ADILSON LOPES KHOURI

Desenvolvimento de técnica para recomendar atividades em *workflows* científicos: uma abordagem baseada em ontologias

São Paulo

2016

ADILSON LOPES KHOURI

Desenvolvimento de técnica para recomendar atividades em *workflows* científicos: uma abordagem baseada em ontologias

Versão corrigida

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 16 de Março de 2016. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Prof. Dr. Luciano Antonio Digiampietri

São Paulo

2016

CATALOGAÇÃO-NA-PUBLICAÇÃO
(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)

Khouri, Adilson Lopes
Desenvolvimento de técnica para recomendar atividades em workflows científicos: uma abordagem baseada em ontologias / Adilson Lopes Khouri ; orientador, Luciano Antonio Digiampietri. – São Paulo, 2016
101 f. : il.

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo
Versão corrigida

1. Desenvolvimento de software. 2. Sistemas de informação gerencial. 3. Ontologia. I. Digiampietri, Luciano Antonio, orient. II. Título

CDD 22.ed. – 005.12

Dissertação de autoria de Adilson Lopes Khouri, sob o título **"Desenvolvimento de técnica para recomendar atividades em *workflows* científicos: uma abordagem baseada em ontologias"**, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 16 de Março de 2016 pela comissão julgadora constituída pelos doutores:

Prof. Dr. Luciano Antonio Digiampietri
Presidente
Universidade de São Paulo (USP)

Prof. Dr. Ivandre Paraboni
Universidade de São Paulo (USP)

Prof. Dr. Marcio Katsumi Oikawa
Universidade Federal do ABC (UFABC)

‘‘Dedico este trabalho para Deus e seus assistentes, para minha família (Arthur, Claudia e Gimayma) e para o meu orientador (professor Dr. Luciano).’’Adilson Lopes Khouri, 2016.

Agradecimentos

Agradecemos a Pró-Reitoria de Pós-Graduação da Universidade de São Paulo (USP) e a agência CAPES que forneceram bolsas de estudo para o estudante. Permitindo completar esse mestrado com publicações na área de computação. Além disso, agradecemos o professor Dr. Clodoaldo Aparecido de Lima por sanar dúvidas referentes a técnica SVM.

‘‘Odin deu um olho em troca de sabedoria. Pois fique sabendo que eu daria muito mais!’’

(Ragnar Lothbrok).

Resumo

KHOURI, Adilson Lopes. **Desenvolvimento de técnica para recomendar atividades em *workflows* científicos** : uma abordagem baseada em ontologias. 2016. 101 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2016.

O número de atividades disponibilizadas pelos sistemas gerenciadores de *workflows* científicos é grande, o que exige dos cientistas conhecerem muitas delas para aproveitar a capacidade de reutilização desses sistemas. Para minimizar este problema, a literatura apresenta algumas técnicas para recomendar atividades durante a construção de *workflows* científicos. Este projeto especificou e desenvolveu um sistema de recomendação de atividades híbrido, considerando informação sobre frequência, entrada e saídas das atividades, e anotações ontológicas para recomendar. Além disso, neste projeto é apresentada uma modelagem da recomendação de atividades como um problema de classificação e regressão, usando para isso cinco classificadores; cinco regressores; um classificador SVM composto, o qual usa o resultado dos outros classificadores e regressores para recomendar; e um *ensemble* de classificadores *Rotation Forest*. A técnica proposta foi comparada com as outras técnicas da literatura e com os classificadores e regressores, por meio da validação cruzada em 10 subconjuntos, apresentando como resultado uma recomendação mais precisa, com medida MRR ao menos 70% maior do que as obtidas pelas outras técnicas.

Palavras-chaves: Sistemas de recomendação. Ontologias. Sistemas gerenciadores de *workflows* científicos. Modelagem de classificadores e regressores.

Abstract

KHOURI, Adilson Lopes. **Development of a strategy to scientific workflow activities recommendation**: An ontology-based approach. 2016. 101 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2016.

The number of activities provided by scientific workflow management systems is large, which requires scientists to know many of them to take advantage of the reusability of these systems. To minimize this problem, the literature presents some techniques to recommend activities during the scientific workflow construction. This project specified and developed a hybrid activity recommendation system considering information on frequency, input and outputs of activities and ontological annotations. Additionally, this project presents a modeling of activities recommendation as a classification problem, tested using 5 classifiers; 5 regressors; a SVM classifier, which uses the results of other classifiers and regressors to recommend; and Rotation Forest , an ensemble of classifiers. The proposed technique was compared to other related techniques and to classifiers and regressors, using 10-fold-cross-validation, achieving a MRR at least 70% greater than those obtained by other techniques.

Keywords: Recommendation systems. Ontologies. Scientific Workflows Management Systems. Recommeder Systems.

Lista de figuras

Lista de tabelas

Sumário

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução, justificativa e objetivos

''Perdoe-me, meu amigo, não pelo o que eu fiz. Mas pelo que estou prestes a fazer.''

(Ragnar Lothbrok)

O termo *e-Science* se refere à ciência que é realizada com o uso intensivo de computadores. Em projetos da *e-Science* existe uma forte relação entre computação e outras áreas do conhecimento (biologia, astronomia, física, entre outras), tal que a primeira fornece ferramentas fundamentais para o sucesso de experimentos científicos computacionais da segunda. Um dos objetivos dessas ferramentas é ocultar detalhes técnicos computacionais, permitindo aos cientistas gerenciarem experimentos com maior facilidade (DEELMAN et al., 2009).

Uma das ferramentas para auxiliar na criação/manutenção de experimentos científicos computacionais são os sistemas gerenciadores de *workflows* científicos (SGWCs). Segundo Digiampietri (2007) e Wang et al. (2010), *workflows* científicos são processos estruturados e ordenados, construídos de forma manual, semiautomática ou automática e que permitem solucionar problemas científicos por meio de sua execução.

Um *workflow* denota a execução controlada de diversas atividades em um ambiente potencialmente distribuído. *Workflows* representam um conjunto de atividades a serem executadas, suas relações de interdependência, entradas e saídas (MEDEIROS et al., 2005). Atividades são as unidades básicas de um workflow e podem ser serviços Web, métodos em alguma linguagem de programação, etc.

Três grandes projetos de *e-Science* e seus desafios computacionais, solucionados com o auxílio de SGWCs, são enumerados por Olabarriaga et al. (2014). O primeiro é a visualização de grandes quantidades de dados Astrofísicos presentes no projeto internacional IVOA[1], o qual tem experimentos com petabytes de dados a serem visualizados. O segundo são cálculos matemáticos em ambientes distribuídos como no projeto internacional HELIO[2] criado por heliofísicos. O último são simulações genéticas em centros médicos de pesquisa como na universidade de Amsterdam (OLABARRIAGA et al., 2014).

Um *workflow científico* modela um experimento científico construído por meio de diversas atividades conectadas que realizam uma tarefa computacional. Alguns SGWCs permitem que seja armazenado o *log* de modelagem e execução do *workflow* junto com seus

[1] http://www.ivoa.net
[2] http://helio-vo.eu

parâmetros de execução, o que facilita sua execução por outros cientistas. As atividades usadas podem ser trechos de código fonte em Java (ou outra linguagem), aplicativos locais, serviços *web* ou outros *workflows* que foram encapsulados para esconder os detalhes computacionais, como a lógica de programação. Isso permite aos cientistas, sem grandes conhecimentos computacionais, ‘‘programarem’’ computadores para realizar seus experimentos sem se preocupar com detalhes de computação.

Atualmente, há um grande número de atividades disponíveis em repositórios como *myExperiment* (ROURE, 2015), que armazena mais de 2.500 *workflows* e *BioCatalogue* (BHAGAT et al., 2014) que disponibiliza 2.464 serviços de bioinformática. A existência desse grande número de atividades acarreta em um grande potencial de reuso (WANG et al., 2010), motivando a construção de sistemas para recomendar atividades aos cientistas durante a composição dos *workflows*. Estes sistemas, possibilitam reduzir a criação de novas atividades redundantes e o tempo total para a construção de *workflows*.

A recomendação de atividades em *workflows* possui algumas peculiaridades que justificam o desenvolvimento de novas técnicas (para recomendar atividades) de recomendação ou a extensão das existentes. Essas peculiaridades são: i) a esparsidade dos dados, isto é, cada *workflow* é tipicamente composto por poucas atividades e as bases de *workflows*, muitas vezes, contêm mais atividades do que *workflows*; ii) a dependência entre as atividades dos *workflows*: ao contrário da recomendação de itens como músicas, a ordem em que as atividades serão executadas é extremamente importante para a correta criação de um *workflow*; e iii) a diversidade da representação, documentação ou anotação das atividades e *workflows*: existem poucas atividades com descrições detalhadas (incluindo algumas descrições formais e anotações ontológicas), enquanto outras possuem apenas a definição dos tipos de dados usados na entrada e o tipo de dado produzido como saída da atividade.

Este mestrado apresenta três contribuições principais. A primeira é uma técnica mista para recomendar atividades em *workflows* baseada em frequência de atividades e ontologia de domínio. A segunda é a modelagem da recomendação de atividades como um problema de classificação, regressão, classificação composta (que utiliza dos resultados de outros experimentos para classificar) e um *ensemble* de classificadores. A terceira é a comparação do desempenho de diferentes técnicas de recomendação de atividades em *workflows* científicos utilizando-se uma mesma base de dados construída a partir de *workflows* reais.

A técnica de recomendação proposta neste projeto é genérica o suficiente para permitir sua aplicação em diferentes contextos, apesar de ter sido projetada para tratar as especificidades da recomendação de atividades em *workflows* científicos.

Os resultados deste mestrado podem auxiliar na redução da redundância de atividades, aumentando a reutilização de atividades existentes e, por conseguinte, reduzindo a necessidade da construção de novas atividades (redundantes) e o tempo necessário para se encontrar a atividade desejada. Segundo Cohen-Boulakia et al. (2014), uma das principais razões para a falta de reuso de *workflows* e atividades é a grande complexidade destes na área de bioinformática. Segundo o autor $98,1\%$ dos *workflows* contêm atividades redundantes ou são redundantes em relação a outros *workflows*.

1.1 Objetivos

O objetivo desse mestrado é criar uma técnica de recomendação de atividades com maior precisão do que as propostas pela literatura. Para tal, será elaborado um sistema híbrido de recomendação de atividades que usará Frequência, Entrada e Saída de atividades, e Ontologia de domínio (FESO).

A hipótese elaborada é: "Ao acrescentar informação semântica, sobre os dados, para um sistema de recomendação será observada uma melhoria nas métricas $S@k$ e MRR".

Como objetivos secundários, será organizada uma base de dados relacional para armazenar as atividades, os *workflows* e as anotações. Será estudado o domínio de bioinformática para elaborar uma ontologia de domínio, usada para anotar os *workflows* e as atividades. Por fim, será proposta uma nova modelagem do problema de recomendação para solucioná-lo como um problema de classificação ou regressão.

Esta dissertação está organizada com a seguinte estrutura: no capítulo 2 são apresentados conceitos básicos deste trabalho, o capítulo 3 apresenta a revisão da literatura correlata. O capítulo 4 descreve a solução baseada em ontologias e frequência, o capítulo 5 compara as soluções da literatura correlata com as propostas neste mestrado. Por fim, o capítulo 6 conclui o trabalho e apresenta possíveis projetos futuros.

2 Conceitos fundamentais

''Não tema a morte. Se ela vier, aceite-a como se estivesse se deitando com uma linda mulher.'' (Ragnar Lothbrok)

Este capítulo apresenta os conceitos básicos utilizados neste texto. Primeiramente são definidos os sistemas gerenciadores de *workflows* científicos e suas principais características (seção 2.1). As duas seções seguintes (2.2 e 2.3) detalham os tipos mais comuns de sistemas de recomendação e desafios para construir um sistema de recomendação de atividades em *workflows* científicos. A seção 2.4 define o que é uma ontologia, suas características, formas de uso e definição de termos.

A seção 2.5 define conceitos para recomendar atividades usando banco de dados de *workflows*. As duas seções seguintes (2.6 e 2.7) descrevem os classificadores e regressores, utilizados neste trabalho [1] para recomendar atividades. A última seção (2.8) descreve o classificador composto e o *ensemble* usados.

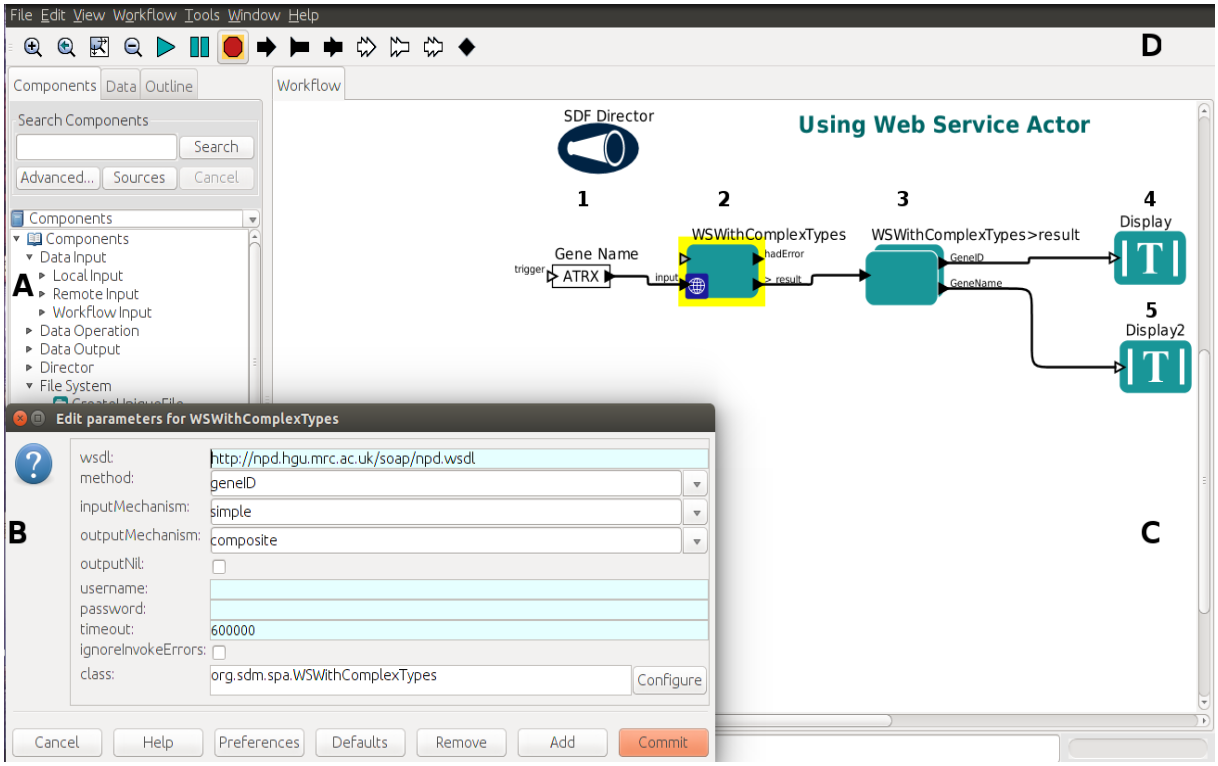
2.1 Sistemas gerenciadores de *workflows* científicos

Sistemas gerenciadores de *workflows* científicos (SGWCs) são infraestruturas de *software* que permitem a construção, reutilização e captura de proveniência de experimentos científicos representados na forma de *workflows* (MCPHILLIPS et al., 2009). Isso possibilita modelar e executar soluções computacionais para problemas científicos, combinando dados e operações sobre dados em estruturas configuráveis formadas por atividades (GARIJO et al., 2014).

As funcionalidades típicas de um SGWC (neste exemplo, o Kepler (KEPLER, 2014)) são apresentadas em quatro regiões da figura 1. A primeira região (**A**) oferece uma lista de atividades para serem usadas na construção do *workflow*. A segunda região (**B**) apresenta os parâmetros de entrada/saída da atividade **2**. A terceira região (**C**) permite arrastar atividades e conectá-las, de forma a permitir a criação do experimento computacional. A última região (**D**) apresenta os conectores das atividades (setas pretas e brancas), controle de execução do *workflows* (seta azul e sinal de pausa) e as ferramentas de ampliar ou reduzir a área (lupas), as quais estão localizadas na região (**C**).

[1] Neste trabalho foram utilizadas as implementações dos classificadores e regressores disponíveis na plataforma R - https://www.r-project.org/

Figura 1 – Sistema gerenciador de *workflows* científicos Kepler



Fonte: Adaptado de Kepler (KEPLER, 2014)

Dentro da área **C** há um *workflow*, que representa um experimento computacional, o qual recebe o nome de um gene e retorna duas informações: i) nome do gene; e ii) identificador na base de dados. Para compreender o processo de construção deste experimento, primeiro serão descritas as funcionalidades de cada atividade e, em seguida, como funciona o fluxo de dados entre elas.

O objetivo das atividades é efetuar algum tipo de processamento sobre os dados (ou usá-los para pesquisa), como os métodos/funções das linguagens de programação. Na figura 1, a atividade número **1** recebe o nome do gene que será pesquisado. A número **2** é o cliente do serviço web e recebe como parâmetros: i) o *WSDL* do serviço; ii) o nome do método a ser utilizado; iii) o tipo de retorno do serviço; e iv) as entradas e saídas do método. A número **3** separa as saídas obtidas da pesquisa e envia-as para as atividades **4** e **5**, responsáveis por exibir os resultados na tela.

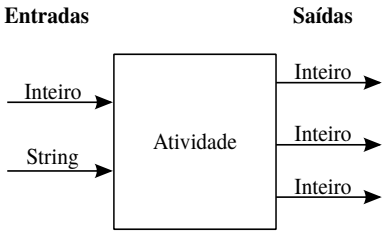
O fluxo de dados entre atividades pode ocorrer de várias formas, conhecidas por *Model of Computation* (MoC) (LUDASCHER et al., 2006). *Workflows* científicos em bioinformática tipicamente usam duas formas de MoC. *Dataflow*, que realiza transformações sobre os dados, analisa/visualiza-os e elabora simulações; e *control flow* mais utilizado

em *workflows* de negócio, enfatiza eventos, fluxogramas e sequências de atividades. Os *workflows* de bioinformática usam esses dois MoC de forma híbrida nos experimentos.

2.1.1 Construção de *workflows* científicos

A construção de *workflows* científicos ocorre pela conexão de diversas atividades, como exibido na figura 1, as quais podem ser rotuladas de acordo com sua estrutura em três classes de atividades. A primeira é conhecida por *atividade simples* que contém apenas uma atividade (GARIJO et al., 2014) como na figura 2, a qual exibe as entradas da atividade (um inteiro e uma *string*) no lado esquerdo e suas saídas no lado direito (três inteiros).

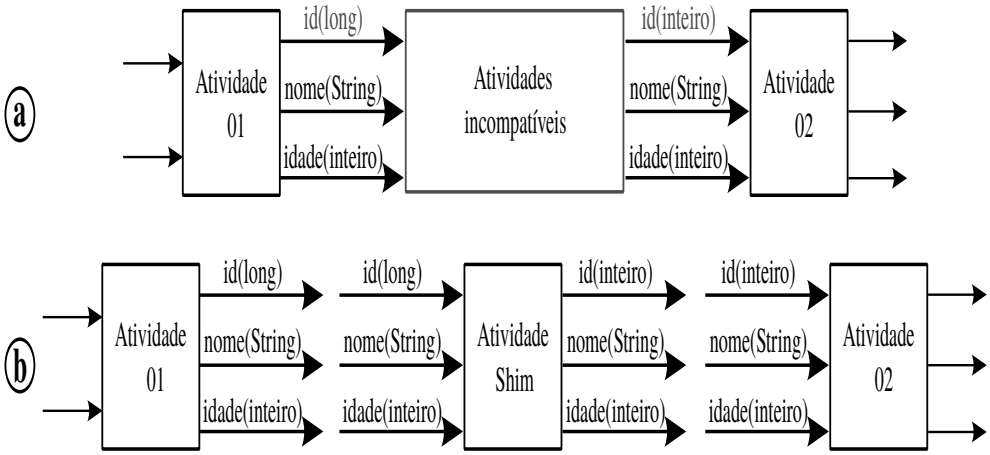
Figura 2 – Exemplo de atividade do tipo *simples*



Fonte: Adilson Lopes Khouri, 2016

O segundo tipo de atividade é denominado *Shim*, tem por objetivo adaptar diferentes tipos de dados (como na figura 3*a*). A *atividade* 1, da figura 3*a*, tem como saída um *id* do tipo *long* e a *atividade* 2, tem como entrada um *id* do tipo inteiro, como consequência ambas são incompatíveis. As atividades do tipo *Shim* efetuam conversões de tipos de dados permitindo a conexão entre atividades incompatíveis como na figura 3*b* (LIN et al., 2009).

Figura 3 – Exemplo de atividade do tipo *Shim*

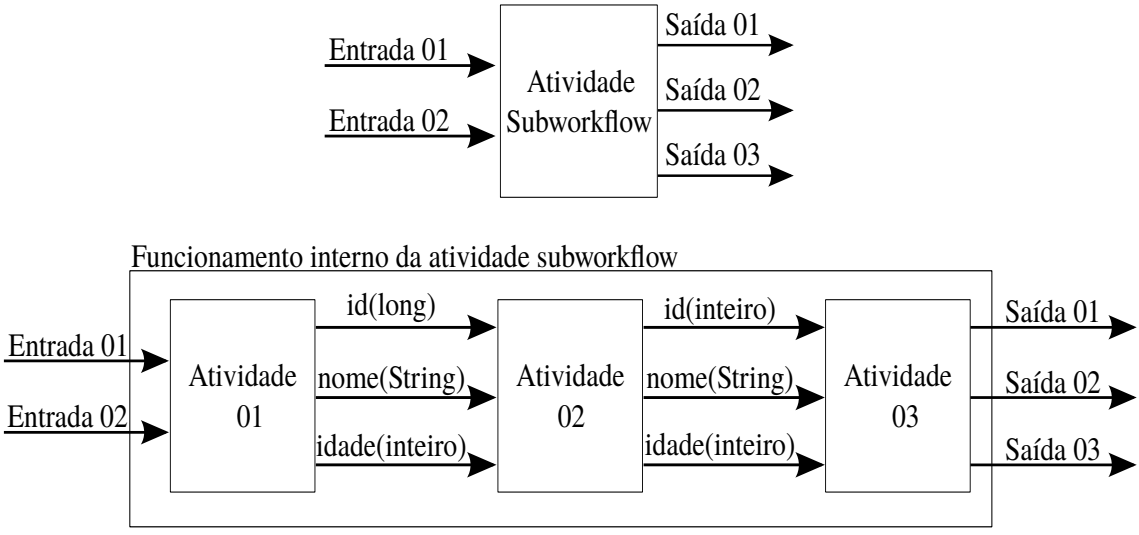


Fonte: Adilson Lopes Khouri, 2016

O terceiro tipo é denominado *subworkflow*, que é uma atividade constituída por um *workflow* interno (conjunto de outras atividades), como exibido na figura 4 (MEDEIROS et al., 2005). Nesse caso, o processamento da atividade é efetuado por um *workflow* completo, cujas entradas e saídas devem conter no mínimo as entradas e saídas do *workflow* interno. Há vários tipos de processamentos internos de atividades, os quais variam de acordo com a estrutura do *workflow*. As mais comuns executam um serviço *web* ou código fonte interno, porém no caso de *subworkflows* é comum usar um processamento misto.

Durante a construção ou execução dos *workflows* (inserção de atividades ou conexão entre as diferentes atividades) alguns SGWCs armazenam as informações relativas à modelagem e execução. Essas informações são conhecidas como proveniência. Segundo Lim et al. (2010), há dois tipos de proveniência: i) *prospective provenance* que modela a especificação de um *workflow*, funcionando como uma abstração/receita do mesmo e que pode ser capturada durante a construção; e ii) *retrospective provenance* que representa as execuções dos *workflows*, quais tarefas foram executadas e quais transformações sobre os dados ocorreram. Esse tipo de proveniência pode ser capturado durante a execução do *workflow*.

Figura 4 – Exemplo de atividade do tipo *subworkflow*



Fonte: Adilson Lopes Khouri, 2016

Para auxiliar na construção de *workflows* foram propostas técnicas para compor automaticamente ou recomendar atividades. Na primeira, o usuário passa para o sistema um problema a ser resolvido e o SGWC seleciona e conecta automaticamente as atividades construindo *workflows* para o usuário. Essa técnica é adequada a usuários que não conhecem

detalhes específicos do processo e/ou não desejam se envolver nas especificidades de como o *workflow* irá resolver o problema.

A segunda técnica ocorre durante a construção manual dos *workflows* e o sistema gerenciador sugere ao usuário algumas atividades que podem ser úteis para o *workflow* em construção. A técnica de recomendação é indicada para usuários mais experientes que desejam ter participação ativa na construção do *workflow*. Há três possibilidades muito conhecidas para recomendar atividades. A primeira é baseada em medidas de similaridade e/ou compatibilidade (ver seção 2.2). A segunda busca atividades em *workflows* ‘‘parecidos’’ com o que está sendo desenvolvido. A terceira busca atividades usadas por usuários com perfil semelhante ao do usuário atual.

2.2 Sistemas de recomendação

Sistemas de recomendação têm como objetivo recomendar itens que sejam interessantes aos usuários, formalizando: seja C o conjunto de todos os usuários, S o conjunto de todos os itens que podem ser recomendados, u a função de utilidade que metrifica o quanto um item s é útil para um determinado usuário c, dada por

$$u : C \times S \rightarrow \mathbb{R} \tag{2.2.1}$$

para cada usuário $c \in C$ se deseja escolher $s^{'} \in S$ que maximize a função de utilidade

$$\forall c \in C, \quad s_c^{'} = \arg\max_{s \in S} u(c, s) \tag{2.2.2}$$

Em sistemas de recomendação, a função utilidade u não está definida para todo o espaço $C \times S$, isso obriga os sistemas de recomendação a extrapolar o espaço conhecido (ADOMAVICIUS; TUZHILIN, 2005). Em outras palavras, recomendando itens não que não foram selecionados como *úteis* pelos usuários mas que serão considerados úteis.

Para solucionar esse problema foram propostas diferentes técnicas para recomendar itens, as quais foram organizadas por Paiva, Costa e Silva (2013) em seis classes. A primeira, chamada *filtro baseado em conteúdo*, recomenda itens similares a outros selecionados anteriormente pelo próprio usuário, suas limitações são: i) análise limitada do conteúdo do item que será recomendado, geralmente há falta de descrição semântica do item; ii) superespecialização: o usuário recebe recomendações similares demais às suas escolhas prévias; e iii) novos usuários precisam avaliar um número mínimo de itens antes que o sistema possa recomendar itens para eles.

A segunda, denominada *filtro colaborativo*, recomenda ao usuário itens que já foram selecionados por outros usuários ‘‘similares’’ a ele, possuindo como limitações: i) o problema de novos usuários (como identificar com quem eles são similares?); ii) novos itens somente serão recomendados ao passo que forem sendo avaliados por usuários; iii) dados esparsos, alguns poucos usuários costumam avaliar muitos itens e a maioria avalia poucos itens tornando a matriz de utilidade (usuários $\times$ itens) esparsa, pois o número de avaliações feitas tende a ser muito menor do que o número de sugestões a serem realizadas. Dessa forma, itens raros (que foram avaliados por poucos) dificilmente serão recomendados.

A terceira, denominada *estratégia híbrida*, combina características das técnicas existentes tentando minimizar suas limitações.

Na quarta, conhecida por *filtro baseado na comunidade*, a recomendação é realizada de acordo com a preferência dos colegas ou amigos do usuário ao invés de preferências de desconhecidos. Trata-se de um tipo de especialização do *filtro colaborativo* herdando suas características. Tipicamente, é baseada em informações da rede social (comunidade) do usuário.

A quinta, denominada *demográfica*, utiliza atributos como região, idade e idioma para recomendar. Surgiu para tentar minimizar o problema de esparsidade e é uma especialização do filtro colaborativo considerando que usuários com dados demográficos parecidos podem ser considerados similares.

A sexta, conhecida por *baseada em conhecimento*, recomenda itens de acordo com o domínio de aplicação. A função de similaridade estima quanto a descrição do problema é similar a solução recomendada. Tem como limitação a necessidade de descrições semânticas (usando, por exemplo, ontologias) sobre o domínio, usuário e o problema.

As técnicas de recomendação citadas necessitam de uma métrica de similaridade para determinar o quanto dois itens/usuários/conceitos são próximos. Zhong, Jinsha e Hong (2009) definem uma função de similaridade genérica s na qual o valor $s(x, x^{'})$ é alto quando x e $x^{'}$ são exemplos parecidos/similares/próximos, caso contrário, possuem um valor baixo. Essa função, de uma forma geral, tem seus valores normalizados entre $[0, 1]$.

De acordo com Zhong, Jinsha e Hong (2009), distâncias são medidas de dissimilaridade, um caso invertido de similaridade, no qual $s(x, x^{'})$ é baixo quando x e $x^{'}$ são

exemplos parecidos ou similares, caso contrário, possuem um valor alto. Uma métrica de distância muito utilizada é a distância Euclidiana (DEZA; DEZA, 2009), calculada por

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.2.3}$$

Outra métrica conhecida é a distância de Manhattan (DEZA; DEZA, 2009), calculada por

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}|x_i - y_i| \tag{2.2.4}$$

Uma medida de similaridade muito conhecida é a similaridade da correlação de Pearson (DEZA; DEZA, 2009), calculada por

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\sum_{j=1}^{n}(x_j - \bar{x})\right)^2\left(\sum_{j=1}^{n}(y_j - \bar{y})\right)^2}} \tag{2.2.5}$$

em que $\bar{y}$ e $\bar{x}$ representam a média dos dados (y e x). Uma distância conhecida para comparar *strings* é a distância de Levenshtein (DEZA; DEZA, 2009), calculada por

$$d_L(\mathbf{x}, \mathbf{y}) = min\{d_H(x^*, y^*)\} \tag{2.2.6}$$

em que x^*, y^* são *strings*, d_H é a função que retorna as possibilidades de alteração de *strings* que resultem em duas strings idênticas. A equação (2.2.6) representa o número mínimo de alterações necessárias para tornar duas *strings* iguais.

2.3 Recomendação em *workflows* científicos

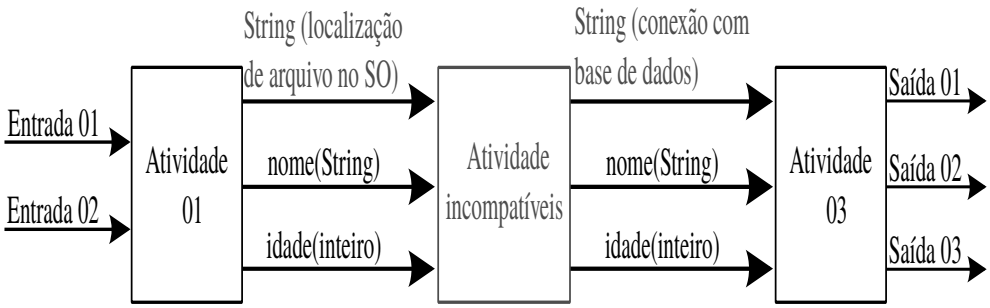
A construção de um sistema de recomendação para *workflows* científicos leva em consideração dois aspectos principais. O primeiro, assim como em qualquer sistema de recomendação, é recomendar itens que satisfaçam ao usuário (como descrito na equação (2.2.2)). Já o segundo aspecto está relacionado à resolução de problemas específicos da recomendação de atividades para *workflows* científicos que são: i) restrições de dependência entre entrada e saída de atividades; ii) dependência semântica entre atividades; e iii) ordem das atividades.

A dependência entre entrada e saída de atividades implica que o tipo de dado (inteiro, *string*, *boolean*) das saídas da atividade recém adicionada no *workflow* (anterior à

recomendação) devem ser compatíveis com os tipos de dados das entradas da atividade a ser recomendada, que virá a seguir na ordem de execução. Suponha que a atividade A tem como saída dois inteiros e uma *string*. Dessa forma, qualquer outra atividade B a ser recomendada para completar o *workflow* que contém A deve ter como entrada dados compatíveis com estes três tipos (ou com um subconjunto deles), ou será necessária a utilização de uma atividade do tipo *Shim*.

Conectar duas atividades por meio de compatibilidade de entrada e saída ou indiretamente, com uso de atividades *Shim* (como na figura 3) não garante que o *workflow* execute ou que o problema do usuário seja solucionado. Isso ocorre em função da possível incompatibilidade semântica entre atividades como na figura 5, na qual as duas atividades são compatíveis sintaticamente (as entradas e saídas das atividades têm os mesmos tipos de dados), porém enquanto a *string* de saída da atividade 01 representa o caminho de um arquivo do sistema operacional, a *string* de saída da atividade 03 representa uma conexão com um banco de dados.

Figura 5 – Exemplo de incompatibilidade semântica entre atividades



Fonte: Adilson Lopes Khouri, 2016

Além das dependências é necessário conectar as atividades na ordem correta. Ao contrário de sistemas de recomendação de músicas, os quais podem recomendar itens em qualquer ordem sem afetar o resultado final da recomendação, nessa área a ordem das atividades é relevante. Por exemplo, dadas duas atividades: uma que consulte um banco de dados e outra que atualize a informação consultada pela primeira, a ordem de execução dessas atividades trará diferentes resultados.

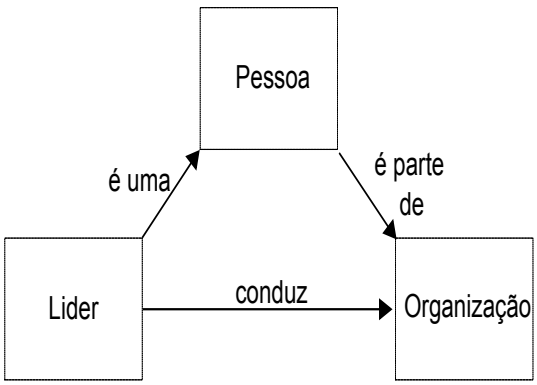
Essas características motivam a criação de técnicas específicas para recomendar atividades em *workflows* científicos, como as citadas no capítulo 3.

2.4 Ontologias

Ontologia é um modelo de representação de conhecimento utilizado para descrever conceitos de um domínio e suas relações, provendo um vocabulário compartilhado para eles. Os conceitos são os substantivos de um determinado domínio de conhecimento e as relações representam as possíveis interações/hierarquias entre esses conceitos (UMAMAHESWARI; PATIL, 2012).

Os elementos básicos, mas não obrigatórios, de uma ontologia são quatro (ALMEIDA; BAX, 2003) e são exemplificados com o auxílio da figura 6. O primeiro são os conceitos, que são organizados em uma taxonomia e representam a conceitualização de um domínio, representados pelos retângulos. O segundo são as relações, representadas pelas setas, são as interações entre os conceitos de um determinado domínio. O terceiro são os axiomas que são usados para modelar sentenças. E o último são as instâncias, que representam os elementos específicos, em outras palavras os próprios dados (ALMEIDA; BAX, 2003).

Figura 6 – Exemplo de ontologia



Fonte: Adaptado de Umamaheswari e Patil (2012)

Há várias propostas na literatura sobre como classificar ontologias. A tabela 1 apresenta um sumário dessas propostas, organizado por Almeida e Bax (2003). Os autores identificaram cinco critérios para classificação de ontologias. O primeiro é baseado em sua função; o segundo considera o grau de formalismo da linguagem usada; o terceiro é em relação a sua aplicação; o quarto é em relação a estrutura (mais genérica ou específica); e o último é em relação ao conteúdo da ontologia.

Para utilizar ontologias em sistemas de recomendação é necessário definir uma métrica de similaridade entre ontologias ou entre conceitos de uma mesma ontologia. Segundo Sanchez et al. (2012), essa métrica pode ser calculada por três principais abordagens:

Tabela 1 – Possíveis classificações de ontologias

Abordagem	Classificação	Descrição
Classificação por *Função* proposta por Mizoguchi, Welkenhuysen e Ikeda (1995)	Domínio	Fornecem vocabulário sobre conceitos e seus relacionamentos.
	Tarefa	Provê vocabulário sistematizado para diferentes domínios.
	Gerais	Vocabulário relacionado a eventos, espaço, casualidade.
Classificação por *Grau de formalismo* proposta por Uschold e Gruninger (1996)	Informal	Linguagem natural
	Seminformal	Linguagem natural restrita.
	Semiformal	Linguagem artificial definida formalmente.
	Formal	Semântica formal, teoremas e provas.
Classificação por *Aplicação* proposta por Jasper e Uschold (1999)	Autoria neutra	Aplicativo escrito para um sistema e usado em outros.
	Especificação	Criada para a manutenção e desenvolvimento de sistemas.
	Acesso comum	Torna o vocabulário inteligível quando este é ininteligível.
Classificação por *Estrutura* proposta por Haav e Lubi (2001)	Alto nível	Descrevem conceitos gerais não relacionados ao domínio.
	Domínio	Descrevem conceitos gerais relacionados ao domínio.
	Tarefa	Descrevem atividades com escopo inferior aos conceitos de domínio.
Classificação por *Conteúdo* proposta por Heijst, Schreiber e Wielinga" (1997)	Terminológica	especificam termos usados para representar um domínio.
	Informação	Estruturas de registros de bases de dados.
	Modelagem de conhecimento	Especificam conceitualização do conhecimento.
	Aplicação	Definições para modelar o conhecimento uma aplicação
	Domínio	Conceitualizações específicas de um domínio.
	Genérica	Conceitualizações genéricas usadas em vários domínios.
	Representação	Conceitualismo usados para representação do conhecimento.

Fonte: Adaptado de Almeida e Bax (2003)

i) contagem por arestas; ii) características da ontologia; e iii) conteúdo da informação. A contagem de arestas utiliza a representação em grafos de ontologias em que cada conceito é representado por um nó e as relações entre os conceitos (por exemplo, relações taxonômicas) são representadas por arestas. Essa abordagem calcula o caminho mínimo de arestas entre dois conceitos. Suas principais limitações são: i) não considerar outros possíveis caminhos ou medidas de similaridade (entre conceitos); e ii) utilizar o mesmo peso para todas as arestas.

A abordagem baseada em características das ontologias utiliza a diferença entre as propriedades em comum dos conceitos e as propriedades para calcular a similaridade. As propriedades citadas são, por exemplo, os sinônimos ou definições textuais que os conceitos contenham (mas não é mandatório que os conceitos das ontologias contenham

essas propriedades). As limitações dessa abordagem são exigir que a ontologia tenha propriedades além da taxonomia e a dificuldade de identificar os pesos corretos a serem atribuídos para cada propriedade (SANCHEZ et al., 2012).

A abordagem baseada em conteúdo da informação presente nas ontologias calcula a similaridade baseada na coocorrência de termos entre duas ontologias. Por exemplo, utilizando a equação

$$IC(x) = -\log_2 P(x) \tag{2.4.1}$$

que calcula o $\log_2$ da probabilidade de ocorrência $P(x)$ e multiplica-o por -1. Sua limitação é que necessita de uma grande e refinada estrutura ontológica/taxonômica para conseguir discriminar diferentes conceitos (SANCHEZ et al., 2012).

2.5 Recomendação a partir de banco de *workflows*

Uma estratégia para se recomendar atividades em um *workflow* é a baseada nas suas atividades existentes. Neste tipo de estratégia, com base nas atividades existentes no *workflow* atual (em construção) procura-se por atividades que foram úteis a *workflows* finalizados. Para isso, algum critério de proximidade entre *workflows* ou entre atividades costuma ser utilizado. A seguir algumas abordagens que usam banco de dados de *workflows* são apresentadas.

2.5.1 Recomendação a partir da última atividade

Uma maneira de se recomendar atividades com base num banco de *workflows* é sugerir ao usuário as atividades que costumam ser mais frequentemente utilizadas após a última atividade inserida pelo usuário. Essa estratégia assume que a sequência de atividades costuma ser conservada entre diferentes *workflows*.

2.5.2 Recomendação a partir de *itemsets*

Uma das maneiras de recomendar itens é por meio da identificação de *itemsets* frequentes (Conjuntos de itens que ocorrem com frequência no conjunto de dados (HAN;

KAMBER; PEI, 2011)). Um dos algoritmos mais famosos para este tipo de recomendação é o algoritmo *Apriori* (AGRAWAL; SRIKANT, 1994).

Algumas definições essenciais para compreender o algoritmo *Apriori* são; i) *itemsets*; ii) *k-itemset*; iii) regra de associação; iv) suporte; iv) confiança; e v) propriedade *Apriori*. Cada conjunto de itens é denominado *itemset* (HAN; KAMBER; PEI, 2011). Um *itemset* com k elementos é chamado de *k-itemset*. Uma regra de associação é uma relação entre *itemsets* e tem o formato $A \rightarrow B$. Cada regra de associação tem um suporte dado por

$$sup(A \rightarrow B) = P(A \cup B) \tag{2.5.1}$$

e uma confiança calculada como

$$conf(A \rightarrow B) = \frac{sup(A \rightarrow B)}{sup(A)} \tag{2.5.2}$$

Uma regra de associação r é considerada interessante caso $sup(r) > \alpha$ e $conf(r) > \beta$. A propriedade *Apriori* diz que para um *itemset* ser considerado frequente todos os *itemsets* contidos nele também devem ser frequentes (HAN; KAMBER; PEI, 2011). Para determinar boas regras de associação deve-se estabelecer dois limiares α e β para o suporte e a confiança da regra, respectivamente.

O Apriori pesquisa por *itemsets* frequentes, que são aqueles que apresentem um suporte mínimo, em uma base de transações usando uma estratégia de três fases iterativas.

A primeira é a geração de *itemsets* candidatos de tamanho k usando os *itemsets* de tamanho $k-1$, de acordo com a propriedade *Apriori* para aumentar as chances de gerar *itemsets* frequentes devem ser criados apenas *itemsets* que contenham $k-2$ itens em comum. A segunda fase é a poda, na qual são pesquisados *itemsets* que contenham *subitemsets* não frequentes, os quais são removidos do conjunto de *itemsets* candidatos. A última fase é o cálculo de suporte, na qual cada candidato tem seu suporte calculado varrendo-se a base de dados uma vez, removendo os que contenham suporte menor que o mínimo.

Um caso especial desse algoritmo é a criação do *itemset* para $k = 1$, aqueles que contêm apenas um item no conjunto. Dessa forma, não podem ser criados a partir de um conjunto de tamanho inferior como no caso geral descrito anteriormente. Nesse caso são criados todos os possíveis conjuntos unitários. Em seguida, a base de dados é usada para calcular seus suportes. Por fim, são removidos os que não contêm suporte mínimo (HAN; KAMBER; PEI, 2011).

Tabela 2 – Bases de dados de exemplo

id	Produto
1	Pão
2	Leite
3	Açúcar
4	Papel Higiênico
5	Manteiga
6	Fralda
7	Cerveja
8	Refrigerante
9	Iogurte
10	Suco

(a) Produtos da base de dados

TID	Itens comprados
101	{1, 3, 5}
102	{2, 1, 3, 7, 5}
103	{4, 9, 2, 1}
104	{5, 2, 1, 3, 9}
105	{1, 8, 6, 4, 3, 5}
106	{9, 2, 8}

(b) Compras efetuadas no supermercado

Fonte: Adaptado de Han, Kamber e Pei (2011)

2.6 Recomendação por classificação

Segundo Han, Kamber e Pei (2011) classificar itens computacionalmente é construir um modelo que permita predizer suas classes discretas. Este modelo é construído em duas etapas: a primeira é o treinamento, no qual o classificador está aprendendo com um conjunto de itens rotulados; a segunda é a fase de testes. Nela, o classificador utiliza as regras aprendidas na fase anterior para classificar dados sem rótulo.

2.6.1 Classificação por vizinhos mais próximos

O algoritmo *K-nearest neighbors* (KNN) (HAN; KAMBER; PEI, 2011), usa cada instância dos dados de teste para localizar seus k vizinhos mais próximos (de acordo com alguma métrica de similaridade ou distância) e atribui para essa instância à classe que contém o maior número de vizinhos (HAN; KAMBER; PEI, 2011). Para exemplificar o funcionamento são utilizados os dados da tabela 3 como conjunto de treinamento.

Tabela 3 – Exemplo de dados de treino para o KNN

alimento	doçura	crocância	rótulo
uva	8	5	fruta
feijão	3	7	vegetal
castanha	3	6	proteína
laranja	7	3	fruta
tomate	6	4	fruta

Fonte: Adaptado de Lantz (2013)

Considerando que os dados da tabela 3 foram utilizados para treinamento e que o algoritmo recebe uma nova instância (o *tomate*) para ser classificada e que ela possui os atributos: i) doçura = 6; e ii) crocância = 4). O algoritmo calcula as distâncias entre a instância *tomate* e todos os dados de treinamento, como mostra a figura 7, obtendo

$$d(\text{uva}, \text{tomate}) = \sqrt{(6-8)^2 + (4-5)^2} = 2,2 \tag{2.6.1}$$

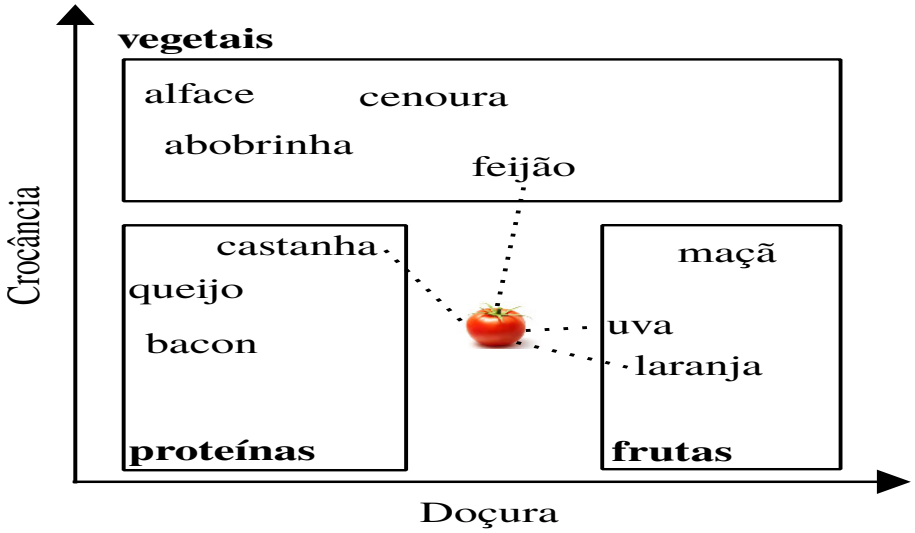
$$d(\text{feijão}, \text{tomate}) = \sqrt{(6-3)^2 + (4-7)^2} = 4,2 \tag{2.6.2}$$

$$d(\text{castanha}, \text{tomate}) = \sqrt{(6-3)^2 + (4-6)^2} = 3,6 \tag{2.6.3}$$

$$d(\text{laranja}, \text{tomate}) = \sqrt{(6-7)^2 + (4-3)^2} = 1,4 \tag{2.6.4}$$

Determinar a classe da instância *tomate* depende do número k de vizinhos escolhidos pelo usuário e pelas distâncias calculadas. Utilizando $k = 1$, a instância seria classificada como fruta, pois o único vizinho (laranja) pertence a este rótulo. Usando $k = 3$, a instância terá três vizinhos: i) laranja; ii) uva; e iii) feijão; totalizando duas frutas e um vegetal consequentemente o tomate seria classificado como fruta novamente.

Figura 7 – Exemplo de classificação usando o algoritmo KNN



Fonte: Adaptado de Lantz (2013)

A escolha de um número muito alto k de vizinhos usados para classificar causa uma redução no impacto de ruídos, porém pode enviesar o classificador tornando-o incapaz de identificar alguns padrões significativos. Em contrapartida, utilizar um valor muito baixo, como $k = 1$, pode tornar o algoritmo suscetível a ruídos.

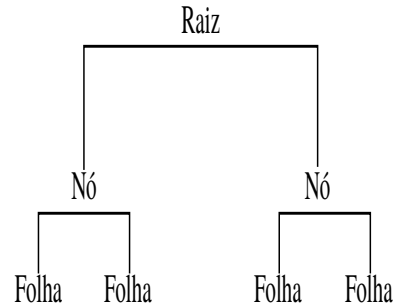
Uma possível abordagem para encontrar um bom valor de k é testar o conjunto de dados para diversos valores e comparar as diferentes desempenhos de classificação.

2.6.2 Árvores de classificação e regressão - classificador e regressor

Técnicas baseadas em árvores de decisão usam uma estrutura em árvore para tomar decisões. Esta estrutura é constituída por um nó *raiz*, diversos nós de *decisão* e os nós *folha*, como ilustrado na figura 8.

O aprendizado das árvores é feito utilizando o conjunto de treinamento, no qual é selecionado o atributo que maximiza algum critério de divisão (ver equações (2.6.7) e (2.6.10)) responsável por particionar o conjunto. Este atributo pode se tornar um nó de *decisão*, no caso das instâncias da partição criada por esta divisão contenham rótulos diferentes, ou um nó *folha*, no caso das instâncias da partição criada por esta divisão contenham o mesmo rótulo (HAN; KAMBER; PEI, 2011).

Figura 8 – Estrutura de árvore de decisão



Fonte: Adaptado de Williams (2011)

O algoritmo CART, proposto por Breiman et al. (1984) utiliza uma abordagem *top-down* que constrói a árvore recursivamente selecionando o melhor atributo baseado no índice de *Gini* (ver equação (2.6.10)) a cada iteração. O crescimento da árvore termina quando não há mais possibilidade de crescimento, o próximo passo é a poda que constrói k árvores de decisão nas quais T_1 é idêntica a T_0 com exceção de um ramo que se transformou em folha e T_k é apenas uma folha. É selecionada a melhor árvore com base no menor erro de classificação (BREIMAN et al., 1984).

Alguns critérios de divisão muito utilizados para classificação em árvores de decisão são: i) entropia (conceito da área da Física); e ii) o ganho de informação (conceito que deriva da teoria da informação). A entropia, mede o grau de incerteza em um conjunto, por exemplo, um conjunto de dados com apenas um valor observado de classe tem entropia igual a zero. Outro conjunto, em que metade dos valores residem em uma classe e a outra

metade em outra terá entropia máxima. O cálculo de entropia (HAN; KAMBER; PEI, 2011) é descrito pelas seguintes equações

$$Entropy(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{2.6.5}$$

$$Info_A(D) = \sum_{i=1}^{m} \left(\frac{D_i}{D}\right) Entropy(D_i) \tag{2.6.6}$$

$$Ganho(A) = Info(D) - Info_A(D) \tag{2.6.7}$$

sendo p_i a probabilidade da classe C_i no conjunto de dados D, m é o número de classes, D_i é um subconjunto de D e a equação (2.6.7) calcula o ganho de informação utilizando o atributo A. Esse cálculo é aplicado em todos os atributos e é escolhido o que obtiver maior ganho de informação.

Outro critério que pode ser adotado para a divisão é o índice de *Gini* (HAN; KAMBER; PEI, 2011), que mede a impureza de um conjunto de dados. Representado pelo número de instâncias encontradas na partição que não pertencem a mesma classe, calculada por

$$Gini(D) = 1 - \sum_{i=1}^{m} (p_i)^2 \tag{2.6.8}$$

$$Gini_A(D) = \sum_{i=1}^{m} \left(\frac{D_i}{D}\right) Gini(D_i) \tag{2.6.9}$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \tag{2.6.10}$$

O atributo escolhido para teste lógico é aquele que maximiza o $\Delta Gini(A)$. Em outras palavras, reduz a impureza do conjunto. Esses critérios são usados para particionar os conjuntos de dados de forma recorrente até que se atinja um critério de parada, determinado pelo algoritmo.

Para tarefas de regressão, o algoritmo tem algumas diferenças. A primeira é usar como métrica de impureza o quadrado da soma residual (SRQ), que, segundo Connor (2007), é calculada por

$$\underset{x_j \leq x_j^R}{\arg\min} \qquad [P_l(Var(Y_l)) + P_r(Var(Y_r))] \tag{2.6.11}$$

cujo objetivo é minimizar a soma da probabilidade da variância ($P(Var(Y))$) dos nós filhos (Y_l nó filho da esquerda e Y_r nó filho da direita) gerados após uma divisão. Outra diferença é que cada nó *folha* será preenchido pela média dos valores das instâncias de treinamento

atribuídos na folha. Por fim, supõe-se uma relação linear entre a variável resposta e os preditores. Outros aspectos como construção, poda e treinamento são idênticos a árvores de classificação (CONNOR, 2007).

2.6.3 Naive Bayes

Classificadores baseados em técnicas Bayesianas utilizam os dados de treinamento para calcular a probabilidade observada de cada classe de acordo com suas características. Esses classificadores são melhor aplicados em problemas nos quais a informação de vários atributos pode ser considerada simultaneamente para gerar uma estimativa de probabilidades de saídas.

O algoritmo *Naive Bayes* utiliza a técnica Bayesiana, que é uma aplicação do teorema de Bayes (LANTZ, 2013) adaptado para classificação, que assume as seguintes condições ingênuas sobre os dados: i) independência de características; e ii) que todas as características são igualmente importantes. No mundo real, essas condições são falhas, mas ainda assim o algoritmo possui um desempenho satisfatório (HAN; KAMBER; PEI, 2011).

Para explicar o funcionamento da classificação por Naive Bayes será usado um exemplo adaptado de Lantz (2013). O objetivo é construir um classificador binário de e-mails com as seguintes classes: i) spam; e ii) não spam. Usando o conjunto de dados de treinamento, cada e-mail é classificado de acordo com a maior probabilidade atribuída pelo classificador (ser spam ou não).

O classificador verifica se o e-mail contém as seguintes palavras *viagra* (W_1), *dinheiro* (W_2), *cancelar assinatura* (W_3) e *comestíveis* (W_4), em seguida, aplica o teorema de Bayes (LANTZ, 2013). Suponha que o classificador deva rotular a mensagem M_1 que contém as palavras: *viagra* e *cancelar assinatura*, para tal, usa-se a seguinte equação

$$P(Spam|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)P(Spam)}{(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)} \tag{2.6.12}$$

Os valores usados para o cálculo das probabilidades são obtidos utilizando o conjunto de treinamento (rotulado). A técnica Naive Bayes aplica o conceito de independência de eventos (2.6.13) na equação (2.6.12)

$$P(A \cap B) = P(A)P(B) \tag{2.6.13}$$

Dessa forma, a probabilidade de um e-mail ser classificado como *spam* é dada por

$$EvSpam = Spam|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4 \tag{2.6.14}$$

$$P(EvSpam) = \left(\frac{P(W_1|Spam)P(\neg W_2|Spam)}{P(W_1)P(\neg W_2)}\right)\left(\frac{P(\neg W_3|Spam)P(W_4|Spam)P(Spam)}{P(\neg W_3)P(W_4)}\right) \tag{2.6.15}$$

aplicando-se a fórmula simplificada novamente, obtem-se a probabilidade do e-mail ser classificado como *não spam*, dada por

$$\overline{EvSpam} = \overline{Spam}|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4 \tag{2.6.16}$$

$$P(\overline{EvSpam}) = \left(\frac{P(W_1|\overline{Spam})P(\neg W_2|\overline{Spam})}{P(W_1)P(\neg W_2)}\right)\left(\frac{P(\neg W_3|\overline{Spam})P(W_4|\overline{Spam})P(\overline{Spam})}{P(\neg W_3)P(W_4)}\right) \tag{2.6.17}$$

Após definir as fórmulas para o cálculo das probabilidades da mensagem, serão utilizados os dados do conjunto de treinamento, exibidos na tabela 4, para calcular suas probabilidades e efetuar a classificação. Usando os dados desta tabela em conjunto com as equações

Tabela 4 – Exemplo do cálculo de verossimilhança

	$\mathbf{W_1}$		$\mathbf{W_2}$		$\mathbf{W_3}$		$\mathbf{W_4}$	
Verossimilhança	Sim	Não	Sim	Não	Sim	Sim	Não	Sim
Spam	$\frac{4}{20}$	$\frac{16}{20}$	$\frac{10}{20}$	$\frac{10}{20}$	$\frac{0}{20}$	$\frac{20}{20}$	$\frac{12}{20}$	$\frac{8}{20}$
Não Spam	$\frac{1}{80}$	$\frac{79}{80}$	$\frac{14}{80}$	$\frac{66}{80}$	$\frac{8}{80}$	$\frac{71}{80}$	$\frac{23}{80}$	$\frac{57}{80}$
Total	$\frac{5}{100}$	$\frac{95}{100}$	$\frac{24}{100}$	$\frac{76}{100}$	$\frac{8}{100}$	$\frac{91}{100}$	$\frac{35}{100}$	$\frac{65}{100}$

Fonte: Adaptado de Lantz (2013)

(2.6.14) até (2.6.17) obtem-se

$$P(EvSpam) = \left(\frac{4}{20}\right)\left(\frac{10}{20}\right)\left(\frac{20}{20}\right)\left(\frac{12}{20}\right)\left(\frac{20}{100}\right) = 0,012 \tag{2.6.18}$$

$$P(\overline{EvSpam}) = \left(\frac{1}{80}\right)\left(\frac{66}{80}\right)\left(\frac{71}{80}\right)\left(\frac{23}{80}\right)\left(\frac{80}{100}\right) = 0,002 \tag{2.6.19}$$

A probabilidade de ser *spam* é igual a verossimilhança de que a mensagem é *spam* dividida pela probabilidade de ser, ou não *spam*, logo obtem-se

$$P(EvSpam) = \frac{0,012}{0,012 + 0,002} = 0,857 \tag{2.6.20}$$

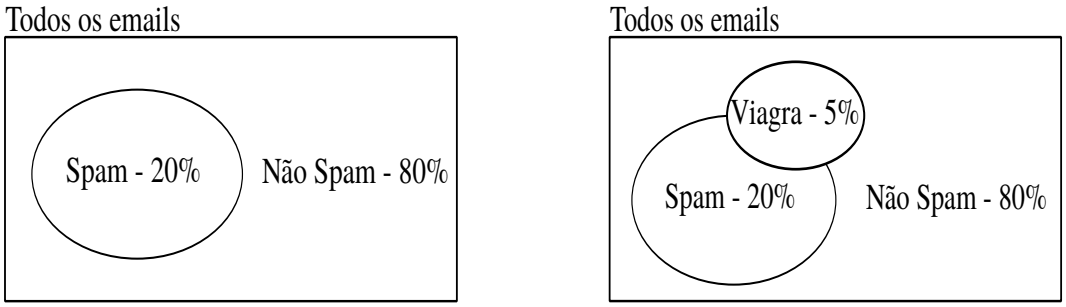
De forma análoga para *não spam* tem-se que

$$P(\overline{EvSpam}) = \frac{0,002}{0,012 + 0,002} = 0,143 \tag{2.6.21}$$

Pode-se observar que a probabilidade da mensagem M_1 ser *spam* é maior que ser *não spam*. Logo será classificada como *spam* pelo algoritmo Naive Bayes.

A teoria da probabilidade Bayesiana é a raiz para o cálculo de verossimilhança de um evento baseado em uma evidência, portanto serão revisados alguns conceitos básicos usados anteriormente. A probabilidade de um evento pode ser obtida pelos dados observados, dividindo o número de ocorrências de um evento pelo número total de dados. Dois eventos são ditos mutuamente exclusivos se eles não ocorrerem ao mesmo tempo. A figura 9a apresenta um exemplo de eventos mutuamente exclusivos. Seja $P(A)$ a probabilidade

Figura 9 – Tipos de eventos probabilísticos



(a) Eventos mutuamente exclusivos (b) Eventos não mutuamente exclusivos

Fonte: Adaptado de Lantz (2013)

de um evento A ocorrer, e dados dois eventos mutuamente exclusivos A e B é válida a seguinte relação

$$P(A) = 1 - P(B) \tag{2.6.22}$$

É comum estudar vários eventos não mutuamente exclusivos para a mesma saída, ou seja, eventos que ocorrem em conjunto. Eles permitem que um evento seja utilizado para prever o outro. A figura 9b apresenta um exemplo de eventos não mutuamente exclusivos.

O evento viagra pode (ou não) ser um *spam*, logo ele é um evento conjunto com o evento *spam*, mas nem todo o e-mail com essa palavra é um *spam*. Nesse caso, para prever um evento com base em outro, é necessário calcular a probabilidade da intersecção dos eventos *viagra* e *spam* $P(Spam \cap Viagra)$ que depende da probabilidade conjunta dos dois eventos (como a probabilidade de um evento está relacionada a outro).

Se todos os eventos forem independentes (não satisfazem a equação (2.6.13)) é impossível prever um evento com dados de outro. No caso da não independência, é válida a relação

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)} \tag{2.6.23}$$

No exemplo da classificação de um e-mail como *spam* ou *não spam*, sem informações adicionais, é possível alegar que há 20% de chance de ser um *spam*, que é a probabilidade *apriori*.

Agora suponha que foram obtidas mais informações e que a mensagem recebida contenha o termo *Viagra*. A probabilidade desse termo ser utilizado em mensagens anteriores é chamada de *verossimilhança* e a probabilidade do termo aparecer em qualquer mensagem é chamado *verossimilhança marginal*.

Aplicando o teorema de Bayes podemos calcular a probabilidade *posteriori*, usando a equação (2.6.24), que mede a probabilidade de uma mensagem ser um *spam*. Se a probabilidade for maior que 50%, o e-mail é mais parecido com *spam*.

$$P(Spam|Viagra) = \frac{\overbrace{P(Viagra|Spam)}^{\text{verossimilhança}} \overbrace{P(Spam)}^{\text{probabilidade apriori}}}{\underbrace{P(Viagra)}_{\text{verossimilhança marginal}}} \tag{2.6.24}$$

Para calcular os componentes da equação (2.6.24) é necessária uma tabela de frequência, a qual armazena o número de ocorrências do termo *Viagra* nos e-mails classificados como *spam* e como *não spam* (calculada utilizando o conjunto de treinamento), esta tabela será utilizada para construir uma tabela de verossimilhança. No exemplo da mensagem

Tabela 5 – Matriz de frequência e de verossimilhança

Frequência	W_1	$\neg W_1$	Total
Spam	4	16	20
Não Spam	1	79	80
Total	5	95	100

(a) Matriz de frequência

Frequência	W_1	$\neg W_1$	Total
Spam	$\frac{4}{20}$	$\frac{16}{20}$	20
Não Spam	$\frac{1}{80}$	$\frac{79}{80}$	80
Total	$\frac{5}{100}$	$\frac{95}{100}$	100

(b) Matriz de verossimilhança

Fonte: Adaptado de Han, Kamber e Pei (2011)

M_1 foi construída a tabela de frequência 5a que permitiu a construção da tabela de verossimilhança 5b. Esta última, em conjunto da equação (2.6.23), permite calcular a probabilidade *posteriori* como

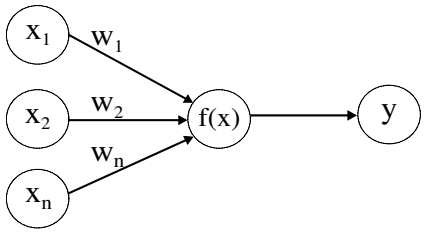
$$\frac{P(Viagra|Spam)P(Spam)}{P(Viagra)} = \frac{\left(\frac{4}{20}\right)\left(\frac{20}{100}\right)}{\left(\frac{1}{80}\right)} = 3,2 \tag{2.6.25}$$

que é uma probabilidade mais precisa, por considerar a probabilidade condicional do termo W_1(viagra), ao invés da probabilidade baseada em independência.

2.6.4 Rede neural - classificador e regressor

Redes neurais artificiais têm como inspiração o cérebro humano, seus neurônios e suas conexões. A computação tenta reproduzir esse modelo de cérebro como uma relação de entradas e saídas ponderadas, que são conjuntos de nós de processamento. Eles são responsáveis por calcular a soma de entradas ponderadas e repassá-las para a função de ativação, responsável por determinar se um sinal será enviado (ou não) para o neurônio seguinte (HAYKIN, 2007).

Figura 10 – Exemplo de neurônio



Fonte: Adaptado de Lantz (2013)

A figura 10 apresenta um neurônio, no qual o vetor $\mathbf{w}$ contém os pesos sinápticos, o vetor $\mathbf{x}$ contém os n sinais de entrada, $f(x)$ é a função de ativação e y, é o sinal de saída que é calculado por

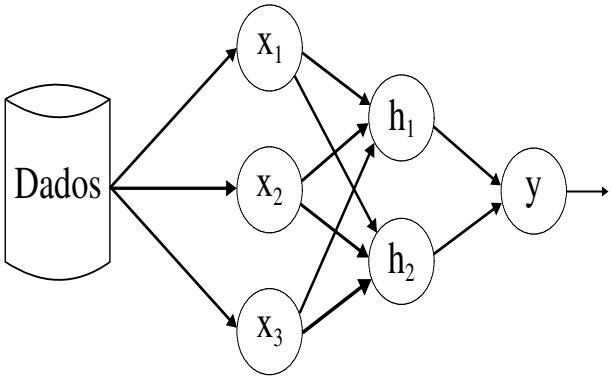
$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{2.6.26}$$

Uma rede neural é um conjunto de neurônios conectados em uma determinada *arquitetura*. Ela consiste no número de neurônios por camada, o tipo de conexão entre eles, o fluxo de informação entre neurônios, a função de ativação e o algoritmo de treinamento utilizado. No restante desta seção são detalhados os itens da arquitetura citados.

Analisando a rede neural da figura 11 pode-se constatar que suas camadas são formadas por: i) três neurônios de entrada (representados pela letra x), que recebem os dados diretamente da fonte; ii) dois neurônios ocultos (representados por h), que recebem dados processados por neurônios e enviam esses dados para outros neurônios; e iii) um neurônio de saída (representado por y) que apresenta a saída da rede.

Neste projeto, serão usadas redes neurais com duas camadas, essa escolha é justificada por Bottou et al. (2007) que demonstram que o aumento do número de camadas não aumenta a capacidade de generalização e/ou precisão das redes neurais.

Figura 11 – Topologia de rede neural



Fonte: Adaptado de Lantz (2013)

O tipo de conexão entre neurônios na figura 11 é denominado *totalmente conectada*, no qual todos os neurônios de uma camada estão conectados com todos os neurônios da camada seguinte. O fluxo de informação é denominado *feedforward* pois os dados trafegam da esquerda para a direita (no sentido das setas pretas) sem retroalimentação.

A função de ativação $f(x)$, exibida na figura 10, determina qual tipo de sinal será enviado para o neurônio seguinte. Lantz (2013) cita as seguintes funções de ativação: a função *degrau* definida por

$$f(x) = \begin{cases} 0 \text{ se } x < 0 \\ 1 \text{ se } x \geq 0 \end{cases} \tag{2.6.27}$$

e a função *sigmoide* definida por

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6.28}$$

A escolha da função de ativação deve considerar possíveis restrições do algoritmo de treinamento, um exemplo de restrição do algoritmo *backpropagation* é que a função deve ser diferenciável em todos os pontos. Apenas a função *sigmoide* (2.6.28) ,das que foram apresentadas nessa seção, satisfaz essa condição.

O algoritmo de treinamento usado neste projeto, denominado *backpropagation*, possui duas abordagens para treinar os dados: i) *online*; e ii) *batch*. Neste projeto é usada a primeira forma, que atualiza os pesos sinápticos, instância por instância, iterativamente após serem apresentadas a rede neural (HAYKIN, 2007). Para explicar o funcionamento deste algoritmo é necessário compreender como calcular o erro de um neurônio j na iteração n, como o da figura 12 que é calculado por

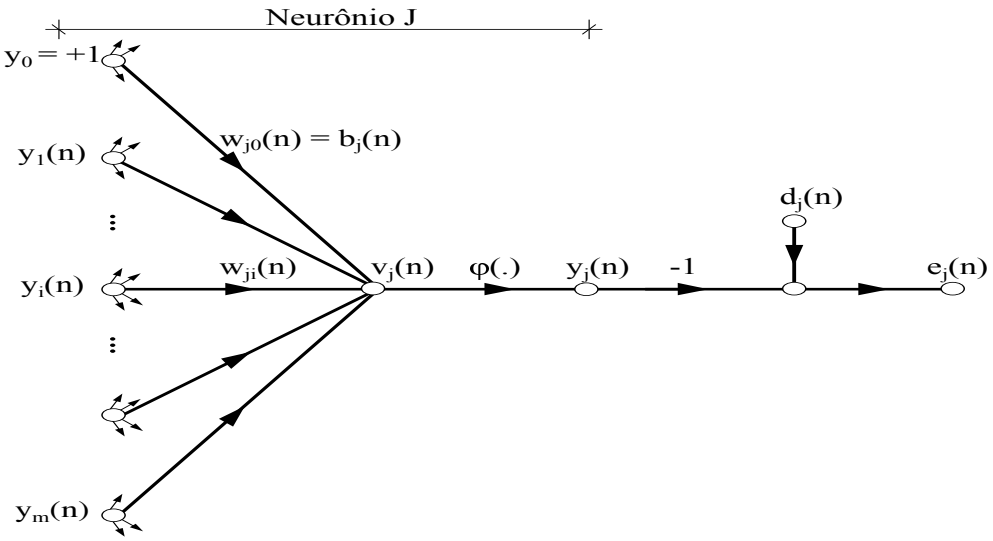
$$e_j(n) = d_j(n) - y_j(n) \tag{2.6.29}$$

sendo $d_j(n)$ o valor de saída esperado e $y_j(n)$ o valor de saída obtido. Além desse erro, é necessário calcular o valor de erro total da camada de saída que é calculado por

$$\xi(n) = \frac{1}{2}\sum_{j=1}^{|C|} e_j^2(n) \tag{2.6.30}$$

sendo $|C|$ o número de neurônios na camada de saída da rede. O objetivo do treinamento é atualizar o vetor $\mathbf{w}$ para minimizar a equação (2.6.30). Essa minimização é realizada pela técnica *descida de gradiente*, a qual começa calculando o vetor de gradiente de cada neurônio. Em seguida, calcula o erro de cada neurônio para as duas possíveis situações: i) para os neurônios de saída (caso simples); e ii) para os neurônios ocultos (caso complexo). Por fim, usa a regra delta (2.6.56) para atualizar o valor $\mathbf{w}$, esta regra tem como requisitos os cálculos: i) do vetor gradiente; e ii) o erro dos neurônios.

Figura 12 – Exemplo de neurônio de saída para cálculo do *backpropagation*



Fonte: Adaptado de Haykin (2007)

A figura 12 exibe um neurônio j sendo alimentado por uma camada de neurônios à esquerda, denominada y_i com $i = 1, 2, 3 \ldots m$. Esse neurônio tem o valor de entrada da função de ativação $v_j(n)$, que é uma soma ponderada entre as entradas e seus pesos, calculada por

$$v_j(n) = \left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) \tag{2.6.31}$$

sendo m o número total de neurônios de entrada. O resultado da equação (2.6.31) é o parâmetro de entrada para a função de ativação, dessa forma é possível escrever

$$y_j(n) = \varphi_j(v_j(n)) \tag{2.6.32}$$

A correção $\Delta w_{ij}(n)$ aplicada nos pesos sinápticos $w_{ij}(n)$ é proporcional a derivada parcial do erro total em relação aos pesos sinápticos $\frac{\partial \xi(n)}{\partial w_{ji}(n)}$ e pode ser escrita, de acordo com a regra da cadeia, como

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \left(\frac{\partial \xi(n)}{\partial e_j(n)}\right)\left(\frac{\partial e_j(n)}{\partial y_j(n)}\right)\left(\frac{\partial y_j(n)}{\partial v_j(n)}\right)\left(\frac{\partial v_j(n)}{\partial w_{ji}(n)}\right) \tag{2.6.33}$$

a derivada parcial da equação (2.6.33) representa um fator de sensitividade que determina em qual direção no espaço de pesos deve-se pesquisar pelas atualizações de $w_{ji}(n)$. Cada uma das quatro derivadas parciais do lado direito da equação (2.6.33) será solucionada para encontrar o gradiente. Diferenciando os dois lados da equação (2.6.30) em relação à $e_j(n)$ é possível reescrevê-la como

$$\frac{\partial \xi(n)}{\partial e_j(n)} = e_j(n) \tag{2.6.34}$$

diferenciando os dois lados da equação (2.6.29) em relação à $y_j(n)$ é possível reescrevê-la como

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.6.35}$$

diferenciando a equação (2.6.32) em relação à $v_j(n)$ é possível reescrevê-la como

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi^{'}(v_j(n)) \tag{2.6.36}$$

diferenciando a equação (2.6.31) em relação à $w_{ji}(n)$ é possível reescrevê-la como

$$\frac{\partial v_j(n)}{\partial w_j(n)} = y_i(n) \tag{2.6.37}$$

substituindo as equações (2.6.34) até (2.6.37) na equação (2.6.33) pode-se reescrever o erro em função dos pesos da rede como

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi^{'}(v_j(n))y_i(n) \tag{2.6.38}$$

a correção $\Delta w_{ji}(n)$ que será aplicada nos pesos $w_{ji}(n)$ é definida pela regra delta como

$$\Delta w_{ji}(n) = -\eta\left(\frac{\partial \xi(n)}{\partial w_{ji}(n)}\right) \tag{2.6.39}$$

sendo η a taxa de aprendizado. Substituindo a equação (2.6.38) na equação (2.6.39) define-se a regra delta como

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n) \tag{2.6.40}$$

na qual o gradiente local $\delta_j(n)$ é definido por

$$\delta_j(n) = \frac{\partial \xi(n)}{\partial v_j(n)} \Leftrightarrow$$

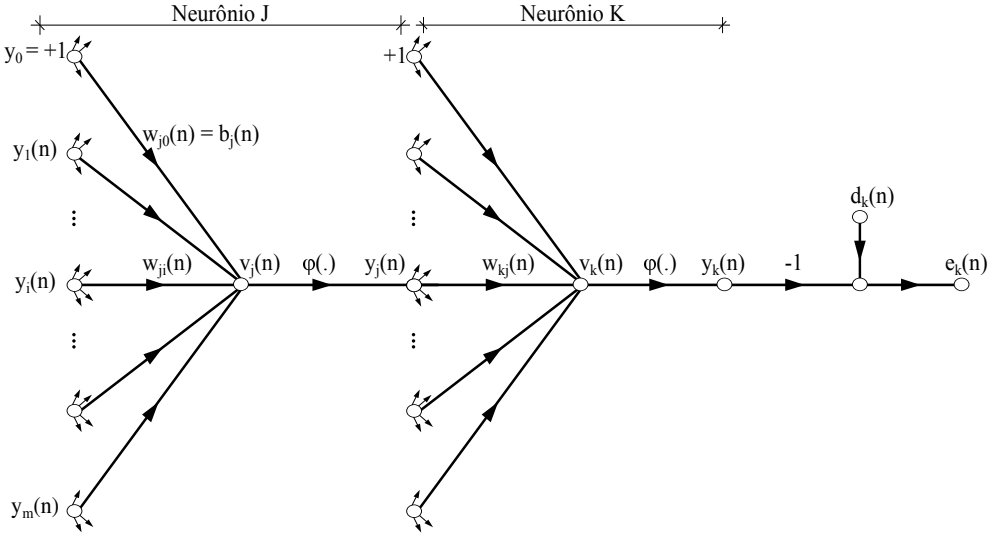
$$\delta_j(n) = \left(\frac{\partial \xi(n)}{\partial e_j(n)}\right)\left(\frac{\partial e_j(n)}{\partial y_j(n)}\right)\left(\frac{\partial y_j(n)}{\partial v_j(n)}\right) \Leftrightarrow \tag{2.6.41}$$

$$\delta_j(n) = e_j(n)\varphi^{'}(v_j(n)) \tag{2.6.42}$$

De acordo com a equação (2.6.42) o gradiente local $\delta_j(n)$ para o neurônio de saída j é o produto do sinal de erro da saída $e_j(n)$ para aquele neurônio e o derivativo $\varphi^{'}(v_j(n))$ da função de ativação (HAYKIN, 2007).

A posição do neurônio na arquitetura da rede influência o cálculo do gradiente local. Dessa forma, apresenta papel fundamental para o cálculo do erro. Se for um neurônio de saída pode-se utilizar a equação (2.6.29) para determinar $e_j(n)$ que será utilizado para calcular o gradiente local com a equação (2.6.42).

Figura 13 – Exemplo de neurônio oculto para cálculo do *backpropagation*



Fonte: Adaptado de Haykin (2007)

Se o neurônio estiver localizado em uma camada oculta, não possui erro associado diretamente. Assim, é necessário retropropagar o erro recursivamente pelos neurônios que estão conectados (HAYKIN, 2007). Um exemplo de neurônio oculto pode ser visto na

arquitetura da figura 13 em que o gradiente local do neurônio oculto j, de acordo com a equação (2.6.42), é dado por

$$\delta_j(n) = -\left(\frac{\partial \xi(n)}{\partial y_j(n)}\right)\left(\frac{\partial y_j(n)}{\partial v_j(n)}\right) \Leftrightarrow \tag{2.6.43}$$

$$\delta_j(n) = -\left(\frac{\partial \xi(n)}{\partial y_j(n)}\right)\varphi^{'}(v_j(n)) \tag{2.6.44}$$

o erro do neurônio j, exibido pela figura 13, é dado por

$$\xi(n) = \frac{1}{2}\sum_{k=1}^{|C|} e_k^2(n) \tag{2.6.45}$$

em que k é um neurônio de saída, diferenciando a equação (2.6.45) em relação à $y_j(n)$ tem-se que

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \left(\sum_{k=1}^{|C|} \frac{\partial e_k(n)}{\partial y_j(n)}\right) \tag{2.6.46}$$

usando a regra da cadeia na equação (2.6.46) pode-se escrevê-la como

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_{k=1}^{|C|}\left[e_k(n)\left(\frac{\partial e_k(n)}{\partial v_k(n)}\right)\left(\frac{\partial v_k(n)}{\partial y_j(n)}\right)\right] \tag{2.6.47}$$

da figura 13 percebe-se que o erro é dado por

$$e_k(n) = d_k(n) - y_k(n) \Leftrightarrow \tag{2.6.48}$$

$$e_k(n) = d_k(n) - \varphi(v_k(n)) \tag{2.6.49}$$

então é possível reescrever o derivativo da equação (2.6.47) como

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi^{'}(v_k(n)) \tag{2.6.50}$$

a soma ponderada do neurônio k, proposto na figura 13, é dada por

$$v_k(n) = \sum_{j=0}^{m}(w_{kj}(n)y_j(n)) \tag{2.6.51}$$

sendo m o número total de entradas (excluindo bias) aplicados no neurônio k, o peso sináptico w_{k0} é igual ao bias $b_k(n)$ aplicado no neurônio k. Diferenciando a equação (2.6.51) em relação a $y_j(n)$ é obtido

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{2.6.52}$$

Utilizando as equações (2.6.50) até (2.6.52) na equação (2.6.47), é obtido o seguinte derivativo

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_{k=1}^{|C|} (e_k(n)\varphi'(v_k(n))w_{kj}(n)) \Leftrightarrow \tag{2.6.53}$$

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_{k=1}^{|C|} \delta_k(n)w_{kj}(n) \tag{2.6.54}$$

Na segunda linha, é utilizada a definição de gradiente local da equação (2.6.41) trocando o neurônio j por k. Usando a equação (2.6.54) na equação (2.6.43) obtem-se a fórmula do gradiente local como

$$\delta_j(n) = \varphi^{'}(v_j(n)) \sum_{k=1}^{|C|} \delta_k(n)w_{kj}(n) \tag{2.6.55}$$

O gradiente local $\delta_j(n)$ exige o cálculo do erro de todos os neurônios da camada seguinte e os pesos sinápticos dessas conexões, dessa forma pode-se escrever a regra delta como

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n) \tag{2.6.56}$$

que será usada para atualizar os pesos do vetor $\mathbf{w}$ no decorrer do treinamento. A técnica de rede neural será usada como classificador e regressor neste projeto. No primeiro caso são usadas as saídas da rede neural para atribuir classes as instâncias. No segundo caso são usados os pesos da rede neural treinada para predizer valores numéricos.

2.6.5 SVM Binário

Suport Vector Machines (SVM) é uma técnica que pode ser usada para classificar dados usando um hiperplano separador. O objetivo do SVM é escolher a posição do hiperplano tal que permita formar partições homogêneas de dados em ambos os lados da superfície de decisão. Considerando dados futuros, a melhor escolha é aquela que maximize a margem entre os dados e o hiperplano separador, nesse caso denominado *hiperplano ótimo* (LANTZ, 2013).

No decorrer desta seção será detalhada a técnica de SVM, iniciando pelo caso de dados linearmente separáveis, seguida pelo uso das margens flexíveis e finalizando com a generalização com uso do *kernel trick*. As explicações desta seção são baseadas nas obras

de Haykin (2007), Lima (2004), Fletcher (1987), Stewart (2013), Burges (1998), Cristianini e Shawe-Taylor (2000) e Chang e Lin (2011).

Dado um conjunto de dados linearmente separáveis como

$$(\{x_1, d_1\}), (\{x_2, d_2\}), \ldots (\{x_N, d_N\}) \text{ com } \mathbf{x} \in R^n \text{ e } d_i \in \{+1, -1\} \tag{2.6.57}$$

no qual o vetor $\mathbf{x}$ representa os dados e o vetor $\mathbf{d}$ representa as classes. Um hiperplano separador para estes dados é definido pela seguinte equação

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.6.58}$$

sendo $\mathbf{x}$ o vetor de entradas, $\mathbf{w}^T$ um vetor de pesos transposto e b um bias. Este hiperplano, para dados linearmente separáveis, acarreta nas seguintes restrições

$$\mathbf{w}^T\mathbf{x} + b \geq 0 \qquad \text{para } d_i = +1 \tag{2.6.59}$$

$$\mathbf{w}^T\mathbf{x} + b < 0 \qquad \text{para } d_i = -1 \tag{2.6.60}$$

dados $\mathbf{w}$ e b, o espaço que separa o hiperplano (2.6.58) e os dados mais próximos é denominado margem de separação (ρ), a qual pode ser vista na figura 14. O objetivo do SVM é maximizar essa margem. O hiperplano com margem máxima é conhecido por *hiperplano ótimo*. Tem sua equação definida como

$$\mathbf{w_0}^T\mathbf{x} + b_0 = 0 \tag{2.6.61}$$

o par de valores ótimos, $\mathbf{w_0}$ e b_0, devem satisfazer as seguintes restrições

$$\mathbf{w_0}^T\mathbf{x} + b_0 \geq +1 \qquad \text{para } d_i = +1 \tag{2.6.62}$$

$$\mathbf{w_0}^T\mathbf{x} + b_0 < -1 \qquad \text{para } d_i = -1 \tag{2.6.63}$$

para garantir que a solução seja única, mesmo reescalando valores de $\mathbf{w_0}$ e b_0, pois a equação (2.6.61) permanecerá inalterada (LIMA, 2004). Os pontos que satisfazem o sinal de igualdade das condições (2.6.62) e (2.6.63) (que são os pontos mais próximos do hiperplano ótimo) são conhecidos por vetores de suporte, que são os pontos destacados na figura 14. Com essas novas restrições, é possível definir a função discriminante como

$$g(x) = \mathbf{w_0}^T\mathbf{x} + b_0 \tag{2.6.64}$$

utilizando a função discriminante é possível definir a margem máxima como uma distância entre: i) hiperplano; e ii) ponto. Pois, para um dado vetor de suporte $\mathbf{x^s}$, para o qual $d^s = +1$, tem-se que

$$g(\mathbf{x^s}) = \mathbf{w_0^T x} + b_0 = \pm 1 \qquad \text{para } d_i = \pm 1 \tag{2.6.65}$$

dessa forma é possível escrever a distância entre os vetores de suporte e o hiperplano como

$$r = \frac{g(\mathbf{x^s})}{||\mathbf{w_0}||} = \begin{cases} \frac{1}{||\mathbf{w_0}||} \text{ para } d^s = +1 \\ \frac{-1}{||\mathbf{w_0}||} \text{ para } d^s = -1 \end{cases} \tag{2.6.66}$$

usando a equação (2.6.66) é possível definir a margem ótima como a distância entre os vetores de suporte da classe 1 ao hiperplano ótimo acrescido da distância entre os vetores de suporte da classe 2 ao hiperplano ótimo obtendo

$$\rho = 2r = \frac{2}{||\mathbf{w_0}||} \tag{2.6.67}$$

Portanto, para encontrar a margem máxima, ρ na figura 14, é necessário maximizar a norma Euclidiana do vetor de pesos definido pela equação (2.6.67) ou de forma equivalente minimizar w tal que

$$\Phi(w) = \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{2.6.68}$$

que é um problema de otimização quadrática convexa (LIMA, 2004) e o fator $\frac{1}{2}$ foi acrescentado para facilitar as derivações. Esse tipo de problema pode ser solucionado em três etapas. A primeira é construir o lagrangiano, a segunda é derivar as condições de otimização e a terceira é solucionar o problema no espaço dual dos multiplicadores de Lagrange (HAYKIN, 2007). Para solucionar esse problema de otimização quadrática é utilizada a técnica de multiplicadores de Lagrange (STEWART, 2013). Para tal, as restrições (2.6.62) e (2.6.63) foram combinadas em apenas uma restrição equivalente

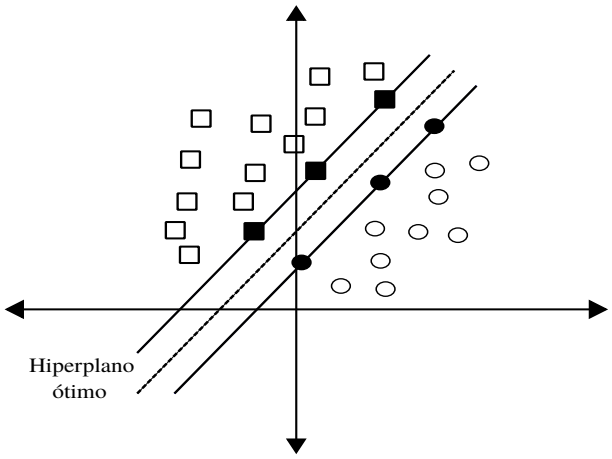
$$d_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 \qquad \text{para } i = 1, 2, 3, \ldots, N \tag{2.6.69}$$

o que permite escrever o seguinte Lagrangiano

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \left\{\alpha_i[d_i(\mathbf{w}^T\mathbf{x} + b) - 1]\right\} \tag{2.6.70}$$

no qual o primeiro termo é a função a ser minimizada e o segundo é a restrição reforçada pelos multiplicadores de Lagrange α. A solução da equação (2.6.70) é dada pelo ponto

Figura 14 – Vetores de suporte e margem do SVM



Fonte: Adaptado de Haykin (2007)

de sela do Lagrangiano (ver figura 15) que deve ser minimizado em relação a $\mathbf{w}$ e b, isto requer duas condições

$$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 \tag{2.6.71}$$

$$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial b} = 0 \tag{2.6.72}$$

aplicando a condição (2.6.71) na equação (2.6.70) obtem-se

$$\mathbf{w} = \sum_{i=1}^{N} (\alpha_i d_i \mathbf{x_i}) \tag{2.6.73}$$

aplicando a condição (2.6.72) na equação (2.6.70) obtem-se

$$\sum_{i=1}^{N} (\alpha_i d_i) = 0 \tag{2.6.74}$$

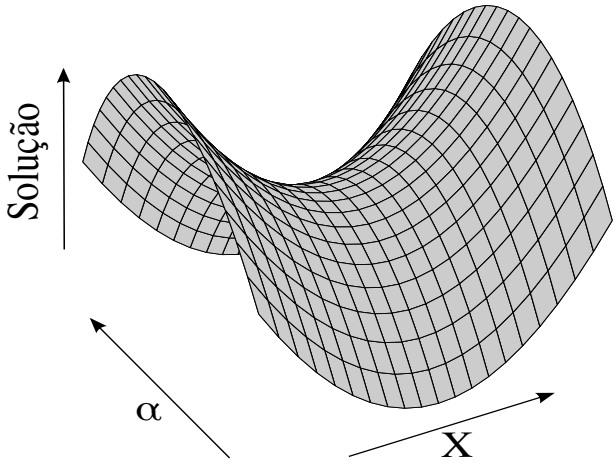
Segundo Fletcher (1987), o problema dual pode ser formulado com duas premissas. A primeira é que se o problema primal tem uma solução ótima, o dual também a possui com os mesmos valores de ótimo. Portanto pode-se solucionar o problema dual e usar seus resultados para facilitar os cálculos para encontrar $\mathbf{w_0}$ e b_0. Se $\mathbf{w_0}$ é a solução ótima para o problema primal então α_0 é a solução ótima para o problema dual, logo é suficiente e necessário que $\mathbf{w_0}$ seja factível para o problema primal e

$$J(\mathbf{w_0}, b_0, \alpha_0) = \min_{\mathbf{w}} J(\mathbf{w}, b, \alpha) \tag{2.6.75}$$

Dessa forma é possível formular o problema dual expandindo a equação (2.6.70) para

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \left(\alpha_i d_i \mathbf{w}^T \mathbf{x_i}\right) - b\sum_{i=1}^{N} (\alpha_i d_i) + \sum_{i=1}^{N} (\alpha_i) \tag{2.6.76}$$

Figura 15 – Ponto de sela do Lagrangiano



Fonte: Adaptado de Haykin (2007)

o terceiro termo da equação (2.6.76) é zero segundo a restrição (2.6.74), podendo ser removido. Pela restrição (2.6.73) é possível escrever

$$\mathbf{w}^T = \sum_{j=1}^{N} \left(\alpha_j d_j \mathbf{w}^T \mathbf{x_j}\right) \tag{2.6.77}$$

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{N}\sum_{j=1}^{N} \left(\alpha_i \alpha_j d_i d_j \mathbf{x_i}^T \mathbf{x_j}\right) \tag{2.6.78}$$

agora é possível reformular a função $J(\mathbf{w}, b, \alpha) = Q(\alpha)$ e, usando o valor de (2.6.78), é possível escrever o problema dual como

$$Q(\alpha) = \sum_{i=1}^{N} (\alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \left(\alpha_i \alpha_j d_i d_j \mathbf{x_i}^T \mathbf{x_j}\right) \tag{2.6.79}$$

dessa forma é possível reformular o problema de otimização quadrática para maximizar a função objetivo como

$$Q(\alpha) = \sum_{i=1}^{N} (\alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \left(\alpha_i \alpha_j d_i d_j \mathbf{x_i}^T \mathbf{x_j}\right) \tag{2.6.80}$$

com as restrições

$$\sum_{i=1}^{N} (\alpha_i d_i) = 0 \tag{2.6.81}$$

$$\alpha_i \geq 0 \qquad \text{para } i = 1, 2, 3, \ldots, N \tag{2.6.82}$$

a nova função objetivo tem como vantagem depender exclusivamente dos dados de treinamento em forma de produto vetorial. Os valores ótimos de α denominados α_0 são

aqueles iguais a zero para os vetores de suporte e os diferentes de zero para os outros dados de treinamento. Determinar o vetor de peso ótimo ($\mathbf{w_0}$) pode ser feito usando a equação (2.6.73) tal que

$$\mathbf{w_0} = \sum_{i=1}^{N_S} (\alpha_{0,i} d_i \mathbf{x_i}) \tag{2.6.83}$$

sendo N_S os vetores de suporte, com os valores de α diferentes de zero. Para o cálculo do bias ótimo (b_0) pode-se utilizar $\mathbf{w_0}$ a função

$$g(\mathbf{x^s}) = \mathbf{w_0}^T \mathbf{x^s} + b_0 = \pm 1 \tag{2.6.84}$$

e pode-se utilizar a relação $d_i^2 = 1$ obtendo

$$b_0 = \frac{\sum_{j=1}^{N_S} \left(d_j - \sum_{i=1}^{N_S} \left(\alpha_{0,i} d_i \mathbf{x_i}^T \mathbf{x^s} \right) \right)}{N_S} \tag{2.6.85}$$

em que $\mathbf{x^s}$ é um vetor de suporte cujo multiplicador de Lagrange é diferente de zero.

O caso em que os dados **não são separáveis** pode ser visualizado na figura 16. A parte *a* da figura exibe o caso no qual o dado (preenchido em preto) foi classificado corretamente, porém após os vetores de suporte. A parte *b* exibe um erro de classificação, pois o dado (preenchido em preto), da classe círculo, está localizada após o hiperplano ótimo. Nessa situação uma possível solução é tentar minimizar o erro de classificação usando variáveis de *folga* ξ_i, para tal a equação (2.6.69) é alterada para

$$d_i(\mathbf{w}^T \mathbf{x} + b) \geq 1 - \xi \qquad \text{para } i = 1, 2, 3, \ldots, N \tag{2.6.86}$$

as variáveis ξ_i metrificam o desvio de um ponto ao hiperplano ótimo. Dessa forma, valores $0 < \xi_i \leq 1$ significam que o dado se encontra no lado correto da superfície de decisão, para valores $\xi_i > 1$ o dado está do lado errado da superfície de decisão. Portanto, o objetivo é minimizar a função

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1) \tag{2.6.87}$$

$$I(x) = \begin{cases} 0 \text{ se } x \leq 0 \\ 1 \text{ se } x > 0 \end{cases} \tag{2.6.88}$$

Para tornar a equação (2.6.87) diferenciável, é usada a seguinte aproximação

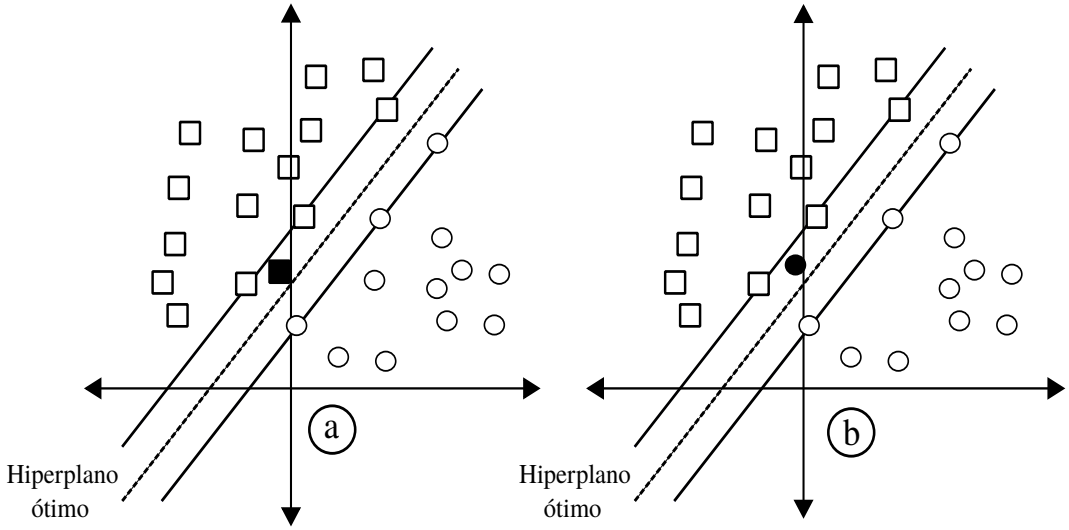
$$\Phi(\xi) = \sum_{i=1}^{N} (\xi_i) \tag{2.6.89}$$

Agora pode-se reescrever o problema (2.6.89) em função do vetor de pesos $\mathbf{w}$

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}(\xi_i) \tag{2.6.90}$$

O primeiro termo é relacionado ao problema de minimizar a norma euclidiana do vetor de pesos, existente pela definição do SVM. O segundo termo controla o problema de dados não separáveis, o qual funciona como um teto do número de erros aceitos no processo de treinamento. Quanto maior o valor da variável C, maior o número de pontos fora da curva no conjunto de dados, quanto menor seu valor espera-se um conjunto de dados com menos *outliers*.

Figura 16 – SVM considerando erro de classificação



Fonte: Adaptado de Haykin (2007)

Dessa forma, o problema pode ser visto como um caso especial do caso linearmente separável, permitindo reformular o primal anterior, ver equação (2.6.70), como

$$\begin{aligned} J(\mathbf{w}, b, \alpha, \beta, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}(\xi_i) \\ - \sum_{i=1}^{N}\left\{\alpha_i[d_i(\mathbf{w}^T\mathbf{x} + b) - 1 + \xi_i]\right\} - \sum_{i=1}^{N}(\beta_i\xi_i) \end{aligned} \tag{2.6.91}$$

cujo ponto de mínimo é dado pelas condições

$$\frac{\partial J(\mathbf{w}, b, \alpha, \beta, \xi)}{\partial \mathbf{w}} = 0 \tag{2.6.92}$$

$$\frac{\partial J(\mathbf{w}, b, \alpha, \beta, \xi)}{\partial b} = 0 \tag{2.6.93}$$

$$\frac{\partial J(\mathbf{w}, b, \alpha, \beta, \xi)}{\partial \xi} = 0 \tag{2.6.94}$$

aplicando a condição (2.6.71) na equação (2.6.91) obtem-se

$$\mathbf{w} = \sum_{i=1}^{N} (\alpha_i d_i \mathbf{x_i}) \tag{2.6.95}$$

aplicando a condição (2.6.72) na equação (2.6.91) obtem-se

$$\sum_{i=1}^{N} (\alpha_i d_i) = 0 \tag{2.6.96}$$

aplicando a condição (2.6.94) na equação (2.6.91) obtem-se

$$\alpha_i + \beta_i = C \tag{2.6.97}$$

segundo Lima (2004) as condições de KKT para o problema são

$$\alpha_i[d_i(w^T x + b) - 1 + \xi_i] = 0 \tag{2.6.98}$$

$$\alpha_i \geq 0 \tag{2.6.99}$$

$$\beta_i \geq 0 \tag{2.6.100}$$

$$\beta_i \xi_i = 0 \tag{2.6.101}$$

utilizando as condições (2.6.95) até (2.6.101) é possível escrever o problema dual como

$$Q(\alpha) = \sum_{i=1}^{N} (\alpha_i) - \frac{1}{2} \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \left(\alpha_i \alpha_j d_i d_j \mathbf{x_i}^T \mathbf{x_j} \right) \right) \tag{2.6.102}$$

com as restrições

$$\sum_{i=1}^{N} (\alpha_i d_i) = 0 \tag{2.6.103}$$

$$0 < \alpha_i \leq C \qquad \text{para } i = 1, 2, 3, \ldots, N \tag{2.6.104}$$

a qual é idêntica ao caso separável acrescida de uma condição limitante (equação (2.6.104)) para o valor dos multiplicadores de Lagrange. O cálculo dos valores ótimos $\mathbf{w_0}$ e b_0 é feito de forma análoga ao caso linear acrescida da restrição (2.6.104). Determinar o vetor de peso ótimo ($\mathbf{w_0}$) pode ser feito usando a equação (2.6.95), tal que

$$\mathbf{w_0} = \sum_{i=1}^{N_S} (\alpha_{0,i} d_i \mathbf{x_i}) \tag{2.6.105}$$

com a restrição

$$0 < \alpha_i \leq C$$

em que N_S são os vetores de suporte e os valores de α satisfazem (2.6.104). Para o cálculo do bias ótimo (b_0) pode-se utilizar $\mathbf{w_0}$ e

$$g(\mathbf{x^s}) = \mathbf{w_0^T x^s} + b_0 = \pm 1 \tag{2.6.106}$$

e pode-se utilizar a relação $d_i^2 = 1$ obtendo

$$b_0 = \frac{\sum_{j=1}^{N_S} \left(d_j - \sum_{i=1}^{N_S} \left(\alpha_{0,i} d_i \mathbf{x_i}^T \mathbf{x^s} \right) \right)}{N_S} \tag{2.6.107}$$

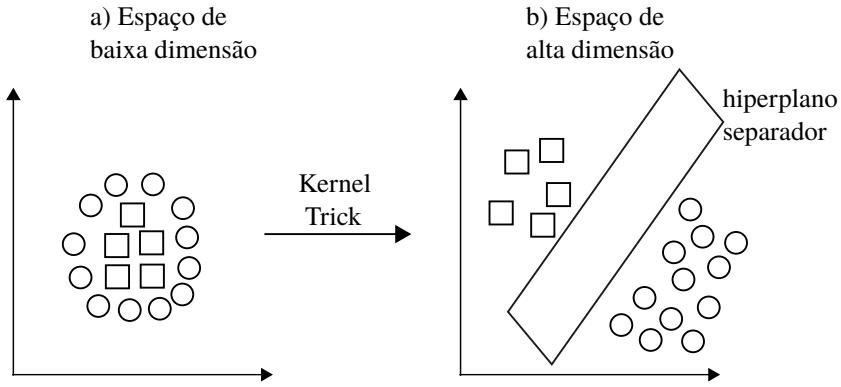
com a restrição

$$0 < \alpha_i \leq C$$

em que $\mathbf{x^s}$ é um vetor de suporte cujo multiplicador de Lagrange é diferente de zero e os valores de α satisfazem (2.6.104).

Quando uma fronteira de decisão linear não for adequado, como na figura 17, é possível mapear os dados de entrada para um espaço de dimensão superior, solucionar o problema que potencialmente se torna linear e mapea-lo novamente para um espaço de dimensão original (HAYKIN, 2007). Para compreender este mapeamento serão alterados

Figura 17 – SVM com dados não linearmente separáveis



Fonte: Adaptado de Lantz (2013)

alguns detalhes das soluções já apresentadas para os casos anteriores. A equação (2.6.83) será reescrita como

$$\sum_{i=1}^{N_S} \left(\alpha_i d_i \phi^{\mathbf{T}}(x_i) \phi(x) \right) = 0 \tag{2.6.108}$$

em que a função $\phi^{\mathbf{T}}(x_i)\phi(x_i)$ é um produto interno de valores que pode ser definido como

$$k(x, x_i) = \phi^{\mathbf{T}}(x_i)\phi(x) \tag{2.6.109}$$

sendo a equação (2.6.109) um *Kernel*, o qual é responsável por mapear os dados em alta dimensão (SHAWE-TAYLOR; CRISTIANINI, 2004). Agora é possível reescrever o hiperplano ótimo como

$$\sum_{i=1}^{N_S} (\alpha_i d_i k(\mathbf{x}, \mathbf{x_i})) = 0 \tag{2.6.110}$$

com o uso da função de *kernel* k é desnecessário calcular o vetor de pesos ótimo $\mathbf{w_0}$, este fato é conhecido por *kernel trick*. Por fim, com o uso do *kernel trick* é possível reescrever o problema de otimização dual como

$$Q(\alpha) = \sum_{i=1}^{N} (\alpha_i) - \frac{1}{2} \sum_{i=1}^{N} \left(\sum_{j=1}^{N} (\alpha_i \alpha_j d_i d_j k(\mathbf{x_i}, \mathbf{x_j})) \right) \tag{2.6.111}$$

com as restrições

$$\sum_{i=1}^{N} (\alpha_i d_i) = 0 \tag{2.6.112}$$

$$0 < \alpha_i \leq C \qquad \text{para } i = 1, 2, 3, \ldots, N \tag{2.6.113}$$

que é idêntico ao caso não separável acrescido da função de *kernel*. Alguns tipos de função *Kernel* utilizadas (HAYKIN, 2007; BURGES, 1998) são:

$$K(x, y) = (\mathbf{x}^T \mathbf{y} + 1)^p \qquad \text{Polinomial} \tag{2.6.114}$$

$$K(x, y) = \exp\left(-\frac{1}{2\sigma^2} ||\mathbf{x} - \mathbf{y}||^2 \right) \qquad \text{Radial} \tag{2.6.115}$$

$$K(x, y) = \tanh(coef0(\mathbf{x}^T \mathbf{y}) - \delta) \qquad \text{Sigmoide} \tag{2.6.116}$$

$$K(x, y) = (\mathbf{x}^T \mathbf{y}) \qquad \text{Linear} \tag{2.6.117}$$

sendo p, σ, δ e $coef0$ parâmetros passados pelo usuário. A solução deste problema é idêntica para o caso anterior acrescentando a função de *kernel*.

2.7 Recomendação por regressão

Segundo Han, Kamber e Pei (2011) a diferença entre classificar e efetuar uma regressão é que a última tem por objetivo predizer valores contínuos ao invés de discretos.

2.7.1 Regressão com SVM

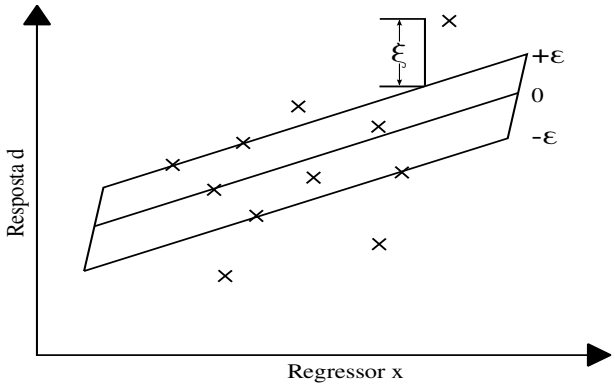
O SVM para regressão tem por objetivo aproximar ao máximo o hiperplano ótimo dos dados com uma dada tolerância para erros (LIMA, 2004). Dado este objetivo, é necessário um modelo insensível em relação a pequenas alterações nos parâmetros. Para atingir este objetivo deve-se utilizar alguma métrica que quantifique a robustez e permita minimizar a degradação máxima do SVM para um determinado valor de insensibilidade denominado ϵ (HAYKIN, 2007). Será usada a função

$$L_\epsilon(d, y) = \begin{cases} |d - y| - \epsilon & \text{para } |d - y| \geq \epsilon \\ 0 & \text{caso contrário} \end{cases} \tag{2.7.1}$$

$$y = \mathbf{w^T}\boldsymbol{\phi}(\boldsymbol{x}) + b \tag{2.7.2}$$

proposta por Vapnik (1998), em que ϵ é definido pelo usuário, d é o rótulo da instância e y é a saída do preditor.

Figura 18 – Tubo de insensibilidade



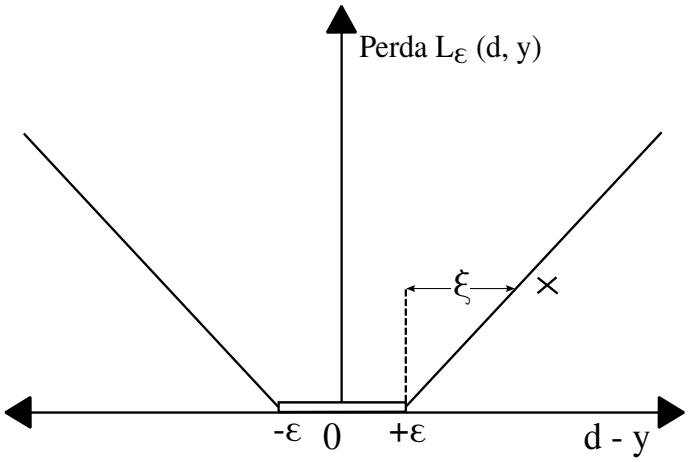
Fonte: Adaptado de Haykin (2007)

A figura 18 apresenta graficamente a relação entre a função de perda (2.7.1) e o regressor, deve-se ressaltar que o valor de erro só será contabilizado se o valor do erro ξ satisfizer a seguinte inequação $|d - y| \geq \epsilon$.

A figura 19 exibe a função de perda 2.7.1 em função do erro. Dessa forma, fica claro que o valor de ϵ é considerado um limiar, a partir do qual o erro começa a ser calculado. Adaptar o SVM para regressão significa encontrar duas variáveis, $\mathbf{w}$ e b, para a relação de regressão definida por

$$d = \mathbf{w}^T\mathbf{x} + b \tag{2.7.3}$$

Figura 19 – Gráfico da função perda



Fonte: Adaptado de Haykin (2007)

em que d é a variável resposta e $\mathbf{x}$ são as variáveis preditoras. A função perda (2.7.1) permite adaptar o problema de otimização formulado como (2.6.68) na seção 2.6.5 para

$$\Phi(w) = \frac{1}{2}||\mathbf{w^T w}|| + C\sum_{i=1}^{N} L_\epsilon(d, y) \tag{2.7.4}$$

com as seguintes restrições

$$d_i - y_i \leq \epsilon + \xi \tag{2.7.5}$$

$$y_i - d \leq \epsilon + \xi^{'} \tag{2.7.6}$$

$$\xi^{'}, \xi \geq 0 \qquad \text{para } i = 1, 2, 3 \ldots N \tag{2.7.7}$$

nas quais ξ e $\xi^{'}$ são variáveis de folga, positivas e idênticas a definida no caso não separável do classificador (ver seção 2.6.5). Utilizando as restrições (2.7.5), (2.7.6) e (2.7.7) e a função objetivo (2.7.4) é possível construir o Lagrangiano Primal, como no caso da formulação do SVM para classificação (ver seção 2.6.5), obtendo

$$J(\mathbf{w}, \xi^{'}, \xi, \alpha, \alpha^{'}, \gamma, \gamma^{'}) = \frac{1}{2}||\mathbf{w}^T\mathbf{w}|| + C\sum_{i=1}^{N}\xi_i + \xi_i^{'} - \sum_{i=1}^{N}(\gamma\xi_i + \gamma^{'}\xi_i^{'})$$

$$- \sum_{i=1}^{N}\alpha_i((\mathbf{w^T x}) + b - d_i + \epsilon + \xi_i) - \sum_{i=1}^{N}\alpha_i^{'}(d_i - (\mathbf{w^T x}) - b + \epsilon + \xi_i^{'}) \tag{2.7.8}$$

a solução do Lagrangiano (2.7.8) é dada no ponto de sela (análoga ao caso não separável do classificador na seção 2.6.5). Dessa forma, deve-se calcular suas derivadas parciais em relação a $\mathbf{w}$, b, ξ e $\xi^{'}$ e igualar todas a zero (HAYKIN, 2007) obtendo as restrições

$$\frac{\partial J(\mathbf{w},\xi^{'},\xi,\alpha,\alpha^{'},\gamma,\gamma^{'})}{\partial \mathbf{w}} = 0 \tag{2.7.9}$$

$$\frac{\partial J(\mathbf{w},\xi^{'},\xi,\alpha,\alpha^{'},\gamma,\gamma^{'})}{\partial b} = 0 \tag{2.7.10}$$

$$\frac{\partial J(\mathbf{w},\xi^{'},\xi,\alpha,\alpha^{'},\gamma,\gamma^{'})}{\partial \xi} = 0 \tag{2.7.11}$$

$$\frac{\partial J(\mathbf{w},\xi^{'},\xi,\alpha,\alpha^{'},\gamma,\gamma^{'})}{\partial \xi^{\prime}} = 0 \tag{2.7.12}$$

aplicando a condição (2.7.9) na equação (2.7.8) obtem-se

$$\hat{\mathbf{w}} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^{'})x \tag{2.7.13}$$

aplicando a condição (2.7.10) na equação (2.7.8) obtem-se

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^{'}) = 0 \tag{2.7.14}$$

aplicando a condição (2.7.11) na equação (2.7.8) obtem-se

$$\alpha + \gamma = C \tag{2.7.15}$$

aplicando a condição (2.7.12) na equação (2.7.8) obtem-se

$$\alpha^{'} + \gamma^{'} = C \tag{2.7.16}$$

o parâmetro desejado $\hat{\mathbf{w}}$ foi obtido pela equação (2.7.13) em função dos multplicadores de Lagrange $\alpha^{'}, \alpha$. O valor de $\hat{b}$ pode ser obtido pelas condições de KKT, que segundo Haykin (2007), são

$$\alpha(\epsilon + \xi + d_i - y_i) = 0 \tag{2.7.17}$$

$$\alpha^{'}(\epsilon + \xi^{'} - d_i + y_i) = 0 \tag{2.7.18}$$

$$(C - \alpha)\xi = 0 \tag{2.7.19}$$

$$(C - \alpha^{'})\xi^{'} = 0 \tag{2.7.20}$$

analisando as equações (2.7.17) e (2.7.18) pode-se concluir que instâncias nas quais $C = \alpha$ ou $C = \alpha^{'}$ residem fora das variáveis de folga ξ e $\xi^{'}$. Multiplicando (2.7.17) por α e (2.7.18) por $\alpha^{'}$ e somando-as obtem-se

$$\alpha\alpha^{'}(2\epsilon + \xi + \xi^{'}) = 0 \tag{2.7.21}$$

sempre que $\epsilon > 0$, $\xi > 0$ e $\xi^{'} > 0$ implica que $\alpha\alpha^{'} = 0$, logo ambos os multiplicadores de Lagrange não podem ser zero simultaneamente. Pelas equações (2.7.19) e (2.7.20) pode-se escrever

$$\xi = 0 \qquad \text{para } 0 < \alpha < C \tag{2.7.22}$$

$$\xi^{'} = 0 \qquad \text{para } 0 < \alpha^{'} < C \tag{2.7.23}$$

as equações (2.7.22) e (2.7.23) usadas em (2.7.17) e (2.7.18) implicam que

$$\epsilon - d_i + y_i = 0 \qquad \text{para } 0 < \alpha < C \tag{2.7.24}$$

$$\epsilon + d_i - y_i = 0 \qquad \text{para } 0 < \alpha^{'} < C \tag{2.7.25}$$

agora que as condições de KKT e suas implicações foram discutidas para concluir o cálculo do estimador ótimo da função de regressão dado por

$$y = \hat{\mathbf{w}}^{\mathbf{T}}\mathbf{x} + \hat{b} \tag{2.7.26}$$

sendo que $\hat{\mathbf{w}}$ já foi calculado pela equação (2.7.13). Falta calcular $\hat{b}$ que será calculado por meio da equação (2.7.26) e restrito às condições (2.7.24) e (2.7.25) obtendo

$$\hat{b} = d_i - \hat{\mathbf{w}}^{\mathbf{T}}\mathbf{x} - \epsilon \qquad \text{para } 0 < \alpha < C \tag{2.7.27}$$

$$\hat{b} = d_i - \hat{\mathbf{w}}^{\mathbf{T}}\mathbf{x} + \epsilon \qquad \text{para } 0 < \alpha^{'} < C \tag{2.7.28}$$

Com o valor de $\hat{w}$ obtido da equação (2.7.13) e dados ϵ e d_i foi possível computar $\hat{b}$. Dessa forma o estimador ótimo (2.7.26) pode ser usado para predizer valores.

2.7.2 MARS

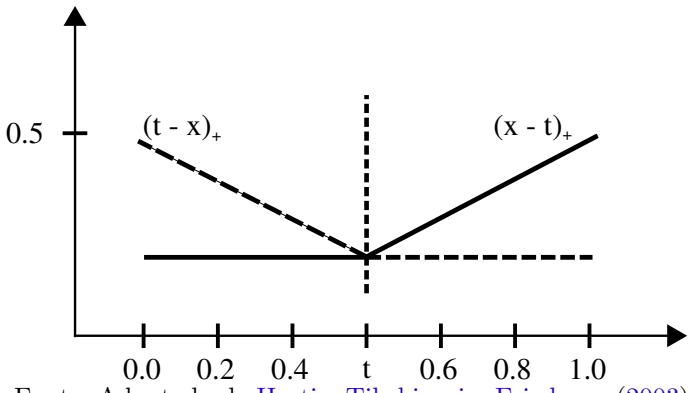
O algoritmo *Multivariate Adaptive Regression Splines* (MARS), segundo Hastie, Tibshirani e Friedman (2003), é uma generalização da regressão linear por função degrau. Para tal, usa-se segmentos lineares de funções com a seguinte estrutura

$$(x-t)_+ = \begin{cases} x-t, \text{ se } x > t \\ 0, \text{ caso contrário} \end{cases} \tag{2.7.29}$$

$$(t-x)_+ = \begin{cases} t-x, \text{ se } x < t \\ 0, \text{ caso contrário} \end{cases} \tag{2.7.30}$$

Assim como na figura 20, na qual são visualizadas as funções espelho $(x-0.5)_+$ e $(0.5-x)_+$, cada segmento é representado por uma função linear e possui um nó no valor de x. Diversas funções espelho são unidas para formar o modelo de regressão MARS, que nesse ponto está pronto para predizer valores.

Figura 20 – Funções espelho usadas pelo MARS



Fonte: Adaptado de Hastie, Tibshirani e Friedman (2003)

2.7.3 Regressão Logística

A regressão logística prediz valores (variável de resposta) contínuos baseados em valores categóricos ou numéricos contínuos (variáveis preditoras). Sua modelagem ocorre por meio do *generalized linear model* que é uma abordagem unificada para modelar alguns tipos de variáveis (AGRESTI, 2015). Este tipo de modelo é composto por três componentes necessários (AGRESTI, 2015; OLSSON, 2002) para predizer valores: i) a distribuição das

variáveis; ii) uma função *link*, cujo objetivo é relacionar a média condicional da distribuição com o preditor linear; e iii) o preditor linear. O preditor linear é dado por

$$g(\mu_Y) = \beta_0 + \sum_{j=1}^{P} \beta_j X_j \tag{2.7.31}$$

, a função *link* é dada por

$$g(\mu_Y) = \log_e \left(\frac{\pi}{1-\pi} \right) \tag{2.7.32}$$

e a distribuição de probabilidade assumida pela regressão logística é a *binomial*. Em relação às equações (2.7.31) e (2.7.32) tem-se que $\pi = \mu_Y$ é a média condicional, que significa que a probabilidade de $Y = 1$, dado um conjunto de valores de X; e $\frac{\pi}{1-\pi}$ é a probabilidade de que $Y = 1$ (OLSSON, 2002). Usando essas equações e assumindo uma distribuição binomial para os dados, a regressão logística está pronta para predizer valores pois os valores dos coeficientes β foram estimados por maima verossimilhança e substituídos na equação (2.7.31) que agora receberá valores de x históricos para prever novos valores.

2.8 Classificadores compostos

Nessa seção serão descritos dois classificadores compostos, denominados dessa forma pois utilizam o resultado de outros classificadores/regressores e o rótulo das instâncias para aprender. O primeiro consiste no uso do classificador SVM (descrito na seção 2.6.5) sobre o resultado dos outros 5 classificadores e 5 regressores. O segundo é um *ensemble* de classificadores conhecido como *Rotation Forest*, o qual foi proposto por Rodríguez, Kuncheva e Alonso (2006).

Ambas as técnicas usam uma matriz M de entrada, criada pelos autores e apresentada pela tabela 6, que contém 8.614 casos de recomendação (que são as linhas da tabela) e uma coluna de rótulo. Cada célula da matriz M representa a posição em que o item removido i foi recomendado pelo sistema j.

2.8.1 Análise de Componentes Principais

A Análise de Componentes Principais (PCA) tem por objetivo maximizar a variância. A abordagem utilizada neste trabalho é baseada em matriz de covariância. O objetivo da PCA é reduzir dimensões de p para d onde $p < d$.

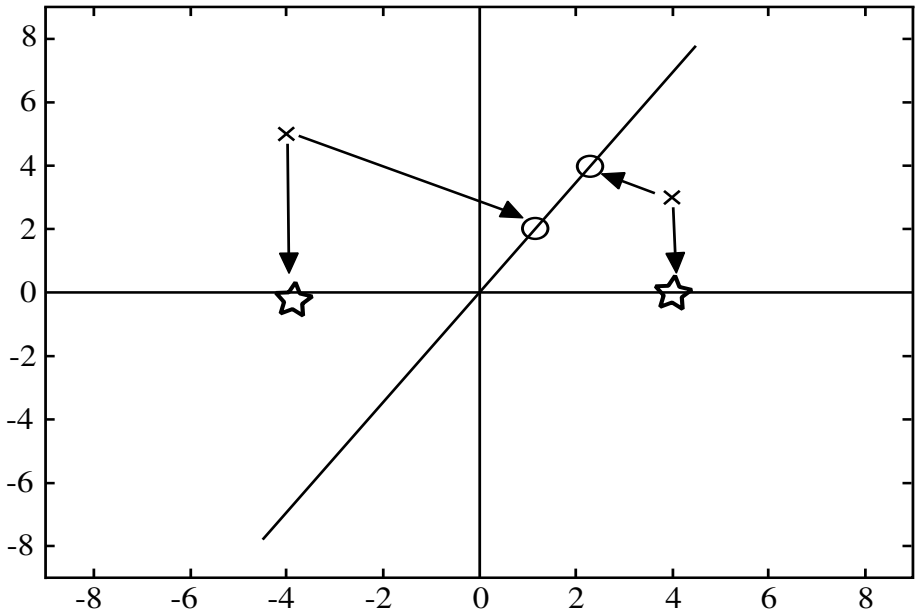
Tabela 6 – Matriz de entrada para classificadores compostos

#	**Workflow**	**Sist. 01**	...	**Sist. N**	**Ativ. removidas**	**Rótulo**
1	01	12	...	1	Ativ 01	T
2	01	13	...	1	Ativ 01	T
⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	01	1	...	1	Ativ 01	T
1	01	4	...	1	Ativ 02	F
2	01	7	...	1	Ativ 03	F
⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	01	8	...	2	Ativ 59	F
	⋮					
1	73	9	...	10	Ativ 02	T
2	73	18	...	0	Ativ 02	T
⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	73	65	...	6	Ativ 02	T
1	73	67	...	7	Ativ 03	F
2	73	5	...	8	Ativ 04	F
⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	73	14	...	9	Ativ 59	F

Fonte: Adilson Lopes Khouri, 2016

O algoritmo transforma o conjunto de dados originais por meio de uma combinação linear das variáveis. Os dados são projetados de forma ortogonal. Um exemplo dessa projeção pode ser visualizado na figura 21, extraída de Martinez (2004).

Figura 21 – Projeção ortogonal dos dados do algoritmo PCA



Fonte: Adaptado de Martinez (2004)

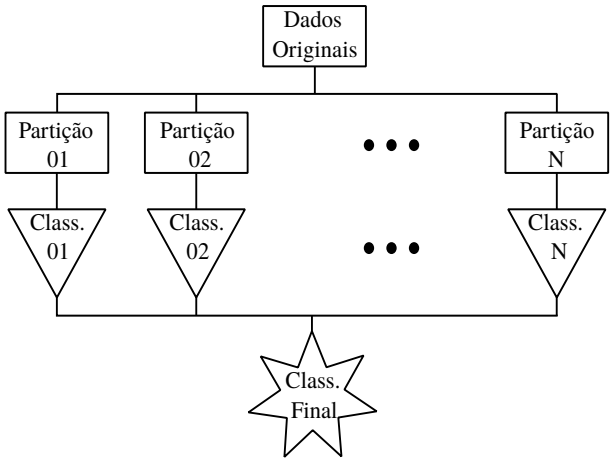
Há duas possíveis implementações de PCA mais usuais, por matriz de correlação e por matriz de covariância. No algoritmo *Rotation Forest* é usada a abordagem de matriz de covariância. O primeiro passo é calcular a matriz de covariância da média dos dados

de entrada centralizados, formando uma nova matriz N. Em seguida, são calculados os autovalores e autovetores da matriz N. Por fim, os autovetores são usados para projetar os dados em outro eixo como na figura 21.

2.8.2 Rotation Forest

Um *ensemble* de classificadores é um conjunto de classificadores, com alguma forma de diversificação, que resolvem o mesmo problema (particionado) e juntam suas soluções para chegar em um resultado final. A figura 22 exibe uma arquitetura típica de um *ensemble* de classificadores, na qual o retângulo, sem numeração, representa o conjunto dos dados originais, os retângulos com numeração são as partições, os triângulos são os classificadores e a estrela de sete pontas a união das respostas dos classificadores.

Figura 22 – Arquitetura de *ensemble* de classificadores



Fonte: Adaptado de Rodríguez, Kuncheva e Alonso (2006)

A técnica proposta por Rodríguez, Kuncheva e Alonso (2006) cria um *ensemble* de classificadores por meio de extração de características. O conjunto de dados é dividido em K subconjuntos (que é um parâmetro do algoritmo). Em seguida a técnica Principal Component Analysis (PCA) (ver seção 2.8.1) é aplicada em cada subconjunto. São mantidas todas as componentes principais, o que, segundo os autores, preserva a variabilidade dos dados. O próximo passo é multiplicar o conjunto de características originais pelos componentes principais. Cada subconjunto é apresentado a seu respectivo classificador, e após cada um destes solucionarem o problema, a instância a ser classificada será atribuída à classe com maior probabilidade. Este valor é obtido pela média de todos os classificadores do *ensemble*.

Considerando que $x = [x_i, \dots x_n]$ é uma instância com n dimensões, X é uma matriz de instâncias com dimensões $N \times n$, Y é o vetor de rótulos de X, que assumem os valores $y_i = [C_1, C_2, \dots, C_n]$, seja $D_1, D_2, \dots, D_L$ o conjunto de classificadores usados e F o conjunto de características. Os classificadores usados nesse *ensemble* são todos árvores de classificação e regressão (CART), descritos na seção 2.6.2. A construção do conjunto de treinamento para o classificador D_i pode ser realizado em quatro passos.

Primeiramente o conjunto de dados original é dividido em K partições disjuntas, seja $F_{i,j}$ o conjunto das características da partição do classificador D_i e seja $X_{i,j}$ o conjunto de dados formado pelas características $F_{i,j}$. Em seguida, para cada classificador, deve-se remover um subconjunto de classes de $X_{i,j}$ e selecionar uma amostra do tipo *bootstrap* (com reamostragem), com tamanho de 75% do número de seus objetos. O novo conjunto será denominado $X'_{i,j}$. O próximo passo é aplicar a PCA em $X'_{i,j}$ para obter a seguinte matriz de coeficientes

$$R_i = \begin{bmatrix} a_{i,1}^{(1)}, a_{i,1}^{(2)}, \dots a_{i,1}^{(M_1)} & [0] & \dots & [0] \\ [0] & a_{i,2}^{(1)}, a_{i,2}^{(2)}, \dots a_{i,2}^{(M_2)} & \dots & [0] \\ \vdots & \vdots & \ddots & \vdots \\ [0] & [0] & \dots & a_{i,K}^{(1)}, a_{i,K}^{(2)}, \dots a_{i,K}^{(M_K)} \end{bmatrix} \tag{2.8.1}$$

Em seguida deve-se rearranjar a matriz R_i para que a ordem dos coeficientes seja a mesma das características de F criando a matriz R_i^a. Dessa forma, o classificador D_i pode ser treinado usando (XR_i^a, Y) como conjunto de treinamento. Para a fase de classificação é usada a seguinte equação

$$\mu_j(x) = \frac{1}{L} \sum_{i=1}^{L} d_{i,j}(XR_i^a) \qquad j = 1, \dots, c \tag{2.8.2}$$

na qual $d_{i,j}(XR_i^a)$ é a probabilidade atribuída pelo classificador D_i que x seja da classe C_j, c é o número total de classes do problema e a equação (2.8.2) representa a confiança média de todos os classificadores do conjunto D de que x seja da classe C_j.

3 Revisão da literatura correlata

‘‘Não odeie seus inimigos. Isso afeta seu julgamento’’(Vito Corleone)

Este capítulo detalha os trabalhos presentes na literatura correlata analisados por meio de uma revisão sistemática. A seção 3.1 detalha as três fases da revisão sistemática: i) planejamento; ii) condução; e iii) extração de dados. A seção 3.2 apresenta os artigos com técnicas correlatas na área de recomendação de atividades em *workflows* científicos. A seção 3.4 apresenta uma comparação entre a solução proposta, os trabalhos correlatos e com as restrições típicas de sistemas de recomendação de *workflows* científicos (ver seção 2.3).

3.1 Metodologia da revisão sistemática

A revisão iniciou com um estudo exploratório sobre o tema permitindo compreender o problema de recomendação em detalhes, encontrar termos específicos da área de *workflows* científicos e definir palavras-chave. Em seguida, foi realizada uma revisão sistemática composta por três etapas, como proposto por Biolchini et al. (2007), sendo elas: i) planejamento; ii) condução; e iii) extração de dados. O objetivo dessa revisão foi responder à pergunta: ‘‘Quais são as técnicas existentes para recomendar atividades em *workflows*?’’.

3.1.1 Planejamento

Nesta etapa, foram definidas: a pergunta que a revisão objetivou responder, as bibliotecas digitais utilizadas na pesquisa, a *string* de busca e os critérios de inclusão e exclusão.

A revisão sistemática pretende responder a pergunta: ‘‘Quais são as técnicas existentes para recomendar atividades em *workflows*?’’. Para responder a esta pergunta, foram selecionadas as bibliotecas digitais: ACMDL (ACMDL, 2014), IEEExplore (IEEE, 2014) e Science Direct (DIRECT, 2014) por serem consideradas as principais bibliotecas digitais da área que disponibilizam artigos completos de diversos periódicos e conferências.

A *string* de busca foi definida com auxílio da metodologia PICO (HUANG; LIN; DEMNER-FUSHMAN, 2006), a qual define quatro grupos de palavras-chave e os relaciona com o operador *AND*, os quais estão enumerados a seguir:

1. **Population:** *scientific*, *workflow* e *pipeline*;
2. **Intervention:** *recommendation*, *provenance*, *suggestion*, *forecast*, *advice*, *design*, *visualization*, *recommender*, *construct*, *proposal*, *guidance*, *counsel*, *composition*, *activity*, *shimming*, *inference*, *reuse*, *reusing*, *semiautomatically*, *similarity*, *match*, *matching*, *complete*, *auto*;
3. **Control:** O alvo da pesquisa são as técnicas usadas;
4. **Output:** O alvo da pesquisa é descobrir como são validadas as técnicas propostas.

Após definir os quatro grupos, foi especificada a seguinte *string* de busca: ‘‘(*scientific* **and** (*workflow* **or** *pipeline*)) **and** (*recommendation* **or** *provenance* **or** *suggestion* **or** *forecast* **or** *advice* **or** *design* **or** *visualization* **or** *recommender* **or** *construct* **or** *proposal* **or** *guidance* **or** *counsel* **or** *composition* **or** *activity* **or** *shimming* **or** *inference* **or** *reuse* **or** *reusing* **or** *semiautomatically* **or** *similarity* **or** *match* **or** *matching* **or** *complete* **or** *auto*)’’.

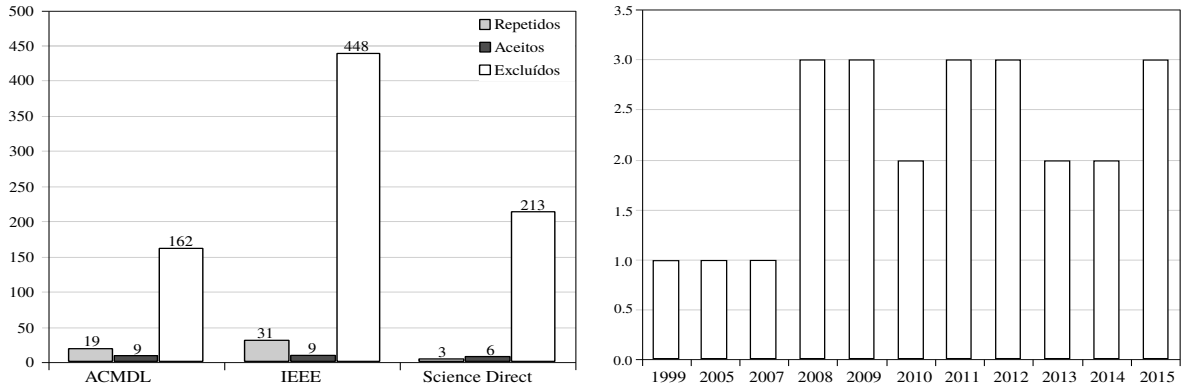
A *string* de busca foi aplicada nas bases de dados selecionadas. Em seguida, foram lidos os resumos e conclusões de todos os artigos obtidos com a pesquisa e foram aplicados o critério de inclusão: i) artigos que utilizam técnica de recomendação de atividades em *workflows*, e os de exclusão em todos os artigos: i) trabalhos não disponibilizados na integra; ii) trabalhos que não descrevem o método utilizado; e iii) trabalhos que não são da área de recomendação de atividades em *workflows*. Para classificar um artigo como ‘‘selecionado para leitura integral’’, ele deve satisfazer o critério de inclusão e não satisfazer nenhum dos critérios de exclusão.

3.1.2 Condução

Submetendo-se a *string* de busca nas bibliotecas digitais foram obtidos 838 artigos. Deste total, 171 oriundos da *ACMDL*, 448 oriundos da *IEEE* e 219 oriundos da *Science Direct*. O resumo de cada artigo foi lido e os critérios de inclusão e exclusão aplicados. Após a aplicação deles, foram selecionados para a leitura integral 24 artigos, os quais estão exibidos em duas figuras: i) a figura 23a, que resume o resultado da aplicação dos

critérios de inclusão e exclusão agrupados por base de dados; e ii) a figura 23b, que exibe a quantidade de artigos e o ano de sua publicação.

Figura 23 – Artigos obtidos com a revisão sistemática



(a) Agrupados por base de dados (b) Agrupados por ano de publicação

Fonte: Adilson Lopes Khouri, 2016

3.1.3 Extração de dados

Os dados extraídos de cada um dos artigos podem ser vistos na tabela 7, junto com suas respectivas referências.

Tabela 7 – Técnicas da literatura correlata

Referência	Técnica	Validação
(TELEA; WIJK, 1999)	Frequência	Estudo de caso
(BOMFIM et al., 2005)	Ontologia	Estudo de caso
(SHAO; KINSY; CHEN, 2007)	*Itemsets* e proveniência de execução	Estudo de caso
(KOOP, 2008)	Frequência e proveniências	*Workflows* e suas proveniências
(OLIVEIRA et al., 2008)	Proveniências	Estudo de caso
(WANG et al., 2008)	Entrada e saída de atividades	Estudo de caso
(SHAO; SUN; CHEN, 2009)	Proveniência de execução	Estudo de caso
(WANG; CAO; LI, 2009)	*Itemsets* e proveniência de execução	Estudo de caso
(ZHANG; LIU; XU, 2009)	Proveniência de modelagem	Usados *workflows* sintéticos
(LENG; EL-GAYYAR; CREMERS, 2010)	Planejador	Estudo de caso
(OLIVEIRA, 2010)	Frequência	Comparação com Koop (2008)
(CEREZO; MONTAGNAT, 2011)	Entrada/saída e semântica	Estudo de caso
(TAN et al., 2011)	Proveniência de execução	Estudo de caso
(ZHANG et al., 2011)	Frequência	Dados do *myExperiment* (ROURE, 2015)

Continua na próxima página

Tabela 7 -- Continuação da página anterior

Referência	Técnica	Validação
(CAO et al., 2012)	Proveniência de modelagem	Comparado com Zhang, Liu e Xu (2009)
(DIAMANTINI; POTENA; STORTI, 2012)	Proveniência de modelagem	Dados do *myExperiment* (ROURE, 2015)
(YAO et al., 2012)	Baseado em confiança	Dados do *myExperiment* (ROURE, 2015)
(GARIJO; CORCHO; GIL, 2013)	Frequência e proveniência de execução	Dados do SGWC *Wings* (GIL et al., 2015)
(YEO; ABIDI, 2013)	Proveniência de execução	Dados do *myExperiment* (ROURE, 2015)
(ZHANG et al., 2014)	Frequência e anotações	Interfaces do (PROGRAMMABLEWEB, 2014)
(GARIJO et al., 2014)	Frequência e Ontologia	Dados da plataforma LONI (REX; TOGA, 2003)
(OLIVEIRA et al., 2015)	*Itemsets* considerando ordem	Comparação com Koop (2008)
(MOHAN; EBRAHIMI; LU, 2015)	Entrada/saída e semântica	Dados do *myExperiment* (ROURE, 2015)
(SOOMRO; MUNIR; MCCLATCHEY, 2015)	Frequência e Ontologia	Dados da plataforma LONI (REX; TOGA, 2003)

Fonte: Adilson Lopes Khouri, 2016

3.2 Análise dos artigos selecionados pela revisão sistemática

Nesta seção são apresentadas as técnicas para recomendar atividades em *workflows* científicos, empregadas ou propostas pelos trabalhos retornados pela revisão da seção 3.1.

O sistema *Smartlink*, proposto por Telea e Wijk (1999), modela os *workflows* científicos como grafos, nos quais as arestas correspondem ao fluxo de dados e os nós às atividades. É criado um grafo das atividades mais utilizadas e elaborada uma busca em profundidade procurando por atividades. A recomendação do *Smartlink* é baseada no grafo de atividades mais utilizadas, o que permite minimizar os seguintes problemas: i) Como conectar duas atividades?; e ii) Quais atividades podem ser conectadas a uma atividade específica? A estratégia de recomendação não foi testada e comparada com a literatura, foi apenas implementada em alguns sistemas e usada.

Bomfim et al. (2005) desenvolveram um sistema de recomendação de atividades em *workflows* científicos baseado em ontologia de domínio que é utilizada para recomendar atividades em *workflows*. São comparadas as anotações da proveniência de modelagem com as do *workflow* em construção e são recomendados os *workflows* que contenham os mesmos conceitos ontológicos. A proposta não considera as dependências de dados e de controle para recomendar atividades. Além disso, a qualidade da recomendação não foi testada e os autores não detalharam a ontologia construída.

Shao, Kinsy e Chen (2007) e Shao, Sun e Chen (2009) propõem minerar a proveniência de execução para encontrar os experimentos que terminam em estado de sucesso. As execuções dos *workflows* são modeladas como grafos acíclicos dirigidos. Cada execução parte do estado inicial até atingir um dos estados finais: i) teste; ii) não finalizado; iii) irrelevante, que não auxilia a atingir o estado de sucesso; e iv) sucesso. São considerados críticos todos os caminhos que partem do estado inicial e terminam no estado sucesso. Não foram realizados experimentos sobre recomendação, apenas sobre tempo de execução para minerar a proveniência. Os autores citam que essas técnicas poderiam ser utilizadas para recomendar os *subworkflows* de sucesso para os *workflows* em construção.

Koop (2008) recomenda *subworkflows* frequentes considerando a estrutura do *workflow*. Para tal, são encontradas (utilizando a proveniência de execução) todas as sequências de atividades posteriores ao nó âncora (aquele que vem antes do item a ser recomendado). Essas atividades serão recomendadas ao usuário. Os testes utilizaram 2.875 *workflows* e suas proveniências de execução, geradas por estudantes durante um curso de visualização de dados. O conjunto de dados foi dividido em: i) treinamento, responsável por gerar caminhos; e ii) testes, responsável por usar caminhos gerados e recomendar.

Oliveira et al. (2008) propõem recomendar trechos de *workflows* baseado em filtro colaborativo sobre a proveniência de execução e de modelagem de outros *workflows*. Sempre que uma atividade é adicionada, o sistema verifica quais as atividades seguintes foram utilizadas, considerando os dados de proveniência, e recomendando-as. Esta estratégia não foi comparada com a literatura, foi apenas implementada no SGWC Vistrails (VISTRAILS, 2015) e foi apresentada uma recomendação de uma atividade.

Na área de recomendação de serviços, Wang et al. (2008) recomendam serviços baseados nos fatores dependência entre serviços, modelos existentes e ordem de execução dos serviços. Considere um modelo de *workflow* no qual um serviço ‘‘b’’ invoca um serviço ‘‘c’’ e o serviço ‘‘c’’ invoca o serviço ‘‘d’’. Em um *workflow* em construção, após a inclusão do nó ‘‘b’’, serão dadas as recomendações ‘‘c’’ e ‘‘d’’ ordenadas pela proximidade com ‘‘b’’. A técnica foi testada em dois problemas da área de bioinformática, porém não ocorreu uma comparação com a literatura correlata.

Zhang, Liu e Xu (2009) propõem uma estratégia de recomendação baseada no subgrafo anterior ao nó âncora (aquele que vem antes do item a ser recomendado) que é definido por meio da proveniência de modelagem. São recomendados os nós posteriores aos subgrafos com maior ocorrência, isto é, é gerada uma lista com todas as possibilidades

de próximos nós baseados em todos os nós anteriores ao nó âncora, de acordo com a proveniência de modelagem. Esta estratégia foi testada por meio de *workflows* e proveniências criados pelos autores.

Wang, Cao e Li (2009) recomendam atividades por meio de *itemsets* frequentes minerados a partir das mudanças ocorridas nos *workflows*. Cada mudança é denominada Δ, uma série destas transforma um *workflow* em outro e consiste na sequência: $\Delta_0, \Delta_1, \ldots, \Delta_N$. É aplicado o algoritmo *Apriori* em todas as sequências de operações Δ. Dessa forma, são obtidas as regras de associação que podem ser usadas como recomendação. A técnica é implementada em um SGWC, porém não ocorrem comparações com a literatura correlata.

Leng, El-Gayyar e Cremers (2010) propõem um planejador que procura por termos de uma ontologia. Primeiramente, os grafos acíclicos dirigidos, que representam os serviços web e suas relações, são modelados como uma quíntupla (P, P_0, G, A, Γ) onde P é o conjunto de proposições, P_0 é o estado inicial, G é o estado a ser atingido, A é o conjunto de ações que transformam uma proposição em outra e Γ é a função que transforma proposições.

No grafo, os serviços serão as ações A, as entradas e saídas de todos os serviços serão as proposições P, a entrada passada pelo usuário será o estado inicial P_0 e a saída esperada pelo usuário será o estado final a ser atingido G. O planejador começa adicionando os estados que satisfaçam as entradas das proposições existentes e que tenham uma similaridade semântica mínima, a qual é calculada por meio de graus de similaridade comparando as anotações semânticas feitas em todos os serviços com termos controlados por uma ontologia.

A similaridade entre dois conceitos, c_1 e c_2, da ontologia é calculada com as seguintes regras: i) c_1 e c_2 são equivalentes, então é denominada exata; ii) c_2 é super conceito de c_1; iii) c_1 é super conceito de c_2; iv) são inexatos. Ao término do algoritmo é encontrado um caminho entre o estado inicial e o final o qual é recomendado. O sistema foi parcialmente testado, pois ainda estava sendo implementado, porém não ocorreram comparações com a literatura correlata, apenas uma recomendação específica para um caso simples.

Oliveira (2010) recomenda atividades de *workflows* científicos utilizando mineração sequencial. Essa abordagem permite utilizar uma modificação do algoritmo *Preorder Linked WAP* (PLWAP) desenvolvido por Ezeife (2005) para recomendar atividades. São determinadas as sequências maximais (aquelas que não estão presentes em outras sequências de um mesmo *workflow*). As sequências são a entrada para o PLWAP que define os caminhos padrões (que são usados como recomendação). Foi usada uma base de

dados de 3.340 *workflows*, essa estratégia de recomendação foi comparada com a proposta por Koop (2008).

Cerezo e Montagnat (2011) elaboraram um sistema que permite construir *workflows* em alto nível, utilizando ontologias de domínio. Essa modelagem é convertida para uma linguagem que pode ser executada por sistemas gerenciadores de *workflow* científico. Durante a tradução, que é semiautomática, são recomendados para o usuário *subworkflows* que possuem entradas e saídas compatíveis sintaticamente e cuja similaridade ontológica é alta em relação ao *workflow* de alto nível modelado. Os autores não citam a estratégia de validação dos resultados, apenas elaboram uma recomendação específica para um caso simples.

Tan et al. (2011) constroem dois grafos: i) *workflows* e seus serviços, representados por nós, e as arestas representam a relação de inclusão de um serviço dentro de um *workflow*; e ii) entre operações, em que os nós representam operações dentro de serviços e as arestas operações entre serviços. Com esses grafos, é possível usar o algoritmo *Apriori* para descobrir quais serviços são utilizados em conjunto por quais usuários, e assim gerar recomendações. Não são citadas comparações com a literatura correlata, apenas uma recomendação específica para um caso simples.

Zhang et al. (2011) constroem redes sociais de: i) *workflows* e serviços; ii) serviços e serviços; e iii) pessoas e serviços. As redes permitem avaliar quais serviços são utilizados por quais *workflows*, com qual frequência e quem são os autores. O sistema foi testado com *workflows* do repositórios *myExperiment* (ROURE, 2015) com validação cruzada, são recomendados os serviços mais frequentemente utilizados em *workflows* distintos.

No contexto de processos de negócio, Cao et al. (2012) aplicam recomendação baseada em grafos durante a construção dos *workflows*. Os grafos prontos são minerados para definir padrões (sequências frequentes). Em seguida, é calculada a distância entre os padrões e o processo de negócio (*workflow*) em construção. Os padrões com menor distância em relação ao processo de negócio em construção são recomendados ao usuário. A distância é calculada utilizando uma métrica elaborada pelos autores que considera a posição do nó no modelo pronto e no subgrafo em construção. A técnica foi comparada com a proposta de Zhang, Liu e Xu (2009).

Diamantini, Potena e Storti (2012) recomendam fragmentos de *workflows*, modelados como um grafo dirigido acíclico, encontrando as menores subestruturas mais representativas de cada grafo. Para tal, empregam um algoritmo de agrupamento da biblioteca SUBDUE

que permite reduzir os nós do grafo utilizando a métrica *Minimum Description Length* (MDL)

$$MDL = \frac{DL(S) + DL(G|S)}{DL(G)} \tag{3.2.1}$$

em que $DL(S)$ é a *Description Length* (DL), que é uma função para computar o número de *bits* necessários para representar a matriz de adjacência do subgrafo S, $DL(G)$ é a *Description Length* do grafo original G e $DL(G|S)$ *Description Length* de G comprimido por S. São recomendados os padrões mais representativos ordenados pelo valor da equação (3.2.1). Para teste foi utilizado um subconjunto de 564 *workflows* do repositório *myExperiment* (ROURE, 2015) obtendo como saída uma hierarquia de agrupamentos similares que, segundo os autores, poderiam ser usados para a recomendação.

Yao et al. (2012) recomendam com base na confiabilidade de serviços e autores: a *ReputationNet* que é um sistema de recomendação de atividades para *workflows* que considera a reputação do autor (escolaridade, especialidade e número de citações) e a popularidade dos serviços (frequência relativa). Os serviços mais populares dos autores mais confiáveis são recomendados. Foram utilizados diversos *workflows* do repositório *myExperiment* (ROURE, 2015) para validar os resultados.

Garijo, Corcho e Gil (2013) mineram a proveniência de execução para encontrar fragmentos frequentes de *workflows* e recomendá-los. Os rastros de proveniência são representados como grafos dirigidos acíclicos. Dado um repositório de *workflows* e sua proveniência de execução, o objetivo é encontrar: i) conjuntos de atividades frequentes; e ii) *subworkflows* frequentes, utilizando o algoritmo de agrupamento do SUBDUE (baseado na equação MDL). Os testes foram feitos com 22 *workflows* e suas proveniências. O resultado de estruturas frequentes encontradas foi comparado com os resultados de uma pesquisa manual. A diferença entre esta proposta e a de Diamantini, Potena e Storti (2012) é que esta considera a proveniência de execução para recomendar, a outra apenas os *workflows* prontos.

Yeo e Abidi (2013) adaptam a técnica de rastro de causalidade para *workflows* científicos, originalmente esta técnica foi usada em *workflows* de negócios. Para isto, os autores armazenam a informação de fluxo dos dados junto com o grafo de causalidade para *workflows* científicos

$$G =< N, DP, L_B, L_A > \tag{3.2.2}$$

em que N é o número de atividades, DP é o conjunto de caminhos do fluxo de dados, $(A, b) \in L_B$ é o conjunto de atividades anteriores, tal que a execução de b é sempre realizada após alguma atividade do conjunto A e $(a, B) \in L_A$ é o conjunto de nós posteriores, tal que a é sempre executada antes de alguma das atividades do conjunto B.

Utilizando o rastro é possível criar um vetor de distâncias entre o nó âncora (que será alvo da recomendação) e os possíveis próximos nós, oriundos de L_A. Esse vetor booleano contém o valor *um* representando a presença de uma atividade do rastro e *zero* caso contrário. Os vetores são gerados para todos os rastros da base de dados e suas similaridades são calculadas por meio da similaridade do cosseno (DEZA; DEZA, 2009). Foram usados 12 *workflows* do repositório *myExperiment* (ROURE, 2015) modificados para receber a recomendação. A modificação consistia na remoção de uma atividade enquanto se esperava que o sistema a recomendasse.

Zhang et al. (2014) constroem uma rede social de *workflows* e serviços (os quais são os nós) e suas possíveis relações (que são as arestas) são do tipo de inclusão ou de autoria. Essa rede pode ser modelada como uma matriz Q em que $q_{i,j} = 1$ indica a inclusão do serviço i no *workflow* j ou como a matriz $S = Q^T Q$ em que $s_{i,j}$ representa o número de *workflows* nos quais os serviços (i, j) são chamados.

Quanto mais vezes dois serviços são utilizados em *workflows*, maior o grau de ligação entre os mesmos. Todos os serviços são publicados com metadados. Dessa forma, os autores calculam o *Term Frequency-Inverse Category Frequency* (TF-ICF) nos metadados que descrevem os serviços e suas categorias. Com base nos valores de TF-ICF cada serviço é classificado em k categorias. Durante a construção do *workflow* são sugeridos serviços de acordo com a métrica *Rank-Biased Overlap* (RBO), descrita no artigo. São recomendados os serviços que possuem maior RBO em relação ao *workflow* em construção. São usados serviços da plataforma ProgrammableWeb (2014) para validar os resultados.

Garijo et al. (2014) recomendam atividades baseados em algoritmos de mineração de grafos para encontrar os *subworkflows* mais frequentes e recomendá-los. A similaridade entre eles e o *workflow* em construção é dada pela métrica MDL (ver equação 3.2.1). Como parâmetros, o usuário deve estabelecer o tamanho mínimo e máximo dos *subworkflow* a serem recomendados. O algoritmo modela o problema como um grafo dirigido, em seguida calcula os fragmentos mais frequentes. Por fim, usa uma ontologia que relaciona os *subworkflows* com os *workflows*. Para validar a proposta, o sistema foi usado por

usuários que compararam os *subworkflows* gerados e sugeridos com *subworkflows* que eles procuraram manualmente.

Oliveira et al. (2015) recomendam atividades baseados em algoritmos de mineração sequencial (como Agrawal e Srikant (1994)), adaptados para considerar a ordem das atividades. A técnica pesquisa os *itemsets* com confiança e suporte mínimos estabelecidos e recomenda-os durante a construção do *workflow* científico considerando a ordem das atividades. Essa técnica foi comparada com o trabalho de Koop (2008) (em termos de consumo de memória e tempo gasto para gerar subgrafos) para sua validação.

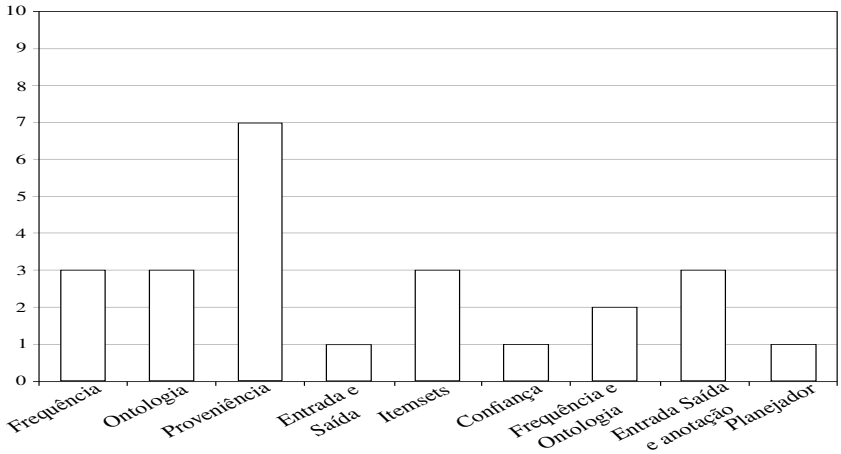
Mohan, Ebrahimi e Lu (2015) desenvolveram uma plataforma *web*, com a funcionalidade de recomendar *workflows* finalizados da base de dados, a qual permite construir *workflows* e anotá-los semanticamente usando *tags* criadas pelo usuário. Durante a construção do *workflow* o sistema pesquisa por todos os *workflows* da base que contenham entrada compatível com a saída do *workflow* em construção e analisa quais contêm as mesmas anotações semânticas que as do perfil do usuário e os recomenda. A validação ocorreu usando 9.886 usuários, 2.664 *workflows* e 9.624 *tags* do repositório *myExperiment* (ROURE, 2015).

Soomro, Munir e Mcclatchey (2015) criaram um sistema de recomendação baseado em mapeamento de atividades em conceitos ontológicos funcionais acrescido de semântica de dados oriunda de uma ontologia de domínio. Durante a construção do *workflow*, cada uma de suas atividades é mapeada em um conceito ontológico, em seguida a base de dados recomenda todos os *subworkflows* que contenham aqueles conceitos ontológicos naquela sequência ordenados pela sua frequência. Os testes foram realizados com 65 *workflows* da plataforma LONI (REX; TOGA, 2003) usando a métrica MRR. Os autores não citam detalhes da ontologia usada.

3.3 Tendências observadas com os resultados da revisão sistemática

A partir da figura 24 é possível notar a existência de uma tendência no uso de técnicas baseadas em proveniência de dados, frequência e dependência da informação. A partir de 2014 a literatura começou a considerar estratégias híbridas que usam proveniência e algum tipo de informação semântica. No ano de 2015 foram publicados dois artigos propondo estratégias híbridas para recomendar que usam frequência e algum tipo de informação semântica para recomendar *subworkflows*.

Figura 24 – Número de artigos por técnica de recomendação



Fonte: Adilson Lopes Khouri, 2016

A técnica baseada em proveniência de dados (mais utilizada na literatura) tem como vantagem considerar diversos dados históricos sobre um mesmo padrão de atividade. Por exemplo, para recomendar uma atividade em um *workflow* que contenha a atividade x, são considerados todos os *workflows* que contenham x e suas atividades posteriores, a atividade com maior frequência é recomendada. Essa abordagem permite minimizar o efeito de *outliers*. Como desvantagem, possui a necessidade de uma base de dados históricos relevantes, caso contrário, *outliers* podem afetar o desempenho.

A técnica baseada em frequência tem como vantagem a simplicidade na implementação e como principal desvantagem a necessidade de uma base de dados com pouca esparsidade no uso de atividades.

A técnica baseada em dependência de informação tem como principal vantagem a facilidade de implementação. Como desvantagem, ela não leva em consideração a semântica dos dados das atividades. Por exemplo, uma *string* que representa o nome de uma espécie de bactéria é considerada similar a uma *string* que representa um CEP.

Outra tendência observada é sobre a validação dos resultados. Não há uma metodologia amplamente utilizada entre os trabalhos analisados para validação. Muitos autores apenas executam a solução uma vez para ‘‘mostrar’’ que sua solução funciona. Não ocorrem testes com dados sintéticos ou reais, o que pode ser verificado na tabela 7 em que 11 artigos estão nessa situação (marcados na tabela como ‘‘Elaborado um estudo de caso’’).

3.4 Comparação da solução proposta com os trabalhos correlatos

Nesta seção são comparadas as técnicas descritas com o projeto deste mestrado em relação as restrições típicas de sistemas de recomendação de *workflows* científicos, que foram detalhadas na seção 2.3 do capítulo de conceitos fundamentais.

Os trabalhos de Shao, Kinsy e Chen (2007) e Shao, Sun e Chen (2009), que consideram a mineração sequencial de atividades como *itemsets* desconsideram a ordem das atividades e a semântica das mesmas. A proposta de Oliveira et al. (2015) desconsidera apenas a semântica das atividades. Esta proposta de mestrado considera a ordem de atividades que é um fator importante na recomendação conforme visto no capítulo de conceitos fundamentais.

Os trabalhos de Koop (2008), Oliveira et al. (2008), Wang, Cao e Li (2009), Zhang, Liu e Xu (2009), Tan et al. (2011), Cao et al. (2012), Diamantini, Potena e Storti (2012), Garijo, Corcho e Gil (2013), Yeo e Abidi (2013) consideram a ordem das atividades, entrada e saída e proveniência dos dados. Suas limitações são a necessidade de dados de proveniência, pois nem todo SGWC armazena essas informações, além de desconsiderar informação semântica dos *workflows* e atividades. Este projeto não necessita de informações de proveniência e considera a semântica da informação por meio de uma ontologia hierarquizada e validada por um especialista da área.

O trabalho de Bomfim et al. (2005) usa apenas um mapeamento entre atividades e ontologia desconsiderando a entrada e saída, o que potencialmente gera recomendações ineficientes. Neste projeto são consideradas às entradas e saídas de cada atividade individualmente, além do uso de uma ontologia de domínio.

Wang et al. (2008), Leng, El-Gayyar e Cremers (2010) desconsideram o uso de semântica das atividades e da frequência de suas ocorrências em pares. Nesse projeto de mestrado são considerados esses dois fatores.

O trabalho de Yao et al. (2012) exige dados que permitam calcular a confiança dos usuários e dos seus *workflows*. Repositórios como *myExperiment* (ROURE, 2015) não exigem dos usuários o preenchimento de todos os seus dados, de forma que grande parte das informações relacionadas a este aspecto não são preenchidas pelos usuários. Além disso, os autores desconsideram a semântica das atividades e *workflows*. Este projeto de mestrado considera a semântica de *workflows* e não necessita da informação sobre a confiança dos usuários.

Os trabalhos de Telea e Wijk (1999), Oliveira (2010) e Zhang et al. (2011) desconsideram o uso de semântica de dados para recomendar, o que é um limitante conforme discutido por Garijo et al. (2014), Soomro, Munir e Mcclatchey (2015). No presente mestrado, a frequência é considerada em conjunto com a ontologia de domínio.

Os trabalhos de Zhang et al. (2014), Mohan, Ebrahimi e Lu (2015), Cerezo e Montagnat (2011) desconsideram o uso de uma ontologia hierarquizada e validada por um especialista. Dessa forma, a qualidade das anotações semânticas é questionável. Nesse projeto foi construída uma ontologia usando uma metodologia e esta foi validada por um especialista.

Os trabalhos de Garijo et al. (2014), Soomro, Munir e Mcclatchey (2015) consideram o uso de frequência e ontologia, como neste projeto, porém recomendam *subworkflows* o que limita as recomendações de atividades. Apenas atividades usadas em fragmentos comuns de *workflows* poderão ser recomendadas. Em outras palavras, se a atividade se encontra no ‘‘meio’’ de um *subworkflow* esta nunca poderá ser recomendada individualmente. No presente mestrado, todas as atividades tem possibilidade de ser recomendadas, mesmo que no final da lista de recomendação. Além disso, apresenta uma recomendação mais abrangente, pois trata o caso de atividades simples, *subworkflows* e *Shims* (ver seção 2.1.1).

Neste mestrado o problema de recomendação de atividades foi também modelado como um problema de classificação e regressão, usando para isso 5 classificadores; 5 regressores; um classificador SVM composto (que usa o resultado dos outros classificadores e regressores para recomendar) e um *ensemble* de classificadores (*Rotation Forest*).

4 Solução proposta

''Poder é sempre perigoso. Atrai o pior e corrompe o melhor. Nunca pedi por poder. Poder só é dado para aqueles que estão dispostos a abrir mão de si por ele.''

(Ragnar Lothbrok)

Este capítulo descreve a solução proposta para o problema de recomendar atividades em workflows científicos, que utiliza uma ontologia de domínio e frequência de atividades para recomendar atividades para cientistas durante a construção de *workflows* científicos. Também são descritas as soluções propostas para o uso de classificadores e regressores para resolver o problema de recomendação de atividades, bem como o uso de um classificador composto e um *ensemble* de classificadores.

Primeiramente será detalhado como a ontologia foi construída, qual a metodologia foi usada e o processo de sua construção. Em seguida será detalhado o conjunto de dados, como foram obtidos, sua organização em um modelo relacional e como ocorreu a modelagem desses dados para solucionar o problema de recomendação. O próximo passo é explicar as alterações necessárias nos dados para serem usados por classificadores e regressores. Em seguida, será explicada a estratégia de validação dos experimentos. Por fim, na última seção do capítulo, será detalhada a solução proposta que utiliza a ontologia construída.

4.1 Desenvolvimento da ontologia

A ontologia foi desenvolvida usando a metodologia *Skeletal* (USCHOLD; KING, 1995), que contém as seguintes fases:

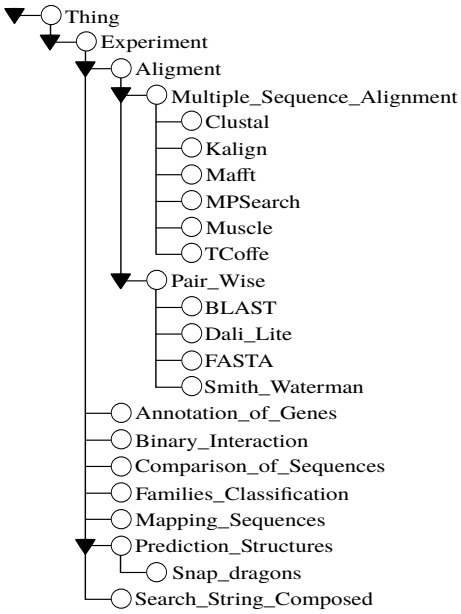
1. Identificar a finalidade;
2. Construção da ontologia:
    a) Captura da ontologia;
    b) Codificação da ontologia;
    c) Integração com ontologias existentes;
3. Validação;
4. Documentação.

A primeira fase, denominada *Identificar a finalidade*, define o objetivo para o qual será construída a ontologia e como ela será utilizada futuramente. Neste projeto, a ontologia foi construída para agregar conhecimento semântico durante a recomendação de atividades, para tal, todos os *workflows* de bioinformática foram anotados com os conceitos desta ontologia. A segunda fase, chamada de *Construção da ontologia*, tem como objetivo construir a ontologia (usando uma linguagem formal) em três subfases: i) *Captura da ontologia*; ii) *Codificação da ontologia*; e iii) *Integração com ontologias existentes*.

Na primeira subfase são identificados os conceitos e suas relações no domínio de aplicação. Para realizar esta subfase foi necessário estudar a área de alinhamento de sequências genômicas com os seguintes materiais de estudo: i) um livro (SETUBAL; MEIDANIS, 1997); e ii) quatro cursos online, disponibilizados pela universidade de São Diego, criados por Pevzner e Compeau (2015a, 2015b, 2015c, 2015d).

A codificação da ontologia, realizada na segunda subfase, usou a ferramenta *Protégé* (PROTEGE, 2014) por ter código aberto, ser muito conhecida na área de ontologias e permitir a utilização da linguagem formal de ontologias OWL (MCGUINNESS; HARMELEN, 2015). A terceira etapa, denominada *Integração com ontologias existentes*, não ocorreu neste projeto pois não foram encontradas ontologias usadas para recomendar atividades em *workflows* científicos na área de bioinformática.

Figura 25 – Ontologia construída utilizando a metodologia *Skeletal*



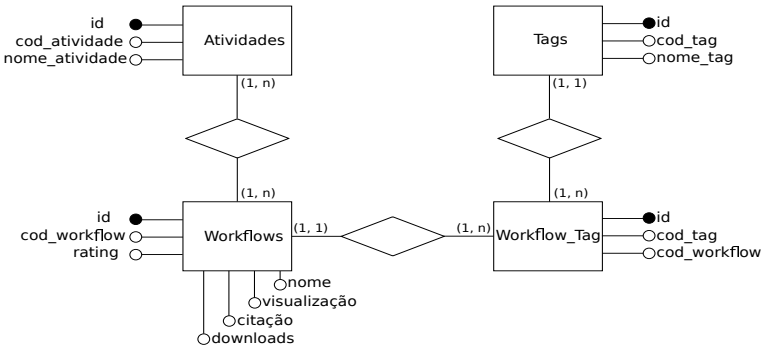
Fonte: Adilson Lopes Khouri, 2016

A ontologia construída pode ser visualizada na figura 25, na qual os círculos são os conceitos do domínio e as linhas de ligação, são as relações entre conceitos. A relação que foi utilizada é a "*é subtipo de*". Ao término da fase de construção, foi realizada a fase de validação da ontologia, realizada por um especialista no domínio de bioinformática. Que é orientador deste projeto, tem formação específica em bioinformática e *workflows* científicos. A documentação da ontologia, que é a última etapa, foi realizada nesta seção da dissertação, na qual foram detalhados os motivos de sua construção, sua motivação, seus usos e a forma de sua validação.

4.2 Modelagem dos dados

Os *workflows* foram obtidos no repositório *myExperiment* (ROURE, 2015), por meio do *software wget* (SCRIVANO; NIKSIC, 2015). Após efetuar o *download* dos 2481 *workflows* em formato *xml*, foi utilizado o analisador de código *Beautiful Soup* (RICHARDSON, 2015), para organizar o conjunto de dados em uma base de dados relacional[1] (ver figura 26). Neste modelo conceitual os retângulos representam as entidades, os losangos representam

Figura 26 – Modelo de dados dos *workflows* científicos



Fonte: Adilson Lopes Khouri, 2016

a relação entre atividades, os círculos brancos representam os atributos das entidades, os círculos pretos representam identificadores e os números próximos a cada entidade representam sua cardinalidade. Este modelo armazena todas as atividades dos *workflows* (de diversas áreas) usando as entidades *Atividades* e *Workflow*. Para registrar qual a área científica (domínio de aplicação) de cada *workflow* foram utilizadas as tabelas *Workflow_Tag* e *Tag*.

[1] www.each.usp.br/digiampietri/baseworkflows/SQL.tar.gz

Os *workflows* da área de bioinformática (totalizando 73) em conjunto com suas atividades (totalizando 280) foram convertidos em uma matriz $M_{i,j}$ em que cada linha i representa um *workflow*, cada coluna j representa uma das 280 atividades e cada célula da matriz M representa a existência $M_{i,j} = 1$, ou não $M_{i,j} = 0$, da atividade da coluna j no *workflow* i. A tabela 8 apresenta um exemplo, fictício, de matriz M. Para a realização dos testes, para cada linha da tabela 8 é removida uma atividade e é recomendada uma lista de possíveis atividades. O objetivo do sistema de recomendação é identificar corretamente qual a atividade está faltando no workflow (isto é, aquela que foi removida).

Tabela 8 – Exemplo de matriz de entrada.

Workflow	**Ativ 01**	**Ativ 02**	...	**Ativ 280**
01	1	0	...	0
02	1	1	...	1
03	1	0	...	1
⋮	⋮	⋮	⋮	⋮
73	1	0	...	0

Fonte: Adilson Lopes Khouri, 2016

4.3 Modelagem dos dados como problema de classificação e regressão

Para usar técnicas de classificação e regressão foram propostas algumas alterações no conjunto de dados original, descrito na tabela 8, as quais podem ser visualizadas na tabela 9. Cada *workflow* foi replicado 118 vezes. Destes, 59 são uma cópia idêntica ao original, enquanto que dos outros 59 foi removida uma mesma atividade para todos os *workflows*, e foi adicionada uma nova atividade representando uma possível recomendação. Dessa forma, para cada *workflow* original haverá 59 instâncias corretas e 59 instâncias incorretas e este tipo de informação será utilizada para treinar os classificadores ou regressores.

A escolha de 59 atividades a serem recomendadas foi feita por duas razões. A primeira é selecionar as 59 atividades com maior frequência na base de dados. A segunda é a limitação computacional: replicar as 280 possíveis recomendações poderia ser inviável em termos de treinamento. Foram replicadas 59 instâncias de *workflows* idênticas consideradas corretas, isto é com a atividade correta não removida, para garantir o balanceamento entre classes. A última alteração foi adicionar uma coluna indicando se a recomendação da atividade proposta é a correta, isto é, a pertencente ao respectivo *workflow* (T) ou não (F).

Tabela 9 – Exemplo de matriz de entrada para técnicas de classificação e regressão

#	***Workflow***	**Ativ 01**	**Ativ 02**	...	**Ativ 279**	**Ativ 280**	**Rótulo**
1	01	1	0	...	0	0	T
2	01	1	0	...	0	0	T
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	01	1	0	...	0	0	T
1	01	0 (removida)	1 (adicionada)	...	1	0	F
2	01	0 (removida)	0	...	1 (adicionada)	0	F
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	01	0 (removida)	0	...	0	1 (adicionada)	F
	⋮						
1	73	1	1	...	0	0	T
2	73	1	1	...	0	0	T
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	73	1	1	...	0	0	T
1	73	1 (adicionada)	0 (removida)	...	1	0	F
2	73	1	0 (removida)	...	1 (adicionada)	0	F
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
59	73	1	0 (removida)	...	0	1 (adicionada)	F

Fonte: Adilson Lopes Khouri, 2016

Na primeira modelagem, descrita na tabela 8, cada linha (instância) recebe uma lista de atividades recomendadas pelas técnicas da literatura correlata e pela técnica proposta nesse mestrado (seção 4.5). Cada lista retornada segue algum critério de ordenação referente à técnica usada. Por exemplo, uma técnica baseada em frequência retorna uma lista de atividades ordenadas pelas suas frequências.

Na segunda modelagem, cada linha classificada como *não pertencente* (F) ao *workflow* é automaticamente adicionada no final da lista de recomendação. As outras atividades (T) são adicionadas ao início da lista e ordenadas, de acordo com suas frequências, anotações ontológicas, ordem alfabética e seletor aleatório, nesta sequência de ordenações estáveis.

Esta modelagem foi utilizada pelos classificadores e regressores descritos no capítulo 2. Adicionalmente, os resultados desses classificadores e regressores foram utilizados pelo classificador composto e pelo *ensemble* de classificadores. Essas estratégias que utilizam classificadores e regressores foram desenvolvidos de forma a não necessitarem de anotações semânticas dos *workflows*.

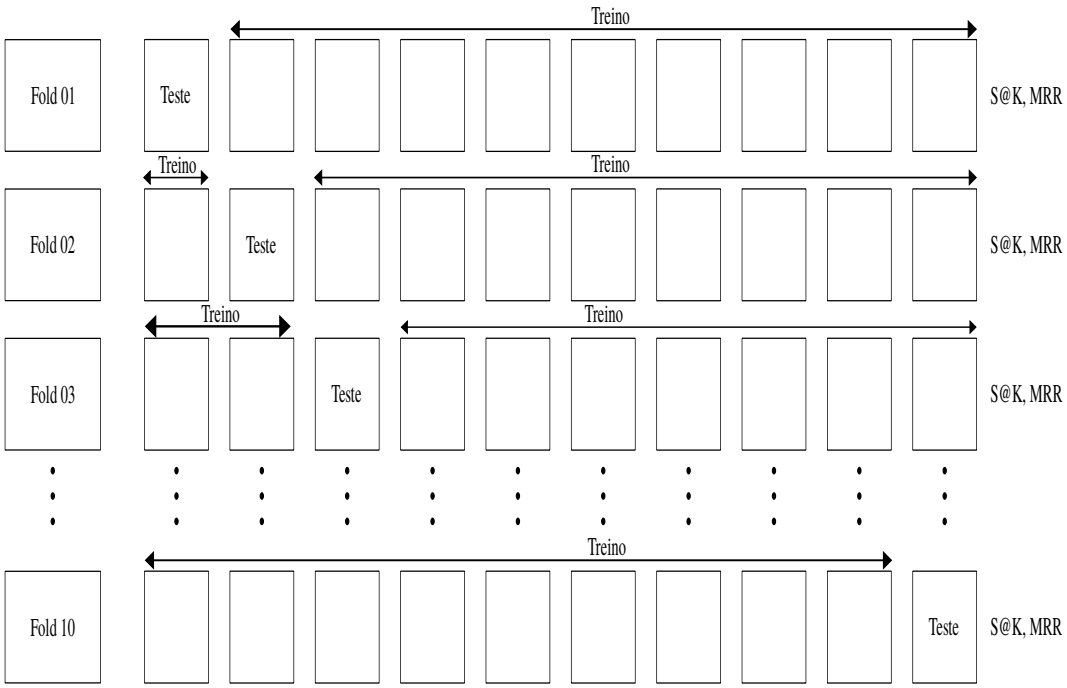
Em ambas as modelagens, após construir a lista de recomendação oficial, aquela com os itens recomendados pelas técnicas, os sistemas de recomendação adicionam todas as outras atividades (que não foram recomendadas), ordenadas alfabeticamente, no final da lista. Dessa forma, todas as possíveis atividades estarão presentes na lista. Portanto as métricas descritas na seção 4.4 sempre poderão ser calculadas.

4.4 Estratégia de validação dos sistemas de recomendação

Para a validação será utilizada a técnica cruzada considerando 10 subconjuntos (10-*fold cross validation*). Nessa técnica, o conjunto de dados é dividido em 10 subconjuntos (*folds*) e são realizadas dez execuções. Em cada uma, 10% dos *workflows* são separados para teste e 90% para treinamento. Assim, para cada execução, o sistema treina com 90% dos dados e o resultado do treinamento é testado para os 10% restantes.

Deve-se ressaltar que 100% do conjunto de dados é rotulado (isto é, fica explícito ao sistema qual atividade foi removida) e assim é possível verificar o desempenho de cada uma das execuções. O teste apresenta os 10% de *workflows*, sem informar os rótulos (a atividade removida), para os sistemas de recomendação que já foram treinados. Ao término das dez execuções são calculadas as médias das métricas: i) *Sucess at rank k* ($S@k$); e ii) *Mean Reciprocal Rank* (MRR). A figura 27 ilustra o processo de separação entre conjunto de treinamento e teste utilizado na estratégia de validação empregada.

Figura 27 – Exemplo de 10-*fold cross validation*



Fonte: Adaptado de Han, Kamber e Pei (2011)

A métrica $S@k$ calcula a probabilidade de um item de interesse estar localizado entre as k primeiras posições da lista de atividades recomendadas. Seus valores residem entre zero e um. Os resultados dessa métrica são cumulativos para valores crescentes de k, isto ocorre pois se uma atividade de interesse estiver entre as cinco primeiras posições da

lista de recomendações, ela também encontra-se entre as dez primeiras posições. No limite, a atividade sempre estará entre as L primeiras posições, sendo L o tamanho total da lista de recomendações. Assim, valores elevados para $S@k$ são considerados bons, especialmente para valores baixos de k. O cálculo dessas métricas é detalhado pelas equações (HARVEY; RUTHVEN; CARMAN, 2010):

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{n_i}\right) \tag{4.4.1}$$

$$S@k = \frac{1}{N}\sum_{i=1}^{N}(I(n_i \leq k)) \tag{4.4.2}$$

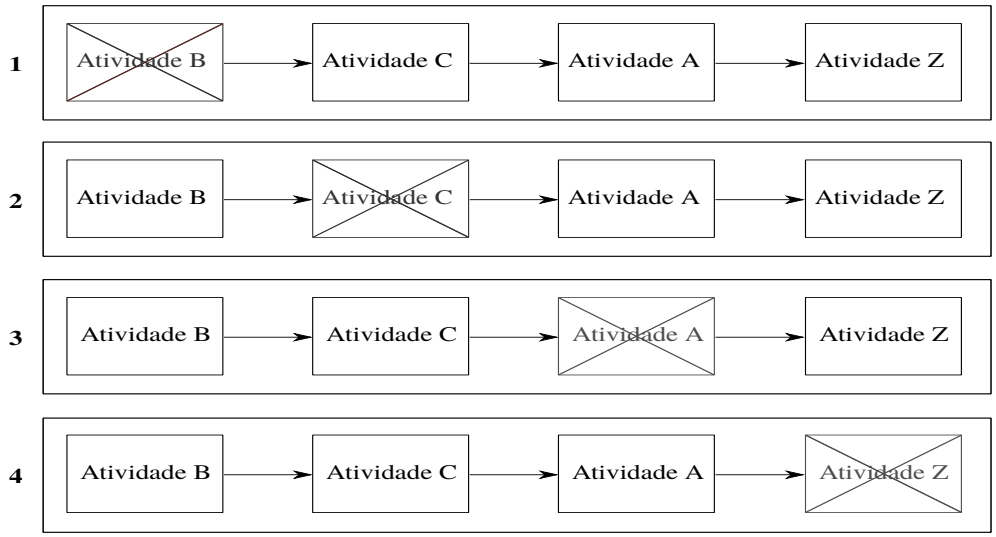
em que N é o número de listas recomendadas, n_i é a posição do item desejado na lista de recomendações i, k é uma posição da lista determinada como parâmetro de entrada da equação (4.4.2) e a função I, indica se a atividade n_i ocorre em uma posição (x) menor ou igual ao parâmetro de entrada k, e é dada por

$$I(x,k) = \begin{cases} 1 \text{ se } x \leq k \\ 0 \text{ caso contrário} \end{cases} \tag{4.4.3}$$

Para exemplificar o uso destas métricas será utilizado um *workflow* fictício representado na figura 28 que teve quatro atividades removidas (**B**, **C**, **A** e **Z**) uma a uma gerando quatro casos que necessitam de recomendações ($\mathbf{1}, \mathbf{2}, \mathbf{3}$ e $\mathbf{4}$). Que foram usados como entradas para quatro sistemas de recomendação distintos. Cada sistema produziu quatro listas de recomendação, uma para cada caso da figura 28, rotulados com a mesma numeração. Assim, a lista 01 é a recomendação correspondente do caso 1 e assim sucessivamente.

As tabelas 10a, 10b, 10c e 10d apresentam os resultados dos quatro sistemas de recomendação. Cada item em negrito das listas representa a atividade que foi removida do *workflow*, que é o item considerado correto (atividades com X na figura 28). Sua posição é determinada na coluna *Rank* dessas tabelas. O sistema de recomendação 01,

Figura 28 – Quatro casos de recomendação de atividades



Fonte: Adilson Lopes Khouri, 2016

Tabela 10 – Exemplo de sistemas de recomendações de atividades

Rank	Lista 01	Lista 02	Lista 03	Lista 04
1	Ativ_A	Ativ_H	Ativ_Z	Ativ_I
2	**Ativ_B**	Ativ_B	Ativ_E	Ativ_C
3	Ativ_C	Ativ_I	**Ativ_A**	Ativ_Z
4	Ativ_D	Ativ_X	Ativ_X	Ativ_B
5	Ativ_E	Ativ_A	Ativ_C	Ativ_S
6	Ativ_F	Ativ_D	Ativ_D	Ativ_N
7	Ativ_G	**Ativ_C**	Ativ_Q	Ativ_K
8	Ativ_H	Ativ_Z	Ativ_B	**Ativ_Z**
9	Ativ_I	Ativ_P	Ativ_F	Ativ_H
10	Ativ_Z	Ativ_F	Ativ_K	Ativ_A

(a) Sistema de recomendação 01

Rank	Lista 01	Lista 02	Lista 03	Lista 04
1	Ativ_A	**Ativ_C**	**Ativ_A**	Ativ_I
2	**Ativ_B**	Ativ_B	Ativ_E	Ativ_C
3	Ativ_C	Ativ_I	Ativ_Z	Ativ_A
4	Ativ_D	Ativ_X	Ativ_X	**Ativ_Z**
5	Ativ_E	Ativ_A	Ativ_C	Ativ_S
6	Ativ_F	Ativ_D	Ativ_D	Ativ_N
7	Ativ_G	Ativ_H	Ativ_Q	Ativ_K
8	Ativ_H	Ativ_Z	Ativ_B	Ativ_U
9	Ativ_I	Ativ_P	Ativ_F	Ativ_H
10	Ativ_Z	Ativ_F	Ativ_K	Ativ_X

(b) Sistema de recomendação 02

Rank	Lista 01	Lista 02	Lista 03	Lista 04
1	Ativ_A	Ativ_H	Ativ_F	Ativ_I
2	Ativ_G	Ativ_B	Ativ_E	Ativ_C
3	Ativ_C	Ativ_I	Ativ_Z	Ativ_A
4	Ativ_D	Ativ_X	Ativ_X	Ativ_B
5	Ativ_E	Ativ_A	Ativ_C	Ativ_S
6	Ativ_F	Ativ_D	Ativ_D	Ativ_N
7	**Ativ_B**	Ativ_Q	Ativ_Q	**Ativ_Z**
8	Ativ_H	Ativ_Z	Ativ_B	Ativ_U
9	Ativ_I	Ativ_P	**Ativ_A**	Ativ_H
10	Ativ_Z	**Ativ_C**	Ativ_K	Ativ_X

(c) Sistema de recomendação 03

Rank	Lista 01	Lista 02	Lista 03	Lista 04
1	Ativ_A	Ativ_H	Ativ_Z	**Ativ_Z**
2	Ativ_Z	**Ativ_C**	Ativ_E	Ativ_C
3	Ativ_C	Ativ_I	Ativ_Z	Ativ_A
4	Ativ_D	Ativ_X	Ativ_X	Ativ_B
5	Ativ_E	Ativ_A	Ativ_C	Ativ_X
6	Ativ_F	Ativ_D	Ativ_D	Ativ_N
7	Ativ_G	Ativ_C	Ativ_Q	Ativ_K
8	Ativ_H	Ativ_Z	Ativ_B	Ativ_U
9	Ativ_I	Ativ_P	**Ativ_A**	Ativ_H
10	**Ativ_B**	Ativ_F	Ativ_K	Ativ_S

(d) Sistema de recomendação 04

Fonte: Adilson Lopes Khouri, 2016

cujos resultados se encontram na tabela 10a, apresenta os seguintes valores de $S@k$

$$S@1 = \frac{1}{4}\Big((I_{L_1} = 0) + (I_{L_2} = 0) + (I_{L_3} = 0) + (I_{L_4} = 0)\Big) = 0,00 \tag{4.4.4}$$

$$S@3 = \frac{1}{4}\Big((I_{L_1} = 1) + (I_{L_2} = 0) + (I_{L_3} = 1) + (I_{L_4} = 0)\Big) = 0,50 \tag{4.4.5}$$

$$S@5 = \frac{1}{4}\Big((I_{L_1} = 1) + (I_{L_2} = 0) + (I_{L_3} = 1) + (I_{L_4} = 0)\Big) = 0,50 \tag{4.4.6}$$

$$S@7 = \frac{1}{4}\Big((I_{L_1} = 1) + (I_{L_2} = 1) + (I_{L_3} = 1) + (I_{L_4} = 0)\Big) = 0,75 \tag{4.4.7}$$

$$S@10 = \frac{1}{4}\Big((I_{L_1} = 1) + (I_{L_2} = 1) + (I_{L_3} = 1) + (I_{L_4} = 1)\Big) = 1,00 \tag{4.4.8}$$

sendo I_{L_1} o resultado da função indicadora (4.4.3). As posições das atividades recomendadas por sistema são

$$\left(\frac{1}{n_i}\right)_{L_1} = \frac{1}{2} \tag{4.4.9}$$

$$\left(\frac{1}{n_i}\right)_{L_2} = \frac{1}{7} \tag{4.4.10}$$

$$\left(\frac{1}{n_i}\right)_{L_3} = \frac{1}{3} \tag{4.4.11}$$

$$\left(\frac{1}{n_i}\right)_{L_4} = \frac{1}{8} \tag{4.4.12}$$

e cuja média é dada por

$$MRR = \frac{\left(\frac{1}{2} + \frac{1}{7} + \frac{1}{3} + \frac{1}{8}\right)}{4} = 0,275297 \tag{4.4.13}$$

De forma análoga os resultados obtidos pelos sistemas 02, 03 e 04 podem ser visualizados na tabela 11.

Tabela 11 – Lista de recomendação ordenada por frequência

Sistema	**S@1**	**S@3**	**S@5**	**S@7**	**S@10**	**MRR**
1	0,00	0,50	0,50	0,75	1,00	0,275297
2	0,50	0,75	1,00	1,00	1,00	0,687500
3	0,00	0,00	0,00	0,50	1,00	0,124206
4	0,25	0,50	0,50	0,50	1,00	0,427777

Fonte: Adilson Lopes Khouri, 2016

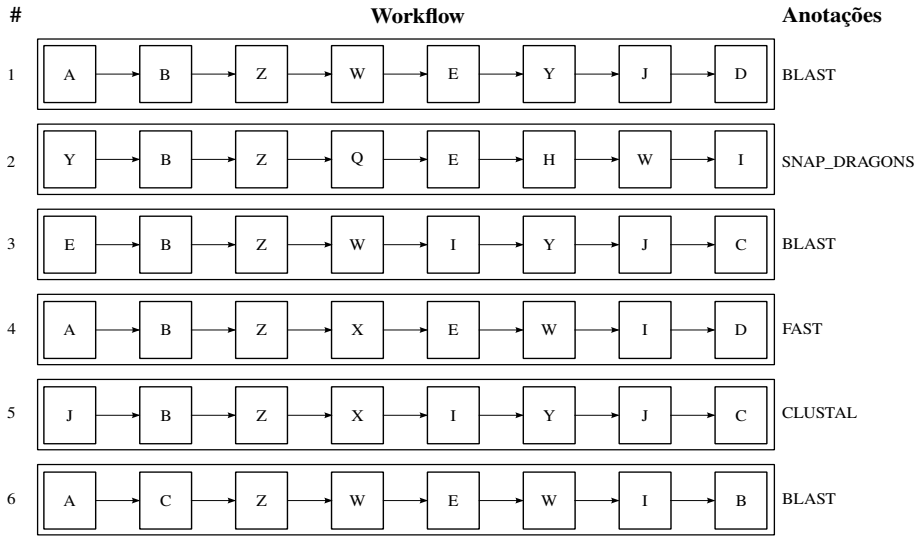
Ao comparar os quatro sistemas é possível constatar que, de acordo com as métricas calculadas, o melhor é o número 02 pois apresenta o maior valor de $MRR = 0,687500$ e os maiores valores observados de $S@1 = 0,50$ e $S@3 = 0,75$.

4.5 Solução híbrida de frequência e ontologia de domínio

Umaa outra solução proposta neste mestrado recomenda atividades usando três conceitos importantes na área de *workflows* científicos: i) frequência de atividades; ii) compatibilidade entre entrada e saída; e ii) semântica de atividades. Para explicar esta proposta, será usada a figura 29 como exemplo. Nela é possível observar seis *workflows* com suas anotações, que simulam uma base de dados de *workflows* científicos.

A solução proposta começa calculando a frequência de ocorrência de cada par de atividades existentes, que é o número de vezes que uma atividade W ocorre imediatamente

Figura 29 – Exemplo de banco de dados de *workflows* científicos



Fonte: Adilson Lopes Khouri, 2016

após uma outra atividade Z. Ao considerar somente atividades que já foram conectadas, previamente na base de *workflows*, a compatibilidade de entrada e saída é garantida por consequência.

Após calcular a frequência é necessário anotar todos os *workflows* da figura 29, usando os conceitos da ontologia construída (ver figura 25). Essa etapa é feita manualmente (de forma não automatizada). Por fim, o algoritmo anota todas as atividades com as mesmas anotações de seus respectivos *workflows*; isto é, se a atividade X (da figura 29) está dentro de dois *workflows* com anotações distintas então esta atividade receberá duas anotações. O resultado final é a tabela 12, que apresenta as frequências e anotações de atividades, nesse ponto o sistema está treinado e pronto para uso do cientista

Para compreender o mecanismo de recomendação treinado será usado outro exemplo, cujo objetivo é simular a interação do usuário com o sistema de recomendação. Suponha que durante a construção do *workflow* 1 (ver figura 25) um cientista insira a atividade Z e solicite uma recomendação. O sistema vai procurar na lista das atividades posteriores a Z ordenadas por frequência e conceito ontológico e irá retornar a lista de recomendação apresentada na tabela 12. A ordenação por conceito ontológico, além de ser estável serve como critério de desempate, quando duas atividades tiverem a mesma frequência. Neste exemplo, de acordo com a lista de recomendação da tabela 12, a atividade W seria recomendada em primeiro lugar ao cientista, o que representa um acerto.

As atividades são anotadas com a mesma anotação dos *workflows* que as contém. Dessa forma, é possível que haja pelo menos uma atividade com mais de uma anotação. Isso

Tabela 12 – Recomendação para a atividade Z ordenada por frequência e conceito ontológico

Posição na Lista	**Ativ**	**Frequência**	**Anotação Atividade**
1	W	3	BLAST
2	X	2	FAST, CLUSTAL
3	Q	1	SNAP DRAGONS
⋮	⋮	⋮	⋮
280	⋮	⋮	⋮

Fonte: Adilson Lopes Khouri, 2016

gera um novo caso de recomendação a ser considerado. Suponha que ambas as atividades W e X contenham dentro de suas listas de anotação o conceito *BLAST*. Nesse caso, seria recomendada a atividade com menor número de anotações, por ser considerada mais específica para o experimento em questão. Caso ambas as atividades tenham o mesmo número de anotações, é utilizada a ordem alfabética de conceitos como critério de desempate. Se ocorrer um novo empate é usado um seletor aleatório.

5 Resultados e discussão

‘‘Tudo o que temos de decidir é o que fazer com o tempo que nos é dado.’’(Gandalf)

Este capítulo compara resultados das técnicas propostas neste mestrado, incluindo as baseadas unicamente em classificadores e regressores e com os classificadores compostos (descritos na seção 2.8) com as da literatura correlata. Para tal, foram usadas as métricas MRR e $S@K$, descritas na seção 4.4 e validação cruzada em 10 subconjuntos (10-*fold cross validation*).

A técnica baseada na Frequência em conjunto com Entrada e Saída e Ontologia (FESO) é a contribuição principal deste mestrado. As técnicas baseadas em: i) aleatoriedade, ii) *Apriori* e iii) frequência em conjunto com entrada e saída (FES), foram propostas pela literatura correlata (ver capítulo 3).

Outra contribuição é a modelagem da recomendação de atividades como um problema de classificação e regressão. Para implementá-la, foram usados os seguintes classificadores: i) CART_C; ii) Naive Bayes_C; iii) Rede neural_C; iv) KNN_C; e v) SVM_C, e os seguintes regressores: i) Binomial_R; ii) MARS_R; iii) Rede neural_R; iv) CART_R; e v) SVM_R, descritos no capítulo 2. Também foram usados: i) um classificador composto (SVM); e ii) um *ensemble* de classificadores (*Rotation Forest*).

5.1 Comparação de resultados

A tabela 13 exibe os resultados de cada sistema recomendador usado. As técnicas que possuem a letra C em subscrito são classificadores; as que possuem letra R em subscrito são regressores; e as que não tem nada são da literatura correlata. Cada sistema efetua suas recomendações de acordo com seus diferentes critérios em uma lista inicial. Em seguida, as atividades não recomendadas são acrescentadas ao final da lista inicial. Dessa forma, a atividade correta sempre será encontrada, e o fator que diferencia os sistemas de recomendação é a posição em que as atividades ocupam na lista de atividades final que contém 280 posições.

O sistema baseado em *Aleatoriedade* não precisou de treinamento. O algoritmo apenas selecionava aleatoriamente as atividades formando uma lista de atividades recomendadas. Esse sistema recomendou menos de 3% das atividades corretas entre as

Tabela 13 – Resultados dos sistemas de recomendação

#	**Técnica**	**S@1**	**S@5**	**S@10**	**S@50**	**S@100**	**S@280**	**MRR**
1	Aleatório	0,0037	0,0260	0,0280	0,0300	0,0400	1,0000	0.033
2	*Apriori*	0,0037	0,0385	0,0559	0,0568	0,0570	1,0000	0,037
3	KNN_C	0,0037	0,0685	0,0959	0,5068	1,0000	1,0000	0,040
4	Rede neural_C	0,0137	0,1507	0,1781	0,8082	1,0000	1,0000	0,089
5	CART_C	0,0274	0,1233	0,3699	0,7671	1,0000	1,0000	0,113
6	CART_R	0,1370	0,1370	0,2603	0,6164	1,0000	1,0000	0,114
7	Naive Bayes_C	0,0274	0,1507	0,3425	0,6301	1,0000	1,0000	0,114
8	Binomial_R	0,0822	0,1918	0,2055	0,8493	1,0000	1,0000	0,136
9	Rede neural_R	0,1096	0,2603	0,2603	0,2603	1,0000	1,0000	0,154
10	MARS_R	0,1233	0,2055	0,2192	0,7260	1,0000	1,0000	0,167
11	SVM_R	0,1233	0,3151	0,4932	0,8493	1,0000	1,0000	0,238
12	FES	0,1474	0,2603	0,3699	0,8671	1,0000	1,0000	0,196
13	SVM_C	0,2425	0,4658	0,4932	0,7123	1,0000	1,0000	0,244
14	SVM composto_C	0,2515	0,4458	0,5232	0,7623	1,0000	1,0000	0,314
15	Rotation Forest_C	0,2925	0,4558	0,5432	0,7723	1,0000	1,0000	0,324
16	FESO	0,3425	0,4658	0,5932	0,8123	1,0000	1,0000	0,334

Fonte: Adilson Lopes Khouri, 2016

dez primeiras posições. A maioria das atividades corretas foram classificadas próximas a posição 140 que é a posição média das listas recomendadas. Os valores das métricas $S@280 = 1$ e $S@100 = 0,0400$ indicam que a maior parte dos itens corretos foi encontrado após a centésima posição. Esse sistema foi proposto como um marco de comparação.

O sistema que usa a técnica *Apriori* obteve seu melhor desempenho quando os parâmetros *confiança* e *suporte* foram definidos como *sem limitação*, isto é, não foi estabelecido um valor de confiança ou suporte mínimo para considerar possíveis regras de associação criadas. Todas as regras foram consideradas válidas. Mesmo sem restringir esses valores, os resultados desse sistema foram superiores apenas ao sistema baseado em Aleatoriedade. Recomendando menos de 6% das atividades corretas entre as 50 primeiras posições, sua precisão ainda é baixa com valor de $MRR = 0,037$. Os baixos resultados dessa técnica acontecem devido ao fato de desconsiderar a ordem das atividades durante a geração das regras e, consequentemente, da recomendação.

O sistema baseado em *KNN* foi treinado para diferentes valores do parâmetro $k = [1 : 100]$ que representa o número de vizinhos mais próximos (de acordo com a distância Euclidiana) que serão considerados para classificar. Este sistema apresentou os melhores resultados de recomendação para o valor de $k = 2$. Mesmo assim, menos de 10% dos itens corretos foram encontrados entre as dez primeiras posições da lista e 50%

dos itens entre os 50 primeiros itens. De acordo com a métrica MRR, a posição média dos itens recomendados foi distante da primeira posição da lista $MRR = 0,040$. Esses resultados indicam que classificar atividades de acordo com a distância entre grupos de vizinhos próximos não é uma abordagem adequada para o problema.

O sistema que usa uma rede neural MLP como classificador teve uma melhoria de quase quatro vezes na métrica $S@1$ de $0,0037$ para $0,0137$ em relação ao *KNN*. Para o treinamento da rede foram usados os parâmetros: i) número de neurônios η (variando entre $1:40$); ii) taxa de aprendizagem α (variando entre $10^{-7}:10^{+1}$); iii) duas camadas escondidas; e iv) arquitetura totalmente conectada. Os melhores resultados de classificação foram obtidos para $\eta = 18$ e $\alpha = 10^{-4}$ obtendo 17% de itens classificados entre as dez primeiras posições da lista, e 80% entre as 50 primeiras posições, o que representa uma melhoria de 30% em relação a técnica *KNN*. O valor da métrica $MRR = 0,089$ apresentou uma taxa duas vezes mais elevada que a do *KNN*, esse aumento de precisão indica que o poder de generalizar da rede neural para solucionar problemas não lineares foi mais eficiente que a capacidade de generalização das técnicas anteriores.

O sistema baseado em CART como classificador, que tem como característica tratar dados categóricos, apresentou um resultado superior ao da rede neural. O treinamento usou os parâmetros: i) valor mínimo de divisão $\gamma = [0:30]$; ii) tamanho máximo da árvore final $\delta = [0:10000]$; iii) valor mínimo de variação para realizar uma divisão $cp = [10^{-7}:10^{+1}]$; iv) função de divisão (ξ) como índice de Gini ou ganho de informação. O melhor resultado foi para $gamma = 0$, $\delta = 30$, $cp = 10^{-3}$ e $\xi =$ Ganho de informação.

Os resultados desse sistema foram aproximadamente duas vezes melhores que os da rede neural. Isso indica uma tendência de bons resultados para técnicas que lidem com dados categóricos por natureza. Essa melhoria indicou um aumento de 26% na métrica MRR que representa um aumento da precisão do sistema, além disso posicionou 13% dos itens procurados na primeira posição e 26% nas primeiras 50 posições.

O sistema baseado em CART como regressor, teve seu melhor valor com os parâmetros $gamma = 2$, $\delta = 20$, $cp = 10^{-5}$ e $\xi =$ Ganho de informação. A recomendação que usou valores contínuos apresentou um resultado superior ao $CART_C$ nas métricas $S@1$ e $S@5$ e um resultado inferior para $S@10$ e $S@50$, e a precisão geral (MRR) do $CART_R$ foi levemente superior.

O sistema baseado no classificador Naive Bayes obteve resultados muito próximos ao do regressor CART. O treinamento ocorreu modificando o atributo *correção de Laplace*

com valores entre $[0 : 100]$. O melhor resultado ocorreu para o valor zero obtendo 34% dos itens recomendados entre as dez primeiras posições e 63% entre as 50 primeiras posições. Em contrapartida, o valor de MRR não sofreu grande variação.

O sistema baseado em regressor binomial apresentou melhoria em relação ao Naive Bayes e à rede neural (técnicas que apresentaram resultados próximos). O treinamento dessa técnica ocorre por máxima verossimilhança de um modelo generalizado linear aproximado por uma distribuição binomial. Os resultados para $S@5$ e $S@50$ foram superiores que das técnicas anteriores e o valor da métrica MRR melhorou em aproximadamente 19% em relação a técnica Naive Bayes. Isto indica que aproximar a variável dependente por uma distribuição binomial e estimar seus parâmetros por verossimilhança é uma ideia potencialmente interessante para tratar este problema.

A rede neural como regressor, que utiliza o peso da rede neural como saída, foi treinada de forma análoga à rede neural usada como classificador. O melhor resultado foi obtido para os valores de $\eta = 10$ e $\alpha = 10^{-2}$ recomendando 26% dos itens corretos entre as dez primeiras posições da lista. A precisão do sistema (MRR) melhorou 13% em relação ao regressor binomial. Esses resultados indicam que usar um regressor ao invés de um classificador apresenta um resultado melhor para esse tipo de problema, quando solucionado com redes neurais.

O sistema que usou o algoritmo MARS como regressor apresentou um resultado superior à rede neural (usada como regressor) em $12,5\%$ na métrica $S@1$, três vezes mais atividades recomendadas entre as 50 primeiras e um aumento de precisão geral (MRR) de 8%. Esse resultado mostra que as curvas criadas pelas diversas funções conectadas do MARS obtiveram uma generalização melhor que da rede neural. O treinamento dos parâmetros foi por verossimilhança.

O regressor SVM apresentou resultados duas vezes melhores que o algoritmo MARS para a medida S@10, pois em 49% das recomendações o item correto estava entre as dez primeiras posições da lista de recomendações. O valor de MRR também foi superior (42%). O treinamento foi feito por otimização de margem com os valores de $c = [10^{-7} : 10^{2}]$, $\epsilon = [10^{-7} : 10^{2}]$, valores de tolerância $\beta = [10^{-7} : 10^{2}]$, funções de *kernel*: i) linear; ii) sigmoide; iii) polinomial; e iv) radial, os parâmetros do *kernel* polinomial são: i) $p = [1 : 10]$ que é a potência da função. Os melhores valores encontrados foram para $c = 1$, $\epsilon = 1$, $\beta = 10^{-4}$, *kernel* polinomial com $p = 2$. Esse resultado é um indício que o problema não é linearmente separável, pois foi usada uma função de *kernel* polinomial para mapear

o problema em alta dimensão e projetá-lo novamente para uma dimensão mais baixa. Os autores acreditam que esta característica foi responsável pelo bom desempenho desse regressor.

Dentre os sistemas propostos pela literatura, o sistema baseado em entrada, saída e frequência (FES) (WANG et al., 2008) é o que apresenta os melhores resultados. Nos experimentos realizados, este sistema identificou o item correto entre as dez primeiras posições da lista de recomendação em 37% dos casos, e obteve um valor de $MRR = 0,196$.

O sistema baseado no algoritmo SVM para classificação foi o único classificador que superou os resultados dos regressores. Seu treinamento foi análogo ao SVM para regressão. Sua melhor execução foi para os valores $c = 10^{-1}$, $p = 10^{-4}$ e *kernel* linear. Esta execução, para a métrica $S@1$ foi 64% melhor que a da técnica FES e o valor da precisão geral (MRR) aumentou 24%. Este resultado indica que a solução utilizando *kernel* para mapeamento em alta dimensão é uma proposta eficiente no caso de classificadores.

O sistema SVM composto, que executa sobre os resultados dos outros sistemas de recomendação, apresentou um desempenho superior ao SVM para classificação. Seu treinamento foi análogo ao do SVM_C e seu melhor desempenho foi para os parâmetros $c = 10^{-2}$, $p = 1$ e *kernel polinomial*. Houve uma melhoria de 3% na métrica $S@1$ e 28% na métrica MRR, essa melhoria é em virtude do uso do resultado de outros classificadores em conjunto com a redução de esparsidade do conjunto de dados.

O sistema utilizando *Rotation Forest* apresentou o segundo melhor resultado, seu treinamento utilizou os parâmetros: i) valor mínimo de divisão $\gamma = [0 : 30]$; ii) tamanho máximo da árvore final $\delta = [0 : 10000]$; iii) valor mínimo de variação para realizar uma divisão $cp = [10^{-7} : 10^{+1}]$; iv) função de divisão (ξ) como índice de Gini e ganho de informação; v) $K = [1 : 10]$ como número de partições; vi) $L = [1 : 10]$ como o número de classificadores; e vii) valores de corte $0,25; 0,5; 0,75$. Essa melhoria foi em virtude de usar em conjunto uma técnica de classificação do tipo *ensemble* e três limiares de corte, os quais foram estabelecidos para converter os valores numéricos (da média dos L classificadores) em valores binários.

A técnica FESO, apresentou um resultado superior às demais. Este considera o uso de frequência, entrada e saída e informações semânticas sobre as atividades. Em comparação com as demais técnicas seu resultado foi superior para todas as métricas calculadas, exceto $S@50$ para algumas técnicas. Em relação à técnica FES, seu resultado foi superior. Em particular, parte dessa melhora é justificada pelos casos em que a atividade

correta teria frequência zero no conjunto de treinamento, pois ela permite recomendar baseada na ontologia (usando as atividades que contenham a ontologia do novo *workflow*). Além disso, para o caso em que há empate entre duas atividades com o critério de entrada e saída e a frequência a técnica proposta apresenta um fator a mais para ser utilizado como desempate.

Algumas tendências observadas com esses resultados foram que aumentar a informação sobre dados na recomendação melhora o seu desempenho, como o resultado dos experimentos: 2, 12 e 14 mostram. Uma segunda tendência é que o classificador SVM foi o único que obteve um melhor resultado que os regressores, indicando que soluções por maximização de espaço entre dados em alta dimensão podem ser uma área de estudo promissora. Uma terceira tendência é o uso de classificadores compostos e *ensembles*, os quais apresentaram resultados promissores. No caso do *ensemble* há um indício que técnicas desse tipo, que usem limiares para converter os valores da média dos resultados do conjunto L em valores binários, têm resultados promissores na recomendação de atividades.

6 Conclusões e trabalhos futuros

''Você tem a força de um homem, mas a mentalidade de uma garotinha.''

(Ragnar Lothbrok)

Este trabalho desenvolveu uma técnica híbrida para recomendar atividades em *workflows* científicos, que usa compatibilidade sintática, frequência e ontologias de domínio para recomendar atividades, denominada FESO. Além disso, também modelou o problema de recomendação como um problema de regressão e classificação em inteligência artificial.

A principal ideia do projeto foi acrescentar informações semânticas estruturadas para o sistema de recomendação. Conforme foi apresentado no capítulo de resultados (capítulo 5), esta estratégia atingiu melhores resultados do que as outras técnicas implementadas, sendo que a medida MRR aumentou 70% em relação as outras estratégias.

Para encontrar as técnicas da literatura correlata, foi realizada uma revisão sistemática (capítulo 3). Nessa revisão foram encontradas as técnicas, suas restrições, suas vantagens e as formas que foram validadas. O próximo passo foi implementá-las e compará-las com as soluções propostas neste mestrado, incluindo as soluções baseadas em classificadores e regressores.

Para realizar a comparação foi organizado um banco de dados relacional de *workflows* e suas atividades. Também foi necessário estabelecer uma metodologia para comparar diferentes técnicas de recomendação de atividades para um mesmo conjunto de dados com as mesmas métricas de validação $S@k$ e MRR (descritas na seção 4.4).

Ao comparar todas as técnicas, foram constatados determinados aspectos do conjunto de dados, como o fato das atividades não serem independentes; o problema não ser linearmente separável; e que técnicas de agrupamento não se mostraram adequadas para solucionar este problema. Com exceção do SVM, regressores apresentaram soluções mais precisas do que classificadores. Além disso, adicionar informação nos sistemas de recomendação melhorou a precisão destes. A seguir serão listadas as principais contribuições deste mestrado e potenciais trabalhos futuros.

6.1 Principais contribuições

Este trabalho teve como objetivo principal especificar uma técnica para recomendar atividades em *workflows* científicos. Este objetivo foi alcançado, pois a técnica FESO e as técnicas baseadas em classificadores/regressores apresentaram resultados superiores aos dos propostos pela literatura. Além deste objetivo primário foram obtidas as seguintes contribuições:

- Foi apresentada uma revisão sistemática sobre a área de recomendação de atividades em *workflows* científicos a qual poderá ser a base para trabalhos futuros.
- Foi construída uma base de dados relacional de *workflows* científicos com suas respectivas atividades. Esta base será disponibilizada na íntegra para uso de outros trabalhos.
- Foram implementadas diferentes técnicas da literatura correlata e foram comparados os resultados da recomendação dessas técnicas com os resultados da solução proposta.
- Até o momento esta pesquisa de mestrado colaborou com a publicação de dois artigos científicos (KHOURI; DIGIAMPIETRI, 2015; DIGIAMPIETRI et al., 2014).

6.2 Trabalhos futuros

No decorrer deste projeto foram identificadas algumas oportunidades de continuidade e evolução do mesmo, são elas:

1. Usar outros classificadores compostos na recomendação de atividades;
2. Criar novas estratégias de recomendação baseadas em redes sociais de pesquisadores ou seus grupos de pesquisa;
3. Obter informação sobre proveniência de *workflows* e adicioná-la aos sistemas de recomendação;
4. Usar atividades de outros SGWC e/ou de outras áreas de pesquisa além da bioinformática;
5. Estudar a relação entre a distribuição dos dados de entrada (atividade), sua esparsidade e a relação que ambas possuem com o aumento ou redução da precisão das recomendações;
6. Utilizar técnicas de redução de dimensionalidade para o conjunto de dados de entrada

7. Adaptar o classificador SVM para considerar ontologias durante a maximização da margem ótima.

Este trabalho organizou por meio de uma revisão sistemática o estado da arte da área de recomendação de atividades em *workflows* científicos. Criou uma técnica para recomendar atividades nos *workflows* científicos que permitiu aceitar a hipótese proposta na seção de objetivos 1.1. Acrescentar informações semânticas estruturadas no problema de recomendação de atividades acarretou uma melhoria nas métricas MRR e $S@k$ dos sistemas de recomendação que as utilizaram. Espera-se que este projeto sirva como semente para futuros estudos na área de recomendação de atividades para *workflows* científicos e auxilie cientistas na construção de experimentos computacionais.

Referências[1]

ACMDL, t. *ACM Digital Library*. 2014. Disponível em: <http://dl.acm.org/>. Citado na página 61.

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, v. 17, n. 6, p. 734--749, 2005. Citado na página 20.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. (VLDB '94), p. 487--499. Disponível em: <http://dl.acm.org/citation.cfm?id=645920.672836>. Citado 2 vezes nas páginas 27 e 70.

AGRESTI, A. *Foundations of Linear and Generalized Linear Models*. 2015. Citado na página 56.

ALMEIDA, M.; BAX, M. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos aplicações, métodos de avaliação e de construção. *Ciência da Informação*, scielo, v. 32, p. 7--20, 2003. Disponível em: <http://dx.doi.org/10.1590/S0100\-19652003000300002>. Citado 2 vezes nas páginas 24 e 25.

BHAGAT, J. et al. *BioCatalogue: a universal catalogue of web services for the life sciences*. 2014. Disponível em: <http://dx.doi.org/10.1093/nar/gkq394>. Citado na página 14.

BIOLCHINI, J. C. de A. et al. Scientific research ontology to support systematic review in software engineering. *Adv. Eng. Inform.*, Elsevier Science Publishers B. V., v. 21, n. 2, p. 133--151, 2007. Disponível em: <http://www.sciencedirect.com/science/article/pii-/S147403460600070X>. Citado na página 61.

BOMFIM, E. et al. Thoth: improving experiences reuses in the scientific environment through workflow management system. In: *Computer Supported Cooperative Work in Design, 2005. Proceedings of the Ninth International Conference*. [s.n.], 2005. v. 2, p. 1164--1170 Vol. 2. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails-.jsp?arnumber=1504261>. Citado 3 vezes nas páginas 63, 64 e 72.

BOTTOU, L. et al. Scaling learning algorithms toward ai. In: _____. *Large-Scale Kernel Machines*. MIT Press, 2007. p. 321--359. ISBN 9780262255790. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6279976>. Citado na página 36.

BREIMAN, L. et al. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984. Citado na página 30.

BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 2, n. 2, p. 121--167, jun. 1998. Disponível em: <http://dx.doi.org/10.1023/A:1009715923555>. Citado 2 vezes nas páginas 43 e 51.

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

CAO, B. et al. Graph-based workflow recommendation: on improving business process modeling. In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012. (CIKM '12), p. 1527--1531. Disponível em: <http://doi.acm.org/10.1145/2396761.2398466>. Citado 3 vezes nas páginas 64, 67 e 72.

CEREZO, N.; MONTAGNAT, J. Scientific Workflow Reuse Through Conceptual Workflows on the Virtual Imaging Platform. In: *Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science*. ACM, 2011. ({WORKS} '11), p. 1--10. Disponível em: <http://doi.acm.org/10.1145/2110497.2110499>. Citado 3 vezes nas páginas 63, 67 e 73.

CHANG, C.-C.; LIN, C.-J. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, ACM, New York, NY, USA, v. 2, n. 3, p. 27:1--27:27, may 2011. ISSN 2157-6904. Disponível em: <http://doi.acm.org/10.1145/1961189.1961199>. Citado na página 43.

COHEN-BOULAKIA, S. et al. Distilling structure in Taverna scientific workflows: a refactoring approach. *BMC bioinformatics*, v. 15 Suppl 1, p. S12, jan 2014. Disponível em: <http://www.pubmedcentral.nih.gov/articlerender-.fcgi?artid=4016501\&tool=pmcentrez\&rendertype=abstract>. Citado na página 15.

CONNOR, S. B. Wiley Encyclopedia of Statistics in Quality and Reliability. In: _____. [S.l.]: Wiley, 2007. cap. Perfect Sampling. Citado 2 vezes nas páginas 31 e 32.

CRISTIANINI, N.; SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. New York, NY, USA: Cambridge University Press, 2000. ISBN 0-521-78019-5. Citado na página 43.

DEELMAN, E. et al. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, v. 25, n. 5, p. 528--540, may 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X08000861>. Citado na página 13.

DEZA, M. M.; DEZA, E. *Encyclopedia of Distances*. 2ed. ed. [S.l.]: Springer Berlin Heidelberg, 2009. Citado 2 vezes nas páginas 22 e 69.

DIAMANTINI, C.; POTENA, D.; STORTI, E. Mining Usage Patterns from a Repository of Scientific Workflows. In: *Proceedings of the 27th Annual {ACM} Symposium on Applied Computing*. ACM, 2012. (SAC '12), p. 152--157. Disponível em: <http://doi.acm.org/10.1145/2245276.2245307>. Citado 4 vezes nas páginas 64, 67, 68 e 72.

DIGIAMPIETRI, L. et al. A framework for automatic composition of scientific experiments: Achievements, lessons learned and challenges. In: *CSBC 2014 - BreSci*. Brasília - DF: [s.n.], 2014. Citado na página 92.

DIGIAMPIETRI, L. A. *Gerenciamento de workflows cientificos em bioinformatica*. Tese (Doutorado) --- Unicamp, 2007. Citado na página 13.

DIRECT, t. S. *Science Direct*. 2014. Disponível em: <http://www.sciencedirect.com/>. Citado na página 61.

EZEIFE, C. I. Plwap sequential mining: open source code. In: *In Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*. [S.l.: s.n.], 2005. p. 26. Citado na página 66.

FLETCHER, R. *Practical methods of optimization*. Chichester, New York, Brisbane: John Wiley & Sons, 1987. ISBN 0-471-91547-5. Citado 2 vezes nas páginas 43 e 45.

GARIJO, D. et al. Common motifs in scientific workflows: An empirical analysis. *Future Generation Computer Systems*, v. 36, p. 338--351, jul 2014. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X13001970>. Citado 2 vezes nas páginas 16 e 18.

GARIJO, D.; CORCHO, O.; GIL, Y. Detecting Common Scientific Workflow Fragments Using Templates and Execution Provenance. In: *Proceedings of the Seventh International Conference on Knowledge Capture*. New York, NY, USA: ACM, 2013. (K-CAP '13), p. 33--40. Disponível em: <http://doi.acm.org/10.1145/2479832.2479848>. Citado 3 vezes nas páginas 64, 68 e 72.

GARIJO, D. et al. Workflow Reuse in Practice: A Study of Neuroimaging Pipeline Users. *Proceedings of the 2014 IEEE 10th International Conference on eScience*, p. 239--246, 2014. Disponível em: <http://www.isi.edu/~gil/papers/garijo-etal-escience14a.p>. Citado 3 vezes nas páginas 64, 69 e 73.

GIL, Y. et al. *Wings: Intelligent Workflow-Based Design of Computational Experiments*. 2015. Disponível em: <http://www.wings-workflows.org>. Citado na página 64.

HAAV, H.-M.; LUBI, T.-L. A survey of concept-based information retrieval tools on the web. In: *5th East-European Conference, ADBIS 2001*. [S.l.: s.n.], 2001. p. 11. Citado na página 25.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. [S.l.]: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791. Citado 8 vezes nas páginas 27, 28, 30, 31, 32, 35, 51 e 79.

HARVEY, M.; RUTHVEN, I.; CARMAN, M. Ranking Social Bookmarks Using Topic Models. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, p. 1401--1404, 2010. Disponível em: <http://doi.acm.org/10-.1145/1871437.1871632>. Citado na página 80.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. [S.l.]: Springer, 2003. Hardcover. Citado na página 56.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 3nd. ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007. Citado 14 vezes nas páginas 36, 37, 38, 40, 43, 44, 45, 46, 48, 50, 51, 52, 53 e 54.

HEIJST, G. van; SCHREIBER, A.; WIELINGA", B. Using explicit ontologies in {KBS} development. *International Journal of Human-Computer Studies*, v. 46, n. 2–3, p. 183 -- 292, 1997. Disponível em: <http://www.sciencedirect.com/science/article/pii-/S1071581996900907>. Citado na página 25.

HUANG, X.; LIN, J.; DEMNER-FUSHMAN, D. Evaluation of PICO as a knowledge representation for clinical questions. In: *AMIA Annu Symp Proc.* American Medical Informatics Association, 2006. p. 359--363. Disponível em: <http://www.ncbi.nlm.nih.gov-/pmc/articles/PMC1839740/>. Citado na página 62.

IEEE, t. *IEEE Xplore Digital Library*. 2014. Disponível em: <http://ieeexplore.ieee.org-/Xplore/home.jsp>. Citado na página 61.

JASPER, R.; USCHOLD, M. A framework for understanding and classifying ontology applications. In: *Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW 99*. [S.l.]: V. Richard Benjamins, 1999. p. 12. Citado na página 25.

KEPLER, T. D. *Kepler your science. Enabled*. 2014. Disponível em: <https://kepler-project.org/>. Citado 2 vezes nas páginas 16 e 17.

KHOURI, A. L.; DIGIAMPIETRI, L. A. A systematic review about activities recommendation in workflows. In: *12ª Conferência Internacional sobre Sistemas de Informação e Gestão de Tecnologia (CONTECSI)*. [S.l.: s.n.], 2015. p. 14. Citado na página 92.

KOOP, D. Viscomplete: Automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Educational Activities Department, v. 14, n. 6, p. 1691--1698, nov 2008. Disponível em: <http://dx.doi-.org/10.1109/TVCG.2008.174>. Citado 6 vezes nas páginas 63, 64, 65, 67, 70 e 72.

LANTZ, B. *Machine Learning with R*. 1nd. ed. Birmingham: Packt Publishing, 2013. Citado 9 vezes nas páginas 28, 29, 32, 33, 34, 36, 37, 42 e 50.

LENG, Y.; EL-GAYYAR, M.; CREMERS, A. B. Semantics Enhanced Composition Planner for Distributed Resources. In: *2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science*. IEEE, 2010. p. 61--65. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper-.htm?arnumber=5573302>. Citado 3 vezes nas páginas 63, 66 e 72.

LIM, C. et al. Prospective and Retrospective Provenance Collection in Scientific Workflow Environments. In: *2010 IEEE International Conference on Services Computing*. IEEE, 2010. (SCC '10), p. 449--456. Disponível em: <http://ieeexplore-.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557202>. Citado na página 19.

LIMA, C. A. de M. *Comitê de Máquinas:Uma Abordagem Unificada Empregando Máquinas de Vetores-Suporte*. Tese (Thesys) --- UNICAMP, 2004. Citado 4 vezes nas páginas 43, 44, 49 e 52.

LIN, C. et al. A Task Abstraction and Mapping Approach to the Shimming Problem in Scientific Workflows. In: *2009 IEEE International Conference on Services Computing*. IEEE Computer Society, 2009. (SCC '09), p. 284--291. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5283946>. Citado na página 18.

LUDASCHER, B. et al. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, John Wiley e Sons, Ltd., v. 18, n. 10, p. 1039--1065, 2006. Disponível em: <http://dx.doi.org/10.1002/cpe.994>. Citado na página 17.

MARTINEZ, W. L. *Exploratory Data Analysis with MATLAB (Computer Science and Data Analysis)*. [S.l.]: Chapman & Hall/CRC, 2004. Hardcover. ISBN 1584883669. Citado na página 58.

MCGUINNESS, D. L.; HARMELEN, F. v. *OWL Web Ontology Language Overview*. 2015. Electronic. Disponível em: <http://www.w3.org/TR/owl-features/>. Citado na página 75.

MCPHILLIPS, T. et al. Scientific workflow design for mere mortals. *Future Generation Computer Systems*, v. 25, n. 5, p. 541--551, may 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X08000873>. Citado na página 16.

MEDEIROS, C. et al. WOODSS and the Web: Annotating and Reusing Scientific Workflows. *ACM SIGMOD Record*, v. 34, n. 3, p. 18--23, 2005. Citado 2 vezes nas páginas 13 e 19.

MIZOGUCHI, R.; WELKENHUYSEN, J. V.; IKEDA, M. Task ontology for reuse of problem solving knowledge. In: *Towards Very Large Knowledge Bases*. [S.l.]: IOS Press, 1995. p. 46--57. Citado na página 25.

MOHAN, A.; EBRAHIMI, M.; LU, S. 2015 ieee international conference on services computing a folksonomy-based social recommendation system for scientific workflow reuse. 2015. Citado 3 vezes nas páginas 64, 70 e 73.

OLABARRIAGA, S. et al. Scientific Workflow Management -- For Whom? *2014 IEEE 10th International Conference on e-Science*, Ieee, p. 298--305, oct 2014. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6972277>. Citado na página 13.

OLIVEIRA, F. T. d. et al. Improving workflow design by mining reusable tasks. *Journal of the Brazilian Computer Society*, Journal of the Brazilian Computer Society, v. 21, n. 1, p. 16, 2015. Disponível em: <http://www.journal-bcs.com/content/21/1/16>. Citado 3 vezes nas páginas 64, 70 e 72.

OLIVEIRA, F. T. de. *UM SISTEMA DE RECOMENDAÇÃO PARA COMPOSIÇÃO DE WORKFLOWS*. Dissertação (Mestrado) --- UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2010. Citado 3 vezes nas páginas 63, 66 e 73.

OLIVEIRA, F. T. de et al. Using provenance to improve workflow design. In: FREIRE, J.; KOOP, D.; MOREAU, L. (Ed.). *Provenance and Annotation of Data and Processes*. Springer Berlin Heidelberg, 2008, (Lecture Notes in Computer Science, v. 5272). p. 136--143. Disponível em: <http://dx.doi.org/10.1007/978-3-540-89965-5_15>. Citado 3 vezes nas páginas 63, 65 e 72.

OLSSON, U. *Generalized Linear Models An Applied Approach*. 2002. Citado 2 vezes nas páginas 56 e 57.

PAIVA, F. A. P. de; COSTA, J. A. F.; SILVA, C. R. M. A Hierarchical Architecture for Ontology-Based Recommender Systems. In: *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*. IEEE, 2013. p. 362--367. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper-.htm?arnumber=6855876>. Citado na página 20.

PEVZNER, P.; COMPEAU, P. *Comparing Genes, Proteins, and Genomes (Bioinformatics III)*. 2015. Online course. Disponível em: <https://www.coursera.org-/course/comparinggenomes>. Citado na página 75.

PEVZNER, P.; COMPEAU, P. *Deciphering Molecular Evolution (Bioinformatics IV)*. 2015. Online course. Disponível em: <https://www.coursera.org/course-/molecularevolution>. Citado na página 75.

PEVZNER, P.; COMPEAU, P. *Finding Hidden Messages in DNA (Bioinformatics I)*. 2015. Online course. Disponível em: <https://www.coursera.org/course/hiddenmessages>. Citado na página 75.

PEVZNER, P.; COMPEAU, P. *Genome Sequencing (Bioinformatics II)*. 2015. Online course. Disponível em: <https://www.coursera.org/course/assembly>. Citado na página 75.

PROGRAMMABLEWEB, C. T. 2014. Disponível em: <https://www.programmableweb-.com/>. Citado 2 vezes nas páginas 64 e 69.

PROTEGE, C. T. 2014. Disponível em: <http://protege.stanford.edu/>. Citado na página 75.

REX, D. E.; TOGA, A. W. The loni pipeline processing environment. v. 19, p. 1033--1048, 2003. Citado 2 vezes nas páginas 64 e 70.

RICHARDSON, L. *Beautiful Soup*. 2015. Disponível em: <http://www.crummy.com-/software/BeautifulSoup/>. Citado na página 76.

RODRÍGUEZ, J. J.; KUNCHEVA, L. I.; ALONSO, C. J. Rotation forest: A New classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 28, n. 10, p. 1619--1630, 2006. Citado 2 vezes nas páginas 57 e 59.

ROURE, C. G. *myExperiment*. 2015. Disponível em: <http://www.myexperiment.org/>. Citado 9 vezes nas páginas 14, 63, 64, 67, 68, 69, 70, 72 e 76.

SANCHEZ, D. et al. Ontology-based semantic similarity: A new feature-based approach. *Expert Systems with Applications*, v. 39, n. 9, p. 7718--7728, jul 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417412000954>. Citado 2 vezes nas páginas 24 e 26.

SCRIVANO, G.; NIKSIC, H. *GNU Wget Introduction to GNU Wget*. 2015. Disponível em: <http://www.gnu.org/software/wget/>. Citado na página 76.

SETUBAL, J.; MEIDANIS, J. *Introduction to computational molecular biology*. [S.l.]: PWS Publishing, 1997. Citado na página 75.

SHAO, Q.; KINSY, M.; CHEN, Y. Storing and Discovering Critical Workflows from Log in Scientific Exploration. In: *2007 IEEE Congress on Services (Services 2007)*. IEEE, 2007. p. 209--212. Disponível em: <http://ieeexplore.ieee.org/lpdocs-/epic03/wrapper.htm?arnumber=4278799>. Citado 3 vezes nas páginas 63, 65 e 72.

SHAO, Q.; SUN, P.; CHEN, Y. Efficiently discovering critical workflows in scientific explorations. *Future Generation Computer Systems*, v. 25, n. 5, p. 577--585, may 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X08000897>. Citado 3 vezes nas páginas 63, 65 e 72.

SHAWE-TAYLOR, J.; CRISTIANINI, N. *Kernel Methods for Pattern Analysis*. 1. ed. [S.l.]: Cambridge University Press, 2004. Citado na página 51.

SOOMRO, K.; MUNIR, K.; MCCLATCHEY, R. Incorporating semantics in pattern-based scientific workflow recommender systems. 2015. Citado 3 vezes nas páginas 64, 70 e 73.

STEWART, J. *Cálculo*. 7. ed. [S.l.]: Cengage, 2013. v. 2. Citado 2 vezes nas páginas 43 e 44.

TAN, W. et al. Providing Map and GPS Assistance to Service Composition in Bioinformatics. In: *2011 IEEE International Conference on Services Computing*. IEEE, 2011. p. 632--639. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper-.htm?arnumber=6009316>. Citado 3 vezes nas páginas 63, 67 e 72.

TELEA, A.; WIJK, J. J. van. Vission: An object oriented dataflow system for simulation and visualization. In: *PROCEEDINGS OF IEEE VISSYM*. [s.n.], 1999. p. 95--104. Disponível em: <http://link.springer.com/chapter/10.1007%2F978-3-7091-6803-5_21>. Citado 3 vezes nas páginas 63, 64 e 73.

UMAMAHESWARI, B.; PATIL, P. Personalized ontology model - a survey. In: *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*. [S.l.: s.n.], 2012. p. 348--353. Citado na página 24.

USCHOLD, M.; GRUNINGER, M. Ontologies: Principles, methods and applications. *KNOWLEDGE ENGINEERING REVIEW*, v. 11, p. 93--136, 1996. Citado na página 25.

USCHOLD, M.; KING, M. Towards a methodology for building ontologies. In: *In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*. [S.l.: s.n.], 1995. Citado na página 74.

VAPNIK, V. N. *Statistical learning theory*. 1. ed. [S.l.]: Wiley, 1998. Citado na página 52.

VISTRAILS, t. *Vistrails*. 2015. Disponível em: <http://www.vistrails.org/index.php-/Main_Page>. Citado na página 65.

WANG, F. et al. A Survey on Scientific Workflow Techniques for Escience in Astronomy. In: *2010 International Forum on Information Technology and Applications*. IEEE, 2010. v. 1, p. 417--420. Disponível em: <http://ieeexplore.ieee.org/lpdocs-/epic03/wrapper.htm?arnumber=5634997>. Citado 2 vezes nas páginas 13 e 14.

WANG, J. et al. Vinca4science: A personal workflow system for e-science. In: *Internet Computing in Science and Engineering, 2008. ICICSE '08. International Conference on*. [s.n.], 2008. p. 444--451. Disponível em: <http://ieeexplore.ieee.org-/xpl/articleDetails.jsp?arnumber=4548305>. Citado 4 vezes nas páginas 63, 65, 72 e 89.

WANG, Y.; CAO, J.; LI, M. Change Sequence Mining in Context-Aware Scientific Workflow. In: *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*. IEEE, 2009. p. 635--640. Disponível em: <http://ieeexplore.ieee.org-/lpdocs/epic03/wrapper.htm?arnumber=5207868>. Citado 3 vezes nas páginas 63, 66 e 72.

WILLIAMS, G. J. *Data Mining with Rattle and R: The art of excavating data for knowledge discovery*. 1. ed. [S.l.]: Springer, 2011. Citado na página 30.

YAO, J. et al. Reputationnet: A reputation engine to enhance servicemap by recommending trusted services. In: *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. [s.n.], 2012. p. 454--461. Disponível em: <http://ieeexplore-.ieee.org/xpl/articleDetails.jsp?arnumber=6274177>. Citado 3 vezes nas páginas 64, 68 e 72.

YEO, P.; ABIDI, S. S. R. Dataflow Oriented Similarity Matching for Scientific Workflows. In: *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*. IEEE, 2013. p. 2091--2100. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6651115>. Citado 3 vezes nas páginas 64, 68 e 72.

ZHANG, J. et al. A Community-Driven Workflow Recommendations and Reuse Infrastructure. In: *2014 IEEE 8th International Symposium on Service Oriented System Engineering*. IEEE, 2014. p. 162--172. Disponível em: <http://ieeexplore.ieee.org-/lpdocs/epic03/wrapper.htm?arnumber=6830902>. Citado 3 vezes nas páginas 64, 69 e 73.

ZHANG, J.; LIU, Q.; XU, K. *FlowRecommender: A Workflow Recommendation Technique for Process Provenance*. 2009. Citado 5 vezes nas páginas 63, 64, 65, 67 e 72.

ZHANG, J. et al. Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse. In: *2011 IEEE International Conference on Services Computing*. IEEE, 2011. p. 48--55. Disponível em: <http://ieeexplore.ieee.org-/lpdocs/epic03/wrapper.htm?arnumber=6009243>. Citado 3 vezes nas páginas 63, 67 e 73.

ZHONG, L.; JINSHA, Y.; HONG, Y. Analysis distances for similarity estimation by Fuzzy C-Mean algorithm. In: *2009 International Conference on Machine Learning and Cybernetics*. IEEE, 2009. v. 1, p. 582--587. Disponível em: <http://ieeexplore-.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5212575>. Citado na página 21.