrid web recommender systems". In: *The adaptive web*, pp. 377–408. Springer-Verlag, 2007. Disponível em: <http://dl.acm.org/citation.cfm?id=1768211>.

RICCI, F., ROKACH, L., SHAPIRA, B., et al. "Recommender Systems Handbook", *Media*, 2011. doi: 10.1007/978-0-387-85820-3. Disponível em: <http://www.springerlink.com/index/10.1007/978-0-387-85820-3>.

SARWAR, B., KARYPIS, G., KONSTAN, J., et al. "Item-based collaborative filtering recommendation algorithms". In: *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295. ACM, 2001a.

MELLO, C., AUFAURE, M., ZIMBRAO, G. "Active learning driven by rating impact analysis". In: *Proceedings of the fourth ACM conference on Recommender systems*, pp. 341–344. ACM, 2010. Disponível em: <http://portal.acm.org/citation.cfm?id=1864708.1864782>.

SARWAR, B., KARYPIS, G., KONSTAN, J., et al. "Item-based collaborative filtering recommendation algorithms". In: *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295. ACM, 2001b. ISBN: 1581133480. Disponível em: <http://portal.acm.org/citation.cfm?id=371920.372071>.

GOLDBERG, K., ROEDER, T., GUPTA, D., et al. "Eigentaste: A constant time collaborative filtering algorithm", *Information Retrieval*, v. 4, n. 2, pp. 133–151, 2001. ISSN: 1386-4564. Disponível em: <http://www.springerlink.com/index/M5458KV8LJ602646.pdf>.

BILLSUS, D. "Learning collaborative information filters", *International Conference on Machine Learning*, 1998. Disponível em: <http://www.aaai.org/Papers/Workshops/1998/WS-98-08/WS98-08-005.pdf>.

GUNES, I., KALELI, C., BILGE, A., et al. "Shilling attacks against recommender systems: a comprehensive survey", *Artificial Intelligence Review*, v. 42, n. 4, pp. 767–799, 2012. ISSN: 02692821. doi: 10.1007/s10462-012-9364-9.

HAN, J. *Data Mining: Concepts and Techniques*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2005. ISBN: 1558609016.

GAMA, J. A., ŽLIOBAITĖ, I., BIFET, A., et al. "A Survey on Concept Drift Adaptation", *ACM Comput. Surv.*, v. 46, n. 4, pp. 44:1–44:37, mar. 2014. ISSN: 0360-0300. doi: 10.1145/2523813. Disponível em: <http://doi.acm.org/10.1145/2523813>.

HUANG, C., GONG, S. "Employing rough set theory to alleviate the sparsity issue in recommender system". In: *Machine Learning and Cybernetics, 2008 International Conference on*, v. 3, pp. 1610–1614. IEEE, 2008. ISBN: 9781424420964. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4620663>.

SCHAFER, J., KONSTAN, J., RIEDI, J. "Recommender systems in e-commerce". In: *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158–166. ACM, 1999. ISBN: 1581131763. Disponível em: <http://dl.acm.org/citation.cfm?id=337035>.

SU, X., KHOSHGOFTAAR, T. M. "A Survey of Collaborative Filtering Techniques", *Advances in Artificial Intelligence*, v. 2009, n. Section 3, pp. 1–20, 2009. ISSN: 1687-7470. doi: 10.1155/2009/421425. Disponível em: <http://www.hindawi.com/journals/aai/2009/421425.html>.

SHARDANAND, U., MAES, P. "Social Information Filtering : Algorithms for Automating "Word of Mouth"". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217. ACM Press/Addison-Wesley Publishing Co., 1995. Disponível em: <http://dl.acm.org/citation.cfm?id=223931>.

FUNK, S. "Netflix Update: Try this at Home", 2006. Disponível em: <http://sifter.org/~simon/journal/20061211.html>.

DUMAIS, S. T. "Latent semantic analysis", *Annual Review of Information Science and Technology*, v. 38, n. 1, pp. 188–230, sep 2005. ISSN: 00664200. doi: 10.1002/aris.1440380105. Disponível em: <http://doi.wiley.com/10.1002/aris.1440380105>.

PATEREK, A. "Improving regularized singular value decomposition for collaborative filtering". In: *Proceedings of KDD Cup and Workshop*, v. 2007, pp. 5–8, 2007. ISBN: 9781595938343.

BRAIDA, F., MELLO, C. E., PASINATO, M. B., et al. "Transforming collaborative filtering into supervised learning", *Expert Systems with Applications*, v. 42, n. 10, pp. 4733–4742, 2015. ISSN: 0957-4174. doi: http:

//dx.doi.org/10.1016/j.eswa.2015.01.023. Disponível em: <http://www.sciencedirect.com/science/article/pii/S095741741500038X>.

ZHANG, S., YAO, L., SUN, A. "Deep Learning based Recommender System: A Survey and New Perspectives", v. 1, n. 1, pp. 1–35, 2017. ISSN: 15232867. Disponível em: <http://arxiv.org/abs/1707.07435>.

SEDHAIN, S., MENON, A. K., SANNER, S., et al. "Autorec: Autoencoders meet collaborative filtering". In: *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 111–112. International World Wide Web Conferences Steering Committee, 2015a.

BARBIERI, J., ALVIM, L. G., BRAIDA, F., et al. "Autoencoders and recommender systems: COFILS approach", *Expert Systems with Applications*, v. 89, pp. 81 – 90, 2017. ISSN: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.07.030. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417417305079>.

LI, B., CHEN, L., ZHU, X., et al. "Noisy but non-malicious user detection in social recommender systems", *World Wide Web*, v. 16, n. 5-6, pp. 677–699, 2013. ISSN: 1386145X. doi: 10.1007/s11280-012-0161-9.

DING, Y., LI, X., ORLOWSKA, M. E. "Recency-based Collaborative Filtering". In: *Proceedings of the 17th Australasian Database Conference - Volume 49*, ADC '06, pp. 99–107, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN: 1-920682-31-7. Disponível em: <http://dl.acm.org/citation.cfm?id=1151736.1151747>.

RICCI, F., ROKACH, L., SHAPIRA, B., et al. *Recommender Systems Handbook*. 1st ed. New York, NY, USA, Springer-Verlag New York, Inc., 2010. ISBN: 0387858199, 9780387858197.

KLUVER, D., NGUYEN, T. T., EKSTRAND, M., et al. "How many bits per rating?" *Proceedings of the sixth ACM conference on Recommender systems - RecSys '12*, p. 99, 2012. doi: 10.1145/2365952.2365974. Disponível em: <http://dl.acm.org/citation.cfm?doid=2365952.2365974>.

SAID, A., JAIN, B. J., NARR, S., et al. "Users and noise: The magic barrier of recommender systems". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 7379 LNCS, pp. 237–248, 2012. ISBN: 9783642314537. doi: 10.1007/978-3-642-31454-4_20.

HU, R., PU, P. "Exploring relations between personality and user rating behaviors". In: *CEUR Workshop Proceedings*, v. 997, pp. 139–142, 2013.

CANTADOR, I., FERNÁNDEZ-TOBÍAS, I., BELLOGÍN, A. "Relating Personality Types with User Preferences in Multiple Entertainment Domains". In: Berkovsky, S., Herder, E., Lops, P., et al. (Eds.), *Late-Breaking Results, Project Papers and Workshop Proceedings of the 21st Conference on User Modeling, Adaptation, and Personalization., Rome, Italy, June 10-14, 2013*, v. 997, *CEUR Workshop Proceedings*. CEUR-WS.org, 2013. Disponível em: <http://ceur-ws.org/Vol-997/empire2013_paper_2.pdf>.

KOREN, Y. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 426–434, New York, NY, USA, 2008. ACM. ISBN: 978-1-60558-193-4. doi: 10.1145/1401890.1401944.

EKSTRAND, M. D., HARPER, F. M., WILLEMSEN, M. C., et al. "User Perception of Differences in Recommender Algorithms". In: *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pp. 161–168, New York, NY, USA, 2014. ACM. ISBN: 978-1-4503-2668-1. doi: 10.1145/2645710.2645737. Disponível em: <http://doi.acm.org/10.1145/2645710.2645737>.

BALTRUNAS, L. *Context-Aware Collaborative Filtering Recommender Systems*. Tese de Doutorado, 2011.

REID, M. D., WILLIAMSON, R. C. "Composite Binary Losses", *The Journal of Machine Learning Research*, v. 11, pp. 2387–2422, 2010.

AMATRIAIN, X., PUJOL, J., TINTAREV, N., et al. "Rate it again: increasing recommendation accuracy by user re-rating". In: *Proceedings of the third ACM conference on Recommender systems*, n. 4, pp. 173–180. ACM, 2009b. Disponível em: <http://portal.acm.org/citation.cfm?id=1639714.1639744>.

ZHU, X., WU, X., CHEN, Q. "Eliminating class noise in large datasets". In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 920–927, 2003.

JINDAL, I., NOKLEBY, M., CHEN, X. "Learning deep networks from noisy labels with dropout regularization", *Proceedings - IEEE International Con-*

*ference on Data Mining, ICDM*, pp. 967–972, 2017. ISSN: 15504786. doi: 10.1109/ICDM.2016.124.

BRUZZONE, L., PERSELLO, C. "A novel context-sensitive semisupervised SVM classifier robust to mislabeled rraining samples", *IEEE Transactions on Geoscience and Remote Sensing*, v. 47, n. 7, pp. 2142–2154, 2009. ISSN: 01962892. doi: 10.1109/TGRS.2008.2011983.

GARCIA, L. P., DE CARVALHO, A. C., LORENA, A. C. "Noise detection in the meta-learning level", *Neurocomputing*, v. 176, pp. 14–25, 2016. ISSN: 18728286. doi: 10.1016/j.neucom.2014.12.100. Disponível em: <`http://dx.doi.org/10.1016/j.neucom.2014.12.100`>.

BERGSTRA, J. S., BARDENET, R., BENGIO, Y., et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems*, pp. 2546–2554, 2011.

RECHENBER, I. *Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Tese de Doutorado, Verlag nicht ermittelbar, 1970.

KINGMA, D. P., BA, J. "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

MILLER, B., ALBERT, I., LAM, S. "MovieLens unplugged: experiences with an occasionally connected recommender system", *Proceedings of the 8th ...*, pp. 263–266, 2003. Disponível em: <`http://dl.acm.org/citation.cfm?id=604094`>.

GUO, G., ZHANG, J., THALMANN, D., et al. "ETAF: An Extended Trust Antecedents Framework for Trust Prediction". In: *Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 540–547, 2014.

HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., et al. "Evaluating collaborative filtering recommender systems", *ACM Transactions on Information Systems*, v. 22, n. 1, pp. 5–53, jan 2004. ISSN: 10468188. doi: 10.1145/963770.963772. Disponível em: <`http://portal.acm.org/citation.cfm?doid=963770.963772`>.

SEDHAIN, S., MENON, A. K., SANNER, S., et al. "AutoRec : Autoencoders Meet Collaborative Filtering", *WWW 2015 Companion: Proceedings of the 24th International Conference on World Wide Web*, pp. 111–112, 2015b. doi: 10.1145/2740908.2742726.